An Analysis of Key Tools for Detecting Cross-Site Scripting Attacks on Web-Based Systems

Harshad Kissoon and Girish Bekaroo(✉)

Middlesex University Mauritius, Flic-en-Flac, Mauritius
HK759@live.mdx.ac.uk, g.bekaroo@mdx.ac.mu

Abstract. During the previous few years, there has been an escalating number of cyberattacks against web-based systems, that adversely resulted in significant data breaches, losses and reputational damages for businesses. Among these cyberattacks, cross-site scripting attacks, also known as XSS attacks, gained significant attention, which makes is imperative to explore detection methods. Taking cognizance of this issue, this paper reviews and analyses key XSS attack detection tools. To accomplish this objective, the study meticulously examines six distinct tools, notably, web application firewalls, intrusion detection systems, dedicated AI-driven tools, SIEM Systems, honeypots and browser extensions, and provides critical insights on their effectiveness. From our key findings, web application firewalls, AI-driven tools and browser extensions emerged as crucial components for detecting different kinds of XSS attacks, showcasing notable effectiveness. However, it is important to note that the efficacy of these tools may vary depending on factors such as application configurations and update frequency, among others.

Keywords: Cross-Site Scripting · XSS Attacks · Web Application Firewalls · Intrusion Detection Systems · Honeypots · SIEM · AI-Driven Tools · RAST · Browser Extensions

1 Introduction

During the previous years, web-based cyber-attacks have known a consequent increase, representing around 26% of all infractions in 2022 [1]. From 2004 to 2021, the web industry has lost around 9.9 billion records where only in 2019, during the outbreak of COVID 19, more than 1 billion records were lost from big enterprises such as Microsoft and Facebook [2]. Cyber-attacks are expected to increase and become worse throughout the coming years, with a variety of attacks facing web platforms including cross-site scripting, zero-day attack and distributed denial of service attacks, among others [3]. Among these attacks, cross-site scripting attacks, also known as XSS attacks, have gain significant attention recently, where around 40% of web attacks in 2019 involved the use of XSS attacks [4]. Cross-site scripting attacks involves the injection of malicious scripts

A. Seeam et al. (Eds.): InterSol 2023, LNICST 541, pp. 3–14, 2024.
https://doi.org/10.1007/978-3-031-51849-2_1

into the code of a trusted application or website. XSS has been a key security issue in web apps, which can also be executed by script kiddies as the attacks are almost the same and the scripts used during previous attacks can be slightly adapted and used during future attacks. In 2019, the popular game, Fortnite, was prey to the XSS attack combined with a single sign on issue which enabled attackers to steal the in-game currency, player's conversation as a reconnaissance through a fake dummy login page [5]. Consequently, the attacker had access to the player's account which contained personal information, game chats and friend lists. In addition, some users who have their credit cards linked to the game, saw that in-game currency was purchased and then sold for real cash. This attack damaged the reputation of the impacted company, thereby leading to a decrease in its number of users. As such, it is important to effectively detect XSS attacks in a timely manner so that consequences can be minimised. However, limited studies have been done to investigate and analyse between different XSS detection tools. Hence, to address the limitations in published literature, this paper compares and analyses different XSS detection tools. The insights derived as part of the study can be utilized by the web-community and researchers in the endeavour to further enhance web security against XSS attacks.

This paper is structured as follows: In the next two sections, a background on Cross-Site Scripting is provided followed by related works on the review and analysis of such tools in published literature. Then, the methodology used for achieving the purpose of this paper is discussed in Sect. 4. The review and analysis of the key tools for identifying cross-site scripting attacks are provided in Sects. 5 and 6. Conclusion of the paper is finally provided in Sect. 7.

2 Background: An Overview of XSS Attacks and Their Consequences

The term cross-site scripting was introduced by Microsoft security engineers in the year 2000 as hackers utilized JavaScript to run an invisible website within a frame of a legitimate website that would enable them to retrieve data entered on the legitimate website and execute malicious code. Thereafter, it became a popular web-application exploit. An illustration of the key steps involved in a typical XSS attack is provided in Fig. 1.

As depicted in Fig. 1, an intruder attempts to find and exploit a vulnerability within a targeted website that can potentially return malicious JavaScript to users. The attacker then writes (or uses) JavaScript-based malicious codes that can be injected in browsers to exploit the targeted vulnerability that would enable "keylogger with backdoor functionalities" to be installed without the targeted user notices. Eventually, the attacker will be able to gain access to the computer and capture cookies which is exchanged between the browser and server of data. As such, by exploiting XSS vulnerability, attackers are able to impersonate users, perform tasks through victim's account, gain access to sensitive data of a user (e.g., credentials, credit card details) or even deface websites. The key types of XSS attacks are described as follows and are summarised in Table 1:

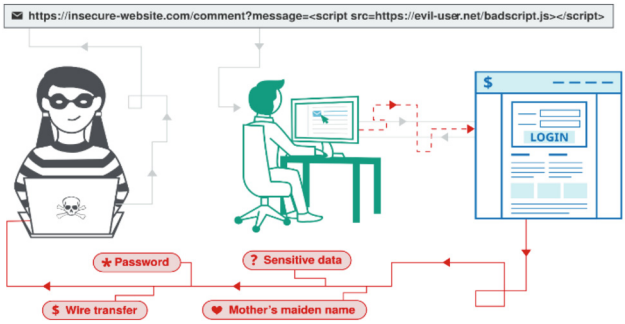

Fig. 1. Cross-Site Scripting Execution [6]

- *Reflected XSS*
  In reflected XSS, which is also known as non-persistent XSS, the code injected by the attacker does not reside on the web server. In reflected XSS, attackers send malicious links to targeted users using different medium such as email, messengers or by embedding the link on other websites, blogs, forums, etc. When a user clicks on the malicious link, the injected code goes to the attacker's web server, and the malicious script is sent back to the victim's browser. Eventually, the code is executed on the victim's browser and the attacker is able to fulfil the intended goal (e.g., stealing confidential information) [7].
- *Stored XSS*
  Stored XSS is also known as persistent XSS attacks and as compared to reflected XSS, the malicious code injected in stored XSS is permanently stored on the servers of targeted websites. Attackers use that method to inject harmful content also known as "payload" written in JavaScript into a vulnerable web app. The scripts inserted are stored permanently on the targeted server or memory. Once any user accesses the information via the web app by knowingly or unknowingly clicking some links, the malicious script is executed, and the attacker is able to access confidential information of the user [8].
- *DOM-based XSS*
  In the Document Object Model (DOM) based XSS, the attack is executed at the client side as compared to the two XSS attacks described above. In this type of XSS attack, the DOM environment in the targeted user's browser is modified so that it crashes or works differently [9]. As such, the page loaded does not change but the code on the client is executed differently due to the modifications within the DOM environment.
- *Induced XSS*
  Induced XSS is the least common type of XSS as compared to the others and is applicable to web applications where HTTP Response Splitting is vulnerability is present on web servers [7]. Attackers exploit this vulnerability by manipulating the HTTP header of the server's response in order to attempt to inject scripts and access confidential information of the victim.

XSS attacks pose various negative issues to both businesses and end users of IT systems [4]. Amongst, data theft is a key issue where successful XSS attacks can enable

Table 1. Summary of the types of XSS attacks.

Type of XSS Attack	In Database	Requesting HTTP	DOM Built in Browser
Reflected		Yes	Yes
Stored	Yes	Yes	Yes
DOM-based			Yes
Induced		Yes	Yes

attackers to steal sensitive data of businesses and users which may include login credentials, personal data and financial data, among others. In addition, XSS attacks can potentially enable attackers to run arbitrary and malicious codes within a victim's browser, leading to unauthorized actions on behalf of the user. Furthermore, XSS can be utilized by attackers to modify contents of legitimate websites whereby defacing them, thus adversely impacting trust and reputation.

3 Related Works

Although numerous studies have been conducted pertaining to detection of web-based attacks, limited work has been done pertaining to the comparisons of detection of Cross-Site Scripting. In a previous study [10], three types of detection to find weaknesses in XSS attacks in web browsers were discussed, namely, static analysis, dynamic analysis, and hybrid analysis. Static analysis involves finding vulnerabilities during the development and testing phases before the web application is deployed to live. Dynamic analysis is the process of checking for any vulnerabilities when the program has been deployed to the live environment and testing is done in a way to simulate attack processes same as an intruder would do. Hybrid analysis is a combination of both static and dynamic analysis to resolve issues found in the static analysis process and provide a better secured web app. Even though the work reviewed these three XSS attack detection types, no comparisons were made between these three approaches. In another paper [11], detection techniques and counter measures against XSS attacks were reviewed. However, reviews were brief, and no comparisons were made.

Another study [12] focused on the detection of SQL injection and XSS vulnerabilities in open-source JAVA web applications using static analysis. However, the approach used for mitigating XSS after the tests resulted in high false positive rate. Moreover, in another study [13], the dynamic analysis approach in JavaScript was used to detect XSS vulnerability in real time. The call graph method was used for finding issues and to derive outputs which were accurate overall. However, both studies focused only on certain XSS vulnerabilities and as such, the reliability of findings are questionable. Another study [14] used the hybrid analysis method to focus on XSS vulnerabilities. For the classification of nodes, static analysis was used and for targeting nodes which are at risk, dynamic analysis was approached. Tests were performed on six web applications built using PHP and the authors did come with proper results. However, the mixture of dynamic and static analysis was found to adversely impact reliability. As such, existing

studies performed limited comparison of XSS detection approaches and hence, this study is relevant to undertake.

4 Methodology

The primary source of this paper has been carried out by filtering research databases like previous studies done in this field and some reports published by companies which faces such issues and their loss encountered [15]. The research databases filtered include IEEE Xplore Digital Library, ACM Digital Library, Research Gate, Google Scholar, Science Digest, and some websites such as OWASP and ENISA were explored. The key terms used in the searching process include "XSS detection tools" and "Cross-Site scripting attack detection tools" and "injection detection tools". Following a pool of 54 results, filtering was conducted to assess relevance and meant the context of Cross-Site Scripting. The selected tools are discussed in the next sections.

5 Cross-Site Scripting Detection Tools

Using the methodology described in the previous section, the selected key vulnerability detection tools are discussed as follows:

5.1 Web Application Firewalls (WAFs)

WAFs can be in the forms of hardware or software, and are typically positioned between a web application and the client to filter and monitor incoming and outgoing HTTP traffic [16]. Such tools utilize different approaches in their endeavour to detect XSS attacks. A key approach involves the use of pattern matching algorithms to analyse the content of HTTP requests and responses to look for specific patterns, keywords, or HTML tags commonly associated with XSS attacks. If a request or response contains suspicious patterns, the WAF can block or sanitize the content. Similarly, web application firewalls can also perform contextual analysis, to analyse the context of web requests and responses in order to detect anomalies and suspicious behaviour. For example, WAFs can check if user-supplied data is used inappropriately within HTML tags or if client-side scripts are modified or injected. By understanding the expected behaviour of web applications, WAFs can identify deviations that indicate XSS attacks.

5.2 Intrusion Detection Systems

Another tool used to detect XSS attacks is intrusion detection systems (IDS) [17, 18]. This approach involves capturing network packets that are in movement and then relevant TCP packets that are filtered and selected to detect irregularities. In other words, IDS can help to detect XSS attacks by analysing network traffic and identifying patterns that match known attack signatures to eventually alert system administrators when it detects suspicious activity that may indicate an XSS attack is in progress. To detect XSS attacks specifically, IDS can monitor for suspicious behaviour, such as unexpected input, HTML

tags, and JavaScript code. IDS can also analyse HTTP headers and cookies for signs of tampering or injection of malicious code. A previous study [17] applied signature-based detection method on the packets to detect XSS attacks. As part of the same study, SNORT IDS, a tool from Cisco, has been used to monitor incoming and outgoing packet on the network with some additional rules configured so that when an intrusion is detected, an entry alert is written in a snort alert file. When an attacker is attacking a web-based system, the SNORT application analyses the application layer from OSI model to detect if the attacker is modifying the tags of the html and the implementing some malicious scripts in the browser to gain access. Thus, the rules defined in SNORT can detect the attacks and add them in a log file. This detection technique can detect all types of XSS attacks categories. For each rule added in the application, the attacks will be intercepted and logged, however, this method works only on the application layer and in case if no rule has been added for an attack technique, the attacker will be able to cause high damage. Similarly, behaviour-based or anomaly-based detection can also be used, or even integration of hybrid approaches.

5.3 Dedicated AI-Driven Tools

Many recent studies focused on the detection of XSS attacks involving the application of artificial intelligence (AI), machine learning and deep learning techniques to create a range of customised tools. These AI-driven tools can be stand-alone ones that are dedicated to detect XSS attacks in real-time or can be integrated within Runtime Application Security Testing (RAST) tools, that integrate with the application during runtime and monitor its behaviour. These tools can detect and even prevent XSS attacks by analysing the actual execution flow and identifying potential vulnerabilities. Such approaches target accurate real-time detection and isolation of attacks without any adverse effect on the running web app. As part of this endeavour, various studies involved the use of genetic algorithms, hybrid approaches, neural network models and dynamic analysis-based approaches, among others, to enhance the detection accuracy of XSS attacks [19]. For instance, a previous study [20] applied Convolutional Neural Network (CNN) to detect XSS attacks using a typical architecture provided in Fig. 2. The process involved removal of letters from characters in order to produce a new matrix and execute the CNN model process.

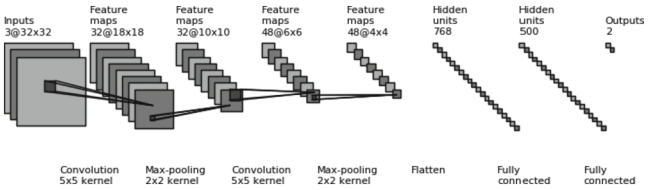

Fig. 2. Architecture of CNN model

The accuracy of different algorithms to detect XSS attacks were also compared as shown in Table 2 where all three algorithms achieved pretty good scores, where the CNN method got above 95% of accuracy and false alarms were minimum.

Table 2. Accuracy of Machine Learning Algorithms.

Algorithm	Score (% Accuracy)
CNN	98.59
RNN	90.25
Logistic Regression	82.12

5.4 Security Information and Event Management (SIEM) Systems

As compared to the AI-driven detection tools discussed above, Security Information and Event Management (SIEM) are more comprehensive security solutions that integrate security information management (SIM) and security event management (SEM) features. These solutions collect, analyse, and correlate security events and logs from various sources across an organization's IT infrastructure, with the goal to provide real-time visibility into security events, facilitate threat detection and response, and support compliance with security regulations. The key features of SIEM include log collection and management, event correlation, threat detection, real-time monitoring, incident response and compliance management, among others. In order to detect XSS attacks, SIEM systems can be configured to analyse logs and network traffic for suspicious patterns and indicators of XSS activity [21]. In addition, SIEM rules and alerts can be set up to trigger notifications or take automated actions when XSS attacks are detected.

5.5 Honeypots

Honeypots, which are tools designed to trap attackers during cyberattacks, can also be utilized in the detection of XSS attacks [22]. For instance, a honeypot can be set up as a web application that contains intentionally vulnerable code, which is designed to be exploited by attackers attempting to inject malicious scripts. The honeypot is then monitored for any suspicious activity, such as unexpected script injections or attempts to access sensitive information. By deploying honeypots on different systems and networks, security professionals can build a comprehensive picture of the XSS threat landscape and develop more effective countermeasures to protect against these attacks. This information can be used to improve the security of real web applications and to develop more effective intrusion detection and prevention mechanisms. A previous study [4] discussed the use of such tools to learn about the pattern and behaviour of attackers during XSS attacks. As part of the study, a low-interaction honeypot was utilized that emulates vulnerabilities that can be exploited through XSS. The key benefit of using this approach is that activities of an attacker can be logged, while also potentially exposing his identity. Nevertheless, this approach also faces the challenges for effectively detecting attackers if they are not successfully lured into the trap or if they hide their identities.

5.6 Browser Extensions

A browser extension is a small program that adds additional features to the browser and these tools can be downloaded from browser stores. Even though there have been

various extensions reported for triggering XSS attacks, such tools can also be utilized to detect such attacks and alert the user in a timely manner. These browser extensions attempt to detect XSS attacks by analysing the content of web pages for potential XSS vulnerabilities and in the process, different approaches are used [23]. For instance, some extensions work by injecting a script into the page, which can then monitor user input and outgoing requests for suspicious activity. Some extensions may also use heuristics or pattern matching algorithms to identify potential XSS payloads in the page's HTML, JavaScript, or other embedded content. If a vulnerability is detected, the extension may alert the user and block the malicious script from executing. One such example of an extension is Counter XSS extension for Google Chrome, which enables users to scan web pages for potential XSS vulnerabilities and displays a warning if it finds any suspicious code. Similarly, the "NoScript" extension for Firefox, blocks JavaScript, Java, Flash, and other active content from running on web pages unless the user specifically allows it. This helps prevent XSS attacks by blocking the execution of malicious scripts that could be injected into web pages.

6 Analysis of XSS Attack Detection Tools

The detection tools were critically analysed on how they effectively detect the several types of XSS attacks. Effectiveness was evaluated based on criteria, such as its ability to detect diverse types of XSS attacks, its accuracy in distinguishing between malicious and benign code, its coverage of different types of XSS attacks, such as stored XSS, reflected XSS, or DOM-based XSS, as discussed earlier in this paper. In order to compare and analyse the XSS attack detection tools based on the two criteria above, published papers and literature was thoroughly reviewed and analysed. Results are summarised in Table 3 and findings are discussed in the next sections:

Table 3. Effectiveness of Detection Tools

XSS Detection Tool	Reflected	Stored	DOM-based	Induced
Web Application Firewalls	High	High	Medium	Medium
Intrusion Detection Systems	Medium	Medium	Low	Low
Dedicated AI-Driven Tools	High	High	High	Medium
SIEM Systems	Medium	Medium	Low	Low
Honeypots	Low	Low	Low	Low
Browser Extensions	High	Medium	High	Medium

6.1 Effectiveness of Detection Tools

Among the tools reviewed, WAFs and dedicated AI-tools were found to have relatively high effectiveness in detecting XSS attacks [16]. WAFs are particularly designed to

protect web applications from attacks, including XSS. These tools provide a dedicated defense layer and offer comprehensive features like signature-based detection, pattern matching, content inspection, and behaviour monitoring. Even though WAFs are typically used on the server side of web applications, they can directly or indirectly detect stored, reflected, DOM-based and induced XSS attacks. Stored XSS attacks can be detected by analysing the content submitted to the web application and the subsequent delivery of that content to other users and reflected XSS can be detected by inspecting HTTP request parameters and the corresponding server responses to identify malicious input that may be reflected back to the user's browser. In addition, WAFs can also detect induced XSS attacks by identifying and blocking suspicious patterns or content that may attempt to deceive users into executing malicious JavaScript code. Furthermore, even though it is more challenging for WAFs to directly detect DOM-based XSS attacks which requires analysis of JavaScript code execution flow and dynamic manipulation of the DOM, the use of advanced techniques such as heuristics, static analysis, and runtime monitoring can help to identify malicious JavaScript behaviours and block them. As such, WAFs can effectively detect and prevent a range of XSS attacks, thus making them highly effective tools for XSS detection. Similarly, AI-driven tools have similar coverage as WAFs, although there are varying approaches and algorithms used in the process of identifying XSS attacks. Dedicated AI-Driven tools and RAST tools are specifically implemented with the aim analyse runtime behaviour of web applications in order to identify security issues, including XSS and are as such highly effective tools [20]. In addition, a range of algorithms and techniques can be employed to also strengthen detection against a range of XSS attacks. As compared to WAFs, machine learning techniques can also be integrated on the client-side within browser extensions to more effectively detect DOM-based or induced XSS attacks. Nevertheless, data is required to be able to train models and improve accuracy of detection.

Furthermore, browser extensions have the ability to effectively detect the key types of XSS attacks but run on client-side. In other words, browser extensions are generally more focused on user-side protection rather than comprehensive server-side XSS detection. For instance, these tools can analyse the content and source codes of web pages (HTML tags or JavaScript) in real-time and detect instances where user-supplied data is directly inserted into the HTML response without proper sanitization or encoding, to eventually flag such incidents and raise alerts. Moreover, browser extensions can monitor user interactions with web applications and identify instances where user-generated content is stored and displayed on subsequent visits [23]. By scanning the stored content and detecting any suspicious or untrusted script injections, extensions can block the execution of malicious code. Furthermore, these tools can analyse and keep track of changes to the DOM in real-time, detecting any unauthorized modifications or dynamic script insertions that could lead to XSS vulnerabilities. Finally, whilst browser extensions may not directly detect induced XSS attacks, they may warn users about potentially malicious links or suspicious website behaviour, to indirectly detect induced XSS attacks.

On the other hand, IDS and SIEM systems were found to have medium effectiveness to low effectiveness in the detection of different types of XSS attacks. IDS and SIEM systems work by identifying certain indicators or patterns related to XSS attacks whereby assisting in detecting XSS attacks either by analysing logs, network traffic, or event

data for suspicious or anomalous behaviour that may indicate an XSS attack. Also, distinguishing between the several types of XSS attacks may not be the focus of IDS and SIEM systems. For instance, although intrusion detection systems can identify certain patterns and anomalies indicative of XSS attempts, they may have limitations in detecting sophisticated or context-specific XSS attacks as compared to dedicated web application security solutions such as WAFs and AI-driven tools. As such, reflected, DOM-based and induced XSS attacks are relatively more challenging to detect using such tools. Similarly, SIEM are more suitable for centralized monitoring and correlation of security events rather than real-time XSS detection. These tools rely on the availability of relevant data and proper configuration to be able to effectively detect XSS attacks and similar to IDS, directly detecting reflected, DOM-based and induced XSS attacks are more challenging.

Finally, honeypots were found to have relatively lower effectiveness as compared to others because of their core focus. The key purpose of honeypots is to lure attackers and gather intelligence on attackers rather than serving as a dedicated XSS detection tool to identify a broad range of XSS attacks. In addition to findings highlighted, it is also important to note that effectiveness of the tools may vary depending on various factors such as the specific tool being used, their configurations, implementations, and update frequency, among others. As such, it is recommended to use a combination of tools and techniques in order to further improve the effectiveness to detect XSS attacks in real-time.

6.2 General Discussions

Overall, analysis showed that AI-driven tools, WAFs and browser extensions are key tools involved in the detection of XSS attacks with relatively higher effectiveness as these tools directly focus on detection of such kinds of attacks. Also, browser extensions are useful tools in the detection of XSS attacks running on the client side as these tools are also easy to install and can potentially detect a range of XSS attacks. Among the various types of XSS attacks, tools have been focusing on directly detecting stored and reflected XSS since these can particularly cause most damages, especially stored XSS attacks. On the other hand, detection tools seem to focus less on induced XSS attacks as these are the least common ones, although the tools attempt to indirectly identify such attacks.

Nevertheless, it is important to note that effectiveness is expected to vary from tool to tool. For instance, there are WAFs available on the market that are expected to be more effective and having more user-friendly interfaces as compared to other WAFs and studying the different tools in practice was beyond the scope of this study. As such, a key limitation of this study is that findings were based on published papers and could be extended by practically using and assessing key tools available on the market in order to derive more critical insights. In addition, more quantitative approaches could be adopted to measure effectiveness. This would help to derive further insights and even better help in decision making related to selection of XSS tools for adoption in practice.

7 Conclusion

This paper reviewed and analysed the effectiveness of six tools to detect different types of XSS attacks on web-based systems. The tools studied are web application firewalls, intrusion detection systems, dedicated AI-driven tools, SIEM systems, honeypots and browser extensions. Among these tools, web application firewalls, dedicated AI-driven tools and browser extensions were found to have a relatively high effectiveness, with the ability to directly or indirectly the different types of XSS attacks studied in this paper, notably stored, reflected, DOM-based and induced XSS attacks. IDS and SIEM systems were found to be relative less effectiveness as these tools work by identifying certain indicators or patterns related to XSS attacks whereby assisting in detecting XSS attacks either by analysing logs, network traffic, or event data for suspicious or anomalous behaviour that may indicate an XSS attack. Finally, honeypots were identified as the least effective tool where their primary focus involves luring and gathering details about the attacker, rather than differentiating between the different kinds of XSS attacks. In addition, it is important to highlight that the effectiveness of XSS attack tools may vary depending on various factors such as the specific applications being used, application configurations, update frequency, and design principles employed. As such, in order to more effectively detect XSS attacks a combination of tools and techniques are recommended. As such, in terms of future works, limitations of this study could be addressed whereby investigating a range of XSS attack detection tools while practically using them in order to quantitatively analyse effectiveness of the tools whereby deriving more critical insights.

References

1. Caitlin, J.: 50 web security stats you should know in 2022. Expert Insights **24** (2022). https://expertinsights.com/insights/50-web-security-stats-you-should-know/. Accessed 12 Nov 2022
2. Brooks, C.: Alarming cyber statistics for mid-year 2022 that you need to know. Forbes (2022). https://www.forbes.com/sites/chuckbrooks/2022/06/03/alarming-cyber-statistics-for-mid-year-2022-that-you-need-to-know/?sh=1f079b247864. Accessed 13 Nov 2022
3. Singh, A., Sharma, A., Sharma, N., Kaushik, I., Bhushan, B.: Taxonomy of attacks on web based applications. In: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) (2019)
4. Rodríguez, G., Torres, J., Flores, P., Benavides, D.: Cross-site scripting (XSS) attacks and mitigation: a survey. Comput. Netw. **166**, 106960 (2020)
5. Dizdar, A.: What is XSS? Impact, types, and prevention. Bright Security (2022). https://brightsec.com/blog/xss/. Accessed 11 Nov 2022
6. PortSwigger. Cross-site scripting. PortSwigger (2022). https://portswigger.net/web-security/cross-site-scripting. Accessed 10 Jan 2023
7. Malviya, V., Saurav, S., Gupta, A.: On security issues in web applications through cross site scripting (XSS). In: 2013 20th Asia-Pacific Software Engineering Conference (APSEC) (2013)
8. Abazi, B., Hajrizi, E.: Practical analysis on the algorithm of the cross-site scripting attacks. In: 2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP) (2022)

9. OWASP. DOM Based XSS. OWASP (2022). https://owasp.org/www-community/attacks/DOM_Based_XSS. Accessed 28 Nov 2022
10. Marashdih, A., Zaaba, Z.: Cross site scripting: detection approaches in web application. Int. J. Adv. Comput. Sci. Appl. **7**(10) (2016)
11. Madhusudhan, R.: Cross channel scripting (XCS) attacks in web applications: detection and mitigation approaches. In: 2018 2nd Cyber Security in Networking Conference (CSNet) (2018)
12. Shar, L., Tan, H.: Automated removal of cross site scripting vulnerabilities in web applications. Inf. Softw. Technol. **54**(5), 467–478 (2012)
13. Toma, T., Islam, M.: An efficient mechanism of generating call graph for JavaScript using dynamic analysis in web application. In: 2014 International Conference on Informatics, Electronics & Vision (ICIEV) (2014)
14. Shar, L., Tan, H., Briand, L.: Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis. In: 2013 35th International Conference on Software Engineering (ICSE) (2013)
15. Veerabudren, K., Bekaroo, G.: Security in web applications: a comparative analysis of key SQL injection detection techniques. In: 2022 4th International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM) (2022)
16. Garn, B., Lang, D., Leithner, M., Kuhn, D., Kacker, R., Simos, D.: Combinatorially xssing web application firewalls. In: 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (2021)
17. Gupta, K., Singh, R., Dixit, M.: Cross site scripting (XSS) attack detection using intrustion detection system. In: 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) (2017)
18. Frenz, C., Yoon, J.: XSSmon: a perl based IDS for the detection of potential XSS attacks. In: 2012 IEEE Long Island Systems, Applications and Technology Conference (LISAT) (2012)
19. Kaur, J., Garg, U., Bathla, G.: Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. Artif. Intell. Rev. 1–45 (2023)
20. Bhardwaj, A., Chandok, S., Bagnawar, A., Mishra, S., Uplaonkar, D.: Detection of cyber attacks: XSS, SQLI, phishing attacks and detecting intrusion using machine learning algorithms. In: 2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT) (2022)
21. Kim, J., Kwon, H.: Threat classification model for security information event management focusing on model efficiency. Comput. Secur. **120**, 102789 (2022)
22. Rahul, S., Vajrala, C., Thangaraju, B.: A novel method of honeypot inclusive WAF to protect from SQL injection and XSS. In: 2021 International Conference on Disruptive Technologies for Multi-disciplinary Research and Applications (CENTCON) (2021)
23. Gupta, S., Gupta, B.: XSS-immune: a Google chrome extension-based XSS defensive framework for contemporary platforms of web applications. Secur. Commun. Netw. **9**(17), 3966–3986 (2016)