ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

Hashing Technique - SQL Injection Attack Detection & Prevention

Parveen Sadotra

Teaching Assistant & (CEH), GDC, R.S. Pura, Jammu (J&K), India

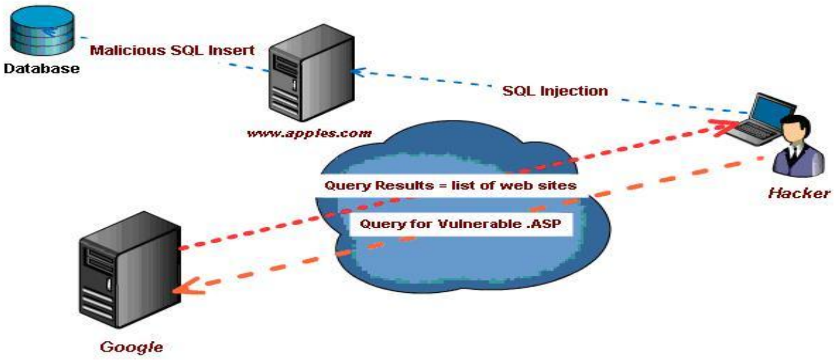
ABSTRACT: In today's digital world, Web applications are being used in numerous ways in recent years to provide online services such as banking, shopping, social networking, etc. These applications operate with sensitive user information and hence there is greater need for assuring their confidentiality, integrity, and availability. Extensive use of websites and web applications has attracted hackers to attack on it using various tricks and techniques. SQL injection is one of most used attack we face nowadays. In SQL Injection Attack (SQLIA) the attacker can trick the server to obtain illegal authorization and asses the database using SQL queries. This is because the developers of the applications do not know fully about the attacks by SQL injection and its causes. This research paper focused on how to detect and prevent SQL injection attacks on websites and web applications using hashing technique

KEYWORDS*:* Hashing, SQL Injection, SQL queries, Web Applications

I. INTRODUCTION

In today's world the digital employment internet usage is rising and lots of advancement in technology has eased our daily lives. The World Wide Web (www) has evolved from a system that delivers static pages to a platform that supports distributed applications, referred to as web applications, and it has become one amongst the most significant rife technologies for data and service delivery. Multiple services are available from a single click through various web applications and websites and there is no need to stand in long queues at the banks, ticket counters to buy tickets or market to buy for the modern trends. As web applications are increasingly used to deliver essential services, they have become an important target for hackers. Many web applications interact with back end database systems, which may store sensitive or confidential information related to health, defence, finance etc. As a result these attacks against them are increasing rapidly. Of those attacks, a serious role is controlled by SQL injection attacks (SQLIA). SQL injection attack is one amongst the intense dangers to web application accustomed gain unauthorized access to database or to retrieve the confidential data present on the database.

A SQL injection attack is done by insertion or "injection" of a SQL query with the input data from the user to the application. A successful SQL injection can read sensitive data from the database, alter database data and perform query such as Insert/Update/Delete and perform administration operations on the database such as shutdown the Database, recover the data present in a file on the Database and perform some commands on the operating system.



II. REVIEW OF LITERATURE

- International Journal of Advanced Research in Computer Science and Software Engineering
- A Journal : A Novel Approach for SQL Injection Prevention Using Hashing & Encryption
- Journal : SQL Injection Attack on Data and Prevention through Hashing
- Ethical hacking Book Hacking Made Easy
- Journal : Securing Web Application against SQL Injection Attack: a Review
- Journal : Detection and Prevention of SQL Injection Attack Using Hashing Technique
- Paper : Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype
- Methodology and Analysis for Various SQL Injection Techniques

III. NEED / IMPORTANCE OF THE STUDY

What makes the threat of SQL injection attacks so risky is the ease by which they can be started and how many web sites are vulnerable to them. Attackers often use large botnets to systematically find out vulnerable web sites to attack with little work being done on their part. Along with this with the fact that the number of sites vulnerable to this type of attack grows each year, it is clear to see why it remains at the top of the most serious vulnerabilities.

Risks Associated with SQL Injection

Even with the ease that an automated SQL injection attack can be launched, if the attackers stand to gain nothing this threat would soon vanish. Unfortunately those who successfully compromise vulnerable web sites can find that this vulnerability can be quite gainful as they give the attacker access to the database so information can be changed, sold or even can be deleted. More advanced techniques can also be used to give the attacker unrestricted access to the system through a backdoor. SQL injection can also be used in tandem with other types of exploits, such as CXX (cross-site scripting), to manipulate how data is showed to a web site's visitors.

Not preventing SQL Injection attacks leaves your website/web applications at risk ultimately to your business at great risk of:

- ✓ Changes to or removal of highly sensitive business information.
- ✓ Steal customer important information such as social security numbers, Pins and passwords, addresses, and credit card numbers.
- ✓ Financial losses to business
- ✓ Damage to brands
- ✓ Theft of intellectual property (IP)
- ✓ Legal liability and fines

Above are the issues that compel us and make it important to study SQL injection and its prevention by Hashing techniques.

IV. STATEMENT OF THE PROBLEM

Although recently there has been a great deal of attention to the problem of SQL injection vulnerabilities, many proposed solutions fail to address the full scope of the problem. There are many types of SQLIAs and unlimited variations on these basic types. Researchers and practitioners do not know about the different techniques that can be used to perform SQLIAs. Therefore, most of the solutions proposed detect or prevent only a subset of the possible SQLIAs.

Developers are faced with multiple challenges when attempting to effectively secure online applications (especially web sites). Properly addressing these issues depends on the state of the application at the time, the developer's priorities and ultimately on the approach that will be chosen. Here are some of the major issues that arise when implementing SQL Injection Attack protection:

DOI: 10.15680/ijircce.2015.0305051

We can see large numbers of web servers are growing daily and so the numbers of installed web applications on these servers are also increasing. Many web sites use open source web applications to provide certain services that are part of the web site, such as a bulletin board e.g. phpVB, a blog e.g., Word Press or they use a content management system e.g. Mambo, Typo3, drupal etc. that can be used to operate the complete web site. Web applications are not only used by private web site providers but also by companies and governmental institutions. Most often a database is used as the primary resource to get back the information that is requested by the user. The information present in the database has been stored by somebody responsible of tending the web site, or the information is created by an internal business process of the company e.g. the currently available articles in an on-line shop. Other possible source of content in a websites may be a remote web service of a news agency that provides current news. The numbers of security problems in software have increased in last some years. Some of the security problems affect web applications that provide dynamic web pages to their customers. Attackers that use these security problems either prying on data contained in the web application e.g., credit card numbers of customers or they use the web application as an attack vector on the visiting customer. Both types of attacks depend on user input that is not validated by the web application. To extract personal information from the web application, “SQL injection” can be used [9, 10]. In this kind of attack, information that is fed by the user is included in database queries that are used to extract content for the web page. Because the user input is not checked for malicious content, arbitrary SQL queries can be executed. These queries can then be used to circumvent safety procedures incorporated in the web application e.g., bypass logins, retrieve personal data of customers e.g., credit card information, social security numbers or execute system commands on the targeted web servers in order to install malicious software on the server.

V. OBJECTIVES

As we know, in 21st century, all the important day to day tasks are being done on internet, collecting any new information we search it from Google, then for anything which we want to buy we can do online shopping from e-commerce websites, for uploading assignment by faculty and students download it using internet only, online banking and many more other work done on internet using various websites. SQL Injection is a web attack system which is being used by hackers to gain and misuse data of that website. SQL Injection is one among the many web attacks used these days and is being applied on several websites which are not secured properly. In this type of web attack the hacker takes advantage of incorrect and incomplete coding of a website which allows him to inject SQL Injection codes into admin login site and the he gains the access to the data present within that website’s database. Our main objective here is to study and analyze hashing technique to prevent these SQL injection attacks on our websites/ web applications.

The main objective of this work has to aware programmer and readers from SQL injection techniques .We have explained security method to secure development of web application .Today, many techniques have came to prevent from SQL injection. But, these days it remains a big issue in development area. Because Black hat hackers are always busy to find out new techniques and applying them in new way. Our work is advantageous for researchers and developers of websites. New techniques will be needed to handle new SQL injection method. We need to survey continuously to this attack.

VI. HYPOTHESIS

Before a web site can be compromised, an attacker needs to find applications that are vulnerable to SQL injection using queries to learn the SQL application methods and its response mechanisms.
The attacker has two ways to identify SQL injection vulnerabilities:

1. **Error messages:** the attacker constructs the correct SQL syntax based on errors messages propagated from the SQL server via the front-end web application. Using the errors received, the hacker learns the internal SQL database structure and how to attack by injecting SQL queries via the Web application parameters.

2. **Blindfolded Injection:** this technique is used by hackers in situations when there are no error messages or response content is returned from the database. In these cases the attacker has limited ability to learn the backend SQL queries in order to balance the SQL injection query. In the lack of database content output

within the Web application, the attacker also wrestles with finding a new way of retrieving the data. Identifying the database when the attacker knows how each database is reacting attacker can recognize the database type and the server that is running it. There are several techniques the attacker uses to identify database objects in a SQL statement.

a) Using a concatenation string:
select f1+f2
from t1

b) Using a semicolon or cash sign ($)

1) Compromising the SQL server

Once the attacker has all information he can build the exploit code.
Some methods used to execute SQL Injection attacks are:

- ✓ Terminating queries using quotes, double-quotes, SQL comments
- ✓ Using stored procedures
- ✓ Database manipulation commands such as TRUNCATE, DROP
- ✓ Using CASE WHEN, EXEC to run nested queries
- ✓ Utilizing SQL injection to create Buffer Overflow attacks within the database server
- ✓ Delivering SQL queries via XML and Web Services
- ✓ Blindfolded SQL Injection techniques:
  - ❖ Blindfolded injection techniques using Boolean queries and WAITFOR DELAY
  - ❖ Comparison queries using commands such as BETWEEN, LIKE, ISNULL

- ✓ IDS signature evasive SQL Injection techniques:
  - ❖ Using CONVERT & CAST commands to mask the attack payload Using Null bytes to break the signature pattern
  - ❖ Using HEX encoding mixtures
  - ❖ Using SQL CHAR() to represent ASCII values as numbers

For example, the attacker decides to go with a basic attack using: 1 = 1--
What happens when this is entered into an input box is that the server recognizes 1 = 1 as a true statement. Since -- is used for commenting, everything after that is ignored making it possible for the attacker to gain access to the database.

VII. RESEARCH METHODOLOGY

For our this paper, we contacted various cyber security professional for their experience, studied various journals and research papers, gone through SQL injection attacks that took place in cyber world and analysed following things about the this threat to websites and web applications: -

Attack Intent (Identifying injectable parameters)
The attacker wants to probe a Web application to discover which parameters and user input fields are vulnerable to SQL injection Attacks.

Performing Database finger printing:
The attacker wants to discover the type and version of database that a Web application is using. Certain types of databases respond differently to different queries and attacks, and this information can be used to “fingerprint” the database. Knowing the type and version of the database used by a Web application allows an attacker to craft database specific attacks.

Determining database schema:
To correctly extract data from a database, the attacker often needs to know database schema information, such as table names, column names and column data types. Attacks with this intent are created to collect or infer this kind of information. These types of attacks use techniques that will extract data values from the database Based on all the study about various types of attacks and preventive measures we studied hashing technique to prevent SQL injection threat on our web applications.

VIII. EFFECTS OF SQL INJECTION ATTACK

As the SQL injections are related with the database and in today's scenario where the database is one of the primary assets of any organization. Therefore, with these SQL injections, cyber-criminals can take complete remote control of the database, and become able to manipulate the database to do anything they wish, including:

- ✓ Insert a command to get access to all account details in a system, including user names and retrieve VNC passwords from registry
- ✓ Upload files
- ✓ Through reverse lookup, gather IP addresses and attack those computers with an injection attack
- ✓ Corrupting, deleting or changing files and interact with the OS, reading and writing files
- ✓ Online shoplifting e.g. changing the price of a product or service, so that the cost is negligible or free
- ✓ Insert a bogus name and credit card in to a system to scam it at a later date
- ✓ Delete the database and all its contents
- ✓ Shut down a database

IX. RESULTS AND DISCUSSION

Following are the results we got from our study on SQL injections attacks. These are self explanatory and no need to discuss on it.

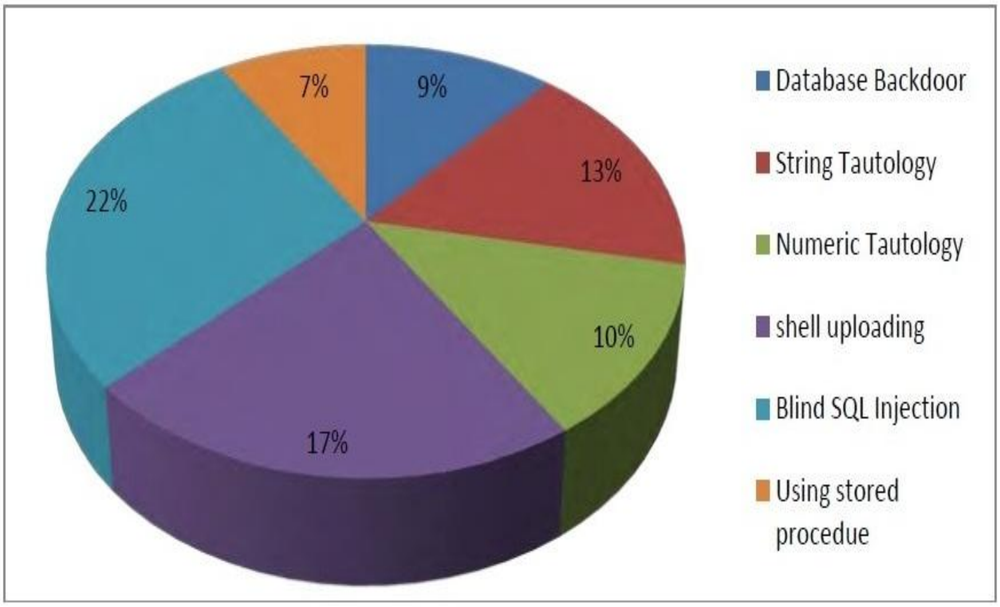


Figure: Percentage of SQL injection methodologies used to perform attack

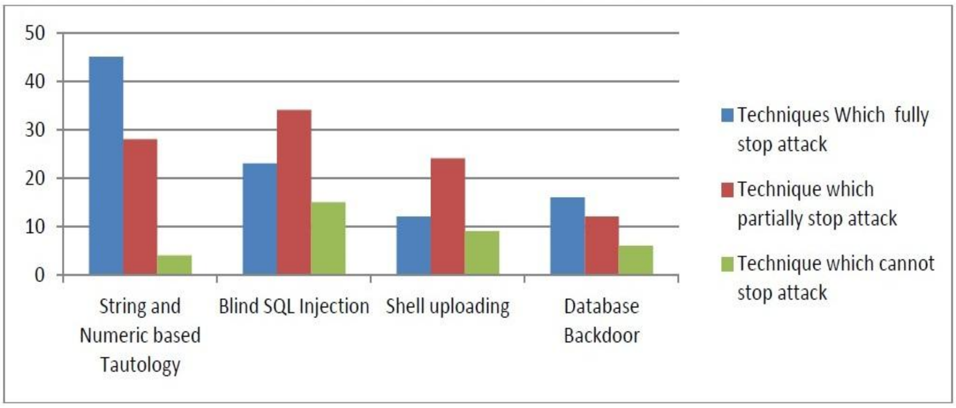


Figure: Percentage of SQL injection Prevention and detection techniques

X. FINDINGS

Database applications have become a core component in control systems and their associated record keeping utilities. Traditional security models attempt to secure systems by isolating core software components and concentrating security efforts against threats specific to those computers or software components.

Basically there are following ways to attack on our web servers: -

Tautologies: The main objective of tautology based attack is to inject code in conditional statements so they're indefinitely taken as correct. Using tautologies, the attacker needs to either bypass authentication or put inject able parameters or extract information from the database. Whenever a conditional statement is injected with code so the result is true then its analysis and result is dependent on the method that the query is evaluated within the application. The attacker mainly stresses on the where clause to inject the code.

Illegal/Logically Incorrect Queries: The aim of the Illegal/Logically Incorrect Queries based SQL Attacks is to gather the information regarding the back end database of the web Application. When a query is wrong or illegitimate, an error message is returned from the database together with helpful debugging data. This error messages help attacker to find vulnerable parameters within the application and consequently database of the application. Attacker injects junk input or SQL tokens in queries that may lead to syntax error, type mismatches, or logical errors by purpose.

Union Query: In this technique of attack, attackers join injected query to the original query by the word UNION and then can receive data concerning other tables from the application. The result of this attack is that the database provides a dataset that is the combination of the results of the initial query with the results of the injected query

Piggy-backed Queries: In it attacker has to execute remote commands or add or change information. In this type of attack, the attacker doesn't build changes in the original queries however inject further queries. this can be totally different from alternative types as a result of the attackers don't seem to be making an attempt to inject the original planned query; instead they're making an attempt to incorporate a new and distinct queries that “piggy-back” on the initial query. So, the information gets multiple SQL queries. The primary is that the proposed query by the application that is performed as normal; the later ones are the injected queries that are performed in addition to the primary. If successful, the attackers will nearly insert any style of SQL command and have them execute with the original query. Vulnerability of this type of attack depends on the sort of database.

Stored Procedure: This type of attack tries to execute stored procedures present in the database with malicious inputs. As stored procedure may well be coded by programmer, so, this part is as inject able as web application forms. Depending upon specific stored procedure on the database there are alternative ways to attack. As an SQL Injection

Attack, intruder input "; SHUTDOWN; --" for username or password. Then the stored procedure generates the following query:
Inference: By this type of attack, attackers modify the behaviour of information or application. There are two standard attacking techniques that are based on inference: **blind injection and timing attacks.**

i) **Blind SQL Injection**: In this sort of attack, helpful information for exploiting the backend database is collected by inferring from the replies of the page after questioning the server some true or false questions. But when attackers attempt to use an application, instead of obtaining a helpful error message, they get a generic page mere by the developer instead. This makes exploiting a possible SQL Injection attack tougher however not possible. Associate attacker will still get access to sensitive information by asking a series of True and False queries through SQL statements.

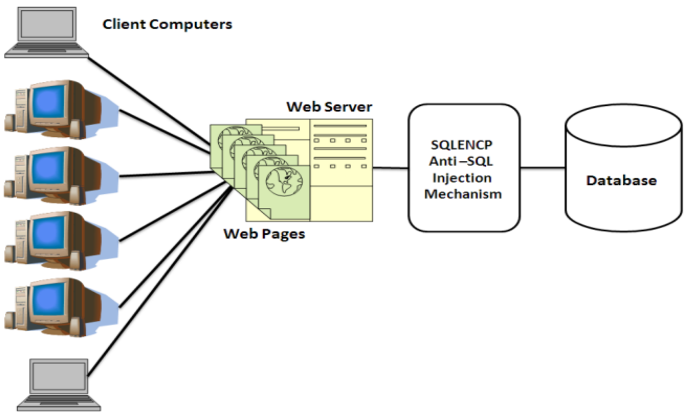
ii) **Timing Attacks:** A timing attack lets an attacker pile up information from a database by observing timing attack delays within the database's responses. This system by victimization if-then statement cause the SQL engine to execute an extended running query or a time delay statement reckoning on the logic injected.

Our security experts follow many techniques to deal with these attacks by hackers, however here we are going to analyze how we can prevent SQL attack on our websites and web applications by use of Hashing Techniques.

XI. RECOMMENDATIONS / SUGGESTIONS

After studying the various SQL injection prevention techniques, we proposed a technique in which we implement a mechanism, that detect & prevent the SQL injections by incorporating the techniques of "HASHING" & "ENCRYPTION".

The concept behind our implemented system is simple: instead of relying on user's permissions, we implement complicated defensive coding techniques with which, we can detect & prevent the SQL injections and provide security to the web application.



Our proposed method simply works on the "HASHING" methods for the secure login technique. In which, we calculate the hash value of the username and passwords for any user and store it in database table along with the simple username & passwords.

DOI: 10.15680/ijircce.2015.0305051

ID	Usernames	Passwords	Remarks
1	admin	admin	
2	redhat	abc@123	
3	root	root123	
4	rakesh	rakesh987	

Table: User table without security guidelines contains only username & passwords

ID	Username	Password	Hash_usernames	Hash_passwords	Hash_EX-OR
1	admin	admin	F59295350096DBA033BE A802EC1A573FBE937EB0	19AA6EE730DFD50936A5 45A683A0716380D9Y8E5	EC73D2D20E29051BEDA 413A265C349655
2	redhat	abc@123	7FE94AC4AC6B93369B5E 8C290ECB15E443771 657	947F02F46B4CA59418870 C21CC4A9391DDC52F1A	2964830C7278B53108C28 1899EB2394D
3	root	root123	F59295650096DBA033BEA 802AG1A573DBE937EB0	45BA6ER780DFD50936A5 45A683A0716390G9E8E9	OJ73D2G67E29051BEDA 413A265C3H9968
4	rakesh	rakesh987	7FE94AG8AC6B93369B5E 8C290EHG15E446871657	787F06J86B4CA5949670 C21kj4A9391DDC52F1G	J584830C72U8B53108C28 1899EB23HG8

Table: User table with security guidelines also contains the hash values

HASH FUNCTION ALGORITHM

In the proposed approach there is a need for one extra column in database, which contains the EX-OR of the Hash values of username and password at the time, when a user account is created for the first time and stores it in the User table. Whenever user wants to login to database his/her identity is checked using user name and password and its hash values. These hash values are calculated at runtime using store procedure when user wants to login into the database. During the authentication of user, the SQL query with hash parameters is used. Hence, if a user tries the injection to the query and our proposed methodology is working with SQL query, it will automatically find out the injections as the potentially harmful content and rejects the values.

Standard Query	SELECT username, password FROM tbl_user WHERE username = @usrname AND password = @pwd
Malicious Code	SELECT username, password FROM tbl_user WHERE username ‘ ’ OR 1=1; ‘/* AND password = ‘*/’
SQL Injection	SELECT username, password FROM tbl_user WHERE hash_exor = Exor(hashval('$user_nm'), hashval('$pass'))
Output	Not Possible, As the direct values not passing to SQL Query.

DOI: 10.15680/ijircce.2015.0305051

Table: Query Testing

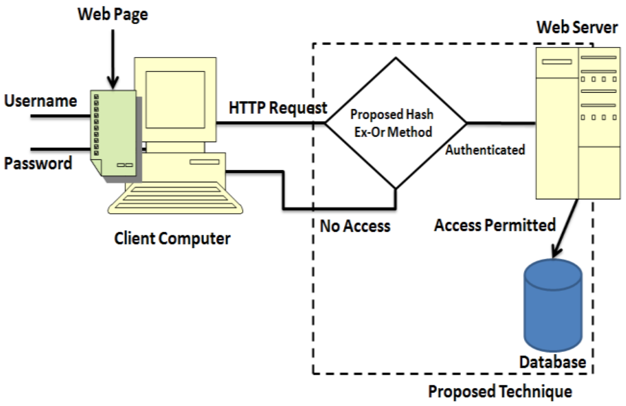


Figure: Proposed Hashing Technique

Therefore, it cannot bypass the authentication process. The advantage of the proposed technique is that the hackers do not know about the hash values of user name and password. So, it is impossible for the hacker to bypass the authentication process through the general SQL injection techniques. The SQL injection attacks can only be done on codes which are entered through user entry form but the hash values are calculated at run time at backend before creating SELECT query to the underlying database therefore the hacker cannot calculate the hash values as it dynamic at Runtime.

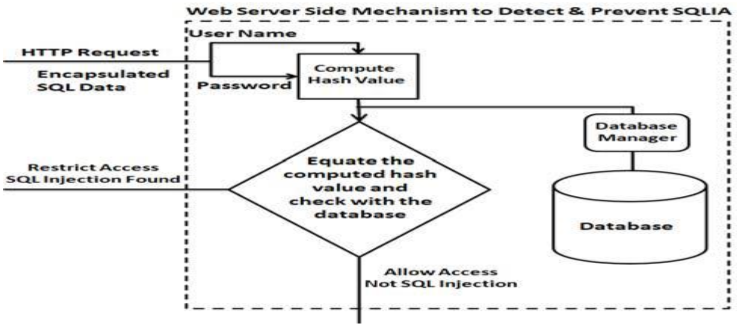

Web Server Side Mechanism to Detect & Prevent SQLIA
User Name
HTTP Request
Encapsulated SQL Data
Password
Compute Hash Value
Database Manager
Restrict Access SQL Injection Found
Equate the computed hash value and check with the database
Database
Allow Access Not SQL Injection

Figure: Proposed Hash Scheme for Detecting SQLIA & Prevent Them (SQLENCP)

Consider a case, where a user is authenticated by the secure login mechanism and login the system. Now, if this authenticated user make any intrusion into the system. How can we defend it?

Hence to prevent after-login attacks we have taken the help of data encryption. As we saw in the previous section that it is possible to collect the highly confidential information by using union operator we find out an alternative way to store all these confidential information's.

In our database, instead of directly storing all confidential information's, we store them in encrypted format with a secure and confidential encryption-key. Now even if the dispatcher user can able to see the atm_pin by using union operation, he is unable to decrypt it without knowing the exact encryption method and encryption- key. So he is unable to do any damage with that encrypted atm_pin.

XII. CONCLUSIONS

During the study of several researches based on the SQL injection Prevention & Attacks, we found that in certain cases these approaches are not effective. Hence, these approaches become usefulness and cannot detect the injections to prevent them. In addition, the attackers can access the database directly in an illegal way. Therefore, we have proposed a new approach that is completely based on the hash method of using the SQL queries in the web-based environment, which is much secure and provide the prevention from the attackers SQL. But, our proposed strategy requires the alterations in the design of existing schema database and a new guideline for the database user before writing any new database. Through these guidelines, we found the effective outcomes in SQL injections Preventions. After that we compared these techniques in terms of their ability to stop SQLIA. Still, we need to improve our approach so that, it can prevent the web application & database from all kind of SQL Injections. We also plan to apply SQL Prevent to dynamic discovery of SQLIA vulnerabilities.

XIII. SCOPE FOR FURTHER RESEARCH

It is clear from above category that SQL injection attacks is one of the biggest classes of security problems faced today by web developers and IT security professionals. In this technique developers require to manually or automated specify the interface to an application when applied to modern complicated web application. The main important SQL injection related issues have been reviewed in this paper. We have planned a new technique which is based on hash function and which is easy and extremely safe from attackers. This paper shows an Authentication method for preventing SQL injection attack and describes its limitation and its application also and the future estimated work describe the efficiency of the system.

In this work, we have focused on the specific area of SQL injection. We believe that this area is in need of further investigation, mainly because there are so many reasons such as SQL injection attacks are most likely to evolve and new vulnerabilities will be found, together with new counter measures to deal with them. Since many hacking sites are available on the web, and since attack methods are well described and distributed between hackers, we believe that information about new attack methods should continuously be surveyed and new counter measures should be developed.

REFERENCES

1. <http://searchsqlserver.techtarget.com/definition/hashing>

2. Parveen Sadotra (CEH), Dr. Anup Girdhar, 2013, *"Information Technology (Amended) Act 2008: A Critical Analysis",* 'Cyber Times International Journal of Technology & Management', Vol 6. Issue 2.Pg.108-112.

3. <https://www.scss.tcd.ie/Owen.Conlan/4d2/4D2-5&6_Hashing_Techniques_v1.02.pdf>

4. <http://searchsqlserver.techtarget.com/definition/hashing>

5. Parveen Sadotra (CEH), Dr. Anup Girdhar, 2013, *"Ontology Based Intrusion Detection systems",* 'Cyber Times International Journal of Technology & Management', Vol 6. Issue 2. Pg.141-147.

6. <http://enggedu.com/tamilnadu/university_questions/question_answer/be_mj_2007/5t h_sem/cse/CS1301/part_b/14_b_2.html>

7. <http://www.eecs.harvard.edu/~michaelm/postscripts/dimacs-chapter-08.pdf>

8. Parveen Sadotra (CEH), Dr. Anup Girdhar, 2015, *"Role Of Cyber security In Private sector Domains",* 'Cyber Times International Journal of Technology & Management', Vol 8. Issue 1. Pg.7-11.

9. <http://www.ijcsit.com/docs/Volume%205/vol5issue01/ijcsit2014050112.pdf>

10. <http://www.ijarcsse.com/docs/papers/Volume_4/4_April2014/V4I4-0205.pdf>

11. Parveen Sadotra (CEH), Dr. Anup Girdhar, 2013, *"Penetration Testing/Cyber security Assessment – XYZ Company",* 'Cyber Times International Journal of Technology & Management', Vol 6. Issue 1. Pg.340-350

12. <http://www.codeproject.com/Articles/9378/SQL-Injection-Attacks-and-Some-Tips-on-How-to-Prev>

13. <http://www.applicure.com/solutions/prevent-sql-injection-attacks>

14. Parveen Sadotra (CEH), Dr. Anup Girdhar, 2014, *"Identity theft - the technical and Legal perspective",* 'Cyber Times International Journal of Technology & Management', Vol 7. Issue 2. Pg.555-564.

15. <http://www.applicure.com/solutions/prevent-sql-injection-attacks>

16. <http://airccse.org/journal/ijdps/papers/3612ijdps01.pdf>

17. <http://www.academia.edu/3713854/SQL_INJECTION_ATTACKS_AND_PREVENTION_TECHNIQUES>