Detecting SQL Injection Attack using Natural Language Processing

Sagar Lakhani
Information Technology
Indian Institute of Information Technology Allahabad
Prayagraj, India
sagarlakhani1999@gmail.com

Ashok Yadav
Information Technology
Indian Institute of Information Technology Allahabad
Prayagraj, India
rsi2021002@iiita.ac.in

Vrijendra Singh
Information Technology
Indian Institute of Information Technology Allahabad
Prayagraj, India
vrij@iiita.ac.in

***Abstract*—In today's digital era, online attacks are increasing in number and are becoming severe day by day, especially those related to web applications. The data accessible over the web persuades the attackers to dispatch new kinds of attacks. Serious exploration on web security has shown that the most hazardous attack that affects web security is the Structured Query Language Injection(SQLI). This attack addresses a genuine threat to web application security and a few examination works have been directed to defend against this attack by detecting it when it happens. Traditional methods like input validation and filtering, use of parameterized queries, etc. are not sufficient to counter these attacks as they rely solely on the implementation of the code hence factoring in the developer's skill-set which in turn gave rise to Machine Learning based solutions. In this study, we have proposed a novel approach that takes the help of Natural Language Processing(NLP) and uses BERT for feature extraction that is capable to adapt to SQLI variants and provides an accuracy of 97% with a false positive rate of 0.8% and a false negative rate of 5.8%.**

***Index Terms*—Structured Query Language Injection, Web Application Security, Machine Learning, Deep Learning, Natural Language Processing**

I. Introduction

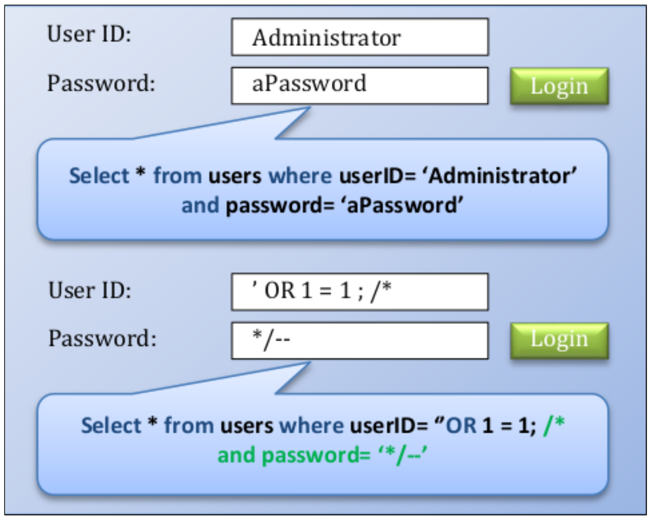
SQL injections have arisen as one of the most dangerous kinds of attacks on web applications and are positioned among the Open Web Application Security Project's (OWASP) top ten vulnerabilities [1]. SQL injection attacks(SQLIAs) are dispatched by entering malicious characters as input in web-based applications contributing to an altered SQL query. The SQLIA is thought of as the most dangerous attack among the injection class since it affects the primary security principles.

The three pillars of cyber security i.e the CIA triad are compromised by the SQLI attack. The CIA triad stands for confidentiality, integrity, and availability. Using SQLI, attackers can gain access to stored user data which directly affects confidentiality part. Also, the attacker can change the data by running data manipulation queries which will compromise the integrity of the data. Also, if the attacker runs a highly computative query, the performance of the service can be affected which is against the availability property of the triad.

During the beginning of the Internet, most websites were static, hence not vulnerable to injection attacks. The first huge SQLIA was performed in 2002 on Guess.com, the attackers gained access to customers' data which included: names, credit card numbers, and expiration dates, and affected around 2,00,000 users. According to the National Vulnerability Database(NVD) reports, SQL Injection comprises 3.26% of the total reported incidents in 2021 [2]. In addition to this in early 2021, around 70 GB of data was stolen from Gab.com through SQLI attack [3].

Nowadays, most web applications utilize a database to store information gathered from the clients as well as to recover data required by the clients. Interaction with these clients for the most part is accomplished through input forms and web cookies. Attackers attempt to take advantage of this interaction by infusing malicious code into these client inputs that will be utilized later to build the SQL queries. Improper validation of the client inputs can contribute to the success of the SQLI attack and, accordingly, can have harmful outcomes, for example, the attackers might access sensitive and confidential information of clients or may also delete the database.

Attackers may play with different input values to check for SQL vulnerabilities, and without proper input validation in place, it might give attackers access to confidential data. In the shown Fig. 1, the attacker has entered input values that might seem random at first but when combined with an SQL query in the back-end, results in a modified query that provides the attacker access to all the existing users(as 1=1 will always be true).



```
Select * from users where userID= 'Administrator'
and password= 'aPassword'
```

```
Select * from users where userID= "OR 1 = 1; /*
and password= '*/--'
```

Fig. 1. SQL Injection Query

The earlier researches in the field of SQL Injection detection were more inclined towards the prevention techniques which includes placing proper input validation and input filtering or the use of stored procedures. They are more of a developer skill dependent rather than solution effectiveness dependent. So, different machine learning based approaches were proposed to detect SQL Injection and hence decreasing the dependency on human skill. But the issue with most of the ML based approaches were that they had high false positive and false negative rate. Hence, there was a need for a model which had less false positive and false negative rates, and high accuracy.

II. Literature Review

A. Makiou et al. [4], presented a Hybrid approach to detect the SQLI attack. The technique proposed by them involved analyzing the HTTP request by dividing them into chunks and trying to detect anomalies in these chunks. The method proposed by the authors included communication between a detection engine and a machine learning classifier. The authors used two different Bayesian classifiers: Bayesian Naive and Bayesian Multinomial to differentiate between legitimate and illegitimate HTTP requests. The features are extracted using an HTTP dissector which is based upon given security rules. The extracted features are given as input to the above-mentioned classifiers. The model is evaluated by inspecting the curve of the true positive against the false positive. Despite the high accuracy of 97.6%, the above solution lacks the handling of false negatives and false positives.

T. K. George et al. [5], proposed a hybrid approach also known as Token-based Detection and Neural Network-based Reconstruction(TbD-NNbR). This framework comprises of a two-module system that is placed between the web application server and the database server on a proxy server. The first module or the TbD module focuses on token matching and matches the existing legal query tokens to input query tokens at runtime. If the runtime-generated dynamic token differs from the legal tokens, it implies an SQLI attack. The limitation to this module was that a large dataset is needed to store valid and legal tokens, otherwise, it can promote an increase in the number of false negatives. The solution to the above issue was provided by the authors by adding another module known as the reconstruction module or NNbR. The reconstruction module as the name suggests is used to reconstruct the tokenized SQL query for the authenticated users to avoid or minimize the denial of service. The malicious queries for authenticated users are reconstructed in such a way that the malicious part is either eliminated or set as null to generate a valid and legal SQL query. This method reduces the denial of service to authenticated users and promotes the availability of service. Despite all this, the model has high false positives which lead to denial of service, which has been solved only for authenticated users.

H. Zhang et al. [6], analyzed the real-time network traffic and performed SQLI detection on that traffic. The authors used feature extraction corresponding to a set of predefined features on the collected network traffic and later on trained the Deep-Belief-Network(DBN) to obtain the model. A comparative analysis was also performed by the authors to analyze the results corresponding to DBN, LSTM, Multi-layer perceptron(MLP), and CNN. Comparatively, DBN outperformed all the others with the highest accuracy of 96.03%. The only limitation to this model was the lack of a proper dataset of illegitimate SQL statements which can be used to train DBN.

Y. Fang et al. [7], made a breakthrough by using the concept of syntactical analysis for the tokenization of SQL queries. The tokens are stored in vectors which are later on used to form the LSTM model. The framework developed by the authors was tested on real-time data and the proposed framework attained the accuracy of 98.6% which was more in comparison to what was achieved from the random forest algorithm. Also, the framework is featureless and works on real-time data, hence can be used to check for 0-day vulnerabilities.

X. Xie et al. [8], proposed a modified version of CNN by integrating Elastic Pooling with it or Elastic Pooling CNN or EP-CNN. The authors have concentrated their research more on detecting the hidden common features of SQLI, so the proposed approach can detect new types of SQLI attacks as well rather than conventional methods such as SVM, Bayesian classifiers, etc. which need artificial features. It contains a pooling layer that helps in highlighting the concerning common features. The proposed approach proved to be effective against new SQLI attacks.

A. Ladole and M. Phalke [9], proposed an approach that will be helpful in the detection of SQLI vulnerabilities with the help of an automatic tool that will generate the test cases that exploit the SQLI vulnerabilities. It operates on the string-based equations which will be further matched with the attacking patterns. It is even capable to detect very small vulnerabilities which might be ignored by other processes. It uses a classification algorithm to train the vectors and put them in a file format and finds the vulnerabilities at the compile time and prevents the web applications from severe SQLI attacks.

Binh An Pham and Vinitha Hannah Subburaj [10], combined the traditional classifiers with advanced feature detection techniques using NLP. The authors used Word Level Term Frequency and Inverse Document Frequency Vectors to determine the relevancy of a term in documents according to the frequency. The term frequency represents the frequency of a word in a document, whereas Document Frequency is aimed to calculate the frequency of words in a set of documents. TF-IDF method assumes the words to be independent of each other and hence extracts the feature in a context-free way. They combined this feature extraction method with classifiers such as Naive Bayes, Logistic Regression, SVM, Random Forest, and Extreme Gradient Boosting.

Lu Yu et al. [11] and Bronjon Gogoi et al. [12], explored the usage of a context-free NLP model for feature extraction and then passed the processed information through an SVM classifier. In the initial step, the query is tokenized and passed through the skip-gram model of word2vec which converts the words into word embeddings represented via vectors. These word embeddings in the case of word2vec are context-free and acts as input for the classifier. The proposed approach is query-dependent rather than source code-dependent and has a negligible performance overhead. The authors of the former paper used an SVM classifier whereas the authors of the latter paper used other classifiers as well like Naive Bayes, Decision Tree, Random Forest, Ada Boost, and Gradient Boosting.

III. Dataset Analysis

The dataset comprises of two columns: Input and Label. The input column contains the input data provided by a user or a client, which will be later converted to SQL queries. Inputs can be of two types: Normal Text Inputs or Malicious Inputs. The Label column stores the label corresponding to legal and malicious inputs. Normal text inputs are labeled as 0 and malicious inputs are labeled as 1. The gathered dataset contains 11,332 malicious inputs corresponding to SQL Injection and 19,588 legal-normal text inputs.

From TABLE I, we can infer that the first 2 inputs are labeled as 1 and can lead to the exposure of confidential information. For example, the first input can generate the following query: SELECT * FROM card_details WHERE userId=” or ’1’ = ’1’. As ’1’ = ’1’ will always be true, it will return all the data in the card_details table whereas the 3rd input not only exposes confidential information but can also make the thread go to sleep for 5 seconds which can affect the availability of the thread for other processes.

TABLE I
Dataset Description

Inputs	**Labels**
dtype: Text	dtype: Numeric
’ or ’1’ = ’1	1
admin’or 1 = 1 or ” = ’	1
1’ or sleep (5) #	1
stacey@mediacar.la	0
8463810912	0

IV. Data Pre-Processing

As a first step towards processing the data, the records that have null or empty values either in input or output are dropped. Then after dropping NA values we move on to filter the unique records based on input values. The data is then shuffled as such to have randomness when dividing the dataset into training and testing. Then as the last step, the data is divided into a 4:1 ratio to get a training and testing dataset.

V. Proposed Methodology

Bi-Directional Encoder Representations from Transformers a.k.a BERT is a method that is used to train a model with contextual learning and hence understand and process textual information. It is pre-trained on a large corpus of text like Wikipedia and then used to perform standard NLP tasks like question-answering. It is a high-performing method as it is the first of its kind and is a bi-directional context-understanding method trained with unsupervised learning.

Representations that are pre-trained can be either contextual or context-free. Contextual representations can be sub-divided into uni-directional or bi-directional. Context-free representations generate a single and unique word embedding representation for every word in the glossary. So same word in different sentences will have the same word embedding representation for context-free models. But in the case of contextual models, each word’s representation is generated based on the remaining words of the sentence.

BERT was an improvement on the existing pre-trained contextual representations like Generative Pre-Training, ELMo, Semi-Supervised Sequence learning, and ULMFit as most of them are either uni-directional or shallowly-bidirectional meaning the embeddings of the word are created only using the words on the left or right. For example, representations of the word “factory” in the sentence “he is a factory worker” depend on “he is a” but do not depend on “worker” for a unidirectional model. Other existing bi-directional representation models were shallow but BERT is deeply bi-directional and depends on both left context and right context starting from the base of the deep neural network. So it uses both “he is a” and “worker” to create the word embedding representation for the word “factory”. In Fig. 2, the architecture of the neural network for BERT is shown compared to other models. The arrows in the figure indicate the flow of information between layers and The green boxes indicate the output contextualized representations of all the words.

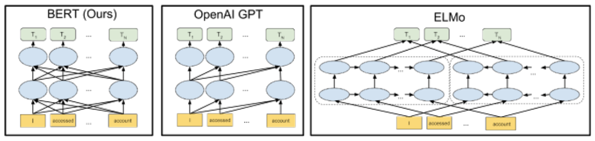


Fig. 2. BERT is deeply bidirectional, OpenAI GPT is unidirectional, and ELMo is shallowly bidirectional.

For contextual learning, BERT uses methods known as masking and next-sentence prediction and trains simultaneously on both. In masking, 15% of the total words in a sentence or an input is masked and then the masked sequence is run through a bi-directional transformer encoder and it tries to predict the masked words based on the other words. For the next-sentence prediction, BERT tries to predict whether two sentences are consecutive or not.

BERT consists of a 12 to 24 layer of encoders and is trained on large vocabularies such as Wikipedia and Brown Corpus for about a million update steps. There are two stages to training BERT to achieve the required task, first is pre-training, which is a very expensive one-time task for any language and can take up to 4 days on 4 to 16 TPUs. Generally, most of the BERT models come already pre-trained on large text corpus, hence giving the flexibility to move to the second stage of training which is Fine-Tuning. Fine-tuning is not as expensive as pre-training and can take up to some hours when trained on GPUs. In this step, BERT is trained on the available dataset corresponding to the given problem to fine-tune the results according to the problem.

So, for the provided problem of SQLI here BERT is fine-tuned on the given dataset. The proposed model uses the BERT transformer along with a classifier layer to classify the input into the 0 or 1 class corresponding to SQLI. From Fig. 3, we can understand the steps involved for the SQLI detection for Proposed Model. The model consists of 5 layers. The first layer is the Input layer which takes the whole text sentence as input. The second layer is the Keras layer which pre-processes these text sentence inputs by tokenizing them and adding padding and sentence beginners and ending tokens. The third layer is the BERT encoder itself which encodes the provided tokens and generates word embeddings. The fourth layer is a Dropout layer that drops and condenses the input which is followed by the last and final classifier layer which classifies the input as 0 or 1 depending on whether the input tests positive corresponding to SQLI or not.

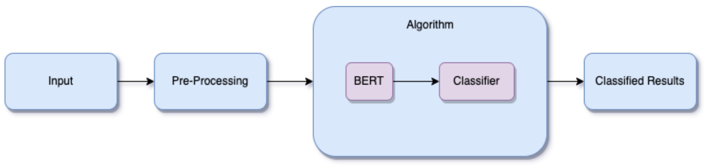


Fig. 3. Bert generates contextual word embedding matrix which is used by a classifier to generate output

Apart from the proposed model, we have also used other classifiers such as K-Nearest Neighbours, Support Vector Machine, Gaussian Naive Bayes, Decision Tree, and Logistic Regression and have also analyzed the performance of the proposed model against these classifiers in section VI.

VI. Results

Our main objective was to optimize false negative and false positive as they have a major contribution towards the precision and security of the system. In the ideal case, the false positive and false negative rates(FPR and FNR) should be less and other metrics such as Precision, Recall, Accuracy, F1-Score, and Specificity should be high.

From TABLE II, we can infer that the proposed model in which BERT was used, outperformed the traditional classifiers in terms of all the evaluation metrics. The results show that the proposed model had FPR and FNR as low as 0.8% and 5.8% and kept accuracy, precision, recall, f1-score, and specificity as high as 97%, 99%, 94%, 96%, and 99% respectively.

VII. Conclusion and Future Work

The SQL Injection attacks are considered one of the most dangerous injection attacks for web-based applications. SQL Injection vulnerability in the past is the reason behind many big attacks that have harmed the confidentiality, integrity, and availability of data and databases. Here, we have analyzed the results generated by different classification models/algorithms and have proposed a solution with the aim to maintain a low False Positive Rate(FPR) and low False Negative Rate(FNR) while keeping accuracy in check and producing state-of-the-art results. The proposed model takes the help of Natural Language Processing(NLP) and uses BERT for feature extraction providing the accuracy of 97% with FPR of 0.8% and FNR of 5.8%.

In the future, this research can be extended to use a combination of other NLP models like RoBERTa and XLnet for feature extraction and passing them through various available classifiers. Also, extending this research to cover other query languages like Hive Query Language(HQL) can be an

TABLE II
COMPARING THE RESULTS OF THE VARIOUS MODELS

Performance Evaluation Metrics							
Technique	**FPR**	**FNR**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Specificity**
K-Nearest Neighbours	0.320	0.193	0.73	0.62	0.81	0.70	0.68
Support Vector Machine	0.299	0.170	0.75	0.64	0.83	0.72	0.70
Gaussian Naive Bayes	0.261	0.067	0.81	0.70	0.93	0.80	0.74
Decision Tree	0.045	0.212	0.89	0.92	0.79	0.85	0.96
Logistic Regression	0.031	0.225	0.89	0.94	0.78	0.85	0.97
Proposed Model	**0.008**	**0.058**	**0.97**	**0.99**	**0.94**	**0.96**	**0.99**

interesting problem as with hive comes the factor of detecting the attack in a distributed environment.

REFERENCES

[1] Owasp top ten security vulnerabilities. *Online*, 2021. URL https://owasp.org/Top10/.

[2] National vulnerability database reports. *Online*, 2021. URL http://nvd.nist.gov.

[3] Far-right platform gab has been hacked—including private data. *Online*, 2021. URL https://www.wired.com/story/gab-hack-data-breach-ddosecrets/.

[4] A. Makiou, Y. Begriche, and A. Serhrouchni. Improving web application firewalls to detect advanced sql injection attacks. *10th International Conference on Information Assurance and Security*, page 35–40, 2014.

[5] T. K. George, K. P. Jacob, and R. K. James. Token based detection and neural network based reconstruction framework against code injection vulnerabilities. *Journal of Information Security and Applications*, 41:75–91, 2018.

[6] H. Zhang, J. Zhao, B. Zhao, X. Yan, H. Yuan, and F. Li. Sql injection detection based on deep belief network. *ACM International Conference Proceeding Series (ICPS)*, 2019.

[7] 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 (ICPS)*, pages 170–174, 2018.

[8] X. Xie, C. Ren, Y. Fu, J. Xu, and J. Guo. Sql injection detection for web applications based on elastic-pooling cnn. *IEEE Access*, 7:151475–151481, 2019.

[9] A. Ladole and M. Phalke. Sql injection attack and user behavior detection by using query tree, fisher score and svm classification. *International Research Journal of Engineering and Technology*, 3:1505–1509, 2016.

[10] Binh An Pham and Vinitha Hannah Subburaj. An experimental setup for detecting sqli attacks using machine learning algorithms. *Journal of The Colloquium for Information Systems Security Education*, 8, 2020.

[11] Lu Yu, Senlin Luo, and Limin Pan. Detecting sql injection attacks based on text analysis. *3rd International Conference on Computer Engineering, Information Science Application Technology (ICCIA)*, 90, 2019.

[12] Bronjon Gogoi, Tasiruddin Ahmed, and Arabinda Dutta. Defending against sql injection attacks in web applications using machine learning and natural language processing. *IEEE 18th India Council International Conference (INDICON)*, 2021.