International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Cross-Site Scripting (XSS) in Web Applications: A systematic literature review

Tenzin Yarphel * and Diksha Rani

Lovely Professional University Phagwara, India.

International Journal of Science and Research Archive, 2025, 15(02), 1658–1667

Publication history: Received on 16 April 2025; revised on 25 May 2025; accepted on 27 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.2.1521

Abstract

Cross-Site Scripting (XSS) continues to be a prevalent and damaging vulnerability in web applications, leading attackers to inject harmful scripts that can put personal data at risk, hijack sessions, and change website content. This research provides a comprehensive literature overview of XSS attacks that classify them as stored, reflected, and DOM-based, and discuss how these attacks have evolved as web technology advanced. Traditional detection methods such as input validation and signature-based filters are becoming less and less effective against sophisticated, evasive payloads. As a result, researchers are beginning to utilize Machine Learning (ML) and Deep Learning (DL) methods as more adaptive and intelligent detection methods. This paper reviews different ML/DL models for XSS detection and examines their methods, datasets, feature engineering methods, and metrics for performance. Also pointed out are significant problems such as class imbalance, adversarial examples, and deployment barrier. This study combines current research so that gaps can be identified and future directions described to build effective, scalable, and real-time XSS detection systems. The study also points out that intelligent automation is crucial in protecting web applications against the increasingly sophisticated threat landscape.

Keywords: XSS; Cross site scripting; Injection

1. Introduction

Cross-site scripting (XSS) is a common web vulnerability that allows attackers to inject malicious scripts into web sites, affecting millions of people across the globe. XSS has been in increased focus over the years, demonstrated by it remaining as one of the most important web application security risks in the OWASP Top 10 [1]. XSS takes advantage of the inherent trust between users and web applications, permitting attacks such as illicit script execution, credential theft, and the distribution of malware, that fundamentally compromise the cyber security climate. Dynamic nature of XSS attacks and increasingly complex techniques that evade protection, illustrate that these attacks are multifaceted and dangerous.

This review intends to systematically research the development of XSS attack types, the recent emergence of detection techniques, and the potential for a defence strategy. Current research highlights inefficiencies in static and dynamic detection techniques, along with increased reliance on machine learning (ML) and deep learning (DL) models to mitigate XSS attacks. Research which has employed bidirectional LSTM with attention mechanisms [2] or convolutional neural networks [3] has displayed promise in isolating the semantic complexity and obfuscation used by attackers in XSS attacks. Generally, these techniques include detection and response capabilities that exceed traditional rule-based systems.

This literature review includes developments made since 2020, focusing on machine learning (ML) and deep learning (DL) detection techniques, statistical approaches, and defence strategies such as Content Security Policies (CSP). In addition, it also considers some real-world challenges related to feature extraction, computational overload, and the

* Corresponding author: Tenzin Yarphel.

demand for datasets which support the diversification of contemporary online application ecosystems. This paper intends to provide a practical perspective on the prevention of XSS attacks, adding to the ongoing discussion on online application security.

2. Importance of Literature review

This literature review can be helpful because it can provide context and unify improvements to our knowledge and mitigation of Cross-Site Scripting (XSS) vulnerabilities and attacks. It attempts to demonstrate the evolution of XSS and the ways in which these vulnerabilities have shifted along with advancements in online technology. This paper is primarily focused on the need for proper detection methods, bridging the growing availability of machine learning approaches and deep learning approaches, in the ongoing fight against increasingly advanced attack methods. This study focuses on an important shift from simple reflected and stored XSS phrases to more advanced DOM-based attacks. It will also discuss various responses to attacks including Content Security Policies (CSP) and runtime monitors and how useful they have been in practice. Recent studies have documented deficiencies with established tools; nevertheless, different techniques can be incorporated allowing machine learning techniques to be extremely beneficial towards improving detection accuracy and lowering false positives. This literature review will provide a relevant overview and will provide helpful considerations for researchers and practitioners. In addition, it will connect theoretical knowledge with practice and help close the gaps on improving web application safety from XSS vulnerabilities.

3. Origin of XSS

Cross-site scripting (XSS) vulnerabilities originated in the late 1990s, concurrent to the introduction of interactive web applications becoming widely used at the time and which relied heavily on user input. In 1999, Microsoft engineers were one of the first to document XSS vulnerabilities, to describe their observations that the failure to properly validate input on the server side left the possibility to allow an attacker to add and execute a malicious script, within a page hosted on a server. The original research highlighted the security risks associated with poor server-side control of input and provided a starting point for more structured use of the vulnerabilities as a tool for research in this area [4]. Early attacks consisted mainly of reflected XSS, in which malicious payloads were added to URLs and executed immediately upon the unwitting user clicking the link. Stored XSS attacks compensated the malicious payload, added to a database that persisted, and with repeated requests and executions affected many users over time. XSS identified significant issues with server-side processing and client-side rendering which permitted attacks that led to scale, in addition to massive theft of data, session hijacking, and other criminal exploits [5]. The impact of these attacks led to extensive research into mitigations, and current methods of XSS mitigation, including input sanitization, Content Security Policies (CSPs), and machine learning based detection of rational attacks, to provide security against both lower level and advanced XSS attacks. Overall, the escalation of XSS attacks highlights the need for continual development to ensure online application security against advancing methods of attack, as they will continue to rise in frequency and efficiency over time.

4. Evolution of XSS

Cross-site scripting (XSS) has emerged since the late 1990s as a simple web exploitation to a complex attack vector. It is a prevalent risk, and it continues to pose a problem because of the ways of the web and the complexity of web technology.

4.1. The Beginnings: Reflected and Stored XSS

XSS was initially about exploiting server-side vulnerabilities or as a separate question was still an ending exploit. Reflected XSS was almost like attacking users in impromptu training by getting them to click an attack link which resulted in the scripts executing immediately and using your browser. Stored XSS is more persistent, as it embeds the scripts into a known source, usually a database and triggers effects everytime anyone accesses this input source. Many of these attacks include stealing session cookies, misusing data access and misusing data and changing the content of the web site. From early on, this began ever so slightly showing weaknesses in web development practice with a particular lack of input validation and output encoding which continues today. Web vendors were just about getting stuff done and building useful, functional products, and then after launch they could think about security. and in the end websites / applications were created that had prevented even the most basic level of control or governance for XSS. The most rudimentary categories of controls were those of input sanitation and blacklisting models.

4.2. Progression to Client Side Exploitation

With the popularity of client-side scripting and JavaScript frameworks now becoming prevalent, XSS began to evolve again. Emerging next was an attack, known as DOM-based XSS, which exploited vulnerabilities specifically within the client-side DOM itself. The attacks were different because attacks were happening without any server-side validation, meaning anything would go. This was a revelation for many, especially when the reality was to be completely aware and cognizant of these new limitations of defenses. Developers have begun to properly review security work involving secure code practice, the data lifecycle, and monitoring your code in real-time.

4.3. Modern Threats: Obfuscation and Chain Attacks

Attackers have now shown their ability to evolve their attacks to include obfuscated payloads and to spread their attacks through a number of vectors. By encoding malicious programs or simply using payloads, such as Base64, and coordinating in attacks that include mass amounts of vulnerabilities with XSS payloads they are now able to lessen the impact of detection scenarios. Additionally, XSS almost always finds its way into greater cyber-attacks, like supply chain/remove or phishing attacks, increasing the impact where we previously have limited binding. For example, defenders may be attacked using XSS to place malware into popular third-party libraries or otherwise use cloud integrations in order to circumvent defenses that are currently in place but are unaware of attack intention. Take all into perspective and note the evolution of complexity in the current threat landscape surrounding XSS attacks, and the continued difficulties these provide to attackers.

4.4. Continued Relevance and Challenges

While counter methods of detection and development of security methods, like Content Security Policies (CSP) have grown, XSS attacks still have a high presence in today's applications. For single-page applications (SPAs) using JavaScript as the main programming language, they keep their vulnerabilities because of their dynamic content, requires THE full complement of security; good coding, machine learning detection for security before, during, and after development along with security evaluation and threat analysis. The evolution of XSS attacks document the ability of cyber threats to adapt, and as web tech continues to and become more complex, so too will the attackers continue to evolve, therefore increasing the need for vigilance, creativity, and familiarity to stop XSS to a lesser degree.

5. Key Milestone In XSS research

The evolution of Cross-Site Scripting (XSS) research is indicative of the ever-changing field of cybersecurity and the ongoing efforts to address this ongoing threat. Important milestones include pioneering research, new technologies, and impactful events that helped to propel the community's understanding of XSS. The public acknowledgment of XSS as a problem in the late 1990s was a turning point for researchers when they recognized the threat associated with not sanitizing input in a web application, which led to the very first split between reflected vs. stored XSS, which was largely targeting server-side security versus client-side security. With Amit Klein's study on DOM-based XSS, the problem was expanded to include client-side vulnerabilities, which was an important reminder about the implications of secure coding [6].

At the time, most of the research was focused on understanding how malicious scripts could exploit web applications and how to prevent these scripts from executing in the browser. These efforts highlighted a significant gap in application architecture and security or pushed the developers to deal with the implications of input validation and output encoding. There have been recent improvements in techniques for detection such as both dynamic taint analysis and concolic execution have advanced techniques to identify vulnerabilities in real-time. In 2023, Jemin Kim and J. Park proposed a framework using both technologies to obtain above 90% detection accuracy in IoT applications. This invention not only improved detection rate but highlighted the importance of automating security testing for large web services. The move to use machine learning in place of XSS detection is an important change well beyond the technologies alone. AI-enabled systems could analyse traffic patterns and identify anomalies that indicated an oddity suggestive of an XSS attack even from altered payloads. This capability is especially important in an environment where threat actors constantly adapt [7].

Analysis and Commentary Despite research papers being focused on technical details, it is important to think about the broader implications of XSS research in the industry. The increasing awareness of XSS as a vulnerability has changed how developers view security in web applications. Ideas like secure development lifecycles (SDLC) and DevSecOps processes have been developed as formal responses to vulnerabilities like these. Additionally, platforms like React and Angular have built-in libraries to mitigate against common vulnerabilities, showing us how research can influence developer best practices. One of the most prominent shows in 2018, British Airways suffered a data breach due to a

JavaScript attack similar to a cross-site scripting (XSS) style attack. In this case, attackers placed malicious malware on the airline's website, specifically on the payment page to gain customers' personal and financial information while completing their purchase. Information from roughly 380,000 payment cards was compromised citing a £20 million fine for breaching GDPR laws, illustrating the incredibly serious ramifications of bad web application security [8].

6. Mechanism of XSS

XSS attacks typically follow a sequence:

6.1. Iinjection

The attacker embeds harmful JavaScript code into the application, typically via user input fields like comment sections, search bars, or URL parameters.

6.2. Execution

Upon the victim's access to the vulnerable page, the script is executed within their browser. This may lead to behaviours such as cookie theft, data transmission to a nefarious server, or diverting the victim to a phishing website.

6.3. Exfiltration or Impact

The attacker's script can collect and send sensitive data, perform actions on behalf of the user (such as changing account settings), or propagate to other users, continuing the attack.

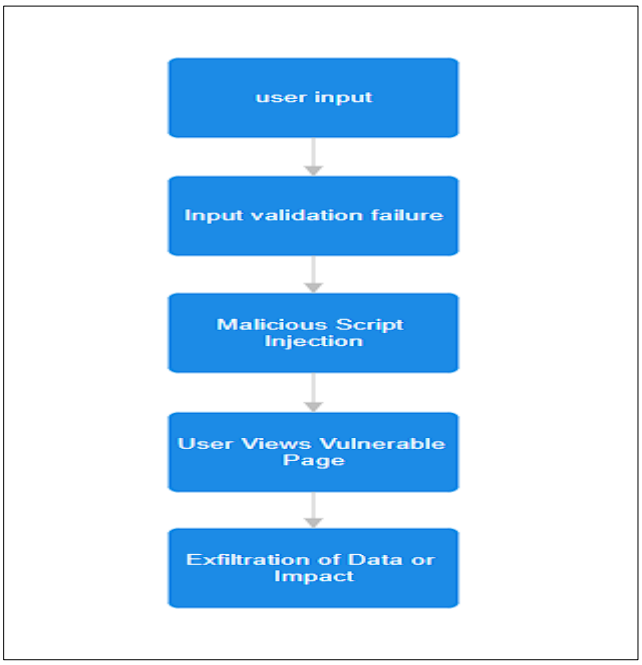

Figure 1 XSS workflow illustrating the stages from payload injection to execution

As shown in Fig. 1. The following steps are followed.

- *User Input:* The attacker provides user input (e.g., through a form or URL parameter).
- *Input Validation Failure:* The web application does not validate or "sanitize" the input.
- *Malicious Script Injection:* The attacker injects a malicious script (e.g., JavaScript) through the input field.
- *User Views Vulnerable Page:* A victim views the web page that includes the injected script.
- *Script Execution in Browser:* The malicious script has now executed in the victim's browser.
- *Exfiltration of Data or Impact*: The attacker can now extract sensitive data, hijack sessions, etc.

7. XSS variant

7.1. Stored XSS (Persistent XSS)

Stored cross-site scripting (XSS) exists when an attacker embeds malicious scripts into a website's database or server, and the script is stored persistently. This script can later be delivered to any users attempting to access the infected webpage. The malicious code is stored on the server, which means it will be persistent from session-to-session and also affect any users who visit the infected website. For example, an attacker submits a comment to a blog which embeds harmful JavaScript that is stored in the server's database. As a result, whenever a user views the comment, the user will run the attacker's script in their browser.

7.2. Reflected XSS

Reflected Cross-Site Scripting exists when an attacker sends a malicious script in a URL or HTTP request, which gets reflected back in the server's response (without being stored). This often happens when a user clicks on an intended link or submits a form with malicious input (e.g. in a comment field). The server then includes that input back into the page content and tells the user's browser to execute the malicious script. Unlike stored XSS, reflected XSS is not persistent; it only affects users who click on the attacker's intended URL or submitted the compromised form directly.

7.3. DOM-based XSS

DOM-based Cross-Site Scripting occurs when the attackers' malicious script manipulates the Document Object Model (DOM) in the browser, rather than injected directly into the server's response. The attacker script runs when users are interacting with the webpage and the script manipulates what you see on the page using client-side JavaScript. The vulnerability occurs because of a mis implementation of how the browser treats and modifies the DOM when the user passes input as data (via URL fragments, or form fields). Failure to correctly sanitize this input could provide an opportunity for the attacker to inject and run malicious JavaScript in the context of the victim's browser. Cross-site scripting (XSS) attacks can have significant consequences for users and owners of web applications. When attackers successfully exploit an XSS vulnerability, they obtain the ability to execute arbitrary scripts in the context of the victim's browser. These attacks can have numerous negative consequences, which can affect both individuals and organizations deeply.

8. Impact of XSS

8.1. Data theft

One of the most common outcomes of XSS attacks is data theft. Attackers can place scripts that extract cookies or session tokens, or other sensitive data from a victim's browser. Once attackers have stolen data, they may be able to take over accounts, perform actions on users' behalf, or gain access to all other sensitive data. When the victims are banking sites, retail sites that use user accounts, and social networking sites, there is a very serious risk of exposing personal and financial information. Studies show that XSS is a frequent attack vector and can bypass security and directly compromise user data [4].

8.2. Session Hijacking

XSS attacks can enable session hijacking, wherein an attacker XSS attacks can help carry out session hijacking directly — for example, where an attacker steals a session token form an authenticated user. An attacker can gain unauthorized access to an account by injecting a malicious script that extracts the session cookie from a victim's browser. This is a major threat for applications that manage monetary transactions and sensitive user data. Further demonstrating how unsafe practices can, through XSS vulnerabilities, enable successful session hijacking, is a study regarding CSP nonce reuse - which is recognized as a highly unsafe [9].

8.3. Defacement of Web Pages

Attackers can exploit XSS to manipulate the visual Attackers can use XSS to manipulate the way a site looks, including its text, graphics, and logos. While nothing might be stolen, there is a major impact on the organization's brand and reputation. Traffic and user trust may decline if a site looks corrupt. Studies on content injection vulnerabilities indicate that unintended script execution may result in serious reputational harm [10].

8.4. Malware Delivery

In more complex XSS attacks, an attacker could place scripts that distribute malware, such as ransomware, keyloggers, and trojans. Once the script executes in a victim's browser, it can download and install malware, which will create even more security risks. Once a system is infected, it may spread throughout a network and potentially cause harm to an entire organization. A recent study presented machine-learning based detection system to identify and neutralize threats before they have an impact on end user [11].

8.5. Phishing and Social Engineering

XSS is often used in phishing schemes, when a malicious actor changes a website's text in order to get a user to voluntarily provide sensitive data. By using bogus login forms, or redirecting users to malicious sites, attackers can obtain credentials and personal information. This method is still a serious security concern, because a compromised browser and interface still mean that users will normally trust a familiar interface. Recent studies on big data enabled XSS detection have pointed out how quickly attackers can use XSS for mass phishing campaigns [5].

9. XSS detection Technique

Finding cross-site scripting (XSS) vulnerabilities from a web application perspective is important for web application security. There are various types of methods for identifying XSS or any vulnerability for that matter, from manual reviews when finding them in the application to automated methods. Each type of method has its own pros and cons. Furthermore, we're also seeing advancements in using machine learning and artificial intelligence (AI) tools to improve detection of XSS vulnerabilities.

9.1. Static Analysis

Static analysis is the act of evaluating the application code without executing it. Static code analysis tools analyze the code for vulnerabilities based on potentially insecure coding techniques and coding patterns that can lead to an XSS.

Advantages: Help identify vulnerabilities early in the development lifecycle, very little impact on the application at runtime.

Limitations: False positives may become an issue. True vulnerabilities may not be identified if they only be evident after the code is executed. Tools like SonarQube, which is created to accelerate static code analysis are very popular and are configurable to indicate potential XSS vulnerabilities [12].

9.2. Dynamic Analysis

Dynamic analysis is when you test the manual inputs of an application while it is executing, it mimics actual real-world attacks in order to find vulnerabilities. Some of the automated tools like the OWASP ZAP and Burp Suite Proxy analyze HTTP requests and responses in real-time while intercepting these communications in order to determine when an application accepts untrusted user input in an insecure manner [13].

Advantages: Effective in identifying vulnerabilities based on an installed application repository, including repository members, such as run time configurations.

Limitations: Testing can often resource intensive and the automated tool needs a working environment in the testing is to be successful.

Dynamic analysis can be useful to isolate DOM-based XSS, which static analyzers traditionally miss.

9.3. Manual Testing

Manual testing is when ethical hackers or security professionals manually test the application for any XSS vulnerabilities. This process would allow for ethical hackers to formulate payloads, put them in the fields provided, and see what, if anything, happens.

Advantages: Effectiveness at finding vulnerabilities that are sophisticated or contextual.

Limitations: Time consuming, requiring skill set only a few have the luxury of having to themselves.

In more cases than not, manual testing can't, and should not be, ignored, especially if you're considering critical business applications where automated testing may be biased in being able to calibrate true positive vulnerabilities from false positives.

9.4. Machine Learning and AI Approaches

Machine learning (ML) and artificial intelligence (AI) have recently entered the forensic tool space and are promising in their potential for identifying XSS attacks. They use models on attack methods, using data sets from attacks and non-attack (benign) behavior to predict XSS vulnerabilities.

Advantages: High degree of accuracy in detecting actual vulnerabilities, once data sets are processed, they can be adjusted to uncover and suspect new attack models.

Limitations: Must have a lot of data sets to teach proper algorithms and require a high amount of CPU [14].

10. Literature review

Alsaffar et al [15] proposed a proposal model to enhance their XSS detection through different advanced techniques for vulnerability detection. Their model recorded 99.47% accuracy, 100% precision, 81% recall and was also effective and efficient compared to other technologies such as Acunetix and XSpider which includes the detection rates. Also, their model incorporates features defined for authorization and authorized vulnerability assessments which adds depth to security assessments. In a piece of work [16], authors declared DeepXSS, a new deep learning unique architecture for XSS attack detection which achieved 99.5% accuracy and 97.9% precision on a real dataset. The authors utilised a decoder that can read nullified payloads, they also used word2vec an algorithm that extracts semantic features from XSS payloads to also create a Long Short Term Memory (LSTM) neural networks to classify. Under a comparison of their method and traditional machine learning models such as ADTree and AdaBoost, DeepXSS produced an improved performance whereby DeepXSS produced an F1 score of 98.7% with an almost perfect ROC curve (AUC 0.98), also, proficiently classifying regular and obfuscated XSS attacks. Irfa'issurur and Josaphat [17] reported on their findings of six machine learning algorithms (KNN, Random Forest, Naïve Bayes, AdaBoost, LightGBM, and XGBoost) with their use of CICIDS2017 and CSE-CICIDS2018 dataset to detect online attacks. Also, they employed SMOTE to deal with data imbalance and dimensionality reduction was achieved through PCA. Their noteworthy findings were their Random Forest model yielding an accurate value of 97.77% accuracy, 84.07% precision, 91.96% recall, and 87.28% F1-score. The research indicated how ensemble learning can be beneficial, ultimately indicating how hyperparameter tuning and real-time network identifiers may assist their research capabilities. The author [18] reported yet again to present an AI-based detection model for XSS attack detection of web applications and further proposed a new feature fusion model called LSTF, which is a combination of LSTM-based temporal features and TF-IDF representations. Their experiments also demonstrated that the Random Forest (RF) model yield a predictive accuracy of 99%, exceeding other machine learning and deep learning models. Validation of their models required k-fold cross validation and hyper parameter tuning, whereby each aspect was demonstrated and evaluated based on performance of the model. They also used Explainable AI (XAI) methods to gain transparency in their decision-making processes. The important message stated was that amalgamating advanced feature engineering with applicable AI models can yield superior detection of XSS attacks to enhance web application security.

The researcher [19] developed an improved LSTM-based model for identifying XSS attacks in Cloud Computing situations. The author's model, CMABLSTM (Character-level Multi-Attention Bidirectional LSTM), improves traditional detection methods by being able to automate feature extraction and handle obfuscation methods effectively. The bi-directional LSTM captures long-term dependencies, and the multi-head attention mechanism allows features to be taken from independent perspectives, thereby improving semantic understanding. Experimentally, the authors obtained an F1 score of 98.71%, which is better than the traditional machine learning and deep learning models. The technique significantly improves detection of XSS attacks, which strengthens the security of cloud-based applications. The author [10] introduced an innovative browser plugin that inserts contextual information into web pages for the user. Unlike traditional content injection systems that are global/static in nature, Context-Auditor takes into account page context and user behavior to choose insertion points and provide relevant information. This works on a dual approach: determining page structure and context, then mapping these to user-specific information by exploiting learned patterns. The research includes experimental evaluations which show its success in providing contextually relevant content. In [20], the researchers provided a new virtual sample generation method to identify cross-site scripting attacks using few-shot learning. They obtained the models using over 210,000 data, with the greatest performance coming from the convolutional neural network (CNN). The method achieved 78.6% accuracy on 2022 CVE samples, far higher than traditional methods and much better generalization on recent XSS attacks.

The paper [2] introduced a deep learning technique for web attacks detection (i.e. Cross-Site Scripting (XSS) and SQL injection) using a Bidirectional Long Short-Term Memory (BiLSTM) network. The authors processed the attack payloads character by character to capture contextual dependencies forward and backward of the attack payload. The authors used word embeddings and normal preprocessing steps and were able to achieve high accuracy in multi-class classification problems. Overall, the experimental results illustrated that the model outperformed traditional detection methods, and they argued that the model was able to detect complex and sophisticated attack patterns. An additional study by J. R. Tadhani [21] proposed a deep learning model to identify, mitigate, and stop Cross-Site Scripting (XSS) and SQL injection (SQLi) attacks against web applications. This research had three separate models, which invoke a hybrid model combining Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks, to create the model that interprets both spatial and temporal information from the input data. The authors trained the model on benchmark datasets and reported that they performed very well with high accuracy and were reporting better performance than the traditional methods to detect the attacks. The authors pointed to that their deep learning model was able to reduce false positive rates, while also being able to identify novel attack patterns, and overall, they felt that deep learning was a significant and important step forward in the proactive security paradigm for web applications.

This research collectively highlights the increasing dependence on machine learning and deep learning methodologies to improve the precision, efficiency, and flexibility of XSS detection systems, with novel ways tackling issues in both web and mobile apps.

11. Challenges in XSS mitigation

XSS mitigation will continue to confront many challenges, many of which reflect the dynamic nature of attack approaches, the complexity of modern web applications, and the difficulty in detecting and preventing their usage. Because attackers constantly evolve their approaches to circumvent mitigation strategies, protecting modern web applications from cross-site scripting attacks can be increasingly difficult. As they improve their methodologies, attackers have developed increasingly sophisticated mechanisms for circumventing mitigation strategies, including community-approved lax handling of security measures in favor of usability and simple approaches to sanitizing input, the development of Content Security Policies (CSP) would also fall under this category. Payload obfuscation, which entails altering or encoding malicious scripts for evasion, has come to the forefront [1]. There are examples of the inability to protect against advanced attacks, such as Steffens et al., [22] as it shows how script gadgets and DOM clobbering techniques undermine even CSPs that are properly configured.

- A significant hurdle in xss (Cross-Site Scripting) attack detection is a lack of full, up to date, labeled datasets, particularly with respect to new, and more advanced attack methods. This inconsistency may make the detection models obsolete and increase the probability of an attack being incorrectly classified. Future research should focus on building full datasets that represent the evolving characteristics of xss threats. Techniques such as active learning, which allows models to learn from the most informative examples and transfer learning, which applies knowledge acquired from similar tasks, will help improve detection accuracy and help models stay current.
- Current literature on xss attack detection shows much higher rates of false positives and false negatives, which can have serious implications either due to unintentional disruption of activity of a benign process or failure to recognize a true threat. Future research should focus on designing more robust and accurate machine learning and deep learning models. This includes investigating more sophisticated feature selection and extraction methods, as well as the application of online learning approaches that allow for model improvements as patterns and attack methods change and evolve, thereby improving potential develop xss detection systems.
- While third-party libraries and external javascript packages can help to accelerate a work stream, they can also greatly expand the attack surface. They can have either an insecure code, not be updated quickly, or be subject to supply-chain attacks. Even well-maintained libraries can become compromised through the introduction of code that can accidentally expose injection vulnerabilities. When developers are not audited extensively, they can end up putting a vulnerability into production in ways unforeseen from simply doing updates or adding features. In the Magecart case, attackers introduced bad JavaScript into third-party packages with known vulnerabilities, including commonly used chat widgets, and analytic products, which were commonly used across thousands of popular websites. Their script would grab sensitive data (even beyond credit card numbers during checkout) and send it to servers owned by the attackers. [8] Victims included British Airways, Newegg, and Ticketmaster.

In the end, the ever-evolving attack mechanisms; the modern web technologies which will make the detection and deterrence of XSS more complex; the difficulty in distinguishing between attacker input and legitimate input makes the challenge of XSS mitigation complex and ongoing. As attackers learn to exploit their attack vectors in new ways, web

developers/owners and security professionals will continually need to strive towards rapid, and more importantly continuous vigilance, in both their proactive and reactive methods of prevention from XSS threats. The work presented in this section demonstrates that despite significant strides made in the furtherance of XSS mitigation strategies, challenges to prevent modern web applications from XSS threats continue.

12. Conclusion

XSS (cross-site scripting) continues to be a prevalent and growing problem in today's web applications and can seriously endanger the confidentiality of data, trust and confidence of users, and integrity of the application. This article reviewed the history and classification of XSS attacks and the trends of increasing complexity and subtlety of newer forms. Traditional detection and prevention methods, while helpful, have failed to adapt to the evolving and hidden aspects of modern XSS payloads. Employing machine learning (ML) and deep learning (DL) methods provide avenues for automated detection and mitigation for XSS vulnerabilities of all forms, relying on the data to explore the exploited dimensions of web pages, data extraction methods, and patterns inherent in XSS attacks to overcome signature-based methodologies. Some obstacles remain concerning data quality, model generalization, and the speed of model application in different web contexts. Future work must focus on the building of robust, interpretable, adapting models to effectively mitigate zero-day XSS type threats while minimizing false positives. Furthermore, the coordination of intelligent approaches with safe software development lifecycles, and real-time monitoring systems could enhance overall web security resilience. As XSS techniques continue to evolve, response options need to evolve too, relying on transdisciplinary developments in the fields of cybersecurity, machine learning, and secure software practices.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] OWASP Foundation, "OWASP Top Ten." [Online]. Available: https://owasp.org/www-project-top-ten/

[2] A. A. Farea, C. Wang, E. Farea, and A. B. Alawi, "Cross-site scripting (XSS) and SQL injection attacks multi-classification using bidirectional LSTM recurrent neural network," in Proc. 2021 IEEE Int. Conf. Progress Informatics Comput. (PIC), Shanghai, China, Dec. 2021, pp. 358–363, doi: 10.1109/PIC54163.2021.9777139.

[3] H. Yan et al., "Cross-site scripting attack detection based on a modified convolution neural network," Front. Comput. Neurosci., vol. 16, pp. 981739, 2022, doi: 10.3389/fncom.2022.981739.

[4] A. Hannousse, S. Yahiouche, and M. C. Nait-Hamoud, "Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey," Computer Science Review, vol. 52, pp. 100634, 2024, doi: 10.1016/j.cosrev.2023.100634.

[5] M. Liu, B. Zhang, W. Chen, and X. Zhang, "A survey of exploitation and detection methods of XSS vulnerabilities," IEEE Access, vol. 7, pp. 182004-182016, 2019, doi: 10.1109/ACCESS.2019.2960449.

[6] S. Bensalim, D. Klein, T. Barber, and M. Johns, "Talking about my generation: Targeted DOM-based XSS exploit generation using dynamic data flow analysis," in Proc. 14th Eur. Workshop Syst. Secur. (EuroSec), Virtual Event, 2021, pp. 27–33, doi: 10.1145/3447852.3458718.

[7] J. Kim and J. Park, "Enhancing security of web-based IoT services via XSS vulnerability detection," Sensors, vol. 23, no. 23, pp. 9407, 2023, doi: 10.3390/s23239407.

[8] S. McNally and K. Curran, "Web application vulnerabilities," in 2024 8th Int. Symp. Comput. Sci. Intell. Control (ISCSIC), Zhengzhou, China, 2024, pp. 359-366, doi: 10.1109/ISCSIC64297.2024.00081.

[9] M. Golinelli, F. Bonomi, and B. Crispo, "The nonce-nce of web security: An investigation of CSP nonces reuse," in Proc. Eur. Symp. Res. Comput. Secur. (ESORICS), Cham, Switzerland: Springer Nature, Sep. 2023, pp. 459–475.

[10] F. Kalantari, M. Zaeifi, T. Bao, R. Wang, Y. Shoshitaishvili, and A. Doupé, "Context-Auditor: Context-sensitive content injection mitigation," in Proc. 25th Int. Symp. Res. Attacks, Intrusions and Defenses (RAID), New York, NY, USA, 2022, pp. 431–445, doi: 10.1145/3545948.3545992.

[11] N. Kshetri, D. Kumar, J. Hutson, N. Kaur, and O. F. Osama, "AlgoXSSF: Detection and analysis of cross-site request forgery (XSRF) and cross-site scripting (XSS) attacks via machine learning algorithms," in 2024 12th Int. Symp. Digital Forensics Security (ISDFS), San Antonio, TX, USA, 2024, pp. 1-8, doi: 10.1109/ISDFS60797.2024.10527278.

[12] P. Ferrara, A. K. Mandal, A. Cortesi, and F. Spoto, "Static analysis for discovering IoT vulnerabilities," Int. J. Softw. Tools Technol. Transfer, vol. 23, no. 1, pp. 71–88, Feb. 2021, doi: 10.1007/s10009-020-00592-x.

[13] M. Albahar, D. Alansari, and A. Jurcut, "An empirical comparison of pen-testing tools for detecting web app vulnerabilities," Electronics, vol. 11, no. 19, pp. 2991, 2022, doi: 10.3390/electronics11192991.

[14] B. Njie and L. Gabriouet, "Machine learning for cross-site scripting (XSS) detection: A comparative analysis of machine learning models for enhanced XSS detection," M.S. thesis, Dept. Comput. Sci., Univ., 2024.

[15] M. Alsaffar et al., "Detection of web cross-site scripting (XSS) attacks," Electronics, vol. 11, no. 14, pp. 2212, 2022, doi: 10.3390/electronics11142212.

[16] Y. Fang, Y. Li, L. Liu, and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning," in Proc. 2018 Int. Conf. Comput. Artif. Intell. (ICCAI), Chifeng, China, 2018, pp. 47–51, doi: 10.1145/3194452.3194469.

[17] M. Irfa'issurur and B. P. Josaphat, "Machine learning for cybersecurity: Web attack detection (brute force, XSS, SQL injection)," InPrime: Indones. J. Pure Appl. Math., vol. 7, no. 1, pp. 1–15, 2025.

[18] F. Younas et al., "An efficient artificial intelligence approach for early detection of cross-site scripting attacks," Decision Analytics Journal, vol. 11, pp. 100466, 2024, doi: 10.1016/j.dajour.2024.100466.

[19] X. Li et al., "An LSTM based cross-site scripting attack detection scheme for cloud computing environments," J. Cloud Comput., vol. 12, no. 1, Jun. 2023, doi: 10.1186/s13677-023-00483-x.

[20] D. Lu and L. Liu, "Research on cross-site scripting attack detection technology based on few-shot learning," in 2023 IEEE 6th Inf. Technol., Networking, Electron. Autom. Control Conf. (ITNEC), Chongqing, China, 2023, pp. 1425-1429, doi: 10.1109/ITNEC56291.2023.10082596.

[21] J. R. Tadhani et al., "Securing web applications against XSS and SQLi attacks using a novel deep learning approach," Sci. Rep., vol. 14, pp. 1803, 2024, doi: 10.1038/s41598-023-48845-4.

[22] M. Steffens, C. Rossow, M. Johns, and B. Stock, "Don't trust the locals: Investigating the prevalence of persistent client-side cross-site scripting in the wild," in Proc. 2019 Network and Distributed System Security Symp., 2019.