A Modified Scheme for Preventing web Application against SQL Injection Attack

Kanchan Choudhary
M.Tech CSE
Amity University Haryana

Anuj Kumar Singh
Assistant Professor
Amity University Haryana

Rashmi Gupta
Assistant Professor
Amity University Haryana

ABSTRACT
Web Application are used for providing information and function on World Wide web for various e-commerce organization, medical-care, business and Government sectors which further requires security and vulnerabilities. This paper describes the type of SQL Injection Attacks and discusses the technique, to avoid them. The type of SQL Injection Attack, procedure for preventing from SQL Injection Attacks and related work which was done for this has been considered and explained in this paper. An effective and efficient scheme is proposed to prevent SQL Injection Attack which is locating between web application and database. In such a way to use SQM and Sanitization Application are necessary to extend security or keep from attacker to abusing the database. Through two way evaluations, it is proved that our proposed scheme is more secure and can forcefully cover all bases of our web-based application.

Keywords
SQL Injection Attack, SQM, Sanitization, Web Security.

1. INTRODUCTION
At this time network based operation acts as an essential task in creatural activity. Web application is a set of rules and taken against web services bluffs. Day to day many tasks are now based on web. Web application has become a much approved program for a wide range of services like webmail, online access, Government websites, online retail sales, and many other services. It provides a huge facility to entrance way of database via Internet. From the usages of web services it causes increases the attacks on the web. There are lots of attacks be found in the web application. But the SQL Injection Attack is the most dangerous and challenging attack for the web application [2][4][12].

1.1 Description of SQL Injection Attack
SQL Injection Attack is the popular method of hacking or cracking at present. During this attack the attacker to compose, scan, renew, rework or destroy data which stored in the database. That type of attack grant to attacker to transform the innovative SQL query to a number of injurious codes in the database to get delicate information or to break down the information from the database. In SQL Injection Attack attacker generates injurious code into a traditional consumer information area of web application to access authorization and endless source [13]. An injurious attacker can extract shaded instruction, transform, or even despoil our entire data stored in a back-end database. SQL Injection is deed shelter vulnerableness at the database layer. It is a straightforward scheme in which attackers insert some SQL cipher to the primary cipher in the database to fetch impressible data or to break down the whole story [4].

1.2 Procedure of SQL Injection Attack
The SQL Injection procedure work by undeveloped terminating context chain and attach a new prescript. Because the prescript may have added chain attach to it before it is completed. Generally SQL procedure set-up in three different stages and shown in Fig.1 [14]:

1. An Attacker sends the injurious HTTP input to the web application.
2. Compose the SQL Reports.
3. Refer the SQL Reports to the back end database.

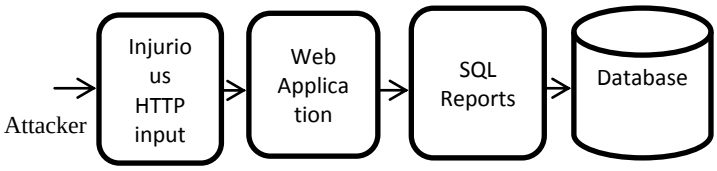

Fig.1. Procedure of SQL Injection Attack

1.3 Classification of SQL Injection Attack
In SQL Injection Attack an attacker workout to shift SQL query by addicting new SQL code word. Thus SQLIA classify into several groups [5][9][17]. These are:-

Tautologies: - That type of attack are commonly adopted to pass around users authorization and to fetch unjustified data by fixing tautology into a modified statement. That occupying SQL Injection Attack is a record that is constantly true.

Example: select * from Student where name='' or 1=1—'and Roll = '--';

Logically Incorrect Queries: - In this type of attack an attacker collects the information about the sample group and architecture of the posterior. If any faulty question is out to a table, some function returns the failure message in form of table name and column..

Example: Collective objects activated on Varchar and reasonless datatypes.

Union Query: - In this attack an attacker establish supplementary statement into the primary SQL statement. This attack can be completed by set up either a UNION query or a comment of the form into accessible framework.

Example: select balance from Account where candid='123' UNION select * from Account';

Stored procedure: - In Stored Procedure technique fall exactly by the schema. It is nothing but a method or it can be ready as instruction. For enable/unofficial consumer the stored procedure results true/false. In SQLIA, using "; SHUTDOWN;--"for end-user and key word.

Example: select marks from Result where name= 'Rahul' and id= '101'; SHUTDOWN;--;

Piggy-Backed Queries: - In this attack an attacker aims to join additional demand to the authentic queue. This query based attack is much vulnerable because an attacker can use it to add implicitly any kind of SQL statement. Supplementary injurious queries are embedded into an authentic inserted query.

Example: select * from Employee where empid= '1111' and pass= '1501'; drop table Employee;--';

Inference: - In this type of attack an interrupter turn over the code of the computer database and function. Inference having two widely known attack approach that are situated on inference: Blind Injection and Timing Attack.

Blind Injection: In Blind Injection attack an attacker aspect to a global sheet implemented by designer. An attacker can still hold up data by asking a range of true/false query using SQL statements [1].

Example: select * from User where id='45' and '1'='0';

Timing Attack: In Timing Attack attacker capture information from a list of database via examine timing delay in the computer database feedbacks. This type of attack uses if-than statement as long as inserting questions.

Example: select * from Table where id='101' or pass= '1'='0';

2. LITERATURE REVIEW

In **2005 McClure, R.A. and Kruger, I.H.** [8] presented a SQL DOM (Domain Object Model) a combination of department the range of group closed through a network architecture. That is protected compile time testing about Dynamic SQL statements. Its aim to display a method to select the SQL DOM automatically by cooperator in real network architecture, determine its function to resolve the problem or compute its achievements. The drawback of this technique is to determine just input cause not actual device equivalent.

In **2006 William G.J Halfond and Alessandro Orso** [16] described a classical approach is known AMENSIA. That is a combination of static analysis and dynamic analysis to identify or stop SQLIA. In static stage AMENSIA wont stable outlines to make layout of the distinct kind of problem an exercise may rightfully create on a place to collect to the information. In dynamic stage AMENSIA take entire inputs before they are transmit to as far as database. The initial drawback of that approach is that achievement is relying on the certainty of its static analysis being query design.

In **2007 X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian and L. Tao** [18] suggested SAFELI a static analysis scheme as discover SQL injection weakness. This scheme target on verify the SQLIA throughput the compilation. The static analysis device obtains two major assets. Primarily, it succeeds a White-Box static analysis or beside, it work with Hybrid constraint solver. White-Box method deals with the unit code or transaction generally along series. And second method achieves an effectual series testing device i.e. capable to contract as well numeral, string and logical operator variables. The limitation of this scheme is not applicable for the dynamic technique.

In **2011 Indrani Balasundaram, E. Ramaraj** [6] described an authentication approach regarding avoid SQL Injection Attack with the help of Hybrid Encryption. This approach accepting the algorithm AES (Advance Encryption Standard) or RSA (Rivest-Shamir- Adleman) against SQL Injection Attacks. An individual private code is distributed for each sender and receiver usage mixture of secret key or general key as RSA encryption. This can't be acceptable for the URL occupying SQL Injection Attack.

In **2011 Allen Pomeroy and Qing Tan** [10] presented a system in order to get exposure within the Internet like SQL Injection Attack through Network Recording. On that system network based approach and device be operate to find the network package consist of get and post order of a web-based information. This system do with web situated IDS (Intrusion Detection System) to generate Network Recording about suspicious function initiative. In occurrence of large number of transaction this system could not be applicable.

In **2012 Veera Venkateswaramma P** [15] proposed CANDID an efficient technique concerning assume Net from SQL Injection Attack. This technique is the progressive Candidate analysis approach to find out the effect of revising the request to design that developed acknowledgement. CANDID approach strongly get the problem framework related each SQL query area that one designed via designer. Its actual deficiency is that creates project above original instruction and applicant's instruction.

In **2013 Debabrata Kar and Suvasini Panigrahi** [3] suggested a lightweight framework to avoid SQL Injection Attack through a Novel Query Transformation device or hashing. That is accepting a Novel Query transformation device which converts a query into its fundamental object in place of the constant layout. This framework is to refer a relevant hashing program to produce individual hash code for every converted query. Lightweight framework is able to freely execute on every terminology and computer database stage as well mini correction. These approaches are acceptable one and only Query Transformation.

In **2015 Kirti Randhe, Vishal Mogal** [7] described a secure mechanism for SQL Injection Attack by the Sanitization Application at Reverse Proxy server. The Sanitization Application exist SQL Injection Preventer that examine a user IP Addresses. This security mechanism partition in to two modules which one is Signature Check and another one is MD5 Hashing. First module confirm the reference via flexible notation on the other side MD5 Hashing generate an individual key for every known person or participants. This Application clarifies the problem of wrong identification or encrypts it earlier than transmitting to the host.

In **2015 Surya Pratap Singh, Avinash Singh, Upendra Nath Tripath, Manish Mishra** [11] designed a Proactive technique to prevention of SQL Injection Attack through the SQL Query Monitor. That is a device which examines each and every query i.e. transmits to the database server via HTTP request. This technique applies the Statistical Database Log for verifying the query produced with HTTP request or allocate in case SQL query are non-injurious.

3. PROPOSED SCHEME

3.1 Proposed Scheme of SQM and Sanitization Application

In this Review paper we proposed scheme (shown in Fig.2) to prevent SQL Injection Attacks to make databases more secure. This scheme is a combination of two methodology which is known as SQM (SQL Query Monitor) and Sanitization in Reverse Proxy Server. Both are having their different way for preventing SQLIA.

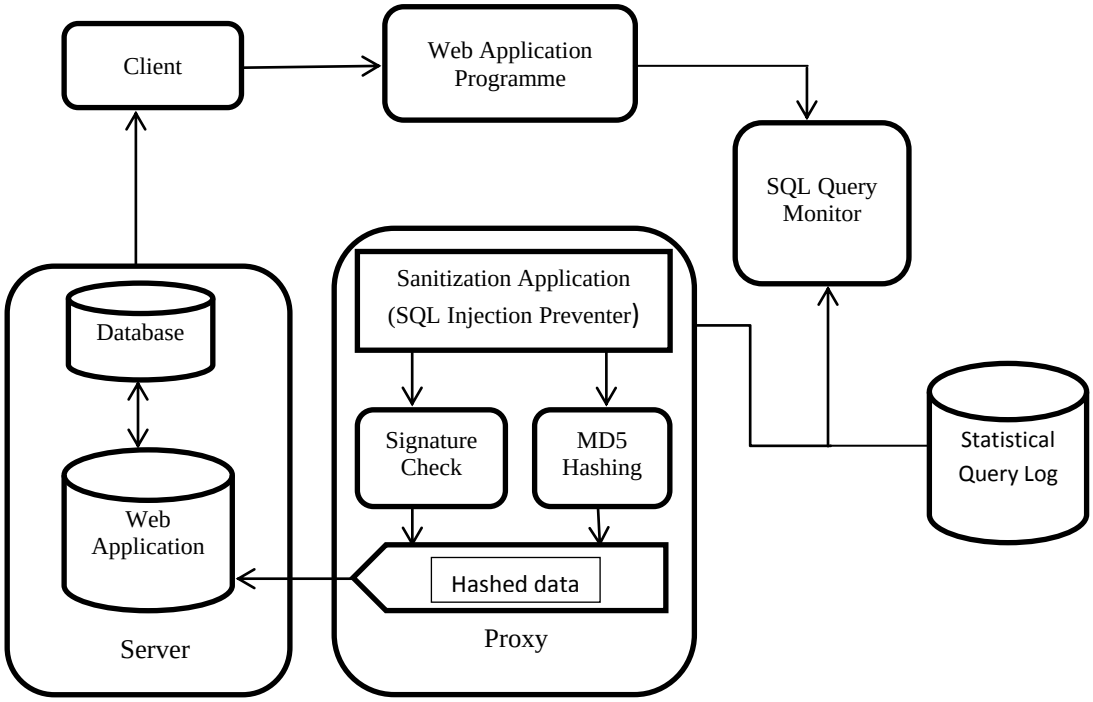

Fig.2. Proposed Scheme to prevent SQLIA

3.2 Modified Scheme for Preventing Web Application against SQL Injection Attack

These following steps show the function of SQL Injection Attack. This algorithm is the observation on the SQLIA, detection and prevention algorithm proposed in that branch.

1. Take the user request/queries.
2. After accepting user request it sends to the SQL Query Monitor through web Application Programme.
3. SQL Query Monitor used to examine all the queries of user.
4. Then the analyses process will be done with the help of Statistical Query log which contains statistical data in the Query Log.
5. If the user request is correlates with the stored data in Query Log it sends to the next module (Sanitization Application) otherwise, it answers as an invalid access.
6. Sanitization Application consists of SQL Injection Preventer to find or avoid SQL Injection Attacks (Tautologies, Logically Incorrect Queries, Union Query, Piggy-Backed Queries, and Inference) and obtain IP Number from the user request.
7. SQL Injection Preventer partitions into two parts i.e. Signature Check and MD5 Hashing.
8. In the first, Signature check proves the user requests have been fully valid or invalid. Beside, MD5 Hashing set a secret code value for every user to justify the request.
9. Only accurate data will be detect in next level i.e. named as Hashed Data which is passed to the web application or database server.
10. At last the database server returns back acknowledgement to the client.

4. COMPARATIVE ANALYSIS OF DIFFERENT TECHNIQUES WITH THE PROPOSED SCHEME

That table discovers the amount regarding technology for stopping or finding of intrusion and additionally obtainable a certain method wants to adjust that program instruction for a particular function Further, new essential feature namely desire as all method is provides and expressed its deficiency in (table 1).

.

Table 1. Comparative Analysis of different techniques with Proposed Scheme

S.No.	Methodology Used	Source Code Modification	Intrusion Stopping	Intrusion Finding	Essential Features	Deficiency
1	SQL DOM(Domain Object Model)[8]	Not needed	Automatic	Automatic	Key Management	Does not determine actual equivalent
2	AMNESIA[16]	Not needed	Report generate	Automatic	IDS system training set	False positives and false negatives
3	SAFELI[18]	Not needed	N/A	Partially Automatic	N/A	Not applicable for the dynamic technique

4	Hybrid Encryption[6]	Needed	Automatic	Automatic	N/A	Can't be acceptable for the URL occupying
5	Network Recording based on IDS (Intrusion Detection System)[10]	Not needed	Automatic	Automatic	N/A	Large number of transaction could not be applicable
6	CANDID[15]	Needed	Automatic	Automatic	Developer training	Works at original instruction and applicant's instruction
7	Query Transformation and Hashing[3]	Not needed	Automatic	Automatic	N/A	Based on Query Transformation
8	SQM (SQL Query Monitor)[7]	Needed	Automatic	Automatic	Developer training	Query based
9	Sanitization Application[11]	Needed	Automatic	Automatic	Proxy Filter	Proxy needed
Proposed Scheme	**SQM and Sanitization Application**	**Needed**	**Automatic**	**Automatic**	**Key Management and Developer training**	**Taking more time**

5. CONCLUSION

The SQL Injection Attack is the largest accessible security risk in the network based computer database in today because all attacker or application programmer attempt to crack the information safety measure accepting similar form of violation. In such a manner this proposed scheme regarding security against SQLIA, is too sensitive. Several clarifications are disposed through distinct experimenters although no one resolution is completely capable in order to anticipate the information against the particular initiative.

Therefore, this paper clarify the features an injection techniques of SQLIA, to assure the database apart from the particular violation this one paper introduce the purpose of SQM (SQL Query Monitor) and Sanitization Application acting as the accepting component via whatever individually may simply approved the ready-made option one and only to failure of the web application accepting SQL Injection Attack

that want to keep enclosed thus there is a lot of field of reference now the present in this field of experimentation.

6. FUTURE SCOPE

This scheme is able to secure information against SQL injection attacks, but consumes more time for analyzing or filtering process as compared to single technique. So, in future will trying to develop this framework via composing it well active, protect and capable for any kind of attacks and will more focus on reducing time complexity of proposed algorithm. So that this modified scheme will be ready to stop SQL Injection Attack effectively and will secure web applications efficiently. This scheme can be applied to real time systems and applications to increase more security and vulnerability to real life systems.

7. REFERENCES

[1] Amirtahmasebi, K, Jalalinia, S.R., and Khadem, S., A Survey of SQL Injection defense mechanism, International Conference for Internet Technology and Secured Transaction (ICITST 2009), 9-12 Nov. (2009), pp. 1-8.

[2] Atefeh Tajpour, Suhaimi Ibrahim, Maslin Masrom," SQL Injection Detection and Prevention Techniques", International Journal of Advancements in Computing August 2011.

[3] Debabrata Kar, Suvasini Panigrahi, Prevention of SQL Injection Attack Using Query Transformation and Hashing, IEEE International Advance Computing Conference (IACC), 2013.

[4] Ettore Merlo et al. "Insider and outsider threat sensitive SQL injection vulnerability analysis in PHP" IEEE 2006.

[5] http://www.softwaretestinghelp.com/application for SQL injection attacks.

[6] Indrani Balasundaram, E.Ramaraj "An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption (PSQLIAHBE), (ISSN 1450-216X Vol.53 No.3 (2011), pp.359-368).

[7] Kirti Randhe, Vishal Mogal "Security Engine for prevention of SQL Injection and CSS Attacks using Data Sanitization Technique", Pune , Vol. 3, Issue 6, June 2015.

[8] McClure, R.A. and Kruger, I.H., SQL DOM: compile time checking of dynamic SQL statements. 27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, pp. 88- 96.

[9] Prasant Singh Yadav, Pankaj Yadav, K.P.Yadav "A Modern Mechanism to Avoid SQL Injection Attacks in Web Applications", IJRREST: International Journal of Research Review in Engineering Science and Technology, Volume-1 Issue-1, June 2012.

[10] Pomeroy, A Qing Tan Sch. of Comput. & Inf. Syst., Athabasca Univ., Athabasca, AB, Canada " Effective SQL Injection Attack Reconstruction Using Network Recording" in Computer and Information Technology (CIT), 2011 IEEE 11th International onference Issue Date: Aug. 31 2011-Sept. 2 2011 On page(s): 552 – 556.

[11] Surya Pratap Singh, Avinash Singh, Upendra Nath Tripath, Manish Mishra "Proactive Mechanism of Protection against SQL Injection Attack", Gorakhpur, Vol.3, Issue 5, 2015.

[12] The Open Web Application Security Project, "OWASP TOP 10 Project", http://www.owasp.org.

[13] Top 10 2013-A1-Injection, available at: http://www.owasp.org/index.php/Top_10_2013- A1-Injection, last accessed 11 June, 2013.

[14] V. Nithya, R.Regan, J.vijayaraghavan,"A Survey on SQL Injection attacks, their Detection and Prevention Techniques""International Journal of Engineering and Computer Science April, 2013.

[15] Veera Venkateswaramma P, "An Effective Approach for Protecting Web from SQL Injection Attacks", International Journal of Scientific & Engineering Research, Volume 3, 2012.

[16] William G.J. Halfond and Alessandro Orso "Preventing SQL Injection Attacks Using AMNESIA," ICSE'06, Shanghai, China, 2006.

[17] W. G.J. Halfond, J. Viegas, and A. Orso, "A classification of SQL injection attacks and countermeasures", In Proceedings of the international Symposium on secure Software Engineering (ISSSE), 2006.

[18] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao., " A Static Analysis Framework for Detecting SQL Injection Vulnerabilities", COMPSAC 2007, pp.87-96, 24-27 July 2007.