



Effective Security Analytical Enhancement for Cross-Site Scripting and Code Injection Vulnerabilities in Web Security

Shreyas Pagare^{1*}, Anish Kumar Choudhary²,
Sanjay Thakur³, Ashutosh Khemriya⁴ and Deepesh Shrivastava⁵

^{1,2,3,4,5}Chameli Devi Group of Institutions, Indore, (M.P),India

*shreyas.pagare@gmail.com

Abstract. SQL injection and Cross-Site Scripting (XSS) have emerged as significant security vulnerabilities in contemporary web applications, eclipsing buffer overflow weaknesses, as indicated by recent vulnerability and exploit reports. Both SQL injection and XSS exemplify a broader category of input validation deficiencies. This research aims to investigate these vulnerabilities comprehensively while proposing a user-centric architecture to guarantee secure data transmission. This paper particularly examines a model that offers a framework for both symmetric and asymmetric encryption techniques, demonstrating greater reliability than traditional encryption methods.

Keywords: SQL Code Injection, Cross-Site Scripting, Cybersecurity.

1 Introduction

Today's financial and everyday lives are increasingly reliant on website services like e-commerce, internet banking, and web-based emails. On the millions of servers in use around the world, there are several active applications. Internet usage has significantly increased, making it a target for attacks from hackers and attackers. Targeting less secure automated machines is simple.

The client SQL query can edit or create malicious SQL commands, run arbitrary code on the target machine, or change the content of a database if the client-provided information is not properly validated.

Declassification, Integrity, authentication process, and Authorization are all in grave danger as a result of the impact and results of SQL injection assaults. It can be challenging to tell the difference between the SQL injection and its effect in a real-world scenario. In a vast number of cases, the attacker conducts prohibited activity using strong client credentials or by leveraging built-in database application features, such as maliciously altering the current SQL Queries of the online application that access key parts of the affected databases.

© The Author(s) 2025

S. Bhalerao et al. (eds.), *Proceedings of the International Conference on Recent Advancement and Modernization in Sustainable Intelligent Technologies & Applications (RAMSITA-2025)*, Advances in Intelligent Systems Research 192,

https://doi.org/10.2991/978-94-6463-716-8_73

The assailant utilizes the identical communication channel for both the assault and the retrieval of results. In-band vulnerability is one of the most common forms of SQL injection attack. There are two categories: Error-based SQL.

Attacker will be able to do activities that cause the database to create error messages. These error code could include details that the attacker might use to find out more about the layout of the database.

Union based SQL:

This method combines several carefully selected statements that the database actually generates into a single HTTP response by using the UNION SQL operator. The relevant information in this answer could be useful to the attacker. There are some common attacks related to code injection as shown in figure 1.

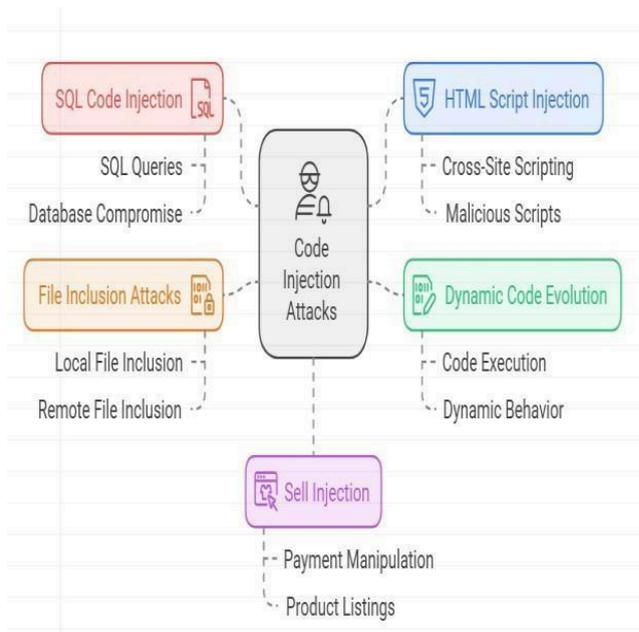


Fig. 1. Code Injection Attacks

There is a gap between the server and client; the client is unaware of how the server manages requests Internet sites and web-based applications increasingly rely on JavaScript, which is becoming more powerful. You should take great care to do everything you can to avoid it because an attacker who successfully attacks XSS has access to a lot of data. Cross-site script vulnerabilities have a root problem that, like other programming issues, depends too much on your data source. but if you have a 'Name' field, it is simple to assume that the only items you receive in this field resemble human names from the standpoint of a reasonable and kind individual. The

study also examines how responsive public safety agencies are to reducing recently identified dangers. A prompt response is essential today, particularly with the prevalent use of CMS platforms such as WordPress, Joomla, or Drupal. Data-driven web services and software applications that enable online business transactions for users. Contemporary businesses and individuals primarily rely on these web applications to interact directly with their varied clientele. Back-end databases are accessed via user input through web applications to retrieve the necessary information. This intrinsic tendency has consequently increased the vulnerability of web services and applications to assaults by hacker collectives. Due to the rising use of web applications for social networking, financial transactions, and health-related issues, internet security is now essential. Software vulnerabilities are becoming increasingly critical as a concern[1].

As per a questionnaire conducted by OWASP, the top ten security vulnerabilities as of June 2019 included source code injection, compromised user authentication and security mechanisms, excessive exposure of sensitive data, XML external entity attacks, compromised security systems, security misconfiguration, cross-site scripting (XSS), highly insecure deserialization, utilization of components with known vulnerabilities, and insufficient logging and monitoring. Nonetheless, XSS and SQL injection have been recognized as among the most perilous of these attack vectors [2].

2 Literature Survey

The usage of encrypting data methods, PHP escape fundamental operations, pattern recognition optimization techniques, and randomization of the instruction set have all been proven to be effective in mitigating SQL injection vulnerabilities and XSS assaults, according to published research. A method to stop XSS and SQL injection attacks was suggested by the authors of [3]. Application programs were divided into two groups: static and dynamic. Fixed web applications have queries that don't change with time or parameter changes, whereas dynamic web applications have queries that change as a result of time or data input.

A SHA-1 encoding method was utilized by the creators of [4] to prevent mass SQL injections. It works by deleting query identifiers from inputs that have been stored and encrypted with the SHA-1 algorithm. Then, before any further processing and storage, any new input will be hashed and checked with input that has already been saved with its hash[5]. Security in content management system researcher [6] analyse that WordPress is the most widely used content management system, hackers are continuously drawn to it. Update the content management system frequently using the database of analytical scan plugins and previously discovered vulnerabilities. Members of WordPress gain from enhanced security, and a committed staff will conduct a thorough code review to check for problems. All scripts must be launched and interpreted by a web browser. A proposed analysis of process[9] description and

configuration enhancements using secure keys and salts was offered in this paper. It will increase the security of content management.

The SQL query will be run if the encrypted input under comparison is identical; else, it will be denied. This method prevents incorrect inputs from being managed by the SQL query directly and attempts to get input values from storage will only return hashed, encrypted data. Similar to this, writers in [5] presented the Boyer and Moore data string matching technique for SQL code injection attack identification and avoidance. Big data and opinion strength rule based fuzzy theory and evolution introduced by researchers [4, 6] for potential SQL injection attack characteristics. Ghfarian suggested a hybrid technique to analysing and preventing code SQL-injection that makes use of the advantages of static and dynamic approaches. The static technique looks for error code written in the SQL query database tier and, the dynamic approach looks for potential runtime vulnerabilities common gateway interface layer. First, a matching algorithm between strings from input SQL queries and stored SQL queries was created. The provided result matched the anticipated valid query. The query will be rejected if vulnerability is found. A method to dynamically assess incoming queries was presented for the dynamic approach [7]. Researcher [8] gives analysis and the suggested methodology; content management systems might be improved to provide more effective and efficient web content security control. With research concentrated on the areas of CMS where it is required, a comprehensive construction of the proposed system is suggested, which will manage all processes at the tactical, strategic, and operational levels. It is demonstrated that the analysis of the proposed work is helpful for identifying problems in content management systems as well as other content-driven fields. The suggested setup would examine the shortcomings, reliability, and efficiency of a secure WordPress CMS and offer suggestions for enhancements [9]. Prevent session hijacking and code injection. Upon user registration, a distinct hash value is generated for their input, which is subsequently compared to the login credentials provided later. Overcontrol was additionally mitigated by creating hash values for authentic system and web browser data, including the web browser name, potential host IP, and version. This hash value will be compared to the subsequent values. The technique can mitigate attacks involving tautology, error code, piggybacking, and UNION. A signature-based approach for executing deep packet inspection on HTTP packet payloads was presented in [10].

The developers proposed a strategy to mitigate SQL injection and XSS attacks. A novel SOA security framework has been developed to protect web services from WSDL attacks, as introduced by researcher. Web applications are categorized into static, whose queries remain unchanged over time or parameters, and dynamic, whose queries fluctuate based on time or input data. Static and dynamic mapping models were employed to identify and mitigate risks from both groups. Prabakar et al. employed analogous tactics utilizing the Aho-Cora sick pattern matching technique. The researcher provides configuration enhancements for securing webpages designed

by content management systems and ensuring content security through the reconfiguration of secure keys and the slots of default CMS indexing. An automated framework for securing Content Management Systems was introduced by the researcher.[11, 12].

3 Attack Patterns Of Cross-Site Scripting And SQL Injunction

One of the associated risks that is frequently detected is cross-site scripting (XSS). It takes use of known and established server vulnerabilities and enables intruders to insert client-side content into web pages seen by many clients, electronic application, and associated module framework vulnerabilities. The attackers inject harmful material into any of the exchanged locations, gaining secure access to sensitive page content, session cookies, and other confidential information stored by the software. In this way, XSS might be assigned as a special case of infusion [13].

Before malicious Script code (Fig. 2) can be executed inside the victim's web browser, the alleged attacker must first figure out how to introduce a payload into a web page. Attackers frequently employ social engineering tactics to get visitors to visit a hacked website that has a JavaScript Payload placed in it. User input or malicious code included directly in the pages of the hacked website is required for an XSS attack to succeed [14, 15].

```

echo "<html><body>";
echo"<h1>comments recieved </h1>";
echo database.recentcconnect;
echo"</html</body>";

echo"<h1>comments recieved </h1>";
echo"<script>insertjavascript();</script>";
echo "<html><body>";

```

Fig. 2. Pseudo-code on the server side is used to view the comments

Because an attacker may transfer malicious code or a java script payload, cross-site scripting harmed websites. The ability to conduct a database query is frequently compromised by a significant bug known as just a SQL injection. With the use of this web-based code attack, firewalls may be bypassed while connecting database backends. Weak encoding and poor data validation have the disadvantage of allowing an attacker to do unauthorized SQL operations.

Before executing fraudulent SQL queries against another database server, an intruder must first locate an input within the web application that contains a SQL query as shown in Fig. 3. A hacker group can then execute the SQL query against the database server after adding a malware to it.

```
txtusername=getrequeststring("username");
txtusername=getrequeststring("password");
sql="SELECT id FROM users WHERE username='"+txtusername+"'"
AND password='"+txtpassword+'";
```

Fig. 3. Authenticating users from server side

A SQL Injection attack's payload (Fig. 4) might be created by setting the password field to OR 1=1 for password. The database server would respond to the following SQL query, which would be the outcome.

```
SELECT id FROM users WHERE username ='username'
AND password = 'password' OR 1=1'
```

Fig. 4. A SQL Injection attack payload

4 Conclusion

This research is believed to analyse especially for XSS and SQL injection. Web applications typically have the weaknesses. It looks that SQL injection attacks are abusing it to gain from the application's participation. It is advised to all researchers and developers to use strong coding conventions and follow security rules while building the apps since cross-site scripting CSS and SQL injections SQLI are the incredibly risk-filled things that happen frequently in the current online attacks.

References

1. A. K. Phulre, S. Pagare, and A. Chakrawati, "Automated Framework for Web Content Security Through Content Management System," in *2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22)*, IEEE, Apr. 2022, pp. 1–4. doi: 10.1109/ICETET-SIP-2254415.2022.9791492.
2. S. Pagare, S. K. Shukla, P. Pandey, V. K. Gupta, H. Jindal, and A. Jain, "Image Forgery Detection Model Analysis using Statistical Splicing Method for Architecture Learning and Feature Extraction," in *2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0*, IEEE, Jun. 2024, pp. 1–6. doi: 10.1109/OTCON60325.2024.10687764.

3. W. Zhang *et al.*, “Deep Neural Network-Based SQL Injection Detection Method,” *Security and Communication Networks*, vol. 2022, pp. 1–9, Mar. 2022, doi: 10.1155/2022/4836289.
4. P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, “Detection of SQL injection based on artificial neural network,” *Knowl Based Syst*, vol. 190, p. 105528, Feb. 2020, doi: 10.1016/j.knosys.2020.105528.
5. P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, “Detection of SQL injection based on artificial neural network,” *Knowl Based Syst*, vol. 190, p. 105528, Feb. 2020, doi: 10.1016/j.knosys.2020.105528.
6. F. K. Alarfaj and N. A. Khan, “Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks,” *Applied Sciences*, vol. 13, no. 7, p. 4365, Mar. 2023, doi: 10.3390/app13074365.
7. M. Alghawazi, D. Alghazzawi, and S. Alarifi, “Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model,” *Mathematics*, vol. 11, no. 15, p. 3286, Jul. 2023, doi: 10.3390/math11153286.
8. Zhang, Wei, Yueqin Li, Xiaofeng Li, Minggang Shao, Yajie Mi, Hongli Zhang, and Guoqing Zhi. "Deep Neural Network-Based SQL Injection Detection Method." *Security and Communication Networks* 2022, no. 1 (2022): 4836289.
9. M. Alghawazi, D. Alghazzawi, and S. Alarifi, “Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review,” *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, pp. 764–777, Sep. 2022, doi: 10.3390/jcp2040039.
10. R. K. Pathak, Mohit, and V. Yadav, “Handling SQL Injection Attack Using Progressive Neural Network,” 2020, pp. 231–241. doi: 10.1007/978-981-15-9671-1_20.
11. P. A. Sonewar and N. A. Mhetre, “A novel approach for detection of SQL injection and cross site scripting attacks,” in *2015 International Conference on Pervasive Computing (ICPC)*, IEEE, Jan. 2015, pp. 1–4. doi: 10.1109/PERVASIVE.2015.7087131.
12. P. Tanakas, A. Ilias, and N. Polemi, “A Novel System for Detecting and Preventing SQL Injection and Cross-Site-Script,” in *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, IEEE, Dec. 2021, pp. 1–6. doi: 10.1109/ICECET52533.2021.9698688.
13. Q. Temeiza, M. Temeiza, and J. Itmazi, “A novel method for preventing SQL injection using SHA-1 algorithm and syntax-awareness,” in *2017 Joint International Conference on Information and Communication Technologies for Education and Training and International Conference on Computing in Arabic (ICCA-TICET)*, IEEE, Aug. 2017, pp. 1–4. doi: 10.1109/ICCA-TICET.2017.8095285.
14. P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, “SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel,” *Journal of Information Security and Applications*, vol. 40, pp. 199–216, Jun. 2018, doi: 10.1016/j.jisa.2018.04.001.
15. B. Soewito, F. E. Gunawan, Hirzi, and Frumentius, “Prevention Structured Query Language Injection Using Regular Expression and Escape String,” *Procedia Comput Sci*, vol. 135, pp. 678–687, 2018, doi: 10.1016/j.procs.2018.08.218.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

