Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 78 (2016) 82 – 87

Procedia Computer Science

International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015, Nagpur, INDIA

Automated Discovery of JavaScript Code Injection Attacks in PHP Web Applications

Shashank Gupta[a] , B. B. Gupta[b*]

Department of Computer Engineering, National Institute of Technology Kurukshetra, 136119, India

Abstract

This paper discussed some of the performance issues in the existing defensive solutions of Java Script injection attacks (e.g. Cross-Site Scripting (XSS) attacks). Moreover, a high level of comparison for such existing solutions has been done based on some useful metrics. Based on the identified performance issues, this paper proposed an automated detection system, which scans the numerous possible locations of web sites for JavaScript injection vulnerabilities. Our detection system, firstly, scans the web site for discovering the injection locations. Secondly, it injects the malicious XSS attack vectors in such injection points. Lastly, it takes an input as the list of different XSS attacks exploited in the second step and scan for these attacks in the vulnerable web application. Detection capability of our automated system is evaluated on a real world PHP web application i.e. BlogIt and results obtained are very promising.

Peer-review under responsibility of organizing committee of the ICISP2015

*Keywords:*Cross-Site Scripting (XSS) attacks; JavaScript Injection Vulnerabilities; XSS Cheat Sheet;PHP

1. Introduction

In the contemporary era of World Wide Web (WWW), online facilities on the Internet are easily accessible by the modern web applications by every single human of different community. Moreover, boom of social networking sites, online shopping sites, Internet banking and all other widespread modern web applications, which provides the dynamic content to the users have increased the demand of user generated HTML content. This content can be the reason for various web vulnerabilities like Cross-Site Scripting (XSS) attacks [19], Information leakage etc. Normally 80% of modern web applications are vulnerable to these attacks. XSS attack is the utmost harmful and widespread security issue affecting web applications. The key motto of XSS attacks is to theft the sensitive resources of user data (like cookies, credit card numbers, passwords etc.) [2-4]. Using these credentials, an attacker can simply hijack the session of any web site or access the sensitive resources of victim's web application. Various malicious HTML links are crafted in a very appealing mode on the web pages of modern web applications so that

Peer-review under responsibility of organizing committee of the ICISP2015
doi:10.1016/j.procs.2016.02.014

*Corresponding Author.
E-mail address: gupta.brij@gmail.com (Tel. +91-1744-233488)

users curious to know about the link can simply click on it, become a victim of XSS attack and compromise its credentials with the attacker.
A successful XSS attack can result in stern security violations for not only the user but also for the web site [20]. If an application doesn't validate its input, a hacker can insert a malicious script and steal cookies, hijack a user's account, transfer private information, cause denial of service attack, manipulate the web content, and many more malicious activities. XSS has appeared as one of the utmost severe dangers on the WWW. XSS falls first in the list of vulnerabilities reported published in [1]. In this paper, we have proposed an automated detection system, which scans several possible locations of injection points for the XSS vulnerabilities. We have also evaluated the detection capability of our proposed system on real world PHP web application i.e. BlogIT [6].
The main contributions of our work are as follows:

- Recent existing defensive state-of-art techniques on JavaScript injection vulnerabilities have been discussed with the key goal to identify the weaknesses and research gaps in their solutions.
- An automated detection system is proposed with the aim to discover injection points in a web application and possible malicious JavaScript injection vulnerabilities.
- Implementation of our proposed system was done on a client-side JavaScript library with few modifications on the web browser.
- Evaluation of our detection system has performed on a tested version of real life PHP web application BlogIT.

Rest of our article is structured as follows: Section 2 highlights the existing work on JavaScript code injection vulnerabilities in the form of their strengths and performance issues. The proposed automated detection framework is illustrated in Section 3. Implementation and evaluation of our work is highlighted in Section 4. Finally, conclusion as well as further scope of our work is illustrated in Section 5.

2. Related Work

In this section, we have discussed various XSS attacks detection and mitigation techniques in order to identify the research gaps in such techniques. Table 1 highlights the strengths and performance issues encountered in the existing state-of-art techniques of defensive methodologies of XSS vulnerabilities.

In addition, Table 2 provides a comparison of same state-of-art techniques described in Table 1 based on nine different categories: Exploitation Location, Discovery Site, Scrutinizing Mechanism, Persistent, Non-Persistent, DOM-Based XSS attack detection, Client-side web browser alterations, Server-side modifications and JavaScript Code Amendments. The statistics shown in the Table 2 clearly highlights that most of these techniques does not provide the protection against persistent as well as DOM injection XSS vulnerabilities. In addition to this, these methodologies need large-possible amendments at browser and web application's server. Although, these existing defensive methodologies included the advanced techniques of sanitization in the source code of web applications, however that too suffer from high runtime overhead and non-acceptable rate of false negatives.

To address such issues, we designed an automated framework for the discovery of JavaScript code injection attacks on a tested suite of real world PHP web applications. This technique is implemented on a client-side web browser in JavaScript library and encounters acceptable rate of false negatives during its testing on PHP web application BlogIT. The next section illustrates the details of our proposed design in a precise manner.

3. Proposed Work

In this paper, we have designed an automated detection system, which scans the numerous possible locations of web sites for script injection vulnerabilities like XSS. We developed an automated framework that explores for the script injection vulnerabilities in the tested version of PHP web applications. This system examines the web sites with the goal of discovering exploitable XSS vulnerabilities. Fig. 1 illustrates the design of our proposed automated detection system. It consists of three key agents: HTML Crawler, XSS Attack Vector Injector and Script Locator. Fig. 2 also explains the sequence of steps of execution of our automated detection system.

3.1 HTML Crawler

HTML crawler is an agent which scans the web site for discovering the injection locations where Persistent XSS attacks can be exploited. This analyzing procedure is analogous to that of HTML parsers and web spiders, since it thoroughly recovers data from the web pages it approaches and it circulates over the web site following the hyper-links it discovers. However, this crawler varies from the usual crawler in two facets: It simply refers the hyper-links with endpoint to the scanned web site neglecting all exterior links and, the info extracted is web forms.

Table 1. Summary of Related Existing Techniques on XSS Attacks

State-of-art Techniques	Strengths	Performance Issues
Kirda et al. 2006 [9]	Noxes support such a mitigation practice of XSS attack that considerably diminishes the amount of connection alert prompts and providing the defense against XSS attacks	Also this tool suffers from low reliability and prohibits the inclusion of benign HTML.
Balzarotti et al. 2008 [10]	This technique uses a bottom-up approach and tries to recreate the source code used by the web application and then run this recreated code with the malicious input values in order to determine the defective sanitization routines.	Confirming the reliability of sanitization routines, for a script-free HTML is naturally a challenging problem due to the continuous updating nature of modern web browsers.
Sun et al. 2009 [11]	It is a client-side technique, employed in the form of Firefox plug-in, for discovering dissemination of the XSS attack vectors without any modifications to web browser architecture.	Incapable to detect persistent and DOM-based category of XSS attacks.
Johns et al. 2008 [12]	It has fine capability to discover XSS attacks via calculating the deviance between the HTTP request as well as its associated HTTP response.	Method of discovering persistent XSS attack suffers from few false positives and requires a more advanced training phase for the collection of more scripts.
Bisht et al. 2008 [13]	Its main strength is that it can evade illicit script data from being a part of the HTTP response web page.	While this technique attempt to sanitize unsafe output, it still influence web browser parsers for inferring the unsafe data as well as vulnerable to threats which utilizes browser parse twists.
Balzarotti et al. 2007 [14]	It is a vulnerability investigation technique, which identifies the unintended workflow of real-world web applications effected by several PHP modules. Also, the code flow execution is monitored to determine whether an intended workflow of the program can be tainted or not without modifying the sensitive information like cookies, session-id etc.	However, this technique needs source code of the web application for instrumentation, which may not be accessible in practice. Moreover, this technique suffers from false negatives.
Jovanovic et al. 2006 [15]	Pixy is the first open source static analysis tool which is aimed at discovering the XSS vulnerabilities in PHP scripts with the help of data flow analysis. The authors had utilized the inter-procedural and context-sensitive methodologies for identifying the vulnerable spots in the source code of PHP 4 program.	This static analysis tool cannot handle the dynamic features of PHP programming language like encrypted data and confused code. Moreover, this static analysis scheme does not present any executable test cases.
Nadji et al. 2009 [16]	This technique provides an assurance that dynamic user-generated data is isolated from fixed data on server, although mutually united at browser for better defense against XSS attack in an integrity protected way.	This technique is proposed to facilitate user-injected content to be safely interpreted on web Browsers, but with no guarantee about guard against XSS attacks on the web browsers.
Futoransky et al. 2007 [17]	This technique provides a well improved PHP interpreter, GRASP that offers an exact dynamic taint examination for PHP as well as permits precaution guidelines to be characterized on tainted JavaScript strings.	Although it achieves a character-level taint tracking but it achieves a 40-100% overhead.
Jim et al. 2007 [18]	BEEP incorporates a white-list of trusted scripts of all web pages as well as inculcates modified browser for cleaning the mistrustful scripts.	This technique has scalability issues, since each user requires a replica of this particular browser.

Table. 2. Comparison of Existing Defensive Solutions on XSS Attacks Based on Well-Defined Parameters

Related Work Techniques	Exploitation Location	Discovery Site	Scrutinizing Mechanism	Persistent XSS Attack Detection	Non-Persistent XSS Attack Detection	DOM-Based XSS Attack Detection	Client-Side Web Browser Alterations	Server-Side Modifications	JavaScript Code Amendments
[Kirda et al. 2006] [9]	Web Browser	Web Browser	Active	No	Yes	No	Yes	No	No
[Balzarotti et al. 2008] [10]	Web Server	Web Browser	Active	Yes	Yes	No	No	Yes	No
[Sun et al. 2009] [11]	Web Browser	Web Browser	Passive	No	Yes	No	Yes	No	No
[Johns et al. 2008] [12]	Web Browser	Hybrid	Passive	Yes	Yes	No	No	No	Yes
[Bisht et al. 2008] [13]	Web Server or Proxy	Web Server	Active	Yes	Yes	Yes	No	Yes	No
[Balzarotti et al. 2007] [14]	Web Server	Web Server	Active	No	Yes	No	No	Yes	Yes
[Jovanovic et al. 2006] [15]	Web Browser	Web Browser	Active	Yes	Yes	No	No	Yes	No
[Nadji et al. 2009] [16]	Web Server	Hybrid	Active	Yes	Yes	No	Yes	No	Yes
[Futoransky et al. 2007] [17]	Web Server	Web Server	Active	Yes	Yes	No	No	Yes	No
[Jim et al. 2007] [18]	Web Server	Hybrid	Passive	Yes	Yes	No	Yes	No	No

3.2 XSS Attack Vector Injector

This injector utilizes the pool of web forms retrieved through the HTML crawler agent and saved in the web form database. This agent will insert a collection of malicious XSS scripts from a well-known database [5] into the wide variety of different input fields of each location of injection points. The different category of XSS attacks utilized for the assessment of our proposed methodology was retrieved via a repository of malicious XSS attack vectors in [5]. Those malicious XSS attack vectors make usage of wide variety of methods of injecting random script which is disregarded by the web application and, in our situation; it is included as genuine data in the web application.

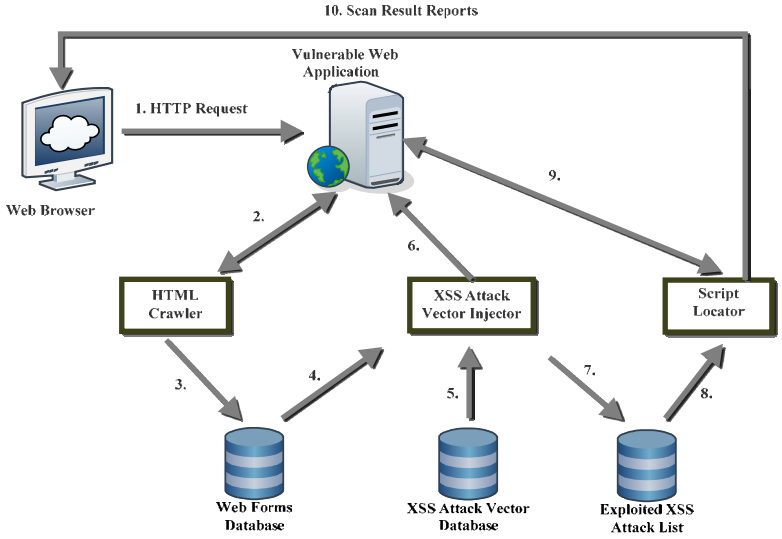

Fig. 1. Proposed Design of Automated Detection System

Algorithm: Automated Detection System	
1.	Discover the possible locations of Injection Points IP_i (for i =1, 2, 3, ----n) in Web Application (W_A) by utilizing the capabilities of HTML crawler.
2.	Extract the collection of Forms F_i (for i=1, 2, 3, ----n) from W_A, which are considered to be possible target for XSS attacks.
3.	Inject the XSS attack vectors X_j (for j = 1, 2, 3, ----n) via XSS attack vector injector into the different Input Fields IF_k (for k = 1, 2, 3, ---n) of each of the possible locations of IP_i.
4.	Finally the Script Locator Agent takes an input as the list of different XSS attacks exploited by XSS Attack Vector Injector and scan for these attacks in the W_A.

Fig. 2. Algorithm of Automated Detection System

3.3 Script Locator

The script locator takes as input the list of exploited XSS attacks, which was previously generated via the XSS attack vector agent, and searches for those XSS attacks in the examined web application. It is essential to look for the malicious inserted scripts as the realization of Persistent XSS attacks cannot be confirmed instantaneously in the reply to the HTTP request which enclosed the malicious XSS attack vector. Therefore, it is conceivable to discover if the web application indicates any of the Persistent XSS vulnerabilities recognized throughout the scanning period. The outcomes achieved in this second phase of scanning of web application are very valuable in the procedure of safeguarding the web application by improving the input validation blunders which led to the realization of such attacks.

3. Implementation and Experimental Results

We have tested our proposed automated detection system on one real-world PHP web application i.e. BlogIt. BloggIt [6] is freely accessible web application platform of blogs that assist users for managing a blog, send mails and post the comments. This web application permits illegitimate users to gain access to sensitive administrative functionality. BlogIt web application and our detection system are installed on a 1.63GHz Dual Core Linux server with 2GB RAM, running Windows 7, Apache Tomcat server (version 2.2.8) and PHP (version 5.5.14). We have collected our datasets as malicious XSS attack vectors from the XSS Cheat Sheet [5]. These attack vectors are exploited on the BlogIt PHP web application. Table 3 shows some of the statistics of observed results on BlogIt PHP web application.

Table 3. Observed Results of BlogIt Web Application

Category of XSS Attacks	**XSS Attack Vector Injected**	**XSS Attack Vector Detected**	**False Negatives**	**Detection Rate**
Event Handlers	30	21	9	70.00%
Character Encoding Attacks	30	20	10	66.66%
Basic XSS Attacks	30	24	6	80.00%
HTML Element Attacks	35	31	4	88.57%
Other Attacks	28	14	14	50.00%

4. Conclusion and Future Work

XSS vulnerabilities are usually exploited to execute mischievous actions by inserting function calls in the possible injection points of web application. Likewise, the current server side XSS vulnerability detection methodologies require variations at both browser and server-side settings as well as transmission of vulnerable data from server to

browser. In this paper, we have proposed an automated detection system, which scans the possible injection locations for the XSS vulnerabilities. The evaluation results showed that our system discovers the XSS vulnerabilities on the BlogIt PHP web application with acceptable runtime overhead.

As a part of further work, we will try to improve the false negative rate our approach. In addition to this, we will also anticipate some mechanism of detecting DOM-based XSS vulnerabilities. Lastly, we wish to spread the utilization of our proposed framework for discovering other vulnerabilities like Phishing, ClickJacking attacks, etc. We will also plan to assess the discovery ability of our detection system on more web applications as a part of our further work.

References

1. Symantec internet security threat report 2009. Retrieved from: http://www.symantec.com/business/threatreport/.
2. Shashank Gupta, B.B. Gupta.PHP-Sensor: A Prototype Method to Discover Workflow Violation and XSS Vulnerabilities in PHP web applications.*in 12th ACM International Conference on Computing Frontiers (CF '15), Ischia, Italy*;2015.
3. Shashank Gupta, Lalitsen Sharma et al.Prevention of cross-site scripting vulnerabilities using dynamic hash generation technique on the server side.*International journal of advanced computer research (IJACR)*;2012, 49-54.
4. Shashank Gupta, Lalitsen Sharma.Exploitation of Cross-site Scripting (XSS) vulnerability on Real World Web Applications and its Defense.*International journal of computer applications (IJCA)*, pp. 28-33;2012.
5. Rsnake. XSS Cheat Sheet, 2008. Retrieved from. https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
6. PHP-blogit: Retrieved from: http:sourceforge.net/projects/php-blogit/
7. Shashank Gupta, B.B. Gupta.BDS: Browser Dependent XSS Sanitizer Book on Cloud-Based Databases with Biometric Applications.*IGI-Global's Advances in Information Security, Privacy, and Ethics (AISPE) series*, pp. 174-191, USA;2014.
8. B.B.Gupta, Shashank Gupta, S.Gangwar, M.Kumar, P.K. Meena.Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and its Defense.*Special Issue of Secured Communication in Wireless and Wired Networks in Journal of Information Privacy and Security, Taylor & Francis Online*, Vol. 11, Issue 2, pp. 118-136;2015.
9. Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic.Noxes: A client-side solution for mitigating cross-site scripting attacks.In SAC'06: *Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 330–337;2006.
10. Balzarotti, Davide, Marco Cova, Vika Felmetsger, Nenad Jovanovic, Engin Kirda, Christopher Kruegel, and Giovanni Vigna.Saner: Composing static and dynamic analysis to validate sanitization in web applications. *In Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 387-401. IEEE;2008.
11. SUN, F., XU, L., AND SU, Z. Client-side detection of XSS worms by monitoring payload propagation. In ESORICS, M. Backes and P. Ning, Eds., vol. 5789 of *Lecture Notes in Computer Science, Springer*, pp. 539–554;2009.
12. M. Johns, B. Engelmann, and J. Posegga. XSSDS: Server-side Detection of Cross-site Scripting Attacks. In Annual Computer Security Applications Conference (ACSAC), 2008.
13. P. Bisht and V. N. Venkatakrishnan. XSS-GUARD: precise dynamic prevention of cross-site scripting attacks. In Detection of Intrusions and Malware, and Vulnerability Assessment, 2008.
14. Balzarotti, Davide, Marco Cova, Viktoria V. Felmetsger, and Giovanni Vigna.Multi-module vulnerability analysis of web-based applications.*In Proceedings of the 14th ACM conference on Computer and communications security*, pp. 25-35. ACM, 2007.
15. Jovanovic, Nenad, Christopher Kruegel, and Engin Kirda.Pixy: A static analysis tool for detecting web application vulnerabilities.*In Security and Privacy, 2006 IEEE Symposium* on, pp. 6-pp. *IEEE*;2006.
16. Nadji, Yacin, Prateek Saxena, and Dawn Song.Document Structure Integrity: A Robust Basis for Cross-site Scripting Defense.*In NDSS*. ;2009
17. Futoransky, Ariel, Ezequiel Gutesman, and Ariel Waissbein.A dynamic technique for enhancing the security and privacy of web applications.*Proc. Black Hat USA*;2007.
18. Trevor Jim, Nikhil Swamy, and Michael Hicks. Defeating script injection attacks with browser-enforced embedded policies.*In WWW'07: Proceedings of the 16th International Conference on World Wide Web*, pp. 601–610;2007.
19. Shashank Gupta, B.B.Gupta.Cross-Site Scripting (XSS) Attacks and Defense Mechanisms: Classification and State-of-Art.*International Journal of System Assurance Engineering and Management, Springer*;2015.
20. Shashank Gupta and B.B.Gupta.XSS-SAFE: A Server-Side Approach to Detect and Mitigate Cross-Site Scripting (XSS) Attacks in JavaScript Code.*Arabian Journal for Science and Engineering, Springer*, pp. 1-24;2015.