Research Article

The Limitations of Cross-Site Scripting Vulnerabilities Detection and Removal Techniques

Isatou Hydara[1], Abu Bakar Md Sultan[2], Hazura Zulzalil[3], Novia Admodisastro[4]

[1,2,3,4]Department of Software Engineering and Information System,
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia
43400 UPM Serdang, Selangor, Malaysia
[1]ishahydara@gmail.com

Article History: Received: 10 November 2020; Revised: 12 January 2021; Accepted: 27 January 2021; Published online: 05 April 2021

Abstract: Web applications have become very important tools in our daily activities as we use them to share and get information, conduct businesses, and interact with family and friends on social media through the Internet. Despite their importance, web applications are plagued with many security vulnerabilities that enable hackers to attack them and compromise user information and privacy. Cross-site scripting vulnerabilities are a type of injection vulnerabilities existing in web applications. They can lead to attacks in web applications due to the lack of proper validation of input data in the affected web pages of an application. Many approaches and techniques have been proposed to mitigate this type of vulnerabilities. However, these solutions have some limitations and cross-site scripting vulnerabilities still remain as a major security problem for web applications. This paper explores and presents the existing techniques for detecting and for removing cross-site scripting vulnerabilities in web application. It gives an overview of cross-site scripting as a security issue in web application and its different types. The advantages as well as the limitations of each techniques are highlighted and discussed. Based on the limitations, some possible future research directions are identified, and recommendations are given as reference for researchers interested in this topic.

Keywords: cross-site scripting, cross-site scripting attacks, cross-site scripting vulnerabilities, web application security

1. Introduction

As technology advances, we rely on the Internet to carry out daily transactions using Desktop and mobile web applications. Banking websites enable customers to transfer money online, pay bills, and conduct many transactions without having to visit any bank branch. Many online stores are also available to support the E-commerce businesses around the world. Social and other sites enable people to consume and share information online. However, this has made web applications more complex (OWASP, 2020), resulting to a lot of security issues affecting them including cross-site scripting (XSS). Security is addressed as an afterthought in most web applications that are currently in used. Despite the efforts of integrating an application’s security requirements throughout the software development lifecycle, many XSS vulnerabilities are still found in web applications. It is reported that 80% of all websites are vulnerable to XSS (Javed & Schwenk, 2014).

XSS vulnerabilities are in the top security vulnerabilities that are in desktop web applications (OWASP, 2020) as well as mobile web applications (Kaur, Pande, Bhardwaj, Bhagat, & Gupta, 2018) and are exploited through XSS attacks on web applications that are available online (OWASP, 2020). Hackers can inject or input malicious code where user inputs are inserted in these applications such as providing username and password when login to an application. If the website does not verify the user inputs or it is done incorrectly, it allows the hacker the opportunity of exploiting for vulnerabilities and conduct malicious activities.

XSS vulnerabilities are categorized into three types, Reflected, Stored and DOM-based XSS (OWASP, 2020). Reflected and stored XSS attacks do occur on the server side while the DOM-based ones occur on the client side of an application. Successful XSS attacks allow attackers to carry out malicious activities such as stealing cookies, transferring private information, hijacking a user’s account, manipulating the web content, or causing denial-of-service attacks.

The existing XSS detection and removal techniques can be categorized into static analysis, dynamic analysis, secure programming, modeling, and hybrid analysis. Each of the categories is discussed in detail in the paper with their limitations.

The rest of the paper is organised as follows. Section II presents the background of and the different types of XSS. Section III details the different detection and removal techniques for XSS vulnerabilities and their limitations, and last but not the least Section IV concludes the paper and provides recommendations for future research.

2. Background of Cross-Site Scripting Vulnerabilities

2.1 Cross-Site Scripting Vulnerabilities

XSS vulnerabilities were first discovered in the 1990s following the emergence of the World Wide Web. They make the top list of the most common security vulnerabilities that are affecting web applications (Hydara, Sultan, Zulzalil, & Admodisastro, 2015). They are classified as input validation problems that make possible the injection of malicious code into trusted web applications. This is due to the failure, during software development, to validate inputs from the web application users that are used in the output (Acunetix, 2020; CWE, 2020; OWASP, 2020; Hussain et al., 2017). This failure to properly verify the user inputs enables hackers to attack an application.

Web applications usually fail to verify the user inputs or they do it incorrectly, thus allowing the hacker the opportunity of exploiting for vulnerabilities(CWE, 2020). Hence, whenever a user visits a web application, the browser can unknowingly execute the malicious code injected by the hacker. Once the browser is infected with the malicious code, the user's sessions are then hijacked and malicious activities can be conducted by the hacker. Figure 1 shows an overview of XSS attack.

Successfully carried out XSS attacks can lead to many problems for both web application and user. The attacker is able to easily inject malicious scripts to an application's user input field and, if not properly validated, can impersonate their victims to carry out many malicious activities. Such activities can include cookie stealing, manipulating web content, causing denial-of-service and transferring private information.

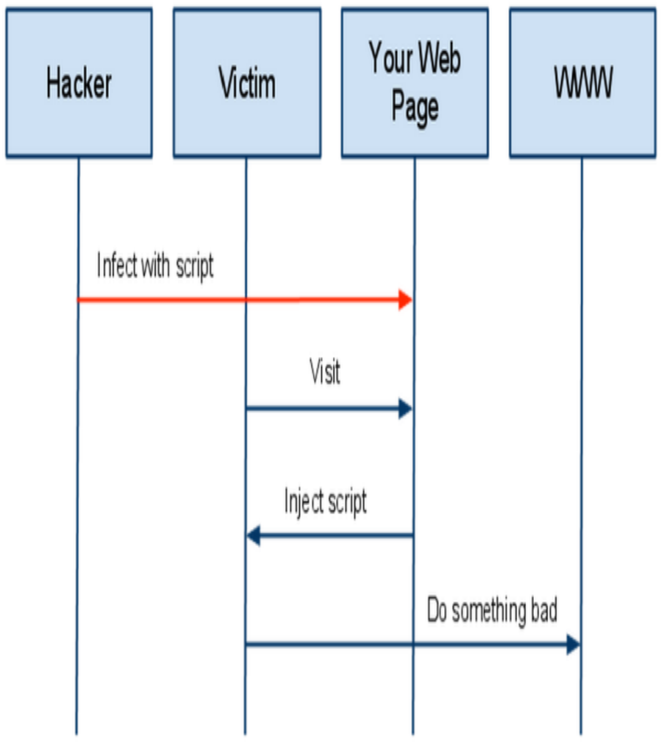

Figure 1. A High Level View of Typical XSS Attack, adapted from (Acunetix, 2020)

2.2 Types of Cross-Site Scripting Vulnerabilities

Reflected XSS (Non-Persistent XSS or Type I XSS) attack occurs when user input is used immediately by the server-side to generate an output page for the user (OWASP, 2020). If the provided input was not validated and is included without any HTML encoding, it can result to the execution of the input provided. Therefore, when invalidated user input has been included in the generated page, then the client-side code is injected and execute into the dynamic page. For example full access to a page's content can be obtained when an attacker manages to convince a potential victim to follow a malicious URL that injects code into the results page.

Stored XSS attacks are the most powerful types of attack. They are also referred to as Persistent or Type II XSS. This form of XSS attack occurs when the inserted user input is first stored on the server (databases, file systems, or other locations). Eventually, this will be displayed to the web application users in a web page without any HTML encoding (OWASP, 2020).

DOM-based XSS (Type III XSS) is quite different from Type I and II. The JavaScript malware payloads do not need to be sent to the Web server to enable exploitation (OWASP, 2020). Pieces of JavaScript code can access a URL request parameter and use it to write some HTML code into its page, if no HTML encoding is done. The newly written data will, therefore, be re-interpreted by the browser such as HTML and this may also add some client-side script.

3. Related Work

Sarmah et al. (Sarmah, Bhattacharyya, & Kalita, 2018) have conducted a comprehensive survey on the detection methods of XSS attacks and classified them according to their deployment areas and mechanisms used. They also identified some tools that support the detection of XSS attacks. Nythiya et al. (Nithya, Lakshmana Pandian, & Malarvizhi, 2015) also survey the existing XSS attack methods and the current approaches for their detection and prevention. They also gave detailed backgrounds on XSS attacks.

Alkhurafi and Al-Ahmad (Al-Khurafi & Al-Ahmad, 2015) survey and discussed the most prevailing web application attacks that are XSS, SQL Injections, Session Management, and Broken Authentication. They illustrated each attack and its different types and identify some possible mitigation methods. Likewise, Al-Ghamdi (AL-Ghamdi, 2013) has identified and discussed various existing techniques for web application security testing, a well the security challenges of new technologies.

Li and Xue (Li & Xue, 2014) conducted a survey on the server-side techniques to secure web applications from attacks including XSS. They discuss three common security vulnerabilities and the types of attacks (input validation vulnerability, session management vulnerability, and application logic vulnerability) that exploit them, and identify existing approaches that to mitigate them. They also highlighted emerging challenges imposed by new programming methodologies and technologies.

Garcia-Alfaro and Navarro-Arribas (Garcia-Alfaro & Navarro-Arribas, 2008) also surveyed the two most common XSS attacks, reflected XSS and Stored XSS and how they are carried out. They then discuss the existing solutions to tackle these attacks and their applicability.

4. Exisiting Xss Detection and Removal Techniques and Their Limitations

This section briefly discusses the existing techniques and approaches for the detection and removal of XSS vulnerabilities in desktop and mobile web applications. They can be categorized into static analysis, dynamic analysis, modeling, and hybrid analysis. Table 1 summarizes the techniques and their limitations.

4.1. Static Analysis

Static analysis techniques are solutions that carry out XSS vulnerability detection at the source code level of web applications(Gupta & Gupta, 2016; Kurniawan, Abbas, Trisetyarso, & Isa, 2018). They help to track data through an application and identify vulnerable parts of a source code thereby detecting XSS vulnerabilities. The most common detection techniques under static analysis are static taint analysis, data flow analysis, string analysis, precise alias analysis, program slicing, and symbolic execution. The advantages of these techniques are that they can be carried out without running the source code and the detected vulnerabilities can be removed directly from the source code. The major limitation of these techniques is their high rate of false positives. This is because of the conservative nature of these techniques. Another limitation is the source codes of applications are needed to conduct these security tests.

4.2. Dynamic Analysis

Dynamic analysis techniques detect XSS vulnerabilities in web applications that are already deployed online during runtime(Gupta, Gupta, & Chaudhary, 2018; Kaur et al., 2018). They intercept and analyze input data coming into an application from users and determine whether it is harmful or not. Such techniques include penetration testing, web monitoring, filtering, dynamic analysis, taint tracking, and flow analysis. Their advantage is they can be carried out without the availability of the application source code. Also, some malicious behaviours in an application can only be detected while running the application. The limitations of these techniques are that attackers can use obfuscation to hide their attack patterns and carry out successful XSS attacks. Also, these techniques require more computing resources to simulate the running environment of web applications.

4.3. Secure Programming

Secured programming techniques detect unsecure coding in the programming environment and ensure that programming guidelines and rules are followed during the development of web applications (Grabowski, Hofmann, & Li, 2012; Johns, Beyerlein, Giesecke, & Posegga, 2010). These techniques include Type Systems (Grabowski et al., 2012) a technique that automatically enforces programming guidelines, and ELET (Embedded Language Encapsulation Type)(Johns et al., 2010) used for enforcing secure code generation in a programming language. Their advantage is they can help developers to adhere to secure coding practices while developing web applications. The limitation of these techniques is that many developers do not like to use them as they feel they slow them down in their work.

4.4. Modeling

Modeling techniques (Elhakeem & Barry, 2013; Gol & Shah, 2015) are not as commonly used in XSS detection as the previous two categories, but they are still important solutions. Existing modeling techniques for XSS detection include model checking, finite state machine, data mining, and threading. Most models are designed to provide guidelines to developers and security testers during coding and testing. They have the advantage of helping analysts and developers to mitigate vulnerabilities in all stages of application development. The major limitation of these is that in most cases the guidelines are not read and followed while coding due to time constraints of development projects.

4.5. Hybrid Analysis

Due to the limitations of the previous techniques and approaches, most research studies are now combining different techniques of XSS detection known as hybrid analysis (Wang, Zhu, Tan, & Zhou, 2017). It is used in order to reduce the limitations of using a single technique or approach. Combining static and dynamic analyses as well as modeling and secure programming has the advantage of providing more coverage in terms of designing, coding, security testing of the code and the running of the application, as well as guiding the developers. Hybrid analysis still have their limitations; however, they are an improvement on previous techniques.

Table 1. Summary of XSS Detection Techniques**.**

TECHNIQUE	DESCRIPTION	ADVANTAGES	LIMITATIONS
Static Analysis	An analysis technique that uses the source code of an application to detect vulnerabilities without running the said application.	Vulnerability testing can be done without running the application, and in the early stage of development.	Higher rate of false positives and needs source code.
Dynamic Analysis	A technique that analyses a web application for potential while it is running.	It does not require source code to be carried out.	Attackers can use obfuscation to bypass defenses.
Secure Programming	A technique that detects unsecure code in the development environment and forces the developers to use secure coding practices.	Vulnerabilities can be detected and eliminated while coding the application.	Hints and suggestions for secure coding may be ignored by developers
Modeling	These techniques are used to demonstrate safe practices when designing and developing web applications.	They enable to mitigate vulnerabilities at all stages of application development.	Guidelines are usually not followed by developers.
Hybrid Analysis	Hybrid techniques are a combination of any two detection techniques that provided better performance than any of the individual techniques.	Has the benefits of the combined techniques	May face limitations of its combining techniques.

5. Conclusion

In this short study, we have investigated the various techniques that have been proposed by previous researchers to solve the XSS security problems. We have discussed the background of XSS attacks and the different types of XSS attacks. We then identified the limitations in the existing techniques we discussed as well as the needed improvements. Based on the limitations identified, new areas of research should be explored continuously with the expectation of discovering more secure ways of developing software and preventing more XSS attacks. Researcher can work on reducing the high rates of false positives and false negatives that limit static analysis and dynamic analysis, respectively. More focus on secure programming and modelling is needed to educate, train and guide developers on the importance of vulnerabilities mitigation. A combination of different techniques with hybrid analysis that will put together the best features of the combining techniques is also a good research direction.

6. Acknowledgment

We acknowledge that this research received support from the Fundamental Research Grant Scheme FRGS/1/2015/ICT01/UPM/02/12 awarded by Malaysian Ministry of Higher Education to the Faculty of Computer Science and Information Technology at Universiti Putra Malaysia.

References

1. Acunetix. (2020). What is Cross Site Scripting and How Can You Fix it? Retrieved February 6, 2020, from http://www.acunetix.com/websitesecurity/cross-site-scripting/
2. AL-Ghamdi, A. S. A.-M. (2013). A Survey on Software Security Testing Techniques. *International Journal of Computer Science and Telecommunications*, *4*(4), 14–18.
3. Al-Khurafi, O. B., & Al-Ahmad, M. A. (2015). Survey of Web Application Vulnerability Attacks. In *2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)* (pp. 154–158). https://doi.org/10.1109/ACSAT.2015.46
4. CWE. (2020). CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (2.5). Retrieved February 5, 2020, from http://cwe.mitre.org/data/definitions/79.html
5. Elhakeem, Y. F. G. M., & Barry, B. I. A. (2013). Developing a security model to protect websites from cross-site scripting attacks using ZEND framework application. In *Proceedings - 2013 International Conference on Computer, Electrical and Electronics Engineering: "Research Makes a Difference", ICCEEE 2013* (pp. 624–629). IEEE. https://doi.org/10.1109/ICCEEE.2013.6634012
6. Garcia-Alfaro, J., & Navarro-Arribas, G. (2008). A survey on detection techniques to prevent cross-site scripting attacks on current web applications. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5141 LNCS*, 287–298. https://doi.org/10.1007/978-3-540-89173-4_24
7. Gol, D., & Shah, N. (2015). Detection of Web Appication Vulnerability Based on RUP Model. In *National Conference on Recent Advances in Electronics & Computer Engineering (RAECE)* (pp. 96–100). IEEE. https://doi.org/10.1109/RAECE.2015.7510233
8. Grabowski, R., Hofmann, M., & Li, K. (2012). Type-Based Enforcement of Secure Programming Guidelines — Code Injection Prevention at SAP. *FAST 2011, Lecture Notes in Computer Science*, *7140*, 182–197.
9. Gupta, S., & Gupta, B. B. (2016). XSS-SAFE: A Server-Side Approach to Detect and Mitigate Cross-Site Scripting (XSS) Attacks in JavaScript Code. *Arabian Journal for Science and Engineering*, *41*(3), 897–920. https://doi.org/10.1007/s13369-015-1891-7
10. Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). Hunting for DOM-Based XSS vulnerabilities in mobile cloud-based online social network. *Future Generation Computer Systems*, *79*, 319–336. https://doi.org/10.1016/j.future.2017.05.038
11. Hussain, A., Mkpojiogu, E.O.C., Nawi, M.N.M. (2017). Capturing customer satisfaction and dissatisfaction in software requirements elicitation for features in proposed software systems. Journal of Engineering and Applied Sciences, 12 (21), pp. 5590-5597.
12. Hydara, I., Sultan, A. B. M., Zulzalil, H., & Admodisastro, N. (2015). Current state of research on cross-site scripting (XSS) - A systematic literature review. *Information and Software Technology*, *58*, 170–186. https://doi.org/10.1016/j.infsof.2014.07.010
13. Javed, A., & Schwenk, J. (2014). Towards Elimination of Cross-Site Scripting on Mobile Versions of Web Applications. In Y. Kim, H. Lee, & A. Perrig (Eds.), *14th WISA: International Workshop*

on Information Security Applications (Vol. LNCS 8267, pp. 103–123). Jeju Island, Korea: Springer International Publishing Switzerland. https://doi.org/10.1007/978-3-319-05149-9

14. Johns, M., Beyerlein, C., Giesecke, R., & Posegga, J. (2010). Secure Code Generation for Web Applications. *Lecture Notes in Computer Science*, *5965*, 96–113.
15. Kaur, G., Pande, B., Bhardwaj, A., Bhagat, G., & Gupta, S. (2018). Efficient yet Robust Elimination of XSS Attack Vectors from HTML5 Web Applications Hosted on OSN-Based Cloud Platforms. *Procedia Computer Science*, *125*, 669–675. https://doi.org/10.1016/j.procs.2017.12.086
16. Kurniawan, A., Abbas, B. S., Trisetyarso, A., & Isa, S. M. (2018). Static Taint Analysis Traversal with Object Oriented Component for Web File Injection Vulnerability Pattern Detection. *Procedia Computer Science*, *135*, 596–605. https://doi.org/10.1016/j.procs.2018.08.227
17. Li, X., & Xue, Y. (2014). A Survey on Server-Side Approaches to Securing Web Applications. *ACM Computing Surveys (CSUR)*, *46*(4), 1–30. https://doi.org/10.1145/2541315
18. Nithya, V., Lakshmana Pandian, S., & Malarvizhi, C. (2015). A survey on detection and prevention of cross-site scripting attack. *International Journal of Security and Its Applications*, *9*(3), 139–152. https://doi.org/10.14257/ijseia.2015.9.3.14
19. OWASP. (2020). Cross-site Scripting (XSS) - OWASP. Retrieved February 5, 2020, from https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
20. Sarmah, U., Bhattacharyya, D. K., & Kalita, J. K. (2018). A survey of detection methods for XSS attacks. *Journal of Network and Computer Applications*, *118*(June), 113–143. https://doi.org/10.1016/j.jnca.2018.06.004
21. Wang, R., Zhu, Y., Tan, J., & Zhou, B. (2017). Detection of malicious web pages based on hybrid analysis. *Journal of Information Security and Applications*, *35*, 68–74. https://doi.org/10.1016/j.jisa.2017.05.008