National College of Ireland

Enhancing Web Security: Detecting and Preventing Reflected Cross-Site Scripting (XSS) Attacks

MSc Research Project

Cyber Security

Suraj Suprabha Raju

Student ID: 23183462

School of Computing

National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Suraj Suprabha Raju
Student ID:	23183462
Programme:	MSc Cyber Security **Year:** 2024.
Module:	MSc Research Practicum
Supervisor:	Khadija Hafeez
Submission Due Date:	12/08/2024
Project Title:	Enhancing Web Security: Detecting and Preventing Reflected Cross-Site Scripting (XSS) Attacks
Word Count:	……5192…… **Page Count**……19…….……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Suraj Suprabha Raju……

Date: ……12/08/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Web Security: Detecting and Preventing Reflected Cross-Site Scripting (XSS) Attacks

Suraj Suprabha Raju

23183462

Abstract

Cross-Site scripting attacks are a significant threat to web application security. This research focuses on the detection and prevention of reflected XSS attacks through a different approach which makes use of different methods. By implementing security methods such as input filtering, sanitization, output encoding, CSRF protection a framework is developed which is successful in preventing XSS attacks. The proposed model also includes live monitoring which allows for a quick response to detected threats. This study enhances web security by preventing and mitigating the risks posed by XSS attacks to web applications.

Keywords: XSS attacks, web security, Cross-Site Scripting (XSS), CSRF protection, Artificial intelligence (AI)

1 Introduction

Importance of web applications has been growing day by day in this world, because it has more functionality in terms of interaction, communication etc in the online business enterprise. However, this surge in internet software utilization has concurrently caused a boom in the variety and complexity of cyber threats. Among the threats, Cross-Site Scripting (XSS) sticks out as one of the most common and dangerous vulnerabilities. XSS vulnerabilities allow attackers to inject malicious scripts into web pages, primarily to get unauthorized access to sensitive data and session hijacking (Geeksforgeeks, 2023). This thesis focuses on growing a whole method for the detection and prevention of Reflected XSS assaults in internet packages.

1.1 Background

Cross-site scripting is a form of protection vulnerability typically located in net programs. One of the types of Cross-site scripting attack is Reflected XSS, which is the most common kind of attack, while an attacker sends malicious input to an internet software, which right now reflects the input lower back to the individual's browser. The browser then executes the malicious script within the context of the sufferer's session (OWASP, 2021). Reflected XSS allows attackers to inject malicious scripts into the URL of legitimate web servers. These

scripts can play a major role to steal individual records, impersonate customers, and perform distinctive malicious moves.

Some of the recent XSS attacks & vulnerabilities are as follows.

- Nearly 75% of large companies in Europe and North America were hit by online cyber-attacks in 2019 with cross site scripting used in 40% of incidents (Kass, 2020).
- Compromised 65 job sites in Asia in 2023 using XSS attacks by the attackers and they had Stolen personal data of over two million job seekers (Bleepingcomputer, 2024).
- British Airways was faced XSS cyber-attack on 2018, where data breach had affected 380,000 booked transactions (Matteson, 2018)
- Joomla fixes XSS flaws in 2023 that could expose sites to remote code execution (RCE) attacks (Toulas, 2024).
- GitLab patched a high-severity vulnerability in 2024, there was a possibility that unauthenticated attackers could exploit it to take over user accounts (Gatlan, 2024).

1.3 Problem statement

Despite significant advancements in cybersecurity where the process of detecting and mitigating attacks is being made by advanced methods, reflected cross-site scripting (XSS) remains a persistent threat to web applications. Existing methods such as dynamic and static analysis, the use of AI to detect and mitigate XSS attacks have shown a weakness in real time scenarios (Rodríguez et al. 2020). This shows that it lacks precision in identifying these types of attacks. AI and machine learning based approaches have proven to be effective, but the lack of datasets and computational power of certain organizations lead to potential inaccuracies.

This study aims to address these challenges by developing a framework/model for the prevention and detection of reflected XSS attacks through advanced sanitization, output encoding, pattern matching, real time monitoring and other advanced methods. This framework/model will be evaluated for its effectiveness in detecting XSS attacks and it will also focus on minimizing false positives and ensuring that a wide range of potential attack vectors are covered also it will help software developers to build secure web applications.

1.4 Research questions

How can we lift the bar of detection and prevention of such attacks called Cross-Site Scripting that appears in web applications?

What are the detection & prevention strategies to mitigate the impact of Reflected XSS attacks on internet applications?

What strategies and tools can effectively support developers in creating secure web applications?

2 Related work

Authors from (Iman Fareed Khazal et al. 2021) uses a combination of URL Analysis, Sanitization and Server-side Processing for detecting XSS attacks. Here the URL is checked whether it comes from a trusted source or not and the URL is also checked for harmful code which is sterilized before allowing user access. This analysis and sanitization are performed

on the server side to reduce the load on the client's browser thereby enhancing the performance. This study (Tan et al. 2023) makes use of the Paths-Attention Model which uses advanced natural language processing techniques to enhance vulnerability detection. The authors make use of a dataset to detect the XSS vulnerabilities, addressing the issue of imbalance of positive and negative samples. After this the experimental results were analyzed in two parts: semantic extraction results and XSS vulnerability detection results. Authors from (Rodríguez et al. 2020) uses a combination of content analysis, input validation, pattern analysis, web scanners, artificial intelligence methods and cookie analysis. Where they used methods including machine learning techniques such as Support Vector Machines, Random Trees and classifiers to predict and detect vulnerabilities. (Chrisando Ryan Pardomuan et al. 2023) The methods used include Detection System Design, Evaluation Process and Payload Sources. The system was designed to detect XSS attacks rather than prevent them. Then the accuracy of the detection system is evaluated using a confusion matrix which focuses on the false positives and false negatives. Evaluation involved simulating HTTP requests to a vulnerable web application. They used the dataset of 499 requests. Authors from (Santithanmanan, K et. al 2024) used machine learning methods, such as k-Nearest Neighbours(k-NN), Decision Tree, Support Vector Machine (SVM), Gaussian Naive Bayes (GNB). The dataset consisted of both benign and malicious URLs. The models were evaluated using 10-fold cross validation to determine the performance metrics which includes accuracy, precision, recall and F1 score. This paper (Tadhani, J.R. et. al 2023) used different methods to secure XSS attacks on web applications, which includes Deep learning approach, dataset creation, benchmark datasets and comparison with traditional methods.

2.1 Critical Analysis

The existing methods rely heavily on server processing capabilities which leads to delays when the traffic is high. The automatic URL sanitization method used can lead to false positives sometimes which may lead to blocking legitimate users from accessing the content. The current approach focussed on reflected XSS attacks and did not take into account other types like DOM-based vulnerabilities and broader attack vectors. The use of imbalanced datasets affects the accuracy and reliability of the proposed systems. To implement the proposed methods the organizations will require high computational power which may not be feasible for all organizations. The static and dynamic methods used are not effective in real time applications. The proposed AI methods show promise, but the lack of comprehensive datasets may affect the reliability or may not be successful in identifying zero-day vulnerabilities. Most of the proposed models only focus on the detection of the XSS attacks and not preventing them. The non-standard encoding of malicious payloads results in a large number of false negatives.

To overcome the challenges mentioned above I am proposing a comprehensive framework/model for the detection and prevention of reflected XSS attacks. Advanced sanitization and Output encoding is done to reduce the risk of false positives which ensures that legitimate users are not blocked from accessing the content. The model is designed to cover a broader range of vulnerabilities, including reflected XSS attacks and other vulnerabilities. The accuracy and reliability of the model is improved by using balanced and

comprehensive datasets. This model does not need high computational power which makes it feasible for a wide range of organizations. Combining dynamic and static analysis the model ensures better real time detection and reduces the risk of false negatives. The model is optimized for real time monitoring which allows for better detection and mitigation of XSS attacks irrespective of the traffic load.

Paper	**Methods Used**	**Main Results**	**Benefits**	**Limitations**
(Iman Fareed Khazal et al. 2021)	URL Analysis, Sanitization and Server-side Processing for detecting XSS attacks.	The method reduced the risk of XSS attacks by allowing only the sanitized URLs to the users which enhanced the overall web application security.	This method is used where the sanitization is done by the server side thereby improving the browser performance. This solution does not rely on specific server-side practices which makes it versatile among different web applications.	If there is a high load of traffic, there can be some delays from the server side. Potential false positives of automatic sanitization may prevent legitimate users from accessing the content. It only focuses on XSS attacks, it may be vulnerable to other forms of attacks.
(Tan et al. 2023)	Paths-Attention Model, dataset to detect the XSS vulnerabilities.	The processing speed of 1000 samples/second was achieved by the Paths-Attention model. Higher F1 score than other models.	Paths-Attention model Process large number of samples in a very short time. The higher F1 score is evidence that there were not many false positives detected which makes it more accurate and precise.	The study focuses on mainly reflected XSS and does not cover other XSS vulnerabilities. Required larger computational power. The study faced challenges with imbalanced datasets used which could affect the result.
(Rodríguez et al. 2020)	Content analysis, input validation, pattern analysis, web scanners, AI methods and cookie analysis, Support Vector Machines, Random	A lower tendency was observed in the use of AI methods despite their potential for enhancing detection capabilities. This shows that the traditional methods to detect vulnerabilities are more commonly employed than artificial	The input validation ensures no invalid data is processed. Pattern analysis proves to be effective in identifying XSS attacks of the same pattern. Automated and comprehensive scanning capabilities of web scanners makes it easier to identify vulnerabilities across large scale applications.	The Dynamic analysis used is not efficient in real time applications. The static analysis lacks precision in identifying XSS attacks. The AI methods are effective but the lack of datasets to train them may prove to be faulty in the longer run. A similar problem occurs in cookie analysis which shows that there is a gap in identifying XSS vulnerabilities.

	Trees and classifiers.	intelligence methods.		
(Chrisando Ryan Pardomuan et al. 2023)	Detection System Design, Evaluation Process and Payload Sources.	The detection system used achieved an accuracy of 88%. It was successful in detecting 180 out of the 239 malicious payloads. 59 false negatives The proposed methods had a higher accuracy than the other models (PHPIDS, XSS Auditor).	The system showed a high accuracy rate of 88% which indicates its effectiveness in detecting malicious payloads. This is done on the server side which means that there are no delays from the client side. The server side can analyze requests before it reaches the client thereby reducing the potential XSS attacks.	The system showed a large number or false negatives which was due to the challenges in handling non-standard encoding of malicious payloads. This system is designed solely for the purpose of detecting XSS attacks. There are no measures implemented to prevent it. The time taken can vary depending on the processing capabilities.
(Santithanmanan, K et. al 2024)	k-Nearest Neighbours(k-NN), Decision Tree, Support Vector Machine (SVM), Gaussian Naive Bayes (GNB).	k-NN: Accuracy of 99.99% with a prediction time of 3.3689 seconds. SVM: Accuracy rate of 99.6% Random forest: 99.5% Linear kernel SVM: 96.32%. The confusion matrix shows that the k-NN model had the least fale positives indicating its effectiveness in detecting XSS attacks.	k-NN model showed a very high accuracy which is one of the benefits of this model. The rapid prediction time of the model allows real-time analysis of URLs also making it user friendly. The use of a diverse set of features allows the model to understand the different types of malicious URLs.	The dataset though diverse was small which indicates that there could be a novelty for zero-day vulnerabilities. The models may struggle to identify new types of XSS attacks which are not included in the dataset. The study suggests real time detection capabilities while implementing them in real life may be a challenging task. And eventually it comes down to performance capabilities of the OS.
(Tadhani, J.R. et. al 2023)	Deep learning approach, dataset creation, benchmark datasets and comparison with	99.84% accuracy on the SQL-XSS Payload dataset. 99.23% accuracy on the Testbed dataset. 99.77% accuracy on the HTTP CSIC 2010 dataset.	The model's high accuracy, low false positive rates and versatility can be applied to various network security applications including IDS and WAFs.	The SQL injection statements in production environments can be very complex which makes it difficult for standardization. The model's performance depends on the quality of the dataset used. For the model to give

	traditional methods			similar results in real time a more comprehensive dataset which covers all possible attacks must be used. Though the model performs very well on the datasets provided, its performance in real world scenarios with different attack vectors may vary.

Table 1: Literature review

3 Research Methodology

Web utility safety is still threatened via Cross-Site Scripting (XSS), in which attackers may want to insert harmful scripts into pages that other users view on the net. These cyberattacks take advantage of flaws in internet packages that deal with human beings entering incorrectly, that may compromise distinct statistics and erode confidence in on-line services. The purpose of this paper is to apply Python to create a framework/model for the detection and prevention of reflected XSS attacks and also it will prevent all other attacks. So, this model will be helpful for the web developing industry to develop a secure web application.

3.1 Research design

This study layout combines exploratory and experimental strategies to very well have a look at XSS attack routes and create a beneficial device for detection and mitigation. With the goal of creating a good-sized contribution to internet application protection, this dual method ensures an intensive investigation of XSS vulnerabilities and related mitigation techniques (Kaur et al. 2023).

Exploratory and Experimental: Investigate XSS attack vectors and develop a Python-based detection and prevention model.

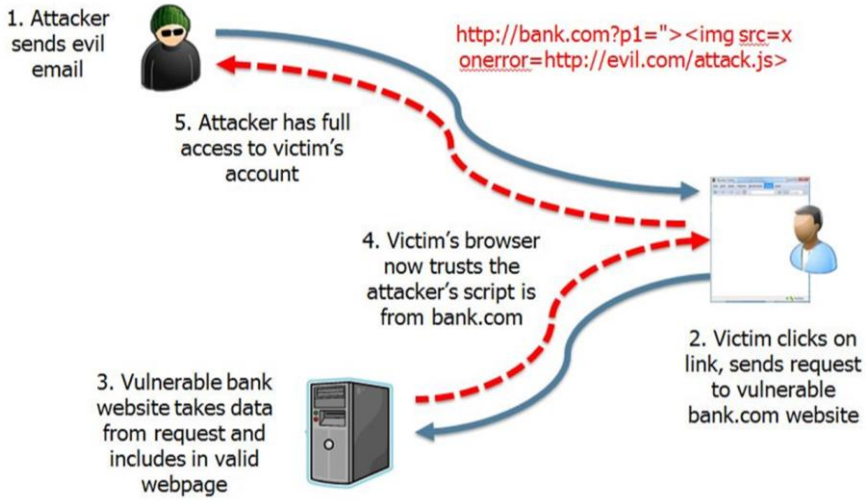


Figure 1: This shows how Reflected XSS works (Meghan Jacquot, 2024)

Reflected XSS happens when a web server application accepts all user inputs and quickly renders back to the client in an unsafe way. Reflected XSS attacks occur when malicious injection affects client users, then the attacker will be able to access all confidential data from the client end (Meghan Jacquot, 2024).

3.2 Detection Mechanism

Heuristic-based analysis and sample popularity are protected within the detecting procedure. Heuristic analysis improves detection accuracy by seeing suspicious styles and behaviors suggestive of XSS assaults, while regular expressions are used to discover malicious XSS payloads in HTTP requests (Younas et al. 2024). Implement regular expressions and heuristic-based detection to identify XSS payloads in HTTP requests. Implement strict input validation the usage of a whitelist method, regular expressions and input sanitization. The custom script is for real time detection and anomaly monitoring.

3.3 Prevention mechanism

In order to mitigate XSS vulnerabilities, powerful preventative strategies are crucial. Robust sanitization methods, along with output encoding and input validation, are the principle subject matter of this work. By screening and sanitizing person inputs and encoding outputs to forestall script execution in online packages, those techniques neutralize feasible XSS vectors (Kshetri et al. 2024). Sanitization techniques are used for input validation and output encoding to neutralize potential XSS vectors in web applications. Robust security methods like CSRF protection and same site attributes for session cookies (ensures that cookies are only transmitted via HTTPS which ensures confidentiality and authenticity of the input) are used (Amal Shaji. 2022). The CSRF token prevents any unauthorised commands being executed by any other actor on behalf of the user.

3.4 Testing Environment

Deploy the tool in a controlled web application environment to monitor and test its performance against various XSS attack scenarios. To prove the web application or the proposed model is safe, thorough testing is conducted through tools like OWASP ZAP and BurpSuite Professional tools.

3.5 Evaluation Metrics

The method used to evaluate the device's efficacy is quantitative evaluation. Important metrics are the detection charge of vulnerabilities, the variety of false positives, the accuracy with which XSS attacks are diagnosed, and the general overall performance of the gadget in various assault eventualities. The effectiveness and dependability of the technology are thoroughly found out by way of these indicators (Chaudhary et al. 2023).

Quantitative Analysis: Measure accuracy, detection rate, false positives, and system performance under different attack simulations.

4 Design Specification

4.1 System Architecture

The system comprises a web application (client) and a Python-based detection and prevention tool (server). The client sends HTTP requests to the server for processing.

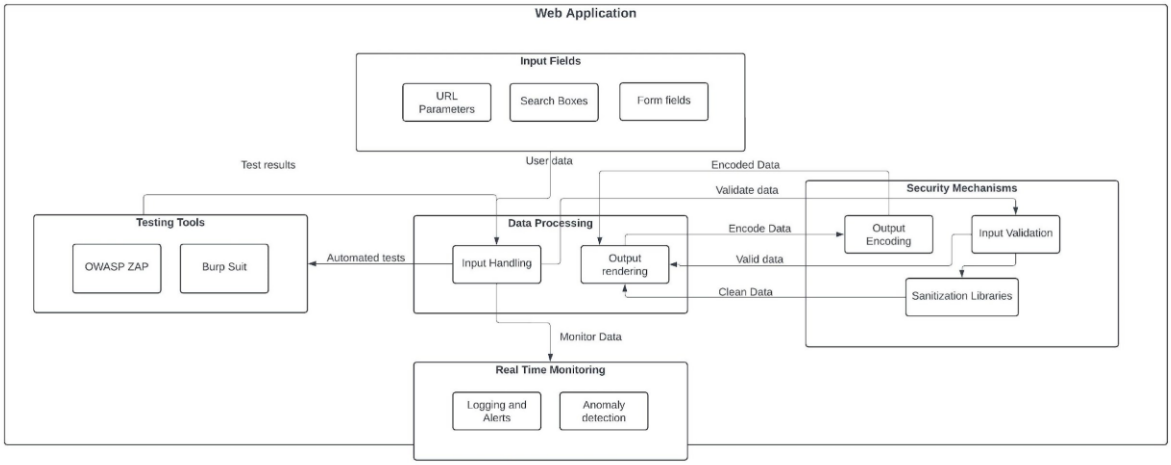


Figure 1: System diagram

The above diagram shows the architecture of the web application. The user data is collected through URL parameters such as search boxes which are then managed and prepared for display. The web application uses the method of input sanitation, output encoding and sanitization libraries to ensure the security conditions are met and the website is free from XSS attacks. To prove the website is safe, thorough testing is conducted through tools like OWASP ZAP and BurpSuite Professional tools. The real time monitoring with logging and anomaly detection makes sure that the user's activities are kept on track and also any suspicious activity is reported. With real time monitoring enabled alerts can be created in case of malicious activities. This overall approach ensures that the web application is thoroughly protected from any kind of XSS attacks.

The diagram illustrates the architecture of a web application with a focus on security mechanisms, data processing, and testing tools.

4.2 Development Environment

Languages and Tools: Python, Flask (web framework), Regex (pattern matching), SQLite (database for logging).

4.3 Detection Module

To identify XSS payloads in HTTP requests from the users or clients, the detection module is mainly including the regular expressions and heuristic-based detection. So, pattern matching is the major feature, where it utilizes regular expressions to detect known XSS patterns in HTTP requests. Moreover, this module includes strict input validation, the usage of a whitelist

method and input sanitization. The custom script is for real time detection and anomaly monitoring.

4.3.1 Activity Logging & Real time Monitoring

Log details of detected and prevented XSS attempts in an SQLite database and the real-time monitoring of HTTP requests and responses to identify and mitigate potential attacks promptly. If a client provides malicious input to the web application, then this model immediately prevents the same and stores its details into the log database as well as displays onto the real time monitoring page.

4.4 Prevention Module

Ensure all user inputs are validated and sanitized to prevent malicious scripts. Strong sanitization methods, along with output encoding and input validation, are the principle subject matter of this work. Encode dynamic content before rendering in the web browser to prevent script execution. By screening and sanitizing person inputs and encoding outputs to forestall script execution in online packages, those techniques neutralize feasible XSS vectors (Kshetri et al. 2024). Sanitization techniques are used for input validation and output encoding to neutralize potential XSS vectors in web applications. Regarding the Test Scenarios, develop test scenarios covering various types of reflected XSS attacks with the help of Burp Suite Professional Tool and dataset.

4.6 Performance Metrics

Accuracy: Measure true positive rate (correct detections) and false positive rate (incorrect detections).

5 Implementation

Python Flask is used to mimic internet requests and responses, which lets in a sizable controlled environment checking out of the device's efficacy (Abhishek et al. 2023). Regular expressions are used within the implementation to mix pattern recognition techniques by way of scanning HTTP requests for known XSS patterns and signatures. By assessing contextual information and user behaviour, heuristic evaluation improves detection accuracy and will increase the tool's capacity to become aware of diffused or changing XSS assault vectors. Implementing sturdy sanitization strategies is crucial to the manner. The internet application handiest accepts safe and anticipated statistics codecs since enter validation strategies are cautiously created to clear out and sanitise consumer inputs (Sharma et al. 2023). Before doubtlessly dangerous scripts are rendered in customers' browsers, output encoding techniques convert them into safe strings, thereby lowering the danger of cross-web site scripting attacks (Hannousse et al. 2024).

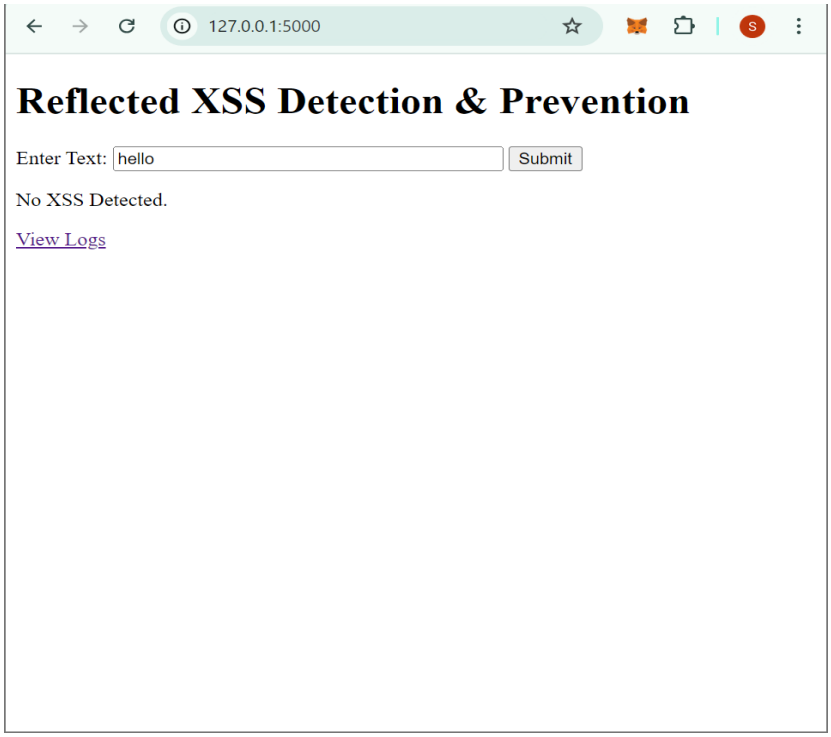


Figure 2: User input - GUI

The above figure shows the home page of the model, where the user enters input through the text box. This model is fully secured against reflected XSS attacks, SQL injection attacks, etc.

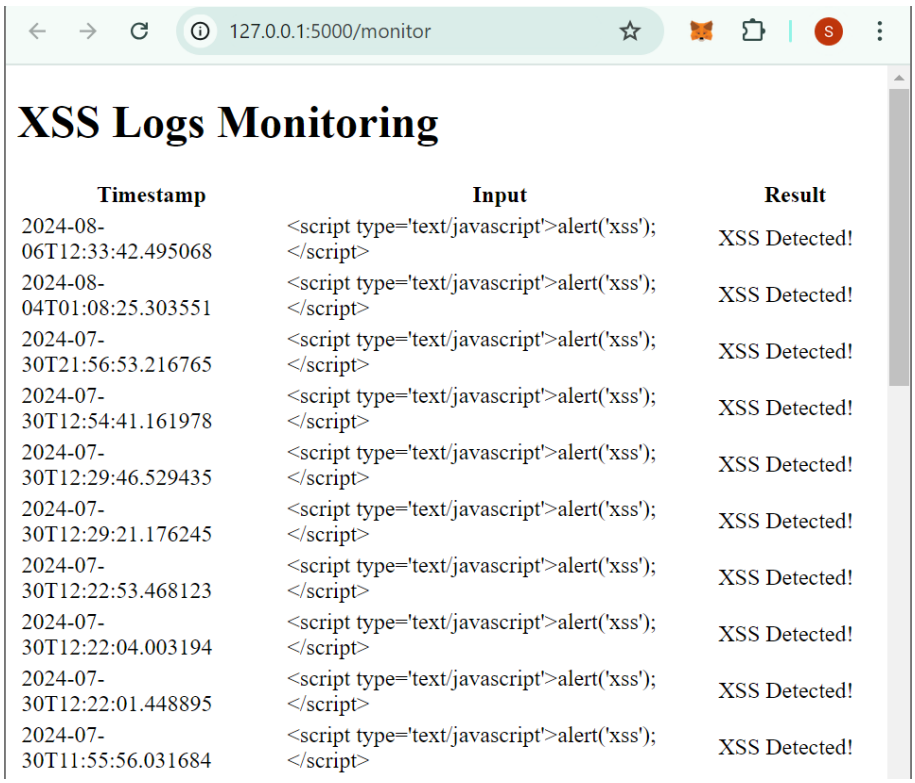


Figure 3: Real time monitoring - GUI

If a user provides malicious input to the web application, then this model immediately prevents the same and stores its details into the log database as well as displays onto the real time monitoring page.

6 Evaluation

Web programs are at serious risk from Reflected Cross-Site Scripting (XSS) attacks, which permit attackers to run malicious scripts within the consumer's browser. Ensuring the safety and integrity of online programs requires the timely detection and prevention of such threats. The evaluation standards, which encompass robustness, scalability, usability, detection accuracy, performance, and assessment analysis, provide a radical framework for evaluating the efficacy of technology meant to counteract reflected XSS attacks.

Evaluation Criteria	Description
Detection Accuracy	Measure the true positive rate (correct detections) and false positive rate (incorrect detections) of the tool in identifying XSS attacks.
Efficiency	Assess the tool's impact on web application performance, including response time and resource utilization under various load conditions.
Robustness	Evaluate the model’s ability to handle different types of reflected XSS attacks, including common payloads and obfuscation techniques.
Scalability	Test the model’s performance and effectiveness in larger web application environments with higher traffic volumes.

Table 2: Evaluation

6.1 Baseline Measurements

A preliminary vulnerability scan of the net utility using both automated technology and human processes becomes the first step inside the baseline measures. The intention of this thorough analysis is to locate modern-day XSS vulnerabilities, which may range from trustworthy script injections to sophisticated exploits that target distinct application components (Hannousse et al. 2024). Prior to the implementation of any security features, the experiment gave an accurate view of the utility's security posture. To nicely prioritise remediation efforts, crucial vulnerabilities were classified based totally on severity levels.

6.2 Post Implementation Testing

OWASP ZAP and Burp Suite professional tools have been used to do a comprehensive testing of this system. The motive of this put up-implementation testing segment was to evaluate how well the safety controls that have been put in place mitigated vulnerabilities that had already been located (Al-Haija et al. 2023). The reason for the retest turned out to be to verify that the number of XSS vulnerabilities had reduced and to assess any new or lingering risks which could be added at some point of the implementation system.

To verify the outcomes of automated gear, manual penetration testing and code reviews have been completed similarly to automatic scans. Using input fields, URLs, and different software interfaces as goals for XSS attacks. By ensuring a radical assessment of the application's safety posture after implementation, manual verification showed the efficacy of the proposed techniques to protect against reflected XSS attacks.

6.2.1 Testing - Burp suite Professional tool

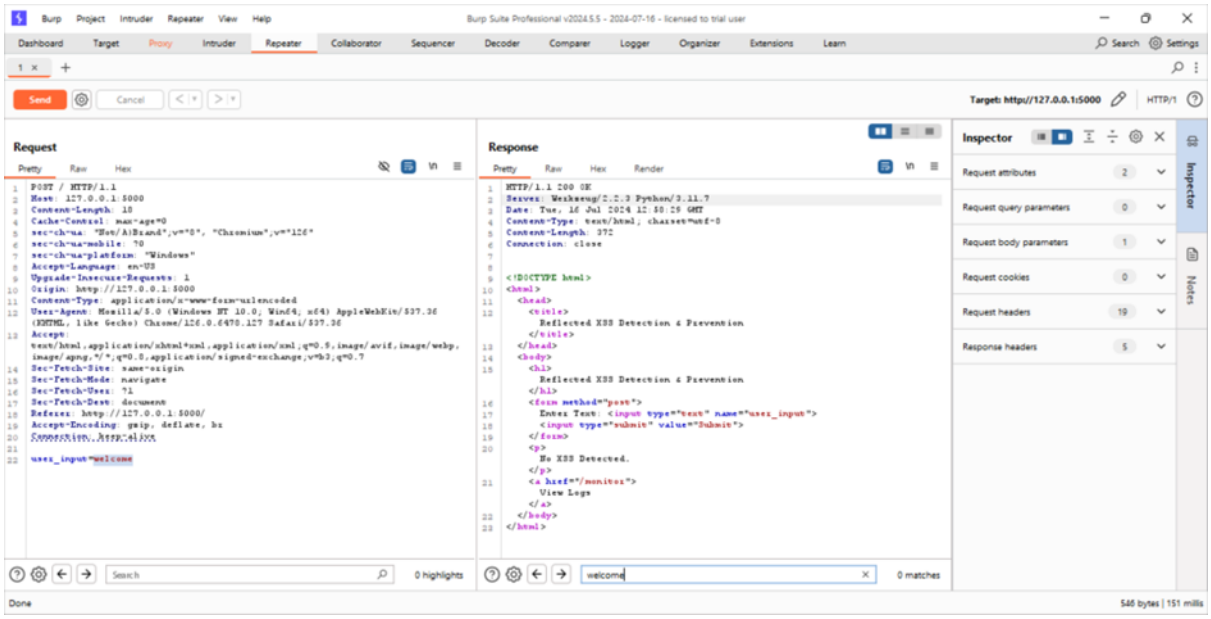


Figure 4: Testing of the proposed system by Burp suite Professional tool

First of all, I have tested whether this application has reflected XSS vulnerabilities or not. I Turned on the intercept feature in the Burp Suite tool and submitted the text “welcome” into the web application. This user input is forwarded into the repeater and intruder of this tool. Figure 4 shows the response part of the repeater does not have the user input, which is “welcome”. So, the reflected XSS vulnerability is not present in this application.

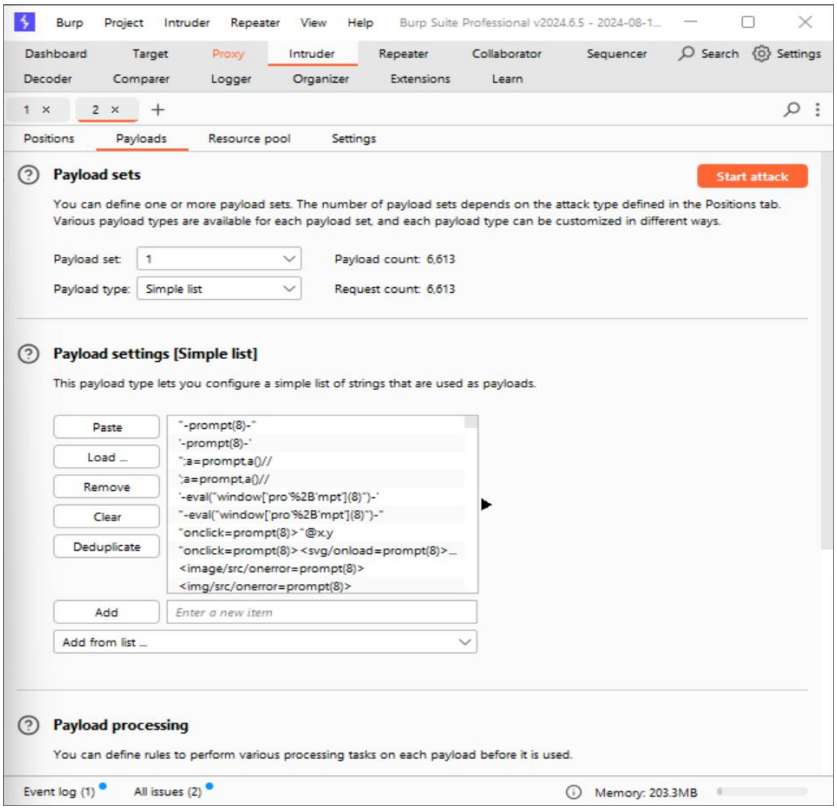
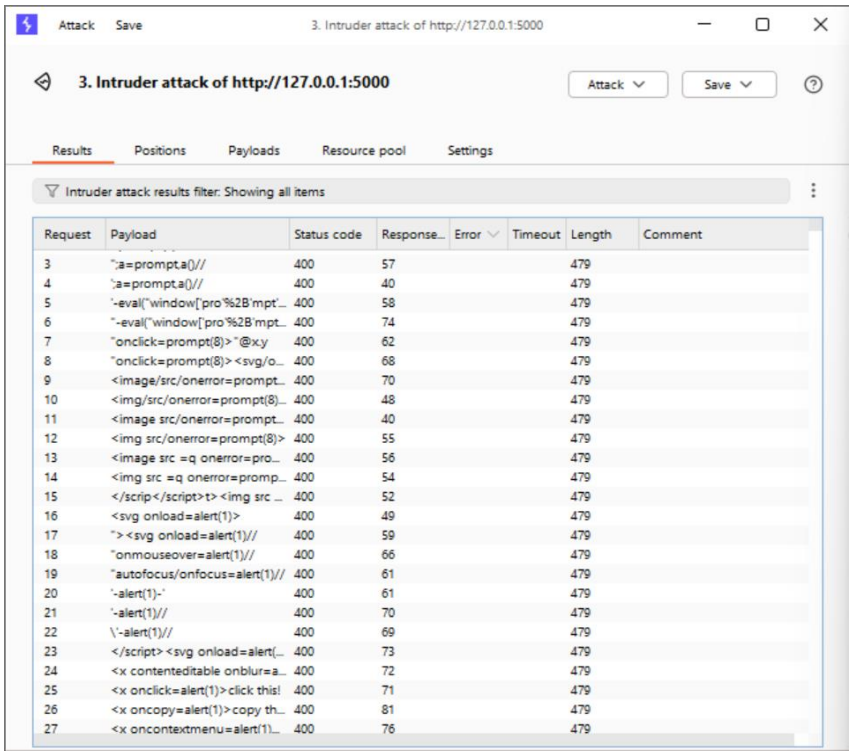


Figure 5: Testing of the proposed system by Burp suite Professional tool

Used a payload set (Riodrwn Rio D. 2021) of 6613 scripts to attack the proposed system through the intruder feature of the burp suite tool.



Request	Payload	Status code	Response...	Error	Timeout	Length	Comment
3	";a=prompt,a()//	400	57			479	
4	';a=prompt,a()//	400	40			479	
5	'-eval("window['pro'%2B'mpt'...	400	58			479	
6	"-eval("window['pro'%2B'mpt...	400	74			479	
7	"onclick=prompt(8)>"@x.y	400	62			479	
8	"onclick=prompt(8)><svg/o...	400	68			479	
9	<image/src/onerror=prompt...	400	70			479	
10	<img/src/onerror=prompt(8)...	400	48			479	
11	<image src/onerror=prompt...	400	40			479	
12	<img src/onerror=prompt(8)>	400	55			479	
13	<image src =q onerror=pro...	400	56			479	
14	<img src =q onerror=promp...	400	54			479	
15	</scrip</script>t><img src ...	400	52			479	
16	<svg onload=alert(1)>	400	49			479	
17	"><svg onload=alert(1)//	400	59			479	
18	"onmouseover=alert(1)//	400	66			479	
19	"autofocus/onfocus=alert(1)//	400	61			479	
20	'-alert(1)-'	400	61			479	
21	'-alert(1)//	400	70			479	
22	\'-alert(1)//	400	69			479	
23	</script><svg onload=alert(...	400	73			479	
24	<x contenteditable onblur=a...	400	72			479	
25	<x onclick=alert(1)>click this!	400	71			479	
26	<x oncopy=alert(1)>copy th...	400	81			479	
27	<x oncontextmenu=alert(1)...	400	76			479	

Figure 6: Testing Results - Burp suite Professional tool

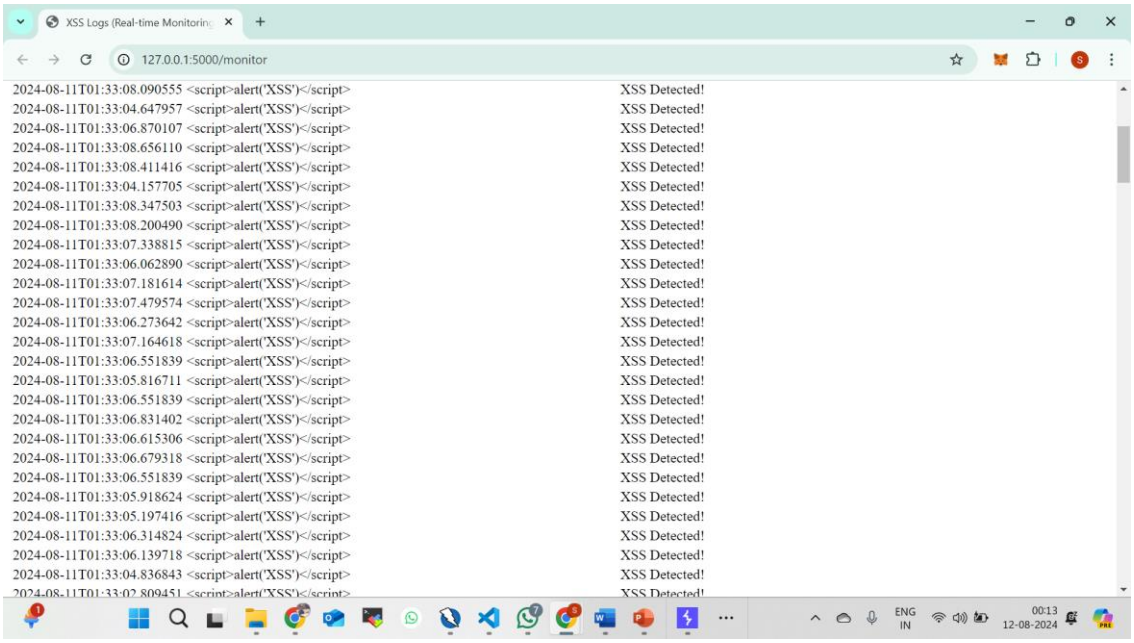


Figure 7: Live monitoring page showing the detected XSS attacks.

After the successful attack implemented by the Burp Suite tool, only the XSS attacks detected are shown in the live monitoring page. This helps the admin find out and filter the XSS attack payload. As per the above results, the proposed system was successfully detected and prevented the whole attacks with less response time. Size of the dataset (Riodrwn Rio D. 2021) is 6613. Logs regarding the detected attacks have been stored into the logs database and displayed onto the real time monitoring page.

6.2.2 Testing - OWASP ZAP Tool

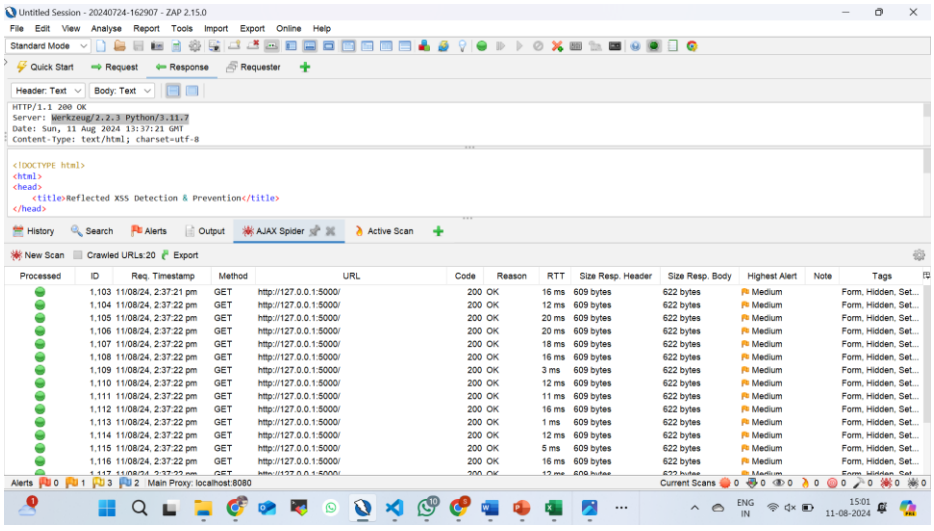


Figure 8: Scanning - OWASP ZAP tool

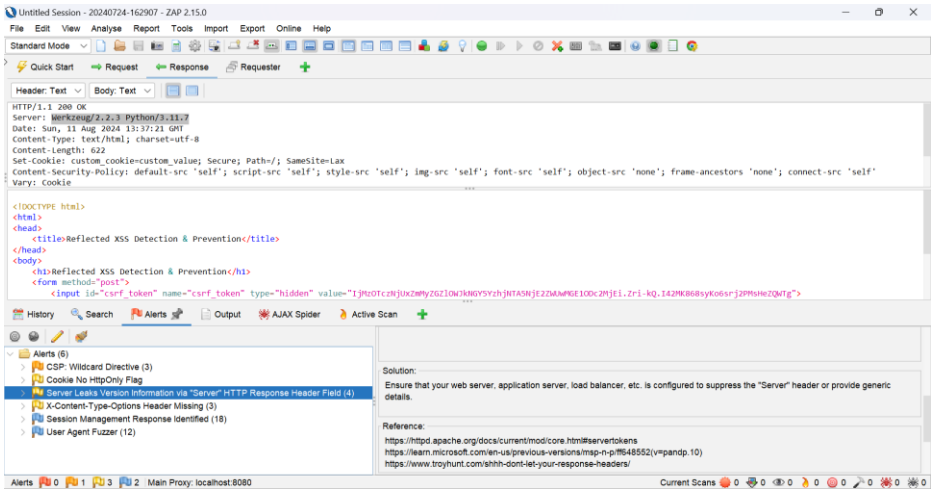


Figure 9: Testing Results - OWASP ZAP tool

To fine tune the results, I did multiple sample testing in order to make my proposed system a secured one against not only reflected XSS attacks but other attacks too. Currently we can see an alert raised for both http header and CSP header (as seen in figure 9), as it is running on the Python flask and hosted on the local host system.

6.3 Results

Existing Models	**Proposed Model**
Fails to detect and prevent Reflected XSS attacks.	It is successful in detecting and preventing Reflected XSS attacks.
Fails to provide live monitoring of the detected XSS attacks.	It provides live monitoring and notifies the user in case of XSS attacks.
Filtering and sanitization of user input is not done.	Filtering and sanitization of user input is done.
Such robust security methods are not used which may indicate vulnerabilities in the model.	Robust security methods like CSRF protection and same site attributes for session cookies (ensures that cookies are only transmitted via HTTPS which ensures confidentiality and authenticity of the input) are used (Amal Shaji. 2022).
No such methods are implemented.	The CSRF token prevents any unauthorised commands being executed by any other actor on behalf of the user.

Table 3: Comparing the existing and proposed models

7 Conclusion and Future Work

This research shows the advantage of using a different approach to detect XSS attacks which uses a list of different methodologies to enhance the detection and prevention of XSS attacks particularly the reflected variant. By combining methods like input sanitization, CSRF protection and secure cookie attributes the proposed and developed model is successful in detecting and preventing XSS attacks while ensuring the confidentiality and authenticity of the user. The model also has live monitoring which shows the XSS attacks thus reducing the time to mitigate them. The model is thoroughly tested for vulnerabilities and constantly updated with security features. These findings suggest that this model can greatly assist web developers to create more secure web applications ultimately raising the bar for detection and prevention of XSS attacks across the internet. The future work includes exploration of Content Security Policies (CSP) and the enforcement of HTTPS for secure communication. These features can further increase the security of the web applications.

References

Abhishek, S., Ravindran, R., Anjali, T. and Shri Amrut, V., 2023, March. Ai-driven deep structured learning for cross-site scripting attacks. In 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA) (pp. 701-709). IEEE.

Al-Haija, Q.A., 2023. Cost-effective detection system of cross-site scripting attacks using hybrid learning approach. Results in Engineering, 19, p.101266.

Amal Shaji, 2022. CSRF Protection in Flask. [Online] Available at: https://testdriven.io/blog/csrf-flask/ [Accessed 30 July 2024].

Bleepingcomputer, 2024. Hackers steal data of 2 million in SQL injection, XSS attack. [Online] Available at: https://www.bleepingcomputer.com/news/security/hackers-steal-data-of-2-million-in-sql-injection-xss-attacks/ [Accessed 5 June 2024].

Chaudhary, P., Gupta, B.B. and Singh, A.K., 2023. Adaptive cross-site scripting attack detection framework for smart devices security using intelligent filters and attack ontology. Soft Computing, 27(8), pp.4593-4608.

Chen, H.-C. a. N. A. a. D. C. a. C. P.-H., 2021. Detection and Prevention of Cross-site Scripting Attack with Combined Approaches. In: *2021 International Conference on Electronics, Information, and Communication (ICEIC).* s.l.:IEEE, pp. 1-4.

Chrisando Ryan Pardomuan, A. K. M. Y. D. M. A. M. A. a. Y. M., 2023. Server-side Cross-site Scripting Detection Powered by HTML Semantic Parsing Inspired by XSS Auditor. *Pertanika Journal of Science & Technology*, 31(3).

Cui, Y. a. C. J. a. H. J., 2020. A Survey on XSS Attack Detection and Prevention in Web Applications. In: *Proceedings of the 2020 12th International Conference on Machine Learning and Computing.* Shenzhen, China: Association for Computing Machinery, p. 443–449.

Meghan Jacquot, 2024. *Differences of Stored XSS and Reflected XSS.* [Online] Available at: https://www.inspectiv.com/articles/differences-of-stored-xss-and-reflected-xss [Accessed 5 June 2024].

Gatlan, S., 2024. *High-severity GitLab flaw lets attackers take over accounts.* [Online] Available at: https://www.bleepingcomputer.com/news/security/high-severity-gitlab-flaw-lets-attackers-take-over-accounts/ [Accessed 5 June 2024].

Geeksforgeeks, 2023. *Geeksforgeeks*. [Online] Available at: https://www.geeksforgeeks.org/the-importance-of-security-testing-in-todays-digital-age/ [Accessed 5 June 2024].

Hannousse, A. a. Y. S. a. N.-H. M. C., 2024. Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey. *Computer Science Review, Elsevier BV,* p. 100634.

Iman Fareed Khazal, Mohammed Abdulridha Hussain, 2021. “Proposed Method to Detect and Prevent Reflected Cross-Site Script Attack”

Kass, D. H., 2020. *Study: Cross-Site Scripting Nearly 40% of All Online Cyber Attacks in 2019.* [Online] Available at: https://www.msspalert.com/news/study-cross-site-scripting-attacks [Accessed 21 June 2024].

Kaur, J., Garg, U. and Bathla, G., 2023. Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. Artificial Intelligence Review, 56(11), pp.12725-12769.

Kshetri, N., Kumar, D., Hutson, J., Kaur, N. and Osama, O.F., 2024, April. algoXSSF: Detection and analysis of cross-site request forgery (XSRF) and cross-site scripting (XSS) attacks via Machine learning algorithms. In 2024 12th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-8). IEEE.

Matteson, S., 2018. *British Airways data theft demonstrates need for cross-site scripting restrictions.* [Online] Available at: https://www.techrepublic.com/article/british-airways-data-theft-demonstrates-need-for-cross-site-scripting-restrictions/ [Accessed 5 June 2024].

OWASP, F., 2021. *OWASP Top Ten 2021*. [Online] Available at: https://owasp.org/www-project-top-ten/ [Accessed 5 June 2024].

Rodriguez, G. a. T. J. a. F. P. a. B. E., 2019. Cross-Site Scripting (XSS) Attacks And Mitigation: A Survey. *Computer Networks*, Volume 166, p. 106960.

Riodrwn Rio D. 2021. Cross Site Scripting (XSS) Vulnerability Payload List. [Online] Available at: https://github.com/payloadbox/xss-payload-list/tree/master/Intruder [Accessed 1 Aug 2024].

Santithanmanan, K., Kirimasthong, K., Boongoen, T. 2024. Machine Learning Based XSS Attacks Detection Method, Advances in Computational Intelligence Systems. UKCI 2023. Advances in Intelligent Systems and Computing, vol 1453. Springer

Sharma, S. and Yadav, N.S., 2023. A multilayer stacking classifier based on nature-inspired optimization for detecting cross-site scripting attacks. International Journal of Information Technology, 15(8), pp.4283-4290.

Tadhani, J.R., Vekariya, V., Sorathiya, V. et al. Securing web applications against XSS and SQLi attacks using a novel deep learning approach. Sci Rep 14, 1803 (2024). https://doi.org/10.1038/s41598-023-48845-4

Tan, Xiaobo and Xu, Yingjie and Wu, Tong and Li, Bohan, 2023. Detection of reflected XSS vulnerabilities based on paths-attention method. Applied Sciences. Applied Sciences, Volume 13.

Toulas, B., 2024. *Joomla fixes XSS flaws that could expose sites to RCE attacks.* [Online] Available at: https://www.bleepingcomputer.com/news/security/joomla-fixes-xss-flaws-that-could-expose-sites-to-rce-attacks/ [Accessed 5 June 2024].

Younas, F., Raza, A., Thalji, N., Abualigah, L., Zitar, R.A. and Jia, H., 2024. An efficient artificial intelligence approach for early detection of cross-site scripting attacks. Decision Analytics Journal, 11, p.100466.