Using Machine Learning for Detecting and Preventing SQL Injection Attacks

Ahamd Farid Aseel[1] [1]*M.Sc. Student, Information Technology Engineering, Faculty of Engineering, College of Farabi, University of Tehran, Iran*
faridaseel.4all@gmail.com

Abstract

This paper investigates SQL Injection attacks and the use of machine learning algorithms as a novel approach to prevent such attacks in databases. It begins by explaining various types of SQL Injection attacks and then introduces the concepts and algorithms of machine learning. The goal of this research is to provide effective solutions for detecting and preventing these attacks. Through examples and practical results, this paper addresses the enhancement of database security through machine learning algorithms. The results show that employing these algorithms can significantly improve database security and contribute to increasing awareness about innovative methods for combating SQL Injection attacks.

Keywords: *SQL Injection, Machine learning, Cybersecurity, Cyber-attack, Artificial intelligence.*

1 Introduction

SQL Injection attacks are one of the oldest and most dangerous threats to web applications. In this type of attack, the attacker injects unreliable inputs into the application, resulting in alterations to the database commands or queries. These alterations can lead to data theft, data loss, and compromise of data integrity.

In essence, in these types of attacks, the attacker sends inappropriate and unreliable inputs to the application. These inputs affect a portion of the command or query and modify the execution of the program. Most vulnerabilities in these types of attacks stem from the validation of inappropriate user inputs [1, 2].

The significance of this issue lies in the fact that attackers can extract sensitive information such as user data, passwords, and financial information from the database, or by altering SQL commands, they can destroy data or even lead to system destruction. However, the attacker may simply aim to gain control of the system without intending to destroy it [3].

Figure 1: SQL Injection Attack

The structure of this text is such that it initially describes various types of SQL Injection attacks, then, while examining past work, various machine learning methods related to the subject will be mentioned. Finally, an evaluation of the effectiveness of machine learning algorithms for detecting intrusions will be addressed.

2 SQL Injection Attacks

SQL Injection attack, as one of the prevalent threats in the digital world, employs malicious SQL codes to manipulate data in databases to gain access to information that was not intended to be accessed. This type of attack, by exploiting vulnerabilities present in database systems and injecting malicious codes, can easily jeopardize online systems. When an attacker successfully exploits SQL Injection, they can fraudulently obtain sensitive information and even gain full control over the database, leading to serious repercussions for organizations and online systems [2, 3].

This section of the article focuses on examining various types of SQL Injection attacks. A comprehensive study regarding the penetration methods of these attacks and their different forms is discussed herein. The aim of this section is to provide an explanation and extensive understanding of SQL Injection attacks to better comprehend cyber threats and raise awareness regarding possible countermeasures against such attacks.

3 Types Of SQL Injection Attacks (SQLIA)

In web applications, most activities involve accessing information from databases. If the data entered by users is not properly validated and authenticated, individuals can gain

Table 1: Example of Injection

admin	admin is Username
`' OR 'x'='x`	`SELECT name FROM member WHERE username='admin' AND password= ' ' OR 'x'='x'`
`' OR 'x'='x ' OR 'x'='x`	`SELECT name FROM member WHERE username =' ' OR 'x'='x' AND password=' ' OR 'x'='x'`
`'OR 'x'='x' - -`	`SELECT name FROM member WHERE username =' ' OR 'x'='x' - - ' AND password=' '`

access to information they were not intended to see. Various methods exist to execute SQL Injection attacks [9].

3.1 Tautologies

Tautology is a type of SQL data structure tampering attack wherein hackers attempt to bypass validations, identify input parameters, and/or extract information from the desired database using WHERE clause conditions that are always true in every interpretation. For instance: `"WHERE password = 'x' OR 'x' = 'x'"` or `"WHERE password = 'x' OR 1=1"`. Therefore, possible signatures for this type of attack include string terminator `" ' "`, `OR`, `=`, `LIKE`, and `SELECT`. Mitigating tautology attacks in SQL data structure tampering attacks can be achieved by precise validation of user inputs on the user side and blocking queries containing tautological conditions in WHERE clauses on the database side [5, 6].

This query is always true because it is augmented with the tautology expression (`'x'='x'`). The double dashes `"--"` indicate to the SQL interpreter software that the rest of the statement is a comment and should not be executed as part of the new command sent to the database or in the execution of stored procedures. It is worth noting that many databases do not require a special character to separate distinct queries, so essentially checking for exceptional or special characters is not an effective

way to prevent these types of attacks.

3.2 Inference

In this type of attack, attackers design queries that, when executed, alter the behavior of the application or database. They use the responses received from the database to modify the query method. This type of attack is based on a rewrite that is executed based on the correct or incorrect response to a question about data values in the database. Typically, attackers target a site that appears to be sufficiently secure, so that when injection is successful, sufficient information is not available through database error messages. Therefore, attackers use various methods to obtain responses from the database. They inject their commands into the site and observe the website's reaction behavior to determine whether these changes indicate any vulnerabilities in the site's parameters or not. This method allows the attacker to gain access to database data and identify vulnerable parameters.

Two well-known attack techniques based on inference that allow attackers to extract data from the database and identify vulnerable parameters are Blind Injection and Timing attacks [5, 6, 7].

3.3 Blind Injection

Developers remove details of error messages. These messages help attackers to infiltrate databases. In this case, attackers attempt to penetrate the database using vulnerability query statements that have logical results. Then, they analyze the differences based on program responses [6, 4].

3.4 Timing Attacks

A timing attack allows an attacker to extract information from a database by observing time delays in the database response. This type of attack is very similar to blind injection but employs a different method. To conduct a timing attack, attackers structure their injection query as an if/then statement, where the conditional branches relate to unknown information about the database content. In one of the branches, the attacker uses an SQL structure that requires a specific time to execute (such as a watch key that causes a delay in the database response). By measuring the increase or decrease in database response time, the attacker can infer which branch in their injection has been executed, and therefore what the injected query's response is [4, 6, 8].

Now, let's consider Table 2. In the first scenario, we have a secure program, and inputs for system entry are properly validated. In this case, both injections return a system login error message, and the attacker understands that the system entry parameter is not vulnerable. In the second scenario, we have an insecure program, and the system entry parameter is injectable. The attacker sends the first query, and due

Table 2: Code Example

```
SELECT accounts FROM users WHERE login='legalUser' and 1=0 -
' AND pass='' AND pin=0
```

```
SELECT accounts FROM users WHERE login='legalUser' and 1=1 -
' AND pass='' AND pin=0
```

to the always-evaluating false condition, the program returns a system login error message. However, at this point, the attacker does not know whether this is because the program properly validated the input and blocked the attack attempt or if the attack itself caused the system login error. The attacker then sends the second query, which is always evaluated as true. If the system login error message is not needed in this case, the attacker realizes that the attack has been successful, and the system entry parameter is injectable.

3.5 Malformed Queries

In this method, when an attacker exploits incorrect or insufficient symbols in the SQL command, an error message is returned from the database containing useful information for troubleshooting. This error message enables attackers to accurately identify vulnerable parameters in the program and the overall database structure. This situation is exploited due to SQL commands designed by attackers or incomplete inputs that result in syntax errors, data type problems, or logical errors in the database. Syntax errors are used to identify injectable parameters. Also, data type errors may be used to infer specific information types or to delete used information. Logical errors may also reveal table names or features that cause errors or mistakes [8].

3.6 Union Query

In this technique, attackers merge an invalid statement with a valid query using the UNION keyword. This merging involves appending a query statement with the structure `"UNION <injected query>"` to the end of a valid statement as much as possible. This action causes the program to retrieve information from both the original query results and another table. Then, a statement with a double dash `"--"` as a comment existing within the query is deactivated. In this query, the original query returns an empty set while the manipulated query statement retrieves data from the same table [11, 8].

Table 3: Code Example

```
SELECT name FROM member WHERE username=''UNION
SELECT password FROM member WHERE username='admin' -- AND password=''
```

Table 4: Code Example

```
SELECT accounts FROM users WHERE login='admin' AND
pass=''  - ' AND pin=123; DROP table users
```

4 Piggy-backed Queries

In this type of attack, the attacker attempts to inject additional queries into the main query. We distinguish this type from others because attackers in this case are not seeking to modify the main query; instead, they try to add new and distinct queries piggybacked onto the main query. The result of this action is the execution of multiple SQL queries by the database. The first query is the main query that is executed normally; subsequent queries are the injected queries that are executed in addition to the main query. This type of attack can be highly destructive. Upon success, attackers can insert almost any type of SQL command, including stored procedures, into the additional queries and execute them along with the main query. Vulnerability to this type of attack usually depends on the configuration of the database allowing multiple commands to be received in a supplementary string [13].

Example: If the attacker enters the text `"'; drop table users --"` into the password field, the program generates the following query (table 4).

After executing the first query, the database recognizes the boundary marker (`";"`) and proceeds to execute the injected second query. The result of executing the second query is the deletion of the users table, potentially removing valuable information. Other types of queries may involve inserting new users into the database or executing stored procedures. Note: Many databases do not require a specific symbol to separate distinct queries, so searching only for a query separator is not an effective way to prevent this type of attack [10, 6, 5].

4.1 Stored Procedures

Due to the extensive capabilities provided by stored procedures in the database, SQL Injection Attack intrusion attempts of this type seek to execute these procedures in the database. Many database vendors provide standard stored procedures that extend database capabilities and provide interaction with the operating system. Therefore,

Table 5: Code Example

```
SELECT name FROM member WHERE username=''; SHUTDOWN; - - password=''
```

Table 6: Code Example

```
SELECT accounts FROM users WHERE login='legalUser';
exec(char(0x73687574646f776e)) -- AND pass='' AND pin=
```

whenever an attacker identifies which development the database is using, they can specifically design SQL Injection Attack intrusions to execute the stored procedures offered by that database, even procedures that interact with the operating system.

There is a common misconception that using stored procedures to write web applications protects them against SQL Injection Attack intrusions. Developers are often surprised that their stored procedures are vulnerable to attacks just like regular programs. Additionally, because stored procedures are often written in specific scripting languages, they may be susceptible to other types of vulnerabilities such as buffer overflows, allowing attackers to execute arbitrary code on the server or elevate their privileges [4, 12, 6, 8].

4.2 Alternate encoding

Alternate encoding SQL Injection attack is a type of attack where hackers attempt to conceal their injection commands using encoding techniques such as ASCII, hexadecimal, and Unicode. Thus, possible signatures for this attack include: exec(), Char(), ASCII(), BIN(), HEX(), UNHEX(), BASE64(), DEC(), ROT13(), and similar methods. Accurate validation of user inputs on the user side, for example, prohibiting the use of any metacharacters such as `"()Char"` and interpreting all metacharacters as regular characters on the database side, can prevent alternate encoding SQLi attacks. In terms of violating the three elements of the CIA triad, SQLi inference attack and alternate encoding are among the different classifications of SQLi. For example, the inference attack does not compromise information security but rather involves an initial data gathering operation by the attacker. Alternate encoding is a method to conceal SQLi attacks from other types. All the aforementioned attacks along with their signatures and prevention methods are listed in Table 1 [4]. Related works on SQLi attacks, detection, and prevention will be discussed in the next section [12].

This example utilizes the ()char function and hexadecimal encoded ASCII. The () char function, by taking an integer or hexadecimal encoded character, returns an instance of that character. The sequence of numbers in the injection part represents the hexadecimal encoding of the ASCII for the string `"SHUTDOWN"`. Therefore, when

interpreted by the database, the query concludes with the execution of the SHUTDOWN command by the database [6].

5 Related Works

A considerable number of studies and research have been conducted in the field of SQL injection penetration and its detection using various methods, including static and dynamic analysis, a combination of techniques, machine learning, hash techniques, etc. [15].

Static analysis examines whether each flow from a source to a precise location depends on confirming the information and also investigates the input sanitation method [15]. Meanwhile, dynamic analysis involves developing advanced query structure extraction for each data and identifying attacks by comparing them with the actual query structure given by the user [17].

AMNESIA, as an integrated method, is a model-based approach that integrates static and dynamic analysis for detecting and preventing SQL injection attacks. It utilizes static analysis to create SQL query models at the time of database access. It then utilizes dynamic analysis before sending queries to the database and compares them with the static models previously created. However, there are query generation methods and specific code snippets that reduce the efficiency of this model and increase the error rate [18].

The Hidden Markov Model (HMM) has been introduced for detecting malicious queries using machine learning in two phases: training and execution. The first phase focuses on collecting known malicious and benign queries, while the second phase concentrates on detecting injection attacks. The author themselves have stated that WHERE clauses and piggybacked queries cannot be identified by this model [19, 21].

Lambert et al. [20, 22] proposed a model that utilizes tokenization technique for detecting SQL injection attacks, hence queries containing aliases, samples, and set operations can also be blocked at the entry point. It examines whether the query generated based on user input yields its desired result and compares the results by applying tokenization technique on a main query and an input query. If the results are the same, there is no injection attack; otherwise, the attack is present. Balasundaram et al. [23] proposed a technique using ASCII-based string matching for detecting SQL injection attacks. This technique utilizes static and dynamic analysis to examine user input fields to identify and prevent SQL injection attacks.

6 Machine Learning Classification Based Modeling

Classification is a supervised learning technique extensively used for modeling cyber-attacks based on various attack categories. In supervised learning, data is always labeled before training. During the training phase, the classifier learns labels so that it can

accurately predict for data that has not been seen before during the testing phase. In our analysis, commonly used machine learning techniques for various purposes are implemented. Several techniques can be summarized as follows.

6.1 Naive Bayes

Naive Bayes is a type of Bayesian network and a common machine learning algorithm [26]. It is a basic probability-based technique that calculates the probability of classifying or predicting the class of a cyber attack in the given dataset. This method assumes that the value of each feature is independent and does not consider the correlation or relationship between features [27]. Naive Bayes consists of two probabilities: conditional probability and class probability. The class probability is determined by dividing the frequency of each class instance by the total number of instances. The conditional probability is the ratio of the repetition of each feature for a given class and the repetition of instances for that class. Naive Bayes is faster than other classifiers.

6.2 Decision Trees

Decision Tree is one of the most popular algorithms for classification and prediction in machine learning. ID3, proposed by J. R. Quinlan [29], is a common top-down approach for building decision trees. Based on this, the C4.5 algorithm [30], and later the BehavDT method [30], the IntruDTree model [32] have been developed for generating decision trees. A decision tree is a tree-like structure in which an internal node represents features, branches indicate outcomes, and leaves represent a class label. These algorithms create decision rules for predicting outcomes for unseen test cases. They provide high accuracy and better interpretability. Decision trees can handle both continuous and discrete data.

6.3 Random Forest

Random Forest is a classifier composed of decision trees as a team learning method [33, 34]. Breiman and his colleagues proposed this method.

6.4 SVM

Support Vector Machine operates by maximizing the distance between data points from the separator boundary, known as the margin. The SVM algorithm can classify with high accuracy and can be used for classification and regression tasks [35].

6.5 Artificial Neural Network

Additionally, in parallel with classical machine learning techniques above, we consider an artificial neural network learning model. The most common architecture of an artificial neural network is a multi-layer perceptron consisting of an input layer with multiple

inputs, one or more hidden layers typically using sigmoid activation functions, and an output layer for attack prediction. This approach utilizes backpropagation for network construction [36].

7 Experimental Evaluation

This section defines performance metrics in the field of intrusion detection and examines the results by conducting experiments on cybersecurity datasets with various attack categories. If TP refers to true positives, FP to false positives, TN to true negatives, and FN to false negatives, then the formal definition of the following metrics is as follows.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

8 Dataset

The most critical part of detecting SQL injection attacks is the data and dataset. This section plays a crucial role in detecting and predicting SQL injection attacks. The data used in this section must be very accurate and meaningful, containing SQL injection attack queries. In this article, the dataset available on Kaggle has been utilized. This dataset consists of two main fields, namely Query and Label. In the Query field, malicious queries and legitimate commands are included. Out of all queries, 11,331 of them are labeled as one, indicating SQL injection attacks, while the remaining 19,595 are labeled as zero, representing harmless queries.

9 Experimental Results and Discussion

In this section, the effectiveness of machine learning algorithms for detecting intrusions has been investigated. For this purpose, an analysis has been conducted on various classification techniques, including Artificial Neural Networks (ANN), Naive Bayes (NB), Support Vector Machine (SVM), Decision Trees (DT), and Random Forest (RF). Additionally, values for precision, recall, F1 score, and accuracy for each of the examined classification models have been evaluated.

To evaluate the performance of each of the classification models in intrusion detection systems, Figures 2 and 3 respectively compare accuracy, precision, recall, and F1 score.

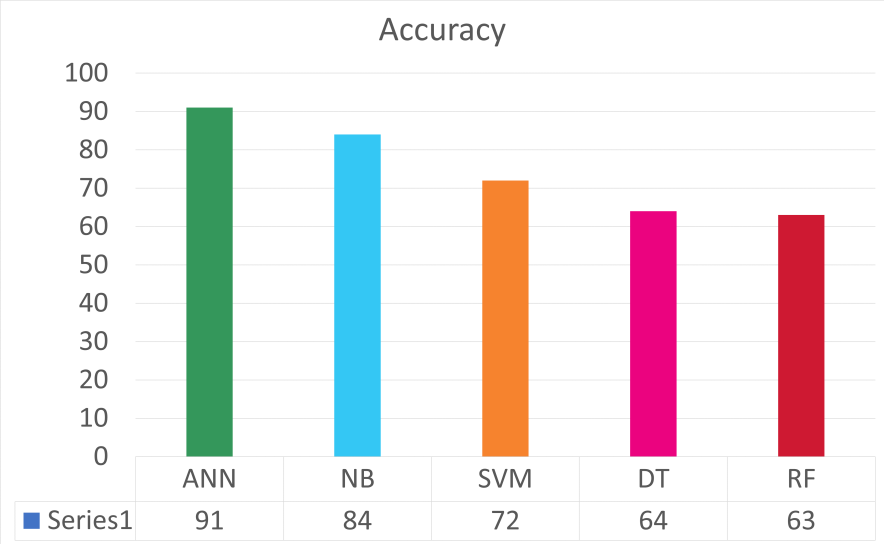

Figure 2: Performance Comparison Results in Terms of Accuracy for Machine Learning-Based Models in Intrusion Detection Systems

For evaluation, the same set of training and testing data is used for each machine learning-based classification model in intrusion detection systems.

Figures 2 and 3 show that the artificial neural network-based intrusion detection model consistently outperforms other classifiers in detecting intrusions. Specifically, artificial neural network achieves the best results in terms of accuracy, precision, recall, and F-score. The reason for this is that the neural network classifier initially creates several neural networks, and for each different network, a set of inference rules is derived. Each neural network in the artificial neural network model acts as a different machine learning classification technique, and considering the majority voting of these networks, more logical rules are generated. Therefore, the artificial neural network model performs better in terms of accuracy, recall, F-score, and precision. Overall, the machine learning-based intrusion detection model discussed above focuses entirely on the data and reflects behavioral patterns of various cyber attacks.

10 Comparison of Traditional Methods and Machine Learning In SQL Attack Prevention

Traditional methods such as static and dynamic analysis discussed in the Related Works section rely heavily on examining predefined and fixed patterns, but they have limitations when dealing with more complex or evolving attacks, as seen with AMNESIA and HMM. These methods often face efficiency issues when confronted with advanced attacks. In contrast, machine learning-based models discussed in the Machine Learning Classification Based Modeling section offer greater flexibility in detecting emerging

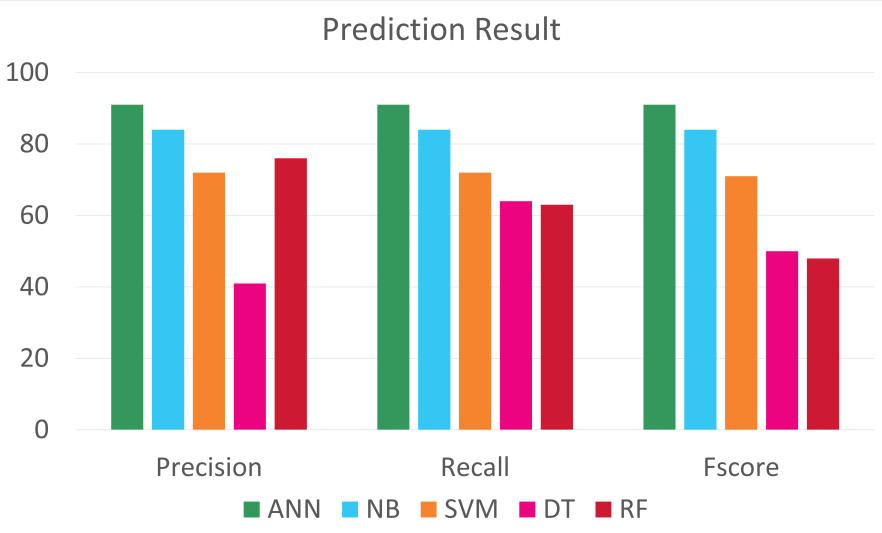

Figure 3: Performance Comparison Results in Terms of Accuracy, Recall, and F-score for Machine Learning-Based Classification Models in Intrusion Detection Systems

threats. Models like Artificial Neural Networks (ANN), due to their ability to learn from data and adapt to new patterns, outperform traditional methods in detecting complex attacks and demonstrate higher accuracy.

11 Conclusion

The capability and efficiency of a machine learning-based intrusion detection model is one of the most fundamental concerns for individuals active in the field of IT, e-commerce, and application developers from a security perspective. In general, cyber-security datasets comprise various types of cyber attacks along with their associated features. Therefore, some classification models may not perform optimally in terms of accuracy and true prediction rates based on diverse attack categories and various features. In this article, we have investigated the performance of data-based intrusion detection models considering well-known classification techniques in the field of machine learning. Additionally, performance metrics such as accuracy, recall, F-score, and overall precision have been evaluated. Our future plans include expanding cybersecurity datasets and designing a data-based intrusion detection system that benefits from a combination of various intrusion detection models. This composite system aims to provide automated security services to the cybersecurity community.

References

[1] A. F. Aseel and Y. Abri, "Vulnerability Analysis of Social Media Accounts Against Cyber Attacks", in Proc. 2nd Conference on Cyberspace, University of Tehran, Iran, 2023, pp. 29-36.

[2] S. S. A. Krishnan, A. N. Sabu, P. P. Sajan, and A. L. Sreedeep, 'SQL injection detection using machine learning', vol, vol. 11, p. 11, 2021.

[3] T. Muhammad and H. Ghafory, 'SQL Injection Attack Detection Using Machine Learning Algorithm', Mesopotamian journal of cybersecurity, vol. 2022, pp. 5–17, 2022.

[4] N. M. Sheykhkanloo, 'SQL-IDS: evaluation of SQLi attack detection and classification based on machine learning techniques', in Proceedings of the 8th International Conference on Security of Information and Networks, 2015, pp. 258–266.

[5] U. Farooq, 'Ensemble machine learning approaches for detection of sql injection attack', Tehnički glasnik, vol. 15, no. 1, pp. 112–120, 2021.

[6] W. G. Halfond, J. Viegas, A. Orso, and Others, 'A classification of SQL-injection attacks and countermeasures', in Proceedings of the IEEE international symposium on secure software engineering, 2006, vol. 1, pp. 13–15.

[7] B. A. Cumi-Guzman, A. D. Espinosa-Chim, M. G. Orozco-del-Castillo, and J. A. Recio-García, "Counterfactual explanation of a classification model for detecting SQL injection attacks", Tecnológico Nacional de México / IT de Mérida, Mérida, México, 2024.

[8] Z. C. S. S. Hlaing and M. Khaing, 'A detection and prevention technique on sql injection attacks', in 2020 IEEE Conference on Computer Applications (ICCA), 2020, pp. 1–6.

[9] W. G. J. Halfond and A. Orso, 'Detection and prevention of SQL injection attacks', in Malware Detection, Springer, 2007, pp. 85–109.

[10] M. Kumar, L. Indu, "Detection and Prevention of SQL Injection Attack", International Journal of Computer Science and Information Technologies, vol. 5, no. 1, pp. 374-377, 2014.

[11] G. Yigit, M. Amavutoglu, "SQL Injection Attacks Detection & Prevention Techniques", International Journal of Computer Theory and Engineering, vol. 9, no. 5, pp. 351-356, October 2017

[12] M. Ŝtamper, "Inferential SQL Injection Attacks", International Journal of Network Security, vol 18, no. 2, pp. 316-325, Mar 2016.

[13] A. Brehmer and M. Teräs, "SQL injection vulnerabilities in open-source projects", Final Project, Mid Sweden University, Östersund, Sweden, Spring 2024.

[14] N. Moradpoor, "A Pattern Recognition Neural Network Model for Detection and Classification of SQL Injection Attacks", The 13th International Conference on Information and Communication Engineering (ICICE15), Jun 2015.

[15] Halfond, W. G. & Orso, A. (2005). AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, 174-183. https://doi.org/10.1145/1101908.1101935

[16] Shar, L. K. & Tan, H. B. K. (2013). Defeating SQL injection. Computer, 46, 69-77. https://doi.org/10.1109/MC.2012.283

[17] Tajpour, A. & Shooshtar, M. J. Z. (2010). Evaluation of SQL injection detection and prevention techniques. In Second IEEE International Conference on Computational Intelligence, Communication Systems and Networks, Liverpool, UK, 216- 221. https://doi.org/10.1109/CICSyN.2010.55

[18] Dharam, R. & Shiva, S. G. (2013). Runtime monitors to detect.

[19] Kar, D., Agarwal, K., Sahoo, A., & Panigrahi, S. (2016). Detection of SQL injection attacks using Hidden Markov Model. 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India. https://doi.org/10.1109/ICETECH.2016.7569180

[20] N. Lambert, K.S. Lin; "Use of Query tokenization to detect and prevent SQLinjection attacks", Proceedings of the 3rd International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China:IEEE (2010). pp: 438-440, 2010.

[21] D. Lu, J. Fei, and L. Liu, "A semantic learning-based SQL injection attack detection technology", MDPI, 2023.

[22] A. Kumar, S. Bhatt, "Use of Query Tokenization to Detect and Prevent SQL Injection Attacks", International Journal of Science Technology & Engineering, vol. 2, issue. 01, pp. 97-103, July 2015

[23] I. Balasundaram, E. Ramaraj, "An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching", International Conference on Communication Technology and System Design, Prodedia Engineering, pp. 183-190, 2012.

[24] I.H. Witten, E. Frank, L.E. Trigg, M.A. Hall, G. Holmes, and S.J. Cunningham, "Weka: practical machine learning tools and techniques with Java implementations", 1999.

[25] J. Han, J. Pei, and M. Kamber, "Data Mining: Concepts and Techniques", Elsevier, Amsterdam, 2011.

[26] G.H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers", in Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[27] J.R. Quinlan, "Induction of decision trees", Mach. Learn., vol. 1, no. 1, pp. 81–106, 1986.

[28] I.H. Sarker, A. Kayes, and P. Watters, "Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage", Journal of Big Data, 2019.

[29] J.R. Quinlan, "Induction of decision trees", Mach. Learn., vol. 1, no. 1, pp. 81–106, 1986.

[30] J.R. Quinlan, "C4.5: programs for machine learning", Machine Learning, 1993.

[31] I.H. Sarker, A. Colman, J. Han, A.I. Khan, Y.B. Abushark, and K. Salah, "Behavdt: a behavioral decision tree learning to build user-centric context-aware predictive model", Mobile Netw. Appl., vol. 1, pp. 1–11, 2019.

[32] I.H. Sarker, Y.B. Abushark, F. Alsolami, and A.I. Khan, "Intrudtree: a machine learning-based cyber security intrusion detection model", Symmetry, vol. 12, p. 754, 2020.

[33] L. Breiman, "Random forests", Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[34] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model", MDIP, 2024.

[35] S. Venkatramulu, M. S. Waseem, A. Taneem, S. Y. Thoutam, S. Apuri, and Nachiketh, "Research on SQL injection attacks using word embedding techniques and machine learning", Journal of Sensors, IoT & Health Sciences, vol. 2, no. 1, pp. 55-66, Mar. 2023.

[36] F. K. Alarfaj and N. A. Khan, "Enhancing the performance of SQL injection attack detection through probabilistic neural networks", MDIP, 2023.