Journal of Big Data

RESEARCH Open Access

Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques

Wubetu Barud Demilie[1*] and Fitsum Gizachew Deriba[2]

*Correspondence:
wubetubarud@gmail.com;
wubetuB@wcu.edu.et

[1] Department of Information Technology, Wachemo University, Hossana, Ethiopia
[2] Department of Computer Science, Wachemo University, Hossana, Ethiopia

Abstract

A web application is a software system that provides an interface to its users through a web browser on any operating system (OS). Despite their growing popularity, web application security threats have become more diverse, resulting in more severe damage. Malware attacks, particularly SQLI attacks, are common in poorly designed web applications. This vulnerability has been known for more than two decades and is still a source of concern. Accordingly, different techniques have been proposed to counter SQLI attacks. However, the majority of them either fail to cover the entire scope of the problem. The structured query language injection (SQLI) attack is among the most harmful online application attacks and often happens when the attacker(s) alter (modify), remove (delete), read, and copy data from database servers. All facets of security, including confidentiality, data integrity, and data availability, can be impacted by a successful SQLI attack. This paper investigates common SQLI attack forms, mechanisms, and a method of identifying, detecting, and preventing them based on the existence of the SQL query. Here, we have developed a comprehensive framework for detecting and preventing the effectiveness of techniques that address specific issues following the essence of the SQLI attacks by using traditional Navies Bayes (NB), Decision Trees (DT), Support Vectors Machine (SVM), Random Forests (RF), Logistic Regression (LR), and Neural Networks Based on Multilayer Perceptron (MLP), and hybrid approach are used for our study. The machine learning (ML) algorithms were implemented using the Keras library, while the classical methods were implemented using the Tensor Flow-Learn package. For this proposed research work, we gathered 54,306 pieces of data from weblogs, cookies, session usage, and from HTTP (S) request files to train and test our model. The performance evaluation results for training set in metrics such as the hybrid approach (ANN and SVM) perform better accuracies in precision (99.05% and 99.54%), recall (99.65% and 99.61%), f1-score (99.35% and 99.57%), and training set (99.20% and 99.60%) respectively than other ML approaches. However, their training time is too high (i.e., 19.62 and 26.16 s respectively) for NB and RF. Accordingly, the NB technique performs poorly in accuracy, precision, recall, f1-score, training set evaluation metrics, and best in training time. Additionally, the performance evaluation results for test set in metrics such as hybrid approach (ANN and SVM) perform better accuracies in precision (98.87% and 99.20%), recall (99.13% and 99.47%), f1-score (99.00% and 99.33%) and test set (98.70% and 99.40%) respectively than other ML approaches.

However, their test time is too high (i.e., 11.76 and 15.33 ms respectively). Accordingly, the NB technique performs poorly in accuracy, precision, recall, f1-score, test set evaluation metrics, and best in training time. Here, among the implemented ML techniques, SVM and ANN are weak learners. The achieved performance evaluation results indicated that the proposed SQLI attack detection and prevention mechanism has been improved over the previously implemented techniques in the theme. Finally, in this paper, we aimed to keep researchers up-to-date, with contributions, and recommendations to the understanding of the intersection between SQLI attacks and prevention in the artificial intelligence (AI) field.

Keywords: Deep learning, Detection, Hybrid, Machine learning, Prevention, SQLI attack, Web application

Introduction

Malware attacks, particularly SQLI attacks, are common in poorly designed web applications. This vulnerability has been known for more than two decades and is still a source of concern [1]. For many years, structured query language (SQL) has been the industry standard for dealing with relational database management systems (DBMS). Since the majority of applications for cyber-physical systems are safety–critical; misbehavior brought on by random errors or online attacks can severely limit their development [2, 3]. Therefore, it's crucial to safeguard cyber-physical systems from suffering this kind of attack.

SQLI attacks on data-driven web applications and systems, also known as SQLI attacks, have been a serious problem since it became common for internet web applications and SQL databases to be connected [4, 5, 6]. An SQLI attack occurs when an attacker takes advantage of a flaw in the web application's SQL implementation by submitting a malicious SQL statement through a fillable field. In other words, the attacker will insert code into a field to dump or alter data or gain access to the backend. As explained by [6] and [7], SQLI is a common attack vector that allows malicious SQL code to access hidden information by manipulating database backends and is regarded as one of the most dangerous injection attacks because it jeopardizes the main security services such as confidentiality, authentication, authorization, and integrity [8, 9]. This information could include sensitive business information, private customer information, or user lists. A successful SQLI attacker can lead to the deletion of entire databases, the unauthorized use of sensitive data, and the unintentional grant of administrative rights to a database.

The increased development and spread of web applications have also increased the number and severity of web attacks [10, 11]. According to [12], the most common vulnerability in web applications is injection. Injection attacks take advantage of a variety of flaws to deliver untrusted user input, which is then processed by a web application [13]. The SQLI attacks entail injecting (inserting) malicious SQL commands into input forms or queries to gain access to a database or manipulate its data (e.g. send the database contents to the attacker, modify or delete the database content, etc.) [14, 15]. Undeniably, most web applications today rely on a back-end database to store data collected from users and/or to retrieve information selected by users [16].

Forms and cookies are commonly used to interact with these users. Different hackers attempt to exploit this feature by injecting malicious code into the user inputs that will later be used to construct the SQL queries. Improper validation of user inputs can result in the success of the SQLI attack, which can have disastrous consequences such as the deletion of the database or the collection of sensitive and confidential data from web application clients [17]. Several research works have addressed the SQLI attack due to its sensitive impact. Some of these works only attempt to detect SQLI after it has occurred and other works have attempted to prevent it from happening in the first place.

In this study, we looked at SQLI attacks that try to bypass the web application firewall and gain unauthorized access to confidential data. These attacks target the HTTP or HTTPS protocol. The victim system is normally not prepared to handle this input, which frequently leads to data leakage and/or the attacker receiving unauthorized access. In this instance, the attacker has access to and/or control over the data, which has an impact on all areas of security, including data availability, confidentiality, and integrity [2].

The SQL query that was maliciously injected is intended to extract or modify data from the database server. Successful injection can cause data loss and/or the total database to be destroyed, as well as authentication, bypass, and modifications to the database by inserting, changing, and/or deleting data. Additionally, such an assault could take control of the hosted OS and run commands on it, usually having greater negative effects [2, 18]. Therefore, organizations are seriously threatened by SQLI assaults.

Even though several techniques have been proposed to combat SQLI attacks, none of these solutions have addressed the full scope of the attacks. As a result, there were no solutions that can prevent or detect all types of SQLI attacks. Recently, researchers have attempted to with AI integrated techniques including deep learning (DL), machine learning (ML), and hybrid techniques to propose more sophisticated solutions [19].

Here, learning from past data reflecting an attack and/or regular data is typically used to build AI approaches to help with threat detection and prevention. Historical information can be used to interpret detected traffic, identify attack patterns, and even forecast future assaults before they happen [2, 20].

To grasp how SQL language can be abused, SQLI attackers and defenders need to understand how it functions [2]. The queries must be prepared in the SQL language and adhere to a specified syntax to retrieve data from databases or modify the data, such as:

```
"SELECT * FROM authortable WHERE book_name = 'Advanced Database Systems'"
```

The aforementioned search will provide all books with book_name "Advanced Database Systems." The queries are typically typed into a web browser and sent to the database management system [2].

What if the attacker in this case extends the original SQL query?

For example:

```
"SELECT * FROM authortable WHERE book_name = 'Advanced Database Systems' OR
'1'='1'"
```

The above query will return all book names in the database, not just the book names labeled as "Advanced Database Systems," because the sentence '1 = 1' is always true. If the stored list of book names is not a secret and the previous example might not pose a problem [2]. If successful, it might return sensitive data, such as passwords, bank accounts, trade secrets, and personal information, which might be regarded as a privacy breach among other negative effects. However, it could be applied to value using different syntax.

According to certain studies, inserting code using 'OR' and a 'TRUE' assertion, such as '1 = 1', is referred to as "tautology" [21]. Other techniques besides tautology can be used, like when an attacker purposefully inserts an incorrect query to make the database server return a default error page. This default error page may contain important information that can help an attacker comprehend the database and craft a more sophisticated attack [2, 21]. In addition to numerous other techniques based on the same concept, incorrectly employing SQL syntax to extract or even edit the data in the targeted database, 'UNION' can also be used to extract information.

Given that SQLI operates in this manner, the question is how to recognize and stop this kind of attack by using DL, ML, and hybrid techniques. By instructing a classifier to develop the ability to recognize, detect, and subsequently prevent an attack, it can be used to support the detection and prevention of SQLI attacks. The classifier can be used to categorize new data, including traffic or data from log files, and is trained using various models. If the classifier is active, it will block data from reaching the database server; if it is passive, it will inform the administrator. Accordingly, three alternative learning techniques have been used to train the classifier to recognize and detect SQLI attacks [2, 22].

The first technique was unsupervised learning (UL), in which features were taken from data that had not been categorized, or data that had neither normal nor pathological labels. The classifier finds hidden structures in the unclassified dataset using information and the Bayesian probability theory. When data are unclassified, it is unclear if they are normal or abnormal (malicious). The UL can make use of a variety of methods, including clustering and density estimation [2, 22].

The classifier was trained using the second technique, supervised learning (SL), using a set of labeled training data. The output was known in advance because the input data were marked, i.e., normal or aberrant. To attain an acceptable classification accuracy, the procedure first comprises a straightforward mapping between the input training data and the known output, which was followed by continuous algorithm and weight change. The classifier was then tested using a test set of data; if the results were within an acceptable range of accuracy, the classifier was then ready to recognize novel data or data that had not been used in training or testing [2].

The fundamental problem with the SL was the time it took to create and label the training and test sets of data, especially for sophisticated attacks. The classification and regression techniques were used to classify the SL. The Bayesian networks, DT, SVM, K-nearest neighbors, and neural networks are some of the most used SL techniques. The third technique, known as semi-supervised learning, combines SL and UL techniques [2, 22]. Accordingly, by using SQLI, the attackers can alter the SQL statement by replacing the user's supplied data with their own data as depicted in Fig. 1.

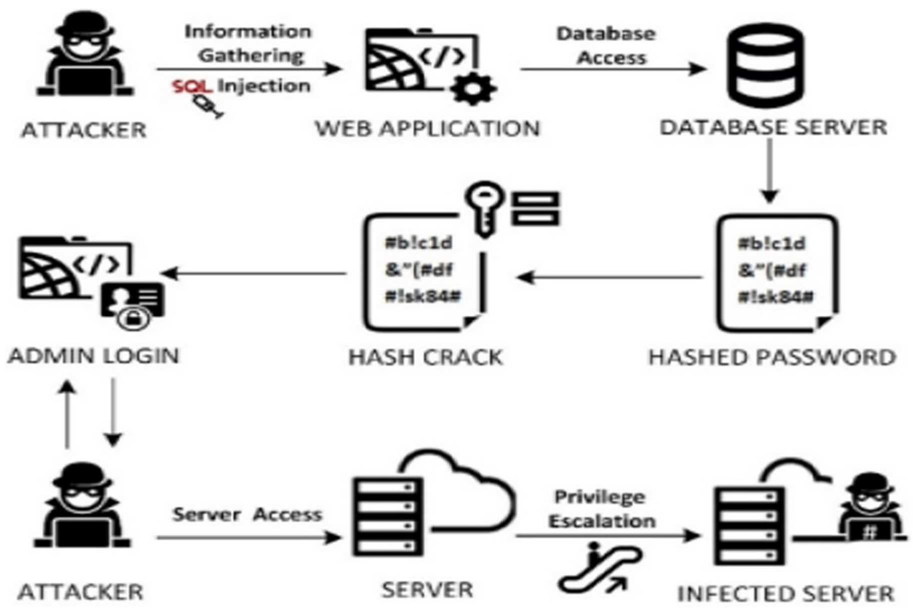

Fig. 1 The SQLI attacks [23]

For this proposed research work, we gathered 54,306 pieces of data from weblogs, cookies, session usage, and from HTTP (S) request files to train and test our model. We divided the acquired data into sections for our model's testing (30%) and training (70%). We used 38,014 of the total dataset for training and 16,292 for model testing. Among the datasets, 47,343 are genuine queries and 6,963 are malicious queries.

Accordingly, this work has the following major contributions.

- Review the recommended and state-of-the-art research works in DL, ML, and hybrid techniques for SQLI attacks that have been published in reputable databases.
- The different SQLI attack detection and prevention countermeasures are classified and discussed.
- Studying and identifying the nature of SQLI attacks and proposing prevention and detection mechanisms.
- Newly proposed solutions are described and discussed, such as those based on DL, ML, and hybrid techniques.
- Keep the researchers up-to-date, with contributions, and recommendations to the understanding of the intersection between SQLI attacks and prevention in the AI field.

Accordingly, the primary goal of this paper is to examine current SQLI attacks, identify their methodologies, strengths, and weaknesses, and finally propose a thorough detection and prevention method.

The rest of the paper is organized into different but interrelated sub-sections. The paper begins by discussing the related works in the "Related works" section, the SQLI query attacks overview in the "SQLI query attacks overview" section, existing methods for SQLI detection and prevention in the "Existing methods for SQLI detection and prevention" section, developing a web-based framework for SQLI attacks detection and prevention in "Developing a web-based framework for SQLI attacks detection and prevention" section, most common attacks on SQLI in "Most common attacks on SQLI" section, proposed frameworks for SQLI detection and prevention in "Proposed frameworks for SQLI detection and prevention" section, result and discussion in "Result and discussion" section, and the conclusion and recommendation in "Conclusion and recommendation" section.

Related works

In this section, different published research works have been considered and included to indicate the research gaps in the area. The paper typically reviews and includes studies that have been published in reputable databases. Many researchers have demonstrated the use of DL, ML, and hybrid techniques to detect SQLI attacks [23].

A review of SQLI prevention in web applications has been presented in [1]. The authors have provided a summary of 14 different varieties of SQLI attacks and how they affect online applications. Their research's main objective was to investigate alternative SQLI prevention strategies and to offer an analysis of the most effective defense against SQLI attacks.

Authors in [2] have conducted a systematic literature review of 36 articles related to research on SQLI attacks and ML techniques. To classify different varieties of SQLI attacks, they have identified the most widely used ML techniques. Their finding revealed that few studies generated new SQLI attack datasets using ML tools and techniques. Similarly, their results showed that only a few studies focused only on using mutation operators to generate adversarial SQLI attack queries. In future work, the researchers aimed to cover the use of other ML and DL techniques to generate and detect SQLI attacks.

A comprehensive study on SQLI attacks, their mode, detection, and prevention has been presented in [4]. The authors have identified how attackers of this kind might exploit such a weakness and execute weak code as well as a strategy to mitigate such detrimental effects on database systems. The researchers' investigation revealed that web operations were frequently used for online administrations ranging from high levels of informal communication to managing transaction accounts and dealing with sensitive user data. The real issue, however, was that this data was exposed to attacks because of unauthorized access, where the attackers gained entry to the system using various hacking and cracking techniques with very malicious motives. The attacker can use more sophisticated queries and creative tactics to get around authentication while also gaining total control over both the server and the web application. Many cutting-edge algorithms have been developed up to this point to encrypt data queries to defend against such attacks by structuring desirable query modification plans. In the paper, they worked together to discuss the history of injection attacks, different forms of injection attacks, various case studies, and defenses against SQLI attacks, along with an appropriate illustration.

In the work of [5], a survey on SQLI attack detection and prevention has been presented. The research, according to the authors, might help laypeople comprehend SQL and its hazards. It also helps researchers and programmers who wanted to learn about all the problems that still plague web applications and what strategies can be employed