The ISC Int'l Journal of Information Security

November 2021, Volume 13, Number 3 (pp. 21–30)

http://www.isecure-journal.org

Selected Paper at the ICCMIT'21 in Athens, Greece

Cross Site Scripting Attack Review **

Afnan Alotaibi [1], Lujain Alghufaili [1], and Dina M. Ibrahim [1,2,*]

[1] *Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia.*

[2] *Computers and Control Engineering Department, Faculty of Engineering, Tanta University, Tanta, Egypt.*

ARTICLE INFO.

Keywords:
Cross-Site Scripting, Cyber Security, Vulnerability Detection

Type: Research Article

doi: 10.22042/ISECURE.2022.0.0.0

dor: 20.1001.1.20082045.2021.13.3.3.0

ABSTRACT

In the present time, web applications are growing constantly in the whole society with the development of communication technology. Since the utilization of WWW (World Wide Web) expanded and increased since it provides many services, such as sharing data, staying connected, and other services. As a consequence, these numerous numbers of web application users are susceptible to cybersecurity breaches to steal sensitive information or crash the users' systems, etc. Particularly, the most common vulnerability today in web applications is the Cross-Site Scripting (XSS) attack. Furthermore, online cyber attacks utilizing cross-site scripting were responsible for 40% of the attack instances that struck enterprises in North America and Europe in 2019. Therefore, cross-site scripting is a form of an injection that targets both vulnerable and non-vulnerable websites, for the injection of malicious scripts. Cross-site scripting XSS operates by directing users to a vulnerable website that contains malicious JavaScript. Then, when malicious code runs in a victim's browser, the attacker has complete control over how they interact with the application. To protect the website or prevent the XSS, must know the application complexity and the way it handles data must be known so it could be controlled by the user. However, Detecting XSS effectively is still a work in progress, and XSS is considered a gateway for various attacks. However, in this paper, we will introduce the XSS attack and the forms of XSS as a review paper. In addition, the methods and techniques that help to detect cross-site scripting (XSS) attacks.

1 Introduction

Nowadays, web applications have become prevalent and covered a lot of aspects with the growth of the entire world. People count on the web applications to achieve their daily tasks be it personal, medical, business or else. This massive use of web applications renders it vulnerable to attacks such as Cross-Site Scripting XSS. XSS considers the top dangerous at-

* Corresponding author.

**The ICCMIT'21 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: 421200110@qu.edu.sa, 421216187@qu.edu.sa, d.hussein@qu.edu.sa, dina.mahmoud@f-eng.tanta.edu.eg

ISSN: 2008-2045

tack in today's web applications since it's like a gateway to a huge number of attacks that have a high privilege of being defaced and destroyed. Based on Cisco's findings in 2018 indicate that all web applications examined had at least one vulnerability, then these vulnerabilities could be used to launch an attack and that will result in greater damage and costs.

Consequently, 40 percent of all attack attempts refer to the XSS since it's the most frequently used technique [1]. An XSS vulnerability is a type of vulnerability that can threaten web applications by inserting malicious code to gain access to sensitive information without the user's consent such as cookies, passwords, credit card numbers, etc. XSS accomplishes these attacks when this script is injected in such a way that the website appears to be error-free without costing any damage to the users and this script is eventually implemented inside the confident domain of the website.

Then the attacker designs and injects a malicious JavaScript payload into the web application for the exploitation of XSS vulnerabilities on web applications. These vulnerabilities are primarily triggered by user inputs not being properly sorted. Fishing attacks and typo squatting caused by exploiting XSS vulnerabilities bring tremendous losses to companies based on what the Internet security threat announced in 2019 [2]. XSS is a web application attack that displays dynamic content to users without verifying or encoding the information they provide. Attackers can employ XSS to send a malicious script that the user's browser considers trustworthy. This malicious script can access any cookies, session tokens, and sensitive information stored by your browser, as well as change the HTML page's content [3].

When A large number of URLs containing malicious code were generated by the attacker and tempt users to press. Hence The attacker might get cookies from users when they clicked on these URLs and use these cookies to log in to their accounts. Moreover, the expression "Cross-site scripting" was introduced in January 2000 by Microsoft security engineers and its first appearance was in the 1990s [4]. Our paper organized as follow, introduction about XSS attacks presented in Section 1. Section 2, related work of XSS attack with some techniques used to detect or prevent it. Section 3, demonstrate the various types of XSS attacks. Section 4, mentioned some solation to detect XSS attack. Finally, we concluded our paper in Section 5.

2 Related Work

The security component is an important aspect that plays an important role in online applications, especially with the increased use of it these days. However, web applications that are linked to web servers provide users with several features that may expose them to security risks. In this section, we'll look at some research on XSS and how to avoid exploiting vulnerabilities to compromise the information of web application users.

The goal of the researchers in this study [5] is to present the results of a thorough literature review by consulting related works on XSS attacks and vulnerabilities. Their methodology includes reviewing approximately 115 studies on XSS using the Barbara Kitchenham document with standard principles for systematic literature evaluation. They concentrated on all studies published since 2000, as well as the methodology, strategies, algorithms, and tools that each study offered.

The authors of study [6] advocated that the Genetic Algorithm (GA) be used with Reinforcement Learning (RL) and threat intelligence to prevent XSS attacks. The purpose of combining GA and RL is to optimize their approach so that they can adapt to new XSS payloads. They also use the GA in conjunction with statistical inference and RL to detect XSS assaults. They also contributed to the creation of a dynamic cross-over rate. To test their model, they used NetBeans IDE 8.0.2 in Java 1.8, GitHub2, and xssed3. They collected 50,540 vulnerable payloads and 1,35,507 normal records using NetBeans IDE 8.0.2 in Java 1.8, GitHub2, and xssed3. 6503 normal records and 3497 susceptible records were tested in their dataset. The results of their proposed approach demonstrate improved XSS detection accuracy and performance.

The study [7] is containing a comprehensive report of different approaches to XSS attack detection. They divided the methodologies into three categories: client-server detection, client-side detection, and server-side detection. Furthermore, the writers confirm the XSS kinds as well as the benefits and drawbacks of each strategy. Their poll includes a list of technologies that can be used to protect against XSS attacks, as well as concerns and obstacles related to XSS. Their methodology for comparing the previous surveys and providing the previous methods in a manner that cover the previous surveys' flaws, such as:

- They followed the methods in the review [5], as well as, classified the defense methods into several analysis mechanisms and grouped them in defense system under deployment site. Furthermore, they provide a discussion of the needful preconditions to begin an XSS attack.
- The authors in [8] are supplying their study with a highly summarized insight of analysis

mechanisms, unlike; the study [7] is providing an extensive review of each category of analysis mechanisms. Moreover, the study [8] categorizes Web application vulnerabilities, depending on the software security approaches. On the other hand, their survey [7] is categorized depending on detection mechanisms under the deployment sites.

- They are following the way the study [9] in notifying the vulnerabilities existing in some techniques for fixing the XSS problem, and they do not follow the [9] in defense methods categorize i.e. they categorize in deployment sites and collect them in analysis mechanisms.

In their study [10], the authors combine static analysis with additional approaches to improve XSS vulnerability detection in PHP web applications. They want to improve the detection system to protect the web application from XSS attacks. To find the efficient identification of XSS problems in PHP web applications, their methodology compares the methods that were previously enhanced by integrating static analysis with other algorithms such as Genetic algorithm (GA). The study's findings include improved detection results and static analysis run time. But after combining the static analysis with algorithms, still a false positive rate in results, which is a static analysis problem.

They discovered the effectiveness of a method that combines static analysis and a genetic algorithm to detect XSS vulnerabilities, with no false positives. However, because the approach is still manual, it is not suitable for large online applications.

The authors of [11] present a strategy for using intrusion detection systems to detect cross-site scripting (XSS) assaults. SNORT IDS was selected by the researchers since it is free, open-source, and lightweight. By installing SNORT IDS on the network, this approach can detect XSS attacks. This technology works by watching network traffic and issuing alerts when the IDS detects any irregular packets. According to the authors, this technique has several limitations, although it is generally effective in detecting XSS assaults. Another strategy for detecting XSS attacks is presented in [12], which is based on the attention mechanism of the Long Short-Term Memory (LSTM) recurrent neural network. There are certain steps in this model. The data must first be processed, which includes data encoding. Then they employ word2vec to elicit XSS payload characteristics. Finally, the LSTM-Attention detection model was used to discriminate XSS samples from normal samples. The detection model for XSS assaults is shown in Figure 1.

Moreover, they concluded their work with the success of LSTM since the experiments had shown that this model had effectiveness to detect XSS attacks. Also, it had some limitations for instance that this approach needs to improve accuracy and performance of classification and it will be addressed in their future work.

In [13] the authors used PHP's functions to prevent XSS attacks and test the efficacy of web pages and provide adequate information for both web developers and consumers when creating and using websites to prevent specific attacks. The PHP will do two tasks prevent and detect. For the detection PHP will first check data that have been entered by the users to a web application. Then for the preventing PHP will validate every input that the users entered and be sure it's have not any malicious injections, if the inputs have any malicious codes the PHP will immediately remove them and prevent them from executing. This technique will be improved in the future to validate and prevent XSS attacks directly.

In [14] they present a proposed a dynamic detection framework (TT-XSS) for DOM-XSS to detects vulnerabilities automatically. Therefore, this framework of dynamic DOM-XSS detection consists from three base modules: URL information collection, taint tracking analysis and automatic vulnerability verification as shown in Figure 2

Moreover, this framework is very important for developers to identify and fix vulnerabilities effectively since it can detect more than 1.8 percent of vulnerabilities. In addition, the authors mentioned some limitations of the TT-XSS framework. The TT-XSS cannot handle two-order inputs and sometimes when the inputs are complex it takes a long time. These limitations will be addressed in their future work.

The authors of [14] advocated using the open-source browser WebKit to combat XSS attacks via a web browser that relies on machine learning classification. Furthermore, this browser divides Web pages into harmful and non-malicious categories. They improved a web browser prototype before embedding it in the web browser to improve the classification module. According to scientists, machine learning classification algorithms can effectively detect and prevent XSS and new assaults. Furthermore, this strategy will help not only the academics but also the users. Finally, the findings show that the Acation module created and supplied a secure online application for users to visit Table 1 that illustrates a summary of the previous studies.

3 XSS Attacks

Incontrovertibly, an XSS attack occurs when the attacker injects malicious code into the web page. On

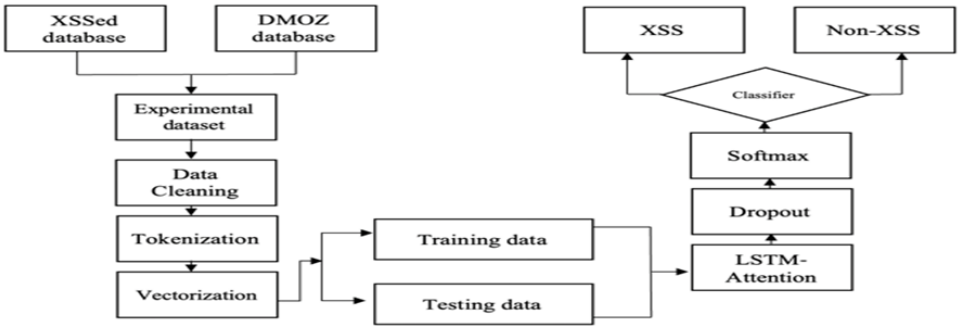

Figure 1. The LSTM model steps [12]

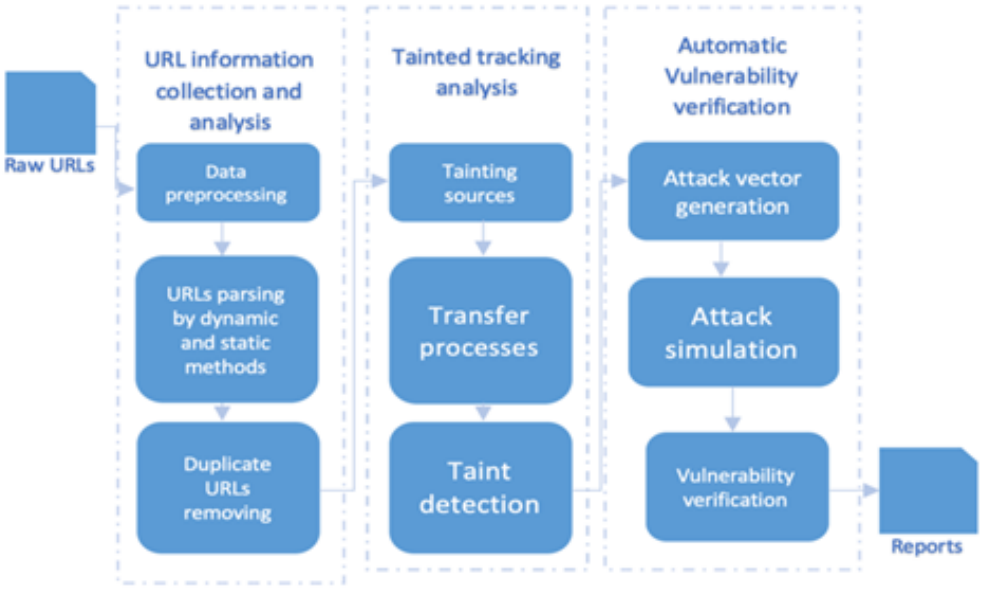

Figure 2. The dynamic TT-XSS process [14]

the other hand, the user visits the same web page and is either asked for information or, in the worst-case scenario, enters confidential information such as a password to a fake website. Admittedly, the lack of verifying user inputs is a key factor in XSS attacks.

Furthermore, this section explains the various forms of XSS attacks that have had a negative impact on several aspects. correspondingly, it could be any kind of attack like phishing, URL hijacking, or ransomware these attacks had their impact on social media apps for instance Twitter, YouTube, etc. Moreover, there are two major types of XSS attacks which as [16].

3.1 Non-persistent XSS attack

This type is known as reflective attack since it includes the reflected action from the web server when the used demand for certain services such as the results of the search process, a reflected message of the server, or another response that contains any or all of the data sent to the server Figure 3 demonstrates this attack.

In addition, for the website that is vulnerable to XSS attack, it already implements two conditions. First, accept the script injection by the website. Second, the website had two users to communicate with.

3.2 Persistent XSS Attack

This attack interacts with web pages rather than simply representing a result, as in the non-persistent attack. Furthermore, this attack is one in which the injection script is an inevitability on the server's databases in various ways like comment fields, logs, forums, and so on [16]. Then the victim demands the stored information which relatively contains injected script [17]. In addition, it is known as a stored attack, as illustrated in Figure 4.

3.3 DOM-based (Document Object Model)

It is a common and dangerous form of XSS attack. Thereupon, the DOM (Document Object Model) based attack that accomplished by insecure data flows

Table 1. A summary of the previous studies

Ref.	Year	Study	Objective	Results
[5]	2015	systematic literature review of XSS	cover XSS attack topics	Information tables of XSS previous studies
[11]	2017	An Approach to detect Cross-site scripting (XSS) attacks via intrusion detection systems	Detecting XSS attacks	This approach had some flaws, but in general, it's efficient enough to detect XSS attacks.
[7]	2018	A comprehensive survey of different approaches of XSS	Find ways protect from the XSS attack	Helpful survey of XSS attack
[13]	2018	Proposed to used PHP's functions	Prevent and detect XSS attack	This technique needs to be improved to detect and prevent XSS attacks directly.
[14]	2018	Dynamic detection framework (TT-XSS)	Detects vulnerabilities automatically	This technique can identify and fix vulnerabilities in an effective way but cannot handle two-order inputs and takes a long time.
[10]	2019	An Investigation on Static Analysis with other Algorithms to Detect Cross-Site Scripting.	improving the detection method of XSS	Are reinforced detection results and the run time. The false-positive rate in results.
[12]	2020	LSTM-Attention detection model	Provide XSS attacks detection	The experiments had shown that this model had effectiveness to detect XSS attacks. But it needs to improve the accuracy and performance of classification.
[6]	2021	Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning	Prohibit the XSS attacks	Better accuracy in XSS's detection. Better performance.
[15]	2021	Machine learning classification techniques	Mitigate XSS	This technique can detect and mitigate XSS. Also had developed and provided a secure web application to browse.

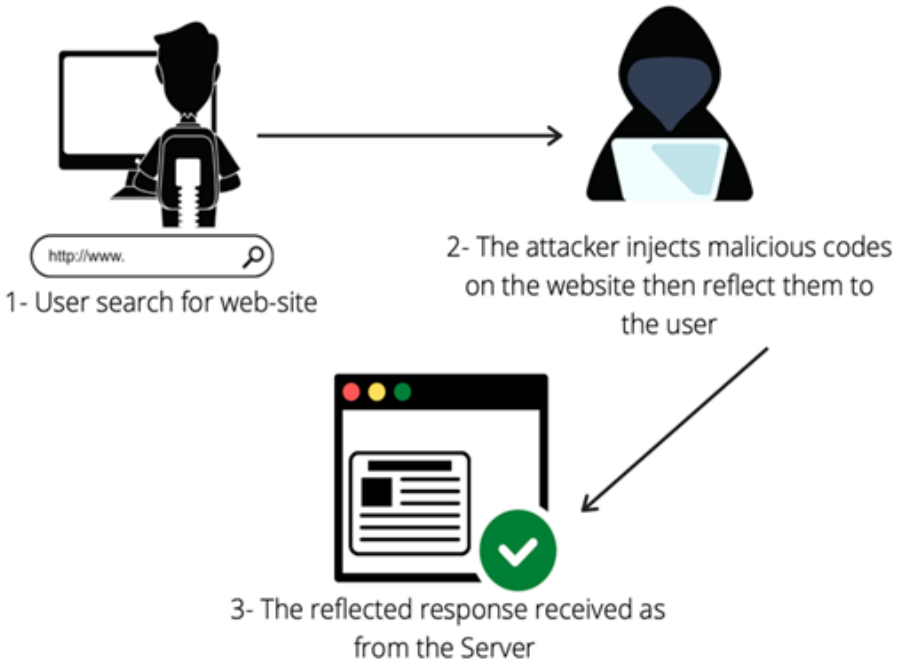

Figure 3. Non-persistent XSS attack

Figure 4. Persistent XSS attack

from sources controlled by the attacker for instance URLs and type inputs [18]. Moreover, according to a recent survey, roughly ten percent of the Alexa Top 5000 websites contain at least one DOM-based XSS vulnerability [19]. This attack is demonstrated in Figure 5.

In [20] the authors proposed an approach to prevent persistent XSS attacks by implementing a pattern filtering method. As known, the success of this attack is contingent on the attacker injecting malicious code into the website's database. For this reason, this approach is based on checking data before storing them in the website's database in a unique way that is not performed by a browser when it is sent to it.

Furthermore, there is no guarantee that the proposed approach will continue to function effectively in the future. Generally speaking, the most effective way to avoid or protect against an XSS attack is using coding for web applications with escaping mechanisms in the appropriate locations. Since the DOM-based attack, has solutions that were insufficient to manage it like using a single method, the developers of [21] used a hierarchical approach during the development process to protect the web application on both the client and server sides.

Moreover, we should have to use multilayer protection to achieve a stable system that is difficult to hack and to prevent DOM-based attacks. For this reason, the authors suggested using a hierarchical approach with multilabel levels.

Table 2 illustrates the three types of XSS attacks and some proposed approaches to address them. Furthermore, TT-XSS and LSTM approaches had been presented in the related work section in a detailed way and can be used to detect all XSS attacks. Moreover, the prevention methods had been clarified in this section.

4 First XSS Vulnerability Detection

As this study mentioned before, the XSS is attacking by inserting malicious code inside web applications to gain access to sensitive information without the user's consent such as cookies, passwords, credit card numbers, etc. Correspondingly, the word "vulnerabilities detection "in XSS attacks is referred to as examining flaws in the source code of the web application.

Nevertheless, the purport of this section involves using source code analysis to detect vulnerabilities either before an application is implemented or before an attacker discovers the vulnerability. On the foundation of the kind of information they use from the web application, vulnerability analysis tools can be divided into three categories: dynamic, static, and hybrid as Figure 6 shown.

4.1 Static Analysis

A static analysis tool looks at the source code of a web application to find vulnerabilities in the code. Through analyzing the source code of the web application, a static analysis tool will detect any possible software path along with the application. As a result, the static analysis will find any vulnerability in any program route. However, the most significant limitation of static analysis is the high prevalence of false positives in report outcomes, which continues to make static analysis results erroneous and have an impact on vulnerability detection final results. False-positive detection occurs when particular paths are identified as insecure while they are truly secure and free of vulnerabilities [10].

Various research has examined static analysis on client-side attacks, such as Gupta and Gupta's (2016a) study, which offered an XSS immune defensive architecture that is considered a Google Chrome browser extension. The XSS framework is built on comparing JavasScript strings and sanitizing them while keeping the context in mind [7]. When server-side vulnerabilities are created, XSS attacks occur. As a result,

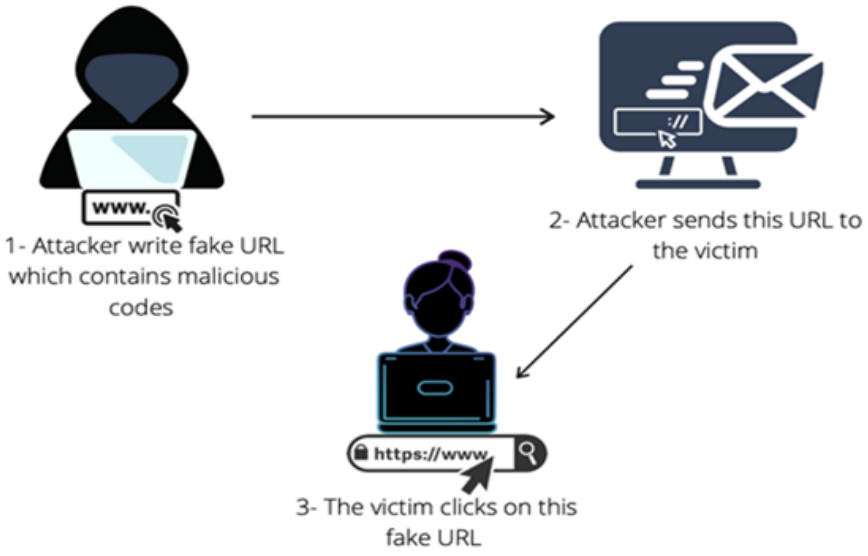

Figure 5. The Document Object Model (DOM) attack

Table 2. XSS attacks with detection and prevention methods

XSS attacks	Detection Method	Ref.	Prevention Method	Ref.
Non-persistent XSS attack	-TT-XSS - LSTM	[14] [12]	--	--
Persistent XSS attack			Pattern filtering method	[21]
DOM-based XSS attack			Hierarchical approach	[22]

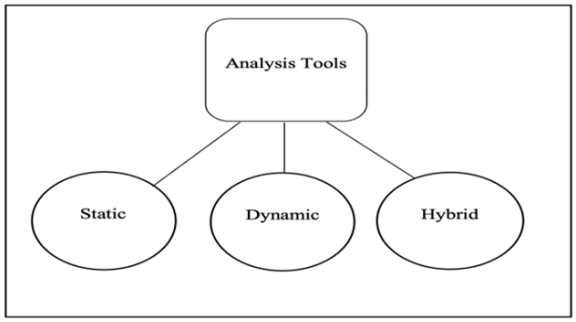

Figure 6. Categories of analysis tools

protection methods to defend the server from such attacks are being created. One of these approaches is Scott and Sharp's (2002) structuring technique, which creates security policies for huge web applications that have evolved in a variety of environments [22].

SPDL, or Security Policy Description Language, is used to code this method, which allows for explicit validation constraints and transformation rules. Minamide (2005) presented a PHP string analyzer to statically evaluate dynamically generated server application web pages. The PHP analyzer takes two main inputs: a PHP program and a list of all possible PHP inputs [23].

4.2 Dynamic Analysis

Instead of communicating with the source code as a user does with a web browser, interactive tools communicate with the web application under investigation. The benefit of dynamic analysis methods over static analysis is that they generate fewer false positives. The fact that the fuzzing attempt tries to stimulate the weakness causes this [10].

The focus of dynamic analysis mechanisms is on runtime behavior or any activity that isn't immediately attributed to the program. The executable code of the application, on the other hand, is examined for security issues. Dynamic analysis processes are thought to be more precise in detecting vulnerabilities and produce fewer false positives. A proxy server-based strategy, which was upgraded by Ismail *et al.* (2009) to secure the user's private information while using the web, is one of the dynamic analysis mechanisms in detecting vulnerabilities from the client-side. The developers wanted to find and gather XSS issues in the application automatically [24].

Furthermore, this method detects vulnerability using two major modes: Request change mode and Respond change mode. Mitropoulos *et al.* (2016) pro-

posed a security solution that combines the web browser's JavaScript engine with a Script interception layer to defend the application from malicious JavaScript XSS attacks. This layer's goal is to identify any upcoming Script from any possible track to the Web browser. Later, the entering Script is compared to a list of whitelisted scripts that are legitimate and valid [25].

Dynamic analysis tools, such as Wurzinger *et al.* SWAP's methodology (Secure Web Application Proxy), actively analyze the program states to find vulnerabilities in the application (2009). SWAP is made up of two components: a reverse proxy that handles all requests and responses between the web server and its clients, and a JavaScript detection element that is installed on the reverse proxy and adjusts the web browser to recognize script content in the server's responses. DOM. Shahriar and Zulkernine (2011), on the other hand, introduce S2 XS2 to detect XSS attacks automatically. The server-side application code employs boundaries before and after in each area that may keep or generate dynamic content, and this technique is primarily based on two concepts: boundary injection and policy creation. Each boundary could indicate the intended content character [26].

4.3 Hybrid Analysis

As the name implies, hybrid analysis approaches incorporate the benefits of both dynamic and static analysis. Static analysis is performed to identify potential vulnerabilities, which is followed by a confirmation stage in which the tool is permitted to try to exploit the flaw. The tool will only reveal the vulnerability if this stage is successful.

On the other hand, hybrid analysis inherits the problem of static analysis, which is a high rate of false positives in the results. As a result, this method is employed less frequently than static and dynamic analysis methods. The static analysis looks at the code of an application without running it, identifying security issues. As a result, there is no run-time overhead, and 100 percent code coverage is feasible because it can analyze all alternative execution paths (unlike testing where code coverage is a popular problem). It's also valuable since it may be applied early in the software development lifecycle, even if only a small piece of the code is useable [10].

Static analysis and dynamic data tainting are proposed by Vogt *et al.* (2007) as a defensive technique against XSS assaults. This method is based on tracing or flagging sensitive information on the client-side. Without the user's approval, no private information will be utilized or shared with an untrustworthy third party, as expected. The Dynamic Data Tainting technique does this by tainting the user's sensitive information and then tracing it dynamically anytime it is accessed by any script [27]. Curtsinger *et al.* (2011), on the other hand, propose ZOZZLE, a classifier-based JavaScript deobfuscator. This method can be used to identify and defeat XSS attacks in a browser. furthermore, an approach that uses an Abstract Syntax Tree (AST) to tell the difference between malicious and benign JavaScript code. This technique employs hierarchical-context sensitive information for detection [28].

Ben Jaballah and Kheir (2016) proposed a Grey box approach for detecting viral interactions with web applications. This strategy combines the advantages of both white box and black box thinking. White box testing techniques can detect critical problems in online applications, but they are unable to handle logical flaws. While black-box testing methodologies analyze the development of the application, they have a significant false-positive rate [29].

Song *et al.* (2009) also offers the Spectogram, a statical anomaly detection network-based sensor that tries to detect anomalies in online traffic. It defeats an XSS attack by using genuine data instead of harmful input. It analyses and isolates the scripts provided in the HTTP request parameters, then creates the script's architecture and content based on these parameters [30]. Static analysis, according to many developers, is the most effective way for finding bugs in a web application. In terms of discovering problems, Microsoft estimates that code review is 20 to 30 times more effective than software testing. Furthermore, when used correctly, it will reveal nearly half of the flaws already present. To improve static analysis performance, researchers began integrating static analysis with different techniques.

5 Conclusion

To sum up, what has been stated earlier, Cross-Site Scripting (XSS) vulnerability is considered to be one of the most prevailing vulnerabilities in the web application which eventually can cause violations or damages to the user or site. Cross-site Scripting (XSS) is a type of code injection attack that occurs on the client side.

By embedding malicious code in a legitimate web application, the attacker plans to perform malicious scripts in the victim's web browser. Moreover, this attack has various forms which have had a negative effect on several factors of web applications. The major forms of XSS are non-persistent XSS attacks and persistent XSS attacks. For the vulnerability detection of XSS through the vulnerability analysis

categories which are: static analysis, dynamic analysis, and hybrid analysis.

References

[1] Germán E Rodríguez, Jenny G Torres, Pamela Flores, and Diego E Benavides. Cross-site scripting (xss) attacks and mitigation: A survey. *Computer Networks*, 166:106960, 2020.

[2] Ibrahim Nadir and Taimur Bakhshi. Contemporary cybercrime: A taxonomy of ransomware threats & mitigation techniques. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–7. IEEE, 2018.

[3] Nadya ElBachir El Moussaid and Ahmed Toumanari. Web application attacks detection: A survey and classification. *International Journal of Computer Applications*, 103(12), 2014.

[4] Miao Liu, Boyu Zhang, Wenbin Chen, and Xunlai Zhang. A survey of exploitation and detection methods of xss vulnerabilities. *IEEE access*, 7:182004–182016, 2019.

[5] Isatou Hydara, Abu Bakar Md Sultan, Hazura Zulzalil, and Novia Admodisastro. Current state of research on cross-site scripting (xss)–a systematic literature review. *Information and Software Technology*, 58:170–186, 2015.

[6] Iram Tariq, Muddassar Azam Sindhu, Rabeeh Ayaz Abbasi, Akmal Saeed Khattak, Onaiza Maqbool, and Ghazanfar Farooq Siddiqui. Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning. *Expert Systems with Applications*, 168:114386, 2021.

[7] Upasana Sarmah, DK Bhattacharyya, and Jugal K Kalita. A survey of detection methods for xss attacks. *Journal of Network and Computer Applications*, 118:113–143, 2018.

[8] Mukesh Kumar Gupta, MC Govil, and Girdhari Singh. Static analysis approaches to detect sql injection and cross site scripting vulnerabilities in web applications: A survey. In *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, pages 1–5. IEEE, 2014.

[9] Rahul Johari and Pankaj Sharma. A survey on web application vulnerabilities (sqlia, xss) exploitation and security engine for sql injection. In *2012 international conference on communication systems and network technologies*, pages 453–458. IEEE, 2012.

[10] Abdalla Wasef Marashdih, Zarul Fitri Zaaba, Khaled Suwais, and Nur Azimah Mohd. Web application security: An investigation on static analysis with other algorithms to detect cross site scripting. *Procedia Computer Science*, 161:1173–1181, 2019.

[11] Kunal Gupta, Rajni Ranjan Singh, and Manish Dixit. Cross site scripting (xss) attack detection using intrustion detection system. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 199–203. IEEE, 2017.

[12] Li Lei, Ming Chen, Chengwan He, and Duojiao Li. Xss detection technology based on lstm-attention. In *2020 5th International Conference on Control, Robotics and Cybernetics (CRC)*, pages 175–180. IEEE, 2020.

[13] Twana Assad Taha and Murat Karabatak. A proposed approach for preventing cross-site scripting. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–4. IEEE, 2018.

[14] Ran Wang, Guangquan Xu, Xianjiao Zeng, Xiaohong Li, and Zhiyong Feng. Tt-xss: A novel taint tracking based dynamic detection framework for dom cross-site scripting. *Journal of Parallel and Distributed Computing*, 118:100–106, 2018.

[15] Vikas K Malviya, Sawan Rai, and Atul Gupta. Development of web browser prototype with embedded classification capability for mitigating cross-site scripting attacks. *Applied Soft Computing*, 102:106873, 2021.

[16] Mehul Singh, Prabhishek Singh, and Pramod Kumar. An analytical study on cross-site scripting. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–6. IEEE, 2020.

[17] Shaimaa Khalifa Mahmoud, Marco Alfonse, Mohamed Ismail Roushdy, and Abdel-Badeeh M Salem. A comparative analysis of cross site scripting (xss) detecting and defensive techniques. In *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 36–42. IEEE, 2017.

[18] Jinkun Pan and Xiaoguang Mao. Detecting dom-sourced cross-site scripting in browser extensions. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 24–34. IEEE, 2017.

[19] Sebastian Lekies, Ben Stock, and Martin Johns. 25 million flows later: large-scale detection of dom-based xss. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1193–1204, 2013.

[20] Imran Yusof and Al-Sakib Khan Pathan. Preventing persistent cross-site scripting (xss) attack by applying pattern filtering approach. In *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, pages 1–6. IEEE, 2014.

[21] Ankit Shrivastava, Santosh Choudhary, and

Ashish Kumar. Xss vulnerability assessment and prevention in web application. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 850–853. IEEE, 2016.

[22] David Scott and Richard Sharp. Abstracting application-level web security. In *Proceedings of the 11th international conference on World Wide Web*, pages 396–407, 2002.

[23] Yasuhiko Minamide. Static approximation of dynamically generated web pages. In *Proceedings of the 14th international conference on World Wide Web*, pages 432–441, 2005.

[24] Peter Wurzinger, Christian Platzer, Christian Ludl, Engin Kirda, and Christopher Kruegel. Swap: Mitigating xss attacks using a reverse proxy. In *2009 ICSE Workshop on Software Engineering for Secure Systems*, pages 33–39. IEEE, 2009.

[25] Dimitris Mitropoulos, Konstantinos Stroggylos, Diomidis Spinellis, and Angelos D Keromytis. How to train your browser: Preventing xss attacks using contextual script fingerprints. *ACM Transactions on Privacy and Security (TOPS)*, 19(1):1–31, 2016.

[26] Hossain Shahriar and Mohammad Zulkernine. S2xs2: a server side approach to automatically detect xss attacks. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 7–14. IEEE, 2011.

[27] Philipp Vogt, Florian Nentwich, Nenad Jovanovic, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. In *NDSS*, volume 2007, page 12, 2007.

[28] Charlie Curtsinger, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Zozzle: Fast and precise in-browser javascript malware detection. In *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[29] Wafa Ben Jaballah and Nizar Kheir. A grey-box approach for detecting malicious user interactions in web applications. In *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, pages 1–12, 2016.

[30] Yingbo Song, Angelos D Keromytis, and Salvatore Stolfo. Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. 2009.

Afnan Alotaibi is a graduated student from Computer Science department, Shaqra University, since 2017. She is currently a Master thesis student in cybersecurity program at Department of Information Technology, College of Computer, Qassim University, Buraydah 51941, Saudi Arabia.

Lujain Alghufaili is a graduated student from Computer Science department, Shaqra University, since 2019. She is currently a Master thesis student in cybersecurity program at Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia.

Dina M. Ibrahim is an Assistant Professor at the Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia from September 2015 till now. In addition, Dina works as an Assistant Professor in the Computers and Control Engineering Department, Faculty of Engineering, Tanta University, Egypt. She was born in the United Arab Emirates, and her B.Sc., M.Sc., and Ph.D. degrees have taken from the Computers and Control Engineering Department, Faculty of Engineering, Tanta University in 2002, 2008, and 2014, respectively. Dina works as a Consultant Engineer, then a Database administrator, and finally acts as a Vice Manager on Management Information Systems (MIS) Project, Tanta University, Egypt, from 2008 until 2014. Her research interests include networking, wireless communications, machine learning, security, and the Internet of Things. She is serving as a reviewer in Wireless Network (WINE) the Journal of Mobile Communication, Computation, and Information since 2015, and recently in the International Journal of Supply and Operations Management (IJSOM). Dina has also acted as a Co-Chair of the International Technical Committee for the Middle East Region of the ICCMIT conference since 2020.