Literature Survey on Web based Recognition of SQL Injection Attacks

Anuradha S Ramchandran[1], Nikita Bhyri[2], Sree Raksha H R[3] , Uthara R Nambiar[4]

[1-4]Student, Dept. of Computer Science Engineering, Dayananda Sagar College of Engineering, Karnataka, India

---

Abstract - *SQL (Structured Query Language) is a query language that allows users to access and manipulate information stored in databases. SQL injection attacks (SQLIA) have become one of the most often employed attacks by hackers, and the attacking strategies evolve with the advancement of website technology on a regular basis.*

Injection attacks were among the top 10 security vulnerabilities, according to surveys done by the OWASP (Open Web Application Security Project) organization in 2013 and 2017. The goal of this survey paper is to provide readers with a brief review of the various research and methodologies that have been developed and proposed to address SQLIA.

***Key Words*: SQL Injection Detection; Web Security; Machine Learning; Cyber Security**

1. INTRODUCTION

Network and web apps have increasingly become an integral aspect of our lives. But as technology advances, apps face greater obstacles. SQL (Structured Query Language) is a query language that allows users to access and manipulate information stored in databases. Various ways for creating and executing SQL statements are available in languages like PHP and Java, which are used for web development. SQL statements are typically created by concatenating strings submitted by visitors via a website. Because of how diverse SQL is, numerous encoding methods open the possibility of being infiltrated by formulating SQL statements. When performing such an attack, malice-intentioned code segments are inserted into the request parameters, which gives rise to the server running unauthorized queries, resulting in information leakage and database malfunction. For example, the hacker can gain access to website users' account names, login passcodes, and other private details, posing a substantial threat to data security. Methods involve injecting or changing user input, cookies, and server-side scripting variables.

Attacking websites by injecting bad SQL queries is now one of the most frequently used techniques by hackers, and strategies change with the evolution of web development. According to surveys done by the OWASP organisation in 2013 and 2017, such attacks were amongst the top 10 security vulnerabilities, i.e., among the most dangerous. As discussed above, a SQL injection attack simply requires the construction of structured query statements without the production of extra software. The structure of SQL language can be changed, and it enables a variety of coding styles. As a result, several classic approaches are unable to detect a large variety of these coding styles. and thus, cannot provide a more effective defensive effect. Many academics have worked on SQL injection detection in recent years, although the detection scope of each project is usually confined to certain subgroups of SQL injection attacks.

A robust SQL injection detection architecture that has the ability to recognize all types of SQL injections and is flexible enough to adapt when an advanced one arises is essential.

The primary goal of this study is to evaluate recent research on various algorithms and methods used to identify SQL Injections in websites.

2. LITERATURE SURVEY

Xin Xie et al. [1] presented a solution to recognize and catch SQLIA in web applications and large weblogs using Elastic-Pooling Convolutional Neural Networks (EP-CNN). This method primarily addresses the issue of detecting SQLIA that are undetectable by traditional methods. The use of CNN in areas such as NLP (Natural Language Processing), image recognition, and computer vision is thriving, but this may have the disadvantage of misplacement, which is luckily overcome in the proposed method because fields of SQLI can exist anywhere in the query. This framework is shown to detect SQLIA quickly and accurately. It is also capable of generating a two-dimensional matrix without distorting or losing any details, performing well in terms of generalization. It can detect new attacks and is more difficult to defeat because of its ability to match irregular characteristics. The prospect for future work is detecting other cyberattacks.

Jothi K R et al. [2] put forward a system to detect SQL Injections using Deep Learning and Neural Networks, including embedding layers and relying on unprocessed information. This gives the system many advantages, such as scalability and the capability to detect a wide variety of such attacks. The model also makes the user's job easier by conducting all of the feature extraction and selection for them. Improvement in the future can be done on various fronts - to boost easy use and efficacy, the AI (Artificial Intelligence) approach may be combined with other SQL injection identification components and therefore the model also can be improved with better component extraction. When compared to other approaches, feature extraction done in the model is less effective, and training effectiveness can be enhanced. The results obtained are not as accurate as those produced when using a CNN-based architecture.

To differentiate between benign and malicious SQL queries, Benjamin Appiah et al. [3] used the Rabin Fingerprinting Method and the Aho Corasick Pattern Matching Algorithm. This system analyses database SQL queries against a database of signatures obtained from existing attacks. In the event that fingerprinting is unable to determine the authenticity of a query, pattern matching will be used to determine whether the queries contain attack signatures. The advantages of this system are that it is capable of identifying a large number of SQL injection attacks with almost zero impact on efficiency and detecting attacks on several web applications that are hosted on the same web server. However, the pattern matching algorithm used is computationally intensive, which can be problematic, and the proposed framework has a trade-off between the time taken to process queries and the amount of memory space needed. When the size is reduced, less time is taken, but efficiency is reduced. It also cannot detect unknown attacks.

To identify and lessen SQLIA, W. H. Rankothge et al. [4] designed a system that makes use of a single tool: WebVIM, which is based on parameterized queries and user input validation. First, the tool analyses the source code to look for SQLIA vulnerabilities on every page of the web application. If a page is SQLIA susceptible, then the security solution is inserted into the source code automatically. Lastly, the web application is tested against SQLIA again to verify that it is secure against SQLIA. This system works better with higher efficiency when compared to already existing tools, i.e., Nessus and OpenVAS and has a 100% success rate with mitigation. The limitation of this system is that it is effective only on PHP based web applications.

Warradorn Sirisang et al. [5] proposed a system to analyses SQLI statements using grammatical analysis. The method has two components: Automated Common Substructure Extraction, abbreviated as ACSE and Parse Tree Substructure Matching, abbreviated as PTSM. The first component captures repeated subparts that exist in SQL injection statement parse trees. The retrieved parse tree from ACSE is used by PTSM to identify harmful sections in input queries and measure similarity between the input queries and the shared substructure. This method has high accuracy - more than 90%. Unlike other similar approaches, this one takes less memory space and does not require any additional libraries or any modifications in the obtained source code. The only limitation is attacks against unknown weaknesses are undetectable.

D. Das et al. [6] performed an experimental study on how to detect multiple SQLIA on dynamic SQL queries using the Edit-Distance Method. The proposed technique differs from the existing ones in the way it verifies using profile-based learning. Subroutine calls are used throughout the proposed run-time verification mechanism for determining if a dynamic query is innocuous or harmful. An algorithm is used to calculate the binary-distance value of the given input test query. This system overcomes application functionality limitations caused by dynamic query patterns, can adaptively adjust the threshold and is able to find SQLIA in multiple website authentication scenarios, however, entails a rigorous training process and has a restriction on the number of variables that can be used in a dynamic SQL query.

Abbas Naderi-Afooshteh et al. [7] proposed a system called 'Joza', which utilizes the Hybrid Taint Inference method to defeat SQLIA in web applications. Using only positive or negative taint inference (TI) poses security risks, which are overcome by using a hybrid approach. The proposed system has two main aspects: positive and negative TI evaluation. Before being allowed to go to the DBMS (Database Management System), instructions meant for it are diverted to each unit. This system safeguards against a large variety of SQL injection attacks with high effectiveness, has no false positives, has minimal overhead, and is simple to implement. The limitation of this system is that it works only with PHP-based applications.

Yong Fang et al. [8] put forward an approach to detect SQL Injection behavior using token word vectors from queries and a long short-term neural network, incorporated by a software called 'WOVSQLI'. The advantages of this tool are that it is featureless, has high values of accuracy, recall and precision. Performs better than a similar algorithm that utilizes random forest and can effectively detect SQLIA. In future research, new data can be used to improve the performance, which is impacted by the size and variety of the dataset used.

B.Hanmanthu et al. [9] recommended a model that uses a 3-tier architecture system along with decision tree classification to detect and protect against SQLIA. This architecture consists of query pre-processing, result analysis and response. Pre-processing of the obtained SQL queries is done to ensure more accuracy when they are used as input to the decision tree algorithm. Classified results from this are sent for analysis and response. When contrasted with a few similar models, the model outperforms them with time and accuracy metrics, but cannot detect all types of attacks, or unknown vulnerabilities.

Chen Ping [10] proposed an approach to detect second-order SQLIA using Instruction Set Randomization. In this proposed approach, new SQL instruction sets are dynamically generated by randomizing known arguments in websites, and a proxy server that assesses each query for standard keywords to detect harmful activity is added. The proxy server also derandomizes queries that do not show any behavior of SQLIA. Second-order SQL injection attacks are notoriously harder to detect than first-order injections and this approach detects these with high accuracy, high efficiency, and low processing costs. It is easy to deploy and does not depend on one database or server platform.

Mohammad Qbea'h et al. [11] proposed a dynamic approach to recognize and avert SQL Injection Attacks that uses finite automata and regular expressions (RE) to formalise alternate encoding attacks and tautology. To check user input and recognize and prevent SQLIA, expressions are introduced into the code. The proposed RE and programs are provided in ASP.net for easy use by developers, without them restructuring or rewriting their own. Other databases, such as Oracle, and programming languages other than ASP.net, such as JAVA and PHP, can be used with the proposed theory. One huge advantage of this approach is that it covers characters from a variety of languages, including English, Arabic, Japanese, and others, which is not done by other approaches. The limitation is that not all SQLIA can be characterized and formalized, and the approach ignores case sensitivity for keywords and operators, which should not be done practically.

Yaohui Wang et al. [12] put forward the method of dynamic and static analysis for detecting SQL vulnerability attacks. This method is based on injection analysis technologies and this study provides a SQL vulnerability detection approach. This method investigates the one-time injection in terms of how data flows and program behavior using a combination of dynamic and static analysis tools. SQL vulnerability dedication set of criteria, which is primarily based on lexical function evaluation, is then applied. Finally, this work integrates alias evaluation technology, conduct version, and SQL that is largely dependent on lexical function evaluation to develop and create a prototype device for SQL vulnerability detection. When comparing detection technology, it is clear that static detection plays the most important function in detection technology. Among other advantages, this approach offers code coverage at a high rate, fast speeds, and fins vulnerabilities at a high rate. Excessive false positive and negative rates, low precision, and high code reliance are some of the disadvantages.

Li Qian et al. [13] did research on attack and aversion technology of SQL injection. A defensive model has been designed to prevent them, checks not just input form data, but also recognizes information from the browser URL bar, which is very valuable for sensitive character identification. To start, the server checks the IP address's authenticity. The user is denied access to the login server if input values are illegal. Secondly, input data is validated on the server by looking at the format, range, length, and type. If the input string matches the SQL prevention rule, then the user is allowed to access the page. Finally, the server verifies the permissions of the user. The system disables the user and sends a message to the system administrator if the number of users exceeds the access permissions. When all verifications are invalid, the server logs the injection attack. This paper describes the characteristics of a SQL injection attack and, as a result, the strategies for preventing it. SQL injection threats may be avoided by using input validation and type-safe SQL parameter processes. On this structure, a defense model is built, which identifies assaults by contrasting original input length to the obtained ones. The method may be used to secure online applications. Such assaults use software flaws to enter destination databases through the user's client. Furthermore, vulnerabilities in web component input validation can be exploited for a chance at attacking.

Taiki Oosawa et al. [14] provided a technique for calculating the zeta distribution parameter and provided a method for detecting SQL injection attacks. They also put their proposed methodology to the test, demonstrating that the attack data detection method works well by comparing it to certain common approaches. By the formula, the zeta distribution closely approximates the distribution of the symbols. According to the results of the detection experiment, the suggested approach identified attack data rather effectively. In the case of normal data, the SVM approach was determined to be superior to our suggested method. It is required to work on including a high number of symbols as well as taking into account the Gaussian kernel parameter's property.

Meharaj Begum A et al. [15] came up with a pattern-based neural network model for SQLIA. The purpose of this research was to use existing parsing and tagging techniques to extract SQL injection patterns using the Pattern-based CNN model. Multi-layer Perceptron is used to train and model pattern-based tags, which performs substantially better than previous algorithms in query categorization, with an accuracy of 94.4 percent. This model deals with inquiry patterns and conquers the shortcomings of similar current systems that compare query strings to basic strings or consider aspects like keyword count and punctuation count. It successfully classifies and detects injections. The signature (pattern) of the expressions along with predicates of a SQL query is an important aspect of this research. The main problem with this study is the dataset size, and it only uses SQLIA to find tautologies. This research is not being expanded to include additional types of injections.

Nagasundari et al. [16] proposed SQLIA detection using ResNet (Residual Networks). The SQL injection detection system based on ResNet learns and identifies SQL injection attacks on its own. This approach is used to train data that has been processed. The UI (User Interface) is designed for a test case in which a user is expected to input either a malicious or a legitimate query. The trained model is capable of distinguishing between fake and authentic input requests with ease. ResNet has been shown in tests to be capable of identifying a wide range of SQLIA. Traditional detection approaches are slow to identify different types of SQLIA, resulting in a greater rate of false positives and false negatives. The proposed ResNet model is designed and developed in such a way that it automatically detects all forms of SQLIAs. The dataset is subjected to a large number of test cases, which enhances accuracy. A downside is that man-in-the-middle attacks are a vulnerability while training and testing the model.

Musaab Hasan et al. [17] proposed a machine learning approach to find SQLIA, a machine learning-based heuristic technique that uses a dataset of 616 SQL statements to train and test 23 different ML (Machine Learning) classifiers. Based on how accurately identification is done, the best five classifiers are picked, and a GUI (Graphic User Interface) program is created employing these five classifiers. The proposed approach is then put to the test, with the results indicating that it is capable of accurately identifying SQL injection assaults (93.8 per cent). The proposed approach has been carefully tested, and the results show that Ensemble Boosted and Bagged Trees classifiers are the most precise. Both have a higher than 0.9 area under the receiver operating characteristic curves, suggesting that the system is working well. The problem is that the dataset necessitates the examination of more non-injected SQL statements, as well as the experimentation and exploration of additional features.

Based on behavior and response analysis, Zeli Xiao et al. [18] proposed a method for detecting SQL injection. Their strategy is focused on the study of attack behavior as well as the reaction and status of the web application under various assaults. This solution corrects various flaws in the present SQL detection system (rule matching is missing, user input variety leads to false positives, and calculation is time-consuming). When a web application is discovered to be under assault, it may quickly stop attackers from continuing their attack by putting the user on a blacklist and denying them access to online apps. The problem here is that the system SQL execution has an uncertainty component, the invariant extraction is not complete, and the invariant error, as well as the pattern matching attack, result in false alarms and omissions.

Pan Lina et al. [19] proposed the method of GreenSQL pattern input whitelist for SQLIA Detection. The GreenSQL pattern input whitelist approach optimizes the input model by creating a patterned input and optimized whitelist based on an assessment of SQL injection attack instructions' characteristics and patterns. The efficacy of input is enhanced when compared to the typical random input command, GreenSQL's learning efficiency can be enhanced, and its IPS (Intrusion Prevention System) intercepting samples may be increased, which elevates its use. This is the quickest mode since we only compute risk for new requests, which don't happen very often. As a result, it is ineffective for often occurring questions.

Solomon Ogbomon Uwagbole et al. [20] recommended detecting and averting SQLIA by using ML to predict and perform analytics. It uses a Two-Class Support Vector Machine. It also provides a framework for detecting and preventing SQLIA on the big data internet. In big data, the method described here is useful as it can not only detect but also prevent attacks and has 98% accuracy. However, it is not a multiclass classifier, hence, cannot group different types of SQL injection attacks.

Angshuman Jana et al. [21] proposed an Input-based Analysis Approach. The purpose is to determine harmful user inputs, which are often made up of special symbols, keywords, or a combination of both. This model can automatically determine and avert threats. The proposed methodology can be used to create an input checker that can automatically detect and prevent SQLIA. There is also an input verifier to detect fraudulent inputs. Database application programs can also be safeguarded.

Pan Lina et al. [22] put forward the GreenSQL Pattern Input Whitelist. A matrix is used to determine the level of risk associated with SQL operations that are launched from the outside. If the analysis score of a SQL command exceeds a certain threshold, the command will be blocked if not already done so; else, it will be pushed to the background left for SQL command processes. In comparison to the traditional input command, the input usefulness is enhanced. However, GreenSQL's learning efficiency has the potential to be improved, which will help to promote its use.

Ao Luo et al. [23] proposed A CNN-based Approach. In this paper, traffic obtained by the client is used as the focus of the study. The payload of SQL injection attacks is carved out of data collecting and sanitization of traffic. After building the CNN network model, this can be used as input data and indicate if threats are present. Practical demonstrations reveal that this strategy has some advantages over the rule-matching-based method. The effectiveness and usability of the CNN-based SQL injection recognition system have also been established. The disadvantages are that the CNN model needs to include more kinds of intrusion scanning capabilities and the dataset can be expanded.

Neel Gandhi et al. [24] put forward a CNN-BiLSTM based approach. For identifying SQL injection attacks, the study proposes a machine learning algorithm founded on a hybrid CNN-Bi-LSTM, trained with a sizable input query set, followed by pre-processing such as tokenization and feature extraction, ML model training, and validation on multiple evaluation matrices. It outperforms other machine learning approaches in terms of accuracy and speed. Better approaches could be employed to tokenize, extract characteristics, and stem the input.

Krit Kamtuo et al. [25] put forward a method to prevent SQLIA in website development using Machine Learning. As it is the major core for SQL injection prediction, this study presents a solution centered on the machine learning and compiler platform. 4 types of machine learning-based models were employed to train 1,100 instances of susceptible SQL statements, out of which Decision Tree proves to be the best model in terms of prediction efficiency and processing time, as per the findings. Future research will focus on the creation of a compiler platform based on an IDE that can analyses query syntax and identify SQLIA in code before the site has been launched.

Inyong Lee et al. [26] proposed a solution based on eliminating SQL query attribute values. This study presents an effective yet simple SQL injection attack detection mechanism. When parameters are provided, the method contrasts the value of a SQL query attribute of web pages against a pre-set one and eliminates it. Static and dynamic analyses are integrated into this strategy. This method compares the SQL queries examined in advance to the SQL queries that have their attribute values removed at runtime. For analysis, the suggested approach merely eliminates attribute values from SQL queries, making it independent of the DBMS. The proposed solution does not require complex operations like parse trees or specific libraries. It can be utilized not only with online apps but also with other database-connected applications, to profile and list queries, and modularize detection software. More research is needed on this issue and also for additional website attacks like XSS (Cross-site Scripting), using the recommended method and classifier techniques.

Rajashree A. Katole et al. [27] proposed an approach to detect SQLIA that involved eliminating parameter values of the given input queries. Here, two methods of query processing and parameter elimination are coupled to create a solution that uses parameters to distinguish between unaltered sources and harmful altered queries. The advantages of this approach are that it does not require any source code modification at all and both static and dynamic analyses are performed on SQL queries. However, this technique is time-consuming and further research is required for better accuracy.

Lwin Khin Shar et al. [28] put forward a method to recognize SQLIA and cross-site scripting attacks by analyzing input sanitization patterns. This research proposes static code features that might be used to predict certain program statements as an alternative to existing taint analyzers. To forecast SQLI and XSS vulnerabilities, we utilize historical data to construct vulnerability prediction models that reflect suggested static features and known vulnerability data. According to the results of the experiment, the proposed vulnerability predictors are effective and successful at what they were intended to do. However, it has been found that predictors produced this way do not give the same results across investigations and therefore the authors aim to undertake further trials on a variety of systems and re-evaluate existing findings.

William G.J. Halfond et al. [29] presented a model titled "AMNESIA," which analyses and monitors SQLIAs. The method is based on earlier work in a security analysis of models and programs and combines static and dynamic analysis tools designed for SQLIAs. The main discoveries are that the knowledge required to predict the structure of a website's queries is contained inside its code and that a SQLIA would breach that structure by introducing more snippets. The study's findings reveal that the technique was successful in stopping attacks without producing erroneous results. Where static analysis is incapable of being used, alternative methodologies for building SQL models will be examined in future studies.

G.Buja et al. [30] presented a model that identifies SQLIA in web applications. The suggested detection methodology will generate a summary of how susceptible the website would be and as a result, reduce the chances of such an attack. There is only one core detecting module. The method will be carried out using the Boyer Moore technique to match strings which improves effectiveness and precision. This approach will help a website developer or admin take further efforts to protect their program from external hacks or exposure to the web application's SQL Injection susceptibilities. Performance can be improved by detecting and adding a couple of extra arguments to the "Parameter Testing Panel."

3. CONCLUSIONS

A lot of careful investigation is being conducted to identify SQL Injection Attacks (SQLIA). We evaluate and review over 30 approaches that employ various algorithms to detect and minimize SQLIA in web applications in this research. Some of the methods, models, and classifiers that were used are Deep Learning, CNN, LSTM, Bi-LSTM, Pattern-based Neural Networks, Decision Trees, Hybrid Taint Inference, etc. The number of data sets utilized varied between 1,000 and 100,000.

When evaluated experimentally, most studies demonstrate great performance, indicating that they are capable of detecting most SQL Injections. Confusion matrix, accuracy, efficiency, precision, and recall are common measures used to assess this degree of performance. The results also suggest that there is room for more study and research in the future to improve and expand the approaches utilized.

REFERENCES

[1] X. Xie, C. Ren, Y. Fu, J. Xu, and J. Guo, "SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN," in IEEE Access, vol. 7, pp. 151475-151481, 2019.

[2] J. K. R, S. Balaji B, N. Pandey, P. Beriwal and A. Amarajan, "An Efficient SQL Injection Detection System Using Deep Learning," 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), 2021, pp. 442-445.

[3] B. Appiah, E. Opoku-Mensah, and Z. Qin, "SQL injection attack detection using fingerprints and pattern matching technique," 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 583-587.

[4] W. H. Rankothge, M. Randeniya and V. Samaranayaka, "Identification and Mitigation Tool for SQL Injection Attacks (SQLIA)," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), 2020, pp. 591-595.

[5] W. Sirisang and V. Suttichaya, "Analyzing SQL Injection Statements Using Common Substructure of Parse Tree," 2017 21st International Computer Science and Engineering Conference (ICSEC), 2017, pp. 1-5.

[6] Das, D., Sharma, U. & Bhattacharyya, D.K., “Defeating SQL injection attack in authentication security: an experimental study”, Int. J. Inf. Secur. 18, 1–22 (2019).

[7] A. Naderi-Afooshteh, A. Nguyen-Tuong, M. Bagheri-Marzijarani, J. D. Hiser and J. W. Davidson, "Joza: Hybrid Taint Inference for Defeating Web Application SQL Injection Attacks," 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, pp. 172-183.

[8] Y. Fang, J. Peng, L. Liu, and C. Huang, ‘‘WOVSQLI: Detection of SQL injection behaviours using word vector and LSTM,’’ in Proc. ICCSP, 2018, pp. 170–174.

[9] B. Hanmanthu, B. R. Ram and P. Niranjan, "SQL Injection Attack prevention based on decision tree classification," 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 2015, pp. 1-5.

[10] C. Ping, "A second-order SQL injection detection method," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2017, pp. 1792-1796.

[11] M. Qbea'h, M. Alshraideh and K. E. Sabri, "Detecting and Preventing SQL Injection Attacks: A Formal Approach," 2016 Cybersecurity and Cyberforensics Conference (CCC), 2016, pp. 123-129.

[12] Y. Wang, D. Wang, W. Zhao, and Y. Liu, "Detecting SQL Vulnerability Attack Based on the Dynamic and Static Analysis Technology," 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015, pp. 604-60.

[13] Li Qian, Zhenyuan Zhu, Jun Hu and Shuying Liu, "Research of SQL injection attack and prevention technology," 2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF), 2015, pp. 303-306.

[14] T. Oosawa and T. Matsuda, "SQL injection attack detection method using the approximation function of zeta distribution," 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2014, pp. 819-824.

[15] M. B. A and M. Arock, "Efficient Detection Of SQL Injection Attack (SQLIA) Using Pattern-based Neural Network Model," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021, pp. 343-347.

[16] Sangeeta, S. Nagasundari and P. B. Honnavali, "SQL Injection Attack Detection using ResNet," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1-7.

[17] M. Hasan, Z. Balbahaith and M. Tarique, "Detection of SQL Injection Attacks: A Machine Learning Approach," 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), 2019, pp. 1-6.

[18] Z. Xiao, Z. Zhou, W. Yang and C. Deng, "An approach for SQL injection detection based on behaviour and response analysis," 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1437-1442.

[19] P. Lin, W. Jinshuang, C. Ping and Y. Lanjuan, "SQL Injection Attack and Detection Based on GreenSQL Pattern Input Whitelist," 2020 IEEE 3rd International

Conference on Information Systems and Computer Aided Education (ICISCAE), 2020, pp. 187-190.

[20] S. O. Uwagbole, W. J. Buchanan and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 1087-1090.

[21] A. Jana, P. Bordoloi, and D. Maity, "Input-based Analysis Approach to Prevent SQL Injection Attacks," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1290-1293.

[22] P. Lin, W. Jinshuang, C. Ping and Y. Lanjuan, "SQL Injection Attack and Detection Based on GreenSQL Pattern Input Whitelist," 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), 2020, pp. 187-190.

[23] A. Luo, W. Huang, and W. Fan, "A CNN-based Approach to the Detection of SQL Injection Attacks," 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), 2019, pp. 320- 324.

[24] N. Gandhi, J. Patel, R. Sisodiya, N. Doshi and S. Mishra, "A CNNBiLSTM based Approach for Detection of SQL Injection Attacks," 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), 2021, pp. 378-383.

[25] K. Kamtuo and C. Soomlek, "Machine Learning for SQL injection prevention on server-side scripting," 2016 International Computer Science and Engineering Conference (ICSEC), 2016, pp. 1-6.

[26] Inyong Lee, Soonki Jeong, Sangsoo Yeo, Jongsub Moon, “A novel method for SQL injection attack detection based on removing SQL query attribute values”, Mathematical and Computer Modeling, Volume 55, Issues 1–2, 2012, pp. 58-68.

[27] R. A. Katole, S. S. Sherekar and V. M. Thakare, "Detection of SQL injection attacks by removing the parameter values of SQL query," 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 736-741.

[28] Lwin Khin Shar, Hee Beng Kuan Tan, ”Predicting SQL injection and cross-site scripting vulnerabilities through mining input sanitization patterns”, Information and Software Technology, Volume 55, Issue 10, 2013, pp. 1767-1780.

[29] William G. J. Halfond and Alessandro Orso. 200, “AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks”, Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering (ASE '05), Association for Computing Machinery, New York, NY, USA, 174–183.

[30] G. Buja, K. B. A. Jalil, F. B. H. M. Ali, and T. F. A. Rahman, "Detection model for SQL injection attack: An approach for preventing a web application from the SQL injection attack," 2014 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), 2014, pp. 60-64.