XSS Attack Detection and Prevention System Based on Instruction Set Randomization

Qijin Wang1,2, Jun Huang3 and Xiaoxia Qi1*
1college of Information Engineering, Anhui Xinhua University;
2Institute of Intelligent Machines,Chinese Academy of Sciences;
3Information Technology Security Evaluation Center, University of Science and Technology of China;
*corresponding author’s e-mail: lushitou710@163.com

Abstract. As one of the most popular security vulnerabilities in Web applications, XSS has been widely researched and applied. This paper presents a XSS detection and prevention system aimed at solving the problem of website being attacked by increasingly sophisticated and severe XSS. The method proposed in this paper encodes the HTML/JavaScript keyword in the Web application by randomization and distinguishes the malicious attacks. The results show that this method not only can effectively detect and defend reflective and storage XSS attacks, but also it has better system response.

1. Introduction

With the rise of Web2.0, Internet has already become the main medium and channel of communication and commercial activities, and there has been an increase in the number of Web applications. But at the same time, Web application security vulnerabilities have been discovered and exploited at an alarming rate. In 2013, among the ten major Web security vulnerabilities from OWASP[1], XSS sits in third place. As a kind of Web security vulnerabilities, XSS is widely considered by researchers and industry insiders as the most prevalent and popular security breach in Web applications. Therefore, XSS has high research significance and research value.

Over the years various tools have been developed for mitigating common Web application vulnerabilities. However, the Web application that developed with function-oriented and simple ideas can easily lead to generate XSS and other security vulnerabilities. At present, there are many detection and defense technologies of Web application, while they each has shortcomings, such as poor practicability, inflexible deployment, greater performance loss and high false alarm rate and missed rate. A.Guha[2] et al. proposed a static control flow analysis method for JavaScript programs, building user expectation behavior model from server perspective, by monitoring the URL request to detect potential malicious attacks. This method mainly analyzes the request issued by XMLHttpRequest, it doesn't working for the request not issued by the XMLHttpRequest object. O.Hallaraker[3] et al. put forward a client detection mechanism for malicious JavaScript programs, monitored the execution of JavaScript by the audit system embedded in the browser, and compared with known malicious attack to analyze potential attacks, perform anomaly or misuse detection. The method relies on the prior extraction of fingerprint information, which is invalid for unknown attacks. Noncespaces[4] cooperative the server with the client, distinguished trusted and entrusted webpage by randomization XML namespaces, the server divides the webpage into the trusted class, randomized the documents, assign a
policy to each trusted class; The client only displays webpage that matched the server allocation policy. Noncespaces is valid for some types of XSS attacks, but not for storage XSS.

In response to the problem of existing XSS detection technology, this paper proposed a new detection method based on instruction set randomization. The method is flexible and easy to deploy, it is applied to the server and need not to modify the client, and it can not only detect the already discovered XSS attacks, but also can detect the evolving undetected new XSS attacks. This method can effectively prevent 0day vulnerability in XSS field, at the same time, it can effectively detect and prevent webpage-Trojan and webpage tampering.

2. Methodology
Due to the inadequacy of filter that Web application to user input, The server is the root. The system presented in this paper deploys defense system on the proxy server, intercepts all webpage returned to user browser, verifies the legality contents, only the trusted webpage will be transmitted to the browser. This prevent the occurrence of XSS at the root. The system is mainly composed of nuclear XSSD proxy, the verifier, rule engine and knowledge base, as shown in figure 1. When the user submits the HTTP request to the Web server, Web server response to it, XSSD proxy server intercepted server response message, forwarded it to the verifier for XSS detecting; The verifier scanned response message, validated by rules provided by rule engine and knowledge base. If validation passes, The verifier will notify the XSSD response message will be forwarded to the user proxy.

![System architecture diagram](image)

**Figure 1. System architecture diagram**

2.1. XSSD proxy
XSSD proxy is an HTTP interceptor, intercepting all response messages sent from the server. Whenever the server sends HTTP response, XSSD will be triggered automatically. Before the response message is sent to the user, XSSD intercept and submit the response message to the verifier. XSSD proxies usually have the filter technology, module technology and aspect-oriented programming technology.

2.2. The verifier
The verifier is the core module of the system. The user requests were verified by the verifier to ensure the security before forwarding to the client. The verifier uses the instruction set randomization and appends a string of randomly generated numbers at the end of the character to encode each HTML/JavaScript keyword, to form a new instruction set that the attacker cannot predict in advance.

2.3. Rule engine
Rule engine is the interface between the verifier and knowledge base. According to the configuration
of knowledge base, the rule engine provides the checking basis that similar to the white list for verifier. The basic idea of rule engine is to explain the business and application logic of the application, and perform statement parsing of complex if-then. The rule engine defines a severity level for the if-then statement rule, that is low, middle and high. When the rule severity level is low, the HTTP response is immediately returned to the client browser. When the rule severity level is middle, the rule engine records the violation to a violation table. When the rule severity level is high, the rule engine represents a possible attack, the server stops sending a response message, and throws an exception. Because of all the violations encountered during processing the response message in the offending table, the rule engine takes different actions depending on the violation.

2.4. The knowledge base
The knowledge base is a configuration file that provides the user with keyword information to match. Keyword types typically have HTML tags, HTML events, JavaScript keywords, etc. The knowledge base decomposes the verifier response message into array to match, two main operations in the matching process, the encoded key set is decoded to the normal keyword set, then transcodes the injected keywords. When the matching process completed, the XSSD proxy forwards the response message to the client browser.

3. System Implementation
To the server insufficient input filter of the reflective and persistent XSS attacks, pages that contain malicious scripts are returned to the user's browser, the browser is only responsible for the implementation of the webpage and cannot identify the script content, therefore it cannot identify whether the site contains malicious code. In the specific implementation, there are usually two ways to solve the problem, one way is to increase the Web server module, the other is to add the proxy server between the browser and the Web server. Due to the first way requirements of the deep understanding of Apache, IIS and other mainstream Web servers, the system is difficult to achieve, so the system adopt the second way, shown in Figure 2.

Figure 2. system module
Proxy server listens to all traffic flowing through Web Server, for the HTTP requests and user input coming from Client browser, Proxy intact forwards to Web Server; For the page that the Web Server returned to the Client, it will be forwarded to the user after Proxy processing, decoding and verification.

The following is a HTML document with some HTML/JavaScript code,

1 <html>
2<head>
3<title>XSS Test</title>
4</head>
5<body>
6<form action="XSS.php" method="POST">
7 please enter your name:<br>
8<input type="text" name="name" value=""></input>
9<script>alert("This is a hacker attack")</script>
10<input type="submit" value="submit"></input>
11</form>
12</body>
13</html>
When the PHP program processes code, it uses the \$_REQUEST[\text{name}] get the name variable coming from user input, but because the page added the HTML/JavaScript script, so when click the 【Submit】 button, the browser will not output the content of name variable, but pops up a dialog box, shows "This is a hacker attack ".

The instruction set randomization was used by adding a string of random numbers at the end of each HTML/JavaScript script. The attacker cannot predict the random numbers, so it is a good way to distinguish between legitimate scripts and malicious attacks. The following is the server coded file after randomization,

1<html>
2<head>
3<title>Test Result</title>
4</head>
5<body>
6<?php
7 echo \$_REQUEST[name];
8 ?>
9</body>
10</html>

Once the user submits the form, the following is the page file generated by the PHP analysis,

1<html>
2<head>
3<title>Test Result</title>
4</head>
5<body>
6<script>alert("This is a hacker attack ")</script>
7</body>
8</html>

The page file is decoded before returning to the client, the client input the <script>alert and no random code, so the page exists XSS injection. By separating the decoding and verification process from the Web server to the Proxy, the proposed system achieves better flexibility, simplicity and security. With the popularity of cloud computing, a proxy can proxy more back-end servers. The main task of proxy is to decode and verify the HTML/JavaScript script in the page, and then delivers the standard page to the client browser. At the same time, the proxy can hide related information of Web server to protect the use of random code.

4. Experimental Evaluation

Using the open source forum PhPBB for quantitative and qualitative evaluation which does not filter user input.

4.1. Quantitative Evaluation

Using the typical XSS script in XSS Cheat Sheet to attack the PhpBB site server which runs the server XSS defense system. For reflective XSS attack, the XSS script is embedded in the HTTP request or submitted in the HTML form, so it will be executed by the server and the page will be returned to the user. For storage XSS attack, malicious code is stored in the Web server, each user will be attacked by browsing the site. In either case, the proxy server checks the validity of the page content and blocks non PhPBB application scripts. Table 1 shows the success rate of the two attacks is 100%.

| XSS Vulnerability type     | Attack numbers | Real-time Detection | Detection Accuracy |
|----------------------------|----------------|---------------------|--------------------|
| Reflective XSS attack      | 84             | 84                  | 100%               |
| Storage XSS attack         | 84             | 84                  | 100%               |
4.2. Qualitative Evaluation

In addition to ensuring the detection accuracy, the performance of system security solutions is also a very important indicator. Usually, the system needs a trade-off between higher security and better application performance, because with the increasing in security defense, there will be a certain loss of system performance. Therefore, an excellent security system should ensure the accuracy and the good performance.

The proposed system uses page response and load time to measure the performance of the Web sample program, that is, from the browser sends the URL request to the browser analysis, display the page to the user. Normally, user does not input invalid characters, table 2 and figure 3 shows the page response and load time statistics and comparison.

| Test number | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| disable     | 687 | 628 | 685 | 646 | 591 | 773 | 639 | 655 | 883 | 712 |
| enable      | 1025| 1085| 1103| 1051| 1018| 1168| 1082| 1047| 1195| 1087|

Figure 3 shows that the response time of the proposed system is basically stable at about 650 milliseconds when the XSS detection is not running. After the XSS detection, the proposed system response time increased to about 1050 milliseconds. Through several detection, the average of two operation conditions differs by 396 milliseconds, it is acceptable.

Then we visit the Web site as a Web attacker and input illegal strings, table 3 shows the system response time statistics.

| Test number | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| disable     | 3018| 2759| 2411| 2925| 2113| 2740| 3210| 2216| 1880| 2646|
| enable      | 1102| 995 | 970 | 1211| 942 | 987 | 1285| 979 | 951 | 980 |

As can be seen from table 3, when the XSS detection is disable, there have longer system response time and larger response time volatility. When the XSS detection is enable, the system response time is basically stable at an average value of 1040ms. When the XSS detection is disable, user inputs invalid character strings, the returned page contains the JavaScript script code, so the browser will call the JavaScript engine to parse the code, this will inevitably increase the system response time. The system response time is related to the execution efficiency of the JavaScript engine, as well as to the construction of malicious JavaScript code in the page.
Conclusion
According to the Web application of XSS attacks, this paper proposes a XSS attack detection and defense method based on instruction set randomization. It encodes the HTML/JavaScript keyword in the Web application by randomization and distinguishes the normal script from the Web application and the malicious script inputted by the user to detect potential XSS attacks. Results of the experiment using an open source sample library show that the method has better detection efficiency and higher practicability.

Acknowledgments
This work was financially supported by the project supported by the key natural science research projects of universities and colleges in anhui province (KJ2016A311,KJ2018A0589).

References
[1] Wichers D. 2013. OWASP Top-10 2013 [Verlag-Springer].
[2] Guha, A., Saftoiu, C., & Krishnamurthi, S. 2015. The essence of javascript. European Conference on Object-oriented Programming [German-Springer] pp 126-150.
[3] Hallaraker O, Vigna G. 2005. IEEE International Conference on Engineering of Complex Computer Systems. IEEE Computer Society, pp 85-94.
[4] Gundy M V, Chen H. 2011. Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks. Network and Distributed System Security Symposium, NDSS 2009, San Diego [USA-ACM] pp 77–84.
[5] Jun Huang. 2014. Research on Xss Detection System Base on ISR[D].University of Science and Technology of China [China].
[6] Boyd S W, Keromytis A D. 2004. SQLrand: Preventing SQL Injection Attacks. Lecture Notes in Computer Science [German-LNCS] pp 292--302.
[7] Shar L K, Tan H B K. 2012. Auditing the XSS defence features implemented in web application programs. Iet Software [UK-IET] pp 377-390.
[8] Salas M I P, Martins E.2014. Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security. Electronic Notes in Theoretical Computer Science [Holland-Elsevier] pp 133-154.
[9] Gupta S, Gupta B B.2015. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. International Journal of Systems Assurance Engineering & Management [German-Springer] pp 1-19.
[10] Gupta B B, Gupta S, Gangwar S, et al.2015. Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and Its Defense. Journal of Information Privacy & Security [USA-JIPS] pp 118-136.