SQL INJECTION DETECTION ANALYSIS USING DEEP LEARNING

Palaparthi Vasavi Sai Lakshmi
Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), Visakhapatnam, Andhra Pradesh

Abstract: In recent years, sharing information via the Internet across various platforms and web-applications has grown increasingly widespread. Users provide vital information to web-based apps, which store it in databases. Due to their Internet accessibility, these programmes and the databases they are connected to are vulnerable to a variety of information security risks. Assaults like Cross-Side Scripting (CSS), Denial of Service (DoS), and SQL Injection attacks are among the risks. The top ten vulnerabilities for web-based applications include SQL Injection attacks. An organisation or corporation may suffer harm as a result of this type of attack since the perpetrator can steal sensitive information. Financial loss, the disclosure of private company information, a decline in the company's stock market value, or any combination of these could be the results. We employed the Naïve Bayes algorithm in this research, which provided superior accuracy of about 99% approximately.

Index Terms: - Structured Query Language (SQL) Injection attacks, Denial of Service Attack (DoS), vulnerabilities.

I Introduction

The majority of the programmes we use on a daily basis are web-based programmes. In order to maximise their exposure, organisations decide to make the applications available online. Internet exposure heightens the security risks associated with unrestricted access. We have grown accustomed to conducting a variety of transactions online thanks to the development of the Internet. In some kind of database, all the information that users enter during these transactions on websites or web applications is kept. Structured Query Language, also known as SQL, is a language that can be used to connect with relational databases. SQL Injection Assaults are a type of web hacking technique that target databases using SQL to initiate attacks and manipulate them to perform what the user wants. The cyber defenders are becoming more and more concerned about SQL Injection Attacks. More than 45% of the discovered web-based assaults from the previous year were caused by SQL Injection and Remote Code Execution attacks. One of the most common cyberattacks continues to be SQL Injection assaults. Although numerous methods have been devised to counteract such assaults, cybercriminals continue to find ways to circumvent the different defences put in place to stop SQL Injection attacks.

The use of machine learning algorithms to identify and stop different cyber security risks has recently generated a lot of discussion. The effectiveness of using supervised and unsupervised learning techniques to identify security threats cannot be disputed, but the computing power and processing time needed to run such complicated algorithms continue to be a major concern for the community working on cyber security, which is constantly evolving. The use of various machine learning methods to identify SQL Injection threats has been the subject of extensive research. In machine learning, there isn't a single ideal algorithm or method that can be used to solve every problem. Before choosing a specific strategy, an issue must be examined against a variety of algorithms that fall within the classification or regression procedures, and the results must be compared for maximum accuracy. Previous studies have used the Naive Bayes technique to detect SQL Injection. In this study, we employ a technique known as the Gradient Boosting Algorithm to identify and stop SQL Injection assaults.For this specific issue, we also built the Naive Bayes method and contrasted the outcomes against Gradient Boosting; the specifics of this comparison are covered in later sections of this study.We introduce SQL Injection assaults at the outset of this work, along with the necessity for and driving force behind developing a more effective SQL Injection detection system. The details of SQL Injection attacks and their many kinds are then covered in section II. The related research that has been done thus far is described in Section III.

There is enough research available from all the important implementations and research work that has been done so far to understand the issue and make improvements. The general strategy we are employing to resolve this problem is supervised learning, which is introduced in Section IV. This section also discusses the two algorithms that were taken into consideration for this experiment.

2 Literature survey

The study and work completed to date on properly identifying SQL Injection attacks is briefly summarized in this section. There are two general categories into which the research on detecting SQL Injections has been divided. The first strategy is to safely write the source code to ensure that SQL queries have enough input validation. The second strategy entails deploying extra software to check SQL queries that are being transmitted across web apps and executed across databases. Recently, some researchers have also emphasized the significance of combining these two methods in order to develop a more accurate SQL Injection detection model. This section discusses both the methodologies and the research that has been done in those areas. A static analysis tool called the JDBC checker was created by Goud et al. to check SQL strings for faults and to confirm whether they include any potentially dangerous queries [3]. It offers to identify and highlight probable SQL query issues as well as validate the validity of the SQL strings. It operates by creating a list of all possible SQL strings that could be executed across a specific application statically and then inspecting each one of those potential SQL strings for harmful content and semantic issues instead of dynamically checking each query as it is generated during runtime. This method may have a number of issues, such as the enormous storage space needed to store all possible SQL queries and the question of how a tool might generate all feasible queries for an application. It has a very high likelihood of failing to forecast SQL query statements that were actually executed. A tool called CANDID, which stands for "Candidate Evaluation for Discovering Intent Dynamically," was created to get around this static analysis constraint. It operates according to the idea of programmer intent. According to the idea of "programmer intent," a SQL query structure should be formed exactly as the application's author intended. In SQL injection attacks, a pattern has been noticed: the malicious SQL query that is performed throughout the database always has a different structure than the one that the programmer intended. The authors of this paper held the opinion that recognizing this structural difference could be a crucial first step in effectively identifying and stopping SQL injection attacks. When the program reaches the point where it will build and execute a SQL query, this tool dynamically creates the SQL query structure that the programmer intended.

In order to detect and stop malicious queries from being executed, the produced SQL query generated by the programmer is compared to the real SQL query supplied by the user. Later, a few researchers had the bright idea to effectively stop SQL Injection attacks by combining the use of static analysis and dynamic monitoring. On the basis of the notion of utilizing both static and dynamic approaches, AMNESIA is a tool [5]. With this method, all potential SQL queries that the program could make are already included in the application code. These all come from created and preserved for comparison purposes. SQL queries generated at runtime are all dynamically monitored at the same time, and each one SQL query is compared to the list of possible SQL queries.

A user-generated SQL query is flagged as malicious and is not permitted to be run on the database if there is no match for it in the list of permissible lawful SQL queries. The major disadvantage of this technology is that it may produce a large number of false positives and is not completely accurate. Similar methods were employed by Buehrer et al. [6] in comparing the queries that were actually created with those that were supposed to be generated by the algorithm. This method only differs in that it uses Parse Trees to produce the desired output. By parsing the statements using the language's grammar, parse trees are produced.

This method is only dynamic, meaning that it has no previously saved set of potential valid queries and only evaluates the SQL queries generated at runtime. This method generates two parse trees: one by processing the user's input to create a tree that would be produced if the user's input were a valid query, and the second tree by analyzing the input itself to determine how the query will actually be run.

One issue about malicious SQL queries, which was also covered previously, is that they always behave differently from how the creator intended. Therefore, this study claims that in the event of a malicious query, the two parse trees generated will be distinct, allowing for more effective detection of SQL Injection attacks.

3 Implementation Study

In the existing system the proposed work was used using machine learning algorithms like SVM and ensemble methods which are used for analysis of attacks these learning algorithms gave an accuracy of 92% but they could not analyze all types of the regular expression which was need to perform the pattern recognition and better probability ratio

.

3.1 proposed methodology

The proposed system particularly includes an event pattern extraction method by aggregating together events with a concurrency feature and correlating between event sets in collected data. Our event profiles have the potential to provide concise input data for various deep neural networks. Moreover, it enables the analyst to handle all the data promptly and efficiently by comparison with long-term history data. And Naïve Bayes classifier was used for analyzing the data which has a better accuracy we achieve 99.96% accuracy on it

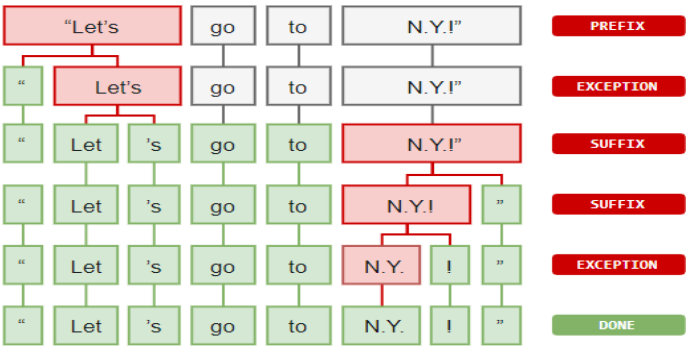

Fig 1:- System Architecture

4. Methodology

Gradient boosting is an Ensemble learning method to reduce errors and provide predictions with better accuracy. Gradient Boosting algorithm uses simple classifiers, mostly decision trees, in a sequential manner, to provide results.

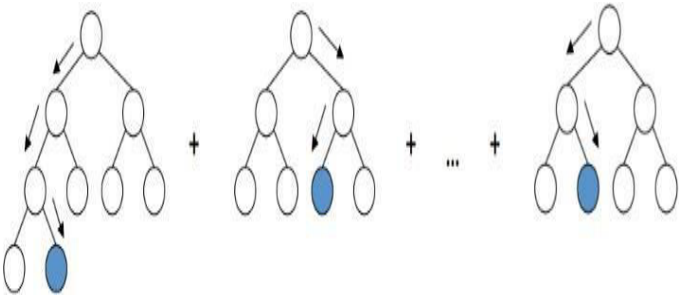

Fig1: Multiple Decision Tree Classifiers used for Gradient Boosting

The algorithm first uses the simple classifier to classify the data. Then the results are considered to calculate the errors or the data points that were not easily fit by the simpleclassifier. The algorithm then focuses on those data points in the next round and tries to fit them as well. In this way the errors are reduced, and outlier data points are also taken into consideration. But overdoing this remodelling can also cause overfitting. Hence learning to stop remodelling at an acceptable accuracy and error rate is also an important point to consider with this approach. In this paper, we use Gradient Boosting machine learning approach to detect and prevent SQL Injections.

I. Naïve Bayes

User gets unauthorized access to someone else's account details and the possession of this information could result in serious consequences for the person whose account information was stolen. This is a case of theft and a violation of data privacy.

This was a very simple example of a SQL Injection attack just for understanding, and most of the websites and web applications today would easily prevent this kind of attack. But there are various and more complex forms of SQL Injection attacks, some of which are described later in detail. The aim of the attackers using SQL Injection is to exploit the database that is connected to a website or a web application. It is extremely important to protect such databases against SQL Injection attacks in order to protect the important data stored in them. Letting an unauthorized user get access to a database can result in many unauthorized actions on the database like deleting tables, retrieving important information and many more terrifying things, and SQL Injection attacks make all of this possible.

II. Supervised Learning

Machine learning algorithms can be broadly classified as Supervised Learning algorithms and Unsupervised Learning algorithms. Supervised learning is a type of machine learning that in its simplest form, works in the following manner. We have a dataset called as training dataset and each individual component of this dataset is labelled. The supervised learning model basically learns the relationship between the data and the label and then uses this learnt information to classify new data that it has never seen before. This new data is called as the test dataset. We use test dataset to determine the accuracy of a supervised learning algorithm. This is how we predict the values or classify never before seen data using supervised machine learning. Supervised learning algorithms can further be broadly classified as Regression algorithms and Classification algorithms.

Regression algorithms are used for predicting a value for an individual data component, for example, predicting the value of a house, or predicting stocks. Usually the values predicted by the Regression algorithms are quantitative or numerical. Classification algorithms are used for classifying individual data components. For example, classifying if the vehicle is a truck or a car, or predicting if it will rain or not on a given day. Classification algorithms are used to predict qualitative values. Figure 3 shows the derived hierarchy of Classification and Regression algorithms. In this section we will focus on

Classification algorithms in Machine Learning, and more specifically the two classification models that are Naïve Bayes and Gradient Boosting.

MODULES:

III. Dataset

i. Plain-Text Dataset
A. Diversity
B. Size
C. Source
ii. SQL Injection Dataset
A.Categories
B. Size
iii. Tokenization:
i. Regular Expressions
A.Token_Count:
B. Token_Value:
C.Token_Type:

SQL Injection Attacks can be broadly classified into the following three categories:

1. Union Based SQL Injection
2. Error Based SQL Injection
3. Blind SQL Injection

5 Result and Evaluation Metrices

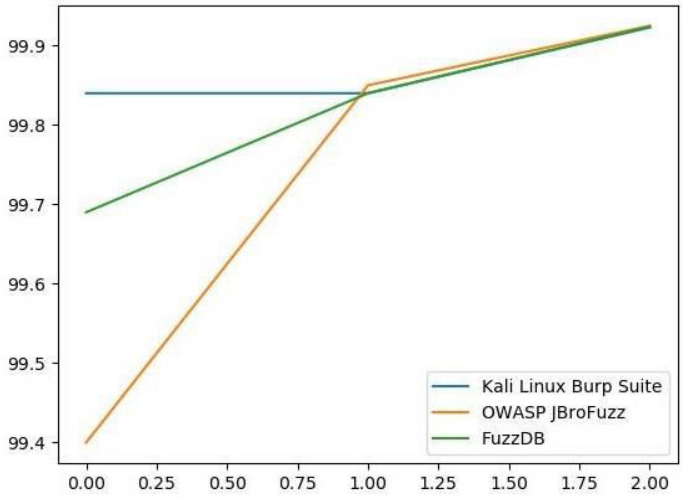

Fig 1

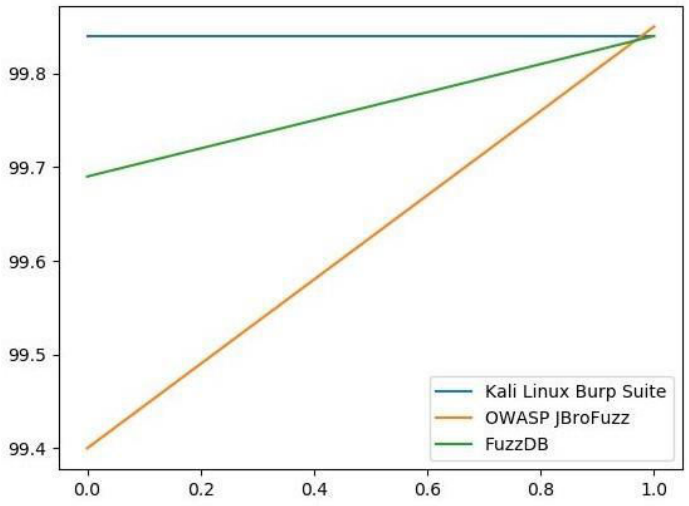

Fig 2

6 Conclusion

SQL Injection attacks remain to be one of top concerns for cyber security researchers. Signature based SQL Injection detection methods are no longer reliable as attackers are using new types of SQL Injections each time. There is a need for SQL Injection detection mechanisms that are capable of identifying new, never before seen attacks. Applying deep learning to the field of cyber-security is being considered by many researchers. Since machine learning in cyber-security is still a developing research area, there are not many libraries and open source tools that are deep learning specific and apply to problems related to threats and attacks.

In this thesis, the SQL Injection detection problem is approached by applying deep learning algorithms. Classification method is used to classify the incoming traffic as a SQL Injection or plain text. One deep learning classification algorithms are implemented on the problem, which are, Naïve Bayes Classifier. Naïve Bayes classifier deep learning model provides results with an accuracy of 99.6%. Ensemble learning methods are said to provide results with better accuracy as they implement multiple simple classifiers to improve error and accuracy. From this project it can be concluded that deep learning approaches can be used for SQL Injection detection, and Naïve Bayes classifier algorithm provides better accuracy. For future work this project could further be modified in terms of usability and efficiency. The deep learning approach for detecting SQL Injections could be used in combination with other

SQL Injection detection mechanisms such as static code analysis and web application firewalls.

The deep learning model can also be advanced further with better feature extraction. In this project tokenization approach is used to create features for the deep learning model. Other approaches can be used for feature extraction and training the model in more effective manner.

7 References

[1]J. Abirami, R. Devakunchari and C. Valliyammai, "A top web security vulnerability SQL injection attack — Survey," 2015 Seventh International Conference on Advanced Computing (ICoAC), Chennai, 2015, pp. 1-9.

[2]Diallo, A. K., Al-sakib, K. P.: A survey on SQL injection:vulnerabilities, attacks, and prevention techniques.2011. Retrieved from http://irep.iium.edu.my/769/1/ISCE2011_paper323.pdf and accessed on 10th June, 2017.

[3]C. Gould, Zhendong Su and P. Devanbu, "JDBC checker: a static analysis tool for SQL/JDBC applications," Proceedings. 26th International Conference on Software Engineering, Edinburgh, UK, 2004, pp. 697-698.

[4] Prithvi Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks", ACM Transactions on Information and System Security (TISSEC), v.13 n.2, p.1-39, February 2010.

[5] William G. J. Halfond , Alessandro Orso, "AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks", Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, November 07-11, 2005.

[6] Gregory Buehrer , Bruce W. Weide , Paolo A. G. Sivilotti, "Using parse tree validation to prevent SQL injection attacks", Proceedings of the 5th international workshop on Software engineering and middleware, September 05-06, 2005

a. Joshi and V. Geetha, "SQL Injection detection using machine learning," 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, 2014, pp. 1111-1115.

[7]William GJ. Halfond and Alessandro Orso," Preventing SQL Injection Attacks Using AMNESIA" ICSE'06, May 20-28, 2006, Shanghai, China ACM 06/0005.

[8] Takeshi Matsuda, Daiki Koizumi, Michio Sonoda, Shigeichi Hirasai, "On predictive errors of SQL injection attack detection by the feature of the single character"

[9] Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on 9-12 Oct 2011, On Page 1722-1727.

[10] Angelo Ciampa, Corrado Aaron Visaggio, Massimiliano Di Penta :"A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications".

[11] P. Joshi, Dissecting Bias vs. Variance Tradeoff In Machine Learning, 2015 [Online] Available: https://prateekvjoshi.com/2015/10/20/dissecting-bias-vs-variance- tradeoff-in-machine-learning/

[12] Xristica, What is the difference between Bagging and Boosting?, 2016 [Online] Available: https://quantdare.com/what-is-the-difference-between-bagging-and- boosting/

[13] Nick Galbreath, 'libinjection', 2012. [Online] Available: https://github.com/client9/libinjection.git

[14] foospidy/payloads, libinjection_bypasses.txt, 29 June 2007. [Online] Available: https://github.com/foospidy/payloads.git

[15] Linguistic Features, 'Spacy'. [Online] Available: https://spacy.io/usage/linguistic- features

[16] J. Brownlee, 'Start With Gradient Boosting, Results from Comparing 13 Algorithms on 165 Datasets', March 30, 2018. [Online] Available: https://machinelearningmastery.com/start-with-gradient-boosting/

[17] G Buehrer, B.W. Weide, P.A.G Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in: 5th International Workshop on SoftWore Engineering and Middlere, Lisbon, Portugal, 2005, pp. 106-113.

[18] Z. Su and G Wassermann "The essence of command injection attacks in b applications". In ACM Symposium on Principles of Programming Languages (POPL'2006), January 2006.

[19] S. W. Boyd and A. D. Keromytis. SQLrand: Preventing SQL Injection Attacks. In Proceedings of the 2nd Applled Cryptography and Network Security Conference, pages 292-302, June 2004.

[20] RA. McClure, and J.H. Kruger, "SQL DOM: compile time checking of dynamic SQL statements," Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on, pp. 88-96, 15-21 May 2005.

[21] Ke Wei, M. Muthuprasanna, Suraj Kothari, "Preventing SQL Injection Attacks in Stored Procedures" Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06 IEEE).

[22] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. ACM Trans. Inf. Syst. Secur., 13(2): 1-39, 2010.

[23] Mei Junjin, "An Approach for SQL Injection Vulnerability Detection," Proc. of TTNG '09, pp.1411-1414, 27-29 April 2009.

[24] YongJoon Park, JaeChul Park, "Web Application Intrusion Detection System for Input Validation Attack "Third 2008 International Conference on Convergence And Hybrid Information Technology.

[25] Needleman, S.B., Wunsch, C.D. "A general method applicable to the search for similarities in the amino acid sequence of two proteins", .T.MoI.BioI.48:443-453, 1970.

[26] Sangita, R., Avinash, K. S., Ashok S. S.: Detecting and Defeating SQL Injection Attacks International Journal of Information and Electronics Engineering, 2011.

[27] Nausheen, K.: Detection and Prevention of SQL Injection Attacks by Request Receiver, Analyzer and Test Model. 2011.

[28] Cristian, N., et al.: CBRid4SQL: A CBR Intrusion Detector for SQL Injection Attacks. 2010.