Open Access License Notice

An Experimental setup for Detecting SQLi Attacks using Machine Learning Algorithms

Binh An Pham
West Texas A&M University
Canyon, USA
bpham1@buffs.wtamu.edu

Vinitha Hannah Subburaj, Ph.D.
West Texas A&M University
Canyon, USA
vsubburaj@wtamu.edu

***Abstract*—SQL injection attacks (SQLi attacks) have proven their danger on several website types such as social media, e-shopping, etc. In order to prevent such attacks from occurring, this research effort investigates on efficient ways of detection and prevention, so that we can preserve each cyber-user's right of privacy. This research effort is aimed at investigating and looking at different ways to protect websites from SQL injection attacks. In this research effort, machine learning algorithms were used to detect such SQLi attacks. Machine Learning (ML) algorithms are algorithms that can learn from the data provided and infer interesting results from the dataset. We used SQL code and user input as our data and ML algorithms to detect malicious code. The machine learning model developed in this research can detect such attacks from happening in future. The precision and accuracy of the machine learning algorithms in terms of predicting the SQLi attacks has been calculated and reported in this research paper.**

Keywords—cybersecurity, SQL injection attacks, machine learning algorithms

I. Introduction

SQL injections (SQLi) are attacks in which an attacker sends SQL statements to an SQL database; the SQL statements allow the attacker to control what the server does. By doing so, an attacker can take full control over the server [1]. Attackers can send code by simply inputting an SQL statement in place of a username and password. The only way an attacker can use this exploit is by looking for inputs on the website in question. Assuming no security measures were taken towards the creation of the web application, this SQL statement would list all customers in the Customer database.

SQL statements can also allow attackers to gain administrator rights to the database as well, which means the attacker can add, edit, or delete data with nothing stopping them from doing it. The login boxes or inputs are the places where the SQL statements are typed, which then sends the malicious code to be run on the server hosting the database [2]. There are several types of SQL injection attacks that could be deployed [3]. Depending on what the attacker's goal is, the SQL injections are sent to the server one after another or all at the same time. Once the attacker gets to the database, they will be able to impose threats from several perspectives. The attacker may have access to very sensitive information and, therefore, can perform obliteration and alteration on that information.

Due to the increasing number of cyber-attacks and security compromises carried out in the Information Technology sector, quality research is carried out in the areas of cybersecurity to prevent such attacks from happening is becoming very crucial. This research has set up an experimental model that can not only be used for continuing research but also used inside classrooms for teaching purposes. The results obtained through this research effort has opened many doors for us to continue working on this project to obtain more accurate results. Since, there is not much research done in this area of using machine learning algorithms to predict SQLi attacks, we believe that the results disseminated in this paper may be useful to the entire computer science research community.

The rest of the paper is structured as follows. Section II discusses the state of the art. Experimental setup is described in Section III. Section IV discusses the results from the research. Section V concludes the paper with future work.

II. Related Work

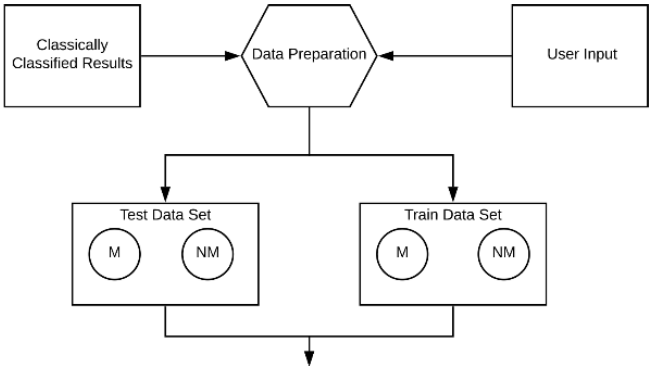
SQL-DOM [4] is one of those prevention methods that was developed to handle the injected HTML's commands. SQL-DOM can turn HTML into structured data thus making it hard for the hackers to enter HTML commands as input. Another preventive way is SQLrand [5], a method that transforms the application Instruction-Set Randomization to an SQL language, and the result of the transformation will be appended by a random number, with which the hacker who tries to perform any SQL injection will not be able to guess the appended number. AMNESIA [6] uses static analysis and runtime monitoring of application code to detect and prevent SQLI attacks. SQL injection, Fast Flux Monitor, Machine Learning, and Ardilla tools are methods to detect SQL injection attacks, while Noxes tool, SQLMap, and Session Shield are methods used to detect and prevent SQL injection simultaneously [7]. Joshi and Geetha in [8] have used a classifier which uses a combination of Role Based Access Control mechanism and Naïve Bayes machine learning algorithm for detecting SQL Injection attacks. Valeur et al. [9] in their paper, discussed a learning-based approach to detect SQL attacks. Ladole and Phalke [10] in their paper have used Query tree, Fisher Score, and Support Vector Machine classification to detect SQL injection attacks. What differentiates the proposed research from existing work is the experimental setup that can be used across different machine learning algorithms to detect SQL Injection attacks. Rawat

and Kumar [11] in their paper have talked about detecting SQL injection attacks using SVM classification algorithm.

Chen. et. al. [12] in their paper discussed a rule matching method of SQL injection detection using machine learning. The paper discussed the use of word vector text representation method and support vector machine (SVM) classification model to detect malicious SQL queries. Gu. et. al. in their paper [13], discussed a traffic-based SQL injection detection framework named DIAVA. This framework used a regular expression model to analyze the work traffic of SQL operations. DIAVA framework used a front-end to collect network related to SQL injection attacks and a backend to evaluate the vulnerability using dictionary attack analysis. Multilevel RegExp model is used to detect the attacks and to determine the vulnerability of leaked data. DIAVA framework used Hyperscan by Intel to perform multilevel matching of RegExps. Das. et. al. [14] in their paper have described edit-distance approach to classify a dynamic SQL query as safe or malicious using a web- profile that is prepared during the training phase along with the dynamic SQL queries. The authors used well-known supervised approaches such as Naive Bayesian, SVM, and Parse-tree based approach to analyze the dataset. The paper does a comparative study of the edit-distance and binary-distance methods with the machine learning classification algorithms. The proposed method of classification had good results with dynamic SQL queries with few overheads.

III. Experimental Setup

In this research effort, we used different machine learning algorithms to detect SQL Injection attacks. The dataset used for this research consisted of both vulnerable and safe SQL code. The datasets considered as input must be preprocessed and a set of features must be extracted from this dataset through a process called feature extraction. The results obtained from the classification were used to determine the vulnerable code. At the end, the classification accuracy, precision, and confusion matrix of each of the machine learning algorithms used will be determined. The model used in this research effort to predict SQLi attacks can be found in Figure. 1.



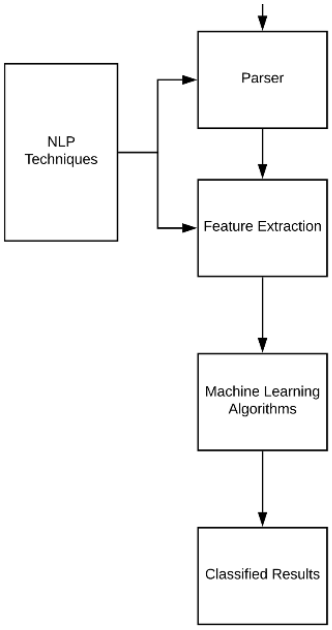


Fig. 1. Proposed model to Detect SQL Injection Attacks using Machine Learning Algorithms

A. Technologies

Python (3.7) was the language of choice because of its relatively simple syntax and the extent of supported libraries aimed at data science is second to none. The selected libraries were sklearn, which provided a wide variety of built-in machine learning algorithms and exceptionally fast setup time, xgboost, which was one of the best libraries to deploy extreme gradient boosting model, and pandas, which was a powerful library for data science. The development environment was IDLE, and the output was in the integrated shell.

B. Datasets

The process of collecting SQL injection (SQLi) malicious and safe queries was very challenging. We initially tried datasets that were manually created. After spending a significant amount of research, we ended up using two online sources primarily to gather our test and training datasets. Malicious queries, which tally up to 783 queries, were taken from [15] and benign queries, which tallies up to 700 queries, were taken from [16-19], and were put together into a text file. Then we assigned the queries with labels, label 0 for benign and label 1 for malicious.

C. Proposed Model

In this section, we proposed a unique model using machine learning to detect SLQi attacks. Instead of tackling the injection on the injection attacks on the server side by guarding the database, this model is designed to act as a filter on the client side. As a result, most of our data are text-based.

1) Data Preparation

For the purpose of reducing data noise and improving precision, unnecessary spaces and escape sequences were

eliminated and all the queries were converted into lowercase form.

2) Training and Testing Datasets

The training and testing sets are taken randomly from the dataset with a conventional ratio of 80-20 (80% for training and 20% for testing) usingtrain_test_splitfunction built into sklearn library:

train_test_split(trainDF['text'], trainDF['label'], test_size = 0.2)

3) Parser

Our model came across a common adversity during the data processing phase where traditional machine learning model explicitly takes in structured tabular numeric data, but our collected data are entirely non-structured texts. This is where parser for text comes into play. Text parsing is the process of transforming given series of text into desired smaller components based on some specific rules. There are two common ways to parse texts: regular expression separation and tokenization. The former parses the targeted text using desired regular expressions such as "[a-z]", "[\t]". The latter divides the text into tokens, where each token can be a character, a word or a phrase. In the case of SQLi attacks, the regular expressions do not determine the malice of a query, the appropriate text parsing method for this model is tokenization. Queries are split into tokens of words. For example:

Parsing "or 1=1 -- 1" into "or", "1=1", "--", "1"

4) Natural Language Processing (NLP) techniques and feature extraction

For NLP, there are many featured engineering techniques, but the one that proves to be the most useful for this SQLi attacks detecting model is Word Level TF-IDF Vectors. TF-IDF stands for Term Frequency and Inverse Document Frequency, which is an important index for term searching and figuring out the relevancy of specific terms in a document. Term Frequency specifically measures how often a word occurs in a document, where Document Frequency determines how often a word occurs in an entire set of documents. The formula to calculate the relevancy of a specific word is as follows:

$$\frac{Term\ Frequency}{Document\ Frequency}$$

or

$$Term\ Frequency \times Inverse\ Document\ Frequency$$

The most significant advantage of TF-IDF is that it will assume that the documents are just bags of words, where each word does not have any correlation to another. This method is simple but powerful for our use case since in SQL, there are no tense or grammar rules like human languages.

5) Machine Learning Algorithms

The proposed model used the following machine learning algorithms:

- Naïve Bayes Classifier
- Support Vector Machine
- Decision Forest
- Logistic regression
- Extreme Gradient Boosting

To preserve the experimental aspect of our model, every parameter for the algorithms are kept to default.

D. Alignment with NICE Framework

The undergraduate research project discussed in this paper includes many of the NICE Framework Knowledge areas [20] such as K0069 - Knowledge of query languages such as SQL (structured query language), K0070 - Knowledge of system and application security threats and vulnerabilities (e.g., buffer overflow, mobile code, cross-site scripting, Procedural Language/Structured Query Language [PL/SQL] and injections, race conditions, covert channel, replay, return-oriented attacks, malicious code), K0234 - Knowledge of full spectrum cyber capabilities (e.g., defense, attack, exploitation). K0235 - Knowledge of how to leverage research and development centers, think tanks, academic research, and industry systems, K0236 - Knowledge of how to utilize Hadoop, Java, Python, SQL, Hive, and Pig to explore data, and K0238 - Knowledge of machine learning theory and principles.

IV. Results & Discussion

A. Evaluation Metrics

In each of the algorithms used the precision, recall, F1 score, support, macro average, weighted average, and the confusion matrix was determined.

1) *Precision*: is the ratio of tp to tp + fp where tp represents the number of true positives and fp represents the number of false positives [21].
2) *Recall*: is calculated by the ratio tp to tp + fn, where tp represents the number of true positives and fn represents the number of false negatives [21].
3) *F1-score*: is the weighted harmonic mean of the precision and recall [21].
4) *Weighted average*: is calculated by calculating the metrics of each label and then determining the average weighted by support [21].
5) *Confusion matrix*: is defined as a matrix C such that Cij is equal to the observations that are known to be in group i but predicted to be group j [22].

B. Results

Figure. 2 show a sample classification report and confusion matrix of Extreme Gradient Boosting algorithm used in this research effort.

```
Classification report

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        76
           1       1.00      1.00      1.00        99

    accuracy                           1.00       175
   macro avg       1.00      1.00      1.00       175
weighted avg       1.00      1.00      1.00       175

Confusion Matrix

         Predicted Class 0  Predicted Class 1
Class 0                 76                  0
Class 1                  0                 99
```

Fig. 2. Classification report and Confusion matrix for Extreme Gradient Boosting algorithm

Table I below summarizes the evaluation results on different metrics used by this machine learning model setup. On this test run, there are a total of 175 support data points, with 76 for class 0 (non-malicious) and 99 for class 1 (malicious). Based on the results, 3 out of 5 algorithms tested yielded 100% accuracy, except for Naïve Bayes (77%) and Support Vector Machine (57%).

TABLE I. EVALUATION RESULTS

Algorithms	**Metrics (Class0/Class1)**				
	Precision	*Recall*	*F1-score*	*Weighted average*	*Accuracy*
Naïve Bayes	1.00/0.71	0.46/1.00	0.63/0.83	0.74	0.77
Logistic Regression	1.00/1.00	1.00/1.00	1.00/1.00	1.00	1.00
SVM	0.00/0.57	0.00/1.00	0.00/0.72	0.41	0.57
Rando Forest	1.00/1.00	1.00/1.00	1.00/1.00	1.00	1.00
Extreme Gradient Boosting	1.00/1.00	1.00/1.00	1.00/1.00	1.00	1.00

V. CONCLUSION & FUTURE WORK

Through this research effort, 1) An experimental setup to run different machine learning algorithms to detect SQL Injection attacks was developed; 2) Research results produced can be used by the research community working on Cyberattacks; 3) Accuracy of the machine learning algorithms used in research were determined; and 4) Research has the potential to be expanded in the future by adding more machine learning algorithms.

This research effort has led to a novel approach in terms of predicting SQLi attacks using machine learning algorithm and provides an alternative approach to the traditional models like AMNESIA, SQLrand, and SQLdom. This research effort if continued in the right direction might result in a SQLi attacks detecting model that can be used across different platforms to detect and prevent SQLi attacks from happening in the future. The results obtained through this research effort are preliminary and needs to be optimized for better results. The experimental model developed must be tested out with larger and more diverse datasets to improve the reliability and accuracy. Furthermore, a more powerful and cutting-edge methodology like deep learning will be taken into consideration, which would improve the automation and longevity of the model developed.

REFERENCES

[1] Acunetix. "What Is SQL Injection (SQLi) and How to Fix It." Web.

[2] Tajpour, Atefeh, Suhaimi Ibrahim, and Mohammad Sharifi. "Web application security by sql injection detectiontools." IJCSI International Journal of Computer Science Issues 9.2 (2012): 332-339.

[3] Vinitha Subburaj, Daniel Thomas Loughran, MayarKefah Salih, "All About SQL Injection Attacks", CISSE 2018, New Orleans, LA.

[4] McClure, Russell A., and Ingolf H. Kruger. "SQL DOM: compile time checking of dynamic SQL statements." Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on. IEEE, 2005.

[5] Boyd, Stephen W., and Angelos D. Keromytis. "SQLrand: Preventing SQL injection attacks." International Conference on Applied Cryptography and Network Security. Springer, Berlin, Heidelberg, 2004.

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

[7] Alwan, Zainab S., and Manal F. Younis. "Detection and Prevention of SQL Injection Attack: A Survey." (2017).

[8] Joshi, A., & Geetha, V. (2014). SQL Injection detection using machine learning. Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on, 1111-1115.

[9] Valeur, Fredrik, Darren Mutz, and Giovanni Vigna. "A learning-based approach to the detection of SQL attacks." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Berlin, Heidelberg, 2005.

[10] Ladole, Aniruddh, and DA, Phalke. "SQL Injection Attack and User Behavior Detection by Using Query Tree Fisher Score and SVM Classification." International Research Journal of Engineering and Technology 3.6 (2016).

[11] Rawat, R. & Kumar, S. (2012). SQL injection attack detection using SVM. International Journal of Computer Applications.

[12] Chen, Zhuang, and Min Guo. "Research on SQL injection detection technology based on SVM." MATEC Web of Conferences. Vol. 173. EDP Sciences, 2018.

[13] Gu, Haifeng, et al. "DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data." IEEE Transactions on Reliability (2019).

[14] Das, Debasish, Utpal Sharma, and D. K. Bhattacharyya. "Defeating SQL injection attack in authentication security: an experimental study." International Journal of Information Security 18.1 (2019): 1-22.

[15] https://github.com/client9/libinjection.git

[16] https://archive.ics.uci.edu/ml/machine-learning-databases/00237

[17] https://www.kaggle.com/wjburns/common-password-list-rockyoutxt

[18] https://www.kaggle.com/hackerrank/developer-survey-2018

[19] https://www.kaggle.com/siddharthkumar25/malicious-and-benign-urls

[20] Newhouse, William, et al. "National initiative for cybersecurity education (NICE) cybersecurity workforce framework." NIST Special Publication 800.2017 (2017): 181.

[21] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.precision_recall_ fscore_support

[22] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_mat