**Bulletin of Electrical Engineering and Informatics**
Vol. 99, No. 1, Month 2099, pp. 1~1x
ISSN: 2302-9285, DOI: 10.11591/eei.v99i1.paperID

# Important Factors to Remember when Constructing a Cross-site Scripting Prevention Mechanism

**Md. Maruf Hassan[1,2], Badlishah R Ahmad[2], Ashrafia Esha[1], Rafika Risha[1], Mohammad S Hasan[3]**
[1]Daffodil International University, Dhaka, Bangladesh
[2]Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia
[3]Staffordshire University, Stoke-on-Trent, United Kingdom

**Article Info**

***Article history:***

Received August 05, 2021
Revised September 30, 2021
Accepted month dd, yyyy

***Keywords:***

Cyber Security
Cross Site Scripting (XSS)
XSS Prevention
Web application vulnerability

**ABSTRACT**

Web application has become an essential part of daily activities to provide easy accessibility that ensures better performance. It is a platform where sensitive information such as username, password, credit card details, operating system and software version etc. is stored that attracts intruders to generate most of their attacks. Intruders can steal valuable data by compromising web application security flaws; Cross Site Scripting (XSS) vulnerability is one of these. Several studies have been conducted in order to prevent the XSS vulnerability. In this research, we searched Scopus Indexed articles published in the last 11 years (between 2008 and 2020) using two keywords ("XSS Attack Prevention" and "XSS Prevention"). The purpose of this study was to conduct a literature review on XSS prevention techniques e.g. strengths and weaknesses, including structural issues and real-time deployment location in order to extract valuable information. This review identified 14 articles among the 25 selected articles that provided various suitable prevention techniques for XSS attacks. Seven articles are based on tools that have been implemented and take into account design, coding, testing, and integrating validation processes, six articles are about server site solutions, and one is about automatic mitigation solutions. As a result, this research will be invaluable in guiding the advancement of XSS prevention techniques.

***Corresponding Author:***

Md. Maruf Hassan
Department of Software Engineering, Faculty of Science and Information technology, Daffodil International University
102, Shukrabad, Mirpur Road, Dhaka – 1207, Bangladesh
Email: maruf.swe@diu.edu.bd

## 1. INTRODUCTION

Web applications are a mandatory requirement for businesses, organizations, and customer-behavior solutions in order to provide easy access and improved performance to their target users in modern life. Security is a major concern in web applications since they contain personal data and information about people. The most common web application vulnerabilities, as per the Open Web Application Security Project (OWASP) are Injection, Broken Authentication and Session Management, Cross-Site Scripting (XSS), Broken Access Control, Security Misconfiguration, Sensitive Data Exposure etc. [1]. Ensighten published a report in 2019 stating that XSS accounts for 40% of web application attacks [2].

XSS is a client-side code injection attack that allows an attacker to execute malicious JavaScript in the browser of another user by injecting vulnerable web application pages. When a random user visits the compromised page, the page will deliver the malicious script into the victim's browser and execute it. Three types of XSS attacks and their removal techniques are – (i) an attacker injects a malicious script which is permanently stored on the targeted database server – Stored XSS vulnerability; (ii) users also inject XSS attacks via phishing emails and other websites when they get a request from a crafted link, after clicking these links the injected code reflects the attack to the user's browser – Reflected XSS attacks or XSS Type-I attack; (iii) Document Object Model (DOM) Based XSS simply means an XSS vulnerability that appears in the DOM instead of the HTML part which occurs when a page is managing an action or performing any specific transactions [3-7].

XSS attacks should be intensely managed for security purposes. During the research, we observed that the majority of the proposed models and implemented tools or techniques are intended to identify and prevent only one or two types of XSS vulnerabilities. It should be noted that while these tools can prevent XSS vulnerabilities, they face obstacles in reducing the attack rate. Therefore, the goal of the study was to assist users in gaining an independent understanding of the existing XSS vulnerability prevention mechanisms as well as their strengths and weaknesses. This article also discussed the deployment location of the XSS vulnerability in web applications.

The remainder of this paper is organized as follows. Section 2 presents some relevant research works, section 3 provides details of the methods of this research. The result, evaluation, and discussion are described in section 4. And finally, section 5 concludes the paper.

## 2. RELEVANT RESEARCH WORKS

Paper [8] proposed utilizing Genetic Algorithm (GA) in conjunction with Reinforcement Learning (RL) and threat intelligence to defeat XSS attacks and the outcomes were not only more adaptable to changes in XSS payloads, but they are also more comprehensible to end-users. Research work [9] examined XSS and its taxonomy including XSS attack devices, as well as analysis and prevention of XSS forgeries. The study in [10] suggested a unique method called obfuscation to safeguard online applications from SQL injection attacks, XSS attacks, and reverse engineering attacks. A comprehensive analysis of XSS exploitation as well as existing detection and prevention mechanisms are discussed in [11]. Paper [12] compared XSS attack detection techniques in terms of algorithm simplicity, algorithm type, and performance metrics. Vital data on the operations of Machine Learning (ML), Predictive Analytics, and the development of the significant web that properly evaluates and eliminates SQL Injection Attack (SQLIA) with experiential value demonstrated in the Receiver operating curve and Confusion matrix was provided in [13]. The goal of the work [14] was to evaluate the effectiveness of various ML algorithms in identifying XSS attacks in web apps and websites, as well as to utilize ML to detect XSS attacks through various ML methods. The study [15] described a multi-layer prevention approach in which the attacker is defended at the API key authentication level using an encryption technique that prohibits the attacker from gaining direct access to the API. Google's secure-by-design engineering approach was proposed in [16] which successfully avoids DOM-based XSS vulnerabilities in large-scale web development. The study [17] proposed an ML technique for detecting stored XSS attacks and defending a Representational State Transfer (REST) web service written in JAVA, which was evaluated in a specifically designed test-bed simulation environment that included the IntelliJ IDEA environment, Postman, and a web browser. A secure framework that may be used to accomplish real-time detection and mitigation of XSS attacks in cloud-based web applications via Deep learning (DL) at a high level of accuracy was presented in [18]. A solution integrating three techniques to determine the most difficult attacking challenges is revealed in [19] by implementing Random Forest (RF), Logistic Regression (LR), k-Nearest Neighbors (k-NN), and Support Vector Machine (SVM) algorithms, Content Security Policy (CSP) approach, Web Application Firewall (WAF), Intrusion Detection System (IDS), and Intrusion Prevention System (IPS). The study [20] investigated utilizing neural networks to identify XSS vulnerabilities utilizing static methods.

## 3. METHODS

### 3.1. Article Searching Procedure

We performed a systematic search strategy to find the publications that detail how XSS vulnerabilities in web applications are exploited. In our systematic searching procedure, we searched two keywords from the Scopus Indexed databases in order to assess the article. We began by searching for publications published between 2008 and 2021 using the term "XSS Attack Prevention" and "XSS Prevention".

### 3.2. Article Inclusion and Exclusion Criterion

We employed a set of criteria to include and reject articles from the set of articles found through a Scopus indexing database search. Then we studied the title, abstract, methodology, and findings of each article to determine which ones to include and reject from the list of papers obtained by our systematic searching process and only articles that were utilized to avoid XSS attacks were considered.

### 3.3. Data Extraction

Each article was assessed based on the following key points: (1) Performance comparison of different types of XSS attack, (2) Overview of different types of XSS vulnerability detection and prevention techniques, (3) Deployment location of XSS vulnerability in web applications.

### 3.4. Defensive Coding

The defense code mechanism is performed in three stages as shown in Figure 1. The XSS prevention strategies were chosen first, followed by various XSS defense code techniques. Finally, injected locations have been identified using defense coding techniques.

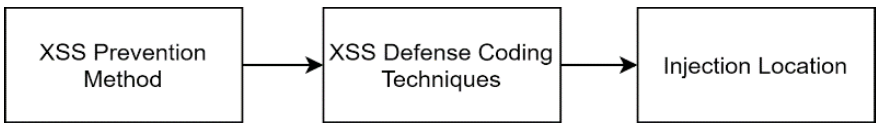

Figure 1. The architecture of Defense Coding Mechanism

Figure 2 presents the categorization of defense coding mechanism.

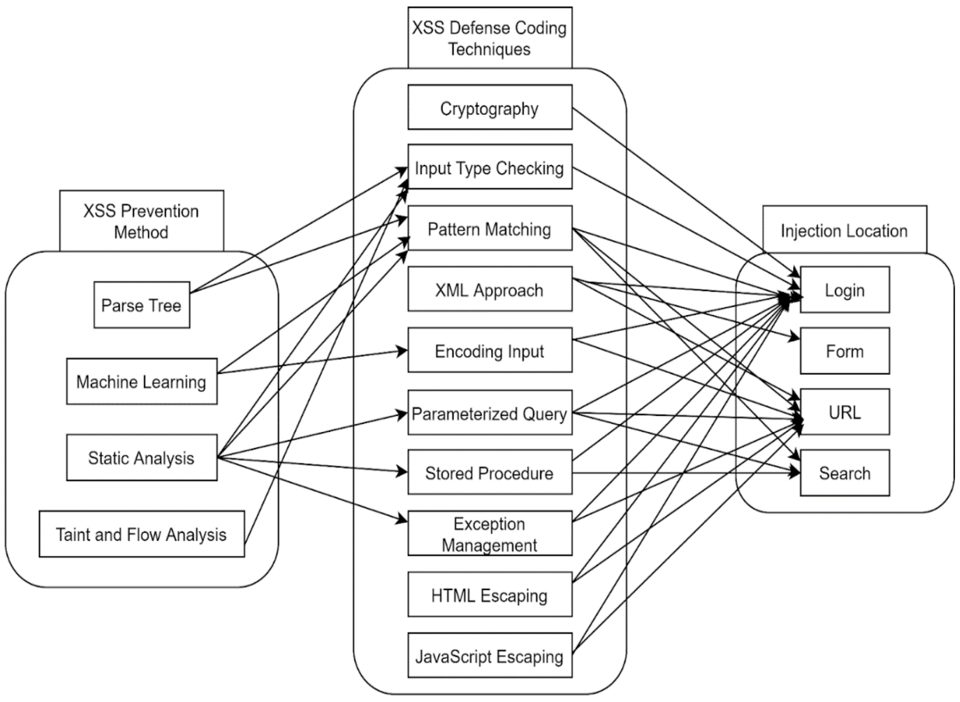

Figure 2. Categorization of Defense Coding Mechanism

An updated XSS defensive methodology for cloud platforms was given in [21] which first scans HTTP requests for embedded URI links that point to URLs of external JS files containing malicious XSS payloads. Both coarse-grained and exact taint tracking are implemented in JavaScript, and the researchers illustrate how the precise taint tracking API may be used to fight against SQL injection and XSS attacks [22]. The study [23] offered a client-side solution using a knapsack cryptographic local proxy with encryption and decryption functionality to protect cookies against XSS attacks. This solution encrypts the cookie's value (session-ID) attribute at the cryptographic local proxy before delivering it to the browser, and then sends the encrypted cookie's requests to the cryptographic local proxy, which decrypts them and forwards them to the web server. A new approach to thwart XSS attacks was presented in [24] that is independent of the languages used to construct web apps and solves XSS vulnerabilities that originate from different interfaces. The approach is structured, configured, and constructed in .Net, XML, and XSD, then tested in a web application written in JSP/Servlets and deployed in the JavaBeans Open-Source Software (JBOSS) application server. It is determined to be effective since it allows for cross-language use with very little configuration to prevent

XSS. A context-sensitive encoder is derived from context-free grammars in order to provide appropriate unparsing of potentially malicious input data for all context-free languages [25]. This unparsing process produces documents in which the input data has no effect on the structure of the document and has no effect on its intended semantics.

The research [26] proposed a dynamic detection framework (TT-XSS) for DOM-XSS using taint tracking at the client side which involved rewriting all JavaScript features and DOM APIs to taint browser rendering. To this purpose, additional data types and methods are introduced to enhance the original data structure's semantic description capabilities, based on which the taint traces were evaluated during page parsing by tainting all sources, sinks, and transfer processes. The study in [27] described a method for detecting and preventing XSS vulnerabilities that used the Knuth-Morris-Pratt (KMP) string matching algorithm to match the user's input string with the stored pattern of the injection string in order to detect any malicious code. To detect and remediate XSS vulnerabilities in web application source code, paper [28] offered a context-sensitive solution based on static taint analysis and pattern matching approaches. The proposed approach was implemented in a prototype tool and tested on a public data set of 9408 samples, with the testing results indicating that the tool outperforms existing popular open-source tools in detecting XSS vulnerabilities. Using Java EE, a REST architectural style, a design pattern facade, and tools that allow testing and validation procedures, the study [29] offered to reduce vulnerabilities and make web applications secure from the aspect of design, development, and deployment.

After studying these articles, we can conclude that a defensive coding mechanism is a type of defensive design that works in the event of a failure, especially when high availability, safety, or security are required. It aims to increase the overall quality of software and source code by making the source code accessible and by ensuring that the software performs appropriately in the face of unexpected inputs or user actions.

## 4. RESULT, EVALUATION AND DISCUSSION

### 4.1. Search Article Results

The article’s search result is summarized in Figure 3.

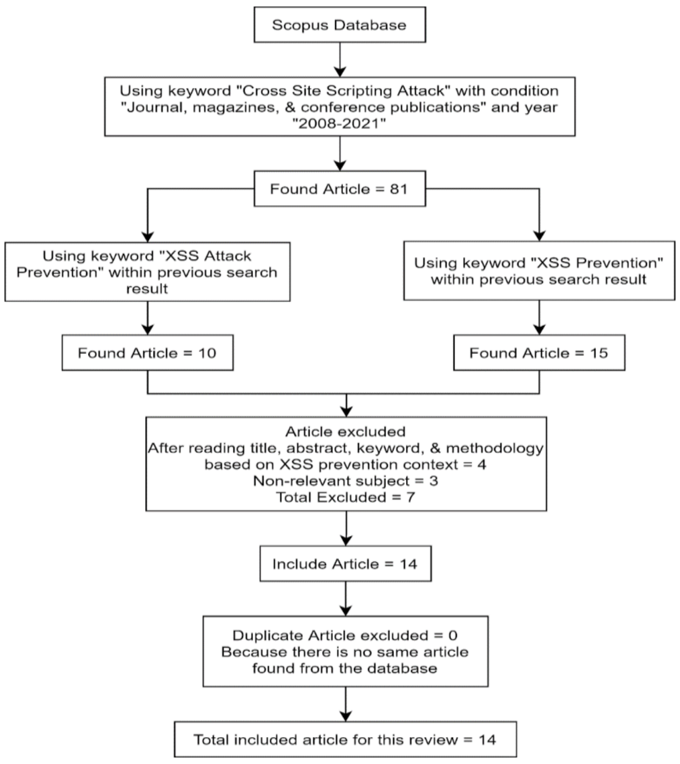
Scopus Database
Using keyword "Cross Site Scripting Attack" with condition "Journal, magazines, & conference publications" and year "2008-2021"
Found Article = 81
Using keyword "XSS Attack Prevention" within previous search result
Using keyword "XSS Prevention" within previous search result
Found Article = 10
Found Article = 15
Article excluded
After reading title, abstract, keyword, & methodology based on XSS prevention context = 4
Non-relevant subject = 3
Total Excluded = 7
Include Article = 14
Duplicate Article excluded = 0
Because there is no same article found from the database
Total included article for this review = 14

Figure 3. The process of article searching

Based on the phrase cross-site scripting attack, we discovered 81 publications published in renowned journals and conferences between January 2008 and December 2021 using our systematic article search process. Then we scanned all of the articles in detail, identifying the most important points in each. We used two keywords to find the publications: "XSS Attack Prevention" and "XSS Prevention," which yielded 10 and 15 articles, respectively. We next studied each article's title, abstract, keyword, and technique before selecting 14 publications for analysis from the Scopus Indexed Databases. As a result of this search, 14 publications were found that explored the XSS vulnerability prevention technique in web applications.

### 4.2. Descriptive Analysis

Table 1 summarizes the findings of the 14 articles on XSS attack prevention strategies.

Table 1. XSS attack prevention technology summary

| Authors | Tools | Strength | Weakness |
|---|---|---|---|
| Mukesh Kumar Gupta [28] | XSSDM | XSS vulnerabilities are precisely detected and mitigated using taint analysis and pattern matching techniques | It is necessary to improve support for the object-oriented paradigm |
| Daniel Guamán et al. [29] | RESTful WS | Reduce vulnerabilities and make web applications secure from the standpoints of design, coding, and deployment while taking testing and validation into account | To ensure the integrity of the system, safety, security, and software development standards should be established |
| Shahriar and Zulkernine [30] | S2XS2 | Create policies and dynamically insert borders | Time-consuming and low detection capability |
| Hao Chen et al. [31] | An execution flow analyzer | Create the FSA in order to simulate the client program's actions | Need to modify the web source code |
| W. Xiao and J. Sun et al. [32] | information flow | The security of sensitive data is ensured by using JSTFlow as a browser proxy | There are restrictions to the sensitive data that has been detected |
| Barhoom and Kohail [33] | server-side solution | Prevent untrusted user input, modify the trusted code structure | Retrieve from the accessible network's server |
| Prithvi Bisht et al. [34] | XSS-GUARD | Define the server-side code and eliminate the no-output code | Do not forbid the permissible benign HTML |
| B. Mewara and s. Bairwa et al. [35] | XSS-ME | Easy accommodation of complex attack | Can detect and prevent only one attack |
| Jasmin A. Caliwag et al. [36] | escaping technique | Capable of preventing XSS attack on the created online inventory system by removing unnecessary data | XSS attack mitigation was the sole focus |
| Swati Maurya [37] | 'Positive Security Model' based 'Server-side solution' | Only those tags that are known to be safe for performing XSS with faster time processing in matching the attack vectors from the Blacklist are permitted | Attackers can circumvent the input sanitizer though it will be blocked later |
| Gupta and Gupta [38] | XSS-SAFE | Sanitization routines are injected into the JavaScript source code to detect and mitigate maliciously injected XSS attack vectors | Only detects the relationship between stored and injected features in the JavaScript source code |
| Chandra, V.S. et al. [39] | BIXSAN: Browser Independent XSS Sanitizer for prevention of XSS attacks | HTML parse tree producer is used to improve the inconsistency of web browser performance as well as to recognize static script tags | Unable to detect XSS of dynamically growing parsing quirks in the XSS cheat sheet as the method evaluated by referring to it |
| Saxena, P. [40] | FLAX: Systematic discovery of client-side validation vulnerabilities in rich Web applications | A lightweight tool in comparison to others, with no false positives and sufficient scalability | FLAX testing has not focused on the complexities of sanitization errors, which persist in the client-side javascript code |
| Wurzinger et al. [41] | SWAP: Mitigating XSS attacks using a reverse proxy | Strong detection of differences between benign and injected javascript code | Many types of XSS attacks are undetectable |

We discovered a method that can automatically insert borders and establish policies to mitigate the attacking probability of an XSS vulnerability [30]. To safeguard the web application from XSS attacks, an execution flow analyzer has been built that can emulate client program behavior [31]. A browser proxy has been designed to secure the security of sensitive data using an information flow approach [32]. A server-side approach has been implemented in some research that limits user input from untrusted sites, removes the no-output script, and readily accommodates complicated attacks [33, 34, 35, 36, and 37]. Several researchers have produced some technologies that can reduce XSS attacks from online applications by taking into account design, coding, testing, and incorporating validation [28, 29, 38, 39, and 40].

### 4.3. Evaluation Based on the Attack Type

Table 2. Evaluation Based on the Attack Type

| Authors | Deployment location | Stored XSS (Persistent) | Reflected XSS | DOM XSS |
|---|---|---|---|---|
| Ran Wang [26] | Client-side | N | N | Y |
| Oluwakemi Christiana Abikoye [27] | Server-side | N | Y | Y |
| Mukesh Kumar Gupta [28] | server-side | Y | Y | N |
| Shahriar and Zulkernine[30] | Server-side | Y | Y | N |
| Hao Chen et al.[31] | Client-side | N | N | Y |
| W. Xiao and J. Sun et al. [32] | Client-side | N | N | Y |
| Barhoom and Kohail[33] | Server-side | Y | N | N |
| Prithvi Bish et al. [34] | Server-side | N | Y | N |
| B. Mewara and s. Bairwa et al. [35] | Server-side | Y | Y | N |
| Gupta and Gupta [38] | server-side | Y | N | N |
| Chandra et al. [39] | server-side | N | Y | N |
| Saxena et al. [40] | Client-side | Y | N | N |
| Ran Wang [38] | Client-side | N | N | Y |
| Joel Weinberger [39] | Client-side | N | N | Y |
| Ankit Shrivastava [40] | server-side | N | Y | Y |
| Wurzinger et al. [41] | Client-side | Y | Y | N |
| JinkunPan et al.[42] | Client-side | N | N | Y |
| Gupta et al.[43] | Server-side | Y | Y | N |
| Gupta and Gupta [44] | Cloud | N | Y | Y |
| Joel Weinberger [45] | Client-side | N | N | Y |
| Ankit Shrivastava [46] | server-side | N | Y | Y |
| Cao et al. [47] | server-side | N | Y | Y |
| Stamm et al. [48] | Client-side | N | Y | N |
| Gundy et al. [49] | server-side | Y | Y | N |
| Agten et al. [50] | server-side | Y | N | N |
| Shahriar et al. [51] | server-side | Y | N | N |

** “Y” indicates a method that can successfully stop an attack of that type and “N” indicates a method that cannot stop an attack of that type.

As stated in table 2, we investigated and evaluated each recommended strategy to see if it might be used to counteract a specific attack. We conducted an analytical evaluation based on our experience because we were unable to assess any of the methods in real-time practices due to the lack of implementation codes for most methods. Except for DOM-based XSS, we found five articles regarding tools developed for server-side XSS attacks that can detect stored and reflected XSS [28, 30, 35, 43, and 49]. Four articles discussed how their implemented tool can only detect stored XSS in server-side web applications [33, 38, 50, and 51]. Only reflected XSS can be detected by two studies that are deployed for server-side location [34, and 39]. Three studies highlighted how their tools can detect reflected and DOM XSS from server-side locations [27, 46, and 47]. Five studies that are deployed for client-side location can only identify DOM XSS [26, 31, 32, 42, and 45]. A tool for detecting stored XSS in client-side web applications was developed in a study [40]. A study developed a client-side tool capable of detecting both stored and reflected XSS [41]. In a paper, techniques were created to detect stored XSS on cloud-based online applications [44]. A study developed a tool for detecting reflected XSS in client-side web applications [48].

### 4.4. Evaluation Based on Deployment

Table 3 presents an analysis of each approach based on different deployment requirements. Three methods are highly resistant to attacks: cryptography, exception management, and parsing. Pattern matching, HTML escaping, JavaScript escaping, and ML are four techniques that are moderately resistant to attack, whereas the XML approach is not.

Table 3. Evaluation based on deployment requirements

| Method | URL | Login | Search | Detect | Prevent | Modify Code Base | Resistant to Attack |
|---|---|---|---|---|---|---|---|
| Cryptography | N | Y | N | N | Y | Y | HIGH |
| Pattern Matching | Y | Y | Y | Y | Y | N | MEDIUM |
| XML Approach | Y | Y | Y | N | Y | N | LOW |
| Exception Management | Y | Y | Y | Y | Y | N | HIGH |
| HTML Escaping | Y | N | Y | Y | N | N | MEDIUM |
| JavaScript Escaping | Y | Y | Y | Y | Y | N | MEDIUM |
| Machine Learning | Y | Y | Y | Y | Y | Y | MEDIUM |
| Parsing | Y | Y | Y | Y | Y | Y | HIGH |

** "Y" indicates the method can be deployed to that injection parameter and "N" indicates the method cannot be deployed to that injection parameter.

## 5. CONCLUSION

In this paper, we presented a case study on the prevention of XSS vulnerabilities in web applications. We classified various types of defense coding techniques based on XSS prevention methods. Furthermore, based on the deployed locations, we discussed the strengths, weaknesses, comparison, and evaluation of various types of XSS prevention techniques. The points raised during our discussion will be useful in making a decision about implementing XSS prevention tools to protect web applications from XSS vulnerability exploitation. Moreover, we have focused on research directions and challenges related to XSS prevention techniques. Although several techniques for preventing XSS attacks have been implemented, their use for real-time deployed location and extraction of estimated valuable information may still be compromised by the factor highlighted in this study.

## ACKNOWLEDGEMENTS

The authors would like to acknowledge an Erasmus+ International Credit Mobility (ICM) 2019 fund for Bangladesh awarded to Staffordshire University, UK.

## REFERENCES

[1] "Open Web Application Security Project | OWASP." report 2017. Available: https://www.owasp.org/index.php/Top_10_2017-Main. [Accessed: 27-Apr-2021]
[2] "Ensighten". Available: https://www.ensighten.com/blog/defending-against-cross-site-scripting-xss-attacks. [Accessed: 31-Jul-2021]
[3] K. Pranathi, S. Kranthi, A. Srisaila and P. Madhavilatha, "Attacks on Web Application Caused by Cross Site Scripting," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2018, pp. 1754-1759. doi: 10.1109/ICECA.2018.8474765

[4] G. Shanmugasundaram, S. Ravivarman and P. Thangavellu, "A study on removal techniques of Cross-Site Scripting from web applications," 2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC), Chennai, 2015, pp. 0436-0442. doi: 10.1109/ICCPEIC.2015.7259498

[5] A. Shrivastava, S. Choudhary and A. Kumar, "XSS vulnerability assessment and prevention in web application," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, 2016, pp. 850-853. doi: 10.1109/NGCT.2016.7877529

[6] S. Tuza, S. Alarabi, S. Alamri and D. N. Innab, "Advanced Approach on XSSDS Technique," 2018 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, 2018, pp. 1-5. doi: 10.1109/NCG.2018.8593178

[7] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, "Current state of research on cross-site scripting (XSS) – A systematic literature review," Information and Software Technology, vol. 58, pp. 170–186, 2015.

[8] I. Tariq, M. A. Sindhu, R. A. Abbasi, A. S. Khattak, O. Maqbool and G. F. Siddiqui, "Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning," Expert Systems with Applications, 168, 114386, 2021, doi:10.1016/j.eswa.2020.114386.

[9] L. J. Rao, S. K. Basha, and V. R. Krishna, "Prevention and Analysing on Cross Site Scripting, "Advances in Intelligent Systems and Computing Intelligent System Design, 731-739, 2020, doi:10.1007/978-981-15-5400-1_69.

[10] D. Kumar, A. Kumar and L. Shing,"Enhance Web Application Security Using Obfuscation," Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12(12):1984-1989, 2021.

[11] K. Vijayalakshmi and E. S. Mohamed, "Case Study: Extenuation of XSS Attacks through Various Detecting and Defending Techniques," Journal of Applied Security Research, 16(1), 91-126, 2021, doi:10.1080/19361610.2020.1735283.

[12] V. S. Stency and N. Mohanasundaram, "A Study on XSS Attacks: Intelligent Detection Methods," Journal of Physics: Conference Series, 1767(1), 012047, 2021, doi:10.1088/1742-6596/1767/1/012047.

[13] G. S. Rani, S. Sarika and P. Rupa,"A study of prevention and detection analysis of SQL injection attack," Seventh International Symposium On Negative Ions, Beams And Sources (Nibs 2020), 2021, doi:10.1063/5.0059318.

[14] B. Gogoi, T. Ahmed and H. K. Saikia,"Detection of XSS Attacks in Web Applications: A Machine Learning Approach," International Journal of Innovative Research in Computer Science & Technology, 9(1), 1-10, 2021, doi:10.21276/ijircst.2021.9.1.1.

[15] A. Kumar, A. Gupta, P. Mittal, P. K. Gupta and S. Varghese, "Prevention of XSS attack using Cryptography & API integration with Web Security," SSRN Electronic Journal, 2021, doi:10.2139/ssrn.3833910.

[16] P. Wang, J. Bangert and C. Kern, C,"If It's Not Secure, It Should Not Compile: Preventing DOM-Based XSS in Large-Scale Web Development with API Hardening," IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021, doi:10.1109/icse43902.2021.00123.

[17] M. Ivanova and A. Rozeva, "Detection of XSS Attack and Defense of REST Web Service – Machine Learning Perspective," The 5th International Conference on Machine Learning and Soft Computing, 2021, doi:10.1145/3453800.3453805.

[18] I. O. Ayo, W. T. Abasi, M. Adebiyi and O. Alagbe," An implementation of real-time detection of cross-site scripting attacks on cloud-based web applications using deep learning," Bulletin of Electrical Engineering and Informatics, 10, 2442~2453, 2021, doi: 10.11591/eei.v10i5.3168.

[19] H. C. Chen, A. Nshimiyimana, C. Damarjati and P. H. Chang, "Detection and Prevention of Cross-site Scripting Attack with Combined Approaches," 2021 International Conference on Electronics, Information, and Communication (ICEIC), 2021, pp. 1-4, doi: 10.1109/ICEIC51217.2021.9369796.

[20] H. Maurel, S. Vidal, and T. Rezk, "Statically Identifying XSS using Deep Learning," SECRYPT 2021 - 18th International Conference on Security and Cryptography, Jul 2021, Virtual, France.

[21] S. Gupta and B. Gupta, "Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment," Procedia Technology, vol. 24, pp. 1595–1602, 2016.

[22] T. Saoji, T. H. Austin, and C. Flanagan, "Using Precise Taint Tracking for Auto-sanitization," Proceedings of the 2017 Workshop on Programming Languages and Analysis for Security, 2017.

[23] D. Dembla, Y. Chaba, K. Yadav, M. Chaba, and A. Kumar, "A Novel And Efficient Technique For Prevention Of Xss Attacks Using Knapsack Based Cryptography," Advances in Mathematics: Scientific Journal, vol. 9, no. 7, pp. 4513–4521, 2020.

[24] J. Shanmugam and M. Ponnavaikko, "A solution to block Cross Site Scripting Vulnerabilities based on Service Oriented Architecture," 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), 2007.

[25] L. Hermerschmidt, S. Kugelmann, and B. Rumpe, "Towards More Security in Data Exchange: Defining Unparsers with Context-Sensitive Encoders for Context-Free Grammars," 2015 IEEE Security and Privacy Workshops, 2015.

[26] R. Wang, G. Xu, X. Zeng, X. Li, and Z. Feng, "TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting," Journal of Parallel and Distributed Computing, vol. 118, pp. 100–106, 2018.

[27] O. C. Abikoye, A. Abubakar, A. H. Dokoro, O. N. Akande, and A. A. Kayode, "A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm," EURASIP Journal on Information Security, vol. 2020, no. 1, 2020.

[28] M. K. Gupta, M. C. Govil, G. Singh, and P. Sharma, "XSSDM: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015.

[29] D. Guamán, F. Guamán, D. Jaramillo, and R. Correa, "Implementation of Techniques, Standards and Safety Recommendations to Prevent XSS and SQL Injection Attacks in Java EE RESTful Applications," New Advances in Information Systems and Technologies Advances in Intelligent Systems and Computing, pp. 691–706, 2016.

[30] H. Shahriar and M. Zulkernine, "S2XS2: A Server Side Approach to Automatically Detect XSS Attacks," 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011.

[31] H. Chen, J. Sun, Q. Zhang, and K. Mao, "An Execution-flow Based Method for Detecting Cross-Site Scripting of Ajax Applications," International Journal of Advancements in Computing Technology, vol. 2, no. 4, pp. 67–76, 2010.

[32] W. Xiao, J. Sun, H. Chen, and X. Xu, "Preventing Client Side XSS with Rewrite Based Dynamic Information Flow," 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming, 2014.

[33] T. S. Barhoom and S. N. Kohail, "A new server-side solution for detecting Cross Site Scripting attack," 2011 International Journal of Computer Information Systems, Vol. 3, No. 2, 2011.

[34] P. Bisht and V. N. Venkatakrishnan, "XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks," Detection of Intrusions and Malware, and Vulnerability Assessment Lecture Notes in Computer Science, pp. 23–43.

[35] B. Mewara, S. Bairwa, J. Gajrani, and V. Jain, "Enhanced browser defense for reflected Cross-Site Scripting," Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014.

[36] J. A. Caliwag, R. A. Pagaduan, R. E. Castillo, and W. V. J. Ramos, "Integrating the Escaping Technique in Preventing Cross Site Scripting in an Online Inventory System," Proceedings of the 2019 2nd International Conference on Information Science and Systems, 2019.
[37] S. Maurya, "Positive security model based server-side solution for prevention of cross-site scripting attacks," 2015 Annual IEEE India Conference (INDICON), 2015.
[38] S. 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, vol. 41, no. 3, pp. 897–920, 2015.
[39] S. C. V. and S. Selvakumar, "Bixsan," ACM SIGSOFT Software Engineering Notes, vol. 36, no. 5, pp. 1–7, 2011.
[40] P. Saxena, S. Hanna, P. Poosankam, and D. Song, "FLAX: Systematic Discovery of Client-side Validation Vulnerabilities in Rich Web Applications," Proc. of the 17th Annual Network and Distributed System Security Symposium (NDSS), 2010.
[41] P. Wurzinger, C. Platzer, C. Ludl, E. Kirda, and C. Kruegel, "SWAP: Mitigating XSS attacks using a reverse proxy," 2009 ICSE Workshop on Software Engineering for Secure Systems, 2009.
[42] J. Pan and X. Mao, "DomXssMicro: A Micro Benchmark for Evaluating DOM-Based Cross-Site Scripting Detection," 2016 IEEE Trustcom/BigDataSE/ISPA, 2016.
[43] S. Gupta and B. Gupta, "Automated Discovery of JavaScript Code Injection Attacks in PHP Web Applications," Procedia Computer Science, vol. 78, pp. 82–87, 2016.
[44] S. Gupta and B. Gupta, "CSSXC: Context-sensitive Sanitization Framework for Web Applications against XSS Vulnerabilities in Cloud Environments," Procedia Computer Science, vol. 85, pp. 198–205, 2016.
[45] J. Weinberger, P. Saxena, D. Akhawe, M. Finifter, R. Shin, and D. Song, "A Systematic Analysis of XSS Sanitization in Web Application Frameworks," Computer Security – ESORICS 2011 Lecture Notes in Computer Science, pp. 150–171, 2011.
[46] A. Shrivastava, S. Choudhary, and A. Kumar, "XSS vulnerability assessment and prevention in web application," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016.
[47] Y. Cao, V. Yegneswaran, P. Porras, and Y. Chen, "Poster," Proceedings of the 18th ACM conference on Computer and communications security - CCS 11, 2011.
[48] S. Stamm, B. Sterne, and G. Markham, "Reining in the web with content security policy," Proceedings of the 19th international conference on World wide web - WWW 10, 2010.
[49] M. V. Gundy and H. Chen, "Noncespaces: Using randomization to defeat cross-site scripting attacks," Computers & Security, vol. 31, no. 4, pp. 612–628, 2012.
[50] P. Agten, S. V. Acker, Y. Brondsema, P. H. Phung, L. Desmet, and F. Piessens, "JSand," Proceedings of the 28th Annual Computer Security Applications Conference on - ACSAC 12, 2012.
[51] H. Shahriar and M. Zulkernine, "Injecting Comments to Detect JavaScript Code Injection Attacks," 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, 2011.