Journal of Theoretical and Applied Information Technology

30th September 2023. Vol.101. No 18
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVING DETECTION SQL INJECTION ATTACKS USING REFERRER HEADER AND URI WEBLOG

ELFRIDA B.A SIAHAAN [1], ABBA SUGANDA GIRSANG [2]

[1,2]Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia

E-mail: [1]elfrida.siahaan@binus.ac.id, [2]agirsang@binus.edu

ABSTRACT

SQL Injection Detection provides the ability to monitor SQL Injection attacks on websites. Currently, researchers are using Deep Learning to detect SQL Injection. However, this detection has limitations, such as high False Positives (FP), False Negatives (FN), and low Accuracy due to the detection of SQL Injection using only URI data. At the same time, attacks do not only occur through URIs but also Referrers. Therefore, this study aims to use the combination of URI and Referrer to detect attacks and discuss the increase in model performance due to adding a Referrer. This study's first step was preprocessing the dataset and then vectorizing it using Word2Vec. The Word2Vec and CNN method was proposed using the combination of URI and Referrer, then compared to Word2Vec CNN using URI. The experimental results show that the proposed method performs better than other methods and gets accuracy over 99% of the payloads with a low error rate.

Keywords: *CNN, Referrer, SQL Injection, Web Log, Word2Vec*

1. INTRODUCTION

The increasing adoption of the Internet in Information Technology advancements is driven by its ability to provide fast and convenient access, enabling efficient task execution. Year by year, the utilization of the Internet has been steadily growing. The International Telecommunication Union (ITU) estimates that Internet users will reach 66% of the world's population, or around 5.3 billion people, by 2022 [1]. With the increasing number of Internet users, there is also a growth in the availability of online services. The Internet is not just a media service provider looking for information and entertainment but also a service provider for businesses online. Companies leverage web-based applications as a platform to enhance and support their online business operations.

Web-based applications use a web browser and server connected to the Internet. The web uses the database to store the user data on the backend. Web-connected to database using Structured Query Language (SQL). Once users input data in a web application, the information is integrated into an SQL query and transmitted to a database. This situation allows attackers to manipulate the data entered through a web application form or parameter by exploiting vulnerabilities. This vulnerability can arise due to a lack of implementation of secure programming code, logic errors, the weak input validation function against SQL injection, and the placement of server variables on the URL. As a result, attackers can create malicious SQL queries that attack databases. Consequently, sensitive information stored in the database becomes vulnerable to exposure. Furthermore, this presents a substantial risk as it grants unauthorized individuals the ability to access, manipulate, and delete data and illicitly commandeer databases or website applications.

According to OWASP TOP 10 2021, SQL Injection attacks are ranked as the third most common attack method used against websites. However, in both 2013 and 2017, SQL Injection attacks held the top position [2]. This demonstrates the significant importance and potential harm associated with such attacks.

Researchers and experts have developed various methodologies and techniques to prevent the execution of SQL injection on the database actively. One of them is implementing secure programming code and applying validation functions on the web page. This function will block special characters that attackers commonly used to create SQL Injection.

Besides making validation functions, others research using pattern-matching techniques with string-matching algorithms to detect pattern matches of known attacks on the system [3], [4]. Still, the Accuracy of detection is highly dependent on the coverage of the knowledge base. It is difficult to detect new types of SQL injection attacks.

With a different approach, past researchers detected and prevented SQL attacks using machine learning techniques. Feature for detection of SQL Injection must be in vector representation. The conventional approach to feature engineering, such as the Bag of Words method, involves manually extracting features from text using techniques like TF, TF-IDF, and n-grams to convert words into numerical representations. This is followed by the classification of attacks or normal behavior using machine learning algorithms like Support Vector Machine (SVM) [5], [6] and Ensemble Machine Learning [7]. Deep learning techniques like Multi-Layer Perceptron (MLP) [8], [9] can also be employed. However, Bag of Words has a weakness: the need for an expert to design features to be used and unable to provide information related to the semantics, structure, order, and context around the words in each document. Bag of Words involves each word individually and cannot capture the semantic relationship between words. Reversing the order of words, it will still yield the same vector value.

The weakness of Bag of Words is its inability to capture the semantic relationship between words, encouraging previous researchers to try research using word embedding, which can capture word semantic relationships. The word embedding algorithm allows word position to be learned based on the words around it so that it can grasp the semantic and syntactic meaning of the word. Apart from capturing semantic and syntactic meanings, automated extraction of word embedding features can be accomplished. One of the Word embedding techniques is the Word2vec technique [10]. Word2Vec is a natural language processing (NLP) algorithm used to produce word vector representations of text. Word2Vec generates vector representations of words from the text. This approach involves converting the features into vectors and then using traditional machine learning algorithms for classification, such as SVM [11], [12], Light GBM, and Cat Boost [13], and deep learning algorithms, such as CNN [14] to classify them as either attacks or normal.

Other studies with Word2vec and machine learning or deep learning algorithms for SQL Injection detection were tested on real websites using weblogs. The research focuses on extracting the Uniform Resource Identifier (URI) portion of the W3C extended log format from weblogs using TF-IDF and detecting potential SQL injection attacks using supervised learning [15]. Another research uses URI as features combined with deep learning CNN as a classification algorithm [14]. The Accuracy achieved in this research is quite good However, using only URI to detect SQL injection with actual data is insufficient because SQL injection attacks can also be carried out on the Referrer section, which is vulnerable to SQL Injection [16].

This study aims to employ Word2Vec as a feature extraction method and CNN as a classification technique to detect the class of sqli or normal based on access logs and to comprehend how URI along with Referrer impact the enhancement of the SQL injection classification model. Word2Vec is chosen because based on [17]research, Word2Vec provides better results in classifying SQL injection compared to Glove and Fast Text. Meanwhile, CNN is selected due to research by [18], which compared various machine learning algorithms and found that CNN achieved the highest accuracy. During the model evaluation, two datasets will be used. The first dataset consists of URIs extracted from real access logs, similar to previous studies [14]. The second dataset, the proposed approach involves combining URI and Referrer extracted from real access logs. This is motivated by [19] research, which highlights that Referrer is also vulnerable to SQL Injection. By adding the referrer, it can reduce false negatives and false positives while performing SQL injection detection on weblogs.

The testing dataset used in this study originates from the private access logs of the Ministry. The model's performance will be evaluated using a confusion matrix, ACC, Precision_score, Recall_score and F1_score.

2. LITERATURE REVIEW

This section reviews the related theories and previous research involved in the Detection of SQL Injection.

2.1 Related Theories

The approach used in this study uses data from weblogs, namely the URI and Referrer sections, then converted into vector representations using Word2vec and applying prediction to detect attacks or non-attacks with the Convolutional Neural Network (CNN) algorithm.

When a user accesses a web resource, the web server generates a weblog to record the relevant information. Web logs contain information about the IP addresses, authentication information, user name, date and time of web access, request containing the type of HTTP request used such as GET, POST, and HEAD and URI accessed by the user, the HTTP status code provided by the server in response to a user's request, file or page size transferred in bytes, Referrer or page that directs the user to the current page, User-agent or the software and operating system used by the user.

Web logs have three formats: Common Log Format, Combined Log Format, and W3C Extended Log Format. The Common Log Format records basic information about client access, including client IP address, user identity, date and time of entry, HTTP method, pages accessed, HTTP status code, and file size accessed and combined Log Format, which is similar to Common Log Format but records additional information related to user access, such as referrers and user agents. The W3C Extended Log Format records more detailed information about client access, such as information about the client's browser, the type of file requested, and the type of server used. This study uses a weblog from a government agency that implements the Common Log Format.

URI is a unique text or characters string used to identify resources on the Internet. URIs represent the address of an Internet-accessible resource, for example, web pages, images, videos, or documents. URI consists of several main components. The first one is a schema that indicates the protocol used for accessing the resource. Standard protocols are HTTP, HTTPS, FTP, and files. The second one is authorities, which show the location of the resource being accessed. Authority can be a domain name, port, or IP address. The three Paths are the paths that indicate the location of the resource. The fourth component refers to a string query that starts with a question mark and contains pairs of keys and values.

Referrer is the information a web browser sends to a web server when a user accesses a web page. Referrer Information contains the URI of the previous web page, which redirects the user to the current web page. Attackers can send SQL Injection via Referrer if the website does not validate the values sent through the Referrer. An attacker can change the Referrer value and try to manipulate the SQL queries. The attacker can use the Referrer by injecting SQL injection syntax into the URI on the Referrer, which will then be processed by the server and integrated into the SQL query.

A data preprocessing stage is required to extract URIs and Referrers from weblogs. Humans can understand this data, but machines cannot understand and perform analytical processes when the input data is presented as a string, thus requiring word embedding. Word embedding is an unsupervised learning using Neural Networks to convert words into a vector representation. The Word2vec algorithm is one type of word embedding created by Miko Lov et al. in 2013 [20]. Word2vec maps each text word into a vector using hidden and fully connected layers. Word2vec makes words with the same context have the same embedding vector. The word vector is extracted using the weight matrix on the trained model's hidden layer. During the training process, model Word2Vec will optimize this weight. The weight will be updated using the backpropagation algorithm based on the difference between the output generated by the model and the actual value (ground truth).

CNN is a type of deep learning algorithm that uses convolution layers. In the convolution layer, a filter or kernel is gradually shifted, thus being able to extract essential features automatically [21]. Then proceed with max-pooling to select the most prominent features for prediction. It ended with text classification based on previous features [22]. CNN is often used for various applications such as image classification [23], [24], facial recognition [25], and natural language processing [26], [27].

2.2 Previous Research

Researchers have repeatedly researched the identification of SQL Injection attacks. Study [28] uses a machine learning model with 23 classification algorithms, using 616 private datasets for training and testing the model. The top 5 algorithms based on Accuracy are Ensemble Boosted, Bagged Trees, Linear Discriminant, Cubic SVM, and Fine Gaussian SVM. The obtained Accuracy is 93.8%.

Other research performs detection by analyzing actual weblogs. Web servers make weblogs available by default, making them widely utilized. The research proposes features selected based on access behaviour mining and grammar pattern recognizer. Then using, machine learning with five algorithms for training, namely: Naïve Bayesian, Random Forest, SVM, ID3, and K-means. Evaluation is done by measuring the False Positive Rate (FPR) and False Negative Rate (FNR). ID3 and Random Forest are better able to detect SQL Injection with a low FNR of 7.7%. But this paper has not measured the model's accuracy [29].

According to research [13], the data contained in the HTTP headers of web logs, such as User-Agent, Pragma, and Cookies, does not have a significant impact on differentiating between regular requests and attacks, so it only uses the Method Host, Path, and Parameters fields. The top Accuracy achieved is 99.36%. Other research measures Accuracy by taking URI, a combination of cs -Uri stems and cs-Uri-query from weblogs, n-grams used for extracting features, and TF-IDF for vectorization and classification using machine learning. Experiments achieved the top Accuracy is 98.56% [15]. This Accuracy can still be improved by using Word2Vec to extract features from URIs and classification using the CNN algorithm [14]. This method obtains an accuracy of 99.50. However, in reality, attacks recorded in weblogs are not limited to being executed through the URI but also through Referrers [16]. Using only URI can reduce Accuracy and increase the FNR value. Therefore, this research proposes combining URIs and referrers from actual weblogs as testing datasets while training datasets using public data.

3. PROPOSED METHODOLOGY

This paper proposes a method for SQL Injection detection using Word2vec for vectorization and CNN for classification on URI and Referrer weblogs. The approach consists of 5 stages: data collection, preprocessing, word embedding process, architectural model design, and evaluation. Here is *Figure 1* illustrates the proposed model design.

3.1 Data Collection

The author collects data in two ways, and the first is collecting private data from a government website by accessing the website on the production server and taking weblogs in .txt form. Then this weblog will be classified as an attack and normal SQL into a spreadsheet. An internal team from the government conducts the data labelling process. The data is stored in two spreadsheet files, one for normal and one for attack data. This private dataset will undergo preprocessing to extract the payload.

Meanwhile, the second method we considered is to acquire publicly available widely distributed datasets, accessible through the following link: https://github.com/skykami/detect_traffic_sqlias_CNN/tree/master/data. The numbering of normal data is 34.500 and 35.393 for attack data. Separate files store normal data and attack data.

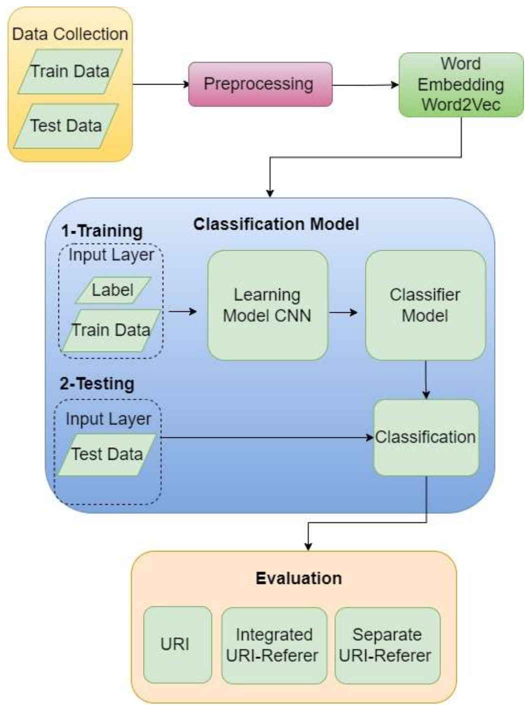

Figure 1: The Proposed Model Design

3.2 Preprocessing

The preprocessing stage involves converting the private data into the payload format. Preprocessing will process various data into more regular data so it can be applied to machine learning algorithms. The private dataset is the original dataset that contains complete information such as GET, POST, and PUT request data. Researchers need to process this data by removing irrelevant information for their study. Here Figure 2 illustrates preprocessing stage.

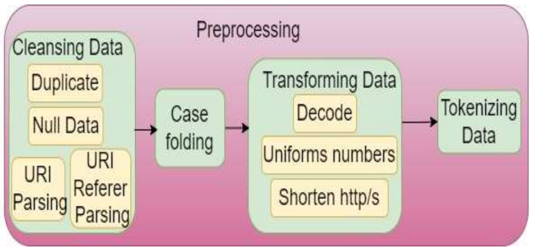

Figure 2: Preprocessing Stage

Several steps are involved in the preprocessing stage: data cleansing, case folding, data transforming, shortening the http/s format, and data tokenization. Data transformation consists of data decoding and digit uniformity.

3.2.1 Data cleansing

Cleansing data aims to remove unnecessary data to detect SQL injection. The original file obtained from the server in .txt format is converted into a spreadsheet format. The .txt file is read, and then each line is truncated using a space (space as a delimiter) to store data elements in 10 fields on the spreadsheet. The next step involves checking for duplicate data by verifying URI fields and Referrer fields. The deletion of duplicate data will continue the result. Afterward, null data checking is performed to check for empty data, then continued with Parsing URI and Referrer. This Parsing aims to take part URI and Referrer, which consists of 2 steps: taking the combination of Referrer and URI field or just taking URI field, and then string splitting process. This process is executed on both the SQL Injection file and the normal file.

The first stage of Data Cleansing for the dataset URI Referrer is taking the valid Referrer and URI fields. URI is in column 5, and Referrer is in column 8. The checking process will be applied to columns 5 and 8 of each row. If both fields contain the character dash "-", the row is not taken as a dataset. If fields have a request to fetch the root document of a website with the following syntax GET / HTTP/1.1, those rows are also not taken as a dataset. If the field starts with OPTIONS and HEAD, they are also not taken as a dataset because neither of them is a request that can send or manipulate data on the server. Besides the conditions above, columns 5 and 8 are considered valid and will be merged with the format “fifth column | eighth column” and saved into a .txt file. This file will be used as a dataset. The same step is taken for dataset URI, the extraction process for the URI field focuses specifically on column 5, following the same approach as previously mentioned.

The next step is the string-splitting process by applying a regular expression. The purpose of string splitting is to extract the vital part of the payload. The first regular expression is used to find text enclosed between the first and second spaces in URI Referrer, and the second regex is used to find text that comes after the vertical bar symbol (“|"). The results of the two regexes are combined and saved into a new normal file or attack file with a .txt format. For URI fields, the string splitting only applied the first regular expression, which aims to find text enclosed by the first set of characters.

3.2.2 Case folding

The process of case folding involves converting all capital letters to lowercase letters. This action is taken because uppercase and lowercase do not affect the Detection of SQL injection attacks.

3.2.3 Data transformation

After complete case folding, decode the URI and Referrer, which aim to encode data to their original form characters. The decoding process converts special characters, such as the % symbol, and certain characters, such as spaces, punctuation marks, and others, back into original characters and converts each digit into its actual number.

The following stage is numerical uniformity; all numbers are uniform to 0. This process is carried out because numbers do not determine whether an attack is or not. However, it will increase the number of data types, thus increasing the complexity of the data. Therefore, it is necessary to do the uniformity of numbers.

The result of numerical uniformity will be processed with shortened http/s. The regular expression will be used to find the format http or https:// followed by letters or digits or punctuation marks “.”, “@”, “&”, “/”, “#”, “!”, “?” and “-“, then it will be changed to http:u.

3.2.4 Data tokenization

Data tokenization aims to separate the payload into smaller word parts. Regular Expression using to perform tokenization. Then creating vocabularies of Word2Vec use the result of tokenization based on the unique words obtained from the payload.

Several rules are used, namely: if the payload contains characters consisting of one or more letters, numbers, underscores, and dots followed by open bracket characters equal to, larger than, or smaller, then it will be considered as one word. Suppose the payload contains characters consisting of one or more letters, numbers, underscores, and dots followed by a closing bracket character. In that case, it will be made into two words: a group of characters and the closing bracket itself.

If the double quotation mark symbol (“) is followed by a character consisting of one or more letters, numbers, underscores, and dots, then followed by simply a quotation mark (“), it is considered as one word. Furthermore, when it comes to certain symbols such as single quotation marks ('), greater than symbols (>), and smaller than symbols (<), followed by a sequence of characters and then the symbol itself, it should be treated as a single word according to the same rule. The final result of the preprocessing process is an array, a collection of tokens from each SQL payload data.

3.3 Word Embedding Word2Vec

Word embedding or vectorization aims to make a model representing words as vectors. This action is performed since classification using CNN requires input as a vector. This research conducts word embedding using the Word2vec technique by utilizing the Gensim library.

The parameters used in this research are a 16-dimensional size vector, which means that one word is represented in a 16-dimensional vector. They used Window 5, the number of words considered as the context of a target word during the training process. Min count 10 means the model will ignore words with a word frequency of less than 10. The last parameter is ten iterations.

It should be emphasized that if a word used in the payload is not present in the vocabulary of Word2Vec, its corresponding vector representation will be unavailable. To address this, it becomes necessary to assign vectors to out-of-vocabulary words. This research will utilize a zero vector with 16 dimensions for words not found in the vocabulary.

3.4 Classification Model

Before classification and after the normal and attack datasets are converted as vectors, each vectorized payload line will be labeled “|0” or “|1”. Normal will be labeled 0, and attack will be labeled 1. Labeling is done to assign a label to each row of payload and then use it to train the CNN model. Each payload row will be randomized, enabling the model to learn using labels from data without a specific order.

The classification model uses the Keras library with a one-dimensional CNN method. CNN is a feed-forward neural network with a convolution structure. The data is partitioned into training, evaluation, and testing data to facilitate the training of the CNN model. If the model's performance during training and testing is not much different, then it can be said that it has worked well. The CNN model architecture employed in this research is depicted in Figure 3. CNN comprises the input, convolutional, max pooling, fully connected, and output layers.

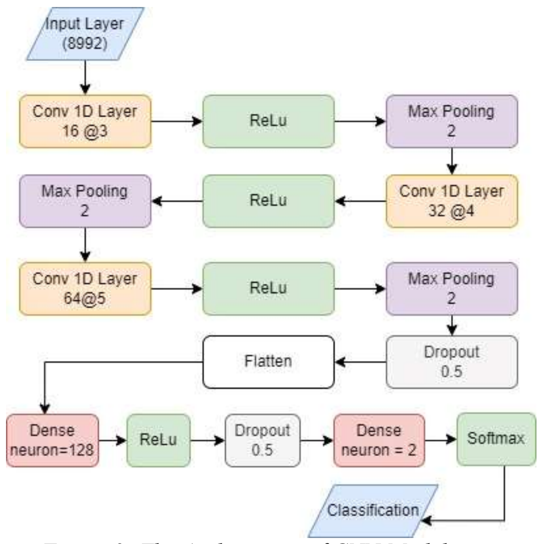

Figure 3: The Architecture of CNN Model

3.4.1 Input layer

The input data used for CNN must have the same size. To achieve uniformity in size, examining each payload and identifying the maximum number of words present is essential. The maximum token for all payloads is 544; each token will be represented in a 16-dimensional vector. All payloads will be set to a fixed length, namely 544 token. If the payload has a length less than 544, it is padded with zero vector at the end of the payload. The vectors present in the input layer have dimension n*m where n is the maximum number of payloads, which is 544 words, and m is 16 dimensions so the input will be 544 x 16 = 8704dimensions.

3.4.2 Convolutional layer

The convolution layer aims to extract features from the input data. Architecture CNN model uses three convolution layers. The first convolution layer has 16 filters with a kernel size of 3. The kernel will move vertically along the entire input matrix with stride one and padding the same. It means the output of this layer has the same size as the input. The second convolution layer has 32 filters and a kernel size of 4. The last is a convolutional layer with 64 filters and a kernel size of 5.

The filter and input matrix weights are multiplied and added together during this convolutional operation. The initial value of the filter weights was set randomly. The result of the convolutional operation will be processed using a non-linear process, the ReLU activation function. The final result of the convolution process will proceed to the max pooling layer.

3.4.3 Max pooling layer

The max pooling layer reduces the features produced by the convolution layer. However, they still retain essential information. This can reduce the time required for training and speed up the input data processing. The main aim is to extract crucial features that effectively capture the inherent characteristics of each feature map.

This layer will take the maximum value as a feature based on a filter. Data is summarised by sliding the window across feature maps and then taking the highest value from each window. Each window utilized for the pooling process has a length referred to as the pool size. The pool size used in this research is 2. The results of this max pooling will be used as input to the next convolutional layer.

3.4.4 Fully connected layer

Each feature will be connected to every neuron in this layer and proceed with the Softmax Activation function. During training, loss calculations are carried out using the Categorical Cross entropy loss function and Adam Optimizer to minimize loss values. The model has been trained with batch 100 batches with 10 and 20 epochs.

When training a neural network, an important trick that needs to be applied to prevent overfitting is the dropout regulation technique [30], with a rate of 0.5. It will randomly deactivate 50% of neurons during the training phase.

3.5 Evaluation

Evaluation is performed to provide an overview of model performance using a test dataset. The performance of the CNN model is evaluated using two different datasets. The first dataset is URI, and the second combines URI and Referrer. There are two ways to assess the performance of the model Combination URI and Referrer during testing. First, Integrated URI-Referrer combines URI and Referrer into one row, then predicts the label. Second Separate URI-Referrer makes the URI and Referrer into one row, but predictions are made separately. The URI and Referrer will be predicted independently. The results of the two payloads are combined using OR function. In other words, if the URI or Referrer has an attack label, the entire row is considered an attack.

Accuracy (ACC), Precision, Recall, and F1-score metrics assess the model's performance. The True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values from the confusion matrix are required to compute the performance model. Evaluation begins by loading the CNN model using Keras, then applied to the testing dataset.

4. RESULT

This stage focuses on presenting and analyzing the results obtained from the research implementation process.

4.1 Results of Data Collection

Two groups were obtained during the data collection phase: public data from GitHub and private data from the government's web server. The public is used for training and validation datasets. For training, we use 34.500 normal data and 35.393 attack data; for the validation, we use 1.074 normal data and 1.079 attack data. So, the total public data is 35.574 for normal and 36.472 for attack data.

Meanwhile, we used a private dataset with 1.001 normal data and 1.026 attack data for the testing dataset. So, the number of data used in this research is 36.575 for normal data and 37.498 for attack data.

4.2 Preprocessing Result

The collected public and private data will be preprocessed initially. Subsequently, the outcomes of each preprocessing step will be elucidated.

4.2.1 Data cleansing result

The private dataset used in this research comes from actual weblogs that must be cleaned first. There is a decrease in the number of datasets after data cleansing. Initially, the private dataset consisted of 1.001 normal datasets and 1.026 attack datasets; reduced, it decreased to 880 normal datasets and 1018 attack datasets.

The decrease in the number occurred due to duplicated data, and happened because URI and Referrer fields have empty data or only contain "-". The number of empty URI and Referrer is 1 for normal files and 0 for attack files. Furthermore, some data have OPTIONS, HEAD, and document root requests. In a normal file, there are 22 data containing OPTIONS requests, 1 data containing HEAD requests, and 8 GET / HTTP/1.1 requests. Meanwhile, there are 2 HEAD requests for the attack file, while OPTIONS requests and GET / HTTP/1.1 were not found. An example of data cleansing results can be seen in Table 1.

Table 1: The Example of Data Cleansing Results

Type	Data Before	Data After
URI	103.41.110.2 - - [27/Sep/2022:03:03:14 +0700] "GET /mobile/plugin/SyncUserInfo.jsp?userIdentifiers=-1)union(select(3),null,null,null,null,null,str(98989*44313),null HTTP/1.1" 404 465 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36"	GET /mobile/plugin/SyncUserInfo.jsp?userIdentifiers=-1)union(select(3),null,null,null,null,null,str(98989*44313),null HTTP/1.1
URI | Referer	180.252.164.62 - - [23/Sep/2022:06:49:01 +0700] "GET /p2kh/themes/kota-hijau/assets/fonts/AbadiMT.ttf HTTP/1.1" 200 70396 "http://sim.ciptakarya.pu.go.id/p2kh/combine/a2721a80ea08696164426a38fe3ad88c-1489131767" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36"	GET /p2kh/themes/kota-hijau/assets/fonts/AbadiMT.ttf HTTP/1.1|http://sim.ciptakarya.pu.go.id/p2kh/combine/a2721a80ea08696164426a38fe3ad88c-1489131767

After that, the string-splitting process is performed. An example of string-splitting results can be seen in Table 2.

Table 2: The Example of String Splitting Results

Type	Data Before	Data After
URI	GET /mobile/plugin/SyncUserInfo.jsp?userIdentifiers=-1)union(select(3),null,null,null,null,null,str(98989*44313),null HTTP/1.1	/mobile/plugin/SyncUserInfo.jsp?userIdentifiers=-1)union(select(3),null,null,null,null,null,str(98989*44313),null
Integrated URI | Referer	GET /p2kh/themes/kota-hijau/assets/fonts/AbadiMT.ttf HTTP/1.1|http://sim.ciptakarya.pu.go.id/p2kh/combine/a2721a80ea08696164426a38fe3ad88c-1489131767	/p2kh/themes/kota-hijau/assets/fonts/AbadiMT.ttf http://sim.ciptakarya.pu.go.id/p2kh/combine/a2721a80ea08696164426a38fe3ad88c-1489131767
Separated URI|Referer	GET /p2kh/themes/kota-hijau/assets/fonts/AbadiMT.ttf HTTP/1.1|http://sim.ciptakarya.pu.go.id/p2kh/combine/a2721a80ea08696164426a38fe3ad88c-1489131767	/p2kh/themes/kota-hijau/assets/fonts/AbadiMT.ttf |http://sim.ciptakarya.pu.go.id/p2kh/combine/a2721a80ea08696164426a38fe3ad88c-1489131767

Table 3: The Number of Dataset Training and Labels

Type	Training	Validation	Number Of Data
Normal	26.277	329	26.606
Attack	25.237	1.069	26.306

For testing the model, 880 data were used as normal data and 1018 as attack data. The following is

Table *4* detailing the distribution of testing data.

Table 4: The number of dataset testing and labels

Type	Testing Using URI	Testing Using Combination URI Referrer	
	URI	URI	URI Referrer
Normal	880	481	399
Attack	1018	736	282

4.2.2 Case folding and data transformation results

Unlike the previous process, this process will not change the number of datasets. An example of data resulting from case folding and data transformation can be seen in Table 5.

Table 5: The Example of Case Folding and Data Transformation Results

Process	Data Before	Data After
Case Folding	1%27%29%20AND%207800%3DBENCHMARK%285000000%2CMD5%280x45746c54%29%29	1%27%29%20and%207800%3dbenchmark%285000000%2cmd5%280x45746c54%29%29
Decode Data	action=http%3A%2F%2Fyeseyingyuan.com%2Fcdn-cgi%2Fnexp%2Fdok3v%3D1613a3a185%2Fcloudflare%2Fmirage2.js	action=http://yeseyingyuan.com/cdn-cgi/nexp/dok3v=1613a3a185/cloudflare/mirage2.js
Number uniformity	do.php?ac=71ee30ae117cddace55bd01714904227') limit 1,1 union all select null, null, null, null, null#	do.php?ac=0ee0ae0cddace0bd0') limit 0,0 union all select null, null, null, null, null#
Shorten http/s	"action=http://www.weipaifuliw.com/wp-content/uploads/2016/07/p407390-390.jpg)"	http:u

4.2.3 Tokenization results

This process is similar to the case folding and data transformation processes, which will not change the number of data sets. Examples of data tokenization results can be seen in Table 6.

Table 6: The Example of Tokenization Result

Data Before	Data After
/wp-login.php?action=http%3A%2F%2Fimg1.voc.com.cn%2FUpLoadFile%2F2017%2F03%2F23%2F201703231119566280.jpg	'wp', 'login.php', 'action=', 'http://u'
action=http://yeseyingyuan.com/cdn-cgi/nexp/dok3v=1613a3a185/cloudflare/mirage2.js	'action=', 'http://u', 'cgi', 'nexp', 'dok0v=', '0a0a0', 'cloudflare', 'mirage0.js'

4.3 Result of Word Embedding Using Word2Vec

Training word2vec is performed using the CBOW learning algorithm with ten epochs. The input data as text is applied to the Word2Vec process. There are two types of Word2Vec utilized: Word2Vec URI, which employs public data and URI from private data, and Word2Vec URI Referrer, which utilizes public data and a combination of URI and Referrer from private data. The resulting Word2vec model has a total vocabulary of 3.875 for combining URI Referrer, while the word model that takes URI data has a vocabulary of 3.835. Each word will be represented in 16 vector dimensions. The result of Word Embedding Using Word2Vec can be seen in Table 7.

4.4 Result Classification Model

This research compares the classification results using URIs and classification combining URIs and Referrers to determine the effect of using Referrers in SQL injection detection. Training is carried out with two types of iterations, namely 10 and 20 iterations. This is done to obtain the best parameters for the CNN model. After that, each training will be validated by a validation dataset.

Primarily, the investigation followed a similar approach to the prior study, utilizing publicly available data for Word2Vec and CNN, resulting in a validation Accuracy of 99.50%.[14]. The Wor2Vec model for this research will use public data, while for data testing, this research will use the private dataset from historical real weblogs. The preprocessing steps follow the previous research, involving the extraction of the URI from the weblogs for both testing and training data. The Word2Vec model acquires a vocabulary totaling 3,759 words, and the CNN model yields a loss = 0.0412, Accuracy = 0.9970, Val_loss = 0.0857, and Val_accuracy = 0.9950 same as what the previous researcher obtained.

Table 7: The Example of Training Model Word2Vec Results with URI

Type	Word	Vector Representation
Word2Vec URI	from	[3.98665 4.2678933 1.6760635 -1.6735022 2.9098976 4.8966675 1.5360914 -1.741722 0.08413549 1.9114813 -0.898581 -5.2906427 -1.4511316 3.6183403 -4.1795454 -0.04517118]
	union	[1.0412706 4.8464556 1.4313778 -0.21667795 2.6466954 5.047411 0.4776745 0.8614029 0.6197214 -2.1690965 -0.23389685 -4.654726 -3.60034 -1.5399555 -0.8271352 0.52281314]

ord2Vec URI Referrer	from	[-0.7363116 -0.38865307 0.43608987 5.1157866 1.0185812 -6.804795 1.4842553 4.847557 - 4.677706 2.301901 2.3717258 0.509522261.4842553 4.847557 -4.677706 2.301901 2.3717258 0.50952226 0.7298955 4.3305016 1.7133666 1.8403624]
	union	[1.2376004 1.4601108 - 1.27879 -0.07862712 2.3637328 -2.8802357 2.170729 2.350196 - 1.731591 -1.6046394 0.30371594 2.2505908 -1.7874576 6.260636 1.9012779 1.5958465]

Then the model is tested with a private dataset, the model get high False Negatives, specifically reaching a count of 36. The confusion matrix and performance model using public and private datasets as testing data can be seen in Table 8 and Table 9.

Table 8: Confusion Matrix Using Public Dataset for Training and Private Dataset for Testing

		Prediction	
		Normal	Attack
Actual	Normal	876 (TP)	4 (FP)
	Attack	36 (FN)	982 (TN)

Accuracy, Precision, Recall, and F1-Score value are obtained from the Confusion Matrix above. The values of performance can be seen in Table 9.

Table 9: Value of Performance Model with Public Dataset for Training and Private Dataset for Testing

Accuracy	Precision	Recall	F1-Score
97.89	99.59	96.46	98.00

The testing results show that model performance experienced a decreased accuracy from the validation accuracy. This can be attributed to the fact that the model struggled to categorize genuine web log data containing attacks via Referrer. Hence, this study utilized a combination of URI and Referrer as the data to be processed in Word2Vec and used as the testing dataset for the CNN model.

The research was continued by conducting experiments by adding private datasets to train data Word2Vec, so data public and data private are used for train Word2Vec. The experiment was carried out with 2 types of experiments, first model Word2Vec train by using data public and URI from private dataset, then continued with the vectorization process, developing the CNN model (the data was segregated for training and testing purposes) and then testing with URI from the private dataset. Second, data Word2Vec was taken from the data public and URI and Referrer on the private data then continued with the vectorization process, making CNN model and testing with URI and Referrer from the private dataset. Results for training the model using a combination of the public dataset and URI from the private dataset are shown in Table 10. In constructing Word2Vec, this research utilizes distinct parameters in contrast to the previous study. Specifically, it employs min_count 3 and epoch 30. Furthermore, the CNN hyperparameters also diverge from the prior research, involving a dropout rate of 0.1 and the utilization of the nadam optimizer.

Table 10: Results of Training Model Using Combination of Public Dataset and URI from Private Dataset

Iteration	Loss	Accuracy	Val_Loss	Val_Acc
10	0.0463	0.9972	0.0932	0.9950
20	0.0484	0.9962	0.1875	0.9800

The best model with URI can be measured by val_loss and val_accuracy values obtained for each iteration. In the 10th iteration, the model has better val_loss and val_accuracy than the 20th iteration. Therefore, at the 10th iteration, the model can be considered the best. The graph in Figure 4 illustrates the training model with 10 and 20 iterations on the URI dataset.

Besides that, results obtained from training the model using a combination of the public dataset and URIs Referrers from the private dataset, are shown in

Table ***11***.

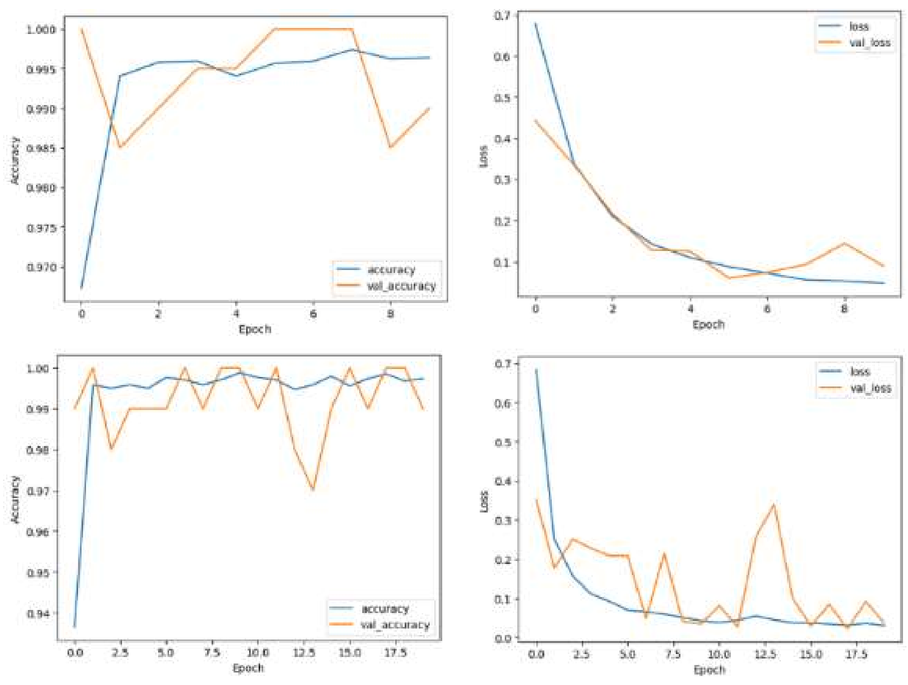

Figure 4: Graph Results of Training Model using a combination of Public Dataset and URI from Private Dataset

.

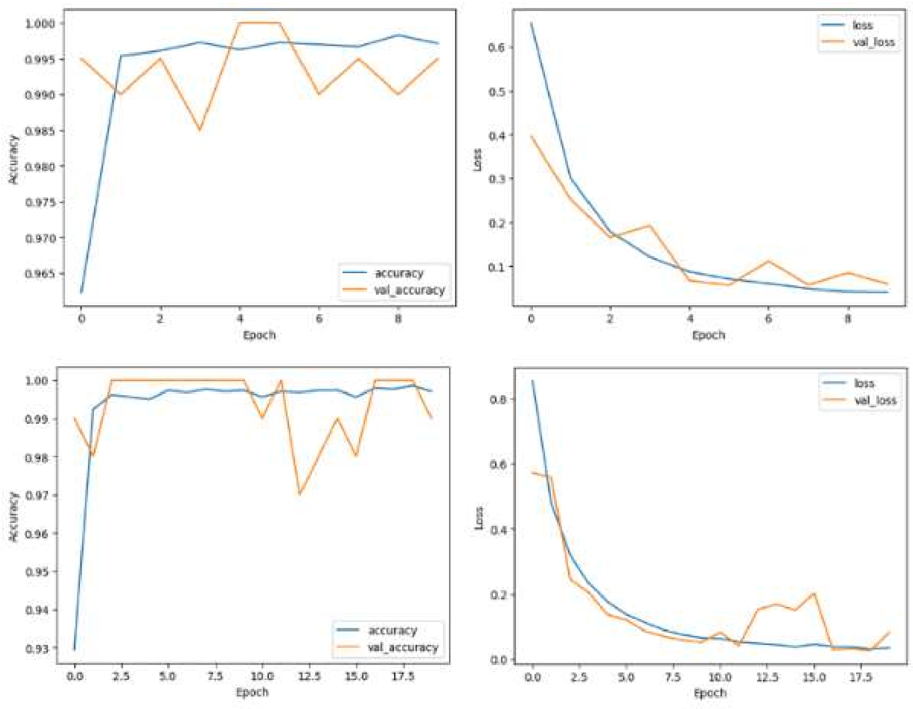

Figure 5: Graph Results of Training Model using a combination of Public Dataset and URI Referrer from Private Dataset

Table 11: Results of Training Model Using A Combination of Public Dataset and URIs Referrers from Private Dataset

Iter	Loss	Accuracy	Val_Loss	Val_Acc
10	0.0405	0.9971	0.0592	0.9950
20	0.0352	0.9971	0.0831	0.9900

The best model with data combining URI and Referrer is obtained in iteration 10, where val_loss and val_accuracy is better than in iteration 20. The graph in Figure 5 illustrates the training model with 10 and 20 iterations on the URI Referrer dataset.

Based on this data and figure, the CNN model, where the Word2Vec model is formed using public data and private data (extracting both URI and Referrer or only extracting URI), attains the same validation accuracy value as the previous research's model, which exclusively employed Word2Vec with URI extraction. Both models obtain an accuracy value above 99%. The next stage needs to be carried out, namely evaluation using testing data to measure which model is the best one.

4.5 Evaluation Results

The evaluation stage is conducted to assess the model's performance using the testing dataset. Good performance on the validation step does not always guarantee that the model can work well on actual data. Therefore, the model needs to be evaluated with the testing dataset to ensure that the constructed model is indeed good. The following presents model evaluation results using the confusion matrix and measured using Accuracy, Precision, Recall, and F1-Score. The research involves two models, both of which will be evaluated using 2 testing datasets. The first dataset contains private weblogs with only extracted URIs, while the second dataset contains private weblogs with both extracted URIs and Referrers. Then, the models' evaluations will be compared with the evaluation results from previous research.

4.5.1 Performance CNN model trained using word2vec on datasets consisting of public and private URIs.

Confusion matrix for the CNN model, where its word2vec embeddings are derived from public data and URIs private data, and the model is evaluated using private URI data. The results for this model, with iterations of 10 and 20, are presented in Table 12 Below.

Table 12: Confusion Matrix for Model URI

Iter			Prediction	
			Normal	Attack
10	Actual	Normal	877 (TP)	3 (FP)
		Attack	37 (FN)	981 (TN)
20		Normal	876 (TP)	4 (FP)
		Attack	36 (FN)	982 (TN)

Based on Table 12, Result, Accuracy, Precision, Recall, and F1-score values were obtained, as shown in Table 13.

Table 13: Performance Model For Testing Using Dataset URI

Iteration	Loss	Accuracy	Val_Loss	Val_Acc
10	97.89	99.69	96.36	98.00
20	97.89	99.59	96.46	98.00

Based on the data above, the best model uses ten iterations. The model achieved 3 False Positives and 37 False Negatives, with 877 True Positives and 981 True Negatives. Meanwhile, the Performance measurement resulted in an Accuracy of 97.89%, 99.69% Precision score, 96.36% Recall score, and 98.00% F1_Score. The results of this model will be used as a baseline to measure whether combining URIs and Referrers provides a better model.

4.5.2 Performance CNN model trained using word2vec on datasets consisting of public and private URI and Referrer.

This research experimented with detecting SQL Injection using URI Referrer in two approaches. The first approach is Integrated URI Referrer, where datasets URI and Referrer are combined and predicted whether it is an SQL Injection or not. The second approach is a Separated URI Referrer. The prediction process for URI and Referrer is performed separately. URI prediction results and Referrer prediction results are combined using OR. Integrated URI Referrer and Separated URI Referrer using 10 and 20 iterations. Table 14 explains the results of the confusion matrix and performance model with an Integrated URI Referrer. In contrast, Table 16 presented the confusion matrix and performance model results with a Separated URI Referrer.

Table 14: Confusion Matrix For Testing Using The Integrated URI Referrer

Iter			Prediction	
			Normal	Attack
10	Actual	Normal	878 (TP)	2 (FP)
		Attack	36 (FN)	982 (TN)
20		Normal	877 (TP)	3 (FP)
		Attack	37 (FN)	981 (TN)

The Accuracy, precision, recall, and F1-score values for Integrated URI Referrer are presented in Table 15.

Table 15: Performance Model For Testing Using Dataset The Integrated URI Referrer Model

Iter	Accuracy	Precision	Recall	F1-Score
10	97.99	99.79	96.46	98.10
20	97.89	99.69	96.36	98.00

The best model is an Integrated URI Referrer using ten iterations. The model achieved 2 False Positives and 36 False Negatives, with 878 True Positives and 982 True Negatives. Meanwhile, the Performance measurement resulted in an Accuracy of 97.99%, 99.79% Precision score, 96.46% Recall score, and 98.10% F1_Score.

The model's performance when the testing data is created using the URI and Referrer format, and subsequently utilized for testing, it can only achieve an accuracy of 97.99%. This can be attributed to the way the combination of URI and Referrer during the testing phase affects the context of words and subsequently impacts vector values of word. Consequently, the input vector data provided to the CNN model for prediction becomes imprecise. Comparatively, the CNN model that employs Word2Vec, trained on a mix of public dataset and private URIs, also demonstrates a similar accuracy level, showing no significant improvement. This is because the attack that occurred did not take place through the URI.

Hence, when conducting tests, it becomes crucial to incorporate the Referrer and independently predict the URI and Referrer, and then consolidate these predictions using an "or" operation. This process aims to derive the predictive value for an individual weblog entry. The confusion matrix and performance model Separated URI Referrer can be seen in Table 16 and Table 17.

Table 16: Confusion Matrix For Testing Using The Separated URI Referrer Model

Iter			Prediction	
			Normal	Attack
10	Actual	Normal	878 (TP)	**2 (FP)**
		Attack	**5 (FN)**	1013 (TN)
20		Normal	877 (TP)	3 (FP)
		Attack	6 (FN)	1012 (TN)

The Accuracy, precision, recall, and F1-score values for the Separated URI Referrer Model are presented in Table 17.

Table 17: Performance Model For Testing Using The Separated URI Referrer Model

Iter	Accuracy	Precision	Recall	F1-Score
10	**99.63**	99.80	96.46	98.10
20	97.89	99.69	96.36	98.00

The best results for the model using Separated URI Referrer were obtained in 10 iterations with 2 False Positives, 5 False Negatives, 878 True Positives, and 1013 True Negatives. Meanwhile, the Performance measurement resulted in an Accuracy of 99.63%, 99,80% Precision score, 96,46% Recall score dan 98.10% F1_Score. Comparing this model with the Integrated URI Referrer model, this model yields much better results. Similarly, when comparing it to the model URI, this model can get the best result.

The result of the experiment in this study is further compared with existing methods in literature and has proven to be an efficient method that can be adopted by researchers for further investigations and improvements. During the testing phase, this model demonstrates improved accuracy when working with the weblog dataset. This model achieved and validation accuracy score of 99.50, but during the testing phase, it reached 99.63, while the previous study[14] that only used URIs managed to attain 97.89. This demonstrates that employing both URI and Referrer can enhance the detection of SQL Injection.

However, the data extraction procedure takes more time due to the preprocessing phase that includes retrieving the payload URI and payload Referrer. Additionally, while making predictions, the data is forecasted twice, once for the payload URI and once for the payload Referrer. Subsequently, these predictions are combined in a final step using the OR operation.

Additionally, we also added the interesting thing. The combination of applying Natural Language Processing (NLP) with Word2Vec and utilizing Convolutional Neural Networks (CNN) illustrates how this technology can generate an efficient approach to counter SQL injection attacks. We added the utilization of information from the Referrer header in this approach, which further emphasizes the importance of extracting data from the Referrer header in security analysis. In the future, this research could be enhanced by developing prediction methods to avoid the need for dual predictions, along with combining prediction outcomes using the OR.

5. CONCLUSION

The use of URI and Referrer in weblogs for conducting SQL injection classification can enhance the model's performance. The approach employed in this study leads to improved accuracy in classifying SQL injection attacks through the analysis of weblogs. What distinguishes our research is the utilization of preprocessing techniques that extract URI and Referrer from the weblogs, coupled with the integration of both URI and Referrer using or in evaluating the model. Our findings demonstrate that combining URI and Referrer to detect SQL injection based on weblog data can enhance the model's accuracy.

This technique enhances accuracy by 2% and lowered False Positives from 37 to 2, False Negatives from 5 to 2, with 878 True Positives and 1013 True Negatives. The performance measurement yielded an Accuracy of 99.63%, Precision score of 99.80%, Recall score of 96.46%, and F1_Score of 98.10%. The proposed method in this research, using both URI and Referrer, managed to reduce the number of False Negatives. However, when attacks occur through Referrer, the previous study failed to handle them. Therefore, this study successfully validates the hypothesis that combining URI with Referrer contributes to improving the model's performance in detecting SQL injection.

REFERENCES:

[1] "Two-thirds of the world's population uses the Internet, but 2.7 billion people remain offline," *www.itu.int*, 2023. https://www.itu.int/itu-d/reports/statistics/2022/11/24/ff22-internet-use/ (accessed May 18, 2023).

[2] "OWASP Top Ten," *www.owasp.org*. https://owasp.org/www-project-top-ten/ (accessed May 19, 2023).

[3] A. Z. M. Saleh, N. A. Rozali, A. G. Buja, K. A. Jalil, F. H. M. Ali, and T. F. A. Rahman, "A Method for Web Application Vulnerabilities Detection by Using Boyer-Moore String Matching Algorithm," in *Procedia Computer Science*, Elsevier, 2015, pp. 112–121. doi: 10.1016/j.procs.2015.12.111.

[4] O. C. Abikoye, A. Abubakar, A. H. Dokoro, O. N. Akande, and A. A. Kayode, "A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm," *EURASIP J Inf Secur*, vol. 2020, no. 1, Dec. 2020, doi: 10.1186/s13635-020-00113-y.

[5] Y. Li and B. Zhang, "Detection of SQL Injection Attacks Based on Improved TFIDF Algorithm," *J Phys Conf Ser*, vol. 1395, no. 1, p. 012013, 2019, doi: 10.1088/1742-6596/1395/1/012013.

[6] M. Radi, "PREDICTING SQL QUERY QUALITY USING MACHINE LEARNING TECHNIQUES," *J Theor Appl Inf Technol*, vol. 15, no. 9, 2023, [Online]. Available: www.jatit.org

[7] U. Farooq, "Ensemble Machine Learning Approaches for Detection of SQL Injection Attack," *Tehnički glasnik*, vol. 15, pp. 112–120, Mar. 2021, doi: 10.31803/tg-20210205101347.

[8] P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, "Detection of SQL injection based on artificial neural network," *Knowl Based Syst*, vol. 190, Feb. 2020, doi: 10.1016/j.knosys.2020.105528.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Jan. 2013, [Online]. Available: http://arxiv.org/abs/1301.3781

[10] V. Vargas-Calderón and J. E. Camargo, "Characterization of citizens using word2vec and latent topic analysis in a large set of tweets," *Cities*, vol. 92, pp. 187–196, 2019, doi: https://doi.org/10.1016/j.cities.2019.03.019.

[11] L. Yu, S. Luo, and L. Pan, "Detecting SQL Injection Attacks based on Text Analysis," 2019.

[12] Z. Chen, M. Guo, and L. Zhou, "Research on SQL injection detection technology based on SVM", doi: 10.1051/matecconf/2018173.

[13] J. Li, H. Zhang, and Z. Wei, "The Weighted Word2vec Paragraph Vectors for Anomaly Detection over HTTP Traffic," *IEEE Access*, vol. 8, pp. 141787–141798, 2020, doi: 10.1109/ACCESS.2020.3013849.

[14] A. Luo, W. Huang, and W. Fan, "A CNN-based Approach to the Detection of SQL Injection Attacks," in *2019 IEEE/ACIS 18th International Conference on Computer and Information Science*

(ICIS), 2019, pp. 320–324. doi: 10.1109/ICIS46139.2019.8940196.

[15] X. D. Hoang, "Detecting Common Web Attacks Based on Machine Learning Using Web Log," in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 311–318. doi: 10.1007/978-3-030-64719-3_35.

[16] N. M. Thang, "Improving Efficiency of Web Application Firewall to Detect Code Injection Attacks with Random Forest Method and Analysis Attributes HTTP Request," *Programming and Computer Software*, vol. 46, no. 5, pp. 351–361, 2020, doi: 10.1134/S0361768820050072.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," Oct. 2013, [Online]. Available: http://arxiv.org/abs/1310.4546

[18] T. K. Sajja and H. K. Kalluri, "A deep learning method for prediction of cardiovascular disease using convolutional neural network," *Revue d'Intelligence Artificielle*, vol. 34, no. 5, pp. 601–606, Oct. 2020, doi: 10.18280/ria.340510.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Aug. 2016, pp. 1135–1144. doi: 10.1145/2939672.2939778.

[20] P. Sumari *et al.*, "A Precision Agricultural Application: Manggis Fruit Classification Using Hybrid Deep Learning," *Revue D Intelligence Artificielle*, vol. 35, pp. 375–381, Oct. 2021, doi: 10.18280/ria.350503.

[21] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving Deep Convolutional Neural Networks for Image Classification," Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.10741

[22] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," *Ecol Inform*, vol. 48, pp. 257–268, Nov. 2018, doi: 10.1016/j.ecoinf.2018.10.002.

[23] Y. Zhang, Z. Zhang, D. Miao, and J. Wang, "Three-way enhanced convolutional neural networks for sentence-level sentiment classification," *Inf Sci (N Y)*, vol. 477, pp. 55–64, Mar. 2019, doi: 10.1016/j.ins.2018.10.030.

[24] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.01271

[25] 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, pp. 1–6. doi: 10.1109/ICECTA48151.2019.8959617.

[26] H. Gao, J. Zhu, L. Liu, J. Xu, Y. Wu, and A. Liu, "Detecting SQL injection attacks using grammar pattern recognition and access behavior mining," in *Proceedings - IEEE International Conference on Energy Internet, ICEI 2019*, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 493–498. doi: 10.1109/ICEI.2019.00093.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," 2014.