International Journal of Multiphysics
Volume 18, No. 3, 2024
ISSN: 1750-9548

The Web applications Cross Site Scripting Attacks and Preventions Using Machin learning Technique

Eman Ibrahim Alyasin[1] , Oguz Ata[2], Bilal A Ozturk[3]
[1.2] Department of E.C.E., Institute of Science, Altinbas University, Istanbul, Turkey
[3]Istanbul Aydin University, Faculty of Engineering . Istanbul, Turkey
Email; emanalyasin20@gmail.com . 203720791@ogr.altinbas.edu.tr, oguz.ata@altinbas.edu.tr

Abstract. Web applications are utilized everywhere these days to share services and data online. Because companies deal with sensitive data, hackers have found them attractive targets. Vulnerabilities persist despite the numerous security procedures we've created to safeguard these applications. Major security issues have been identified in web applications used by various organizations, such as banks, healthcare providers, finance companies, and retail businesses. Cross-site scripting (XSS) attacks are one of the most significant issues, according to a report from White Hat Security. These attacks enable hackers to execute harmful programs on a user's web browser, resulting in issues such as the theft of data, cookies, passwords, and credit card numbers. This study focuses on the primary weaknesses present in contemporary web applications, particularly XSS attacks. We go over the many kinds of XSS attacks, provide instances from the real world, and describe how they operate. We also examine defenses against these attacks, discussing what works and what doesn't.

Keywords: Cross Site Scripting , Web attacks, Machin learning XSS Mitigations

1.Introduction

Whether it is to gather information or complete various duties, the Internet is now an integral part of people's daily lives. Web applications are the foundation of these online activities, offering access to nearly all forms of information. Web-based services, including e-commerce, banking, transportation, webmail, and blogs, are now accessible to businesses. This prominence also draws the attention of hackers interested in exploiting these applications' vulnerabilities. According to a 2007 survey, 70% of web applications are susceptible to hacking. As these applications can be accessed from any location on Earth, they are even more alluring targets.

The objective of hackers is to exploit system vulnerabilities to acquire confidential data. Cross-site scripting (XSS) is a prevalent vulnerability in dynamic web applications. The Open Web Application Security Project (OWASP) has identified web application vulnerabilities as a significant target for hackers, with many attacks falling under the injection category. As emphasized by OWASP, it is crucial to concentrate on XSS.XSS attacks alter an HTML element's logic, semantics, or syntax by inserting new keywords or operators. This type of attack is triggered by the absence of appropriate input validation, which enables attackers to incorporate malicious code into web applications. Social networking sites and other web applications are frequently targeted. Attackers exploit this vulnerability by exploiting security aspects such as confidentiality, integrity, and authorization. Traditional security measures, such as firewalls and Intrusion Detection Systems (IDS), frequently fail to prevent XSS attacks because these attacks utilize ports exposed to regular web traffic. Most intrusion detection systems (IDS) concentrate on the network and IP layers, while XSS operates at the application layer. Researchers have suggested various tools and techniques to assist developers in addressing these vulnerabilities. Nevertheless, specific existing methodologies are impractical due to their inability to address all types of assaults or their requirement to modify the original code, which can decrease performance. Identifying the optimal combination of mitigation techniques, platform, and browser is imperative. The paper is organized as follows: Section 2 addresses background information and XSS attacks. Section 3 compares the efficacy of two mitigation methods based on various factors. Section 4 details the implementation and discusses the results. Section 5 concludes the paper.

2. XSS ATTACK

Web application assaults are more severe than other software attacks because they are accessible to all users and have a broader attack surface. A web browser is all that hackers need to access and attack them. Web application assaults receive a high score when the DREAD model is implemented to assess system severity. The reason for this is that these attacks can be readily replicated and executed with only a web browser, which can potentially impact many users. Additionally, these vulnerabilities can be exploited to conduct more severe attacks on the local systems of users. Cross-site scripting (XSS) is a web application vulnerability in which an attack occurs when a victim's browser executes a malicious script from the perpetrator. In 2023, XSS comprised 9% of all newly reported vulnerabilities

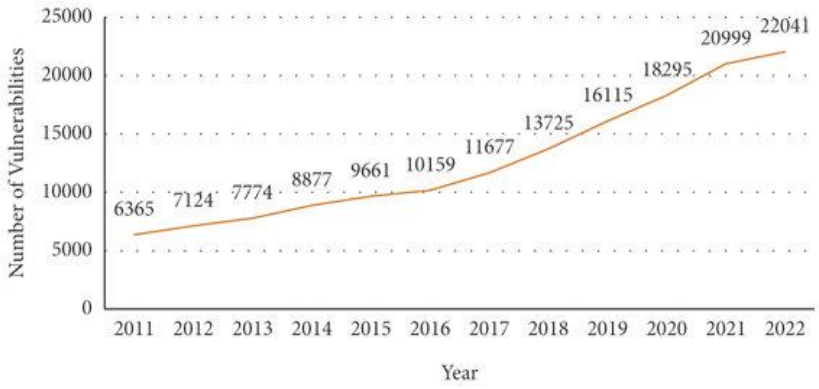

Fig.1 XSS Vulnerabilities over year

2.1. XSS Attack Types

Stored XSS, Reflected XSS, and DOM-based XSS are the primary varieties of Cross-Site Scripting (XSS) attacks.

A. Stored XSS

Malicious code permanently stored on a target server, such as in a database, a message forum, or a remark field, is referred to as stored XSS or persistent XSS. The malicious script is executed in the user's browser when they access the infected page. Example: An assailant posts a malicious script in a blog comment. The script executes in the browsers of other users when they read the comment, thereby obtaining their cookies or other sensitive data.

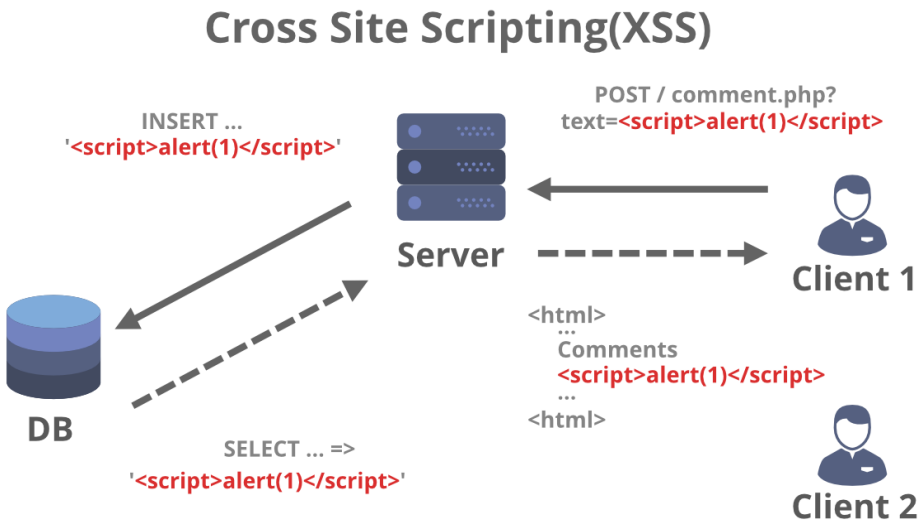

Fig. 2. Stored XSS diagram

B. Reflected XSS

Reflected XSS is a phenomenon in which malicious code is reflected off a web server, typically through a URL. The malicious code is included in the server's response when a user clicks on a link that contains the script. The user's browser then executes the code. Example: An adversary may embed a malicious script in an email and execute it when the user clicks the link. This could compromise the user's data.

C. DOM XSS

DOM-based XSS occurs when the vulnerability is on the client rather than the server-side code. The malicious script modifies the webpage's Document Object Model (DOM), causing the browser to execute the script. Example: An assailant uses a malicious script to generate a URL. The script modifies the web page's DOM and executes when the user opens the link, which may result in data theft or other malicious activities. Comprehending these XSS attacks is essential for creating effective security measures that safeguard web applications and consumers.

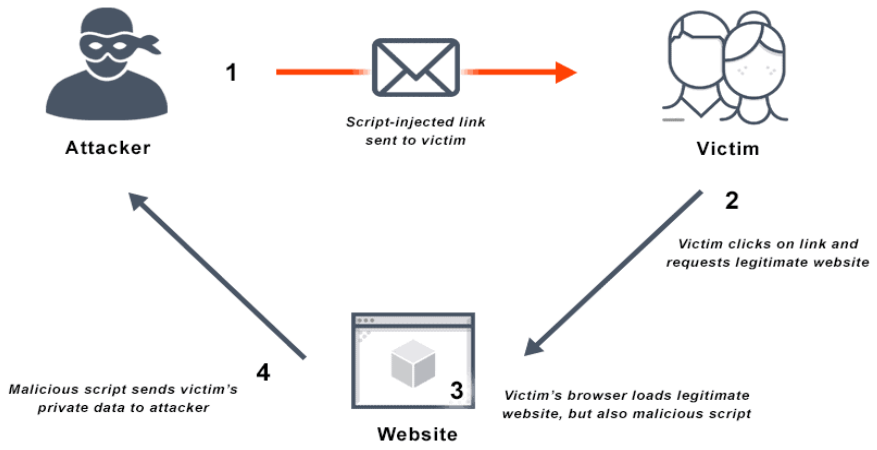

Fig. 3. DOM and Reflected XSS diagram

2.2 Reasons Behind XSS

XSS has been a significant concern in the internet community for many years in the context of prevalent vulnerabilities. Cross-Site Scripting (XSS) is a vulnerability in a web application that results from the application failing to adequately validate user input prior to transmitting it to the client's web browser. The prevalence of XSS vulnerabilities is influenced by a variety of factors. Initially, XSS can have an impact on web applications that display untrusted input, and the prerequisites for effectively exploiting XSS are minimal. Secondly, numerous web application programming languages have unsafe defaults for managing untrusted input, which frequently result in the direct printing of the input onto the output page. This facilitates the injection of malevolent scripts by attackers. Finally, the various methods, which are frequently browser-specific, by which JavaScript interpreters can be invoked make it difficult to ensure that untrusted input is properly validated. This complexity complicates the process of dependably preventing XSS vulnerabilities across various web applications.

2.3 Mitigations Techniques

Mitigating cross-site scripting (XSS) attacks is essential to guaranteeing the security of web applications. Numerous approaches are implemented to prevent or mitigate the consequences of XSS vulnerabilities. The purpose of these methods is to safeguard user input and prevent the introduction of malicious scripts into web pages. It is imperative that we examine a variety of widely used strategies for preventing XSS.

A. Input Validation

A prevalent method of verifying input from potentially detrimental sources is input validation. For instance, a server-side routine can eliminate any letters from the result to ensure that it only contains numbers, in the event

that a form only permits numerals and not letters. This prevents XSS attacks by converting specific characters into their HTML equivalents. The following characters are included:

(,), [,], >, ;, :, <, ', ", /, \

B. HTTP Cookies

The HTTP flag is an effective defense against Cross-Site Scripting (XSS) attacks. It is introduced when cookies are transmitted to the client via HTTP response headers. It is intended to reduce the likelihood of information disclosure. This flag indicates that a cookie is inaccessible through scripting. Consequently, it restricts malicious scripts' capacity to interact with cookies, thus reducing their susceptibility to XSS attacks.

C. Html Encoding

The most straightforward way to implement mitigation strategies is to encode (or "HTML quote") all non-alphanumeric characters in HTML that the user provides. This process prevents them from being read as HTML, lowering the risk of being used without permission.

D. Content Security Policy (CSP) CSP headers define rules for what resources can be loaded on a web page, limiting the execution of scripts from untrusted sources.

E. Checklist and Standards

Web designers and maintainers can lower the risk of cross-site scripting (XSS) by adhering to best practices and using checklists. This is a list of things that they can do to help stop XSS attacks.

- Ensure web pages validate user inputs for malicious code before returning them
- Convert non-alphanumeric characters to HTML entities before displaying user input in search engines and forums.
- Use testing tools in the design phase to eliminate XSS vulnerabilities. Develop standard scripts with private and public keys to verify genuine scripts.

E.1 Zero Trust Model

Zero trust in input data means assuming all user input is potentially harmful and treating it accordingly. It entails thoroughly validating and sanitizing all input, regardless of source, to reduce security risks such as XSS attacks.

E.2 WAF (Web Application Firewalls)

WAFs filter out malicious data from web traffic. They act as a barrier between the web app and the internet, preventing malicious traffic from reaching the application and protecting against XSS and other web threats.

E.3 Regular Security Audits

Regularly auditing web applications for vulnerabilities and implementing necessary patches or updates helps maintain security against evolving threats.

E.4 Security Awareness Training

Educating developers and users about the risks of XSS attacks and best practices for prevention enhances overall security posture.

3. Conclusion and future work

Web/browser attacks exploit AAA (authorization, authentication, and accounting) vulnerabilities in Cloud Systems, often uploading malicious programs to cause damage. SQL injection and Cross-Site Scripting are common and significant breaches in web application security. Cloud security systems need thorough analysis to address vulnerabilities in web servers, as attackers may inject malicious services. While these attacks are common, they're often overlooked, posing a serious threat as web-based software users increase. Mitigation techniques

include filtering dangerous characters and converting them to HTML equivalents to block SQL injection and XSS attacks. Future work involves in-depth vulnerability analysis to protect cloud services from such attacks.

Resources

1. Kaur, J., Garg, U., & Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. Artificial Intelligence Review, 56(11), 12725-12769.
2. Liu, Z., Fang, Y., Huang, C., & Xu, Y. (2023). MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model. Computers & Security, 124, 103015.
3. Kaur, J., Garg, U., & Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. Artificial Intelligence Review, 56(11), 12725-12769.
4. Rodríguez, G. E., Torres, J. G., Flores, P., & Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. *Computer Networks*, *166*, 106960.
5. Liu, Z., Fang, Y., Huang, C., & Han, J. (2022). GraphXSS: an efficient XSS payload detection approach based on graph convolutional network. Computers & Security, 114, 102597.
6. Lei, L., Chen, M., He, C., & Li, D. (2020, October). XSS detection technology based on LSTM-attention. In 2020 5th International Conference on Control, Robotics and Cybernetics (CRC) (pp. 175-180). IEEE.
7. Melicher, W., Fung, C., Bauer, L., & Jia, L. (2021, April). Towards a lightweight, hybrid approach for detecting DOM XSS vulnerabilities with machine learning. In *Proceedings of the Web Conference 2021* (pp. 2684-2695).
8. Kumar, S., Pathak, S., & Singh, J. (2022). An enhanced digital forensic investigation framework for XSS attack. Journal of Discrete Mathematical Sciences and Cryptography, 25(4), 1009-1018.
9. Kumar, S., Pathak, S. K., & Singh, J. (2022). A comprehensive study of XSS attack and the digital forensic models to gather the evidence. *ECS Transactions*, *107*(1), 7153.
10. Nair, S. S. (2024). Securing Against Advanced Cyber Threats: A Comprehensive Guide to Phishing, XSS, and SQL Injection Defense. *Journal of Computer Science and Technology Studies*, *6*(1), 76-93.
11. Shaymaa Adnan Abdulrahman, Bilal Alhayani,A comprehensive survey on the biometric systems based on physiological and behavioural characteristics,Materials Today: Proceedings,Volume 80, Part 3,2023,Pages 2642-2646,https://doi.org/10.1016/j.matpr.2021.07.005.
12. Alhayani, B.A., AlKawak, O.A., Mahajan, H.B. et al. Design of Quantum Communication Protocols in Quantum Cryptography. Wireless Pers Commun (2023). https://doi.org/10.1007/s11277-023-10587-x
13. B. T. Sabri and B. Alhayani, "Network Page Building Methodical Reviews Using Involuntary Manuscript Classification Procedures Founded on Deep Learning," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-8, doi: 10.1109/ICECCME55909.2022.9988457.
14. Omar A. AlKawak, Bilal A. Ozturk, Zinah S. Jabbar, Husam Jasim Mohammed,Quantum optics in visual sensors and adaptive optics by quantum vacillations of laser beams wave propagation apply in data mining,Optik,Volume 273,2023,
15. Ismaeel, N.Q., Mohammed, H.J., Chaloob, I.Z. et al. Application of Healthcare Management Technologies for COVID-19 Pandemic Using Internet of Things and Machine Learning Algorithms. Wireless Pers Commun (2023). https://doi.org/10.1007/s11277-023-10663-2.
16. Mohanty, Niharikaa | Pradhan, Manaswinia | Mane, Pranoti Prashantb | Mallick, Pradeep Kumarc; * | Ozturk, Bilal A.d | Shamaileh, Anas Atefe : Intelligent Decision Technologies, vol. Pre-press, no. Pre-press, pp. 1-26, 2024
Rane ME, Bhadade US. Multimodal score level fusion for recognition using face and palmprint. International Journal of Electrical Engineering & Education. 2020;0(0). doi:10.1177/0020720920929662