A Framework for Detecting SQL Injection Attack using Recurrent Neural Networks

S.Saranya.J[1] , M.Barath Kumaran[2], V.Gowtham[3], J.Mukthar Shakir[4],S.Senthamizhan[5]
Department of Computer Science and Engineering and Technology,
[1] Head of the Department , RAAK College of Engineering and Technology, Pondicherry, Pin-605010,India
[2,3,4,5] Student , RAAK College of Engineering and Technology, Pondicherry, Pin-605010,India

Abstract- *OWASP has identified SQL Injection as the primary security threat among the top 10. SQL Injection attacks can have severe consequences, such as data breaches and website failure. To address this issue, an adaptive deep forest-based approach has been developed to identify complex SQL Injection attacks. The deep forest structure is optimized, and the input for each layer comprises the raw feature vector and the average of prior outputs to combat the problem of degrading features with increased layers. The AdaBoost algorithm-based deep forest model leverages error rates to update the feature weights on each layer. This model automatically adjusts the tree model structure and manages multi-dimensional, fine-grained features, effectively avoiding overfitting issues. The proposed system aims to achieve improved results by implementing Recurrent Neural Networks (RNN).*

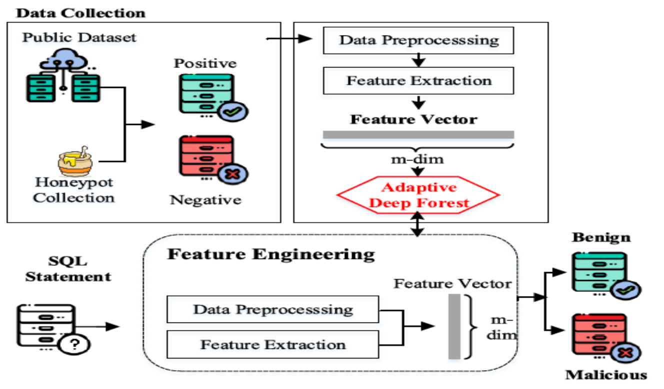
Keywords: SQL injection attack, website analysis, OWASP Top 10 , AdaBoost algorithm and Recurrent Neural Networks.

I. INTRODUCTION

The OWASP has identified SQL Injection as the top security threat among the ten that exist. A SQL injection attack is executed by injecting a SQL query into the application through the input data from the client. A successful exploit of SQL injection can result in sensitive data breaches, alteration of database data, and execution of administration operations on the database. SQL injection is often observed when a user is requested to provide input, such as a username or user id, and instead of providing this, the user submits an SQL statement which is then executed on the database. SQL injection attacks typically fall under three categories: In-band SQLi, Inferential SQLi (Blind), and Out-of-band SQLi. The In-band SQLi is the most common type of SQLi attack, where the attacker uses the same communication channel to launch attacks and gather results. Boolean, a sub-variation of In-band SQLi, prompts the application to return a result based on whether the query submitted is true or false. Several machine learning techniques have been employed for the detection of code injection attacks. The most used machine learning methods and preprocessing stages were identified. A classification-based approach using an ensemble algorithm was proposed to detect the level of SQL injection attacks in web applications. This method can identify new attacks based on their irregular matching characteristics, and it is more difficult to bypass.

II. EXISTING WORK

The current system implements an adaptive deep forest-based approach to detect complex SQL injection attacks. The structure of the deep forest is first optimized by concatenating the raw feature vector and average of previous outputs in each layer. However, the original features of deep forests tend to degrade with the increasing number of layers, which is addressed in this model. The AdaBoost algorithm is utilized based on the deep forest model to update the weights of features on each layer using error rate. This results in an adaptive deep forest model (ADF) that improves feature performance by transforming features based on multi-grained scanning, followed by layer-by-layer characterization learning using a cascade structure. Compared to deep neural networks, the ADF offers advantages in hyper-parameter settings and performance. AdaBoost, a Boosting technique used as an Ensemble Method in Machine Learning, re-assigns weights to each instance, with higher weights assigned to incorrectly classified instances. A forest may be less interpretable than a single decision tree, as it may require significant memory for storage due to the need for retaining information from several hundred individual trees.



III. PROPOSED SYSTEM

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. Recurrent Neural Networks enable you to model time-dependent and sequential data problems, like stock exchange prediction, artificial intelligence and text generation. Models under the Recurrent Neural Network are: Long Short Term Memory (LSTM) Gated Recurrent Unit (GRU) Long Short Term Memory (LSTM) is a kind of recurrent neural network (RNN) design applied in the deep learning field. LSTM has a feedback connection that is unrelated to standard feed forward neural networks. It cannot only process some data points but also entire sequences of information. The LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The cell recollects values over arbitrary time breaks and therefore the three gates control the data flow into and out of the cell. LSTM networks are compatible for categorizing, handling and producing guess supported statistical data, subsequently there are often delays of unidentified period between main events during time sequences. LSTMs were established to overcome the discharging and disappearing gradient problems which will be come across when training traditional RNNs. The LSTM unit contains a memory cell that has three gates described below: Input gate (i): The input gate computes the sum of input that is allowed to pass through it and is calculated by $i = \sigma (x_t U^i + s_{t-1} W^i)$ The sigmoid function plots the input value between [0, 1] and this value is multiplied by the weight vector (Ui). This helps the gate manage the quantity of input that is transferred through the input gate. Forget gate (f): The forget gate helps the network to choose what and how much information from the earlier level to transfer to the succeeding level. The sigmoid function maps the value of this function between 0 and 1. It is given by: $f = \sigma (x_t U^f + s_{t-1} W^f)$ If no input wants to be transferred to the next level, the previous memory is multiplied with the zero vector, which creates the input value zero. Likewise, if the memory at s_{t-1} needs to pass to the next level it is multiplied by 1 vector. If only some part of the input is to be passed, then the resultant vector is multiplied with the input vector. Output gate (o): The output gate, describes the output passed at each step of the network. It is given by: $o = \sigma (x_t U^o + s_{t-1} W^o)$ BiLSTM means bidirectional LSTM, which means the signal transmits backward as well as forward in time. Gaterecurrent unit GRU is a type of deep learning algorithm that is enhanced from the LSTM algorithm to minimize the complication of the algorithm by using update gate and reset gate. The update gate is used to regulate hidden state volume to be forwarded to the next state. The reset gate is used to define the consequence of the previous hidden state information. Update Gate (z): It determines how much of the past information needs to be passed along into the future. It is similar to the Output Gate in an LSTM recurrent unit. $z = \sigma (x_t U^z + s_{t-1} W^z)$ Reset Gate (r): It defines how much of the past information to forget. It is similar to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit. $r = \sigma (x_t U^i + s_{t-1} W^r)$ BiGRU means Bidirectional GRU's are a kind of bidirectional recurrent neural networks that allow for the use of information from both previous time steps and later time steps to make predictions about the current state.

PYTHON

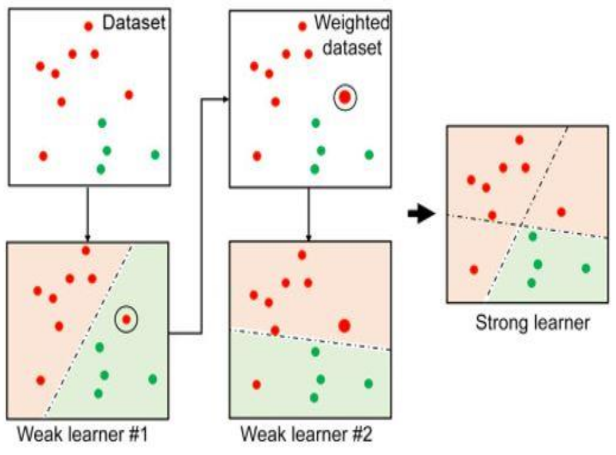
Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Advantages

- RNN can process inputs of any length.
- An RNN model is modelled to remember each information throughout the time which is very helpful in any time series predictor.
- Even if the input size is larger, the model size does not increase.
- The weights can be shared across the time steps.
- RNN can use their internal memory for processing the arbitrary series of inputs which is not the case with feedforward neural networks.

IV. ALGORITHM

Adaboost, also known as Adaptive Boosting, is a popular algorithm used in machine learning for classification problems. It works by combining multiple weak learners to create a strong classifier, with each weak learner trained on a subset of the data. Adaboost's main strength lies in its versatility, as it can be used with a wide range of weak learners, including decision trees, neural networks, and support vector machines. This flexibility makes Adaboost a popular choice for many different types of classification problems. However, one of the biggest challenges with Adaboost is its sensitivity to outliers, as they can significantly impact the algorithm's performance. In contrast, recurrent neural networks (RNNs) are a type of neural network that can process input data in a sequential manner due to their feedback loops. RNNs are particularly useful for tasks such as speech recognition and natural language processing, where the order of the input data is important. They can also remember the context of previous inputs and use this information to make better predictions, making them effective for tasks that involve time series data or sequences. However, RNNs have their own limitations, including the vanishing gradient problem, where the gradients used to update the network's weights become very small, making it difficult to train the network effectively. In summary, Adaboost and recurrent neural networks are powerful tools in the machine learning toolbox. While Adaboost is versatile and can be used with many weak learners, it is sensitive to outliers. On the other hand, RNNs are useful for sequential data but face challenges with the vanishing gradient problem. Understanding the strengths and limitations of these algorithms can help us choose the right one for the task at hand and improve our ability to create accurate and effective models.



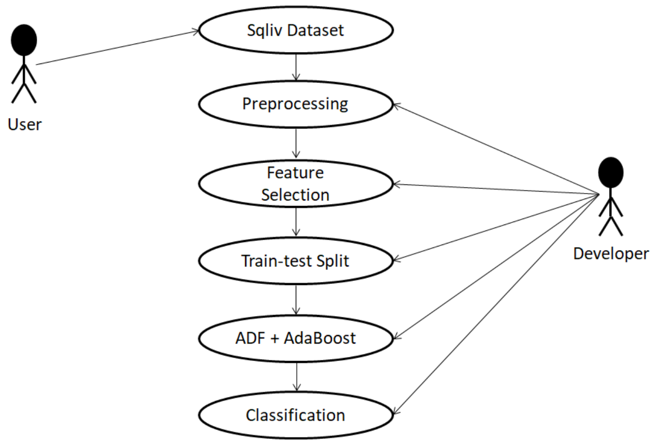
V. REQUIREMENT ANALYSIS

In terms of hardware requirements, an Intel Core i3 processor is needed along with 8 GB of RAM and a 64 GB hard disk. As for the software requirements, Google Colaboratory serves as the integrated development environment (IDE), while the operating system required is Windows 10. The coding language utilized is Python 3.7.5, and the libraries necessary for the project include Tensorflow 2.9.2 and Keras 2.2.4. These hardware and software specifications are essential for the proper functioning and execution of the project

VI. SYSTEM DESIGN.

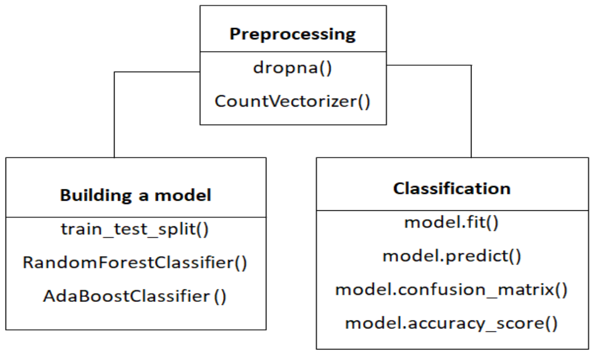
USE CASE DIAGRAM

A use case diagram is a powerful tool for modeling the dynamic behavior of a system. It captures the essence of a system by illustrating the use cases, actors, and their interrelationships. In essence, it represents the system's functionality and the roles played by different actors in executing various tasks, services, and functions. It provides a high-level view of a system's functionality, enabling stakeholders to visualize and understand how users interact with the system. A use case diagram is an essential aspect of software development as it helps to identify the user's needs and requirements, thereby enabling developers to design and implement systems that meet those needs.



CLASS DIAGRAM

The class diagram provides a static representation of an application, depicting the objects and their relationships within the system. It defines the various classes in the system and how they relate to each other. Each class is represented by its objects, and it may also inherit from other classes. The class diagram is utilized to visualize, document, and describe different aspects of the system, and can even be used to construct executable code. This type of diagram displays the attributes, functions, classes, and relationships among the objects, providing a comprehensive overview of the software system.



COLLABORATION DIAGRAM

The collaboration diagram is a type of diagram used in software design to visualize the relationships and interactions between objects in a system. Unlike sequence diagrams, which show the flow of messages between objects over time, collaboration diagrams illustrate the static architecture of objects in an object-oriented programming system. Objects in the system are depicted as boxes, and their features, such as attributes and methods, are shown within the boxes. The relationships between objects are represented by lines connecting the boxes. The collaboration diagram is also referred to as a communication diagram, as it shows how objects in a system communicate and collaborate with each other.

VII. MODULES

PRE-PROCESSING

Pre-processing refers to the preparation and transformation of raw data into a format that can be used to train machine learning and deep learning models. This involves cleaning and filtering out unwanted characters, stemming, and converting the text into a numerical representation using techniques like TF-IDF vectorization. Proper pre-processing of data improves the accuracy and effectiveness of machine learning models.

NORMALIZATION

Normalization is an essential data pre-processing step in machine learning that involves transforming the values of numerical columns into a standard scale. This is necessary to ensure that each feature is treated equally and to avoid any issues that may arise from varying scales. Standard Scaler and MinMax Scaler are two widely used normalization techniques. Standard Scaler scales the data to have a mean of zero and a standard deviation of one, while MinMaxScaler scales the data to a range between zero and one. By applying normalization to the data, machine learning models can perform better and produce more accurate results.

TF-IDF VECTORIZER

The Tfidf Vectorizer is a tool that creates a sparse matrix of word occurrence frequencies, using an in-memory vocabulary to map popular words to feature indices. In machine/deep learning, the dataset is typically split into a training set and a test set. The training set is used to refine the model, while the test set is used to evaluate the model's predictions on data that has not been seen before.

HYPERPARAMETER SETTINGS

.In the convolutional neural network, the activation function selects the most significant feature from the convolved features' word vectors. Different filters with varying windows are combined by the activation function. Nonlinear functions such as ReLU, sigmoid or tanh are used to compress vector values within defined ranges. Commonly used optimizers like SGD, Adagrad, and Adam are used to reduce the model's error rate and improve accuracy. Dropout, which reduces complexity between links in the fully connected dense layer, is a user-defined variable with an input range of 0 to 1. The number of epochs, or repetitions of a training approach, is determined by the training data. Max pooling is a pooling technique that selects the most prominent feature from the filter-covered area of the feature map. The output of the max-pooling layer includes the most prominent features from the previous feature map.

LSTM AND GRU UNITS

In LSTM and GRU networks, memory cells are denoted by the number of units they have, which determines their capacity to remember and compare information with previous evidence. The information stored in these memory cells is carried forward to the next time step for further training.

DEMONSTRATION

The LSTM and GRU models were built on Python 3.6.5 and Keras 2.0 API with Tensorflow backend on a Windows 10 64-bit system. The pre-processing stage included tokenizing, removing numbers and special characters, and replacing upper case to lowercase. The LSTM and GRU models were used to identify SQLi in a dataset split into 25% testing and 75% training sets. The input was passed through embedding, and the sigmoid optimizer was used to minimize the model's loss during training. Different pooling windows and feature maps sizes were tested, with the best performing window size being 5. A dropout value of 0.1 was deemed optimal for the model, which was also trained for 10 epochs with a batch size of 64.

VIII. SYSTEM STUDY

FEASIBILITY STUDY

In the initial phase of a project, the feasibility of the proposed plan is assessed, and a business proposal is presented, including a rough outline of the project and cost estimates. During the system analysis phase, it is necessary to conduct a feasibility study of the proposed system to ensure that it will not be a financial burden on the company. To conduct a feasibility analysis, a clear understanding of the system's major requirements is essential.

ECONOMICAL FEASIBILITY

The purpose of this investigation is to evaluate the financial consequences that the system will have on the company. The resources that can be allocated to the research and development of the system are limited. As a result, it is necessary to justify the expenses. As a result, the created system is cost-effective, thanks to the usage of freely available technologies. Only custom-made products required purchase.

TECHNICAL FEASIBILITY

The purpose of this study is to assess the technical feasibility of the system, which refers to the technical needs of the system. The system should not excessively consume the available technical resources as this may put a strain on the client. Therefore, the system must have a moderate requirement, requiring only minimal or no changes for its implementation.

SOCIAL FEASIBILITY

This study focuses on evaluating the user acceptance of the system, which includes the training process to ensure efficient use of the system. The user should not feel intimidated by the system and must accept it as a useful tool. The level of acceptance by the user is influenced by the training methods employed to make the user familiar with the system and build their confidence. Constructive feedback from the user is valuable as they are the ultimate end-users of the system.

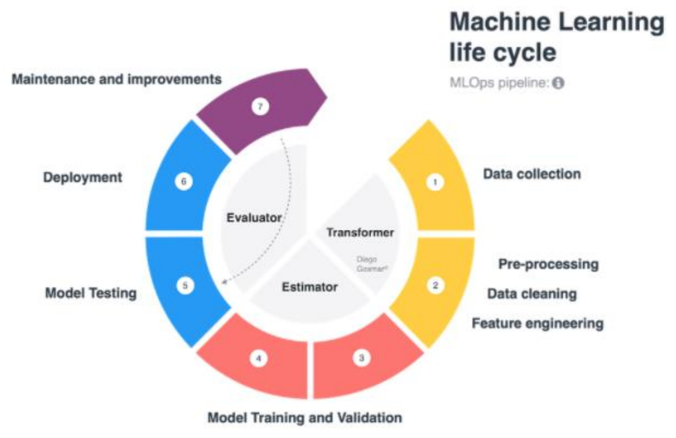
IX. SYSTEM TEST

QUALITY ASSURANCE

Quality assurance is an essential step to ensure that the software system fulfills all your requirements. It involves verifying if all the previously agreed features have been incorporated and if the program functions as intended. During program testing, it is important to document all parameters used as outlined in the technical specification document.

DETECT BUGS AND FLAWS

Software testing is an essential aspect of software development to ensure that the code meets the requirements and functions as expected. It helps to detect errors, vulnerabilities and bugs in the code during development, and different types of testing can catch bugs that are only visible at runtime. However, traditional software testing and machine learning (ML) testing differ in how they function. While traditional software testing is human-driven with programmers providing input data and logic, the machine validates the logic and checks for the desired behaviour of the system or program. In contrast, ML testing involves programmers inputting data and the desired behaviour to produce the logic of the machine, which is then tested repeatedly to see if the learned logic remains consistent. This ensures that the system understands the logic and develops a model according to the desired behaviour. A testing model summarises how to think about test development, and several issues need to be considered when writing model tests, including continuous testing. Building continuous testing procedures into the model testing strategy gives faster delivery and feedback to developers. Different types of tests can be used, including pre-train tests, post-train tests, minimum functionality tests, invariant tests, and directional testing. Pre-train tests catch bugs before running the model, while post-train tests deal with job behaviour and check whether the model performs correctly. Minimum functionality tests assess model performance based on specific cases found in data, allowing for the identification of critical instances where prediction errors can have serious consequences. Invariant tests assess whether the model prediction remains consistent despite changes in the input data. Directional testing checks how perturbations in the input change the model behaviour, ensuring that the trained model performs correctly, and changes in input affect the model prediction. Traditional ML model development can have slow iteration cycles due to manual and script-driven processes. However, automated ML pipelines shorten the time between training models and deploying them, resulting in faster iteration cycles. checks how perturbations in the input change the model behaviour. If the trained model performs correctly, the changes in input will affect the model prediction. For example, if we had a model that estimates the price of a house, taking into account square footage, we would want to see that added space makes the house prices go up. Traditional ML model development can have slow iteration cycles, due to manual and script-driven processes. With an automated ML pipeline, fast iteration cycles shorten the time between training models and deploying them.



ENHANCING MACHINE LEARNING WITH TESTING

An ML library is a useful resource for developers as it provides readily available functions and routines to write complex programs without the need to create all the code from scratch. However, despite being well-tested, ML libraries for modeling are not perfect and can benefit from additional testing to ensure that the system operates as intended. Testing plays a crucial role in enhancing the quality of ML models by identifying errors and vulnerabilities, and ensuring that the system functions correctly when integrating code from the library. Test cases are employed to evaluate whether the system meets the requirements and performs correctly. The testing process is considered complete only when all the functional and non-functional requirements of the product are satisfied. During test case execution, five parameters are taken into account. Regression testing is a type of testing that covers previously tested software to ensure that it continues to function correctly even after the introduction of changes in the component or module. For example, if a dialer was tested previously and later upgraded, regression testing would be performed to ensure it still works as expected. Any issues that arise during regression testing are known as regressions.

X. CONCLUSION

The current model proposed a novel adaptive deep forest-based method to detect complex SQL injection attacks. Initially, the structure of deep forest is optimized by concatenating the input of each layer with the raw feature vector and the average of previous outputs. This method effectively addresses the issue of degradation in original features with an increase in the number of layers. Additionally, an AdaBoost algorithm-based deep forest model is used, which updates the weights of features on each layer using error rate during training. This approach assigns different weights to different features based on their influence on the final outcome, which allows for the automatic adjustment of the tree model structure and the handling of multi-dimensional fine-grained features, effectively avoiding overfitting problems. To further improve the results, the project aims to incorporate Recurrent Neural Networks (RNN) in the future and compare its performance with existing approaches. Overall, this proposed approach has shown promising results for the detection of SQL injection attacks

REFERENCES

[1] https://owasp.org/www-community/attacks/SQL_Injection
[2] https://www.w3schools.com/sql/sql_injection.asp
[3] https://www.imperva.com/learn/application-security/sql-injection-sqli/
[4] Stanislav Abaimov et al., "A survey on the application of deep learning for code injection detection", Array 11 (2021) 100077.
[5] Omer Kasim, "An ensemble classification-based approach to detect attack level of SQL injections", Journal of Information Security and Applications 59 (2021) 102852.
[6] Jianguo Zheng et al., "Pattern Mining and Detection of Malicious SQL Queries on Anonymization Mechanism", IEEE Journal, 2021.
[7] Xin Xie et al., "SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN", IEEE Journal, 2019.
[8] QI LI et al., "A SQL Injection Detection Method Based on Adaptive Deep Forest", IEEE Journal, 2019.
[9] https://blog.testproject.io/2022/01/17/machine-learning-testing-for-beginners-the-all-in-one-guide/#
[10] (1994-2010) Priya G.1 , Saravanan K.2 * and Renuka, C.3