iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** V **Month of publication:** May 2020

DOI: http://doi.org/10.22214/ijraset.2020.5222

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

Prevention of SQL Injection using Whitelisting

Manoj Prabhakar M[1], Viswakesavan R[2]
[1, 2]*SRM Institute of Science and Technology*

Abstract: ***Existing vulnerabilities of Web system threaten the regular work of information systems. The most common Web system vulnerability is SQL injection. This is usually done by altering SQL statements that are used within web applications. In our project when the user enter the data into the web application, input validation is used. Then the input is sanitized using whitelisting algorithm. Then prepared statements are used for escaping all user specifed input. Then the data is sent to the database and the information is retrived securly.***
Keywords: SQLi, DBMS, Php

I. INTRODUCTION

SQL Injection(SQLi) is a big threat that is based on the SQL serve database. It is usually done by slight modification to those SQL code so that when server reads it the injection gets executed and the hacker gets the job done. Millions of the web app's are getting affected by SQLi attacks every year. Therefore for identifying and preventing these kind of attack is given most priority. We propose a hybrid model defensive system to identify and prevent these kind of attacks. The proposed hybrid system is made up of PHP and JS(JavaScript). This model is situated in the second level stage where it behaves as a gateway before it enters the database module. With this system the SQL queries that passes through the second level stage i.e. the CGI state a validation checker is added that monitors the all the execution dynamically.

SQLi which is SQL Injection is a type of an injection attack which makes it possible to execute powerful malicious SQL statement. These statements could control a database server in arrearn at any web application. Attacker could use any SQLi vulnerabilities to easily bypass any application security measure. They could go around authorization and authentication of a web app or web pages and retrieve any content of the complete SQL database. They also can use SQL Injection to add, remove and edit records in database. An SQL Injection vulnerability might affect any web app or website that uses a SQL database such as Oracle, MySQL, SQL Server or others. Attackers might vulnerable use it to earn unauthorized access to your data or information of customers personal data, information, intellectual property, trade secrets and more. SQL Injection attacks are one of the oldest, most dangerous, and most prevalent web application vulnerabilities. The Open Web Application Security Project organization OWASP organization lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

II. LITERATURE SURVEY

SQLi could be used in an wide range of ways to ultimate cause serious problems. By levering SQL Injection, an attacker can bypass authentication, access, remove and edit data within a database. In some cases, SQL Injection could even be used to run command on the OS, capability of allowing an attacker to elaborate to huge damaging attacks inside of a network that sits behind a firewall.

SQL Injection could be classified into three significant categories Out of band SQLi, Inferential SQLi, and In band SQLi.

A. In-band SQLi (Classic SQLi)

In-band SQLi is the most general and easy-to-exploit of SQL Injection attacks. In-band SQLi occurs when an attacker is able to use the same link channel to both launch the attack and gather results.

The two most general types of in band SQLi are Error-based SQL Injection and Union based SQLi.

B. Error-based SQLi

Error based SQL Injection is an in band SQLi technique that relies on error messages thrown by the database server to obtain datas about the structure of the database. In some case, error based SQLi alone is enough for an attacker to enumerate an complete database. While errors are very useful during the development phase of a web app, they should be disabled on a live site, or logged to a file with limitation access instead.

C. Union-based SQLi

Union based SQL Injection is in-band SQLi technique that simplies UNION SQL operator to merge the results of two or many SELECT statements into a single result which is then sent back as part of the HTTP response.

D. Inferential SQLi (Blind SQLi)

Inferential SQLi, not like in-band SQLi, this may take long time for an attacker to exploit, therefore, it is just as harmful as any other form of SQL Injection. In a consecutive SQLi attack none of the datas is literally transmitted through the web apps and the attacker unable to see the result of an attack inband like which is why such attacks are equivalently assigned to as blind SQL Injection attacks. Instead, an attacker is could rebuilt the database structure by transmitting payloads, observing the web app's resulting and response behavior of the database server.

There are two types of inferential SQL Injection are in Blind-time-based SQLi and Blind-boolean-based SQL Injection.

E. Boolean-based Blind SQLi

Boolean-based SQLi is an Inferential SQLi technique that relies on transmitting a SQL statements to the database which forces the application to came back in a distant result relies on whether the statement came back a FALSE or TRUE result.

The content within the Hyper Text Transfer Protocol (HTTP) reply will be vary, or remains the same based on the result. This allow an attacker to achieve if the payload used send back false or true, even though there is no data from the database is returned. This attack is typically slow like as especially on huge databases from an attacker will need to specify a database, character by character.

F. Time-based Blind SQLi

Time based SQLi is an surmised SQLi technique that depends on sending a SQL statement that to the database which forces the database to hold for a specified amount of time such as seconds before replying. The react time would indicate to the attacker whether the result of the statement is FALSE or TRUE. Depending on the result, an Hyper Text Transfer Protocol (HTTP) reply would be returned with a delay, or came back quickly. This allows an attacker to depends if the payload used came back false or true, even though no data from the database is came back. The attack is mostly slow such as notably on huge databases from an attacker will demand to enumerate a database character by character.

G. Out-of-band SQLi

Out of band SQLi is not very common, mostly because it relies on features being enabled on the database server being used by the web apps. Out of band SQLi take place when an attacker is cannot be able to use the same channel to cast the attack and gather results. Out of band techniques, offer an attacker an substitute to inferential time based techniques, especially if the server responses are not very reliable such as making an inferential time-based attack unreliable.

Out of band SQLi methods which rely on the SQL database server capability to make Hyper Text Transfer Protocol (HTTP) or DNS inquire to handover data to an attacker. In this case with Microsoft SQL Server's xp_dirtree command, which could be used to make a DNS request to a server and attacker controls, as well as Oracle Database's UTL_HTTP package, that could be used to pass Hyper Text Transfer Protocol (HTTP) request from SQL and SQL or PL to a host which is control attacks..

III. PRPOSED WORK

A. Prepared statements with Parameterized Queries

The proper syntax of how the database queries should be written should be taught to all developers first for a use of prepared statements with the binding of variable bindings also known as parameterized queries. The queries simple to write and find than most of those dynamic queries. Parameterized queries they force the developer that they have to first define all the available SQL codes and mainly to pass them in each of those parameters for the query used for any further purpose. This coding style can also allow the back end database to mainly distinguish between the code and the data, regardless of what the user input is mainly supplied. Prepared statements make sure that the malicious attacker not able to change the main intent of the query, even if the SQL commands are intentionally inserted by an attacker. In the safe , for example, if an attacker were to enter the value of a table' or '1'='1, then the parameter query would not be that easily vulnerable and it would instead look for any username which can literally match the entire value of table' or '1'='1.

B. Stored Procedures

Stored procedures can never be always safe from these SQL injection. However, by using certain standard stored procedure programming constructs has the same effect as of the parameter queries when they are safetly implemented that is a main for most of the stored procedure languages that is being used. They require the database query developer to just build a SQL statements with the parameters which are parameterized automatically unless the developer really does something large out of the default norm. The difference a between prepared statement and a stored procedure is that the SQL query for any stored procedure could be created and also saved in the database server and then called from its application. Both of these techniques

can have the same effectiveness in the prevention of SQL injection so that your organization should choose whether which approach makes it the most sense for you. Note that the Implemented safely means that the stored procedure does not include any of the unsafe dynamic SQL generation method. Database query developers does not usually generate dynamic SQL inside the stored procedure. However eventhough it is possible it should be avoided totally . If it cannot be avoided, then the stored procedure must use the input validation method or proper escaping method to make sure that all the users value supplied input to those stored procedure that cann't be used for injecting SQL codes into dynamically generated queries. Auditors should always be looking for uses of special execute, execute or extra exec within the SQL Server that is in stored procedures. Similar type of audit guidelines are mainly necessary for similar kind of functions for other vendors that is used.

C. Whitelist Input Validation

Various parts of SQL queries are not in legal locations for the use of bind variables, by using the names of the tables or columns, and by using order indicators like DESC or an ASC. In such type of situations, these query redesign or input validation is the most preferable appropriate defense.

For the names for tables or columns, these values come mainly from the code, and not from the user parameters.

But if the user parameter values are being used for targeting many different table names and column names, then the parameter values should be always mapped to the main legal or expected table or column names to make sure that it the unvalidated user input does not end up in the query provided beforehand. This is a symptom for poor design and full re-writing should be considered if the time allows.

D. Escaping All User-Supplied Input

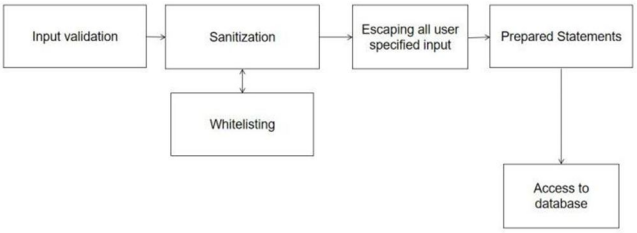
This technique should be only used as a last resort only when none of the above-mentioned methods is feasible. The Input validation is the most probably a better choice when compared as this methodology is more delicate when compared to all other defenses and that we cannot guarantee that it will prevent all the SQL Injection in all the situations.

This mechanism is mostly used to get away from the input of the user before using them in a query. It is especially very database specific in its implementation. It is usually only recommended to retrofit legacy code when they are implementing input validation is not that cost-effective. Applications that are built from the scratch, or applications that are requiring minimum risk tolerance should be created or re-written using stored procedures, parameterized queries or something of Object Relational Mapper (ORM) that do builds our queries for us. In this technique, each of the DBMS supports one of the more characters that the escaping schemes mostly are specific to certain kinds of database queries. If you then do escape all the user-supplied provided input using the most proper escaping scheme for the back end database we are using, the DBMS won't confuse that the input with the SQL code will be written by the database query developer, thus avoiding any of the possible SQL injection dangerous vulnerabilities. A somewhat in a point a special case of escaping it is the process of where the hex-encode is the entire string that is received from the user that this can be seen as the escaping in every character. The web app should be hex-encoded for the outer user input so that before including it in the SQL statement. The SQL statement should take this info into this fact and act accordingly compare the provided data.

S.No	Titile	Content	Author	Year
1	SQL INJECTION ATTACKS AND PREVENTION TECHNIQUES	The method used in this paper is SWART. SWART is an Application Intrusion Detection System tool which secures web application by providing early warning to the attacker or the malicious user then it might be possible that the application is not further exploited for finding the loopholes..	S. Gadgil1, Sanoop Pillai2, Sushant Poojary.	November 7, 2017
2	Prevention of SQL Injection Attack Using Query Transformation and Hashing	In this paper they have proposed a lightweight approach to prevent SQL Injection attacks by a novel query transformation scheme and hashing..	D. Kar, Suvasini Panigrahi	13 May ,2013
3	Understanding and discovering SQL injection vulnerabilities	This paper creates a case study that considers two scenarios using ASP.NET 2015 and SQL Server 2014.	A. A. Sarhan, S. A. Farhan, and F. M. Al-Harby	January 18,2018

4	Detecting SQL vulnerability attack based on the dynamic and static analysis technology	It implements the SQL vulnerability determination algorithm which is based on lexical feature comparison.	Y. Wang, D. Wang, W. Zhao, and Y. Liu	24 September 2015
5	Vulnerability detection and prevention of SQL injection	An automatic detection tool for SQL injection vulnerability based on web crawler is designed and implemented	B J. Santhosh Kumar and P. P. Anaswara	May 4,2018
6	Design and implementation of an automatic scanning tool of SQL injection vulnerability based on Web crawler	Functions such as URL (Uniform Resource Locator) optimization and similarity determination are added to each module's characteristics, so that the vulnerabilities can be scanned more accurately and quickly.	X. Li, J. Qu, G. Yao, J. Chen, and X. Shen	January 21, 2018

IV. ARCHITECTURE DIAGRAM



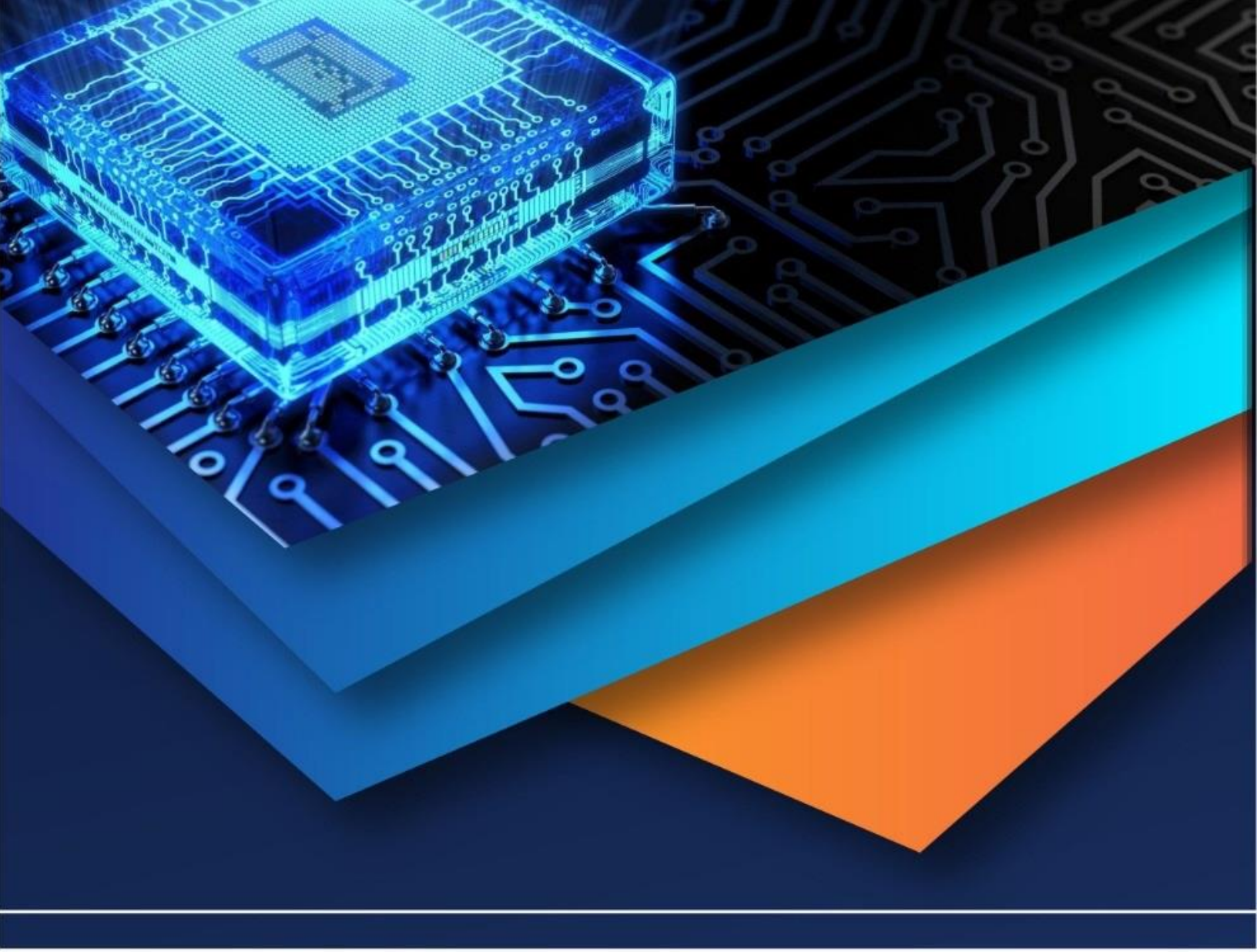
V. CONCLUSION

The main idea of our project is to create a website that provides maximum prevention to a sql injection based attacks and keeping the data protected.

REFERENCES

[1] William. G. Halfond, Alessandro. Orso, "Using Positive Tainting and Syntax Aware Evaluation to Counter SQL Injection Attacks", 14th ACM SIGSOFT international symposium on Foundations of software engineering 2006, ACM.
[2] Sruthi Bandhakavi, Prithvi Bisht, P. Madhusudan, "CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluation", 14th ACM conference on Computer and communications security, ACM, USA, page:12-24.
[3] Marco Cova, Davide Balzarotti. "Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications", International Symposium On Recent Advances in Intrusion Detection, Volume: 4637, Pages:63-86, 2007.
[4] William G.J. Halfond, Jeremy Viegas and Alessandro Orso, "A Classification of SQL Injection Attacks and Countermeasures", IEEE, 2006.
[5] Zhendong Su and Gary Wassermann, "The Essence of Command Injection Attacks in Web Applications", ACM SIGPLAN Notices, Volume: 41, Pages: 372-382, 2006.
[6] Security, DataSunrise (February 1, 2019). "Methods of SQL Injections Detection". DataSunrise Database Security. Retrieved August 28, 2019.
[7] Security, Penta (May 26, 2016). "What is SQL injection and how can you prevent it from happening?". Penta Security Systems Inc. Retrieved August 8, 2019.
[8] "How to Enter SQL Comments", IBM Informix Guide to SQL: Syntax (PDF), IBM, pp. 13–14, retrieved June 4, 2018
[9] Yap, Jamie (July 12, 2012). "450,000 user passwords leaked in Yahoo breach". ZDNet. Archived from the original on July 2, 2014. Retrieved February 18, 2017.Damon Poeter. 'Close-Knit' Russian Hacker Gang Hoards 1.2 Billion ID Creds Archived July 14, 2017,
[10] Nicole Perlroth. Russian Gang Amasses Over a Billion Internet Passwords Archived February 27, 2017
[11] "The Bobby Tables Guide to SQL Injection". Archived from the original on November 7, 2017."; DROP TABLE "COMPANIES";-- LTD". Companies House. Archived from the original on November 7, 2017. Retrieved October 30, 2017.
[12]] Y. Wang, D. Wang, W. Zhao, and Y. Liu, ''Detecting SQL vulnerability attack based on the dynamic and static analysis technology,'' in Proc. IEEE Comput. Softw. Appl. Conf., Jul. 2015, pp. 604–607
[13] D. Mitropoulos, P. Louridas, M. Polychronakis, and A. D. Keromytis, ''Defending against Web application attacks: Approaches, challenges and implications,'' IEEE Trans. Depend. Sec. Comput., vol. 16, no. 2, pp. 188–203, Mar./Apr. 2017
[14] E. Pollack, ''Protecting against SQL injection: Applications, performance, and security in microsoft SQL server,'' in Proc. Dyn. SQL, 2019, pp. 31–60
[15] A. Maraj, E. Rogova, G. Jakupi, and X. Grajcevci, ''Testing techniques and analysis of SQL injection attacks,'' in Proc. Int. Conf. Knowl. Eng. Appl. (ICKEA), 2017, pp. 1–11.
[16] Y. Fang, J. Peng, L. Liu, and C. Huang, ''WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM,'' in Proc. ICCSP, 2018, pp. 170–174.

[17] Q. Li, F. Wang, J. Wang, and W. Li, ‘‘LSTM-based SQL injection detection method for intelligent transportation system,’’ IEEE Trans. Veh. Technol., vol. 68, no. 5, pp. 4182–4191, May 2019
[18] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, ‘‘Improving defect prediction with deep forest,’’ Inf. Softw. Technol., vol. 114, pp. 204–216, Oct. 2019
[19] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ‘‘Deep learning approach for intelligent intrusion detection system,’’ IEEE Access, vol. 7, pp. 41525–41550, 2019.
[20] S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, ‘‘Using deep convolutional neural network architectures for object classification and detection within X-ray baggage security imagery,’’ IEEE Trans. Inf. Forensics Security, vol. 13, no. 9, pp. 2203–2215, Sep. 2018
[21] L. Zheng, L. Yuan, X. Peng, G. Zhu, Y. Guo, and G. Deng, ‘‘Research and implementation of Web application system vulnerability location technology,’’ in Proc. Int. Conf. Adv. Intell. Syst. Comput., 2010, pp. 937–944.
[22] X. Li, J. Qu, G. Yao, J. Chen, and X. Shen, ‘‘Design and implementation of an automatic scanning tool of SQL injection vulnerability based on Web crawler,’’ in Proc. Int. Conf. Secur. Intell. Comput. Big Data Services, 2018, pp. 481–488.
[23] J. Choi, H. Kim, C. Choi, and P. Kim, ‘‘Efficient malicious code detection using N-gram analysis and SVM,’’ in Proc. Int. Conf. Netw.-Based Inf. Syst., Sep. 2011, pp. 618–621.
[24] L. K. Shar and H. B. K. Tan, ‘‘Defeating SQL injection,’’ Computer, vol. 46, no. 3, pp. 69–77, Mar. 2013
[25] B. Appiah, E. Opoku-Mensah, and Z. Qin, ‘‘SQL injection attack detection using fingerprints and pattern matching technique,’’ in Proc. Int. Conf. Softw. Eng. Service Sci., Nov. 2017, pp. 583–587.
[26] L. Li, J. Qi, N. Liu, L. Han, and B. Cui, ‘‘Static-based test case dynamic generation for SQLIVs detection,’’ in Proc. Int. Conf. Broadband Wireless Comput., Nov. 2015, pp. 173–177
[27] W. Yi, L. Zhoujun, and A. Guo, ‘‘Literal tainting method for preventing code injection attack in Web application,’’ J. Comput. Res. Develop., vol. 49, no. 11, pp. 2414–2423, 2012.
[28] 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,’’ in Proc. DSN, Jun. 2015, pp. 172–183.
[29] C. Gould, Z. Su, and P. Devanbu, ‘‘Static checking of dynamically generated queries in database applications,’’ in Proc. 26th Int. Conf. Softw. Eng., May 2004, pp. 645–654
[30] B J. Santhosh Kumar and P. P. Anaswara, ‘‘Vulnerability detection and prevention of SQL injection,’’ Int. J. Eng. Technol., vol. 7, no. 2.31, pp. 16–18, 2018.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY