International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Automating SQL Injection Prevention: A Machine Learning Perspective

V.Sowmiya[a],V.Ganapathi Gayathri[b] ,R.Kaviyarasan[c] , R.Sailesh[d] R.V.Sanjaydharsan[e]

[a]Student, ComputerScience & Engineering, Paavai College of Engineering, Namakkal
.[b]Student, Bio Technology, P.S.R Engineering College,Sivakasi
[c]Student, Artificial Intelligence & Data Science, Paavai College of Engineering, Namakkal
[ed]Student, ComputerScience & Engineering, Paavai College of Technology, Namakkal.

ABSTRACT :

When online programs don't use secure codes, it will cause vulnerability leading to a cyber attack. According to the statistics, the SQL injection technique is most frequently used in cyber attacks connected to data theft. For any web system to counter this threat, an efficient SQL injection detection is therefore required. This study employs a machine learning technique wherein the SQL injection detector is trained.

Is assessed against testing data after being trained on training data. The training and testing datasets' preparation is essential to the research. Training sets are utilized by the detector to create the knowledge base, while test sets are used to assess the detector's performance. The detection's outcome demonstrates that the suggested method has a high degree of accuracy in differentiating between legitimate and fraudulent web requests.

Keywords: Machine Learning, Signature-Based, Knowledge-Based, SQL Injection, SQL Injection Tools

Introduction

Since data is currently an organization's most valuable asset, cyber threats and attacks against the database of the company are growing. The perpetrators and danger to the privacy of data are hackers, who may, for instance, use SQL Injection Attacks (SQLIA) to target weak websites. Additionally, a plethora of tools are currently available that may be used to automatically carry out hacking operations and assess a website's vulnerabilities. An attacker has a greater probability of accessing the web system database with the help of these technologies. A web developer is prone to write scripts with vulnerabilities if they lack the necessary security understanding. Putting safe measures in place to protect websites from these kinds of attacks is challenging. They created systems that are susceptible to SQL injection attacks as a result. SQL injection is a kind of attack where malicious SQL queries are injected into the database to modify the website and reveal sensitive data. Therefore, if the SQL injection is identified early on, it can assist the security analyst or officer in stopping the assault. SQL injections can be found using intrusion detection systems (IDS) and intrusion prevention systems (IPS) (Patil et al., 2017). In this study, a SQL injection attack is identified by machine learning by comparing the website's access log file with a database of known dangerous properties.

Malware attacks are frequent in poorly constructed web applications, especially SQLI attacks. Despite being known for over 20 years, this vulnerability continues to raise concerns [1]. Relational database management systems (DBMS) have been handled by the industry standard for many years: structured query language (SQL). The development of cyber-physical systems might be severely limited by misbehavior resulting from random mistakes or online attacks, as most applications for these systems are safety-critical [2, 3]. Protecting cyber-physical systems from this form of attack is therefore essential.

Since it became popular for internet web applications and SQL databases to be connected, SQLI assaults, also known as SQLI attacks on data-driven online applications and systems, have been a critical issue [4, 5, 6]. An SQL injection (SQLI) attack happens when an attacker uses a fillable field in a web application to submit a malicious SQL statement, taking advantage of a weakness in the SQL implementation. To put it another way, the attacker will enter code into a field in order to access the backend, dump, or change data.

The proliferation and development of web applications have led to a rise in both the volume and intensity of web attacks [10, 11]. Injection is the most prevalent vulnerability in web applications, according to [12]. A web application processes untrusted user input that is delivered by injection attacks, which take use of a number of vulnerabilities [13]. In order to access a database or modify its contents (e.g., send the database contents to the attacker, modify or delete the database content, etc.), malicious SQL commands are injected, or inserted, into input forms or queries in SQL injection attacks [14, 15]. It is indisputable that the majority of web apps in use today rely on a back-end database to store user data and/or retrieve user-selected data [16].

Cookies and forms are frequently used to communicate with these users. By inserting harmful code into the user inputs that are subsequently utilized to create the SQL queries, many hackers try to take advantage of this feature. The success of the SQLI attack can be attributed to inadequate validation of user inputs, which can lead to devastating outcomes like the deletion of the database or the acquisition of private and sensitive information from web application customers [17]. Because of the sensitive nature of the SQLI attack, numerous studies have been conducted on it. While some of these efforts focus solely on detecting SQLI that has already happened, others try to stop it before it starts. We examined SQL Intrusion (SQLI) attacks in this study, which aim to get past the web application firewall and obtain unauthorized access to sensitive information. The HTTP or HTTPS protocol is the target of these attacks. Usually, the victim system is unprepared to handle this input, which commonly results in unauthorised access to the system by the attacker or data loss. This situation affects all aspects of security, including data availability, confidentiality, and integrity, since the attacker has access to and/or control over the data [2].

The remaining portions of the work are divided into distinct but connected subsections. Beginning with the "Related works" section, the paper moves on to discuss related works; the "SQLI query attacks overview" section discusses SQLI query attacks; the "Existing methods for SQLI detection and prevention" section discusses existing methods; the "Developing a web-based framework for SQLI attacks detection and prevention" section discusses developing a web-based framework for SQLI attacks detection and prevention; the "Most common attacks on SQLI" section discusses the most common attacks on SQLI; the "Proposed frameworks for SQLI detection and prevention" section proposes frameworks for SQLI detection and prevention; the "Result and discussion" section discusses results and discussion; and the "Conclusion and recommendation" section concludes the paper.

Related Works

In [1], a review of SQLI prevention in web applications is provided. An overview of 14 distinct SQLI attack types and their effects on web applications is given by the writers. The primary goals of their research were to examine different SQLI prevention tactics and provide an analysis of the best defense against SQLI assaults. A thorough literature assessment of 36 publications pertaining to studies on SQLI attacks and machine learning techniques was carried out by the authors in [2]. The most popular machine learning algorithms have been identified in order to categorize various SQLI attack variants. Their research showed that not many studies used machine learning tools and approaches to create new SQLI attack datasets.

A thorough analysis of SQLI attacks, including their mechanism, detection, and mitigation, is provided in [4]. The authors have noted a method to lessen the negative impact on database systems as well as how attackers of this type may take advantage of a vulnerability and run shoddy code. Web operations were widely employed for online administrations ranging from high degrees of informal contact to monitoring transaction accounts and handling sensitive user data, according to the researchers' investigation. The true problem, though, was that this data was vulnerable to attacks due to unauthorized access, whereby the attackers used a variety of hacking and cracking techniques to get into the system with extremely bad intentions.

A survey on SQLI attack detection and prevention has been provided in the work of [5]. The authors speculate that the research could aid laypeople in understanding SQL and its risks. It is also beneficial to academics and programmers who wish to understand more about the various issues still plaguing web applications and the tactics that may be used to prevent SQL Injection attacks. From the standpoint of the researcher, it was expected that online applications would be protected from such harmful attacks if web application developers followed the guidelines given in their study.

A presentation on end-to-end DL for web attack detection was made in [10]. This work has provided three new insights into the study of autonomous intrusion detection systems. First, they evaluated the viability of a technique for identifying online attacks based on the resilient software modeling tool (RSMT), which independently observes and characterizes the runtime behavior of web applications. They have also explained how RSMT trains a stacked denoising autoencoder to encode and reconstruct the call graph for end-to-end DL. A low-dimensional representation of the raw features with unlabeled request data was used to recognize anomalies by computing the reconstruction error of the request data.

Second, they have explained how RSMT trains a stacked denoising auto encoder to encode and reconstruct the call graph for end-to-end DL. The reconstruction error of the request data is computed to identify anomalies using a low-dimensional representation of the raw features with unlabeled request data. Thirdly, they have looked at the results of empirically testing RSMT on fictitious datasets and purposefully compromised real-world applications. Ultimately, the results showed that with the least amount of labeled training data and domain expertise, the proposed method could effectively and precisely identify threats like SQL injection, cross-site scripting, and deserialization. A presentation on black-box detection of web application vulnerabilities related to parameter tampering and XQuery injection can be found in [8]. A black-box fuzzing technique has been proposed to find XQuery injection and parameter tampering vulnerabilities in web applications that are driven by native extensible markup language (XML) databases. A functional version of XiParam was developed and tested on unreliable web apps with BaseX, a native XML database, serving as the backend. The experimental analysis clearly shown that the prototype was effective in obstructing the detection of both parameter manipulation and XQuery injection vulnerabilities.

A DL-based method for detecting and preventing SQLI attacks has been presented in the work of [16]. The authors have proposed a SQLI detection technique that leverages NLP and DL frameworks and does not rely on a background rule base, based on a thorough review of both

domestic and international research. Through the process of enabling the machine to automatically detect the language model features of SQLI assaults, the technique has improved detection accuracy, reduced false alarm rates, and offered some defense against attacks that were never detected beforehand.
Using MATLAB, the detection of SQLI attacks has been demonstrated, evaluated, and contrasted using 23 ML classifiers [23]. A model known as ATTAR has been presented by the authors of [24] to identify SQLI assaults by the extraction of SQLI attack features from web access log analysis. A grammatical pattern recognizer and access behavior mining were used to choose the features. Finding unknown SQLI statements that hadn't been utilized in the training set of data was the primary goal of this model. For training, five machine learning techniques were employed: NB, random forest, SVM, ID3, and k-means. The outcomes of the experiment demonstrated that the models based on random forest and ID3 had the highest accuracy in identifying SQL Injection attacks.
A model known as ATTAR has been presented by the authors of [24] to identify SQLI assaults by the extraction of SQLI attack features from web access log analysis. A grammatical pattern recognizer and access behavior mining were used to choose the features. Finding unknown SQLI statements that hadn't been utilized in the training set of data was the primary goal of this model. For training, five machine learning techniques were employed: NB, random forest, SVM, ID3, and k-means. The outcomes of the experiment demonstrated that the models based on random forest and ID3 had the highest accuracy in identifying SQL Injection attacks. For SQLI attack detection, the authors of [25] have developed a hybrid CNN-BiLSTM-based model. The authors provided a thorough comparison of the various ML approaches available for SQLI attack detection. When compared to other ML strategies that have been reported, the CNN-BiLSTM approach yielded an accuracy of almost 98%. An ML classifier has been presented by the developers of [26] to identify SQLI vulnerabilities in PHP code. Several machine learning (ML) algorithms, such as random forest, logistic regression, SVM, multilayer perceptron (MLP), LSTM, and CNN, were trained and assessed. According to the authors, a model based on MLP produced the highest recall (63.7%) and f-measure (74.6%), whereas CNN offered the best precision (95.4%).

Most Common attacks on SQLI

As far as we are aware, creating, updating, and accessing data in a database is possible using the programming language SQL. By carefully constructing SQL instructions, a hacker can purposefully make an application fail, delete data, steal data, or obtain unauthorized access [90]. We present a thorough overview of the several kinds of SQLI attacks that have been found so far in order to address the previously mentioned problem. We offer specific mitigation options along with descriptions and examples of how each form of attack might be executed.

3.1 Tautology attack

In a tautology attack, the attacker tries to utilize a conditional query input, as (1 = 1) or (− −), to test always true. Using the WHERE clause, the attacker injects the condition and turns it into an always-valid tautology [91, 92]. This kind of attack is frequently used to get access to databases on websites without requiring website authentication [1].As a result, the initial condition is transformed into a tautology by the SQL query results, making it possible for an unauthorized user to access every entry in the database table, for example. Many variations of tautological claims in database requests are recognized and prevented by Guardium. The most prevalent tautology assault was the only one examined in earlier research, however all tautology attacks that result in injectables need to be looked at.

3.2 Union Query

In this type of attack, the attacker creates a SELECT statement that resembles the original query and only uses the UNION operator if both queries have the same form [1]. In order to accomplish this, the right table name, number of columns, and data types from the initial query must be known. Each query produces the same amount of columns as a result, and either of two requirements will be met or an attack on the union query will be initiated [12, 13]. The injected query will fail if a column's data type is incompatible with string data Since the string after(–) serves as a comment for the SQL parser, it is ignored. In general, the second query in a union is harmful [94].

3.3 Piggybagged Query

More query statements are concatenated onto the original query ";" in the piggybacked query attack [1]. This method is particularly dangerous since it allows an attacker to inject almost any SQL command. Determinate attacks include things like data extraction, addition, or modification, denial of service (DoS), and remote command execution [16]. An attacker tries to insert new queries into the initial query in this kind of attack. In contrast to previous forms, instead of altering the original query, attackers try to create new and unique inquiries that "piggyback" on it [26]. By initially identifying the correct SQL query through suitable validation or by utilizing a variety of detection systems, these kinds of illegal activity can be prevented. Static analysis can be used to prevent this kind of attack, and run-time monitoring is not necessary.

3.4 Incorrect Query

This type of attacker exploits an incorrectly executed database query [1]. It will display database error messages, which often provide important details that let an attacker figure out the details of the application's database. Finding injectable parameters, doing database fingerprinting, and extracting data are all part of the attack objective. With the help of this technique, an attacker can obtain vital information on the makeup and operation of a web application's back-end database [16]. It is believed that the hack served as a test run for further information-gathering attacks in the future. This attack exploits the fact that application servers' default error pages are typically very informative. This method can be used by an attacker to change the stored procedures in the database [1]. The method will yield true or false results for both allowed and unauthorized users. Users have the option to store and access their features at any time. The ability to use it is accompanied with a set of SQL queries. The hacker utilizes malicious SQL instructions to run the stored procedures that are already installed in the database [12]. This results in the recompiling of the cached stored procedure query plans. The limitation of a stored procedure is that it can only be utilized within the database.

3.5 Stored Procedure Query

This method can be used by an attacker to change the stored procedures in the database [1]. The method will yield true or false results for both allowed and unauthorized users. Users have the option to store and access their features at any time. The ability to use it is accompanied with a set of SQL queries. The hacker utilizes malicious SQL instructions to run the stored procedures that are already installed in the database [12]. This results in the recompiling of the cached stored procedure query plans. The limitation of a stored procedure is that it can only be utilized within the database.

3.6 Inference Query

This attack involves recasting the query as an operation and executing it based on the response of a true-or-false query regarding the values of database data [20]. When using this injection technique, attackers try to breach a site's security to the point where, in the event that an injection is successful, there is no way for users to access feedback in the form of database error messages. Attackers must use a different strategy to obtain a response from the database because error messages in the database are not accessible to offer feedback.

Result And Discussion

Three different injection parameters were used in this study. The first way is through a user input field, which enables a web application to request data from a backend database using HTTP (S) POST and GET. The second way is through cookies, which enable the restoration of a client's state data upon the client's return to a web application. If a web application uses the contents of cookies to create SQL queries, then an attacker can take advantage of this vulnerability to modify cookies and send them to the database server. Finally, by examining session usage data and identifying browsing patterns, a server variable can be produced. Logging these variables to a database without sanitization could result in SQLI vulnerability because attackers can construct requests to the server or insert malicious input into the application's client-end, forging the values in HTTP (S) and network headers. As a result, every attack that is sent to the server is recorded and kept in the database as attack log data. Additionally, attack log data is separated into two groups: attacks and regular data. We trained and evaluated vulnerability classifier models using a variety of machine learning approaches to see which method worked best. Neural networks, conventional NB, DT, SVM, RF, and LR are among the algorithms in the set. based on the hybrid and MLP methodologies that we employ in our research. The confusion matrix of the classification results by taking each class as positive samples separately is shown in Tables 1

Table 1: Performance Evaluation of training set

No	**Techniques**	**Evaluation Metrics**				
		Precision	**Recall**	**F1-score**	**Training set accuracy**	**Training time (in sec.)**
1	**NB**	**88.33%**	**87.89%**	**88.11%**	**89.40%**	**08.73**
2	**DT**	**93.09%**	**92.75%**	**92.92%**	**95.70%**	**53.01**
3	**SVM**	**97.15%**	**98.02%**	**97.58%**	**98.80%**	**19.06**
4	**RF**	**98%**	**94%**	**92%**	**98%**	**9.54**

5	**ANN**	**99.05%**	**99.65%**	**99.35%**	**99.20%**	**19.62**
6	**Hybrid**	**99.54%**	**99.61%**	**99.57%**	**99.60%**	**26.15**

The research presented in Table 1 and Fig. 1 indicates that the hybrid technique (ANN and SVM) outperforms other ML approaches in terms of 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%). But for NB and RF, their training times—19.62 and 26.16 s, respectively—are excessively long. In light of this, the NB approach performs best in training time and poorly in accuracy, precision, recall, f1-score, and training set evaluation metrics.

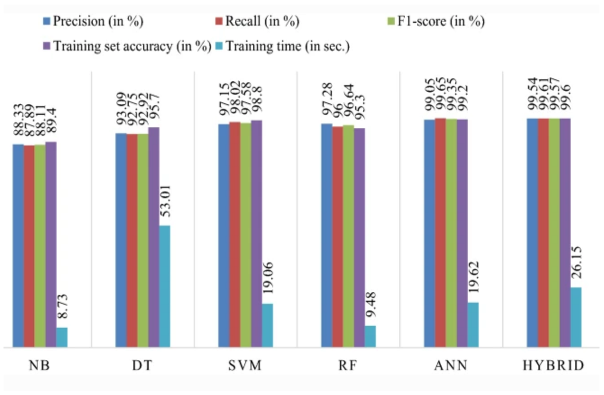

Figure 1: Performance Evaluation of various ML Models

It is advised that future researchers in the field use hybrid techniques with careful consideration of large-scale datasets to improve the performance in the evaluation metrics (detection and prevention rates) of the systems, even though there hasn't been any research on detecting and preventing SQLI attacks that has produced 100% accuracy. As a result, the effectiveness of any SQLI attack systems' detection and prevention rate is dependent upon a thorough analysis of the datasets, taking into account factors like data volume, platform of collection, and technique selection. In conclusion, regarding the advice given in [24], the system's detection and prevention rate has been raised by the expansion of its training and testing datasets and the application of the suggested methodologies.

Conclusion

The most hazardous online application attacker is SQL Injection. Web applications are at serious risk from this kind of attacker, which might have serious ramifications for security and privacy concerns. Attacks against web applications are growing more frequent and powerful. Hackers are inspired to create new assaults by the abundance of data that is readily available online. Numerous investigations have been carried out to lessen the impact of this attack, either by identifying it when it happens or by avoiding it before it starts. We assessed different approaches to SQLI detection and prevention. First, we have outlined the many categories of SQLI attacks that have been identified to date. Then, the methods under review were assessed for their capacity to identify and avert SQL Injection assaults. In order to identify and stop all kinds of SQLI assaults, we determined which DL, ML, and hybrid techniques are most frequently utilized. Additionally, we investigated the various processes and identified strategies that could be used to identify and stop SQL Injection attacks from various web apps. We then determine the requirements for each technique and create a thorough framework for identifying and averting SQLI assaults using machine learning and hybrid approaches. We looked into whether hybrid and ANN are the most effective methods for categorizing SQLI according to our assessment of the model's performance.

Furthermore, the test set's performance evaluation results in metrics like the hybrid approach (ANN and SVM) outperform other ML approaches in terms of accuracy 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%). However, their test time is excessively high (i.e., 11.76 and 15.33 ms respectively). As a result, the NB technique performs poorly in terms of accuracy, precision, recall, f1-score, test set evaluation metrics, and best in terms of training time. Here, SVM and ANN are weak learners among the implemented ML techniques. Ultimately, the goal of this study project was to provide researchers with current information, insights, and suggestions regarding the relationship between AI prevention and SQLI attacks. Future researchers should try maximizing the dataset and experimenting with other approaches in a real-world setting.

REFERENCES

1. Johny JHB, Nordin WAFB, Lahapi NMB, Leau YB. SQL Injection prevention in web application: a review. In: Communications in computer and information science, vol. 1487 CCIS, no. January. 2021. p. 568–585. https://doi.org/10.1007/978-981-16-8059-5_35.
2. Alghawazi M, Alghazzawi D, Alarifi S. Detection of sql injection attack using machine learning techniques: a systematic literature review. J Cybersecur Privacy. 2022;2(4):764–77.
3. Han S, Xie M, Chen HH, Ling Y. Intrusion detection in cyber-physical systems: techniques and challenges. IEEE Syst J. 2014;8(4):1052–62.
4. Dasmohapatra S, Priyadarshini SBB. A comprehensive study on SQL injection attacks, their mode, detection and prevention. 2021. p. 617–632. https://doi.org/10.1007/978-981-16-3346-1_50.
5. Hu J, Zhao W, Cui Y. A survey on SQL injection attacks, detection, and prevention. In: ACM international conference on proceeding series, no June. 2020. p. 483–488. https://doi.org/10.1145/3383972.3384028.
6. Blog. What is SQL injection attack? Definition & FAQs|Avi networks.
7. Imperva. SQL (structured query language) injection. Imperva. 2021.
8. Deepa G, Thilagam PS, Khan FA, Praseed A, Pais AR, Palsetia N. Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications. Int J Inf Secur. 2018;17(1):105–20. https://doi.org/10.1007/s10207-016-0359-4.
9. Dizdar A. SQL injection attack: real life attacks and code examples. 2022.
10. Pan Y, et al. Detecting web attacks with end-to-end deep learning. J Internet Serv Appl. 2019. https://doi.org/10.1186/s13174-019-0115-x.
11. Zhang W, et al. Deep neural network-based SQL injection detection method. Secur Commun Networks. 2022;2022:1–9. https://doi.org/10.1155/2022/4836289.
12. Pattewar T, Patil H, Patil H, Patil N, Taneja M, Wadile T. Detection of SQL injection using machine learning: a survey. Int Res J Eng Technol (IRJET). 2019;6(11):239–46.
13. Fang Y, Peng J, Liu L, Huang C. WOVSQLI: detection of SQL injection behaviors using word vector and LSTM. In: ACM international conference on proceeding series. 2018. p. 170–174. https://doi.org/10.1145/3199478.3199503.
14. Li Q, Wang F, Wang J, Li W. LSTM-based SQL injection detection method for intelligent transportation system. IEEE Trans Veh Technol. 2019;68(5):4182–91. https://doi.org/10.1109/TVT.2019.2893675.
15. Chen D, Yan Q, Wu C, Zhao J. SQL injection attack detection and prevention techniques using deep learning. J Phys Conf Ser. 2021;1757(1):012055. https://doi.org/10.1088/1742-6596/1757/1/012055.
16. Abaimov S, Bianchi G. A survey on the application of deep learning for code injection detection. Array. 2021;11(June):100077. https://doi.org/10.1016/j.array.2021.100077.
17. Son S, McKinley KS, Shmatikov V. Diglossia: detecting code injection attacks with precision and efficiency. Proc ACM Conf Comput Commun Secur. 2013;2:1181–91. https://doi.org/10.1145/2508859.2516696.
18. Yan R, Xiao X, Hu G, Peng S, Jiang Y. New deep learning method to detect code injection attacks on hybrid applications. J Syst Softw. 2018;137:67–77. https://doi.org/10.1016/j.jss.2017.11.001.
19. P. Vähäkainu and M. Lehto, "Artificial intelligence in the cyber security environment," Proc. 14th Int. Conf. Cyber Warf. Secur. ICCWS2019 Artif., 2019.
20. Singh G, Kant D, Gangwar U, Singh AP. SQL injection detection and correction using machine. |In: Emerging ICT bridging future—proceedings of the 49th annual convntion of Computer Society of India, vol. 1. 2015. p. 435–442. https://doi.org/10.1007/978-3-319-13728-5.
21. Marashdeh Z, Suwais K, Alia M. A survey on SQL injection attack: detection and challenges. 2021.
22. Hasan M, Balbahaith Z, Tarique M. Detection of SQL injection attacks : a machine learning approach. In: 2019 international conference on electrical computing technologies and applications. 2019.
23. Gao H, Zhu J, Liu L, Xu J, Wu Y, Liu A. Detecting SQL injection attacks using grammar pattern recognition and access behavior mining. In: 2019 IEEE international conference on energy internet. 2019. p. 493–498. https://doi.org/10.1109/ICEI.2019.00093.
24. Gandhi N, Patel J, Sisodiya R, Doshi N, Mishra S. A CNN-BiLSTM based approach for detection of SQL injection attacks. In: 2021 international conference on computational intelligence and knowledge economy. 2021. p. 378–383.
25. Zhang K. A machine learning based approach to identify SQL injection vulnerabilities. In: 2019 34th IEEE/ACM international conference on software engineering and automation. 2019. p. 1286–1288. https://doi.org/10.1109/ASE.2019.00164.
26. Li Q, Li W, Wang J, Cheng M. A SQL injection detection method based on adaptive deep forest. IEEE Access. 2019;7:145385–94.
27. Uwagbole SO, Buchanan WJ, Fan L. An applied pattern-driven corpus to predictive analytics in mitigating SQL injection attack. In: 2017 seventh international conference on emerging security technologies. 2017. https://doi.org/10.1109/EST.2017.8090392.
28. Ahmed M, Uddin MN. Cyber attack detection method based on nlp and ensemble learning approach. In: 2020 23rd international conference on computer information technology (ICCIT). 2020. https://doi.org/10.1109/ICCIT51783.2020.9392682.
29. Tripathy D, Gohil R, Halabi T. Detecting SQL injection attacks in cloud saas using machine learning. 2020.