ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

Comprehensive Review of Advanced Techniques for Mitigating SQL Injection Vulnerabilities in Modern Applications

Amit Hariyani[1]*; Dr. Prashant Dolia[2]

[1]Smt. Chandaben Mohanbhai Patel Institute of Computer Applications, Charotar University of Science and Technology, Off. Nadiad-Petlad Highway, Changa, Anand, 388421, Gujarat, India
[2]Department of Computer Science, M. K. Bhavnagar University, Sardar Patel Campus, Bhavnagar, 364002, Gujarat, India

Corresponding Author: Amit Hariyani[1]*

Publication Date: 2025/04/16

Abstract: SQL injection (SQLi) remains a major security threat to database-driven applications, making it essential to protect the confidentiality, integrity, and availability of data. In this research, we summarize effective strategies to prevent SQL injection attacks (SQLIAs), such as parameterized queries, stored procedures, Object Relational Mappers (ORM), input validation, input escaping, and Web Application Firewalls (WAF). We assess each technique based on how well it works, how easy it is to use, and its impact on performance, with real-world examples to show their use. Our literature review covers research from the past five years, highlighting the changing nature of SQLi threats and the improvements in prevention methods. This study offers a detailed look at effective SQLi prevention techniques and their implementation, and a comparison of their effectiveness. By understanding and using these approaches, organizations can significantly reduce the risk of SQLIAs and protect their important data.

Keywords: *SQL Injection, Database Security, Parameterized Queries, Stored Procedures, ORM, Input Validation.*

How to Cite: Amit Hariyani; Dr. Prashant Dolia. (2025). Comprehensive Review of Advanced Techniques for Mitigating SQL Injection Vulnerabilities in Modern Applications. *International Journal of Innovative Science and Research Technology*, 10(3), 3063-3070. https://doi.org/10.38124/ijisrt/25mar1982.

I. INTRODUCTION

SQL Injection (SQLi) poses a significant threat to applications that rely on databases [1] and is widely regarded as one of the most severe security issues affecting websites. This type of attack occurs when user inputs are not handled correctly in SQL queries [2], allowing attackers to run any SQL code they want on a database. Successful SQLi attacks can lead to unauthorized access to sensitive data, data breaches, data manipulation, and complete control of the database server [3]. Despite improvements in web security, SQLi remains a significant risk; therefore, it is important to develop effective prevention methods [4]. SQLi attacks are popular owing to their simplicity and effectiveness. Attackers can create harmful inputs that change SQL queries, bypass security checks, and gain unauthorized access to data [5]. Many high-profile data breaches have been caused by SQLi attacks, leading to financial losses, damage to reputations, and legal problems for affected organizations [6]. Databases store important information, such as personal data and financial records; therefore, their security is very important.

To counter these threats, several techniques have been developed to reduce the risk of SQLi [7]. One of the best methods is to use parameterized queries [8], also known as prepared statements. These separate the SQL code from the user inputs, ensuring that the inputs are treated as data and not as executable code, which stops the attack. Stored procedures that encapsulate SQL logic within a database also help to reduce direct exposure to SQL queries and enable input validation and security policies within the database layer [9].

The Object Relational Mapper (ORM) [10] simplifies database interactions by automatically handling parameterizations, thereby reducing the risk of SQLi. ORMs offer a high-level programming interface that makes database operations easier for developers, while ensuring security [11]. Input validation and sanitization improve security by verifying that user inputs meet the expected formats and remove harmful characters [12]. Escaping input characters is another method for preventing SQLi and treating special characters in user inputs safely [13]. Web Application Firewalls (WAFs) [14] provide an extra layer

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

of defense by inspecting incoming requests and filtering malicious inputs before they reach an application. WAFs are particularly effective in protecting against various web application attacks, including SQLi [15].

➢ *The Objectives of this Study are as Follows:*

- To evaluate the effectiveness of SQLi prevention techniques
- To analyze recent advances in SQLi detection and prevention
- To compare the practical implementation of SQLi prevention methods
- To identify and discuss the challenges and limitations of current SQLi prevention methods
- To propose a comprehensive strategy for SQLi mitigation

This study provides an in-depth examination of various strategies for combating SQLIAs. It reviews the effectiveness, ease of use, and performance impact of techniques, such as parameterized queries, stored procedures, ORMs, input validation, input escaping, and WAFs. It also addresses the challenges of balancing security with 2 performance and emphasizes the need for ongoing development and education in secure coding practices.

The rest of this paper is structured as follows: Section 2 covers related works on SQLi vulnerabilities and prevention techniques. Section 3 provides a brief overview of the proposed methodology for identifying and mitigating SQLi vulnerabilities. Section 4 discusses the experimental setup and analysis of the results. Section 5 concludes the study and outlines future research directions in this field.

II. RELATED WORK

The study covers research from the last five years, focusing on the changing nature of SQLi threats and advancements in countermeasures. Researchers have emphasized the importance of using a comprehensive approach for SQLi prevention by combining multiple techniques to ensure strong protection.

The ongoing challenges of SQLi threats highlight the importance of cybersecurity. As a regular threat to the OWASP top ten list [16] of critical web application security risks, SQLi requires constant monitoring and improvements in prevention methods. A comprehensive approach that combines static and dynamic analyses is necessary to effectively identify and address vulnerabilities, as noted by Alsmadi and Farooq [17, 18]. They also pointed out the limitations of traditional detection methods and recommended using hybrid machine-learning models, which have been shown to improve accuracy and detection rates.

In recent years, using Machine Learning (ML) and Artificial Intelligence (AI) to detect SQLi has become more popular [19, 36, 37]. Brindavathi and Demilie [20, 21] found that ML models can identify SQLi patterns and stop attacks in real-time. Their studies showed that ML-based methods can outperform traditional signature-based methods by learning and adapting to new attack vectors, thereby offering stronger defence mechanisms. Similarly, Kakisim and Tang [22, 23] investigated the use of deep learning for SQLi detection, and highlighted that neural networks can identify complex and previously unknown injection patterns.

Web Application Firewalls (WAFs) are crucial for preventing SQLi attacks by filtering and monitoring HTTP requests in web apps. Mukhtar and Azer [24] found that WAFs effectively stop many SQLi attempts; however, they should be used alongside other security measures such as input validation and parameterized queries for maximum protection. WAFs add an extra layer of security, but their effectiveness relies on regular updates and proper configurations to address new threats.

Appropriate input validation and sanitization are essential for preventing SQLi attacks. Fadlalla and Elshoush [25] found that secure coding practices, including strict input validation, are vital in preventing harmful inputs from reaching a database. Ali [26] reviewed various input validation techniques and discovered that while these methods are effective as a first line of defence, they need to be used with other security measures for full protection.

Parameterized queries and stored procedures are well-recognized for their effectiveness in preventing SQLi. Sidik [27] noted that parameterized queries are essential for separating SQL code from user inputs, thereby eliminating the attack vector. Lu [28] highlighted that stored procedures encapsulate SQL logic within a database, reduce the exposure to direct SQL queries, and enable better input validation and security policies at the database level.

ORMs such as SQLAlchemy [29, 30] simplify database operations by managing interactions and parameterization, which help to reduce the risk of SQLi and makes database management easier for developers. Dash [31] also noted that, although ORMs may add some performance overhead, their benefits in terms of security and developer efficiency make them a key tool in modern application development.

Fredrick et al. [32] found that automated tools for preventing SQLi are essential for spotting vulnerabilities early in the development process. By integrating these tools into continuous integration and deployment (CI/CD) pipelines, the risk of SQLi can be significantly reduced by identifying and fixing vulnerabilities before they can be exploited. Angshuman et al. [33] emphasized the need to educate developers and provide ongoing training in secure coding practices to further reduce SQLi risks.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

Table 1: Summary of Literature Review Findings

Authors	Focus	Key Findings
Alsmadi et al. [17]	SQLi detection and prevention techniques	Emphasized hybrid approaches combining static and dynamic analysis
Umar Farooq et al. [18, 36]	Machine learning for SQLi detection	Improved accuracy with hybrid machine learning Models
Mukhtar and Azer [24]	Effectiveness of WAFs	WAFs need to be combined with other techniques for optimal security
Kakisim et al. [22]	Deep learning for SQLi detection	Neural networks show promise in detecting complex patterns
Fadlalla and Elshoush [25]	Secure coding practices	Importance of developer education and secure coding practices
Fredrick et al. [32]	Automated tools for SQLi prevention	Automated detection mechanisms can identify vulnerabilities early
Brindavathi and Demilie [20, 21]	Machine learning-based SQLi detection	Machine learning improves detection rates compared to signature-based methods
Tang et al. [23]	Real-time SQLi mitigation with neural networks	Effective real-time SQLi mitigation using neural networks

III. METHODOLOGIES

This study examines six primary methods for preventing SQLi: parameterized queries, stored procedures, ORMs, input validation, input escaping, and WAFs. We assess each method based on how well it works, how easy it is to use, and how it affects performance.

A. Parameterized Queries

Parameterized queries use placeholders for user inputs, which ensures that these inputs are treated as data and not as executable code. This method is illustrated with examples in Python using SQLite.

```
Python
# Example of parameterized query in Python using SQLite
import sqlite3
def get_user_data(user_id):
    conn = sqlite3.connect('example.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users WHERE id =
?", (user_id,))
    result = cursor.fetchall()
    conn.close()
    return result
```

B. Stored Procedures

Stored procedures keep SQL code within the database, which limits direct access to SQL queries and allows input validation. This study includes examples of stored procedures in SQL Server.

```
Sql

-- Example of stored procedure in SQL Server

CREATE PROCEDURE GetUserData
    @UserId INT
AS
BEGIN
    SELECT * FROM Users WHERE Id = @UserId
END
```

C. Object Relational Mappers

ORMs simplify database interactions, automatically manage parameterizations, and reduce the risks of SQLi. This paper provides an example of ORM using SQLAlchemy in Python.

```
Python
# Example of ORM usage in Python with SQLAlchemy

from sqlalchemy import create_engine, Column, Integer,
String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

Base = declarative_base()
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String)

engine = create_engine('sqlite:///example.db')
Session = sessionmaker(bind=engine)
session = Session()
def get_user_data(user_id):
    return session.query(User).filter(User.id == user_id).all()
```

D. Input Validation and Sanitization

Input validation checks that user inputs match the expected formats, whereas sanitization removes harmful characters. This study includes examples of input validation using regular expressions in Python.

```
Python
import re
def validate_input(user_input):
    if re.match("^[a-zA-Z0-9_]+$", user_input):
        return True
    return False
```

E. Input Escaping

Escaping special characters from the user input prevents SQLi by ensuring that these characters are treated

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

as plain text. This method is demonstrated using Python's MySQLdb library.

```
Python
import MySQLdb
def escape_input(user_input):
    conn = MySQLdb.connect('localhost', 'user', 'passwd',
'db')
    cursor = conn.cursor()
    escaped_input = MySQLdb.escape_string(user_input)
    cursor.execute(f"SELECT * FROM users WHERE name
= '{escaped_input}'")
    result = cursor.fetchall()
    conn.close()
    return result
```

F. Web Application Firewalls

WAFs add an extra layer of security by examining incoming requests and blocking harmful input. This study covers popular WAFs, such as ModSecurity and AWS-WAF.

IV. RESULTS AND ANALYSIS

This study found that no single method could provide full protection against SQLIAs. Because SQLi threats constantly evolve, a comprehensive strategy that combines multiple techniques is required to cover all aspects of application security. This approach not only addresses SQLi vulnerabilities at different levels but also strengthens the overall application security. For example, while parameterized queries and ORMs are highly effective at the code level, they might not protect against all types of attacks that target other parts of the application. In such cases, input validation and escape add an extra defence layer by ensuring that only safe inputs reach the database. Additionally, WAFs can detect and block SQLi attempts to bypass other defences, thereby providing an essential safety net.

A comparative analysis of SQLi prevention techniques (Table 2) shows that parameterized queries and ORMs are the most effective at preventing SQLi, with minimal impact on performance. ORMs simplify database interactions by allowing developers to work with objects rather than with raw SQL queries. Parameterized queries use placeholders for user input, keeping SQL code separate from data to prevent changes to the query structure. Stored procedures contain SQL code within the database and perform predefined tasks, offering strong protection, but being more complex to implement and maintain.

Input validation ensures that only acceptable data are processed by setting rules for type, length, format, and range. While input validation and escape are useful as initial defenses, they should be used with other methods for complete protection. WAFs protect web applications from various attacks, including SQLi attacks, by monitoring and filtering HTTP requests. They provide valuable broad security, but may add some latency.

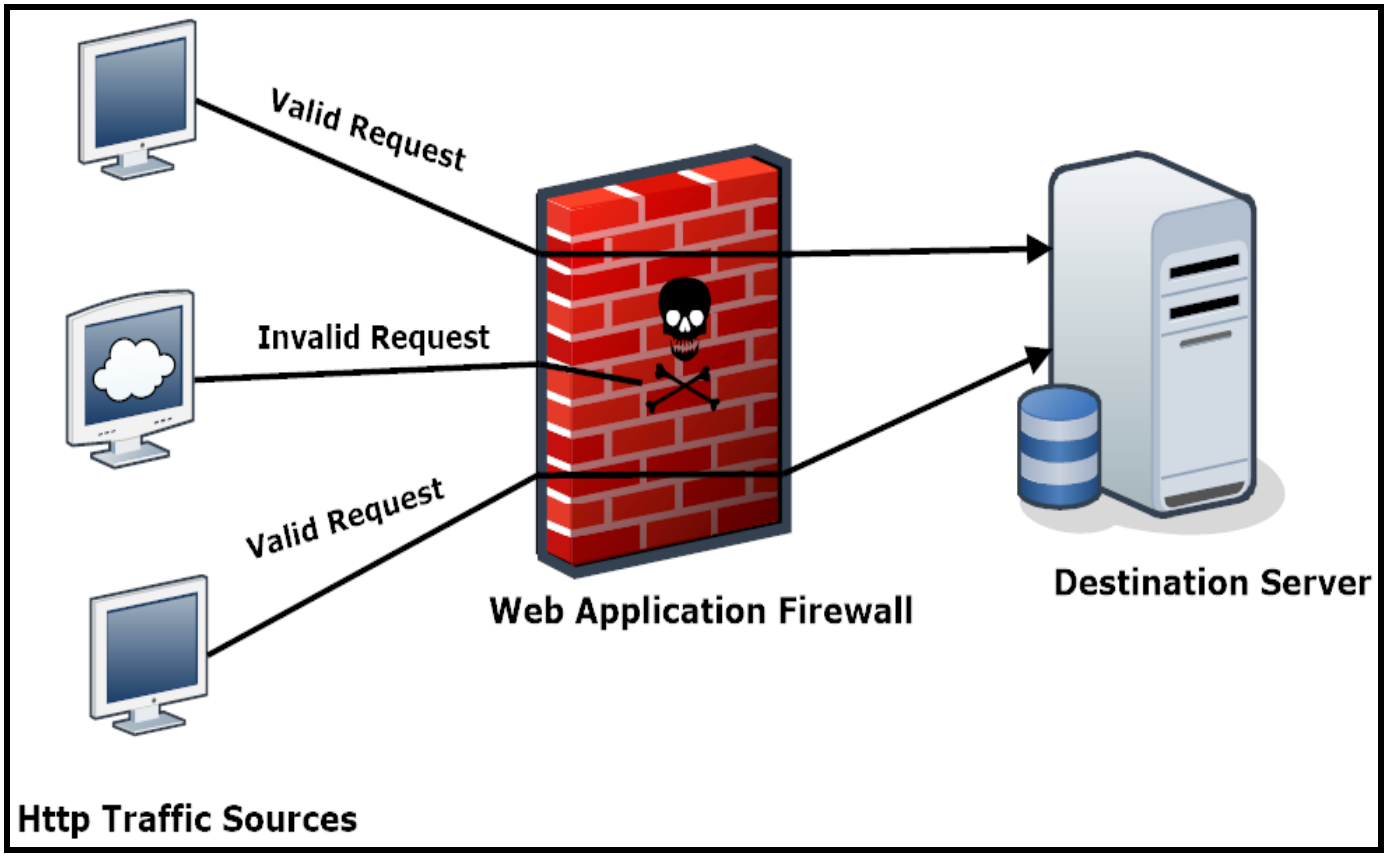

Fig 1: Web Application Firewall Architecture

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

Table 2: Comparative Analysis of SQL Injection Prevention Techniques

Technique	Effectiveness	Ease of Implementation	Performance Impact	Example Use Cases
Parameterized Queries	High	Moderate	Low	Web applications
Stored Procedures	High	Moderate	Moderate	Enterprise systems
ORMs	High	High	Moderate	Large-scale applications
Input Validation	Moderate	High	Low	User input forms
Input Escaping	Moderate	Moderate	Low	Legacy systems
Web Application Firewalls	High	High	Variable	Web services

Fig. 2 compares different techniques based on their effectiveness, ease of implementation, performance impact, flexibility, and maintenance needs. It offers a clear overview to help choose the best technique or a combination of techniques for a specific situation.

The effectiveness of SQLi prevention methods depends on the specific application scenarios. Parameterized queries consistently provide strong protection in all cases, making them a key part of any defence strategy. Input validation and stored procedures add extra security, particularly when used alongside parameterized queries. WAFs provide immediate, though not fully comprehensive protection. ORM frameworks build security during the development process, reducing the risk of human error. Using a combination of these methods customized to the application design and vulnerabilities is the best way to reduce SQLi risks.

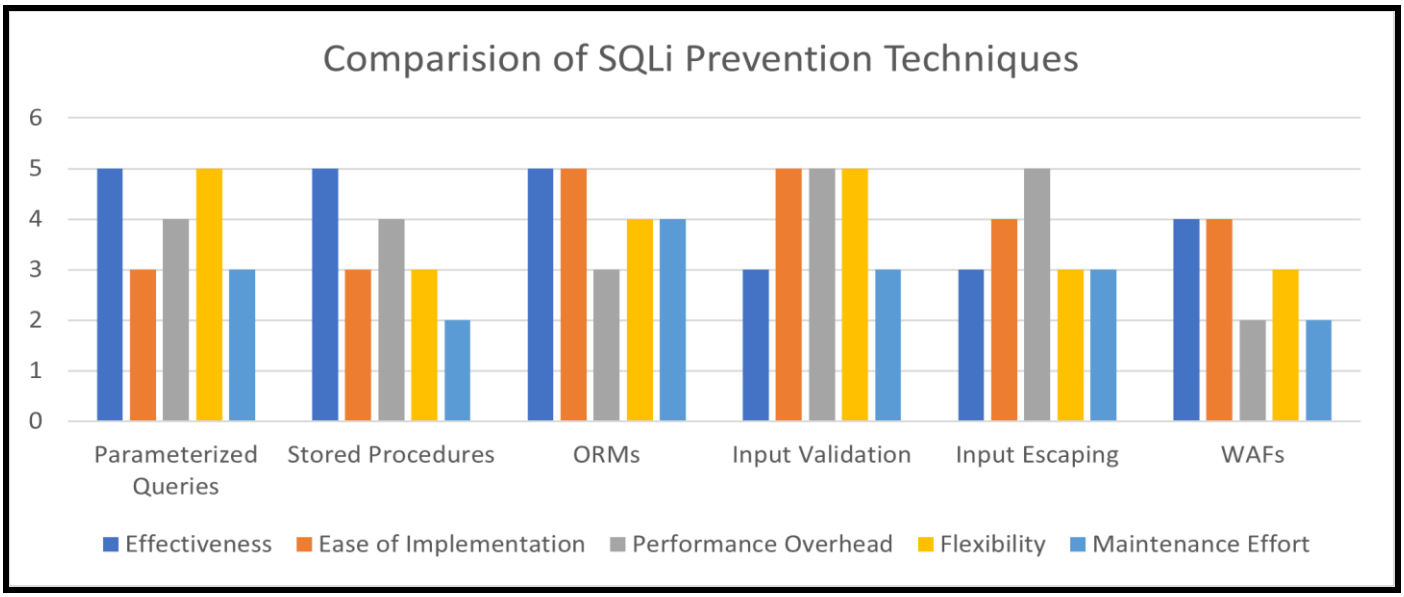

Fig 2: Comparison of SQLi Prevention Techniques based on Different Criteria

Table 3: Effectiveness of SQL Injection Prevention Techniques in Various Scenarios

Technique	Scenario	Effectiveness	Key Considerations
Parameterized Queries	Web applications	High	Simple to implement; low performance impact
Stored Procedures	Enterprise systems	High	Requires complex implementation; high performance for large systems
ORMs	Large-scale applications	High	Abstracts database operations; moderate performance impact
Input Validation	User input forms	Moderate	Easy to implement; often used as a supplementary measure
Input Escaping	Legacy systems	Moderate	Requires careful handling; low performance impact
Web Application Firewalls	Web services	High	Provides broad protection; potential latency introduced

Developers and system administrators must consider how SQLi prevention techniques affect system performance. Although security is crucial, it is important to choose methods that do not significantly slow down the system. Input validation has a minimal performance impact and provides cost-effective initial defense. Parameterized queries have a low to moderate impact, offering strong security with a manageable effect on performance. Stored procedures also have a low-to-moderate impact, combining good security with potential performance benefits from precompiled execution plans. WAFs have a moderate impact, providing broad protection, but increasing latency and resource use. ORM frameworks also have a moderate impact, simplifying development and boosting security, but require careful optimization to prevent performance issues.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

V. DISCUSSION

A key challenge in using SQLi prevention methods is finding an appropriate balance between security, performance, and usability. Parameterized queries and ORMs are very effective but can complicate development, requiring extensive developer training, which increases development time and costs. Input validation and escaping are simpler to implement, but may not fully protect against SQLi on their own; they work best when combined with more robust methods. WAFs are powerful but can add latency and require regular updates to stay current with new threats.

Table 4: Comparative Performance Impact of Prevention Techniques

Technique	Performance Impact	Latency Introduced	Resource Consumption	Typical Use Case
Parameterized Queries	Low	Minimal	Low	General web applications
Stored Procedures	Moderate	Moderate	Moderate	Enterprise applications
ORMs	Moderate	Low to Moderate	Moderate	Complex, Largescale systems
Input Validation	Low	Minimal	Low	Form inputs and simple queries
Input Escaping	Low	Minimal	Low	Older or less frequently updated systems
Web Application Firewalls	Variable	High	High	High-traffic web services

ML and AI are promising approaches to SQLi prevention.Machine learning models can process vast amounts of data to detect patterns and anomalies that may signify SQLIAs. These models can continuously learn and adapt to new attack methods, offering flexible defences that evolve with emerging threats. Recent studies, such as those by Hasan et al. [34] and Alkhathami et al. [35], have shown that ML methods can significantly improve the detection of SQLi attacks compared to traditional signaturebased approaches. However, these advanced techniques require specialized knowledge and resources, which may not be available to all organizations. Secure coding practices are crucial for effective SQLi prevention. Fadlalla et al. [25] highlighted that educating developers about SQLi risks and training them in secure coding practices are essential. This includes properly using parameterized queries, stored procedures, and ORMs as well as implementing effective input validation and escape.

Regularly investing in training and awareness programs is essential for organizations to keep their development teams up-to-date with the latest security practices and threats. Additionally, including security checks throughout the development process, such as code reviews and automated vulnerability scans, helps to identify potential SQLi vulnerabilities early. Despite advancements in SQLi prevention techniques, several challenges persist. The ever-evolving nature of SQLi attacks requires constant updates to prevention strategies. This means that ongoing research and development is needed to identify new vulnerabilities and create effective countermeasures. The practical examples and case studies in this study show how different SQLi prevention techniques are applied in real-world situations. These examples emphasize the need to tailor prevention strategies according to the specific needs and contexts of various applications. For instance, enterprise systems might benefit more from stored procedures that can handle complex business logic and enforce database-level security policies. On the other hand, web applications with high user interaction may use parameterized queries and ORMs to ensure secure and efficient data handling. Understanding the strengths and weaknesses of each technique helps organizations to implement a well-rounded and effective SQLi prevention strategy. Ongoing monitoring and adjustments are essential for effective SQLi prevention. This means regularly updating WAFs and other security tools to detect new attack patterns, performing regular security audits, and staying updated on the latest SQLi research and prevention methods. By adopting a proactive and adaptable approach to SQLi prevention, organizations can significantly improve their ability to protect their databases from malicious attacks and maintain the integrity and security of their data.

VI. LIMITATIONS

This study identified several limitations of the SQLi prevention methods. These include inconsistent implementation quality, constantly changing threat landscapes, and the challenges of combining multiple defences. The effectiveness of these methods can vary significantly depending on the development environment, and attackers continuously develop new techniques to bypass existing defences, thereby requiring ongoing updates to prevention strategies. Additionally, strong prevention techniques can create trade-offs between security, performance, and usability and may require specialized knowledge and resources that not all organizations can afford. Combining multiple techniques can also lead to compatibility issues, and many methods rely on developers to consistently use secure coding practices, which can be variable. Defensive tools such as WAFs mainly protect against network or application layer threats and may not fully cover vulnerabilities at the database or application logic levels. Real-world applications often include legacy codes and third-party components, which complicate the implementation of advanced security measures. To overcome these limitations, a comprehensive approach is needed that combines ongoing research, advanced technologies, and practical strategies, such as developing integrated tools, improving developer education, and exploring AI and ML for adaptive defences against evolving SQLi threats.

VII. CONCLUSION AND FUTURE WORK

In conclusion, SQLi continues to pose a significant threat to database security, necessitating robust preventive measures. This study has examined various effective

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

techniques for mitigating SQLIAs, including parameterized queries, stored procedures, ORMs, input validation, input escaping, and WAFs. The analysis demonstrates that employing a combination of these strategies offers the most comprehensive protection against SQLi vulnerabilities.

Future research should focus on developing integrated and effective prevention strategies for SQLi attacks. This includes exploring how machine-learning models and anomaly detection can work with traditional defenses. Continued emphasis on educating developers and applying secure coding practices is essential for addressing the root causes of vulnerabilities. Additionally, expanding the use of automated tools that can detect and fix SQLi vulnerabilities during development could greatly reduce human error and enhance the overall security.

ACKNOWLEDGEMENT

I would like to express my gratitude to the Department of Computer Science, M.K. Bhavnagar University for its support. I also extend my thanks to Dr. Prashant Dolia, my research supervisor, for his invaluable constructive suggestions and ideas, which greatly enhanced the quality of this paper.

DECLARATIONS

- **Funding Statement:** This study received no external funding.
- **Data Availability:** The datasets and code produced in this study are available from the corresponding author upon request.
- **Author Contributions:** Mr. Amit Hariyani contributed to the design, implementation, writing, and analysis of the results for the manuscript. Dr. Prashant Dolia conceived the project and provided supervision.
- **Ethical Approval:** This article does not contain any studies with human participants or animals performed by any of the authors.
- **Conflict of Interest:** The authors declare no conflict of interest.

REFERENCES

[1]. L. Ma, D. Zhao, Y. Gao and C. Zhao, ”Research on SQL Injection Attack and Prevention Technology Based on Web,” 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA), Xi’an, China, (2019), pp. 176-179, doi: 10.1109/ICCNEA.2019.00042.

[2]. Omer Aslan, Semih Serkant Aktu ¨ ˘g and Merve Ozkan-Okay, ”A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions”, (2023) Electronics 12(6):1-42, DOI: 10.3390/electronics12061333.

[3]. A. Al Anhar and Y. Suryanto, ”Evaluation of Web Application Vulnerability Scanner for Modern Web Application,” 2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST), Yogyakarta, Indonesia (2021), pp. 200-204, doi: 10.1109/ICAICST53116.2021.9497831.

[4]. Xue Ping-Chen, “SQL injection attack and guard technical research”, Procedia Engineering, Volume 15, (2011), Pages 4131-4135, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2011.08.775.

[5]. Harshavardhan Gaddam and M. Maheshwari, ”SQL Injection-Biggest Vulnerability of the Era”, EasyChair Preprint no. 4175, September 13, (2020)

[6]. Yuchong Li, Qinghui Liu, ”A comprehensive review study of cyberattacks and cyber security; Emerging trends and recent developments”, Energy Reports, Volume 7, (2021), Pages 8176-8186, ISSN 2352-4847, https://doi.org/10.1016/j.egyr.2021.08.126.

[7]. Ma, L., Gao, Y., Zhao, D., Zhao, ”Research on SQL injection attack and prevention technology based on web.”, International Conference on Computer Network, Electronic and Automation (ICCNEA), pp. 176–179 (2019)

[8]. Mona Alsalamah 1, Huda Alwabli 1, Hutaf Alqwifli 1, and Dina M. Ibrahim, ”A Review Study on SQL Injection Attacks, Prevention and Detection”, The ISC Int’l Journal of Information Security, November (2021), Volume 13, pp. 1-9

[9]. Raniah Alsahafi, ”SQL Injection Attacks: Detection And PreventionTechniques”, International Journal of Scientific And Technology Research, Volume 8, Issue 01, January (2019). pp. 182-185

[10]. https://owasp.org/www-project-web-security-testing guide/latest/4- Web Application Security Testing/07-Input Validation Testing/05.7- Testing for ORM Injection

[11]. https://deep4k.medium.com/orm-injection-80ffa48d305e

[12]. Parveen SULTANA and Nishant SHARMA, “Prevention of SQL Injection Using a Comprehensive Input Sanitization Methodology”, Recent Developments in Electronics and Communication Systems (2023), pp. 276- 282, doi:10.3233/ATDE221269

[13]. https://offensive360.com/second-order-sql-injection-attack/, December 21, (2021).

[14]. V. Clincy and H. Shahriar, ”Web Application Firewall: Network Security Models and Configuration,” 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, (2018), pp. 835-836, doi: 10.1109/COMPSAC.2018.00144.

[15]. Saher Manaseer and Ahmad K. Al Hwaitat, “Centralized Web Application Firewall Security System”, Modern Applied Science (2018); Vol. 12, No. 10; 2018

[16]. Open Web Application Security Project (OWASP), ”The Open Web Application Security Project (OWASP): SQL Injections as Critical Weakness in

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1982

Web-Based Systems", (2023), https://owasp.org/www-project-top-ten/
[17]. Alsmadi, I, AlEroud, A & Saifan, AA 2021, "Fault-based testing for discovering SQL injection vulnerabilities in web applications", International Journal of Information and Computer Security, (2021) vol. 16, no. 1-2, pp. 51-62. https://doi.org/10.1504/IJICS.2021.117394
[18]. Umar Farooq, "Ensemble Machine Learning Approaches for Detection of SQL Injection Attack", (2021) Tehni˘cki glasnik. 15. 112-120. 10.31803/tg20210205101347.
[19]. Zhou, Fei, Honghai Fan, Yuhan Liu, Hongbao Zhang, and Rongyi Ji. (2023). "Hybrid Model of Machine Learning Method and Empirical Method for Rate of Penetration Prediction Based on Data Similarity" Applied Sciences 13, no. 10: 5870. https://doi.org/10.3390/app13105870
[20]. B. Brindavathi, A. Karrothu and C. Anilkumar, "An Analysis of AI-based SQL Injection (SQLi) Attack Detection," 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), Trichy, India, (2023), pp. 31-35, doi: 10.1109/ICAISS58487.2023.10250505.
[21]. Demilie, W.B., Deriba, F.G. Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques. J Big Data 9, 124 (2022). https://doi.org/10.1186/s40537-022-00678-0
[22]. Kakisim, A.G. A deep learning approach based on multi-view consensus for SQL injection detection. Int. J. Inf. Secur. 23, 1541–1556 (2024). https://doi.org/10.1007/s10207-023-00791-y
[23]. Peng Tang, Weidong Qiu, Zheng Huang, Huijuan Lian, Guozhen Liu, Detection of SQL injection based on artificial neural network, Knowledge-Based Systems, Volume 190, (2020), 105528, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2020.105528.
[24]. B. I. Mukhtar and M. A. Azer, "Evaluating the Modsecurity Web Application Firewall Against SQL Injection Attacks," 2020 15th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, (2020), pp. 1-6, doi: 10.1109/ICCES51560.2020.9334626.
[25]. F.F.Fadlalla and H.T.Elshoush, "Input Validation Vulnerabilities in Web Applications: Systematic Review, Classification, and Analysis of the Current State of the Art", IEEE Access, (2023), Digital Object Identifier 10.1109/ACCESS.2023.3266385.
[26]. M.H.Ali and M.N.Jasim, "Review of SQL injection attacks: Detection, to enhance the security of the website from client-side attacks", Int. J. Nonlinear Anal. Appl. 13 (2022) 1, 3773-3782 ISSN: 2008-6822 (electronic) http://dx.doi.org/10.22075/ijnaa.2022.6152
[27]. R.F.Sidik, S.N.Yutia and R.Z.Fathiyana, "The Effectiveness of Parameterized Queries in Preventing SQL Injection Attacks at Go", Proceedings of the International Conference on Enterprise and Industrial Systems (ICOEINS 2023), 10.2991/978-94-6463-340-5 18
[28]. Lu, Dongzhe, Jinlong Fei, and Long Liu. 2023. "A Semantic Learning-Based SQL Injection Attack Detection Technology" Electronics 12, no. 6: 1344, (2023). https://doi.org/10.3390/electronics12061344
[29]. https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/
[30]. https://docs.sqlalchemy.org/en/20/orm/
[31]. https://www.analyticsvidhya.com/blog/2022/07/a-brief-introduction-to-sqlalchemy/
[32]. Ochieng, Fredrick and Kaburu, Dennis and John, Ndia G., "AutomationBased User Input SQL Injection Detection and Prevention Framework", (May 2, 2023). Computer and Information Science; Vol. 16, No. 2; (2023); https://doi.org/10.5539/cis.v16n2p51, Available at SSRN: https://ssrn.com/abstract=4439431
[33]. Angshuman Jana and Dipendu Maity, "Code-based Analysis Approach to Detect and Prevent SQL Injection Attacks" , 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), (2022), DOI: 10.1109/ICCCNT49239.2020.9225575
[34]. M. Hasan, Z. Balbahaith and M. Tarique, "Detection of SQL Injection Attacks: A Machine Learning Approach," 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, (2019), pp. 1-6, doi: 10.1109/ICECTA48151.2019.8959617.
[35]. J. M. Alkhathami and S. M. Alzaharani, "Detection of Sql Injection Attacks Using Machine Learning in Cloud Computing Platform", Journal of Theoretical and Applied Information Technology, (2022), pp. 5446 – 5459.
[36]. Reddy, M., Latchoumi, T., Balamurugan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention." Eur. J. Mol. Clin. Med. 7, 3543–3553 (2020)
[37]. Pattewar, T., Patil, H., Patil, H., Patil, N., Taneja, M., Wadile, T.: "Detection of SQL injection using machine learning: a survey". Int. Res. J. Eng. Technol. (IRJET) 6, 239–246 (2019)