Cyber Attack Detection Method Based on NLP and Ensemble Learning Approach

[1]Bathku .Raghavendar yadav,[1]bathkuraghavendaryadav@gmail.com,M.tech Scholar,
[2]Dr. P. Raja Prakash Rao, rajaprakasharao.p@hmgi.ac.in Associate professor of ECE,
Holy Mary Institute of Technology and Science, Bogaram, Medchal Makhajgiri Dist, Telangana, 501301.

Abstract:

The use of web apps has rapidly increased in the last few years. Thanks to web apps, information sharing and billions of transactions over the Internet have become commonplace. Users provide web-based applications with essential and private information, which is then stored in data bases. Because linked databases and web applications may be accessed via the Internet, they are vulnerable to cyberattacks. The most common type of cyberattack is SQL Injection. Attackers use SQL code injection to steal intended data. Web application users may experience negative consequences from SQL Injection. This research has led to the proposal of an Ensemble Learning Algorithm and Natural Language Processing (NLP)-based SQL Injection detection technique. Natural language processing (NLP) produces feature patterns, extracts features, converts each text into a vector of numbers, and creates a bag-of-words model (BOW model). BOW model railroads Forest Classifier at Random. An algorithm for ensemble learning is the Random Forest Classifier. In order to improve detection performance, this approach constructs a single prediction model by combining many machine learning algorithms. The trained classifier more accurately classes SQL Injection payloads from the dataset.

Index Terms: regular expression, over fitting, random forest classifier, and natural language processing

Introduction

The Open Web Application Security Project (OWASP) develops web application security-related articles, techniques, documentation, tools, and technologies. An online community lists SQL Injection as one of the top 10 vulnerabilities that affect Web applications. OWASP [1], [2]. The proprietor and user of a website are at significant danger from SQL Injection. Attackers use a website's input fields to deliver SQL Injection payloads. We run queries for those payloads. After then, the linked databases respond to the attackers. An attacker may be able to determine the pattern of data by examining these responses, which might allow them to get access to databases and alter the data that is kept there. Users' and web application owners' confidential information may be exposed by this attack. Therefore, in order to identify attacks, it is required to analyse the syntax and payload pattern that users send. Static analysis, dynamic analysis, and web framework are ways that are now available to identify SQL Injection. Special characters from user queries are filtered by Web Framework. This technique cannot detect SQL code injection when user-sent SQL queries lack special characters. The syntax of data entered by users is verified during static analysis. This method's drawback is that it can only identify tautology—all other forms of SQL Injection are not detected. SQL Injection is detected by the dynamic analysis through a comparison of SQL Injection codes with user queries received. However, the drawback is that every kind of SQL Injection function must gather data for comparison. Compared to conventional approaches, machine learning techniques are more effective at detecting SQL Injection. Research has been conducted utilising deep learning and machine learning techniques to identify SQL code injection. Certain machine learning models experience overfitting and under fitting issues when attempting to classify payloads as SQL Injection or Normal payloads.

These issues raise the rate of false positives and false negatives in machine learning models while decreasing detection accuracy and rate. When a classifier is trained with an excessive number of training datasets and features that have been retrieved from those datasets, overfitting occurs [6]. Overfitting creates a flexible model that incorrectly classifies testing dataset payloads. In order to reduce overfitting, fewer training datasets and features must be used during the classifier training phase. Undertaking will result from this approach [6]. A model is underfitted when it is trained using only a small training dataset and extracted features [6].An underlying machine

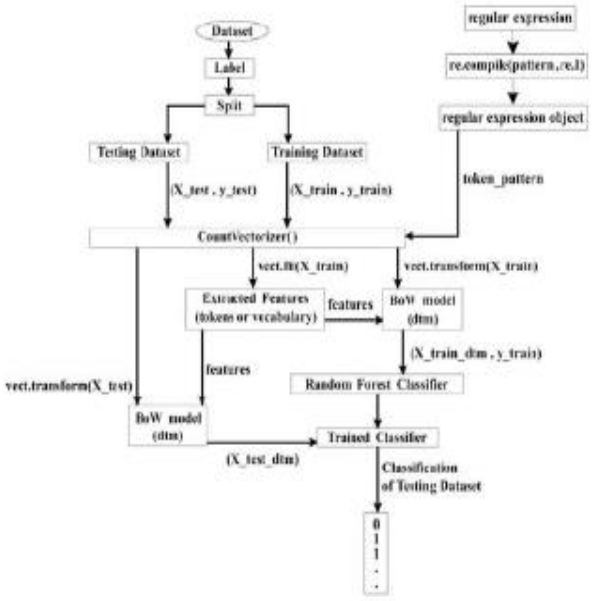
..ISSN: 2040-0748 Vol-13 Issue-02 Aug 2024

learning model incorrectly classifies testing payloads. Lowering the likelihood of under fitting can be achieved by increasing the quantity of training datasets and features. Consequently, there will be a greater chance of overindulging. A trained classifier cannot classify payloads with the proper class due to over- and under-fitting. The primary goal of this research project is to provide an efficient detection technique that will reduce the incidence of over- and under-testing, improve detection accuracy and rate, and lower the rates of false positives and false negatives. This research project offers an appropriate means of effectively detecting SQL Injection. Regular expressions are created by analysing the syntax of all varieties of SQL injection payloads, regular payloads, SQL query patterns, and syntactical linkages between them. In the feature extraction phase, these regular expressions are employed as a token pattern to construct a BOW model. The dataset is then classified by this model using an ensemble learning technique known as the Random Forest Classifier.

The term "SQL Injection Detection method" describes how to locate SQL injection patterns inside datasets. Growing awareness of the injection assault and its negative effects has led to several studies looking on the detection of SQL Injection .A machine learning strategy to detecting SQL Injection was proposed by authors in [7] in 2018. Only the HTTP request portion was present in the dataset. Feature extraction was done using regular expressions. Subsequently, the Word2Vec technique converted these characteristics into dense vectors, created a word vector model, and trained the SVM classifier. The classifier kernel functions and parameters were changed during training.95.4% detection rate (True Positive Rate) is attained in this experiment.A classification method to categorise online scripts as harmful or non-malicious was suggested by the authors in [8].

Tokenizing and Blank separation are two feature extraction techniques that are suggested, and TF-IDF is utilised for vectorization. To categorise web codes, three machine learning models—SVM, Naive-Bayes, and KNN—were developed. Gaussian kernel SVM classifier achieves 99.16% detection accuracy, which is better than other models' performance. However, the syntactical link between features is destroyed during feature extraction.When training the classifier, the trained classifier will classify more accurately if the number of observations is greater than the number of features.In [9], authors classified queries as dangerous or benign using a SQL Injection detection technique. There are a lot of both harmful and benign searches in the dataset. The dataset was classified by the trained Naive-Bayes classifier, which attained an 89% detection rate. However, one factor contributing to form is classification is the users' participation as features in the model's training. Because the user's role will be recorded in the weblog history as admin if an attacker uses the admin identity to alter the database. Additionally, the classifier ignores feature dependencies and is unable to identify all forms of SQL Injection. A decision tree and hidden mark ov model (HMM) based anomaly detection technique was presented in [10]. The dataset is made up of actual industry log messages. The dataset is divided into normal and anomalous data categories by the decision tree model. Only standard data, along with its parameters and values, are extracted by data extractors. Several HMMs are trained using these parameters and values. The HMMs then serve as a detector of anomalies. 93.54% detection accuracy and 89.90% detection rate (TPR) are attained with this method [10]. Only the number of parameters was employed in this study to distinguish between normal and abnormal data [10].

Methodology



SDataset Collection

GatheringOnline applications are subjected to penetration testing using open source tools like Sqlmap and Libinjection. These tools analyse whether an application is susceptible to a SQL Injection attack by passing SQL Injection codes as payload to web apps.The payloads produced by these instruments were recorded, and the SQL Injection dataset is made

..ISSN: 2040-0748 Vol-13 Issue-02 Aug 2024

up of all of these payloads [12]–[15]. For the experiment, 36,569 thousand payloads were employed.B. Labelling Datasets Because text data cannot be understood by the machine, data with the labels "SQL injection" and "norm" are mapped to 1 and 0. 19,303 thousand standard payloads and 17,266 thousand SQL Injection payloads are employed for this investigation. Table I shows the sample output following labelling of the experimental dataset.

Separate Dataset Using the train-test-split technique, we separated all labels and payloads into four separate variables. This technique divided the whole dataset in half, setting 25% aside for testing and 75% for training. The Random Forest Classifier is trained only using features collected from these 27,426 thousand training datasets. Section D: Extracting Features and Generating BOW Models Some other names for features are inputs, tokens, predictors, and at-tributes. To build a token pattern, one may utilise the regular expression library in Python and its method re.compile(). Token patterns are supplied in as regular expression groups to the re.compile() method. The regular expression object is created by compiling these patterns using the re.compile() function.This item is a pattern that represents a token. The features extraction process involves importing the CountVectorizer class. The CountVectorizer() method extracts 23,715 characteristics and stores them in the vocabulary. It uses class regular expression objects as token patterns.•vect.t(X train) memorises the training data's token pattern for usage in its vocabulary.

A vocabulary consisting of extracted features may be created using the vect object and the fase() function of the ConfVctorizer() object.This dataset is used for training purposes.A set of text documents may be transformed into numerical feature vectors with the help of Count Vectorizer. The payloads in this study are treated as plain old text documents. "Bag of n-grams" is the name given to this particular approach, which involves tokenization, counting, and normalisation. The ngram range is set to (1,1) for this study. Table II displays the example output of the BoW model.The filtered token pattern is used by vect.transform(X train) to construct a document-term matrix (dtm) from the training data. In this matrix, tokens are represented by columns and each payload by rows, and the BoW model is denoted by dtm. The model is now being fed into a Random Forest classifier.

Random Forest Classifier

A random forest, as the name suggests, is an ensemble of several separate decision trees. Ensemble algorithms include Random Forest Classifier [5, 16]. The number of trees was fixed to 10 for this study purpose. Ten decision trees make up the forest as a whole. There are two main tenets of the random forest classifier that are followed throughout the training phase:•In a random forest, the classifier is trained using a bootstrap dataset, which is a random subset of the training dataset. These samples are collected using a replacement method. All of the trees in the forest will have less variance (no overfitting) and less bias (no underfitting) since they were trained on separate data. In a testing sample, the two classes represented by the anticipated classes from each tree are like votes. The final forecast is produced by tallying votes. The final projected class is the one with the most votes. Bagging is the name given to this whole procedure. Every decision tree takes into account a subset of the characteristics while deciding how to divide each node. A node will be divided based on the feature with the greatest Gini Gain and the lowest weighted Gini impurity. Every node in each tree will follow this procedure. With a split, the quantity of impurity has decreased, which is called Gini Gain. What remains after a split is the quantity of impurity, which is called weighted gini impurity. Since the classifier is trained with distinct attributes for each tree in the forest, it is less likely to overfit or underfit. In feature selection, the classifier uses a random selection process.Here are the steps to implement a classifier:

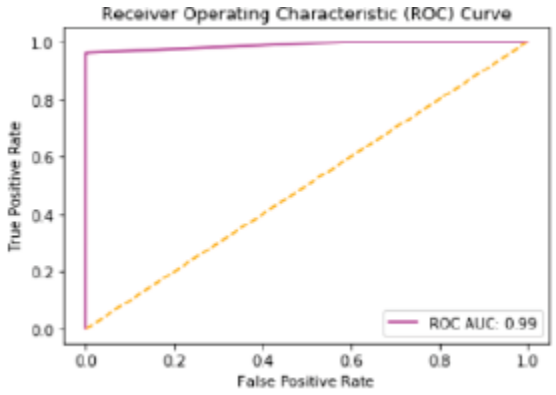
Generating a bootstrap dataset for each tree is the first step.Figure out how much gini impurity is present at each node.To get the number of features, multiply the total number of features by 2.5.•Find the Gini Gain and weighted gini impurity values for each feature.Choose the feature that divides nodes according on their Gini index.•Continue from step two to step five until the gini impurity is zero.One way to find out how valuable a feature is is to ask it a question. In essence, these numbers represent the feature threshold. To reach to the bottom of the tree, you need to ask these questions and apply their replies. If the question's response is correct, the node after it is the left node; otherwise, it's the right node.•To make a forest, repeat steps one through seven until you have ten trees.next this, the next steps constitute the prediction operation:Predicts and saves the goal result using the testing dataset and the rules of each randomly generated decision tree.Identify which projected class received the most votes.•The final forecast from the random forest should be the class with the most votes.Not all characteristics and observations are visible to each tree. Over- and under-correlation of 10 trees makes the forestless vulnerable to these problems [18].Section F: Identifying and Categorization Now that we've finished training the model, we can see how well it predicts the right classes on the testing dataset. That is why the test dataset is converted into a

document-term matrix (BOW model).A document-term matrix, representing the BOW model produced from the testing data, is constructed by using the fitted token from vect.fit(X_train) in the vect .transform(X_test) function. The dataset used for testing is X test. The vect tokens used in the training set make up the columns of this BOW model, while the testing dataset samples make up the rows. The testing dataset is not used to extract features.The model receives this BOW model as input.algorithm of the Random Forest Classifier class. By using the predict() method, the trained classifier assigns the right class to payloads received from the BOW model. A total of ten decision trees will provide a yes/no prediction for every test sample. A vote by the ten trees in the forest determines the projected class of an input sample. To begin, the classifier tallies up the votes for each anticipated category. After then, the forest's final forecast is the class with the most votes.

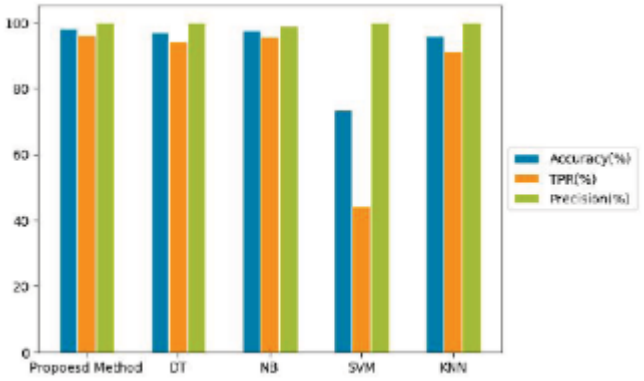
Results

Classifiers are taught and their performance is evaluated using the sickit-learn module, a Python machine learning tool. The functions needed to construct performance assessment metrics are provided by the metrics module of the scikit-learn toolkit. To measure the classification performance of the proposed method and other classifiers, evaluation metrics are utilised. These metrics include confusion matrix, accuracy, precision, True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), False Negative Rate (FNR), Responder Operating Characteristic (ROC) Curve, and Area Under the Curve (AUC). As a metric for machine learning classification issues, the confusion matrix measures performance. What came from all the muddle.

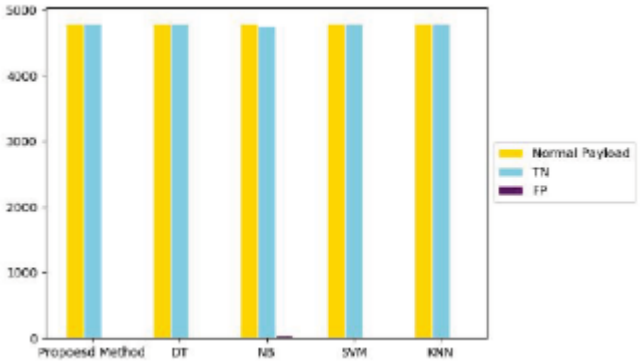
True Negative	False Positive	False Negative	True Positive
4792	1	168	4182



ROC Curve and AUC



Result Comparison (Accuracy, TPR, Precision).



Confusion Matrix Result Comparison

Method	Accuracy(%)	Precision	TPR	FPR	TNR	FNR
Proposed Method	98.1515	0.9997	0.96137	0.0002	0.99979	0.03862
Decision Tree Classifier	97.16	0.9997	0.9406	0.0002	0.9997	0.05931
Naïve Bayes Classifier	97.58	0.9911	0.9577	0.0077	0.9922	0.0422
SVM Classifier	73.466	1.0	0.4422	0.0	1.0	0.557
KNN Classifier	95.87	0.9997	0.9135	0.0002	0.9997	0.0864

Conclusion

In this study, we exhibit a method that can effectively identify SQL Injection payloads. Token pattern creation and feature extraction are carried out with the help of natural language processing. In order to obtain better classification, Random Forest Classifier tunes all relevant parameter values.the end. Using a bagging strategy in the classifier prevents it from being too confident or under confident. Accuracy (98.15%), detection rate (96.13%), false positive rate (0.02%), and false negative rate (3.86%) are all achieved by this proposed method. This detection approach outperforms Decision Tree, Naive Bayes, SVM, and KNN on the testing dataset when it comes

to classifying SQL Injection and Normal payloads.A deep learning model, when applied to this suggested method in the future, may greatly improve it. Using the Word embedding approach for feature extraction and a Recurrent Neural Network model for dataset classification yields accurate results. The word embedding technique maps all characteristics into a one-dimensional vector space, representing both the syntactical and semantic relations among features.Using the training dataset, a recurrent neural network may learn sequential features. This model learns patterns and then uses that knowledge to make predictions based on testing data. There is need for improvement in this study so that it can identify other forms of cyberattack.

References

1. Diallo, Abdoulaye Kindy & Pathan, Al-Sakib. (2012). A DetailedSurvey on Various Aspects of SQL Injection in Web Applications:Vulnerabilities, Innovative Attacks, and Remedies.International Journalof Communication Networks and Information Security.5.
2. [2] owasp.org. 2020. OWASP Top 10 Security Risks Vulnerabilities. Re-trieved August 20, 2020 from https://sucuri.net/guides/owasp-top-10-security-vulnerabilities- 2020/.
3. [3] sqlwiki.netspi.com. 2019. Injection Types. Retrieved January 5, 2020from https: //rb.gy/xbb6qf.
4. [4] udemy.com. 2019. Hands On Natural Language Processing (NLP) usingPython. Retrieved November 18, 2019 from rb.gy/7iihvo.
5. [5] Kowsari, Kamran & Jafari Meimandi, Kiana & Heidarysafa, Mojtaba& Mendu, Sanjana & Barnes, Laura & Brown, Donald & Id, Laura &Barnes,. (2019). Text Classification Algorithms: A Survey. Information(Switzerland).10. 10.3390/info10040150.
6. [6] medium.com. 2019. Ensemble Learning and Random Forest. Retrieved January 20, 2020 from rb.gy/w6xktt.
7. [7] Chen, Zhuang & Guo, Min & zhou, Lin. (2018). Research on SQL injec-tion detection technology based on SVM. MATEC Web of Conferences.173. 01004. 10.1051/matecconf/201817301004.
8. [8] R. Komiya, I. Paik and M. Hisada, ”Classification of malicious web codeby machine learning,” 2011 3rd International Conference on AwarenessScience and Technology (iCAST), Dalian, 2011, pp. 406-411, doi:10.1109/ICAwST.2011.6163109.
9. [9] A. Joshi and V. Geetha, ”SQL Injection detection using machine learning,” 2014 International Conference on Control, Instrumentation,Communication and Computational Technologies (ICCICCT), Kanyaku-mari, 2014, pp. 1111-1115, doi: 10.1109/ICCICCT.2014.6993127.
10. [10] Q. Cao, Y. Qiao and Z. Lyu, ”Machine learning to detect anomaliesin web log analysis,” 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 519-523,doi: 10.1109/CompComm.2017.8322600.
11. [11] Wang Y., Li Z. (2012) SQL Injection Detection via Program Tracingand Machine Learning. In: Xiang Y., Pathan M., Tao X., Wang H.(eds) Internet and Distributed Computing Systems. IDCS 2012. LectureNotes in Computer Science, vol 7646. Springer, Berlin, Heidelberg.https://doi.org/10.1007/978-3-642-34883-9 21
12. [12] gist.github.com. 2017. libinjection-bypasses.txt. Retrieved January 20,2020 from https://rb.gy/cn3ag9
13. [13] github.com. 2016. HttpParamsDataset. Retrieved January 11, 2020 fromhttps://github.com/Morzeux/HttpParamsDataset
14. [14] github.com. 2016. foospidy/payloads. Retrieved January 10, 2020 fromhttps://rb.gy/vwpqhy
15. [15] github.com. 2017. Scott-Park/MachineLearning. Retrieved January 12,2020 from https://rb.gy/2k2gvh
16. [16] Xu, Xiaodan & Liu, Huawen & Yao, Minghai. (2019). Recent Progressof Anomaly Detection. Complexity. 2019. 1-11. 10.1155/2019/2686378.

17. [17] towardsdatascience.com. 2018. An Implementation and Explanationof the Random Forest in Python. Retrieved January 2, 2020 fromhttps://rb.gy/evyvzf
18. [18] towardsdatascience.com. 2018. Feature Selection Using Random forest.Retrieved January 5, 2020 from https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597

Author's details:

I am Dr. P. Raja Praaksha Rao, have pursued my Graduation, two post graduations (M.C.A and M.Tech(ECE) from Osmania University and JNTU Hyderabad respectively, and Ph.D in wireless communications from Shri JJT University, Jaipur, Rajasthan, working as Associate Professor of ECE in Holy Mary College of Technology and Science, Hyderabad. As of now having a total teaching experience of 28 years, worked at different levels of administration like, Head of the department and Principal, have published various technical papers on wireless communications in reputed Journals

BathkuRaghavensar yadav M tech with specification of Electronics and communication Engineering in Holy Mary Institute of Teconology and Science ,Bogaram.