International Journal of Innovative Research in Engineering & Management (IJIREM)
ISSN: 2350-0557, Volume-9, Issue-5, October 2022
https://doi.org/10.55524/ijirem.2022.9.5.2
Article ID IJIR-1252, Pages 11-17
www.ijirem.org

A Comprehensive Review for Detection and Prevention Techniques for SQL Injection Attack in Cloud Computing

Munish Saran[1], Rajan Kumar Yadav[2], Pranjal Maurya[3], Sangeeta Devi[4], and Upendra Nath Tripathi[5]

[1,2,3,4] Research Scholar, Department of Computer Science, DDU Gorakhpur University, Gorakhpur, India
[5]Associate Professor, Department of Computer Science, DDU Gorakhpur University, Gorakhpur, India

ABSTRACT- In today's world web applications are integral part of our day today life. Currently there are infinite numbers of web users around the world. These web applications allows users to use the services provided by them upon just a simple clicks from anywhere in the world. Due to rapid growth as well as competition in the business the service providers are making use of the web applications to attract the user. Some of the common examples of the web applications are banking applications, social networking applications, ecommerce applications etc. There exists a variety of attacks that imposes threat on these web applications. One of such attack is known as SQL Injection attack. Research has shown that about 64% of the overall web applications running worldwide are prone to SQLIA. SQL injection is a SQL code injection technique, which forces the database to execute malicious SQL commands that can perform unwanted actions on the underlying database such as getting access to private information or even deleting the entire tables or the database itself. So the prevention against such an attack is must for the web applications.
Various research work in this area have been carried out so as to provide better and more accurate defence mechanism against SQLIA, but still the incident of SQLIA are reported time and again even with big cloud service providers. This paper reviews some latest work from some of the best journals in this area.

KEYWORDS- QL injection attack(SQLIA), Cloud Security, Machine Learning, SQL injection vulnerability, Web application, Structured Query Language.

I. INTRODUCTION

The web application provides the services as requested by the user (client) by taking the request in form of user inputs. Then this submitted request is posted back to the web server and accordingly the appropriate service is invoked and the result is returned to the client. In this overall life cycle of processing any type of request the destination is database. During the request submission the attacker can impose the SQL Injection Attack. Some of the major SQLIA goals for the attacker are Database finger printing, Extracting data, Modifying data, Modifying database schema, Remote control, Bypassing authentication. Fig 1 shows the working of SQL Injection.

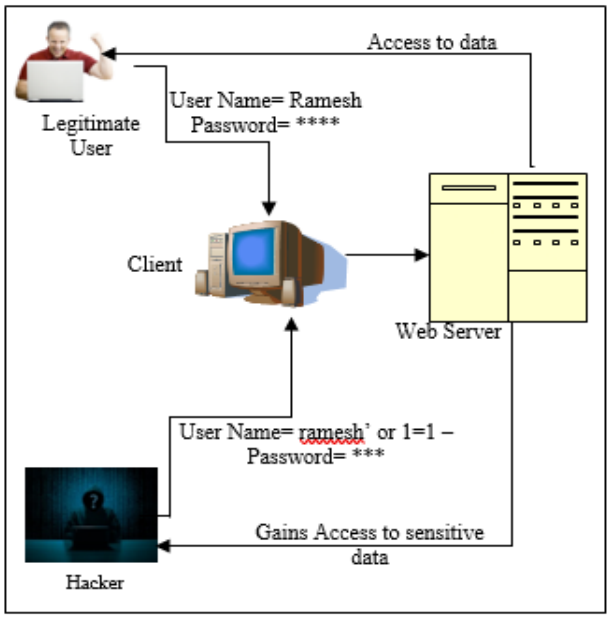

Figure 1: SQL Injection Attack Working

II. ATTACK IMPLEMENTATION MECHANISM

Malicious SQL commands can be introduced by the attackers into a web application which relies on taking the inputs by many different input mechanisms. Some of the common input mechanisms are discussed in this section. [19, 20]

- **Injection through user input** - Web application can read input of the user that the users provide in several controls such as text-boxes, text areas, password fields etc. and this input is submitted via HTTP GET or POST requests. In this case, the attackers inject malicious SQL commands in these user input controls.
- **Injection through cookies** - Cookies are small amount of information that are stored on clients machine. The stored information contains client's state information generated by Web applications, attacker may misuse this cookie's contents which can initiate sql injection (if this web application uses the cookie's contents to build SQL queries.
- **Injection through server variables** - Web applications use these server variables in many ways, such as counting the number of users visited the web

application, for logging of usage information, identifying browsing trends etc. Modification to these variables causes sql injection attack on the web application.

- **Second-order injection -** In this type the attackers enters malicious inputs to the database directly and when this input is used at a later time, SQL Injection Attack gets initiated. For example:- If an attacker tries to impersonate the admin (whose user name = "admin123") and changes the password of the admin in the application with a new password. The sql query for the updation of password is as follows:

Query_Str="UPDATE users_details SET user_password='" + new_password +"' WHERE user_Name='" + user_name + "' AND password='" +old_password + "'"

The injected malicious sql query will be as following (here it is assumed that the old password is old_Admin and the new password is new_Admin).

UPDATE users_details SET user_password =" new_Admin" WHERE userName="admin123"--" AND password= "old_Admin".

As "--" is treated as comment operator in SQL and everything following – will be ignored by the database. Therefore, the attacker is successfully able to update the admin password in the database.

III. SQLI NJECTION ATTACKS TYPES

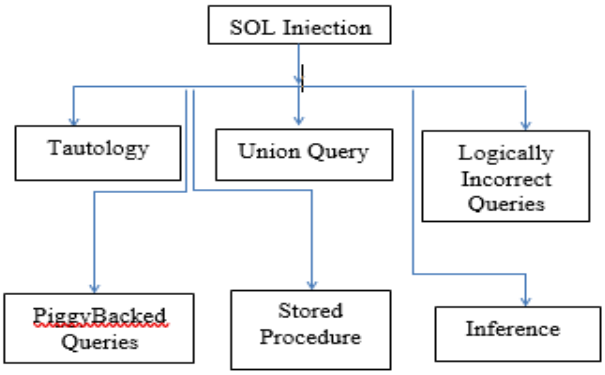

Figure 2: SQL Injection Attack Types

- **Tautologies-**In Tautology type of sql injection attack all the condition result including the negative ones becomes true all the time. Thus bypassing the actual logic and leading to unauthorized access to the database as well as exploiting the sensitive data. Since the condition is to be set as true every time in this type of SQLIA, the fields in the WHERE condition is targeted by the attacker. [16] For example if an attacker inputs user id as ADMIN and password as anything' or 'Y'='Y, the query thus formed will be as follows

  SELECT * from user WHERE user_id = 'ADMIN' and password = 'anything' or 'Y'='Y'

  The above mentioned query becomes a tautology condition and evaluates to true giving unauthorized access the attacker.
- **Logically Incorrect Queries-** Logically Incorrect Queries- The attacker in this type of attack intentionally gives commands that causes logical, syntax or conversion errors. Information such as database name, tables names or column names used in the database are exposed by making use of such type of syntax errors in the sql commands.

  For example the error message given by an application which is using MICROSOFT SQL SERVER as the backend database as "Microsoft OLE DB Provider for SQL Server" is due to failed conversion of a given data type to another data type caused by an incorrect query given by the attacker. If the attacker gives the input as "convert (float , (select employee_name from sysobjects where xtype='u'))".

  The resultant query thus formed will be as

  SELECT acc_details from bank_details WHERE password = convert (float , (select employee_name from sysobjects where xtype='u')).

  In this case the attacker tries to fetch the table name from the metadata table and then perform type casting of table name into float data type which is an invalid type conversion. Thus application displays the Microsoft OLE DB Provider for SQL Server" error message exposing the backend database name "Microsoft SQL SERVER" as well as the table's column name "account_info" due to incorrect type conversion.
- **Union Query-** Union query allows, an attacker fetch information or the sensitive data from a table other than the original table that is meant to provide that particular information by injecting an sql query including UNION SELECT clause. [20]

  For example by injecting UNION clause such as: "UNION SELECT payment_details from Payment where acctNumber=563214 - -" in the login control field of the application, which results in the following sql query:

  SELECT User_Info FROM users_details WHERE login= "UNION SELECT payment_details from Payment where acctNo=563214 - -".

  There is no user details available in the user_details table with login equal to "" in the application, so the first query returns null value in this case while the second query returns data from the table "Payments", such as the payment details of the user for account number = 563214.
- **PiggyBacked Queries-** This type of SQLIA allows the attacker to piggyback some additional queries to the original sql command.

  Example : If an attacker wants to delete the table schema from the database, he can inject the following value in the login field of the application "'; drop table ACCOUNT_DETAILS –". That results into the following sql query at the backend:

  SELECT user_info FROM user_details WHERE login_id ='SAM' AND login_password= "'; drop table ACCOUNT_DETAILS –".

  As the delimiter marks the end of the first query and treat everything after it as complete second query and thus goes on to delete the ACCOUNT_DETAILS table from the database due to additional piggyback query.
- **Stored Procedures-** Stored procedures are the major backbone for the backend database. These stored procedures are required to run sql queries and fetch the data from the database in the application. [21] The use of stored procedure minimizes the risk of many attacks

including SQLIA as well as protects from direct exposure of sensitive data at the same time. The stored procedures are provided with the capability of interacting with the operating system in order to accomplish several task. But this feature of the stored procedure can be misused by the attacker and cause SQLIA.

For example in order to shutdown the operating system, the attacker may inject the input as "'; SHUTDOWN;" in user id field of the application and thus forming the sql command as

SELECT * from user_info FROM users_details WHERE login_id ='sam' AND login_password=" "'; SHUTDOWN;".

Although the below mentioned stored procedure is used to execute the command in the application, but the command "'; SHUTDOWN; forces the operating system to shut down as soon as it is executed.

```
CREATE PROCEDURE CHECK_USER_AUTHENTICATION
@userName varchar2, @userPassword varchar2
AS
EXEC("SELECT user_info FROM users_details WHERE login_id='" +@userName+ "' and password='" +@ userPassword );
        GO
```

Fig 2 display all the types of SQL Injection Attacks.

IV. PREVENTION TECHNIQUES

A. Prepared Statement

One of the simple ways to avoid SQLIA is to make use of prepared statement which uses parameters for the values to be inserted in the database instead of directly inserting the user input which may contain malicious scripts capable for causing SQLIA. [27]

```
string Command1 = "Select Count(ID) from tblEmployees where UserName = @UserName and Password=@Password";
```

In the above query @UserName and @Password are parameters to the query.

```
string Command2 = "select * from tbl_customers where city_name = @city";
```

In the above query @city is parameters to the query.

B. Stored Procedures

Stored procedures are stored in the database containing all the commands that are to be executed when invoked from the web application. In this way all the user inputs are not allowed to form a sql query directly and execute in the database but rather they are given to stored procedures to which in turn forms legitimate query to get executed. [28]

```
CREATE PROCEDURE stpUpdateMemberByID
        @MemberID int,
        @MemberName varchar(50),
        @MemberCity varchar(25),
        @MemberPhone varchar(15)
         AS
        BEGIN
           UPDATE tblMembers
           Set MemberName = @MemberName,
              MemberCity = @MemberCity,
              MemberPhone = @MemberPhone
           Where MemberID = @MemberID
        END
    GO
```

C. Validating User Input

User input is captured by the web application in various web application controls. Web applications must make use of validations of several types on each of the such controls used for receiving inputs from the user which basically checks the syntax of the user input. Examples for such validation controls are in ASP.NET are RequiredFieldValidation Control, CompareValidator Control, RangeValidator Control, RegularExpressionValidator Control, CustomValidator Control, ValidationSummary. [21, 25]

D. Encrypting Data

All the data stored in the database must be in encrypted form, so that in any case if the malicious query gets an entry in the database must not able to read the data and thus bypass SQLIA. Below is an example of how to create a encrypted stored procedure in sql server. [29, 30]

```
CREATE PROCEDURE dbo.usp_GetCatsByName @catname varchar(70)
    WITH ENCRYPTION
    AS
    SELECT CatId, CatName, Phone
            FROM dbo.Cats WHERE CatName = @catname;
GO
```

E. Limiting Privileges

User access is restricted in case of limiting privileges. According to the specific authorization, users must be allowed to have access to the database so as to prevent unauthorized user passing malicious sql query which may lead to SQLIA. [18]

V. LITERATURE REVIEW

Some of the tools and techniques for detecting and preventing SQL injection are given below:

Gu et al. [1] proposed a framework for the detection of SQLIA named as DIAVA which is capable of recognizing the malicious SQL queries exchanged in the application and notify the admin about the attack. The damage caused by the SQLIA is rapidly analyzed by this framework and thus proves out to be very effective in detecting as well as preventing of SQLIA on the web applications. DIAVA performs SQLIA detection via a model which is based on regular expression and the evaluation of disclosed data is performed by making use of dictionary based engine.

Tripathy et al. [2] gave a solution to detect the SQLIA in web application by the use of machine learning. The result showed that the machine learning algorithm detected the malicious SQL queries with more than 98% accuracy from the normal SQL queries and thus proves out to be a better solution for prevention against SQLIA. The pre-processing of data such as data cleaning is performed as the data is collected from various sources which is followed by feature selection to determine the best set of optimized features that are actually responsible for the attack. The approach make used for the following classifier for model selection namely AdaBoost Classifier, Random Forest, SGD, Tensorflow Linear classifier, Decision Tree, Deep AN and Tensorflow boosted tree. The experimental results showed that the

among these algorithms RandomForest with 10 selected features showed better results for Precision, Accuracy, F1 score , Recall, specificity and sensitivity.

Aliero et al. [3] proposed to automate the SQLIV (SQL Injection Vulnerability) in SQLIA by making use of black box testing. In this context the author proposed an improved scanner based on object-oriented methodology that reduces the false negative as well as false positive results for SQL injection vulnerability in SQLIA. The scanner works with four major components which are crawling, attacking, analysis as well as reporting. The accuracy of the proposed scanner is tested against three vulnerable applications and the result showed that the proposed scanner is much efficient than the existing ones.

Another machine learning approach for the detection as well as prevention of SQLIA is proposed by Hasan et al. [4]. This model lies between the application and the application database so as to allow only non-malicious sql query to get executed in the database. Among the twenty three different machine learning classifiers which were used to check the accuracy, it was discovered that the Ensemble Boosted Tree achieved an accuracy of 93.8%.

Wang et al. [5] proposed long short-term memory (LSTM) deep learning methodology for detection of SQLIA for resolving the security concerns in the transport systems. This methodology depicts the high accuracy and false positive rate in comparison with other traditional SQLIA approaches which has high false negative rate and false positive rate. The risk of over fitting the dataset is drastically reduced by the proposed LSTM based deep learning model.

Latchoumi et al. [6] proposed another machine learning approach as a defence mechanism against SQLIA. For classification of normal SQL queries against the malicious SQL queries, all the SQL queries are tested against well trained SVM algorithm which is capable enough to detect the malicious queries and thus guard against SQLIA.

Hlaing et al. [7] also suggested a SQLIA detection as well as prevention mechanism in their research work. The proposed approach works in two steps. The input sql query is divided into set of tokens which are double dashes (--), space ( ), sharp symbol (#) as well as strings that are preceded by symbol. After the collection of tokens from the given input sql query, this token set is checked against the predefined reserved lexions (which are predefined legitimate sql commands, operators, symbols etc.) so as to prevent SQLIA. If match is found between token set and predefined reserved lexions, then it is concluded that the SQLIA was attempted and the given sql query is not allowed to get executed in the database. The outcome of the research showed better result for prevention against SQLIA.

An adaptive deep forest (ADF) model for the purpose of detecting SQLIA is suggested by LI et al. [8] instead of traditional machine learning models which are prone to over-fitting and thus produced better results in terms of precision, accuracy, f1-score and recall. The attack detection accuracy is improved to a great extent in ADT as compared to deep forest due to use of multi-grained scanning for feature transformation and layer by layer learning. Apart from improved detection accuracy high robustness, flexibility as well as reduced computational cost is also achieved by ADT (adaptive deep forest).

Knuth-Morris-Pratt algorithm for detection as well as prevention of SQLIA and XSS is given by Abikoye et al. [9]. The KMP algorithm is used to create a filter() function which is responsible for matching the given user input against the stored malicious strings capable for causing SQLIA. The proposed methodology not only detects the all types of SQLIA including union-based, boolean-based, batch query, error-based, like-based but also sends alert messages to the admins as well as creates the corresponding log records.

Durai et al. [10] proposed a detection as well as prevention methodology against SQLIA for web application that are hosted in cloud environment and is making use of http or https protocols. They named this methodology as SQLIO (SQL Injection based on Ontology). The ontology model has three frameworks in this proposed methodology namely attack, vulnerability and threat model. The threat model figures out the category of the attacks while the vulnerability model list out the vulnerability of the web application and the attack model protects against these attacks. This model helps in categorizing the attacks according to the severeness of the attack caused on the web application and thus proves out to be crucial in detecting as well as protection against SQLIA on cloud based web application.

Another mitigation technique against cloud based applications is given by Patil et al [11]. The query given by the user is divided into tokens and all the tokens thus formed are compared with tokens that are capable of causing SQLIA at the web server end. And in case of any suspicious tokens found in the sql query, the query is returned by the web servers preventing the SQLIA. This implementation has given better results for cloud applications which suffers from numerous security threats.

SQLIA prevention framework for cloud application at the application level is also given by Yassin et al. [12]. The framework is hosted on Amazon Web Services and can be effectively adopted by the SaaS providers as their security framework. The http request for a hosted cloud service is thoroughly examined for an suspicious tokens capable for SQLIA as well as relationship of time between the query and request is also monitored by this SQLIDaaS framework.

A predictive analysis utilizing machine learning algorithm is proposed by Uwagbole et al. [13] for detection and protection mitigation scheme against SQLIA. The data set is prepared which contains symbols and tokens responsible for SQLIA and pre-processed to follow machine learning pipeline. The application is trained and tested with the data set against a web application hosted on cloud environment. The machine learning classifier acts as the prevention for malicious sql request thus acting as a defense against SQLIA.

Leelavathy et al. [14] gave a defence methodology against DDoS and various types of SQLIA in their research paper. Request packets are inspected and a software puzzle approach is applied to detect the suspicious request that may become cause for SQLIA. The summary for literature review is summarized in Table 1 and 2 shows the comparison of SQL Attacks in various techniques and Table 3 distinguishes the various works on the basis of detection or prevention techniques.

Table 1: Summary of the approaches proposed to defense against SQLIA

REFRRENCES	PROPOSED FRAMEWORK/TOOL	TECHNIQUE USED	ADVANTAGE
Gu et al. [	DIAVA	Regular Expression based model	Detection as well as prevention against SQLIA.
Tripathy et al. [2]	ML Model	Random Forest ML Classifier	High accuracy in detection for SQLIA.
Aliero et al. [3]	Scanner	Black box testing	Accuracy testing done against three vulnerable web applications.
Hasan et al. [4]	ML Model	Ensemble Boosted Tree.	93.8% accurate results obtained.
Wang et al. [5]	ML Model	Long Short-Term Memory (LSTM) based on deep learning	Overfitting of the dataset is reduced to a great extent thus giving better results.
Latchoumi et al. [6]	ML Model	SVM Algorithm	More accurate detection of malicious sql query is attained
Hlaing et al. [7]	Scanner	Predefined lexions based checking of sql queries	Capable of sanitizing the sql query by matching against predefined list of tokens
LI et al. [8]	Adaptive Deep Forest	Deep Learning	Overfitting of the dataset is reduced to a great extent thus giving better results.
Abikoye et al. [9]	-	Knuth-Morris-Pratt algorithm	Detection of all types of SQLIA is made possible
Durai et al. [10]	SQL Injection based on Ontology (SQLIO)		Capable for SQLIA detection as well as prevention for cloud based applications.
Patil et al [11]	IDS	Comparison against malicious tokens	Better defence against SQLIA is achieved.
Yassin et al. [12]	SQLIDaaS	Analysis of web service request	Defence against SQLIA for SaaS based cloud web applications.
Uwagbole et al. [13]	ML Model	ML based classification	Detection as well as Prevention for SQLIA increased with ML Model.
Leelavathy et al. [14]	Deep Packet Inspection (DPI)	Software puzzle based approach	Defence against DDoS and SQLIA made possible

Table 2: Comparison of various techniques with respesct to various types of sql injection attacks

REF	TUT	ILL	PIG	UNI	SP	INF	ALT
[1]	YES	YES	YES	YES	YES	YES	YES
[2]	YES	YES	YES	YES	YES	YES	YES
[3]	YES	YES	YES	YES	YES	YES	NO
[4]	YES	NO	YES	NO	YES	YES	YES
[5]	YES	YES	YES	YES	YES	NO	NO
[6]	YES	NO	NO	YES	YES	YES	YES
[7]	YES	YES	NO	NO	YES	YES	YES
[8]	YES	YES	YES	NO	YES	NO	YES
[9]	YES	YES	YES	YES	YES	YES	NO
[10]	YES	YES	YES	YES	YES	YES	YES
[11]	YES	YES	YES	YES	YES	YES	YES
[12]	YES	YES	YES	NO	YES	YES	YES
[13]	YES	YES	NO	YES	NO	YES	YES
[14]	YES	YES	YES	YES	YES	NO	NO

TUT: Tautology, ILL:- Illegal/Incorrect, PIG:- Piggy-back, UNI:- Union, SP:- Stored-Procedure, INF:- Inference, ALT:-Alternate Encoding

Table 3: Comparison of various techniques with respesct to detection and prevention

Methodology	Detection (D) Prevention (P)
DIAVA based on regular expression [1]	D/P
RandomForest classifier [2]	D/P
Scanner based on object-oriented [3]	D
Ensemble Boosted Tress classifier [4]	D/P
Deep learning using LSTM [5]	P
SVM algorithm [6]	D/P
Predefined reserved lexions [7]	D
Adaptive deep forest [8]	D/P
Knuth-Morris-Pratt algorithm [9]	D/P
SQL Injection based on Ontology [10]	D/P
Multilevel System[11]	D/P
SQLIDaaS framework [12]	D/P
Machine Learning Predictive Analysis [13]	D/P
Deep Packet Inspection [14]	D/P

VI. CONCLUSION AND FUTURE WORK

Security of database is one of the prime issue which is to be taken care of. There are several types of attacks in the web applications where SQL Injection also has an important place. The attackers can inject the malicious sql code at various points of the application and can gain access to the database. There are several types of tools and frameworks available working on different technique for detection/prevention for SQL Injection attack, some of which is discussed in this paper. In our future work we propose a development of a security mechanism for ensuring the security for SQL injection attack for the web

applications and testing its effectiveness, efficiency, and performance.

REFERENCES

[1] Gu H., Zhang J., Liu T., Hu M., Zhou J., Wei T., Chen M, "DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data". IEEE TRANSACTIONS ON RELIABILITY, pp. 188-202, 2019.

[2] Tripathy D., Gohil R., and Halabi T., "Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning". IEEE International Conference on Big Data Security on Cloud (BigDataSecurity), High Performance and Smart Computing (HPSC) and Intelligent Data and Security (IDS), 2020.

[3] Aliero M.S., Ghani I., Qureshi K.N., Rohani M.F, "An algorithm for detecting SQL injection vulnerability using black-box testing". Journal of Ambient Intelligence and Humanized Computing, pp. 249-266, 2019.

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

[5] Li Q., Wang F., Wang J., Li W., "LSTM-Based SQL Injection Detection Method for Intelligent Transportation System". IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, pp. 4182-4191, 2019.

[6] Latchoumi T.P., Reddy M.S., Balamurugan K., "Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention". European Journal of Molecular & Clinical Medicine, 2020.

[7] Hlaing Z.C.S.S., Khaing M., "A Detection and Prevention Technique on SQL Injection Attacks". IEEE Conference on Computer Applications(ICCA), 2020.

[8] LI Q., LI W., WANG J., CHENG M., "A SQL Injection Detection Method Based on Adaptive Deep Forest". IEEE Access, pp. 145385-145394, 2019.

[9] Abikoye O.C., Abubakar A., Dokoro A.D., Akande O.N., Kayode A.A, "A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm". EURASIP Journal on Information Security, 2020.

[10] Durai K.N., Subha R., Haldorai A, "A Novel Method to Detect and Prevent SQLIA Using Ontology to Cloud Web Security". Wireless Personal Communications, 2020.

[11] Patil A., Athawale S.V., Tathawade. P., Laturkar A., Takale R., "A Multilevel System to Mitigate DDoS, Brute force and SQL Injection Attack for Cloud Security". IEEE, International Conference on Information, Communication, Instrumentation and Control, 2017.

[12] Yassin M., Slimane H., Talhi T., Boucheneb H., "SQLIIDaaS: A SQL injection intrusion detection framework as a service for SaaS providers". IEEE 4th International Conference on Cyber Security and Cloud Computing, 2017.

[13] Uwagbole S.O., Buchanan W.J., Fan L., "Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention". 3rd International Workshop on Security for Emerging Distributed Network Technologies, 2017.

[14] Leelavathy S., Jaichandran R. Shobana R., Bhaskaran S., Aravindh, Prathyunnan., "A Secure Methodology to Detect and Prevent Ddos and Sql Injection Attacks". Turkish Journal of Computer and Mathematics Education, 2021.

[15] Jemal I., Cheikhrouhou O., Hamam H. Mahfoudhi A., "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning". International Journal of Applied Engineering Research, pp. 569-580, 2020.

[16] Hu J., Zhao W., Cui Y., "A Survey on SQL Injection Attacks, Detection and Prevention". ICMLC: International Conference on Machine Learning and Computing, pp. 483-488, 2020.

[17] Uwagbole S.O., Buchanan W.J., Fan L., "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention". IFIP/IEEE International Symposium on Integrated Network Management, 2017.

[18] Alwan Z.S., Younis M.F., "Detection and Prevention of SQL Injection Attack: A Survey". International Journal of Computer Science and Mobile Computing, pp. 5-17, 2017.

[19] Marashdeh Z., Suwais K., Alia M., "A Survey on SQL Injection Attack: Detection and Challenges". International Conference on Information Technology (ICIT), 2020.

[20] Sharma K., Bhatt S., "SQL injection attacks - a systematic review". International Journal of Information and Computer Security, pp. 493-509, 2019.

[21] Fu X., Wang Z. , Chen Y., Chen Y., Wu H., "SQL Injection in Cloud: An Actual Case Study". International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 137-147, 2019.

[22] Xiao F., Zhijian W., Meiling W., Ning C., Yue Z., Lei Z., Pei W., Xiaoning C., "An old risk in the new era: SQL injection in cloud environment". International Journal of Grid and Utility Computing, pp. 43-54, 2021.

[23] Kourai, K., T. Azumi, and S. Chiba. A self-protection mechanism Pietraszek T., Berghe C.V. Defending against Injection Attacks through Context-Sensitive String evaluation. Recent Advances in Intrusion Detection, pp: 124-145.

[24] Su Z., Wassermann G. The essence of command injection attacks in _b applications. ACM Symposium on Principles of Programming Languages.

[25] Hegde A.K., Jayanthi P.N., "A Survey on SQL Injection Attacks and Prevention Methods". International Research Journal of Engineering and Technology, pp. 535-537, 2020.

[26] McClure RA., Kruger I.H., "SQL DOM: compile time checking of dynamic SQL statements". International Conference on, pp. 88- 96.

[27] Wei K., Muthuprasanna M., Kothari S. Preventing SQL Injection Attacks in Stored Procedures. Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06 IEEE).

[28] Chowdhury S., Nandi A., Ahmad M., Jain A., Pawar M., "A Comprehensive Survey for Detection and Prevention of SQL Injection". 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021.

[29] Bhateja N., Sikka S., Malhotra A., "A Review of SQL Injection Attack and Various Detection Approaches". Smart and Sustainable Intelligent Systems, 2021.

[30] Johny J.H.B., Nordin W.A.F.B., Lahapi N.M.B., Leau Y., "SQL Injection Prevention in Web Application: A Review". International Conference on Advances in Cyber Security, 2022.

ABOUT THE AUTHORS

Munish Saran received the Bachelor of Technology (B.Tech.) in Computer Science Engineering (CSE) from Babu Banarasi Das National Institute of Technology & Management and Master of Technology (M. Tech. Gold Medal) in Computer Science Engineering (CSE) from Madan Mohan Malaviya University of Technology. He is currently Ph.D. research scholar in the Department of Computer Science, DDU Gorakhpur University. His research interest includes Cloud Computing, IoT, Machine Learning and Deep Learning. He was previously working in Infosys as senior system engineer for 4 years.

Rajan Kumar Yadav received the Bachelor of Science (B.Sc.) in computer Science from Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur (Uttar Pradesh, India) and Master of Computer Application (MCA) from Madan Mohan Malaviya University of Technology. He is currently Ph.D. Research Scholar in the Department of Computer Science, DDU Gorakhpur University. His Research interest includes Cloud Computing, Machine Learning and IoT.

Pranjal Maurya received the Bachelor of Technology (B.Tech.) in Computer Science Engineering of Technology & Management and Master of Technology (M.Tech.) in Computer Science Engineering (CSE) from Madan Mohan Malaviya University of Technology. She is currently Ph.D. research Scholar in the Department of Computer Science, DDU Gorakhpur University. Her research interest includes WSN, Cloud Computing, IoT, Machine Learning and Deep Learning. She was previously working in Institute of Technology & Management as Assistant Professor for 1 years.

Sangeeta Devi received the Master of Computer Application (MCA) from IGNOU New Delhi and Master of Technology (M.Tech.) from AKTU Lucknow. She is currently Ph.D. research Scholar in the Department of Computer Science, DDU Gorakhpur University. Her research interest includes Data Science, WSN, IoT, Machine Learning and Deep Learning.

Dr. Upendra Nath Tripathi is currently Associate Professor in the Department of Computer Science, Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur. He has 21 years of teaching and research experience. His areas of interests are Database, IoT, Machine Learning, Cloud Computing and Data Science.