

SQL-CB-GuArd: a deep learning mechanism for structured query language injection attack detection

AsifIqbal Sirmulla, Prabhakar Manickam

School of Computer Science and Engineering, Reva University, Bangalore, India

Article Info

Article history:

Received Mar 4, 2024

Revised Jul 3, 2024

Accepted Jul 26, 2024

Keywords:

Attention mechanism
Bidirectional long short-term memory
Convolutional neural network
Deep learning
Gated recurrent unit
Natural language processing
Structured query language injection attack

ABSTRACT

Structured query language (SQL) injection attacks, which take advantage of input field vulnerabilities to introduce malicious code into database queries, are a serious danger to database-driven programs and systems. Intruders can now alter, recover, or remove sensitive data because of illegal access. Strong artificial intelligence (AI) based security solutions are required to reduce SQL injection threats, as these assaults' significance highlights. This study's main goal is to create automated AI-based techniques that can identify structured query language injection attack (SQLIA) in real time eliminating the need for human intervention. Although machine learning (ML) and deep learning-based techniques have received a lot of interest in this field, ML-based techniques have problems with accuracy and false negatives. Deep learning (DL) is therefore commonly used in these text data processing and natural language processing (NLP) applications. We have introduced a hybrid DL approach for SQLIA detection in this paper. The pre-processing step performs decoding, generalization, and tokenization to improve the learning performance. The proposed approach uses combination of convolutional neural network (CNN), bidirectional long short-term memory (Bi-LSTM), gated recurrent unit (GRU) with attention mechanism. The combination helps to improve the pattern learning capacity. The proposed approach is validated on publically available data and experimental analysis reported that the proposed SQL-CB-GuArd achieves better accuracy of SQLIA detection.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

AsifIqbal Sirmulla
School of Computer Science and Engineering, REVA University
Bangalore, India
Email: aisirmulla@gmail.com

1. INTRODUCTION

In today's world, information technology has become essential in various aspects of our lives, including communication, transportation, and banking. While these advancements bring many benefits, they also open doors for malicious actions in cyberspace [1]. Similarly, the modern web-based technologies and cloud based systems got very popular due to their diverse range of applications [2], [3]. Therefore, these systems are easily accessible for every individual. However, these systems raise serious concerns over security of the systems. Several studies have reported the increase in cyberattacks. For example, in 2020, there were about 4,000 cyberattacks recorded daily worldwide. Cybersecurity ventures estimated that by 2021, cybercrime would cost the world economy \$6 trillion annually, compared to \$3 trillion in 2015 [4]. Ransomware attacks have seen a significant increase, with a 150% rise in 2020 compared to 2019 [5]. Similarly, phishing attacks remain a major threat, with over 240,000 unique phishing websites detected each month in 2020 [6]. The anti-phishing working group reported a 22% increase in phishing attacks in the first

half of 2021 compared to the same period in 2020. These attacks often lead to data breaches, which have become a significant concern. In 2020, over 37 billion records were exposed in breaches, marking a 141% increase from 2019 [7], [8]. Based on IBM's cost of a data breach report [9], the average time to detect and remediate a data breach is 280 days. Hence, it's crucial to protect cyber infrastructures from harmful threats.

Similarly, the IT infrastructure also has noticed a tremendous growth and its dependency on the technology has increased drastically. Therefore, all most all applications which are used in daily life are utilizing web-based system to operate them efficiently. As discussed before these web-based applications are easily accessible to individuals therefore it brings various security challenges because of illegal access [10]. When users use web apps or portals, user-generated data gets stored within these platforms or their respective backend databases. These databases are overseen by database management systems (DBMS), which execute request queries scripted in the form of structured query language (SQL). Both the frontend app and backend db are susceptible to various forms of assaults due to their internet accessibility. Amongst the various vulnerabilities outlined by OWASP, structured query language injection attack (SQLIA) rank among the top 10 [11]. Recent trends indicate a staggering increase in attack rates, exceeding 300% over the past decade, with attackers employing sophisticated tactics like obscured or cipher codes to bypass security systems [12]. SQLIA occurs when an SQL query is injected or inserted via a client-side data input field into the online app or portal [13]. The injected query can be malicious, potentially resulting in database modification, identity spoofing, sensitive information leakage, and repudiation issues if successfully executed [14]. Therefore, several methods have been introduced to address the SQLIA issue such as input validation, parameterized queries, web application firewalls (WAF), input filtering and sanitization, database activity monitoring (DAM), error handling and logging, and black box and white box testing [15]. Table 1 shows a comparative analysis where the advantages and drawbacks of these models are briefly described.

Table 1. Traditional methods of attack detection

Method	Description	Drawback
Input validation	Putting in place stringent input validation procedures to guarantee that data submitted by users satisfies expectations regarding format, length, and data type.	Fails in complex applications with numerous input fields.
Parameterized queries	Using prepared statements or parameterized searches when interacting with databases. By separating SQL code from user input, these techniques stop injected SQL instructions from being executed.	It requires significant changes in existing codebase and implementation can be error prone
WAF	It keeps an eye on and filters HTTP traffic going between the internet and an online portal. Based on specified rulesets, WAFs are able to identify and prevent SQL injection activities.	It may lead to generate the false positive and false negative leading to block the legitimate traffic or allowing the malicious traffic
Input filtering and sanitization	It removes or neutralize potentially harmful characters or commands from user input before processing it.	Sometimes it fails to catch all variations of input malicious traffic
DAM	It offers tools for real-time DAM and analysis. It can detect anomalous SQL queries or behaviors indicative of a potential SQLIA.	It leads to produce high volume of alerts
Error handling and logging	It implements error handling within the application to capture and log any SQL-related errors. Analyzing error logs can help identify potential SQL injection attempts.	Large log volumes becomes time-consuming

Presently, machine learning (ML) based systems have gathered huge attention in various applications such as image processing, video processing biomedical domain, textual data analysis and natural language processing (NLP) tasks. ML based methods have been considered as promising method to detect the SQL attacks and other intrusion successfully. Nofal and Amer [16] presented a combined approach where neural network and fuzzy logic methods were used for SQLIA detection. Sheykhkanloo [17] presented in neural network based model which uses uniform resource locator (URL) generator, URL classifier, and NN model. Similarly, support vector machine (SVM), decision tree (DT), and k-nearest neighbors (KNN) classifiers are also employed for SQLIA detection. However, the reliability and generalization of these methods remains a challenging task. Therefore, deep learning (DL) based methods have been adopted to improve the performance. the literature review presents the brief discussion about these methods.

In this work, we present a hybrid DL based solution for SQLIA detection. The proposed work uses combination of data pre-processing, vectorization, and implementation of DL model. The data pre-processing stage implements decoding, generalization, and tokenization. In next stage vectorization is performed. Finally, the DL model is implement which uses convolutional neural network (CNN), bidirectional long short-term memory (Bi-LSTM), and gated recurrent unit (GRU) with attention mechanism.

Rest of the article is arranged in following section. Section 2 presents brief literature review about standard methods. Section 3 showcases the proposed model. Section 4 presents the outcome of proposed model along with the comparative analysis with existing methods. Finally section 5 presents the concluding remarks about the work.

2. LITERATURE SURVEY

This section provides a quick overview of the literature on current techniques for detecting SQLIA through the use of DL and ML techniques. Vartouni *et al.* [18] focused on enhancing WAF using deep ML algorithms, specifically focusing on anomaly detection. Initially, attributes are constructed from HTTP data using two models: n-gram and one-hot. Subsequently, employing auto-encoder long short-term memory networks (AE-LSTM) as an unsupervised DL technique, informative features are extracted and reduced. Finally, an ensemble isolation forest is utilized to train the classifier exclusively on normal data. Thalji *et al.* [19] presented DL based model by using recurrent neural network (RNN) based pattern learning approach. Moreover, it implements autoencoder based system along with RNN. The encoder module considers the input data and compresses the data to lower dimension. This encoded data then fed into the decoder module which reconstructs the data. Finally, long short-term memory networks (LSTM) and dense layer based RNN model is employed for classification.

Gandhi *et al.* [20] reported that traditional prevention methods, such as rule-based matching, have limitations in detecting a wide range of SQLIA. One major challenge is the constant evolution of malicious SQL queries by hackers. To address this, leveraging ML algorithms for predicting SQLIA proves promising. This paper proposes a hybrid CNN-Bi-LSTM method for identification of SQLIA. the complete architecture utilizes several layers such as embedding layer, convolution layer, and max pooling with Bi-LSTM. Finally, dense and output layers are used to obtain the final outcome. Sun *et al.* [21] reported accuracy and false negative issues of existing methods. Recent ML research on SQLIA identification has focused mainly on extracting the features. However, the efficacy of identification relies heavily on the accurateness of features extracted, which may not adequately address more complex SQLIA. To tackle these challenges, a novel approach to SQLIA detection, combining an enhanced TextCNN and LSTM is presented in this work. In order to acquire local features, this technique first vectorizes samples inside the corpus and then uses an enhanced TextCNN. The sequence information present in the samples is then captured using a Bi-LSTM network. Understanding that LSTM is less successful with longer sequences, it employs an attention mechanism to shorten the gap between any two words in the sequence to one. The feature selection procedure also incorporates pre-trained word vector features acquired using bidirectional encoder representations from transformers (BERT) for transfer learning. A mixed ML model based on LSTM and SVM was reported by Fang *et al.* [22]. Initially the likelihood ratio is used, and then SQL tokenization. This produces a word vector model in SQL. This model is given into the LSTM model as a train set, and it classifies the patterns as authentic and injected queries. Kherbache *et al.* [23] talked about how feature selection, which involves choosing the most critical qualities to increase efficiency and reduce false positives, is seen to be a crucial step in anomaly-based intrusion detection. Authors offered a mixed agglomerative hierarchical clustering approach with SVM classification based on this idea. Based on their commonalities, features are categorized and feature selection is carried out according to variance. A CNN-based model for online security was provided by Jemal *et al.* [24], who also examined the significance of the hyper-parameters included in CNN models.

3. PROPOSED MODEL

This section presents the proposed DL based model for SQLIA detection. As discussed before, the traditional methods have several disadvantages which are further addressed by DL based methods. However, the ever evolving process of attackers affect the SQLIA detection performance. In order to overcome the issues of existing models, we introduced a new hybrid DL model which utilizes four different concepts to improve the performance. The overview of these concepts is given as follows:

- CNN: it is a class of DL mechanism commonly employed for image and non-image (including textual) data. The CNN based method utilizes learnable filters (also called kernels or convolutional kernels) to convolve over the input data, fetching local patterns and features.
- LSTM: It is a kind of RNN architecture intended to recognize long-term relationships in sequential data and solve the vanishing gradient issue. LSTM networks are appropriate for jobs where comprehending context over long distances is critical because they are able to selectively store or discard info over lengthy durations. They are now a common option for many sequence modeling tasks, particularly those involving time-series data or jobs involving NLP, such text creation, sentiment analysis, and machine translation.

- GRU: Akin to the LSTM, the GRU is another kind of RNN design that is intended to handle the vanishing gradient issue and identify long-term relationships in sequential data. GRUs are more effective in training and deploying than LSTMs since they have fewer parameters and need less computing power.
- Attention mechanism: When making predictions, neural networks, especially sequence-to-sequence models, employ this technique to concentrate on pertinent portions of the input data. It enables the model to focus on specific segments of the input sequence, giving each segment varying degrees of significance according to how relevant it is to the present forecast.

The proposed architecture is designed based on the combination of these architectures. Figure 1 depicts the complete proposed architecture for SQLIA detection. It includes four different phases where first phase includes data pre-processing which includes decoding, generalization, and tokenization, next phase performs vectorization, phase III uses this vectorised data through the proposed classifier model and finally, the classification outcomes are analyzed.

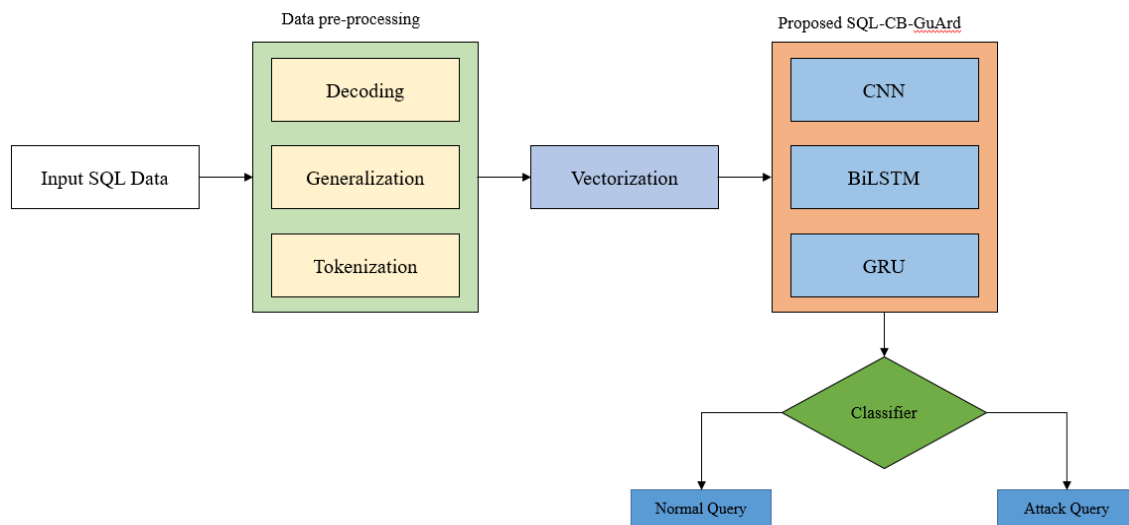


Figure 1. Proposed SQL-CB-GuArd (SQL-CNN Bi-LSTM Gated Recurrent Attention Mechanism)

3.1. Data pre-processing

This step performs several tasks as the data pre-processing phase such as data decoding, generalization, and tokenization. Data decoding includes generalization and tokenization tasks. In next phase, vectorization converts the dataset into numerical vectors.

3.1.1. Decoding

The first stage of this work performs data pre-processing where data decoding, generalization and tokenization tasks are performed. The data decoding refers to the process of converting encoded or sanitized input data back to its original form before further analysis or processing. The decoding process involves reversing the encoding or sanitization applied to the input data, so that the original input can be analysed for signs of SQLIA. For example, if user input has been URL-encoded or hyper text markup language (HTML)-encoded to prevent SQL injection, the defence mechanism would need to decode the input data to recover the original user input before passing it to the SQL injection detection model. Similarly, if special characters have been escaped or replaced with safe alternatives, you would need to reverse this process to obtain the original input data. Table 2 shows the example of URL encoded and HTML encoded inputs by user and their corresponding decoded output. This step converts the complex hyperlinks in simplified forms.

Table 2. Example of decoding URL encoding and HTML decoding

	Input	Decoded	Operations
URL encoding	"Hello%20World%21"	Hello World!	%20 replaced with space character
HTML decoding	"<script>alert('XSS Attack!')</script>"	<script>alert('XSS Attack!')</script>	the HTML entities < and > are replaced with < and >, respectively.

3.1.2. Generalization

In this stage, the data generalization operation is performed where the main aim is to minimize the redundant and irrelevant distribution. This generalization process includes several steps such as:

- Handling the URL: generally, the input data consist of several URLs which are replaced with a placeholder string such as “https://website”. This step ensures the mitigating the impact of specific URLs, especially if the URLs are not relevant to the task at hand or if they contain sensitive information.
- Replacing numbers: Numbers in the data are replaced with a placeholder value, such as '0'. This step is often done to generalize numerical data and focus on the structural aspects of the data rather than the specific numeric values. It can help reduce the dimensionality of the data and mitigate the influence of numeric outliers.
- Removing extra unique qualities: Certain unique qualities in the data, such as control characters or blank characters, are removed. Control characters, such as newline characters or tab characters, can introduce noise or artifacts in the data that may interfere with analysis or modeling. Removing these characters helps in standardizing the data and making it more responsive to processing. For example, the SQL injection query is "SELECT * FROM users WHERE username = 'admin' --' AND password = 'password'" and after removing unique qualities it becomes "SELECT * FROM users WHERE username = 'admin' AND password = 'password'".

3.1.3. Tokenization

Tokenization plays important role in this domain of SQLIA detection. This process breaks down the SQL queries or input into various smaller and meaningful units which are called as tokens. Each token represents the word, symbol or other elements in the query. Moreover, it helps to extract the significant features for analysis. The tokenization performs several steps such as splitting queries into token, removing comments, extracting features from tokens such as token frequencies, n-grams (sequences of tokens), syntactic patterns, or semantic information derived from the tokens, and building the vocabulary. However, this work uses different scripting languages therefore the feature extraction is performed based on input data. Thus, the proposed tokenization approach performs several steps to accomplish the tokenization process such as identifying starting and ending labels, identification of windows events and function names, assigning unique tokens, analysing the vocabulary list and predefined delimiter replacement. Table 3 shows the complete overview of tokenization process.

- Identification of starting and ending labels: this process involves identifying the starting and ending labels of various elements in web content such as identifying the HTML tags, event handles, JavaScript function etc. For example, in HTML, tags are enclosed within angle brackets (< >), and JavaScript functions are defined using the function keyword followed by a pair of parentheses and curly braces ({ }).
- Identification of window event and function names: in this phase, the proposed tokenization process involves identifying the certain features related to the windows events and function names which are commonly used in XSS attacks. These event handlers are *onclick*, *onmouseover*, and *onload*. Similarly, the function names are “eval”, “alert”, “prompt”.
- Unique token assignment: once the starting and ending labels, as well as Windows event and function names, are identified, unique tokens are assigned to each of these features. These tokens serve as representations of the features and are used to create a vocabulary list.
- Checking tokens in vocabulary list: in this step the tokens obtained from the input data is checked against the vocabulary list which consist of predefined tokens to represent the identified attributes. If the obtained token matches with any of the predefined attribute, then it is retained otherwise it is replaced with the predefined delimiter or placeholder token.

3.2. Vectorization

This is the next step after pre-processing which is used to represent the SQL queries or input data as numerical vectors. This representation is helpful for ML and DL methods to perform classification tasks. Several methods have been introduced to perform the vectorization such as bag of word, term frequency-inverse document frequency (TF-IDF), and word embeddings etc. In this work, we employed word2vec model for word embedding. After completing the pre-processing tasks, this word embedding model is applied. According to this, initially vocabulary is created which is constructed by selecting the most common words from tokenized data. This vocabulary consists of unique words present in the corpus and serves as the basis for generating word embeddings. Typically, words are represented by numerical indices in the vocabulary.

After vocabulary, the *word2vec* model need to be trained. Word2Vec utilizes a neural network architecture, specifically either the “continuous bag of words (CBOW)” model or the “skip-gram model”, to learn distributed representations of words based on their contexts. This stage involves training the neural

network to estimate a word's probability provided its surrounding words (for CBOW), or given a target word, its adjacent words (for skip-gram). This training process involves adjusting the neural network's parameters (word vectors) to minimize prediction errors.

Let W represent the Word2Vec model, which consists of a neural network with parameters θ . The training process aims to minimize a loss function \mathcal{L} by adjusting the parameters θ . Mathematically, this can be represented as (1):

$$\min_{\theta} \cdot \frac{1}{N} \sum_{i=1}^N \mathcal{L}(W(T_i), t_{ic}) \quad (1)$$

Where $W(T_i)$ is the output of the Word2Vec model for the input token sequence T_i , and t_{ic} is the context token for the central word in the sequence.

Following training, every word in the vocabulary has an embedding, or dense vector representation, in a continuous vector space. This is known as the Word2Vec model. Similar words are represented by adjacent vectors in the vector space as a result of these embeddings, which reflect the semantic links between words. The vector embeddings generated by Word2Vec are mapped with the vocabulary created earlier. Each word in the vocabulary is associated with its corresponding vector representation in the continuous vector space. For example, let us consider that we have a SQL query given as:

“SELECT * FROM users WHERE username='admin'”

This query is processed through the Word2Vec model to produce embeddings for each word in the query. The outcome of this Word2Vec model is presented as:

```
Word2Vec('SELECT')=[0.1, 0.2, 0.3],
Word2Vec('*')=[0.2, 0.3, 0.4]
Word2Vec('FROM')=[0.3, 0.4, 0.5]
Word2Vec('users')=[0.4, 0.5, 0.6]
Word2Vec('WHERE')=[0.5, 0.6, 0.7]
Word2Vec('username')=[0.6, 0.7, 0.8]
Word2Vec('=')=[0.7, 0.8, 0.9]
Word2Vec('admin')=[0.8, 0.9, 1.0]
```

Further, in order to obtain the final vector representation of given SQL query, we perform averaging on the Word2Vec embeddings with the help of tokens, which is expressed as:

SQL_Query_Vector = (Word2Vec('SELECT') + Word2Vec('*') + Word2Vec('FROM') + Word2Vec('users') + Word2Vec('WHERE') + Word2Vec('username') + Word2Vec('=') + Word2Vec('admin'))/8
 =([0.1, 0.2, 0.3]+[0.2, 0.3, 0.4]+[0.3, 0.4, 0.5]+[0.4, 0.5, 0.6]+[0.5, 0.6, 0.7]+[0.6, 0.7, 0.8]+[0.7, 0.8, 0.9]+[0.8, 0.9, 1.0])/8
 =[0.475, 0.575, 0.675] is the final SQL_Query_Vector. This vector captures the semantic meaning of the query in a continuous vector space, which can be used as input to proposed hybrid DL model. The proposed architecture is discussed in next section.

Table 3. Tokenization process

Tokenization	Input (potentially with SQLIA)	Tokenized Output	Description
Starting and end label identification	<input type="text" name="username" value="" OR 1=1; --">	['<', 'input', 'type', '=', '"text"', 'name', '=', '"username"', 'value', '=', '" OR 1=1; --"', '>']	<input> is identified as a starting label, and its corresponding ending label > is also recognized.
window event and function names identification	<script> var username = 'admin'; var password = 'password'; function authenticate() { var query = "SELECT * FROM users WHERE username='" + username + " AND password='" + password + "';"; executeQuery(query); } </script>	['<', 'script', '>', 'var', 'username', '=', '"admin"', ';', 'var', 'password', '=', '"password"', ';', 'function', 'authenticate', '(', '(', ')', '{', 'var', 'query', '=', '"SELECT * FROM users WHERE username=\'" + username + "\' AND password=\'" + password + "\';"', '}', 'executeQuery', '(', '(', 'query', ')', ')', ';', '}', '<', '/script', '>']	the JavaScript function executeQuery and the SQL query string within the authenticate function are identified as features of interest. The SQL query string concatenates user inputs username and password, making it vulnerable to SQL injection.
Unique token assignment	<input type="text" name="search" value=""); DROP TABLE users; --">	['<', 'input', 'type', '=', '"text"', 'name', '=', '"search"', 'value', '=', '""); DROP TABLE users; --"', '>']	he SQL injection payload); DROP TABLE users; -- is treated as user input and tokenized accordingly.

3.3. Proposed SQL-CB-GuArd (SQL-CNN Bi-LSTM gated recurrent attention mechanism) for SQLIA detection

This section describes the proposed DL based solution for SQLIA detection. The proposed architecture is developed by using the combination of CNN, Bi-LSTM and GRU models to obtain the final classification model. The complete detail of this model is described as follows.

3.3.1. Convolutional neural network

The CNN model plays important role in extracting the features from the input data. For obtaining the features from the input data, it uses several kernels to conduct convolution operations. To increase the network's flexibility, the kernel is made up of bias term and trainable weight coefficients. This is the fundamental layer of the network and the output of this layer is processed further where ReLU activation function is performed which is expressed as (2):

$$f(x) = \max(0, x) \quad (2)$$

The ReLU activation helps to mitigate the vanishing gradient and improves the training process. In next step, max pooling operation is performed. The pooling operation downsamples the duplicate data and focuses on identifying the invariance in the network. here, two pooling operations are performed such as max pooling and average pooling. The max pool helps to select the maximum value and average pool helps to select the average value as output. The pooling operations helps to reduce the spatial dimension of feature maps resulting in the increase of network efficiency. Finally, the fully connected layer is employed to establish the relation between input and output. The outcome of CNN model can be expressed as (3):

$$y_i = CNN(x_i) \quad (3)$$

According to the SQLIA model, we consider that the input sequence is denoted by X with dimension (T, D) where T is the sequence length and D is the embedding dimension. Let Z^c represents the output feature map with the dimension (T', F) after applying the CNN and max pooling where T' is the sequence length obtained by applying pooling and F represents the number of filter. Thus, the convolution operation can be represented as (4):

$$Z^{(c)} = ReLU(Conv(X)) \quad (4)$$

3.3.2. Bidirectional long short-term memory

The outcome of CNN model is fed into the Bi-LSTM model. This model is based on the working of RNN model. However, the traditional RNN models suffer from the issue of vanishing gradient problem. The basic LSTM model consist of three components called as input, forget and output gates as depicted in Figure 2. The main aim of LSTM model is to regulate the states of these gates to improve the learning performance. The forget gate f_t is used to ensure whether to keep the information of previous state ($c_t - 1$) or not. The input gate (i_t). Similarly, the input gate (i_t) determines the extent to which information from the input text (x_t) and the previous hidden state ($h_t - 1$) should influence the updating of the cell state. Its output can be either 0 or 1. The value of c_t represents the newly generated cell state resulting from computational operations involving $c_t - 1$, f_t , and i_t . Meanwhile, the output gate (o_t) regulates the transmission of information from the current cell state to the hidden state, with its value also being either 0 or 1. These operations can be expressed as (5):

$$\begin{aligned} f_t &= \text{sigmoid}(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \\ i_t &= \text{sigmoid}(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \\ c_t &= c_{t-1} \odot f_t + i_t \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \\ o_t &= \text{sigmoid}(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad (5)$$

Where $x_t \in R^n$ represents the input vector, $W \in R^{v \times n}$, $b \in R^v$ and n, v represents the input vector dimension and number of words in the vocabulary.

In the LSTM network, information propagates solely in a forward direction, implying that the state at time t is influenced only by information preceding t . However, to fully capture the semantic context of an input review, subsequent information is equally significant as previous ones. Thus, for a more comprehensive representation of contextual information, the Bi-LSTM model is utilized. The Bi-LSTM model consists of two LSTM networks and has the capability to process input reviews in both forward and backward directions. Therefore, we adopt the Bi-LSTM model in this work. The Figure 3 depicts the Bi-LSTM layer.

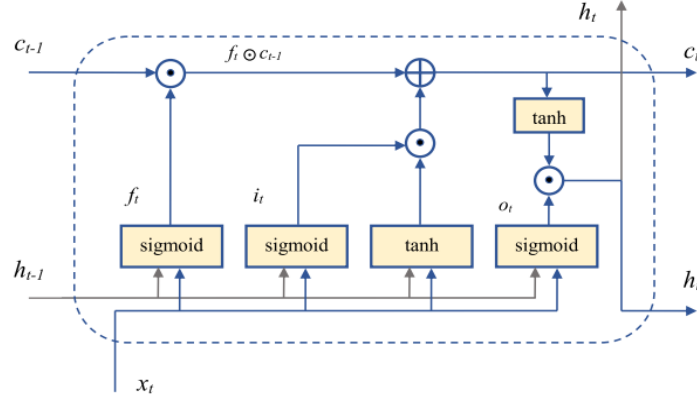


Figure 2. LSTM model

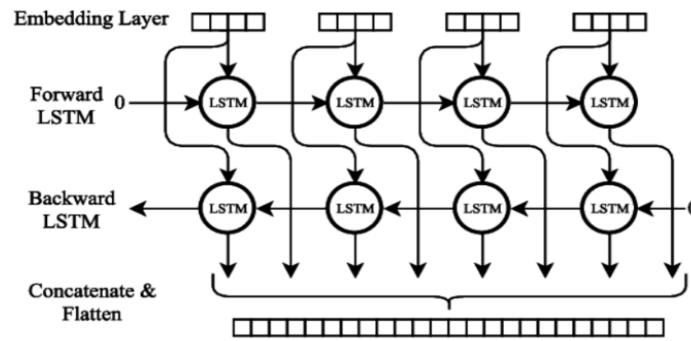


Figure 3. Bi-LSTM architecture

The forward and backward LSTM processes are expressed as (6):

$$\begin{aligned}\vec{h}_t &= LSTM(x_t, \vec{h}_{t-1}) \\ \tilde{h}_t &= LSTM(x_t, \tilde{h}_{t+1})\end{aligned}\quad (6)$$

Finally, the output of Bi-LSTM is summarized by concatenating the forward and backward states as (7):

$$h_t = [\vec{h}_t, \tilde{h}_t] \quad (7)$$

According to the proposed model, the $H^{(l)}$ represents the output hidden states obtained from Bi-LSTM model. The dimension of $H^{(l)}$ are given as $(T', 2H)$ where H represents the number of hidden units. The Bi-LSTM operation can be expressed as (8):

$$H^{(l)} = BiLSTM(X) \quad (8)$$

3.3.3. Gated recurrent unit and attention model

Along with CNN and Bi-LSTM, we incorporate the GRU model which plays important role in sequence modelling tasks. In a typical GRU cell, there are two gates: the reset gate (r) and the update gate (z). Similar to an LSTM cell, the computation of the hidden state output at time t involves the hidden state at time $t-1$ and the input time series value at time t can be expressed as (9):

$$h_t = f(h_{t-1}, x_t) \quad (9)$$

According to the proposed model, the output of GRU can be expressed as (10):

$$H^g = GRU(H^{(l)}) \quad (10)$$

In order to improve the overall performance of SQLIA, we also incorporate the attention mechanism where α represents the attention weights which are computed with the help of hidden states $H^{(g)}$. The attention weights can be computed as (11):

$$\alpha = softmax(Linear(H^{(g)})) \quad (11)$$

With the help of hidden states, the context vector can be obtained by estimating the weighted sum of these states, which can be expressed as (12):

$$C = \sum_{t=1}^{T'} \alpha_t \cdot H_t^{(g)} \quad (12)$$

The overall architecture of proposed model is represented in Figure 4 which shows the layer, its parameters and connection to another layer.

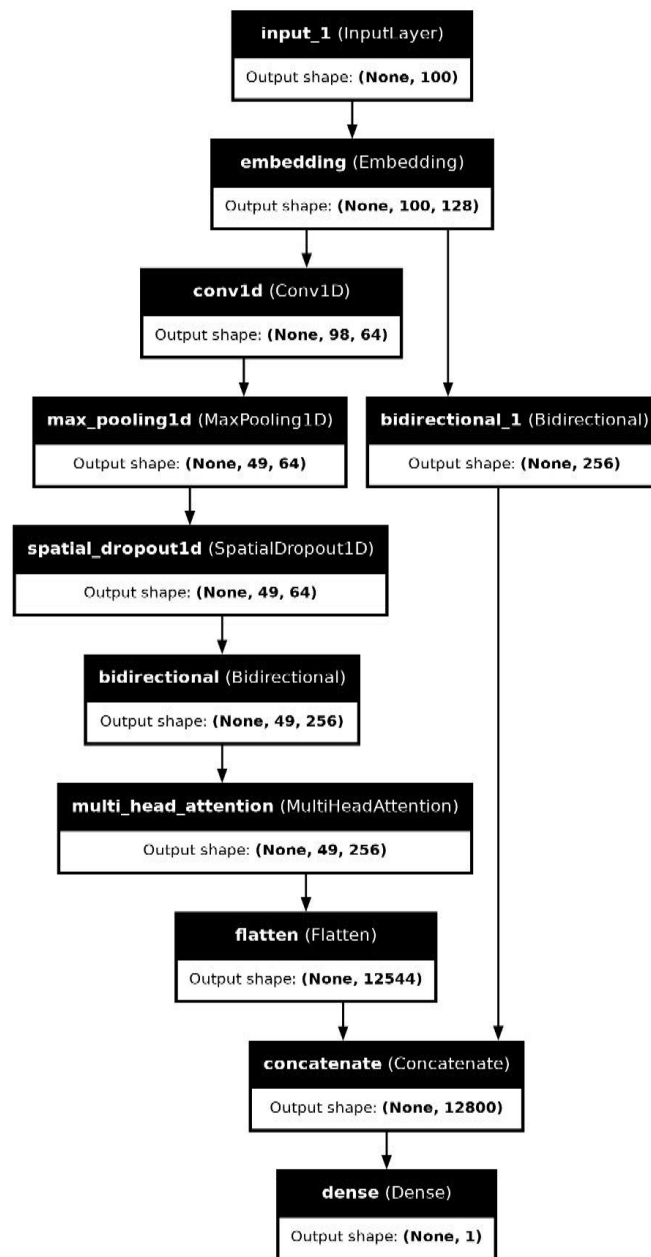


Figure 4. overall architecture of proposed model

4. RESULTS AND DISCUSSION

This section presents the results of the suggested technique and makes a comparison between the acquired performance and several current SQLIA detection schemes. The first subsection presents the overview of dataset used including performance measurement parameters. Next subsection describes the obtained performance, and finally a comparative analysis is presented.

4.1. Dataset details and performance measurement parameters

The performance of proposed model is validated on publically available datasets which are obtained from GitHub and Kaggle websites. Table 4 presents the dataset details. In this table, five different CSV dataset files are used which are named as dataset 1, dataset 2, dataset 3, dataset 4, and dataset 5 which are finally combined together to generate the final dataset.

Table 4. Dataset details

Dataset	SQLi Query	Legitimate Query
Dataset 1	158	994
Dataset 2	1594	994
Dataset 3	5433	994
Dataset 4	22559	19009
Dataset 5	29379	20002

Similarly, we utilized a publicly available SQL query dataset [21] for our research experiments. The dataset comprises three files: “sqli.csv,” “sqliv2.csv,” and “SQLiV3.csv,” which we combined. For initial preprocessing, we encoded the target labels ‘Attack’ (1) and ‘Benign’ (0). A PC equipped with an Intel i7 10th GEN CPU (2.20 GHz), 136 GB of RAM, and an 8 GB NVIDIA graphics card was used for all testing. Tensorflow 1.4.1 is used as a backend calculation, Python 3.7.13, the sklearn 1.0.2 package for ML methods, and Keras 2.8.0 to construct neural networks. The simulation settings utilized in this investigation are displayed in Table 5.

Table 5. Simulation parameters

Parameter Name	Considered value
Hidden Units for LSTM	[16, 32, 64]
Filters for CNN	[16, 32, 64]
Memory Units for LSTM	32
Embedding Dimension	32
Hidden Units for FCN	250
Training batch	50
Optimizer	Adam
Loss Function	Binary Cross Entropy
Cross Validation	10-Fold

According to this experiment, we have considered 32 memory units for LSTM model and the filter size for CNN is also considered as 32. However, these parameters were tested for a range of [16, 2, 64]. This model uses a Fully connected layer and ReLU activation function is also applied for output layer. The performance of this approach is measured with the help of confusion matrix which uses “Normal” and “Attack” classes to identify the true positive, false positive, true negative and false negative. Table 6 shows the obtained confusion matrix.

- True positive (TP): When a classifier properly predicts the positive class from the provided test set, it is said to be true positive.
- True negative (TN): it demonstrates that, given the test data, the classifier model accurately forecasts the negative class. The accuracy of the classifier is indicated by the true positive and true negative numbers. These categories need to correspond with the true positive and true negative values, though.
- False positive (FP): indicates that the positive class was predicted by the classifier model in error.
- False negative (FN): indicates erroneously predicting the negative class by the classifier.

With the help of this confusion matrix, we measure the performance in terms of average accuracy, precision, recall, and F-Score. These parameters can be expressed as [25]:

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad (13)$$

$$\text{Recall} = \frac{TP}{TP+TN} \quad (14)$$

$$P = \frac{TP}{TP+FP} \quad (15)$$

$$F = \frac{2*P*Sensitivity}{P+Sensitivity} \quad (16)$$

Table 6. Confusion matrix

Actual class	Predicted class	
	Normal	Attack
Normal	True positive	False negative
Attack	False positive	True negative

4.2. Comparative analysis

With the help of aforementioned parameters, we measure the performance of proposed model and compare the obtained outcome with existing methods. Sun *et al.* [21] presented deep feature extraction model and implemented several classifiers. Table 7 shows the obtained performance. In order to achieve this performance, we have considered the combined dataset as discussed before where the final dataset is grouped into two classes as “attack” and “benign” query.

Table 7. Comparative analysis for ML and DL (class wise analysis)

Classifier	Class	Precision	Recall	F1-score	Accuracy
KNN	Attack	0.98	0.95	0.97	0.96
	Benign	0.95	0.98	0.90	
	Average	0.97	0.97	0.98	
RF	Attack	0.98	0.99	0.99	0.98
	Benign	0.99	0.98	0.98	
	Average	0.99	0.99	0.99	
LSTM	Attack	0.88	0.87	0.87	0.87
	Benign	0.87	0.88	0.87	
	Average	0.87	0.87	0.87	
Proposed SQL-CB-GuArd	Attack	0.99	0.99	0.99	0.99
	Benign	0.99	0.99	0.99	
	Average	0.99	0.99	0.99	

According to this experiment, the existing methods i.e. KNN, RF, and LSTM methods have used deep feature extraction model whereas proposed model uses combination of pre-processing along with GRU and attention mechanism to enhance the pattern learning process. Therefore, the average accuracy has been reported as 0.96, 0.98, 0.87, and 0.99 by using KNN, RF, LSTM, proposed SQL-CB-GuArd, respectively. Similarly, Fang *et al.* [22] presented Bi-LSTM based DL approach for SQLIA detection. Table 8 shows the comparative analysis as discussed in [22]. The complete dataset is divided into 70% training and remaining 30% is used for testing purpose.

Table 8. Comparative analysis for ML and DL (overall performance analysis)

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random forest	97.88	97.90	97.88	97.89
SVM	88.22	88.76	88.22	88.36
Logistic regression	71.02	73.49	71.02	70.57
MLP	90.99	91.49	90.99	91.15
Bi-LSTM	99.26	99.26	99.25	99.24
Proposed SQL-CB-GuArd	99.80	99.55	99.60	99.25

Further, we compared the performance with the traditional DL and supervised ML models to measure the performance of proposed approach. Table 9 shows the obtained performance in terms of accuracy, precision, recall, and F1-score. The experimental analysis shows that the traditional LSTM model reported the less accuracy as 62.32% whereas proposed SQL-CB-GuArd has reported highest classification accuracy as 99.20%.

Table 9. Comparative analysis for DL methods

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Autoencoder [14]	94.00	95.00	90.00	92.00
LSTM [10]	62.32	66.23	65.16	64.23
KNN [10]	82.69	71.51	88.56	79.13
DT [10]	92.33	89.58	89.74	89.66
SQLNN [10]	96.16	97.28	92.23	94.68
Proposed SQL-CB-GuArd	99.20	99.35	99.40	99.10

5. CONCLUSION

This articles discussed the threats to web based systems where SQLIA are considered as one of the most crucial threat to these systems. Therefore, maintaining security in these applications remains a challenging task. Several methods have been introduced to deal with this issue where ML based automated methods have gained attention from research community. However, ML based methods face several challenges which need to be addressed. Therefore, researchers have adopted DL based methods. In this work, we adopt the DL method to improve the SQLIA detection performance. The proposed model performs data pre-processing, vectorization and hybrid DL model to obtain the improved performance. The pre-processing phase includes data decoding, generalization and tokenization. Further, vectorization model is implemented to obtain the final feature vector. Finally, CNN, Bi-LSTM, and GRU models are employed. Further, attention mechanism is also incorporated to improve the feature learning performance.




REFERENCES

- [1] A. A. R. Farea *et al.*, "Injections attacks efficient and secure techniques based on bidirectional long short time memory model," *Computers, Materials and Continua*, vol. 76, no. 3, pp. 3605–3662, 2023, doi: 10.32604/cmc.2023.040121.
- [2] A. Kumar, S. Dutta, and P. Pranav, "Analysis of SQL injection attacks in the cloud and in WEB applications," *Security and Privacy*, vol. 7, no. 3, 2024, doi: 10.1002/spy2.370.
- [3] M. Qasaimeh, R. A. Hammour, M. B. Yassein, R. S. Al-Qassas, J. A. L. Torralbo, and D. Lizcano, "Advanced security testing using a cyber-attack forecasting model: A case study of financial institutions," *Journal of Software: Evolution and Process*, vol. 34, no. 11, 2022, doi: 10.1002/smr.2489.
- [4] S. Morgan, "2019 official annual cybercrime report," *Cybersecurity Ventures*, Herjavec Group, 2019.
- [5] L. Brew, L. Drazovich, and S. Wetzel, "The impact of COVID-19 on the security and resilience of the maritime transportation system," *Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience, CSR 2021*, pp. 510–517, 2021, doi: 10.1109/CSR51186.2021.9527935.
- [6] H. Tupsamudre, S. Jain, and S. Lodha, "PhishMatch: a layered approach for effective detection of phishing URLs," *arXiv-Computer Science*, pp. 1-30, 2021.
- [7] A. Arshad, A. U. Rehman, S. Javaid, T. M. Ali, J. A. Sheikh, and M. Azeem, "A systematic literature review on phishing and anti-phishing techniques," *Pakistan Journal of Engineering and Technology, PakJET*, vol. 4, no. 1, pp. 163-168, 2021.
- [8] S. Heister and K. Yuthas, "How blockchain and AI enable personal data privacy and support cybersecurity," in *Blockchain Potential in AI*, 2022, doi: 10.5772/intechopen.96999.
- [9] A. Dixit, J. Quaglietta, K. Nathan, L. Dias, and D. Nguyen, "Cybersecurity: guiding principles and risk management advice for healthcare boards, senior leaders and risk managers," *Healthcare quarterly*, vol. 25, no. 4, pp. 35–40, 2023, doi: 10.12927/hcq.2023.27019.
- [10] N. Khan, J. Abdullah, and A. S. Khan, "Defending malicious script attacks using machine learning classifiers," *Wireless Communications and Mobile Computing*, vol. 2017, 2017, doi: 10.1155/2017/5360472.
- [11] OWASP, "OWASP top ten: Top 10 web application security risks," OWASP, 2021. Accessed: Feb. 26, 2024. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [12] F. K. Alarfaj and N. A. Khan, "Enhancing the performance of SQL injection attack detection through probabilistic neural networks," *Applied Sciences*, vol. 13, no. 7, 2023, doi: 10.3390/app13074365.
- [13] N. Khan, J. Abdullah, and A. S. Khan, "Towards vulnerability prevention model for web browser using interceptor approach," *2015 9th International Conference on IT in Asia: Transforming Big Data into Knowledge, CITA 2015*, 2015, doi: 10.1109/CITA.2015.7349842.
- [14] V. Jain, M. S. Gaur, V. Laxmi, and M. Mosbah, "Detection of SQLite database vulnerabilities in android apps," *Information Systems Security (ICISS 2016)*, Springer, Cham, pp. 521–531, 2016, doi: 10.1007/978-3-319-49806-5_31.
- [15] M. McPhee, *Mastering kali Linux for web penetration testing*. Birmingham, United Kingdom: Packt Publishing Ltd, Jun. 2017.
- [16] D. E. Nofal and A. A. Amer, "SQL injection attacks detection and prevention based on neuro—fuzzy technique," *Studies in Big Data*, vol. 77, pp. 93–112, 2021, doi: 10.1007/978-3-030-59338-4_6.
- [17] N. M. Sheykhkanloo, "A learning-based neural network model for the detection and classification of SQL injection attacks," *Deep Learning and Neural Networks*, pp. 450–475, 2019, doi: 10.4018/978-1-7998-0414-7.ch026.
- [18] A. M. Vartouni, S. Mehralian, M. Teshnehlab, and S. S. Kashi, "Auto-encoder LSTM methods for anomaly-based web application firewall," *International Journal of Information and Communication Technology*, vol. 11, no. 3, pp. 49–56, 2019.
- [19] N. Thalji, A. Raza, M. S. Islam, N. A. Samee, and M. M. Jamjoom, "AE-Net: novel autoencoder-based deep features for SQL injection attack detection," *IEEE Access*, vol. 11, pp. 135507–135516, 2023, doi: 10.1109/ACCESS.2023.3337645.
- [20] N. Gandhi, J. Patel, R. Sisodiya, N. Doshi, and S. Mishra, "A CNN-BiLSTM based approach for detection of SQL injection attacks," *Proceedings of 2nd IEEE International Conference on Computational Intelligence and Knowledge Economy, ICCIKE 2021*, pp. 378–383, 2021, doi: 10.1109/ICCIKE51210.2021.9410675.
- [21] H. Sun, Y. Du, and Q. Li, "Deep learning-based detection technology for SQL injection research and implementation," *Applied Sciences*, vol. 13, no. 16, 2023, doi: 10.3390/app13169466.




- [22] Y. Fang, J. Peng, L. Liu, and C. Huang, "WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM," *ACM International Conference Proceeding Series*, pp. 170–174, 2018, doi: 10.1145/3199478.3199503.
- [23] M. Kherbache, K. Amroun, and D. Espes, "A new wrapper feature selection model for anomaly-based intrusion detection systems," *International Journal of Security and Networks*, vol. 17, no. 2, pp. 107–123, 2022, doi: 10.1504/IJSN.2022.123298.
- [24] I. Jemal, M. A. Haddar, O. Cheikhrouhou, and A. Mahfoudhi, "Performance evaluation of convolutional neural network for web security," *Computer Communications*, vol. 175, pp. 58–67, 2021, doi: 10.1016/j.comcom.2021.04.029.
- [25] F. Dass, M. Dass, C. Feresia, and M. Foozy, "A comparative study of SQL injection detection using machine learning approach," *Applied Information Technology And Computer Science*, vol. 3, no. 2, pp. 19–031, 2022, doi: 10.30880/aitcs.2022.03.02.002.

BIOGRAPHIES OF AUTHORS



Asif Iqbal Sirmulla    holds Masters Degree in Computer Science from Reva University, India in 2020. He also recieved his Bachelors from VTU, India in 2013. He is currently Research Scholar at Reva University, Bangalore. His research includes machine learning, data mining, and natural language processing. He has published 5 papers in international journals and conferences. He can be contacted at email: aisirmulla@gmail.com.



Dr. Prabhakar Manickam    is presently working as Professor in School of Computer Science & Engineering, REVA University – Bangalore with an aim of designing a project for farmers in India to cultivate appropriate crop in appropriate time for better yield. He has published about 15 patents and has obtained 5 patent grants in various fields including internet of things, image processing, education, and security in WSNs. He was awarded 'Excellence in Research', by Novel Research Academy, Puducherry, India. He has delivered many guest lectures on Internet of Things in 'Border Security Force Signal Training School (BSF-STs) for Officers' - Bangalore. He can be contacted at email: prabhakar.m@reva.edu.in.