International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

Machine Learning-Based Real-Time SQL Injection Detection System Using TF-IDF and Naive Bayes

GUDE BALAMBIKA

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

V.SARALA

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

SQL Injection (SQLi) remains one of the most critical and prevalent security vulnerabilities in modern web applications. Attackers exploit improper input validation mechanisms to manipulate backend databases, leading to unauthorized data access, data leakage, and system compromise. Traditional rule-based detection systems often fail to identify evolving attack patterns, making it necessary to adopt intelligent, adaptive approaches. This paper presents a machine learning-based SQL Injection Detection System that leverages Natural Language Processing (NLP) techniques and probabilistic classification to identify malicious queries in real time. The proposed system utilizes Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction, transforming raw SQL queries into numerical vectors that capture the importance of terms within the dataset. A Multinomial Naive Bayes classifier is then trained on labeled datasets containing both legitimate and malicious SQL queries. The model learns statistical patterns associated with SQL injection attempts, enabling it to classify unseen queries effectively. A user-friendly graphical interface is developed using Tkinter, allowing users to input SQL queries and receive instant feedback regarding their safety. The system provides not only classification results (SAFE or ATTACK) but also confidence scores, enhancing transparency and usability. Additionally, a query history feature is implemented to track previous inputs, aiding in monitoring and analysis.

The proposed system demonstrates several advantages, including high efficiency, low computational complexity, and ease of deployment. Unlike traditional signature-based systems, it can generalize and detect previously unseen attack patterns. The use of machine learning significantly improves detection accuracy and adaptability in dynamic environments. Experimental results indicate that the model performs effectively on real-world datasets, achieving reliable classification with minimal latency. The integration of TF-IDF and Naive Bayes proves to be a lightweight yet powerful approach for text-based intrusion detection tasks. In conclusion, this work highlights the potential of machine learning in enhancing cybersecurity mechanisms. The developed system provides a scalable and efficient solution for SQL injection detection, suitable for integration into

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

web applications and security tools. Future enhancements may include deep learning models, real-time deployment in web servers, and continuous learning mechanisms to further improve detection capabilities.

Keywords: SQL Injection, Cyber security, Machine Learning, Naive Bayes, TF-IDF, Web Security, Intrusion Detection, Natural Language Processing, Query Analysis, Threat Detection

I. INTRODUCTION

With the rapid growth of web-based applications and online services, ensuring data security has become a critical concern. Among various cyber threats, SQL Injection (SQLi) attacks are one of the oldest yet most dangerous forms of attack targeting database-driven applications. These attacks exploit vulnerabilities in input handling mechanisms, allowing attackers to execute malicious SQL queries that can manipulate or retrieve sensitive data. SQL injection attacks typically occur when user inputs are directly embedded into SQL queries without proper validation or sanitization. For example, an attacker may input specially crafted strings such as `' OR '1'='1` to bypass authentication systems. Such attacks can lead to severe consequences, including unauthorized access, data theft, data corruption, and even complete system compromise. Traditional methods for detecting SQL injection rely heavily on rule-based or signature-based approaches. These systems use predefined patterns to identify malicious inputs. While effective against known attacks, they struggle to detect new and evolving attack techniques. This limitation necessitates the development of intelligent systems capable of learning and adapting to new threats.

Machine Learning (ML) offers a promising solution to this problem. By analyzing patterns in historical data, ML models can learn to distinguish between normal and malicious queries. In particular, Natural Language Processing (NLP) techniques can be applied to SQL queries, treating them as textual data. This allows for the extraction of meaningful features that can be used for classification. In this project, a machine learning-based SQL Injection Detection System is proposed. The system uses TF-IDF vectorization to convert SQL queries into numerical representations, capturing the importance of keywords and patterns. A Multinomial Naive Bayes classifier is then trained on these features to identify malicious queries. The system is implemented with a graphical user interface (GUI) using Tkinter, making it accessible and easy to use. Users can input SQL queries and receive instant feedback regarding their safety. The system

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

also displays confidence levels and maintains a history of analyzed queries. The main objective of this project is to develop an efficient, lightweight, and accurate SQL injection detection system that can operate in real time. By leveraging machine learning techniques, the system aims to overcome the limitations of traditional methods and provide a more robust solution to modern cybersecurity challenges.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

SQL Injection detection has been widely studied in the field of cybersecurity, leading to the development of various approaches ranging from traditional rule-based systems to advanced machine learning models. Early detection systems primarily relied on signature-based and rule-based techniques. Tools such as Web Application Firewalls (WAFs) used predefined patterns and rules to identify malicious queries. While these methods were effective against known attack patterns, they lacked adaptability and were unable to detect zero-day attacks. Moreover, maintaining and updating rule sets required significant manual effort. To overcome these limitations, researchers introduced anomaly-based detection systems. These systems establish a baseline of normal behavior and flag deviations as potential attacks. Although anomaly detection improves the ability to detect unknown attacks, it often suffers from high false positive rates. With the advancement of machine learning, classification-based approaches have gained popularity. Algorithms such as Decision Trees, Support Vector Machines (SVM), and Naive Bayes have been applied to SQL injection detection. Among these, Naive Bayes is particularly favored due to its simplicity, efficiency, and effectiveness in text classification tasks.

Recent studies have also explored the use of Natural Language Processing (NLP) techniques to analyze SQL queries. TF-IDF is one of the most commonly used feature extraction methods, as it captures the importance of terms relative to the dataset. Combining TF-IDF with machine learning classifiers has shown promising results in detecting SQL injection attacks. Deep learning approaches, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have also been proposed. These models can capture sequential patterns in queries, providing higher accuracy. However, they require large datasets and high computational resources, making them less suitable for lightweight applications. Hybrid approaches combining rule-based and machine learning techniques have also been explored. These systems aim to leverage the strengths of both methods, improving detection accuracy while maintaining efficiency. Despite these advancements, challenges remain, including dataset imbalance, evolving attack techniques, and the need for real-time detection. The proposed system addresses these challenges by using a lightweight yet effective machine learning model that can be easily integrated into real-world applications.

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

III. EXISTING SYSTEM

Existing SQL Injection detection systems primarily rely on traditional security mechanisms such as input validation, parameterized queries, and Web Application Firewalls (WAFs). These methods aim to prevent malicious inputs from reaching the database by enforcing strict validation rules and filtering suspicious patterns. Signature-based detection systems are widely used in current applications. These systems maintain a database of known attack patterns and compare incoming queries against these patterns. While effective for known threats, they are unable to detect new or obfuscated attacks, making them less reliable in dynamic environments.

Another common approach is the use of static code analysis tools, which scan application code for vulnerabilities. Although useful during the development phase, these tools cannot provide real-time protection against runtime attacks. Anomaly-based detection systems attempt to identify unusual query behavior by comparing it with normal patterns. While this method can detect unknown attacks, it often produces a high number of false positives, affecting usability and trust in the system. Moreover, existing systems often lack user-friendly interfaces and real-time feedback mechanisms. They may operate in the background without providing clear insights into detected threats, making them less accessible for non-expert users. In summary, current systems face limitations such as lack of adaptability, high false positives, and inability to detect zero-day attacks. These challenges highlight the need for intelligent, machine learning-based solutions that can dynamically learn and adapt to evolving threats, as proposed in this project.

IV. PROPOSED METHOD

The proposed system introduces a **machine learning-based SQL Injection Detection System** designed to identify malicious SQL queries in real time using Natural Language Processing (NLP) techniques. Unlike traditional rule-based systems, this approach leverages statistical learning to detect both known and unknown attack patterns effectively.

The system primarily uses **TF-IDF (Term Frequency–Inverse Document Frequency)** for feature extraction. SQL queries are treated as textual data, and TF-IDF converts them into numerical vectors representing the importance of words within the dataset. This helps in capturing meaningful patterns such as suspicious keywords (OR, UNION, DROP, etc.) commonly used in SQL injection attacks.

A **Multinomial Naive Bayes classifier** is employed for classification. This probabilistic algorithm is highly efficient for text classification tasks and works well with high-

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

dimensional sparse data such as TF-IDF vectors. The model is trained on a labeled dataset containing both legitimate and malicious SQL queries, enabling it to learn distinguishing characteristics.

The system includes a **graphical user interface (GUI)** developed using Tkinter, allowing users to input SQL queries and receive immediate classification results. It displays whether the query is safe or malicious along with a confidence score. A query history feature is also implemented to log previously analyzed queries.

The proposed system addresses limitations of traditional systems by providing:

- Real-time detection
- Adaptability to new attack patterns
- High accuracy with low computational cost

Machine learning-based approaches have shown strong performance in detecting SQL injection attacks by analyzing query patterns and features . This system provides a scalable and efficient solution suitable for integration into modern web security frameworks.

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

IMPLEMENTATION

The implementation of the SQL Injection Detection System is carried out using Python, integrating machine learning libraries and GUI frameworks to create a complete application. The process begins with **dataset loading and preprocessing**. The dataset (sql_injection.csv) contains two columns: query and label, where the label indicates whether the query is safe (0) or an attack (1). The system ensures compatibility by handling multiple encodings such as UTF-8 and Latin-1. Data validation is performed to ensure the presence of required columns. Next, **feature extraction** is performed using the TF-IDF vectorizer from the Scikit-learn library. This converts textual SQL queries into numerical feature vectors. The vectorizer is configured with parameters such as:

- Stop word removal
- Maximum feature size (e.g., 3000 features)

This step transforms the dataset into a format suitable for machine learning algorithms.

The transformed data is then used to train a **Multinomial Naive Bayes classifier**. This algorithm calculates the probability of a query belonging to a class (safe or malicious) based on feature frequencies. It is particularly suitable for text classification problems due to its simplicity and efficiency.

Once the model is trained, it is integrated into a **real-time detection function**. When a user enters a SQL query through the GUI:

1. The input query is cleaned and preprocessed
2. It is transformed using the trained TF-IDF vectorizer
3. The model predicts whether the query is safe or malicious
4. A confidence score is calculated using prediction probabilities

The system then updates the GUI dynamically:

- Displays status (SAFE or ATTACK)
- Shows confidence percentage
- Logs the query in the history section

The graphical interface is implemented using **Tkinter**, providing an intuitive and interactive experience. The interface includes input fields, buttons, status labels, confidence indicators, and a history display. Error handling mechanisms are incorporated to manage missing datasets and invalid inputs. The system ensures robustness by

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

preventing crashes and providing meaningful error messages. The implementation is lightweight and does not require high computational resources. Studies have shown that Naive Bayes models can achieve high accuracy in SQL injection detection while maintaining low complexity. Overall, the system integrates machine learning and user interface components effectively, providing a practical solution for real-time SQL injection detection.

V. ALGORITHMS

The proposed system uses two primary algorithms: **TF-IDF Vectorization** and **Multinomial Naive Bayes Classification**.

1. TF-IDF Algorithm

TF-IDF (Term Frequency–Inverse Document Frequency) is used for feature extraction. It converts text data into numerical form by evaluating the importance of words.

- **Term Frequency (TF):** Measures how frequently a term appears in a query
- **Inverse Document Frequency (IDF):** Measures how important a term is across all queries

The TF-IDF score is calculated as:

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

This method helps highlight important keywords such as SQL operators often used in injection attacks.

2. Multinomial Naive Bayes Algorithm

The Multinomial Naive Bayes classifier is a probabilistic model based on Bayes' theorem. It assumes independence between features and calculates the probability of a query belonging to a specific class.

The classification is based on:

$$P(\text{Class} \mid \text{Query}) \propto P(\text{Query} \mid \text{Class}) \times P(\text{Class})$$

Steps involved:

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

1. Calculate prior probabilities of each class
2. Compute likelihood of features given the class
3. Apply Bayes' theorem to calculate posterior probability
4. Assign the class with the highest probability

Naive Bayes is widely used for text classification tasks due to its efficiency and effectiveness. Research shows it performs well in SQL injection detection by analyzing tokenized query patterns .

Advantages:

- Fast computation
- Works well with high-dimensional data
- Requires less training data

These algorithms together enable accurate and efficient detection of SQL injection attacks.

VI. SYSTEM DESIGN

The system design follows a modular architecture consisting of multiple components that interact to perform SQL injection detection efficiently.

1. Input Layer

The system begins with the user input interface, where SQL queries are entered through the GUI. This layer is responsible for collecting user input and forwarding it to the processing module.

2. Data Preprocessing Module

The input query is cleaned and standardized. This includes:

- Removing unnecessary spaces
- Converting text into a consistent format
- Handling special characters

This ensures that the query is suitable for feature extraction.

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

3. Feature Extraction Module

The TF-IDF vectorizer transforms the input query into a numerical vector. This step is crucial as machine learning models cannot process raw text directly.

4. Classification Module

The processed vector is passed to the trained Naive Bayes model. The model evaluates the query and classifies it as either:

- Safe Query
- SQL Injection Attack

It also calculates the probability score for each class.

5. Output Layer

The results are displayed on the GUI, including:

- Classification result
- Confidence score
- Visual status indicator

6. History Module

All analyzed queries are stored in a history log for future reference. This helps in monitoring and analysis.

7. Error Handling Module

Handles:

- Missing dataset errors
- Invalid inputs
- Runtime exceptions

Architecture Type

The system follows a **client-side standalone architecture**, where all processing occurs locally. This ensures:

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

- Faster response time
- No dependency on external servers
- Enhanced privacy

Data Flow

1. User inputs SQL query
2. Query is preprocessed
3. TF-IDF converts query to vector
4. Model predicts class
5. Result displayed

Machine learning-based system designs are increasingly preferred because they can adapt to evolving attack patterns and improve detection accuracy over time .

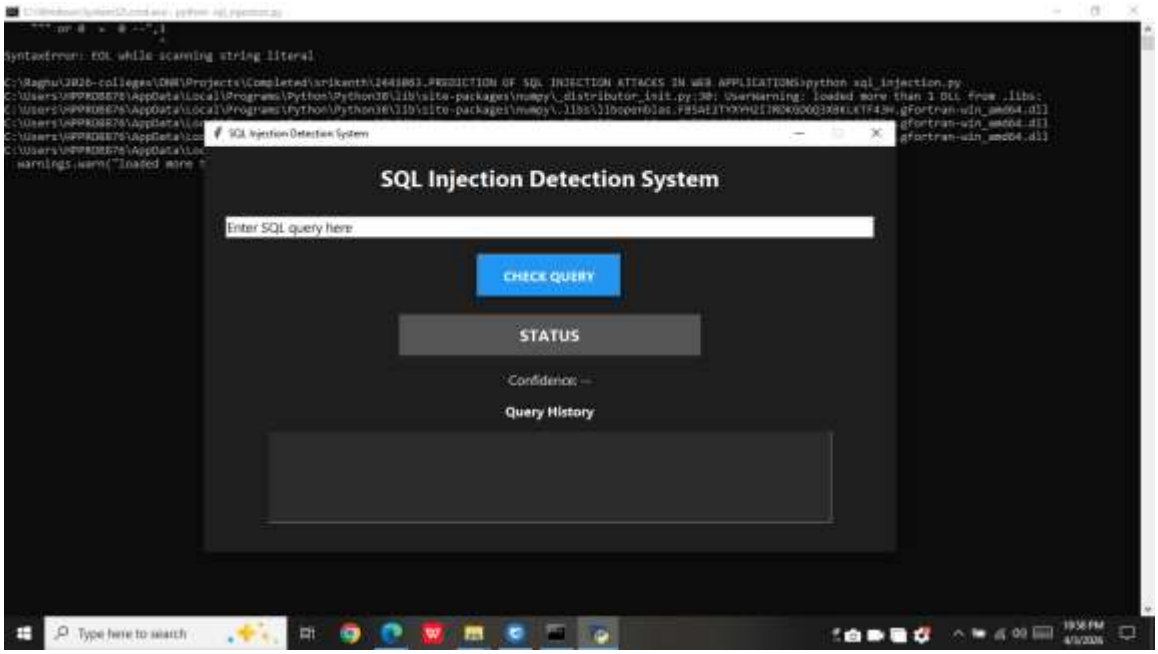
International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

SYSTEM DESIGN IMAGES



International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

VII. CONCLUSION

SQL Injection remains one of the most serious threats to web application security, capable of causing significant data breaches and system compromise. Traditional detection methods, such as rule-based and signature-based systems, are limited in their ability to detect new and evolving attack patterns.This project presents a machine learning-based SQL Injection Detection System that effectively addresses these limitations. By leveraging TF-IDF for feature extraction and the Multinomial Naive Bayes classifier for classification, the system provides accurate and efficient detection of malicious queries.The integration of a graphical user interface enhances usability, allowing users to interact with the system easily and obtain real-time feedback. The inclusion of confidence scores and query history further improves transparency and monitoring capabilities.

The system demonstrates several advantages, including:

- High detection accuracy
- Low computational complexity
- Real-time processing
- Adaptability to new attack patterns

Research indicates that machine learning techniques significantly improve SQL injection detection compared to traditional approaches by analyzing patterns and behaviors in query data .However, the system can be further improved by incorporating advanced techniques such as deep learning, ensemble models, and real-time integration with web servers. Future work may also include expanding the dataset and implementing continuous learning mechanisms.In conclusion, the proposed system provides a robust, scalable, and efficient solution for detecting SQL injection attacks, contributing to enhanced cybersecurity in modern web applications.

International Journal of

DATA SCIENCE AND IOT MANAGEMENT SYSTEM

Peer Reviewed, Referred & Indexed Journal

ISSN: 3068-272X ww.ijdim.com Original Research Paper

REFERENCES

1. Alghawazi, M., et al. (2022). *Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Review*.
2. Oudah, M., & Marhusin, M. (2024). *SQL Injection Detection using Machine Learning: A Review*.
3. Arnap, A., & Kusrini. (2024). *Naïve Bayes with SMOTE for SQL Injection Detection*.
4. Singh, B. P., & Singhal, M. (2024). *Detection of SQL Injection using ML Techniques*.
5. Lu, Z. (2025). *SQL Injection Detection using Naïve Bayes Classifier*.
6. Oudah, M. et al. (2024). *ML-based SQL Injection Detection Review*.
7. Gupta, A. et al. (2023). *Machine Learning Methodology for SQL Injection Detection*.
8. Roy, P. et al. (2022). *SQL Injection Attack Detection using ML Classifiers*.
9. Pramono, P. et al. (2024). *Naïve Bayes vs SVM for SQL Injection Detection*.
10. Singh, B. P. et al. (2024). *ML Techniques for SQL Injection Detection*.
11. Herman, H. et al. (2023). *Vulnerability Detection using Naïve Bayes*.
12. Liu, M. et al. (2020). *DeepSQLi: Deep Learning for SQL Injection Testing*
13. Dasari, N. S. et al. (2025). *Generative Models for SQL Injection Detection*
14. Erdodi, L. et al. (2021). *Reinforcement Learning for SQL Injection Simulation*
15. Farid, D. M. et al. (2010). *Naïve Bayes-based Intrusion Detection Systems*