World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

SQL injection vulnerability analysis

Smithu B S [1], Leela C P [2,*] and Nagashree N [3]

[1] *Department of Computer Science and Engineering, Government Polytechnic Channasandra-560067, Karnataka, India*
[2] *Department of Computer Science and Engineering, DACG, Government Polytechnic, Chikkamagaluru 577101, Karnataka, India*
[3] *Department of Electronics and Communication Engineering, DACG, Government Polytechnic, Chikkamagaluru 577101, Karnataka, India*

World Journal of Advanced Research and Reviews, 2021, 09(01), 312–318

Publication history: Received on 13 January 2021; Revised 25 January 2021; accepted on 29 January 2021

Article DOI: https://doi.org/10.30574/wjarr.2021.9.1.0018

Abstract

Web applications are an integral part of today's digital landscape, serving various functions from e-commerce to social networking. However, they are also prime targets for cyber-attacks, with SQL-Injection vulnerabilities posing a significant threat to their security. This project addresses the critical issue of SQL-Injection vulnerabilities in web applications by offering a comprehensive analysis, leveraging Python and classical machine learning algorithms such as Naïve Bayes. The research method employed in this project involves procuring real-world datasets, conducting data pre-processing, and using decision tree classifiers. These steps collectively provide an automated and scalable solution for identifying, understanding, and mitigating SQL-Injection riskslearning methods like Deep Neural Networks (DNN). A comprehensive comparative analysis of these algorithms has been carried out, assessing their performance based on accuracy metrics.

Keywords: SQL-Injection; vulnerabilities; web Application; Machine Learning; AI

1. Introduction

In the ever-evolving landscape of digital applications and interconnected systems, security remains a paramount concern. As organizations and individuals rely increasingly on web-based applications to handle sensitive information and critical operations, the threat posed by cyber-attacks has grown in both scale and sophistication. One of the most prevalent and potentially devastating vulnerabilities that malicious actors exploit is known as SQL injection as shown in fig 1.1. This vulnerability occurs when improper input validation and sanitization in an application's code enable attackers to manipulate Structured Query Language (SQL) queries in ways unintended by the developers. The result can be unauthorized access, data breaches, and even the compromise of entire systems. Machine learning (ML) plays a crucial role in the context of SQL Injection vulnerability analysis by enhancing the detection and prevention of such security threats. Here are some key aspects of how ML contributes to SQL Injection vulnerability analysis:

The Application SQL-Injection Vulnerability Analysis is a comprehensive process designed to identify, assess, and mitigate the risks associated with SQL injection vulnerabilities within applications. This analysis delves into the intricate interplay between software design, input validation, and database interactions to uncover weaknesses that could be exploited by attackers. By conducting a thorough examination of an application's codebase and its underlying database architecture, security professionals can pinpoint potential points of entry for malicious SQL injection attempts.

* Corresponding author: Leela C P

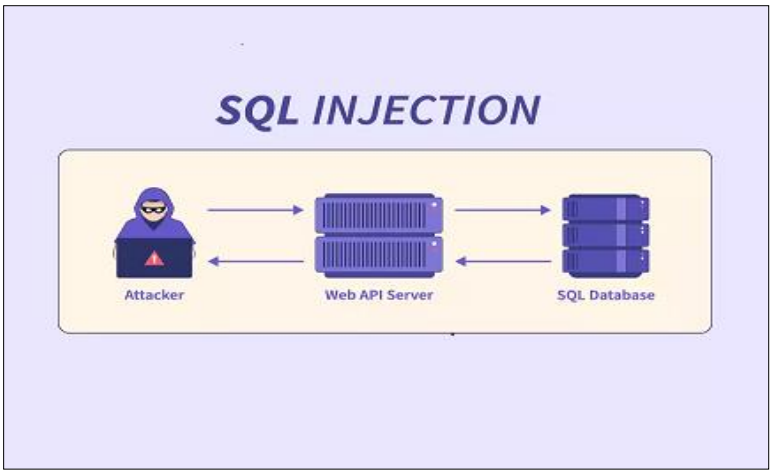

Figure 1 SQL Injection

2. Related Work

I Muyang Liu et al [1], in their paper, the critical concern of security in web applications is addressed, focusing on the particularly damaging SQL injection (SQLi) attack. Automated testing for SQLi vulnerabilities is crucial, but challenging due to the vast array of potential attack variants and semantic possibilities. To tackle this, the paper introduces DeepSQLi, a novel tool based on deep natural language processing. DeepSQLi employs deep learning and sequence prediction to generate test cases for identifying SQLi vulnerabilities. By harnessing the semantic understanding of SQL attacks, DeepSQLi can transform user inputs into related and potentially more sophisticated test cases. Comparative experiments against SQLmap, a prominent SQLi testing automation tool, were conducted on various real-world web applications. The results highlight DeepSQLi's effectiveness, outperforming SQLmap by identifying more vulnerabilities with fewer test cases and at a significantly faster pace.

Rasoul Jahanshahi et al [2], their paper discusses the pressing issue of SQL injection (SQLi) attacks and their impact on web application security. It highlights that existing approaches lack support for object-oriented programming, rendering them ineffective in safeguarding widely-used web applications like Wordpress, Joomla, and Drupal against SQLi attacks. To address this gap, the authors introduce a novel hybrid static-dynamic analysis method tailored for PHP web applications. This approach restricts database access for individual PHP functions and introduces a tool named SQLBlock, designed to minimize the attack surface of vulnerable PHP functions by using query descriptors that showcase their benign functionality.because it relies on extensive features and automatic flow extraction.

In [3] the authors introduce "Spider-Syn," a meticulously curated dataset derived from the Spider benchmark, which serves as the foundation for text-to-SQL translation. In Spider-Syn, natural language questions are deliberately altered from Spider by substituting schema-related words with meticulously chosen synonyms that mirror authentic question paraphrases. The study's findings reveal a marked drop in accuracy when the explicit connection between natural language questions and table schemas is disrupted due to the removal of this schema correspondence. This drop occurs even in scenarios where the synonyms aren't adversarially selected to execute the most severe form of adversarial attacks.

In [4] the authors introduced a novel RL framework for text generation rooted in soft Q-learning (SQL). This approach integrates recent RL advancements, including path consistency learning, to capitalize on the strengths of both on-policy and off-policy updates. By adopting this perspective, the framework effectively learns from sparse rewards, mitigating the challenges faced by existing RL methods in the domain. The proposed approach is applied to a diverse set of text generation tasks, encompassing scenarios like learning from noisy/negative examples, generating adversarial attacks, and prompt creation for language models. Experimental results consistently demonstrate the superiority of the SQL-based approach over both task-specific algorithms and conventional RL methods.

In [5] the authors proposeda robust compiler-level defenses against these attacks and outline supplementary mitigating measures applicable in editors, repositories, and build pipelines. These measures aim to mitigate risks while compilers

are enhanced to counter the Trojan Source threat effectively. The paper also chronicles a coordinated industry-wide vulnerability disclosure effort encompassing compilers, editors, and repositories. This initiative underscores the diverse responses from firms, open-source communities, and stakeholders, exemplifying how different entities address vulnerability disclosure in the context of a widespread and impactful attack vector.

In [6] the authors introduced DaNuoYi, an innovative automated injection testing tool designed to concurrently generate test inputs for various types of injection attacks targeting a WAF. The tool draws inspiration from cross-lingual translation in natural language processing. It recognizes that though test inputs for distinct injection attacks display syntactic differences, they might possess semantic similarities. DaNuoYi leverages this shared semantic understanding across multiple programming languages to create intricate test inputs that reveal injection vulnerabilities challenging to identify otherwise.

In [7] the authors the authors curate ADVETA, a pioneering benchmark designed for evaluating the robustness of Text-to-SQL models in the face of ATPs. This benchmark incorporates realistic and natural ATPs to comprehensively assess model vulnerabilities under real-world conditions. The evaluation reveals significant performance degradation across state-of-the-art models when subjected to ADVETA's perturbations, underscoring their susceptibility. In response to ATPs, the paper introduces an adversarial training example generation framework tailored to enhance contextualization of tabular data. The proposed approach not only significantly bolsters models' robustness against table-side perturbations but also empowers them to better withstand perturbations originating from the natural language side. Experimental results validate the efficacy of the framework, demonstrating marked improvements in both aspects of model robustness.

3. Proposed System

The increasing reliance on web-based applications and interconnected systems has led to a corresponding rise in cyber threats, exposing organizations to potential data breaches, unauthorized access, and compromised system integrity. Among these threats, SQL injection vulnerabilities stand out as a significant concern. These vulnerabilities emerge when improper input validation and sanitization in application code enable attackers to manipulate SQL queries, potentially leading to unauthorized access, data leakage, and system manipulation. Despite existing security measures, organizations continue to grapple with the challenge of identifying, assessing, and mitigating SQL injection vulnerabilities effectively. Current solutions often lack comprehensive methodologies, fail to integrate seamlessly into development lifecycles, or struggle to adapt to evolving attack techniques. There is a pressing need for an advanced and systematic approach to SQL Vulnerability Analysis that addresses these limitations, providing organizations with the means to proactively detect and remediate SQL injection vulnerabilities across various stages of application development and operation.

The objectives include:

- To obtain useful data from SQL injection detection dataset
- Apply pre-processing techniques to the dataset
- To implement Gaussian Naïve Bayes algorithm and train a model
- Obtain model performance metrics.

The fig 2 shows the architecture of the intended project. The dataset is taken from the SQL injection detection dataset.The data is processed with reading and examination. Here, the dataset is visualized using Panda's library in Python. With the newly obtained data, a baseline model is built. Implementation of the Naïve Bayes model is done with the dataset, and model fitting is performed.

Flowchart is very much essential for understanding and also estimating the cost of the project quality. The flowchart of this project is shown in fig .3

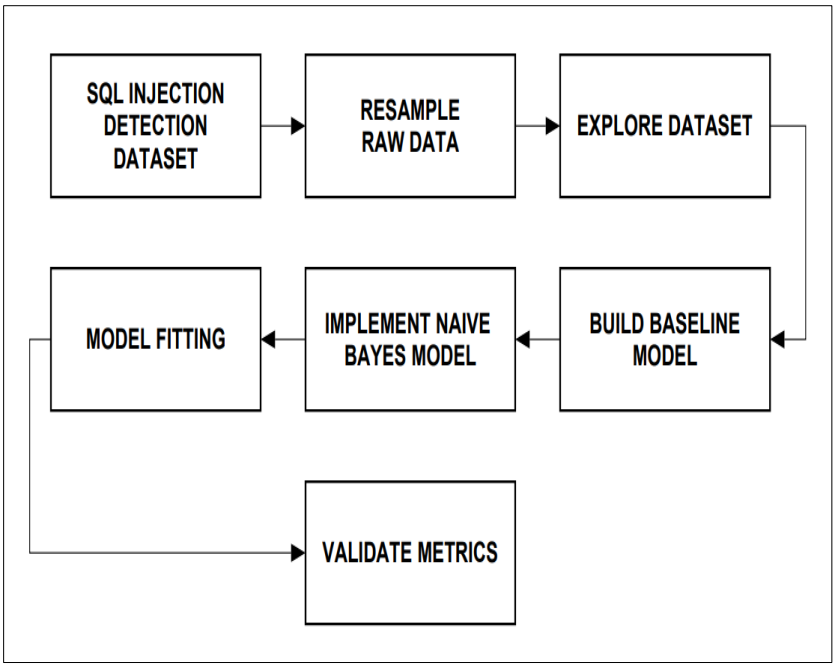

Figure 2 Proposed System

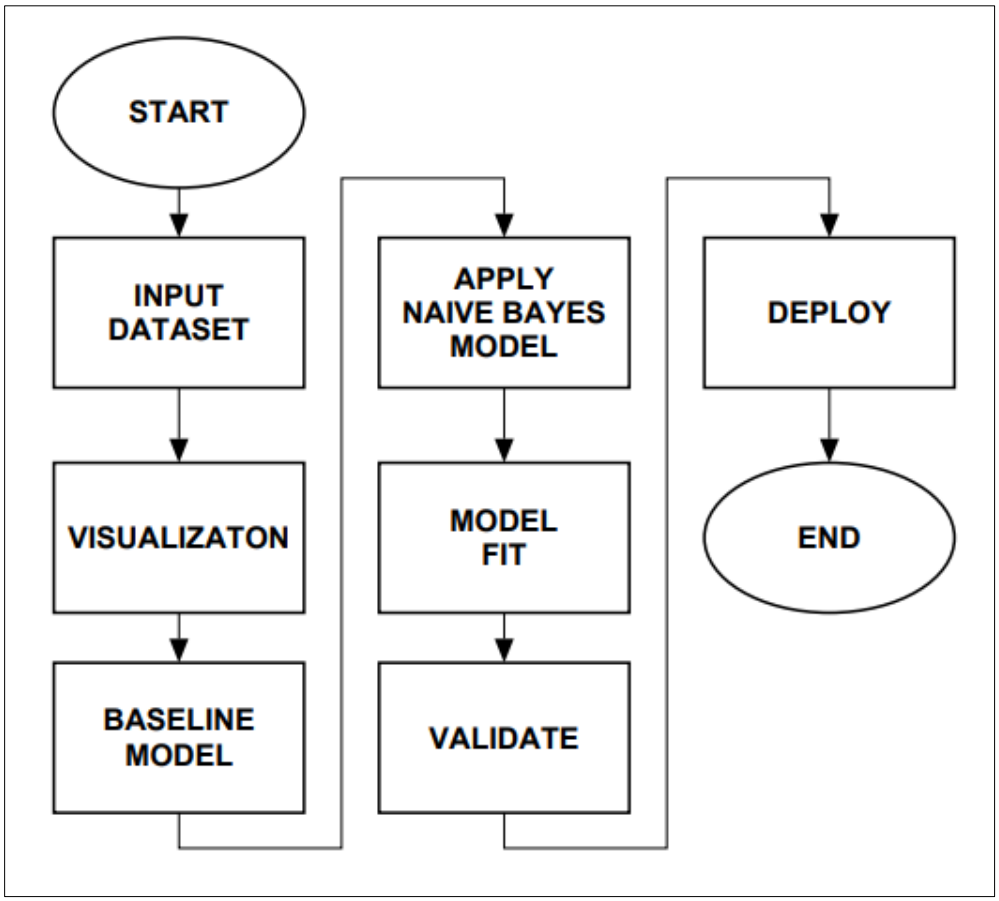

Figure 3 Flow Chart

The data flow diagram of Level 2 shows the whole process of the project, with the result output with the predicted model is shown in Fig 4.

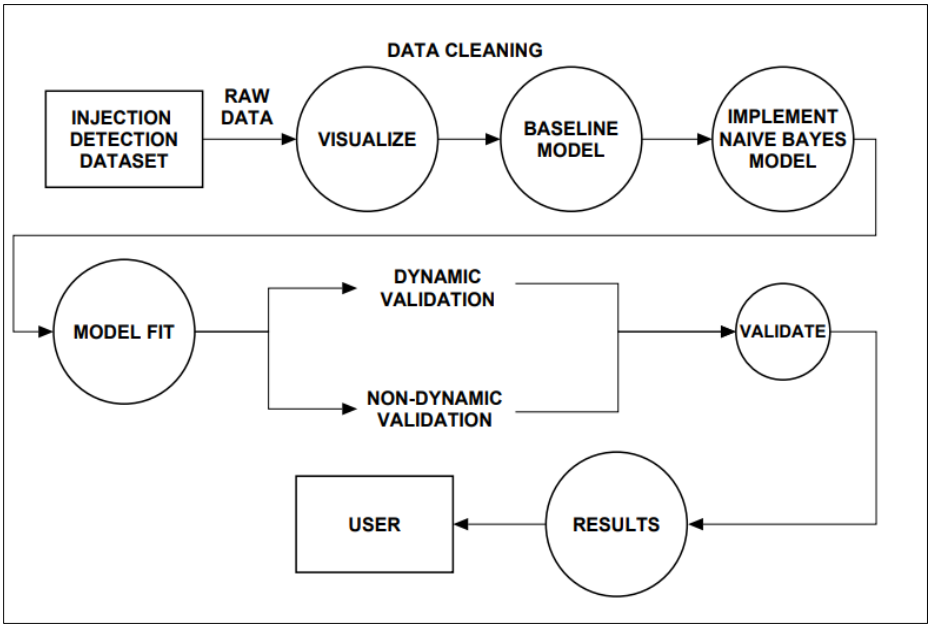

Figure 4 Dataflow Diagram of LEVEL 2

4. Results

The proposed approach of implementing a Gaussian Naïve Bayes model for SQL injection detection yielded promising results on the test dataset. The model demonstrated a notable accuracy of approximately 82.03%, indicating its proficiency in correctly classifying instances as safe or vulnerable SQL queries. This accuracy measure reflects the overall correctness of the model's predictions.Furthermore, the F1-score, which harmonizes precision and recall, was computed at approximately 80.53%. This score underscores the model's ability to strike a balance between accurately identifying positive cases (vulnerable queries) and avoiding false positives. The sensitivity metric, a crucial measure of the model's capability to accurately identify vulnerable cases, achieved an impressive value of around 98.63%. This indicates that the model is adept at correctly detecting SQL injection vulnerabilities, minimizing the risk of false negatives.In terms of specificity, the model showcased a value of about 72.00%, reflecting its proficiency in correctly identifying safe cases and minimizing the occurrence of false positives.The precision metric, which gauges the accuracy of positive predictions, was found to be approximately 68.05%. This measure affirms the model's competence in providing accurate positive classifications, thereby reducing the chances of incorrectly flagging legitimate SQL queries as vulnerable. The Permormance metrics of system is show in Table 1.

Table 1 Performance metrics of system

Implementation	**Naïve Bayes**
Accuracy	82.03
F1-score	80.53
Precision	68.04
Specificity	98.62

Furthermore, a bar chart as shown in Fig.5 shows the visualization of Performance metrics Evaluation.

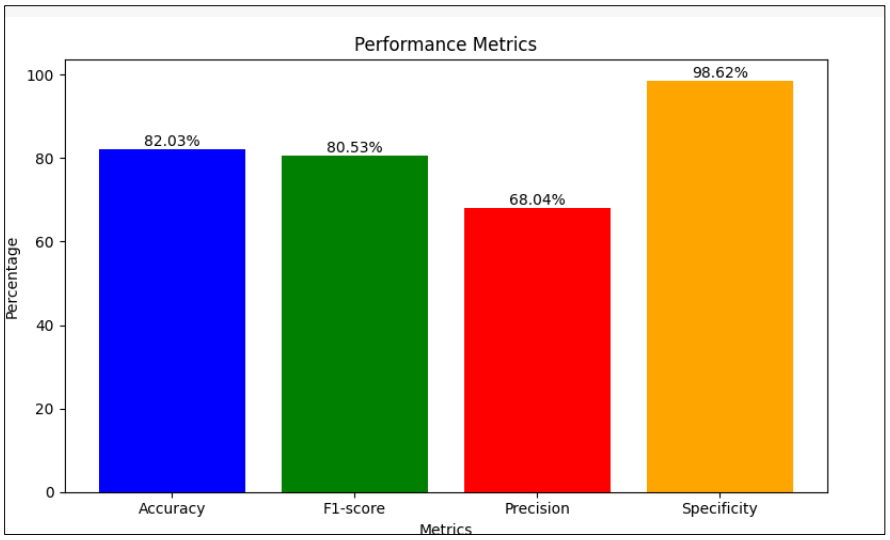

Figure 5 Visualization of Performance Evaluation

5. Conclusion and Way Forward

In this study, we addressed the critical concern of SQL injection vulnerabilities in web applications through the implementation of a Gaussian Naïve Bayes model. The effectiveness of the model was demonstrated in accurately detecting SQL injection vulnerabilities, showcasing its potential to enhance the security of web applications.Our research revealed that the Gaussian Naïve Bayes algorithm, when applied to the SQL Injection Dataset, exhibited commendable performance metrics. The achieved accuracy of approximately 82.03% and F1-score of around 80.53% underscored the model's proficiency in correctly classifying queries as safe or vulnerable, while considering the trade-off between precision and recall.The high sensitivity metric value of approximately 98.63% highlighted the model's robustness in identifying vulnerable SQL queries, minimizing the risk of overlooking potential threats. Meanwhile, a specificity value of approximately 72.00% indicated the model's capacity to accurately classify safe queries, thus mitigating false positives. The precision metric, at approximately 68.05%, accentuated the model's ability to provide accurate positive classifications, further contributing to its reliability in real-world applications. These results collectively underline the significance of leveraging machine learning techniques, particularly the Gaussian Naïve Bayes algorithm, in detecting and preventing SQL injection vulnerabilities. The model's prowess in identifying potential threats while maintaining a balance between accuracy and precision holds great promise for bolstering the security landscape of web applications.It is imperative to acknowledge that the success of this study lays the foundation for further research and refinement of the model. Future work may entail the exploration of more advanced machine learning approaches, the incorporation of additional features, and the evaluation of the model's performance in diverse real-world scenarios.

Future enhancement

As part of future enhancements, this project has the potential for further refinement in terms of usability and efficiency. One avenue for improvement involves integrating the machine learning-based SQL Injection detection approach with complementary mechanisms like static code analysis and web application firewalls. Additionally, the machine learning model itself can be elevated through more sophisticated feature extraction techniques. Alternatives such astokenization or other approaches can be explored to enhance feature extraction, ultimate leading to a more effective training process for the model.

Reference

[1] Liu, Muyang, Ke Li, and Tao Chen. "DeepSQLi: Deep semantic learning for testing SQL injection." Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2020.

[2] Jahanshahi, Rasoul, Adam Doupé, and Manuel Egele. "You shall not pass: Mitigating sql injection attacks on legacy web applications." Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. 2020.

[3] Gu, Haifeng, Jianning Zhang, Tian Liu, Ming Hu, Junlong Zhou, Tongquan Wei, and Mingsong Chen. "DIAVA: a traffic-based framework for detection of SQL injection attacks and vulnerability analysis of leaked data." IEEE Transactions on Reliability 69, no. 1 (2019): 188-202.

[4] Junjin, Mei. "An approach for SQL injection vulnerability detection." In 2009 Sixth international conference on information technology: new generations, pp. 1411-1414. IEEE, 2009.

[5] Yeole, A. S., and B. B. Meshram. "Analysis of different technique for detection of SQL injection." In Proceedings of the International Conference & Workshop on Emerging Trends in Technology, pp. 963-966. 2011.

[6] Shar, Lwin Khin, Hee Beng Kuan Tan, and Lionel C. Briand. "Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis." In 2013 35th International Conference on Software Engineering (ICSE), pp. 642-651. IEEE, 2013.

[7] Sharma, Chandershekhar, and S. C. Jain. "Analysis and classification of SQL injection vulnerabilities and attacks on web applications." In 2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014), pp. 1-6. IEEE, 2014.