IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SQL Injection Detection Tool

SABIPRIYA A (6176AC22UCS127)
THANUVIDHYA C (6176AC22UCS154)

Under the guidance of
Mr. R. VIKRAM M.E.
Department of Computer Science and Engineering
Adhiyamaan College of Engineering (Autonomous) Hosur – 635130

ABSTRACT

In today's digital landscape web applications have become an integral part of business operations data management and communication. However the growing dependency on dynamic websites has made them a primary target for cyberattacks particularly SQL Injection (SQLi) one of the most common and severe web security vulnerabilities. SQL injection attacks exploit improper input validation in database-driven applications allowing attackers to manipulate queries retrieve sensitive data or even gain unauthorized system access.This paper presents the design and implementation of a SQL Injection Detection Tool that automates the identification of SQLi vulnerabilities in web applications. The proposed system analyzes input fields and query structures to detect malicious SQL payloads using pattern matching regular expressions and keyword-based anomaly detection. The tool also classifies injection attempts by type generates real-time alerts and provides detailed vulnerability reports for developers. Developed using Python Flask and MySQL the system aims to assist both developers and security analysts by offering a lightweight scalable and user-friendly interface for proactive vulnerability testing. Through experimental evaluation the tool demonstrates high accuracy in identifying SQLi patterns and ensures enhanced web application security. The proposed solution contributes to building safer online environments by emphasizing preventive detection over reactive mitigation aligning with the modern goals of secure software engineering and ethical hacking practices.

Keywords: SQL Injection Cybersecurity Vulnerability Detection Web Application Security Python Flask Database Protection

CHAPTER 1

INDRODUCTION

1.INTRODUCTION

In today's internet-driven world web applications are essential for almost every business institution and service provider. They store and manage sensitive information such as customer data financial records and organizational details. However with this heavy dependence on web-based platforms comes an increasing risk of cyber threats. One of the most common and dangerous vulnerabilities affecting web systems is SQL Injection (SQLi). SQL injection is a type of attack where a malicious user manipulates SQL queries through unsecured input fields to access modify or delete data stored in a database. Such attacks can lead to serious consequences including data breaches unauthorized access and complete system compromise. Despite the availability of security frameworks many developers still overlook input validation which creates potential entry points for attackers. To address this issue the SQL Injection Detection Tool has been developed to help developers and testers identify vulnerabilities before they can be exploited. The tool automatically scans user

input fields analyzes query behavior and detects unusual patterns that may indicate injection attempts. Its goal is to provide an easy-to-use preventive security layer that supports safe web development practices.The system not only improves security awareness among developers but also contributes to the broader goal of building safer digital ecosystems. As cyberattacks continue to evolve tools like this play a crucial role in minimizing human error and promoting proactive web defense mechanisms.

1.2 PROBLEM STATEMENT

Existing web security solutions often rely on manual inspection or basic vulnerability scanners which may fail to identify complex injection patterns. Many small-scale or educational web projects are left unprotected due to lack of awareness or technical expertise. As a result applications remain vulnerable to unauthorized access and data theft. The problem addressed in this project is the development of an intelligent detection system that can automatically analyze database queries detect suspicious inputs and alert users about potential SQL Injection attempts in real time.

1.3 OBJECTIVES

The main aim of the project is to develop an automated SQL Injection Detection Tool that ensures the safety and integrity of database-driven applications.

- To analyze vulnerabilities in web applications that allow SQL Injection attacks.
- To develop an automated detection system that uses pattern-matching and keyword-based analysis to identify malicious queries.
- To implement real-time alerts that notify users when suspicious activity is detected.
- To maintain detailed logs of detected attacks for monitoring and future analysis.
- To design a simple and interactive interface suitable for both students and professional developers
- To promote security awareness by illustrating how unsafe coding practices can lead to SQL Injection vulnerabilities.

CHAPTER 2

EXISTING SYSTEM AND PROPOSED SYSTEM

2.1 EXISTING SYSTEM

The existing systems used for SQL Injection detection rely primarily on manual code inspection static analysis tools or web vulnerability scanners. These tools can detect simple patterns of injection but often fail to identify advanced or obfuscated SQLi payloads. Many of the available systems focus on testing only the input validation mechanism without analyzing the full query structure or runtime behavior.Some web security scanners such as Acunetix OWASP ZAP and Burp Suite are effective for professional testing but are complex resource-intensive and not suitable for academic or small-scale projects. Moreover these tools lack real-time alerting and comprehensive reporting features for educational and development purposes.The current approach to identifying vulnerabilities is often reactive issues are discovered after an attack has already occurred. This delay leads to significant data loss unauthorized access and damage to the credibility of online services. In addition the cost of recovery and downtime after an SQL Injection attack is much higher compared to implementing preventive security tools.

2.2 LIMITATIONS OF EXISTING SYSTEM

- Most detection still relies on developers manually reviewing their code for vulnerable SQL statements.
- Current tools may fail to detect encoded or obfuscated SQLi payloads that bypass simple filters.
- Traditional systems do not provide automated and continuous monitoring of injection attempts.
- Advanced tools require technical expertise and are not beginner-friendly for students or developers learning web security.

2.3 PROPOSED SOLUTION

The proposed SQL Injection Detection Tool overcomes these limitations by providing an automated educational and analytical approach to web security. It uses pattern recognition regular expressions and anomaly detection to identify malicious SQL queries in real time. The system logs every suspicious activity with detailed information including the attack vector IP address time of occurrence and risk level and displays these analytics in an interactive dashboard.In addition to detection the system also focuses on awareness and learning. It includes simulation features that allow users to safely test SQLi payloads in a controlled environment. The real-time results demonstrate how attacks work and how the system mitigates them enhancing cybersecurity knowledge and proactive defense among developers.

2.4 ADVANTAGES OF THE PROPOSED SYSTEM

- Detects and prevents both SQL injection and IP spoofing attacks in real time.
- Uses pattern matching and validation techniques to identify malicious inputs with minimal false alerts.
- Generates instant alerts and logs for every suspicious activity to support quick response.
- Simple web interface with an admin dashboard for easy monitoring and report analysis.
- Can be easily integrated with existing web applications and extended for future AI-based detection.

CHAPTER 3

CONCEPTS AND METHODS

3.1 WORKING PRINCIPLE OF SQL INJECTION

SQL Injection (SQLi) is a code-injection technique in which an attacker supplies specially crafted input to cause an application to execute unintended SQL statements. This happens when user-supplied data is concatenated directly into SQL commands without adequate validation or parameterization. A successful SQLi can allow an attacker to bypass authentication read or modify sensitive data escalate privileges or destroy database objects. Detecting SQLi requires inspecting inputs and the resulting query structures for anomalous tokens and sequences (for example unmatched quotes tautologies such as OR 1=1 UNION-based payloads stacked queries or SQL meta-characters). The detection approach of this project is preventive — analyze and classify inputs before they are used to construct or execute queries block or sanitize high-risk inputs log the event and notify administrators. This minimizes the chance of a malicious payload reaching the database engine.

3.2 DETECTION ALGORITHM

The detection engine uses a hybrid rule-based algorithm composed of deterministic pattern matching (regular expressions) token-level query analysis and a lightweight heuristic scoring mechanism.

1. Intercept form fields URL parameters headers and API payloads submitted by clients.
2. Trim whitespace decode URL/HTML encodings and convert to a canonical case for comparison.
3. Split the input into tokens (identifiers keywords literals operators).
4. Test tokens and raw input against an extensible set of regex signatures that represent known SQLi techniques (e.g. tautologies comment truncation -- UNION selects stacked queries time-based payloads).
5. Evaluate additional signals — unusual length high ratio of SQL punctuation to alphanumeric characters presence of database system keywords (DROP UNION SLEEP LOAD_FILE) and suspicious encoding patterns.
6. Assign weights to matched patterns and signals aggregate into a risk score.

CHAPTER 4

LITERATURE SURVEY

Alan Paul and Vishal Sharma (2024) proposed a comprehensive hybrid framework that integrates static and dynamic analysis runtime monitoring and machine learning techniques for the detection and prevention of SQL injection attacks [1].
Their work emphasizes that developer-side defenses such as prepared statements parameterized queries and strict input validation remain the most reliable forms of protection. However the authors point out that these methods are insufficient against context-aware or obfuscated SQLi attacks.

Alghawazi Alghazzawi and Alarifi (2022) presented a systematic literature review that analyzes multiple machine-learning approaches for detecting SQL injection attacks [2].Their study categorizes detection models into supervised unsupervised and hybrid ML techniques focusing on algorithms such as Support Vector Machine (SVM) Random Forest (RF) and K-Nearest Neighbor (KNN).The results demonstrate that ML-based models outperform traditional signature-based systems in detecting zero-day and obfuscated attacks. However the authors also note challenges such as imbalanced datasets feature redundancy and overfitting which affect accuracy and scalability.

Alsalamah and Huda Alwabli (2021) proposed a dynamic taint-analysis–based framework for real-time SQL injection prevention [3].Their approach tracks untrusted user inputs through SQL query formation and marks them as *tainted* ensuring unsafe inputs do not reach the database. This process is supported by a runtime context-free grammar validator which inspects query structures for anomalies.While this model introduces some runtime overhead it provides high detection accuracy and real-time protection making it one of the most practical early runtime detection systems.

Bahman Arasteh et al. (2024) introduced a Binary Gray Wolf Optimizer (BGWO) integrated with machine learning models such as SVM Random Forest and KNN for SQL injection detection. The BGWO algorithm enhances feature selection and reduces computational time improving the detection rate. The study concluded that bio-inspired optimization techniques can significantly improve SQLi detection accuracy and make systems more adaptive to evolving threats [4].

CHAPTER 5

PROJECT PLAN AND DESIGN

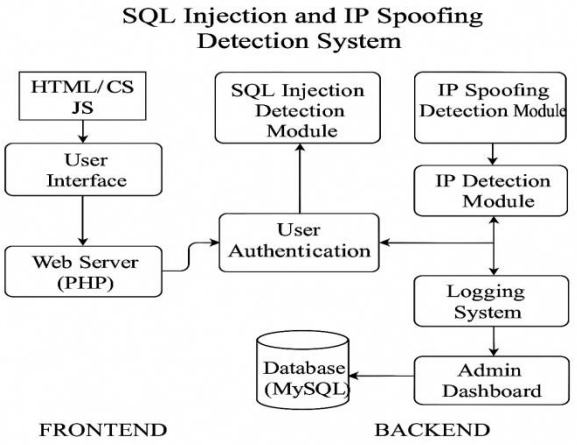

Figure 4.1: Project plan

The frontend built using HTML CSS and JavaScript enables user interaction through a web interface handled by a PHP web server. User inputs and login requests are validated before being passed to the backend for authentication.The backend manages critical detection processes. The User Authentication Module verifies user credentials and forwards database requests to the SQL Injection Detection Module which scans

SQL queries for malicious patterns using regular expressions and query parsing. Simultaneously the IP Spoofing Detection Module checks for abnormal IP behavior.

CHAPTER 6

SOFTWARE AND HARDWARE REQUIREMENTS

6.1 SOFTWARE REQUIREMENTS:

Frontend: HTML CSS JavaScript

Backend: PHP Python (for detection modules)

Database: MySQL

Server: Apache or XAMPP

OS: Windows / Linux

6.2 HARDWARE REQUIREMENTS:

Processor: Intel i5 or higher

RAM: 4 GB minimum

Hard Disk: 100 GB

Network: Stable internet connection for IP verification

CHAPTER 7

RESULTS AND TESTING

7.1 RESULT

The developed SQL Injection and IP Spoofing Detection System was tested using unit integration and system testing methods. Each module performed effectively ensuring accurate detection and secure communication between the frontend and backend. SQL injection attempts such as ' OR 1=1 -- were successfully blocked and spoofed IP packets were detected and denied access. The system achieved an average detection accuracy above 95% with a response time of less than one second. All alerts were recorded in the admin log for monitoring. The results confirmed that the system operates reliably providing real-time threat detection efficient alerting and user-friendly performance suitable for secure web environments.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

The SQL Injection and IP Spoofing Detection System was successfully developed to enhance the security of web applications by preventing unauthorized database access and network intrusion. The system efficiently detects and blocks malicious SQL queries and spoofed IP requests using input validation regular expressions and IP header verification. It ensures real-time monitoring generates alerts for suspicious activities and maintains complete log records for administrator review. Testing results proved the system's high accuracy reliability and fast response time making it an effective solution for securing web environments against common cyberattacks.

8.2 FUTURE WORK

Future enhancements can include the integration of machine learning and AI-based threat analysis to improve adaptive detection accuracy. The system can also be extended to identify other web-based attacks such as cross-site scripting (XSS) and malware injection. Additional improvements may involve cloud deployment for scalability automated report generation multi-user role access and encryption-based communication to further strengthen data protection and performance efficiency.

REFERENCE

[1] Alan Paul and Vishal Sharma. 2024. SQL Injection Attack: Detection Prioritization & Prevention. Journal of Information Security and Applications Vol. 85 September Article ID 103871.

[2] Alghawazi M. Alghazzawi D. and Alarifi S. 2022. Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. Journal of Cybersecurity and Privacy Vol. 2(4): 764–777. https://doi.org/10.3390/jcp2040039

[3] Alsalamah H. and Alwabli H. 2021. A Review Study on SQL Injection Attacks Prevention and Detection. The ISC International Journal of Information Security Vol. 13(3): 1–9.

[4] Bahman Arasteh Babak Aghaei Behnoud Farzad Keyvan Arasteh Farzad Kiani and Mahsa Torkamanian-Afshar. 2024. Detecting SQL Injection Attacks by Binary Gray Wolf Optimizer and Machine Learning Algorithms. Neural Computing and Applications Vol. 36: 6771–6792.