Scientific Publications

Journal of Artificial Intelligence and Big Data, 2024, 4, 1333
www.scipublications.com/journal/index.php/jaibd
DOI: 10.31586/jaibd.2024.1333

Article

Intelligent Detection of Injection Attacks via SQL Based on Supervised Machine Learning Models for Enhancing Web Security

Rahul Vadisetty [1,*], Purna Chandra Rao Chinta [2], Chethan Sriharsha Moore [2], Laxmana Murthy Karaka [3], Manikanth Sakuru [4], Varun Bodepudi [5], Srinivasa Rao Maka [6], Srikanth Reddy Vangala [7]

[1] Wayne State University, Master of Science, USA
[2] Microsoft, Support Escalation Engineer, USA
[3] Code Ace Solutions Inc, Software Engineer, USA
[4] JP Morgan Chase, Lead Software Engineer, USA
[5] Deloitte Consulting LLP, Senior Solution Specialist, USA
[6] North Star Group Inc, Software Engineer, USA
[7] Department of Computer Science, University of Bridgeport, USA

*Correspondence: Rahul Vadisetty (rahulvy91@gmail.com)

Abstract: The most prevalent technique behind security data breaches exists through SQL Injection Attacks. Organizations and individuals suffer from sensitive information exposure and unauthorized entry when attackers take advantage of SQL injection (SQLi) attack vulnerability's severe risks. Static and heuristic defense methods remain conventional detection tools for previous SQL injection attacks study's foundation is a detection system developed using the Gated Recurrent Unit (GRU) network, which attempts to efficiently identify SQL Injection attacks (SQLIAs). The suggested Gated Recurrent Unit model was trained using an 80:20 train-test split, and the results showed that SQL injection attacks could be accurately identified with a precision rate of 97%, an accuracy rate of 96.65%, a recall rate of 92.5%, and an F1-score of 94%. The experimental results, together with their corresponding confusion matrix analysis and learning curves, demonstrate resilience and outstanding generalization ability. The GRU model outperforms conventional machine learning (ML) models, including K-Nearest Neighbor's (KNN), and Support Vector Machine (SVM), in terms of identifying sequential patterns in SQL query data. Recurrent neural architecture proves effective in the detection of SQLi attacks through its ability to provide secure protection for contemporary web applications.

Keywords: Web Application Security, Cyberattacks, SQL Injection Attacks (SQLIA), Machine Learning (ML), SQL Injection Dataset

How to cite this paper:
Vadisetty, R., Rao Chinta, P. C., Moore, C. S., Karaka, L. M., Sakuru, M., Bodepudi, V., Maka, S. R., & Vangala, S. R. (2024). Intelligent Detection of Injection Attacks via SQL Based on Supervised Machine Learning Models for Enhancing Web Security. Journal of Artificial Intelligence and Big Data, 4(2), 109-119.
DOI: 10.31586/jaibd.2024.1333

Received: August 27, 2024
Revised: October 23, 2024
Accepted: November 26, 2024
Published: December 19, 2024

1. Introduction

The modern world relies heavily on web applications since they serve as vital tools for executing business operations and delivering government services, and conducting personal activities. Online banking, along with e-commerce and healthcare portals and educational platforms, have become fundamental structures that enable multiple business sectors to work efficiently and manage data and deliver services. Organizations now depend extensively on web-based technologies, prompting a critical need for system protection due to its vital importance [1]. The expanding nature of web applications leads to increased attacks that exploit weaknesses to interrupt services while simultaneously stealing sensitive information and damaging organizational credibility [2].

The SQLIAs stand as one of the most dangerous and widely used cyber threats among other forms [3]. SQLIAs function through invalid field input to embed damaging SQL code that lets attackers avoid security measures and obtain sensitive data while also facilitating database changes and removals. Security professionals face SQLi attacks as a major ongoing threat since web application vulnerabilities lead to them in over 65% of cases [4].

Web security enhancement alongside risk mitigation depends heavily on successful research into SQLi attack detection. The detection capability of traditional input validation combined with signature-based systems fails to detect modern or advanced variations of SQLIAs [5]. Web application security enhancement demands a proper detection system for SQLIAs. Most modern injection techniques remain undetected by traditional protection measures, which combine input validation and static rule enforcement.

The detection accuracy, along with false positive reduction, has motivated researchers to adopt ML and DL models due to their adaptive and intelligent features [6]. The combination of supervised machine and DL models can become a successful method for efficient SQLi attack detection in web security systems [7]. The system uses learned models to extract knowledge from past SQL query data, which includes harmless as well as harmful requests, enabling automatic detection of potential threats during real-time processing.

1.1. Motivation and Contribution of Paper

The security threat from SQLi bypasses conventional detection techniques while maintaining its importance as a critical web application vulnerability. Current rule-based detection systems experience difficulties when trying to identify complicated and emerging SQLi attack forms. Detecting of diverse SQLi attacks in real-time requires new technology because existing systems lack smart enough detection capabilities. The research introduces a complete structure for SQLi attack detection through DL methodology. The key contributions are as follows:

- Utilized a Kaggle SQLi dataset with over 30,000 SQL queries, including ~6,000 malicious entries across multiple SQLi types.
- Applied comprehensive preprocessing, duplicate removal, missing value handling, tokenization, encoding, and normalization.
- Performed feature selection using G-test scoring to enhance classification accuracy.
- Developed a GRU-based DL model for sequential analysis of SQL queries.
- Utilizing the disorientation matrix's accuracy, precision, f1-score, and recall, evaluate the model's efficiency.

1.2. Structure of paper

The remainder of the document is structured. Section II provides a background study on the Detection of SQLi Attacks. In Section III, the methodology that proposed in detailed. In Section V, the outcomes, analysis, and discussion are compared. Section V presents the study's conclusion and plans for further research.

2. Literature Review

A few significant studies that are connected to the Detection of SQLIAs for Enhancing Web Security are examined and filtered in order to carry out the present task. Table I provides a concise comparison of key research works on SQLi detection using ML, highlighting proposed methods, datasets used, key findings, and identified challenges.

Hasan, Balbahaith and Tarique (2019) developed a heuristic technique built around ML to stop SQLi attacks. It trains and evaluates 23 distinct ML classifications using a

dataset of 616 SQL expressions. It builds a top five classifiers served as the foundation for the Graphical User Interface (GUI) application amongst the ones mentioned above, chosen for their identification accuracy. The results of testing the suggested technique demonstrate that it can recognize SQLIAs with a from head to foot grade of accuracy (93.8%) [8].

Noor et al. (2019) Based on documented attack patterns, a new ML-based approach is suggested to identify cyber risks. To create threats and TTPs obtained from reliable sources in a semantic network of malware with related detection methods. The system is trained on a dataset of TTP taxonomies and threat artefacts. Its performance is evaluated using threat notifications. Despite the implementation of fictitious and missing TTPs, the system remains in place and effectively detects attacks with a 92% accuracy rate and few FP. On average, an internet connection trained with 133 TTPs from 45 security domains detects data breaches in 0.15 seconds [9].

Zhang (2019) introduces an ML classifier intended to find PHP code that has SQLi vulnerabilities. Using validation of input and sanitization attributes taken from source code files, algorithms for classifiers were trained and evaluated using both traditional and DL-based ML methods. A CNN-trained model had the best accuracy (95.4%) on ten-fold cross-verification, while the Multilayered Perceptron (MLP)-based model had the greatest level of recall (63.7%) and the strongest f parameter (0.746) [10].

Ul Islam et al (2019) Create a supervised learning method to identify NoSQL injections. It uses a variety of supervised learning methods and designs key characteristics by hand. Their tool's cross-validation of 10 times yielded an F2-score of 0.93. Additionally, they tested their tool against NoSQL Map, a NoSQLi generating tool, and discovered that its detection percentage is 36.25% higher than that of Screen, which is the sole NoSQL intrusion identification tool currently on the market [11].

McWhirter et al. (2018) offer a unique way to categorize queries made with SQL based just on the original query string's properties. Several test datasets taken from the Flashback testbed datasets are used to assess the suggested solution. For Select type operations, the demonstrating the applications accuracy was 97.07%, and for Insert type queries, it was 92.48%. This low success rate results from the characteristic process for extraction being confused by unclean quote marks in valid inputs [12].

Chattopadhyay et al. (2018) The document investigates obstacles when implementing machine learning methods for attack identification systems. This paper investigates various built-in difficulties related to machine learning techniques for intrusion detection definition and execution methods. Their research evaluates multiple machine learning techniques against different datasets in order to select the optimal solution for changing usage patterns by utilizing diverse performance measurement metrics. The amount of data processed by servers has grown dramatically so their security becomes a top priority [13].

Table 1 summarizes various ML-based approaches for SQL injection detection, highlighting methods, datasets, performance metrics, and limitations. While most achieve high accuracy, common challenges include small datasets, limited generalizability, scalability issues, and technology-specific constraints.

Table 1. Summary of the study on Detection of SQL Injection Attacks using machine learning

Author	Proposed Work	Dataset	Key Findings	Challenges/Gaps
Hasan, Balbahaith, and Tarique (2019)	Developed a heuristic ML-based algorithm and GUI app using the top 5 of 23 classifiers	616 SQL statements	Achieved 93.8% accuracy in detecting SQLi attacks	Small dataset size; scalability to real-world scenarios not validated
Noor et al. (2019)	suggested an arrangement based on semantic ML to connect risks and TTPs via probabilistic networks	TTP taxonomy dataset (133 TTPs, 45 threat families)	Detected threats with 92% accuracy; low false positives; 0.15s average detection time	Specific to TTP-based threats; generalization to SQLi-specific detection not tested
Zhang (2019)	Designed ML classifiers (CNN, MLP) to detect SQLi vulnerabilities in PHP code using code-level features	PHP source code files	CNN achieved 95.4% precision; MLP achieved 63.7% recall and F-measure of 0.746	Limited to PHP; varying performance across different classifiers
Ul Islam et al. (2019)	Created a NoSQL injection supervised learning tool. detection with a novel dataset	Custom-designed NoSQL injection dataset	Achieved 0.93 F2-score; outperformed Sqreen by 36.25%; database-agnostic	Limited availability of NoSQL datasets; manual feature engineering required
McWhirter et al. (2018)	Gap-Weighted String Subsequence was used. Kernel + SVM on SQL query strings for classification	Amnesia testbed datasets	Achieved 97.07% (Select) and 92.48% (Insert) accuracy; adapted to unseen threats	Lower accuracy with unsanitized quotation marks; sensitive to input anomalies
Chattopadhyay et al. (2018)	examined the difficulties in implementing ML methods for identifying malware	Multiple datasets (unspecified)	Compared various ML techniques across datasets; summarized performance based on different metrics; identified optimal techniques for evolving patterns.	Lack of clarity in dataset specifics; issues in defining and generalizing ML approaches to dynamic, real-world intrusion patterns; scalability concerns.

3. Research Methodology

The research design SQLi attack detection requires a methodical methodology that includes data collection, the preprocessing phase and model development, and performance evaluation that is illustrated in Figure 1. Initially, the SQLi dataset sourced from Kaggle contains over 30,000 SQL queries. The dataset is rigorously pre-processed, which includes dealing with unavailable values, removing duplicates, tokenization of queries into logical components, numerical encoding, and normalization. Feature extraction is performed using G-test scoring to identify the most discriminative features relevant for classification. After that, a performance evaluation is carried out by training a GRU model on 80% of the data while testing is done on the remaining 20%. The model's effectiveness is measured using precision, accuracy recall, and F1-score, supported by visualizations like confusion matrix, accuracy and loss curves, highlighting the GRU model's superior ability to detect SQLi attacks compared to traditional ML models.

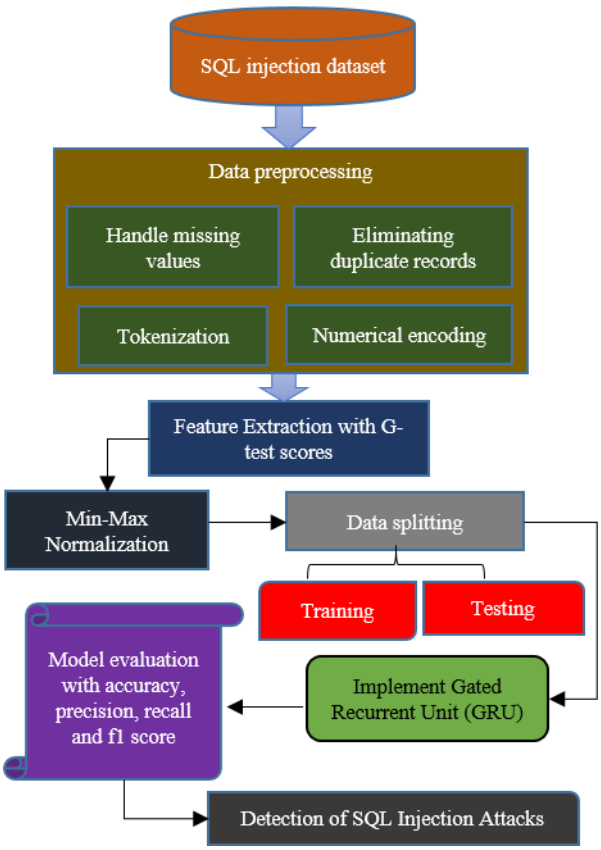

Figure 1. Proposed flowchart for Detection of SQL Injection Attacks

Each step of a proposed flowchart for Detection of SQLi Attacks for Enhancing Web Security is provided below:

3.1. Data Collection

In this research, it uses the SQLi dataset from Kaggle. Most of the 30,919 query statements from various websites are "SELECT FROM" and similar SQL variations. The three forms of SQLi Union Based, Blunder Based, and Blind SQL Injections amount to around 6,000 in this dataset the correlation of data is shown in Figure 2.

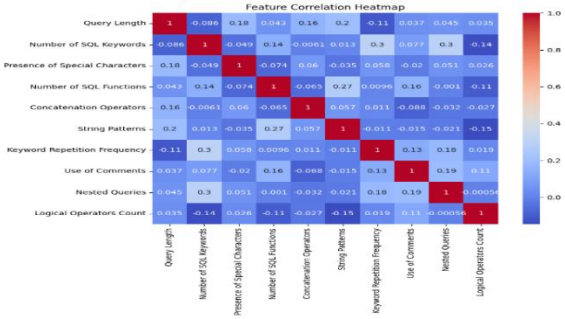

Figure 2. Feature Correlation Matrix

Figure 2 illustrates the correlations between various SQL query features using a color gradient from blue (negative) to red (positive). It highlights relationships among attributes like query length, SQL keywords, special characters, and logical operators. Diagonal values show perfect self-correlation, while off-diagonal elements reveal feature

interdependencies. This visualization aids in understanding which characteristics commonly co-occur, offering insights for query analysis and security detection.

3.2. Data Preprocessing

During the data pre-processing step, SQL queries need sequential and numerical inputs to be properly prepared. The data preparation includes multiple steps, which start with tokenization and end with normalization, that ensure effective data structure. This procedure puts emphasis on dependent execution sequences because they enable the recognition of search patterns for SQLi attacks. The data pre-processing protocol includes the following sequential steps:

- **Handle missing value:** The algorithm replaces absent Numeric values by inputting the mean of column statistics to preserve original numerical centralities.
- **Eliminating duplicate records:** To prevent redundancy and overfitting during model training the technician removes duplicate entries from the records.
- **Tokenization:** The tokenization process divides SQL queries into key terms as well as relation operators and literal values and unique symbols. Through tokenization the system learns about typical keyword arrangements and SQL attack-related character sequences because it provides structure to sequence analysis.
- **Numerical encoding:** After tokenization, an embedding layer converts each token into a numerical identity, creating dense vector representations of syntactic and partially semantic relationships.

3.3. Feature Extraction with G-Test Scores

The G-test is a feature extraction method that evaluates the dependency between categorical features and the target variable. It uses a likelihood ratio to measure how much a feature's distribution differs across classes, selecting features with high scores for better model performance.

3.4. Data Normalization

Normalizing data is an essential component of training. There is a significant variation in the range of numerical figures in the collected data. Characteristics with such a wide range of values already have a much bigger impact on the classifier than characteristics with a narrower range of values; hence, a fraction (decimal) part of the information was omitted. Using the normalization Equation (1), it translates a given property to the interval [0, 1]:

$$x_m - \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

where x_{min} is the lowest of all the original features,x_{max} is the biggest of all the original features, and x is the value before the feature value is processed. The mapping interval's highest and lowest values are denoted by the terms high and low, appropriately

3.5. Data Splitting

The dataset is separated into two parts of size 80% and 20% for training and testing purposes, respectively.

3.6. Gated Recurrent Unit (GRU) Model

An LSTM neuronal network's additional time dedicated to training its models yields a more complicated internal structure and challenging parameter adjustment [14]. GRU is

a condensed form of an LSTM. The GRU model offers comparable prediction accuracy to the LSTM model while requiring fewer hours of training time. GRU reduces the merging of the memory's module to only two regulating components—the current gate and the reset gate—by combining the input and recollection gates of the LSTM into a single maintenance gate. The portal of updates is shown as Z_t, regulates how much information from the previous data state is incorporated into the current state. While the prior neuron handles fewer historical items, higher update gate values give current information precedence over past information. The main function of the apprise gates is to maintain memory and then identify extended patterns that occur in water quality data sequences. The capture process of information through the update gate utilizes the Equation (2) which shows:

$$Z_t = \sigma(W_Z * [h_{t-1}, X_t]) \tag{2}$$

The release gate component R_t, functions as a pivotal part in determining the amount of stored historical data. The retention of past historical data becomes more effective as reset gate values decrease since this setting helps identify short-term trends in parameterisation data on the condition of water. Equation (3) serves to compute the reset gate information that looks like this:

$$R_t = \sigma(W_r * [h_{t-1}, X_t]) \tag{3}$$

here h_t represents the unit's emission state at time t and $\bar{h}_t$ symbolizes the roundabout municipal value of h_t. The current unit data gets stored into this value before transmission but an estimate of the previous output needs calculation using Equation (4).

$$\bar{h}_t = tanh(W_{\bar{h}} * [r_t * h_{t-1}, X_t]) \tag{4}$$

The projected results for data on water quality parameters appear as Equation (5)

$$h_t = (1 - Z_t) * h_{t-1} + Z_t * \bar{h}_t \tag{5}$$

here X_t denotes the numerical value of the supplied data at this time t_t, $h_{t-1}$is the current memory cell's outcome for water's cleanliness characteristic data $t-1$, W_Z, W_r,and $W_{\bar{h}}$ indicate the cell's weight matrix, "[]" represents the relationship between two matrices, "$*$" stands for the tanh procedure is the divided curves of the registration operation, and the matrix whose result is σ is the participation constant.

3.7. Evaluation Metrics

The evaluation process for a classification model in ML consists of a confusion matrix, which takes table form. It is a matrix that summarizes the class labels for a collection of test data, both real and expected. The quantity of each class's TP, FP, TN, and FN is publicized in the confusion matrix. The amount of accurate predictions for the positive the number of accurate forecasts for the negative class is TN, the number of erroneous predictions for the positive class is FP, the number of incorrect estimations for the negative class is FN, and the class TP. Accuracy, precision, recall, and F1 score are among the assessment metrics that may be calculated using the confusion matrix.

Accuracy: The Measure achieves correct predictions through Accuracy however, this method works less effectively when dealing with unbalanced datasets. It is given as Equation (6).

$$Accuracy = \frac{\text{TP+TN}}{\text{TP+Fp+TN+FN}} \tag{6}$$

Precision: The focus of precision rests in determining accurate positive predictions to reduce false positives. The precision ratio measures positively predicted instances compared to all predicted instances for decision-making validity. It is expressed as Equation (7).

$$Precision = \frac{\text{TP}}{\text{TP+FP}} \tag{7}$$

Recall: Model recall represents the ability to detect all positive cases effectively and avoid false negative errors. The mathematical expression of recall appears as Equation (8).

$$Recall = \frac{\text{TP}}{TP+FN} \tag{8}$$

F1 score: F1-score determines evaluation control by combining precision and recall measurement to achieve class balance especially when dealing with imbalanced datasets. Mathematically, it is given as Equation (9).

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{9}$$

The combined effect of these metrics reveals how accurate and effective the model proves to be for target variable predictions.

4. Results and Discussion

A combination of these evaluation metrics helps determine both the accuracy and effectiveness of target prediction by the model. A multithreaded computer system enabled the execution using its Intel Core i7 CPU operating at 3.4 GHz in combination with an NVIDIA Geforce GTX 980M GPU having 8 GB of RAM and additional system components. The evaluation metrics performance of the proposed GRU model can be found in Table 2. The model demonstrated 96.65% accuracy, which proves its ability to detect predictions with high precision. The model exhibits precision at 97%, which demonstrates its capacity to detect genuine positives with low false alarm rates, and a recall of 92.5% proves its achievement of identifying most actual SQLi instances. The F1-score level of 94% demonstrates that the GRU model maintains proper precision-recall balance when detecting SQLi attack set variables, thus establishing its reliable and robust nature.

Table 2. Experiment Results of Proposed Models for Detection of SQL Injection Attacks

Performance Matrix	Gated Recurrent Unit (GRU) Model
Accuracy	96.65
Precision	97
Recall	92.5
F1-score	94

Table 2 shows the training accuracy (red dashed line with circles) and validation accuracy (green solid line with circles) over 20 epochs. The training accuracy generally fluctuates between approximately 0.97 and 0.99, showing an overall high performance on the training data. The authentication accuracy, representing the representation's performance on unseen data, varies between roughly 0.96 and 0.98, indicating good generalization with some fluctuations across epochs. Notably, there isn't a significant and

consistent disparity between validation accuracy and training outcomes, indicating that the training data is not being overfitted by the model.

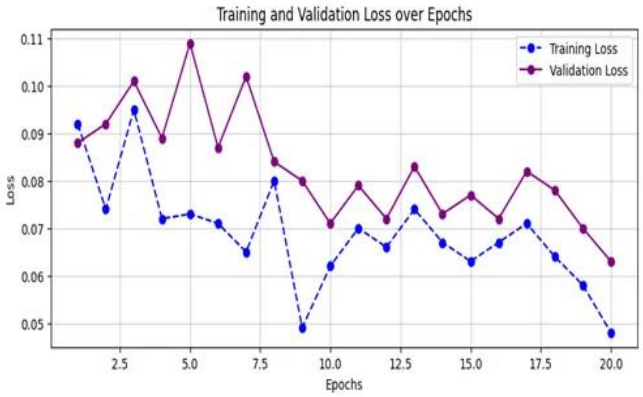

Figure 3. Loss Curves for the GRU

Figure 3 shows the training loss (blue dashed line with circles) and validation loss (purple solid line with circles) over 20 epochs. Both training and validation loss generally range roughly between 0.05 and 0.11, suggesting that the model is learning and that the error is becoming less with time. But the loss of validity exhibits more variability and, at times, spikes higher than the training loss, suggesting that when compared to training data, the arrangement of concepts performs less reliably when dealing with uncertain data.

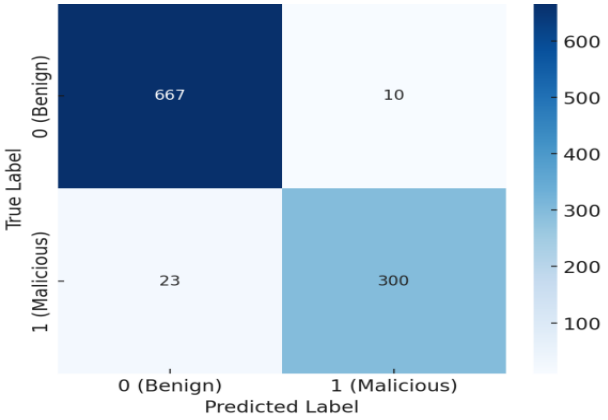

Figure 4. Confusion Matrix for the GRU model

Figure 4 presents the GRU algorithm's confusion matrix. cataloguing presentation, with 2 classes, malicious and benign. The matrix indicates 667 TN and 300 true positives, showing the model accurately identified both benign and malicious cases. Misclassifications were minimal, with only 10 FP and 23 FN, demonstrating the GRU model's strong classification capability. The model performs slightly better at identifying benign instances but overall shows balanced and effective detection across both classes.

4.1. Comparison with Discussion

Here, the experimental results are compared between ML and DL models (see Table 3) for detecting SQLi attacks. The GRU model delivered 96.65% accuracy which surpassed all other traditional ML-based models. The SVM scored 91% accuracy with KNN at 87.6%. SQLi attack detection operates more efficiently with GRU models because they

successfully capture sophisticated temporal data patterns better than traditional ML solutions.

Table 3. Comparison between ML and DL models for SQL Injection Attack Detection on SQL Injection Dataset

Models	Accuracy
Support Vector Machine (SVM) [15]	91
K-Nearest Neighbors (KNN) [16]	87.6
Proposed GRU	96.65

The proposed model using GRU technology provides multiple benefits for SQLi detection by understanding sequential SQL query dependencies, which results in better accuracy rates. Traditional ML models differ from the GRU network because it maintains effective processing of inputs with different lengths and learns complete context for better accuracy measurements. The model effectively detects benign from malicious queries through its 96.65% accuracy rate accompanied by 94% F1-score measurement. The model shows excellent generalization to unknown data through its minimal overfitting behavior and steady validation results which makes it a dependable solution for real-time web application security operations.

5. Conclusion and Future Study

Security exploits escalate because of growing Information Technology application usage within distributed environments. The primary goal of SQLi attacks is database theft from back-end server organizations by penetrating their customer database systems. The accuracy of the suggested GRU-based DL framework in detecting SQLi assaults is high, achieving 96.65% with 97% precision, 92.5% recall, and 94% F1-score. When it comes to identifying sequential patterns in SQL queries, the suggested model outperforms SVM (91%), and KNN (87.6%) in conventional machine learning models. One considerable drawback of the model comes from its dependence on fixed labeled data records that cannot capture potential changes or camouflage within SQLi assaults.

The field requires immediate attention on these identified deficiencies. Research efforts should concentrate on acquiring extensive datasets that present integrated SQL and NoSQL attack types. The development of attack-resistant models needs equal importance together with their ability to adapt through diverse attack vector evolution. Deep learning and transfer learning approaches should be studied to automate feature extraction because they would make the process less dependent on domain expertise. Practical deployment needs the integration of ML-based tools into real-time security frameworks with low latency performance and high scalability.

References

[1] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," *ACM SIGPLAN Not.*, 2006, doi: 10.1145/1111320.1111070.

[2] J. Minhas, "Blocking of SQL Injection Attacks by Comparing Static and Dynamic Queries," *Int. J. Comput. Netw. Inf. Secur.*, 2012, doi: 10.5815/ijcnis.2013.02.01.

[3] C. Bockermann, M. Apel, and M. Meier, "Learning SQL for database intrusion detection using context-sensitive modelling (extended abstract)," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009. doi: 10.1007/978-3-642-02918-9_12.

[4] *Kalla, D., & Chandrasekaran, A. (2023). Heart disease prediction using machine learning and deep learning. International Journal of Data Mining & Knowledge Management Process (IJDKP), 13(3). Science*.

[5] E. Shafie and A. Cau, "A framework for the detection and prevention of SQL injection attacks," in *11th European Conference on Information Warfare and Security 2012, ECIW 2012*, 2012.
[6] M. S. Aliero and I. Ghani, "A component based SQL injection vulnerability detection tool," in *2015 9th Malaysian Software Engineering Conference, MySEC 2015*, 2016. doi: 10.1109/MySEC.2015.7475225.
[7] M. Y. Kim and D. H. Lee, "Data-mining based SQL injection attack detection using internal query trees," *Expert Syst. Appl.*, 2014, doi: 10.1016/j.eswa.2014.02.041.
[8] N. M. Sheykhkanloo, "Employing Neural Networks for the detection of SQL injection attack," in *ACM International Conference Proceeding Series*, 2014. doi: 10.1145/2659651.2659675.
[9] Kuraku, D. S., & Kalla, D. (2023). Phishing Website URL's Detection Using NLP and Machine Learning Techniques. *Journal on Artificial Intelligence-Tech*
[10] M. Hasan, Z. Balbahaith, and M. Tarique, "Detection of SQL Injection Attacks: A Machine Learning Approach," in *2019 International Conference on Electrical and Computing Technologies and Applications, ICECTA 2019*, 2019. doi: 10.1109/ICECTA48151.2019.8959617.
[11] U. Noor, Z. Anwar, A. W. Malik, S. Khan, and S. Saleem, "A machine learning framework for investigating data breaches based on semantic analysis of adversary's attack patterns in threat intelligence repositories," *Futur. Gener. Comput. Syst.*, 2019, doi: 10.1016/j.future.2019.01.022.
[12] K. Zhang, "A machine learning based approach to identify SQL injection vulnerabilities," in *Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019*, 2019. doi: 10.1109/ASE.2019.00164.
[13] M. R. Ul Islam, M. S. Islam, Z. Ahmed, A. Iqbal, and R. Shahriyar, "Automatic detection of NoSQL injection using supervised learning," in *Proceedings - International Computer Software and Applications Conference*, 2019. doi: 10.1109/COMPSAC.2019.00113.
[14] P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, "SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel," *J. Inf. Secur. Appl.*, 2018, doi: 10.1016/j.jisa.2018.04.001.
[15] M. Chattopadhyay, R. Sen, S. Gupta, and others, "A comprehensive review and meta-analysis on applications of machine learning techniques in intrusion detection," *Australas. J. Inf. Syst.*, vol. 22, 2018.
[16] Kuraku, S., Kalla, D., Samaah, F., & Smith, N. (2023). Cultivating proactive cybersecurity culture among IT professional to combat evolving threats. *International Journal of Electrical, Electronics and Computers*, *8*(6).
[17] Y. H. Rajarshi Tarafdar, "Finding Majority for Integer Elements," *J. Comput. Sci. Coll.*, vol. 33, no. 5, pp. 187–191, 2018.
[18] K. Kamtuo and C. Soomlek, "Machine learning for SQL injection prevention on server-side scripting," in *20th International Computer Science and Engineering Conference: Smart Ubiquitos Computing and Knowledge, ICSEC 2016*, 2017. doi: 10.1109/ICSEC.2016.7859950.
[19] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," in *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*, 2017. doi: 10.23919/INM.2017.7987433.
[20] Kuraku, D. S., Kalla, D., Smith, N., & Samaah, F. (2023). Safeguarding FinTech: elevating employee cybersecurity awareness in financial sector. *International Journal of Applied Information Systems (IJAIS)*, *12*(42).
[21] Kalla, D., Smith, N., Samaah, F., & Polimetla, K. (2022). Enhancing Early Diagnosis: Machine Learning Applications in Diabetes Prediction. *Journal of Artificial Intelligence & Cloud Computing. SRC/JAICC-205. DOI: doi. org/10.47363/JAICC/2022 (1)*, *191*, 2-7.