A Novel Approach Exploiting Machine Learning to Detect SQLi Attacks

Ahmed Abadulla Ashlam
Department of Computer Science
University Of Reading
Reading, UK
a.ashlam@pgr.reading.ac.uk

Atta Badii
Department of Computer Science
University Of Reading
Reading, UK
atta.badii@reading.ac.uk

Frederic Stahl
German Research Centre for Artificial Intelligence GmbH (DFKI)
Laboratory Niedersachsen, Marine Perception, 26129 Oldenburg, Germany
frederic_theodor.stahl@dfki.de

2022 5th International Conference on Advanced Systems and Emergent Technologies (IC_ASET) | 978-1-6654-2735-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/IC_ASET53395.2022.9765948

***Abstract*— The increasing use of Information Technology applications in the distributed environment is increasing security exploits. Information about vulnerabilities is also available on the open web in an unstructured format that developers can take advantage of to fix vulnerabilities in their IT applications. SQL injection (SQLi) attacks are frequently launched with the objective of exfiltration of data typically through targeting the back-end server organisations to compromise their customer databases. There have been a number of high profile attacks against large enterprises in recent years. With the ever-increasing growth of online trading, it is possible to see how SQLi attacks can continue to be one of the leading routes for cyber-attacks in the future, as indicated by findings reported in OWASP. Various machine learning and deep learning algorithms have been applied to detect and prevent these attacks. However, such preventive attempts have not limited the incidence of cyber-attacks and the resulting compromised database as reported by (CVE) repository. In this paper, the potential of using data mining approaches is pursued in order to enhance the efficacy of SQL injection safeguarding measures by reducing the false-positive rates in SQLi detection. The proposed approach uses CountVectorizer to extract features and then apply various supervised machine-learning models to automate the classification of SQLi. The model that returns the highest accuracy has been chosen among available models. Also a new model has been created PALOSDM (Performance analysis and Iterative optimisation of the SQLI Detection Model) for reducing false-positive rate and false-negative rate. The detection rate accuracy has also been improved significantly from a baseline of 94% up to 99%.**

Keywords*— *Data mining, OWASP, SQL injection, attacks, false positive, false negative, CountVectorizer.

I. INTRODUCTION

Global uptake of Communication technology and the resulting hyper-connectivity of people and the internet of things has massively increased the information flows along with a significant increase in the number of users of the internet and information technology, and all has rendered the system exposed to the emergence of negative practices for some users, and cyber-attacks, crime and fraudulent activities exploiting the online channels. SQLi is one of the oldest of these attacks, but it remains a serious threat to the confidentiality, integrity, and availability of web databases to this day. The top 10 web application vulnerabilities reported by OWASP (The Open Web Application Security Project) in the past five reports show that SQLi ranks first in vulnerabilities [1]. Structured Query Language is a set of instructions used for manipulating and accessing data in the database [2]. Web programming languages such as JAVA and PHP can provide several ways to create and execute SQL statements. [3]. Developers typically generate SQL statements through cascading strings that users sent from web pages. Due to the great diversity of SQL languages, there are a very large number of coding methods for creating SQL statements, thus given such automated SQLi attack facility with pervasive capabilities, it is possible to attack any systems anywhere by creating SQL statements [4]. During a SQLi attack, the attacker inserts pieces of malicious code into the request transaction, causing the server to execute illegal queries, resulting in data leakage and database corruption. For instance, an attacker can obtain private data such as usernames and passwords of website users through SQL injection, which poses a serious data security threat [5, 6]. "Fig. 1" shows how an attacker can send malicious statements to the application, following the injection of user input. Over the recent years, computer security has faced severe difficulties arising from theft in application-layer-based attacks such as credit card details [7]. The lack of assurance of the authenticity of the data entered by the user is one of the most common weaknesses in the security of web applications, as well as the design environment (including the vulnerabilities that mean that the attack pre-conditions are satisfied). Almost all major vulnerabilities of web applications are due to this weakness that leaves the system exposed to the risk of various SQLi attacks such as interpreter injection, cross-site scripting injection, buffer overflows, and SQLi attack [8].

It has become essential to provide a comprehensive architecture to detect and prevent all types of SQLi attacks with the ability to update when a new type of attack appears. Thus, building a model to detect all types of SQLi attacks is also a challenge due to the diversity and complexities of representations of such attacks. On the other hand, the observed feature set from available datasets tends, display critical sensitivity with respect of only particular SQLI attacks; this makes it rather challenging to optimise the feature extraction and training to build models for detection of hybrid attacks. In addition, when various types of attack are detected separately, the features that do not have a high significance, would need to be excluded although when several attacks are detected together by classical machine learning methods, the models tend to provide correct results. Various machine learning and deep learning algorithms have been applied to detect and prevent these attacks. However, such preventive attempts have not limited the incidence of cyber-attacks and the resulting compromised database as reported by the Common Vulnerabilities and Exposures (CVE) repository [22].

In this study, based on extensive exploratory data analysis at the application payload level and feature vector optimisation and machine learning a novel SQLi attack detection with enhanced performance over the state-of-the-art has been developed and validated. The feature extraction and coding has deployed CountVectorizer. The comparative analysis of performance of various models as carried out in this study has included SVM, Decision Tree, KNeighbours,

978-1-6654-2735-7/22/$31.00 ©2022 IEEE

AdaBoost, and Random Forest to detect SQLi attacks. These machine learning methods provide highly accurate predictions on test data. The innovation.

In this study is based on the feature engineering process performed on the payloads as well as the evaluation of different machine learning models based on these features. A PALOSDM was used to reduce the false positive rate and the false negative rate generated by the machine learning module. The results show that the models achieved more than 99% accuracy. Additionally, other performance metrics such as Confusion Matrix and Sensitivity-Specificity Analysis have been established for the proposed algorithm. The remainder of this paper is organised as follows. Section II reviews the related work; section III presents the research methodology to develop the SQLi detection solution. In Section IV, the experimental setup is described, and analysis of results; finally, Section VI concludes the paper.

II. Related Work

The first research project to classify security in operating systems and software was carried out in the 1970s. [9] Provided a classification of security vulnerabilities in an operating system. An error in categorisation was made by Bisbey and Hollingworth. Both studies examined security flaws in their operating systems. Security vulnerabilities such as SQLi attacks had not yet emerged that time [10].

More recently, classifications of security vulnerabilities has been provided by the Open Web Application Security Project (OWASP) [1] and Common Weakness Enumeration (CWE) whereby SQL injection attacks have been prominently placed. Have been developed. In a relational database, Structured Query Language enables us to query, add, modify, and delete data. A typical service-oriented architecture relies on a backend server response based on a database and in SQL statements the server becomes the means to interact with the database. Code generation is one of the main causes of SQL vulnerabilities due to the practice of passing SQL statements to the database, which are created by the concatenation method. As a result, attackers are able to rapidly create new SQL statements by simply entering SQL codes or keywords to iterate through and change the SQL statement so as to realise an attack. Based on the outcome of the investigation, it is evident that the root cause of system exposure to the risk of attacks is that developers put too much trust in the data provided by the users, by not filtering the data entered by the users. If the server-side fails to perform a reasonable check in order to discover the intent of a SQL statement, then this enables the attacker to take control of the server. The system thus becomes vulnerable to various types of attack, notably SQL injection for exfiltration of data and/or corruption of the database and indeed also DDOS attacks.

In “Fig. 1”, the principles and operational basis of the SQLi attack are illustrated [11].

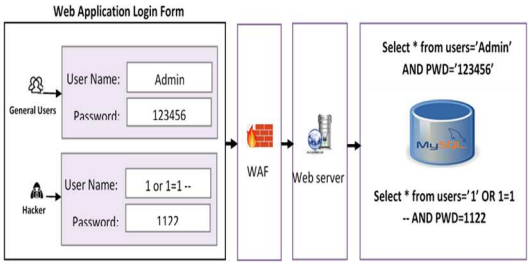

Fig. 1. SQL Injection Attack Process

In order to prevent SQLi attacks, most techniques use the stored procedures to enable access to the database. Parsing, dynamic tainting, pattern matching, information flow analysis, defensive coding and penetration testing are the most frequently used techniques to detect as well as prevent a vulnerabilities subset that might cause SQLi attacks. Some promising results with regard to the use of machine learning, data mining and research on text analysis methods for SQL injection detection have been achieved [13]. In fact, stored procedures rely on the implementation of such techniques to provide their protection against SQLi using stored procedures which however is not guaranteed to offer full protection For instance, applications use stored procedures, escaping of characters, and different forms of input validation in certain approaches, these usages could be vulnerable to SQLi attacks [14]. Huang et al. proposed WAVES, a black box technique used to test for SQLi vulnerabilities in web applications.

Machine learning is used to guide testing to improve general penetration testing techniques, but like all black box testing techniques, the use of these based on static analysis cannot provide security guarantees [15]. However, properties have been used instead of taxonomy in order to classify security flaws by Seacord et al. Their approach allocates properties to parts of the source code; then security flaws are classified based on those properties. Therefore, properties that can be matched successfully can be classified as known categories [16]. A classification system based on machine learning is proposed to detect and prevent SQLi attacks by monitoring datasets collected from known attacks reported in the literature. Then, the proposed classification system tests the dataset by checking the signature list to verify and validate the results of the token-based stage. The classification system tests a support vector machine algorithm for preventing malicious web requests from entering the target backend database. The advantage of the proposed work is that it is performed on a huge data set; however, it lacks implementation for sufficiently diverse set of machine learning classifiers. In addition, the authors did not provide any models to test their system [17]. The database has suffered much damage and changes resulting from this attack. Since it is difficult to identify unknown attacks using the current raw formatting method, a SQLi attack detection strategy based on machine learning has been proposed [18]. It is recommended to use a new query transformation system and hashing technology. This technique is a lightweight way to prevent SQLi attacks. When applied to an e-commerce application prototype,

experimental results show that it can successfully and effectively block various SQLi attack instances. This method can also be easily implemented in any language or on any database platform with only minor modifications. Large web applications have hundreds of places where users can enter data, and each location can provide opportunities for SQLi. Attackers can steal the confidential data of the organisation through these attacks, resulting in the loss of the market value of the organisation [19].

III. METHODOLOGY

The research followed a statistical, analytical and empirical methodology, whereby the severity (likelihood and impacts) of various SQLi attacks was studied accordingly, an analysis of the modus operandi and possible impacts of, various SQL injection types of attack that websites and applications are exposed to, was performed. Hence, datasets of the latest instances of SQL injection attacks were obtained. Subsequently, the system was designed and implemented as illustrated in "Fig. 2", which presents the workflow of the proposed method; this includes: Data Collection, Preprocessing, Feature extraction, CountVectorizer, a Classification Model, Performance Analysis, PALOSDM, Recalculation of the Confusion Matrix and Results.

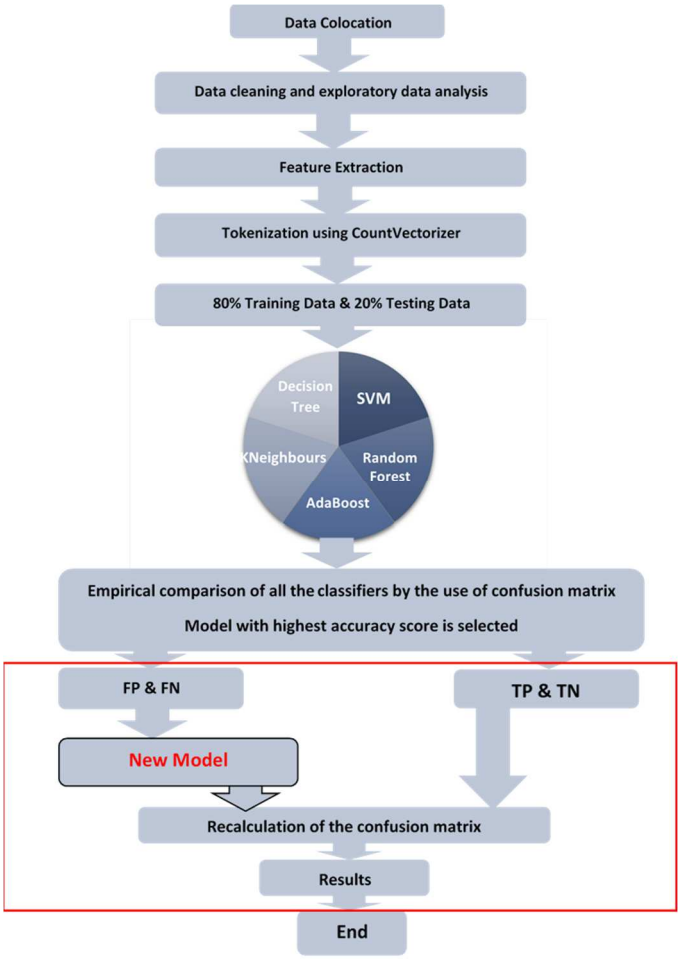

Fig. 2. Workflow of the proposed method

A. Data Collection

Gathering a dataset for SQLi attacks is challenging since no data sets were available with public access to the actual SQLi attacks launched. Due to this, the dataset has been taken from the National Vulnerability Database [20] and a python library named Lib-Injection [21]. Libinjection is an open-source tool which is used for web-applications penetration testing, which have data on all categories of SQLi attacks. Moreover, normal SQL queries have been added to this dataset for discriminating a normal text from SQLi queries. This was to balance the dataset to sufficient size to enable the optimal training of the model so that all types of SQL injections could be correctly identified. Accordingly, model development was implemented with the resulting dataset of 1950 SQLI queries and 2000 normal SQL queries.

B. Data Pre-processing

Tokens and stop words removal is a process in which a text is split into individual words to be processed to enable carrying out certain operations on words. This process is normally called tokenisation. There is also another process named stop words which includes words that have no intrinsic contextual meaning. Those words do not change the actual sentence, so these were removed in order to improve the model performance.

C. Feature extraction

Malware feature extraction for classifier model development requires an epistemological analysis of the feature space with an ontological commitment to the particular attack-tree-theoretic ontology of the preconditions and triggers of the attack as well as its dynamic behavioural model [19]. Segmentation is a process that enables users to understand the real meaning of all labels in the presented dataset. Word segmentation enables the translation of a text for input to a machine learning problem. Thus, another preprocessing step to enable the modelling process was the vectorization of the samples. The first step in this process was to quantify the words, for which one-hot code is considered as the most common method. CountVectorizer is a technique that transforms a corpus of text into a vector used in this vectorization process. This permits the pre-processing of a text before generating the vector representation; it was used to transform the word list to a long vector with only one dimension as 1s, others as 0s.

D. Classification Model

Five machine-learning algorithms namely SVM, Decision Tree, K-Nearest Neighbors, AdaBoost, and Random Forest were implemented using Python to train the dataset. After the classification, the models was re-trained to reduce the false positive and false negative results. The performance of each classifier was optimised and the most accurate classifier was identified as SVM with an accuracy of 94% and lowest false positive and false negative.

E. Performance analysis and Iterative optimisation of the SQLI Detection Model (PALOSDM)

In this phase four indicators of the algorithm with the highest performance were acted on to execute the supplementary training for the enhanced model namely True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values. The FP and FN represented the SQL statements that had been classified incorrectly, and which were used for re-training the algorithm. The algorithm was re-tested, and the resulting confusion matrix enabled the iterative re-training of the model to arrive at the finally optimised process.

IV. THE RESULTS AND DISCUSSION

The accuracy of the results of any machine learning-based system is measured based on the True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) values in the results generated through the testing the model with the TP and TN being the correctly results and FP and FN representing the incorrect ones and the higher the TP and TN values, the more accurate the classification. The confusion matrix shown in Table 1 presents the result of the model development and testing to identify the optimal supervised machine-learning model for the detection of SQLi. After vectorization, 80% was used for training the dataset with the remaining randomly selected 20% of the data instances set aside for training. As can be seen in Table 2 below, the Support Vector Machine performed with the highest accuracy (94%), precision (96%), recall (91%), F1-Score (94%), and specificity 96% (Table 4). However, these preventative attempts have not limited the consequences and damage to attacked databases. However, it was proposed to seek further improvements upon the results shown in (Table 4), through avoidance of any overfitting to reduce the error rate further. Using the new method lead to enhanced performance of the model, raising the detection rate from a baseline of about 94% to 99% accuracy as illustrated in (Table 3).

Table 1. The guide of confusion matrix

TP Classified correctly	FP Classified incorrectly
FN Classified incorrectly	TN Classified correctly

Table 2: Accuracy comparison of different ML Classifiers

Model	Accuracy score
SVM	94%
Decision Tree	93%
KNeighbours	92%
AdaBoost	92%
Random Forest	90%

Table 3: Comparing accuracy (SVM and SVM with PALOSDM)

Classifier	**Confusion Matrix**		**Accuracy score**
SVM	91	3	94%
	8	89	
SVM with PALOSDM	99	1	99%
	0	91	

Table 4: The improvements upon the results

Model	**SVM**	**SVM with PALOSDM**
Accuracy score	94%	99%
Precision	96%	1
Recall	91%	98%
F1	94%	99%
Specificity	96%	1

Importantly, by resulting in lower False Negative (FP) and False Positive (FP), the method has enhanced the overall reliability of the SQLi detection using the proposed approach. In “Fig. 3”, the comparison of SVM classification results and the results SVM with PALOSDM.

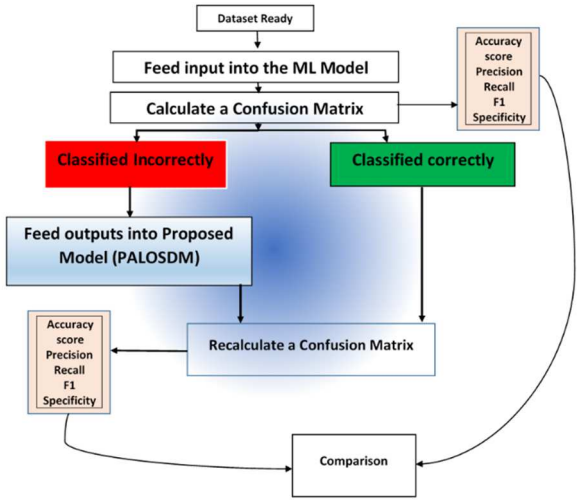

Fig. 3 comparing accuracy (SVM and SVM with PALOSDM)

V. CONCLUSION AND FUTURE WORK

According to reports from the security consortiums, OWASP, injection vulnerabilities remain the most common and dangerous attacks on web applications. SQLi attacks, were selected as the focus of this research study given that these are increasing sharply with some malicious content for unrestricted access to databases, exfiltration and extraction of sensitive information. New technologies have emerged to deal with such attacks. However, it is difficult to find the most suitable solution for a particular web application. Therefore, a new model (PALOSDM) has been proposed to address the difficulties in SQLi classifier model development so as to achieve enhanced performance in SQLi detection. The first step in this work was to create a balanced dataset containing both normal and malicious SQL queries. In order to obtain the best possible results and distinguish malicious queries from normal text, the data was subjected to a set of machine learning algorithms, which achieved the highest accuracy rate of 0.94. Subsequently, PALOSDM was proposed to enhance the accuracy of the results. This provided for an accuracy of more than 99% with a small error rate approximately, which proved to be very effective in order to distinguish between SQLi attack queries from normal (non-attack) text input and SQL queries.

This method has been used on static stored data, the proposed future work is to adapt this method to be used on real-time data.

REFERENCES

[1] OWASP, "OWASP Top Ten," 2021. [Online]. Available: https://owasp.org/Top10.

[2] D. Mitropoulos, P. Louridas, M. Polychronakis, and A. D. Keromytis, "Defending against web application attacks: approaches,

challenges and implications," IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 2, pp. 188-203, 2017.

[3] E. Pollack, "Protecting Against SQL Injection," in Dynamic SQL: Springer, 2019, pp. 31-60.

[4] Q. Li, W. Li, J. Wang, and M. Cheng, "A SQL injection detection method based on adaptive deep forest," IEEE Access, vol. 7, pp. 145385-145394, 2019.

[5] A. Maraj, E. Rogova, G. Jakupi, and X. Grajqevci, "Testing techniques and analysis of SQL injection attacks," in 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA), 2017: IEEE, pp. 55-59.

[6] D. Das, U. Sharma, and D. Bhattacharyya, "Defeating SQL injection attack in authentication security: an experimental study," International Journal of Information Security, vol. 18, no. 1, pp. 1-22, 2019.

[7] A. Saha and S. Sanyal, "Application layer intrusion detection with combination of explicit-rule-based and machine learning algorithms and deployment in cyber-defence program," arXiv preprint arXiv:1411.3089, 2014.

[8] D. Tripathy, R. Gohil, and T. Halabi, "Detecting SQL injection attacks in cloud SaaS using machine learning," in 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), 2020: IEEE, pp. 145-150.

[9] R. P. Abbott, J. S. Chin, J. E. Donnelley, W. L. Konigsford, S. Tokubo, and D. A. Webb, "Security analysis and enhancements of computer operating systems," NATIONAL BUREAU OF STANDARDS WASHINGTONDC INST FOR COMPUTER SCIENCES AND …, 1976.

[10] R. Bisbey and D. Hollingworth, "Protection analysis: Final report," ISI/SR-78-13, Information Sciences Inst, vol. 3, 1978.

[11] D. Chen, Q. Yan, C. Wu, and J. Zhao, "Sql injection attack detection and prevention techniques using deep learning," in Journal of Physics: Conference Series, 2021, vol. 1757, no. 1: IOP Publishing, p. 012055.

[12] S. McDonald, "SQL Injection: Modes of attack, defense, and why it matters," White paper, GovernmentSecurity. org, 2002.

[13] C. Anley, "Advanced SQL injection in SQL server applications," 2002.

[14] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, "Web application security assessment by fault injection and behavior monitoring," in Proceedings of the 12th international conference on World Wide Web, 2003, pp. 148-159.

[15] R. C. Seacord and A. D. Householder, "A structured approach to classifying security vulnerabilities," CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2005.

[16] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention," in 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017: IEEE, pp. 1087-1090.

[17] A. Sivasangari, J. Jyotsna, and K. Pravalika, "SQL Injection Attack Detection using Machine Learning Algorithm," in 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021: IEEE, pp. 1166-1169.

[18] D. Kar and S. Panigrahi, "Prevention of SQL Injection attack using query transformation and hashing," in 2013 3rd IEEE International Advance Computing Conference (IACC), 2013: IEEE, pp. 1317-1323.

[19] A. Badii and D. Patel, "Evolving features-algorithms knowledge map to support NIDS data intelligence and learning loop architecting–a generalised approach to NIDS pattern feature space knowledge processing," 2008.

[20] U.S. government, "National Vulnerability Database " 2021. [Online]. Available: https://nvd.nist.gov/.

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

[22] U.S. Department of Homeland Security “Common Vulnerabilities and Exposures," 2021. [Online]. Available: https://cve.mitre.org/.