Vulnerability Detection and Prevention: An Approach to Enhance Cybersecurity

---

BY

AMINA ZAHID

1. **ROLL No: ------------------------22015919-013 ---------------------------**
2. **REGISTRATION No: ----------17034298109------------------------------**
3. **DEGREE PROGRAM: -----MS COMPUTER SCIENCE ---------------**
4. **DEPARTMENT: ---------------- COMPUTER SCIENCE ----------------**
5. **FACULTY: ------------COMPUTING AND IT ----------------------------**
6. **CAMPUS NAME: --------HAFIZ HAYAT CAMPUS ------------------**
7. **SUPERVISOR NAME: ----------------DR. UMAR SHOAIB -------------**
8. **DEGREE ENROLLMENT SEMESTER: ---------FALL 2022-----------**
9. **FREEZED OR MISSED SEMESTER(S) (IF ANY): --------NONE------**
10. **DATE OF SYNOPSIS SUBMISSION TO THE DEPARTMENT: --------
---26/02/2024---**
11. **DATE OF APPROVAL FROM DDRC: ---------------------------------------**

Scholar Signature: --

Supervisor Signature: --

Chairperson/Head Signature: --

Convener Faculty Board Signature: ---

Date of Approval by ASRB: ---

Table of Contents

Contents		Page

ABSTRACT:

In websites, SQL Injection (SQLI) is considered to be among the most prevalent vulnerabilities. The logical portion of the database is basically compromised by this attack. Previous cybersecurity research has faced challenges due to a lack of appropriate appliances. Therefore, more development is required. A customized SQL Injection dataset has been used to assess a variety of algorithms for the purpose of identifying and detecting SQL injection attacks as well as prevention strategies.

1. INTRODUCTION:

Worldwide, there are 5.3 billion internet users as of October 2023, making up 65.7% of the world's population (Statista, 2023). The frequency and potency of web attacks have increased in step with the development and expansion of web applications. The study presented in (List of Data Breaches and Cyber Attacks in 2023, 2023) states that 114 security incidents in October 2023 were made public. Because of these incidents, 867,072,315 records were compromised, exceeding 5 billion records for the year. The Open Worldwide Application Security Project (OWASP Top Ten 2023, 2023) states that the injection vulnerability is still the most prevalent in online applications. In actuality, the majority of contemporary web applications either store user-supplied data in a back-end database or retrieve user-selected data from one. Web servers employ forms and cookies, which are the most widely used ways of communication with these users, to identify and track users as they visit between pages of a website as well as to identify users who return to the site. By inserting hazardous code into these user-provided fields, which is then used to create the SQL queries, hackers try to take advantage of this feature. A database could be erased or users' private information could be obtained from online applications if Structured Query Language Injection (SQLI) attack is successful due to inadequate user input validation.

SQLI attacks are most dangerous type of injection attacks because they compromise four primary security services: integrity, authorization, confidentiality, and authentication (Deepa et al, 2018). In order to access a database and alter its contents, a SQL injection attack typically involves inserting malicious SQL commands (Fang et al, 2018) into queries or input forms (e.g., sending the attacker the contents of the database, changing or deleting the content of the database, etc.) (Li et al, 2019).

Researchers have identified a number of counterstrategies to stop the SQLI attack. The most popular types of techniques are hybrid and dynamic, which combine elements of both static and dynamic (Minhas et al, 2013). The purpose of static analysis is to find mistakes and discrepancies in the generated SQL queries (Gupta et al, 2014). Dynamic analysis is another complex method that helps the system identify SQL injection in valid queries (Bockermann et al, 2009). Static and dynamic analysis benefits are combined in a hybrid approach. Firstly, detection models are constructed and trained using the static analysis. Then, it investigates these models using dynamic analysis to ascertain which is the optimal choice (Minhas et al, 2013).

Moreover, In order to protect data, prevent unauthorized access, maintain data integrity, avoid financial loss, preserve system availability, uphold customer trust, adhere to legal and regulatory requirements, minimize future costs, and promote a security-aware culture, it is imperative that SQL injection vulnerabilities be found and fixed. These weaknesses may result in loss of money, interruptions to services, and illegal access to private data. Consistently identifying and addressing vulnerabilities shows a dedication to security, improving an organization's standing and lowering the probability of new vulnerabilities. Machine learning serves as the basis for both hybrid and dynamic analytic techniques. As the SQLI attack is so sensitive, it has been the focus of numerous works. A portion of these initiatives are limited to finding SQLI instances. Despite the encouraging outcomes that ML and DL have shown in the field of cybersecurity, there are still certain real-world obstacles to their widespread use. The lack of real-world implications for ML and DL in cybersecurity hinders their widespread use. DL and ML algorithms, especially for cybersecurity applications, need to be further developed and optimized. This will require more research and collaboration between academic institutions and the business sector. A heuristic approach to SQLI attack detection is presented in this work. Furthermore, on our own SQLI dataset, we are going to employ DL and ML approaches.

The rest of research is organized in this manner. A review of the literature is discussed in Section II. The suggested framework is illustrated in Section III, which covers every step from building the dataset to selecting the SQL statement features. Section IV provides an explanation of the selection procedure for the classification algorithms. The same section covers the training and testing of ML and DL classifiers. Section V presents the analysis and results. Section VI contains recommendations for further research as well as conclusions.

2. LITERATURE REVIEW:

Researchers in the field of network security have been examining how to detect and prevent SQL injection vulnerabilities employing DL and ML. Numerous research works have provided different methods for identifying and mitigating SQL injection attacks. These techniques enhance detection accuracy and lower false alarms by utilizing DL frameworks, ML algorithms, and natural language processing models. Ensemble machine learning algorithms such as Gradient Boosting Machine (GBM), Adaptive Boosting (AdaBoost), Extended Gradient Boosting Machine (XGBM), and Light Gradient Boosting Machine (LGBM) require additional processing and timing when it comes to SQLI attack detection (Minhas et al, 2013). To improve detection efficiency, research has also been done on semantic feature extraction from SQL statements.

A CNN-based SQL injection detection model is presented in (Luo et al, 2019), which shows good accuracy, precision, and recall rate while outperforming conventional approaches in terms of detection efficacy and resilience against attack obfuscation. However, the performance and generalizability in real-world scenarios may be impacted by (Luo et al, 2019) incomplete discussion of CNN's limitations, dataset, comparisons, computational requirements, and challenges with CNN-based SQLI detection. Subsequent investigations ought to concentrate on refining the deep learning model, incorporating additional intrusion detection features, and testing additional deep learning approaches. All things considered, the CNN-based model works and is practical.

To monitor the development and security implications of vulnerabilities in software products, (Williams et al, 2018) presents a data mining framework that makes use of diffusion-based storytelling and the Supervised Topical Evolution Model. The lack of effort in analyzing cybersecurity corpora to examine the evolution of vulnerabilities is acknowledged in (Williams et al, 2018). It recommends more investigation to evaluate the integrated data mining framework's scalability and generalizability. The STEM model and diffusion-based storytelling approach may or may not be beneficial. Perhaps not covering other topics, this paper concentrates on vulnerabilities in software products. This study presents a data mining framework for analyzing software product vulnerabilities that uses diffusion-based storytelling and the Supervised Topical Evolution Model to provide accurate patterns and faster convergence.

Despite technological advancements, the authors (Roy et al, 2022) encourage the use of machine learning techniques, such as unsupervised learning and K-means clustering algorithm, on publicly available SQLI dataset to enhance SQLI vulnerability identification and prevention. Although real-world scenarios, the Kaggle SQLI dataset, comparisons, computational requirements, and potential vulnerabilities in ML classifiers are absent from this paper, it does present a runtime technique for mitigating SQLI attacks. The study focuses on detecting SQLI attacks in web-based systems using ML classifiers, including Logistic Regression, AdaBoost, Random Forest, Naive Bayes and XG-Boost. Naive Bayes is found to be the most effective method, with high accuracy.

The study (Liu et al, 2020) repeats experiments 20 times for six Java-based System Under Tests with the aim to evaluate the effectiveness of SQLI testing using metrics and evaluation techniques. This reduces randomness and increases computational resources. One tool for testing SQL injection vulnerabilities is called DeepSQLi. To create test cases, deep learning is employed. DeepSQLi performs better at detecting SQLI vulnerabilities than SQLmap. DeepSQLi operates more quickly and needs fewer test cases.

An adaptive deep forest-based technique for SQLI detection is presented in (Li et al, 2019). The feature weights are altered via the AdaBoost algorithm. When compared to conventional ML techniques, the suggested adaptive deep forest-based method for SQLI detection has shown to be more efficient. But as the count of layers rises, Deep forests diminish their original characteristics, and DL techniques perform better than the suggested approach.

The study (Zhang et al, 2020) examines the fundamentals of SQL injection attacks, suggests a detection tool, and discusses how to detect them. The study looks into SQL injection vulnerabilities and how to find them. It also suggests and tests a tool called SQLiscan that can be used to find these vulnerabilities. Static detection techniques are used to analyze vulnerabilities and find SQL injections with the SQLiscan tool.

The suggested technique (Parashar et al, 2021) uses text summarization and ML for classification to find SQL injection vulnerabilities in text. A Random Forest model has been applied to web text and tweets in order to test the suggested technique for detecting SQL injection vulnerabilities. Errors in text summarization and the elimination of words that are obvious in order to avoid false positives are limitations.

Four filtration techniques are described in (Balasundaram et al, 2012) as a means of detecting and preventing input validation vulnerabilities, such as SQL injection. Constraint Validation and Malicious Text Detector are two of the techniques employed. Among the drawbacks are the requirement for developers to initialize trusted strings and the possibility of second-order attacks stemming from the persistent storage of these strings. By using both static and dynamic analysis, the method successfully ensures that the prototype tool prevents all attacks without producing false positives.

Web applications are at serious risk from SQL injection, although there are a number of ways to stop these attacks (Mavromoustakos et al, 2016). Black-box testing with a web crawler and Parse Tree Validation Approach are two of the techniques employed. The limitations include no error checking for queries, control character and SQL keyword limitations.

The research (Luo et al, 2019) presents a ML algorithm that can identify online SQLI attacks database applications with a high accuracy rate. By employing a feature identifier code and a graphical user interface (GUI), this paper presents a ML-based algorithm that achieves the highest accuracy in preventing SQL injection attacks. A suggested technique for identifying SQLI attacks was not thoroughly tested or documented in this paper, nor were its limits, scalability, or potential false positives addressed. The study suggests a heuristic method for identifying SQLI attacks that combines ML techniques utilizing both static and dynamic analysis.

The research (Li et al, 2021) finds fifteen vulnerabilities using CNNs, DBNs, bidirectional RNNs, and shallow learning models. Seven of these were missed and reported to suppliers; the remaining eight were covertly fixed. The methodology has proven to be beneficial in identifying vulnerabilities that have not been reported to the National Vulnerability Database. Convolutional neural networks (CNNs) are used in (Hwang et al, 2022) to identify vulnerabilities using deep learning techniques. When compared to state-of-the-art techniques, its suggested CodeNet-based vulnerability detection solution exhibits good detection performance and detection speed.

The main objectives of the article (Odeh et al, 2023) are to identify and prevent backend technology fingerprinting, SQLI, remote execution of code, and cross-site-scripting (CSS) threats. In (Odeh et al, 2023), the proposed approach effectively detects and prevents vulnerabilities in web applications and also highlights the importance of advanced methods for diagnosis and prevention. The main objectives of (Zhang et al, 2023) is to decompose a syntax-based Control

Flow Graph (CFG) into many implementation scenarios. Path representations are learned using properly trained computer models and CNNs. When it comes to F1-score, precision, and recall, the SQLI detection method described in (Marjanov et al, 2022) performs better than the majority of baselines. The primary goal of the (Marjanov et al, 2022) study is to identify source code vulnerabilities using machine learning algorithms that preserve code organization. According to the study (Marjanov et al, 2022), there are several hurdles to overcome before it becomes possible to identify vulnerabilities in source code, such as the requirement for pipelines for code organization, a consensus benchmark, and improved coverage of error kinds and languages.

Methods Used	**Results**	**Limitations**	**References**
To find temporal themes, used diffusion-based storytelling and the supervised topical evolution model.	The framework addresses scalability, reveals new vulnerability correlations, and presents STEM-P, a parallel version that is faster than the standard STEM model.	It acknowledges the lack of effort in analyzing cybersecurity corpora to study vulnerability evolution. It suggests further research to assess the scalability and generalizability of the integrated data mining framework. The STEM model's and the diffusion-based storytelling technique's efficacy might differ. It focuses on software product vulnerabilities and may not cover other areas.	Williams, M. A., Dey, S., Barranco, R. C., Naim, S. M., Hossain, M. S., & Akbar, M. (2018, December). Analyzing evolving trends of vulnerabilities in national vulnerability database. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 3011-3020). IEEE.
Shallow learning models, CNNs, DBNs, and bidirectional RNNs are employed.	Fifteen vulnerabilities were found, seven were unidentified and reported to suppliers, and eight were patched covertly. The methodology has proven to be helpful in identifying vulnerabilities that have not been disclosed to the National Vulnerability Database.	Pay close attention to finding security holes in C/C++ software source code. Requirement for more comprehensive vulnerability syntax properties	Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). Sysevr: A framework for using deep learning to detect software vulnerabilities. IEEE Transactions on Dependable and Secure Computing, 19(4), 2244-2258.
Methods for identifying source code vulnerabilities	The study highlights the challenges in detecting	The consensus on evaluating machine learning techniques for	Marjanov, T., Pashchenko, I., & Massacci, F. (2022). Machine

using machine learning use pipelines that preserve the organization of the code.	vulnerabilities in source code, including the need for a consensus benchmark, better coverage of error types and languages, and pipelines for code organization.	vulnerability identification is lacking, and discussions on error types and programming languages are restricted.	Learning for Source Code Vulnerability Detection: What Works and What Isn't There Yet. IEEE Security & Privacy, 20(5), 60-76.
Splitting up a syntax-based Control Flow Graph (CFG) into several possible ways to execute it Convolutional neural networks and pre-trained code models are used to learn path representations.	The suggested method performs better in terms of F1-Score, Precision, and Recall than the most recent baselines.	A challenge in separating information about vulnerabilities from other information. limits on input length cause inefficiency when handling large codes.	Zhang, J., Liu, Z., Hu, X., Xia, X., & Li, S. (2023). Vulnerability Detection by Learning from Syntax-Based Execution Paths of Code. IEEE Transactions on Software Engineering.
The system's main objectives are to identify and stop SQLI, remote code execution, cross-site-scripting assaults, and backend technology fingerprinting.	Web application vulnerabilities are efficiently found and prevented by the suggested system. The study emphasizes the value of sophisticated techniques for prevention and detection.	To improve detection and prevention capabilities, the suggested vulnerability system needs more investigation, machine learning integration, extensive testing, and comparison analysis.	Odeh, N., & Hijazi, S. (2023). Detecting and Preventing Common Web Application Vulnerabilities: A Comprehensive Approach. International Journal of Information Technology and Computer Science
Deep learning approaches employing CNN CodeNet-based vulnerability detection method.	When compared to cutting-edge techniques, the suggested CodeNet-based vulnerability detection method exhibits good detection performance and detection time. The outcomes of the experiments conducted under different kinds of vulnerabilities are shown.	In picture analysis, smart contract semantics and context are disregarded. Ignorance of context and semantics can lead to false detection alarms.	Hwang, S. J., Choi, S. H., Shin, J., & Choi, Y. H. (2022). CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. IEEE Access, 10, 32595-32607.
Deep natural language processing, neural language models, and word sequence prediction were used in this study.	In detecting SQLI vulnerabilities, DeepSQLI performs better than SQLmap.	The study evaluates SQLI testing's effectiveness using metrics and evaluation methods, repeating experiments 20 times for six Java-based System Under Tests to mitigate randomness	Liu, M., Li, K., & Chen, T. (2020, July). DeepSQLi: Deep semantic learning for testing SQL injection. In Proceedings of the 29th ACM SIGSOFT International

		and increase computational resources.	Symposium on Software Testing and Analysis (pp. 286-297).
It presents a Convolutional Neural Network (CNN) based SQL injection detection model that efficiently handles SQL injection attacks by extracting attack payloads from network flow.	CNN accomplished a remarkable accuracy rate.	Performance and generalizability of the CNN-based SQLI detection model in the study may be impacted by improper usage of CNN, datasets, comparisons, computing needs, and challenges.	Luo, A., Huang, W., & Fan, W. (2019, June). A CNN-based Approach to the Detection of SQL Injection Attacks. In 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS) (pp. 320-324). IEEE.
AdaBoost algorithm has been incorporated into a deep forest model to increase its adaptability for SQL injection detection.	The suggested approach outperforms both DL and traditional ML techniques in its ability to detect sophisticated SQLI attacks.	The initial traits of deep forests deteriorate as the count of layers rises, and deep learning methods outperform the recommended strategy.	Li, Q., Li, W., Wang, J., & Cheng, M. (2019). A SQL injection detection method based on adaptive deep forest. IEEE Access, 7, 145385-145394.
It deals with ML classifiers to identify SQLI attacks in web-based systems, such as Random Forest, AdaBoost, and Logistic Regression.	The ideal strategy is determined to be Naive Bayes, which has the highest accuracy.	It offers a runtime method for preventing SQL injection attacks, but it is devoid of comparisons, real-world examples, the Kaggle SQLI dataset, computational requirements, and possible weaknesses in ML classifiers.	Roy, P., Kumar, R., & Rani, P. (2022, May). SQL injection attack detection by machine learning classifier. In 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC) (pp. 394-400). IEEE.

TABLE 1: An Overview on Literature Review

This is quick recap of prior research on ML and DL techniques to SQLI vulnerability identification. Apart from the author-created dataset, this study will employ ML and DL techniques.

3. OBJECTIVES OF RESEARCH:

The following are the main objectives of the study in question:

• Assessment of the efficacy of various algorithms: this involves analyzing the outcomes of various algorithms in order to determine the best architecture for vulnerability detection.

• The overall goals of the research conducted in this study are to increase reliability while also improving the usability, accuracy, and reliability of models in vulnerability detection.

4. RESEARCH QUESTIONS:

- Which algorithm performs best for detecting vulnerability?
- What kind of vulnerability can be discovered?
- How to prevent from vulnerability?
- Which performance metrics are considered in this research?

5. PROPOSED WORK & METHODOLOGY:

A novel heuristic approach for detecting SQLI attacks is presented in this research. Using our own SQLI dataset, we apply appropriate algorithms. We intend to carry out our research in phases. Additionally, here is workflow diagram.

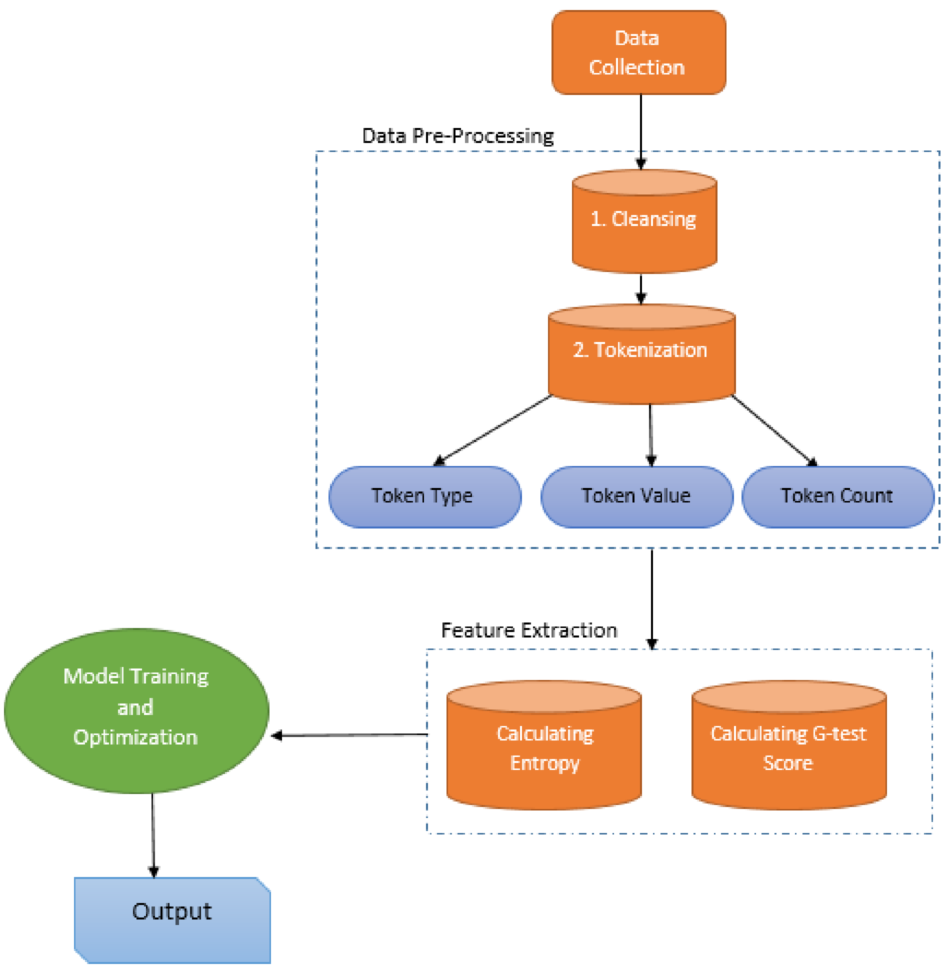

Figure 1: Systematic Workflow for Vulnerability Detection

Dataset Creation: In this phase, we are going to be collecting SQLIs to get our dataset ready for the experiments.

Dataset Pre-Processing: To preprocess our dataset, we are going to employ suitable steps (such as normalization and cleansing etc.) as shown in figure.

Feature Extraction: We're going to use the proper feature extraction methods (such g-test score and entropy calculation etc.).

Model Training and Optimization: In this stage, we will train our model in order to get the desired outcome. We will also adjust the parameters in order to increase efficacy.

Output: Identifying vulnerabilities, detecting them, and taking preventative measures are the goals of this stage.

6. EXPECTED OUTCOMES:

Lowering false positives and negatives, the algorithms can help identify possible threats early on. They can also prioritize vulnerabilities according to severity, automate detection procedures, and adjust to new threats. As a result, security teams have less work to do and can scale more easily. Additionally, continuous learning and predictive analytics are improved. By integrating with current security systems, ML and DL solutions can offer a defensive approach. Cost savings can result from automation and increased accuracy because they minimize the need for manual vulnerability assessments and incident response. They can also guarantee regulatory compliance and make a significant contribution to a thorough risk assessment. An organization's cybersecurity posture can be greatly improved by utilizing deep learning and machine learning for vulnerability detection.

7. REFERENCES:

Number of internet and social media users worldwide as of January 2024. Retrieved from Digital Population Worldwide 2023: https://www.statista.com/statistics/617136/digital-population-worldwide/

List of Data Breaches and Cyber Attacks in 2023 – 8,214,886,660 records breached. (2023). Retrieved from IT Governance: https://www.itgovernance.co.uk/blog/list-of-data-breaches-and-cyber-attacks-in-2023.

OWASP Top Ten 2023 – The Complete Guide. (2023, December 23). Retrieved from Reflectiz: https://www.reflectiz.com/blog/owasp-top-ten-2023/

Deepa, G., Thilagam, P. S., Khan, F. A., Praseed, A., Pais, A. R., & Palsetia, N. (2018). Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications. International Journal of Information Security, 17, 105-120.

Fang, Y., Peng, J., Liu, L., & Huang, C. (2018, March). WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM. In Proceedings of the 2nd international conference on cryptography, security and privacy (pp. 170-174).

Li, Q., Wang, F., Wang, J., & Li, W. (2019). LSTM-based SQL injection detection method for intelligent transportation system. IEEE Transactions on Vehicular Technology, 68(5), 4182-4191.

Gupta, M. K., Govil, M. C., & Singh, G. (2014, May). Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey. In International conference on recent advances and innovations in engineering (ICRAIE-2014) (pp. 1-5). IEEE.

Bockermann, C., Apel, M., & Meier, M. (2009, July). Learning sql for database intrusion detection using context-sensitive modelling. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 196-205). Berlin, Heidelberg: Springer Berlin Heidelberg.

Minhas, J., & Kumar, R. (2013). Blocking of sql injection attacks by comparing static and dynamic queries. International Journal of computer network and Information Security, 5(2), 1.

Luo, A., Huang, W., & Fan, W. (2019, June). A CNN-based Approach to the Detection of SQL Injection Attacks. In 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS) (pp. 320-324). IEEE.

Williams, M. A., Dey, S., Barranco, R. C., Naim, S. M., Hossain, M. S., & Akbar, M. (2018, December). Analyzing evolving trends of vulnerabilities in national vulnerability database. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 3011-3020). IEEE.

Roy, P., Kumar, R., & Rani, P. (2022, May). SQL injection attack detection by machine learning classifier. In 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC) (pp. 394-400). IEEE.

Liu, M., Li, K., & Chen, T. (2020, July). DeepSQLi: Deep semantic learning for testing SQL injection. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 286-297).

Li, Q., Li, W., Wang, J., & Cheng, M. (2019). A SQL injection detection method based on adaptive deep forest. IEEE Access, 7, 145385-145394.

Zhang, T., & Guo, X. (2020, September). Research on SQL injection vulnerabilities and its detection methods. In 2020 4th Annual International Conference on Data Science and Business Analytics (ICDSBA) (pp. 251-254). IEEE.

Parashar, D., Sanagavarapu, L. M., & Reddy, Y. R. (2021, February). Sql injection vulnerability identification from text. In 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (pp. 1-5).

Balasundaram, I., & Ramaraj, E. (2012). An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching. Procedia Engineering, 30, 183-190.

Mavromoustakos, S., Patel, A., Chaudhary, K., Chokshi, P., & Patel, S. (2016, December). Causes and prevention of SQL injection attacks in web applications. In Proceedings of the 4th International Conference on Information and Network Security (pp. 55-59).

Luo, A., Huang, W., & Fan, W. (2019, June). A CNN-based Approach to the Detection of SQL Injection Attacks. In 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS) (pp. 320-324). IEEE.

Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). Sysevr: A framework for using deep learning to detect software vulnerabilities. IEEE Transactions on Dependable and Secure Computing, 19(4), 2244-2258.

Hwang, S. J., Choi, S. H., Shin, J., & Choi, Y. H. (2022). CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. IEEE Access, 10, 32595-32607.

Odeh, N., & Hijazi, S. (2023) Detecting and Preventing Common Web Application Vulnerabilities: A Comprehensive Approach. International Journal of Information Technology and Computer Science.

Zhang, J., Liu, Z., Hu, X., Xia, X., & Li, S. (2023). Vulnerability Detection by Learning from Syntax-Based Execution Paths of Code. IEEE Transactions on Software Engineering.

Marjanov, T., Pashchenko, I., & Massacci, F. (2022). Machine Learning for Source Code Vulnerability Detection: What Works and What Isn't There Yet. IEEE Security & Privacy, 20(5), 60-76.