ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

Authentication Scheme to Protect Web Applications against SQL Injection Attack Using Hash Functions

Pooja Saini[1], Sarita[2]

M.Tech Student, Dept of Computer Science & Engg, Doon Valley Institute of Engg. & Technology, Karnal, India[1]

Assistant Professor, Dept of Computer Science & Engg, Doon Valley Institute of Engg. & Technology, Karnal, India[2]

Abstract: Online business expects web applications to be secure, efficient and reliableto the users against SQL Injection Attacks. The SQL Injection Attack exploits a security vulnerability occurring in the backend database layer of a web application which is the results of poor input validation in code and website administration. This allows attackers to obtain unauthorized access to the user sensitive information or change the intended web application through SQL queries. In the past, researchers have developed several methods/techniques to overcome the SQL injection problems. However, these approaches either have limitations or fail to cover full scope of the problem. In this paper, a hash function based authentication scheme including data validation is proposed to protect web applications against the SQL Injection attacks. A review of the different types of SQL injection attacks and cases of how attacks of that type could be performed is presented. The proposed techniqueis found to be quite useful and secure for protecting web applications against SQL Injection attacks.

Keywords: SQL Injection attack, Hash function algorithm, Database Security, validation.

I. INTRODUCTION

Information and Communication technologies have witnessed a rapid growth in businesses, enterprises, governments and it found that web applications can give effective, efficient and reliable solutions to conducting e-commerce. Web applications in various sectors like e-commerce, online banking, enterprise and supply chain management, e-governance, etc. conclude that at least 92% of these web applications are vulnerable to some form of attacks [1].SQLIV (Structured Query Language Injection Vulnerabilities)are one of the most serious threats to web applications [2]. Web applications vulnerable to SQL injection may allow an attacker to gain complete access to their underlying databases of information systems. Because these databases often contain sensitive user information, the resulting security violations can include identity theft, loss of confidential information, and fraud.In some cases, attackers can even use an SQL injection vulnerability to take control of and even corrupt the system that hosts the web applications. According to Open Web Application Security Project (OWASP), SQL injection attacks (SQLIA) stands firstin the top 10 threats for web application security in 2013 [3]. In SQL injection attack, attacker provides SQL code rather than the legitimate input in the input fields of the web application in order to vary the meaning of the original SQL query issued by the backenddatabase. Once the attacker gains access to the database, it can alter any sensitive information. To implement security guidelines inside or outside of the database, the access to the sensitive databases needs to be monitored. Detection and prevention of SQL injection attacks are a topic of active research in the academia and industry.Several automatic tools and security system were Implemented to achieve the purpose, but none of them were complete or accurate enough to guarantee an absolute level of security of web applications. The aim of the paper is to review the different types of SQL injection attacks and to propose a hash function algorithm based authentication scheme to secure web applications against SQL injection attacks.

II. SQL INJECTION BACKGROUND

Injecting a web application is the synonym of having access to the data stored in the database. The data sometimes could be confidential and of high value like the financial secret of banks or transactions or secret information of some kinds of information system, etc. An unauthorized access to this data by a crafted user can threat their confidentiality, integrity, and authority. As a result, the system could bear heavy loss in giving proper services to its users or it may face complete destruction. SQL injection is most commonly used by hackers to steal data from information systems of organizations. If it happens against the information systems of a hospital, the private information of the patients may be leaked out which could threaten their reputation or may be a case of defamation [4]. These attacks are designed not only to breach the database security and steal the entire content of the database, but also, to make arbitrary changes to both the database schema and contents.SQL is a special-purpose programming language used to communicate with databases of information system. SQL can insert, retrieve, update and delete data. Of course, any system can be misused, and the most common form of misuse of SQL is an SQL injection [5]. SQL injection is a technique using

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

the above operations against the database in a way that it no more fulfils the desired results but give the attacker an opportunity to run his own SQL command against the database that too using the front end of websites [6]. The SQL injection technique tricks the target into passing malicious SQL code to a database by embedding portions of code with user input [5]. An SQL injection is a kind of injection vulnerability in which the attacker tries to inject arbitrary pieces of malicious data into the input fields of an web application, which, when processed by the application, causes that data to be executed as a piece of code by the backend SQL server, thereby giving undesired results which the developer of the web application did not anticipate, leveraging almost a complete compromise of system in most cases. Two important characteristics of SQLIAs used for describing attacks are injection mechanism and attack intent.

A. Injection Mechanisms

Different injection mechanisms are used to introduce malicious SQL statements into vulnerable web applications. We explain the four most common mechanisms as follows.

1) Injection through user input: In this injection mechanism, the SQL commands are injected through the user inputs, which in most attack scenarios, is a web form. The input from these web forms up on submission is sent to the backend application through HTTP GET or POST requests.

2) Injection through cookies: A web application stores the client's related state information in the client's machine, which is used to restore the client's session in future sessions. Since, the most web applications are using these cookies to build SQL queries, the client who has the access to the cookies saved in his machine, could tamper the contents to embed his malicious code and execute an attack.

3)Injection through server variables: Server variables are used in trend analysis and usage statistics in most web applications, as they contain detailed header information such as network headers and HTTP headers. These variables can be used in executing SQL Injection attacks when these server variables are logged to the database without proper sanitization to remove the malicious contents. So, attackers can forge the headers to include the malicious code which will trigger the attack, when the server variable is logged in the database using the SQL queries.

4) Second-order injection: Second-order injections are different from the other injection mechanisms in a way that the second-order injections will craft the injection code in such a way that, it will not trigger initially when it reaches the database, but the attack will be triggered at a later stage, when the crafted data is used.Second-order injections can be especially difficult to detect and prevent because the point of injection is different from the point where the attack actually manifests itself. A developer may properly escape, type-check, and filter input that comes from the user and assume it is safe. Later on, when that data is used in a different context, or to build a different type of query, the previously sanitized input may result in an injection attack.

B. Attack Intentions

Different types of SQL Injection attacks are performed for different intended purposes. Ten possible intents are as follows.

1)Identifying injectable parameters:In this case, the goal of the attacker is to identify the parameters and input fields that are vulnerable to SQL Injection attacks.

2) Performing database finger-printing: In this case, the attacker wants to identify the database type and version, which will help the attacker to craft attacks specific to the database type.

3) Determining database schema:The attackers need to figure out the exact schema information of the database, in order to extract data from a database. These kinds of attacks are aimed at collecting the database schema, which includes table name, column name.

4) Extracting data:These types of attacks have a final goal of extracting unauthorized sensitive data from a database.

5) Adding or modifying data:These attacks are performed to add or modify information in the database of information systems of an organization.

6) Performing denial of service:These attacks are performed to disrupt the service provided by a database of a web application. This can be done, by disrupting a particular service or by shutting down the database itself.

7) Evading detection:These attacks aim at avoiding detection from various audit/security mechanisms in place.

8) Bypassing authentication: These attacks are performed to bypass the authentication mechanisms in database of web applications, since bypassing authentication allows them to get the access privileges of legitimate users.

9) Executing remote commands: These attacks attempt to execute malicious arbitrary commands remotely in the database.

10) Performing privilege escalation: These attacks use the implementation errors in the database to exploit the database user privileges.

III. TYPES OF SQL INJECTION ATTACKS

Six different types of attacks are performed together or sequentially depending on the goal of attacker [7]. For a successful SQLIA, the attacker should append a syntactically correct command to the original SQL query. The following six classifications of SQLIAs are presented.

A. Tautologies

A tautology is a formula which is true in every possible interpretation. In a tautologybased attack, the code is injected using the conditional OR operator such that the query always evaluates to TRUE. These attacks are usually bypass user authentication and extract data by inserting a tautology in the WHERE clause of a SQL query. The query transform the original condition into a tautology, causes all the rows in the database table are open to an unauthorized user. A typical SQL tautology has the form "or<comparison expression>", where the

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

comparison expression uses one or more relational operators to compareoperands and generate an always true condition. If an unauthorized user input user id is abcde and password is anything' or 'x'='x' then the resulting query will be:
select * from user_details where userid = 'abcde' and password = 'anything' or 'x'='x'.
Based on operator precedence, the WHERE clause is true for every row, therefore the query will return all records. In this way, an attacker will be able to view all the personal information of the users.

B. Logically Incorrect Queries

In this type of injection an attacker is try to gather information about the type and structure of the back-end database of a web application. The attack is considered as preliminary step for further attacks. If an incorrect query is sent to a database, some application servers return the default error message and the attacker takes the advantage of this weakness. They inject code in vulnerable or injectable parameters which creates syntax, type conversion, or logical error. Through this type of error one can identify the data types of certain columns. Logical errors often expose the names of tables and columns of database of web application [8].

C. Union Query

This type of attack can be done by inserting a UNION query into a vulnerable parameter which returns a dataset that is the union of the result of the original first query and the results of the injected query. The SQL UNION operator combines the results of two or more queries and makes a result set which includes fetched rows from the participating queries in the UNION. Basic rules for combining two or more queries using UNION as follows:
Number of columns and order of columns of all queries must be same.
The data types of the columns on involving table in each query should be same or compatible.
Usually returned column names are taken from the first query.
Suppose the attacker enters ' UNION SELECT * FROM emp_details -- in userid field and abcde in password field as userid and password which generates the following query:
SELECT * FROM user_details WHERE userid =" UNION SELECT * FROM EMP_DETAILS --
' and password = 'abcde' . The two dashes (--) comments out the rest of the query i.e. ' and password = 'abcde'. Therefore, the query becomes the union of two SELECT queries. The first SELECT query returns a null set because there is no matching record in the table user details. The second query returns all the data from the table emp_details.

D. Piggybacked Queries

In this attack the hacker inject additional queries to the original query; as a result the database receives multiple SQL queries. The first query is valid and executed normally; the subsequent queries are the injected queries, which are executed in addition to the first. Due to misconfiguration a system is vulnerable to piggy-backed queries and allows multiple statements in one query. Let an attacker inputs abcde as usrerid and '; drop table xyz -- as password in the login form. Then the application will generate the following query:
select * from user_details where userid = 'abcde' and password = ''; drop table xyz -- '
After completing the first query the database would recognize the query delimiter (";") and execute the injected second query. The result of executing the second query would be to drop table xyz, which would destroy valuable information.

E. Stored Procedure

A stored procedure is a type of SQL injection try to execute store procedures present in the database. Most of the databases have standard set of procedures that extend the functionality of the database and allow for interaction with the operating system. The attacker initially tries to find the database type with other injection method like illegal/logically incorrect queries. Once an attacker determine which databases is used in backend then he try to execute various procedures through injected code. Stored procedures can be vulnerable to execute remote commands, privilege escalation, buffer overflows and even provide administrative access to the operating system.
If an attacker injects ';SHUTDOWN; --into either the userid or password fields then it will generate the following SQL code : select * from user_details where userid = 'abcde' and password = ''; SHUTDOWN; -- '
The above command cause database to shut down [8].

F. Inference

By this type of attack, intruders change the behaviour of a database of web application.

There are two well-known attack techniques that are based on inference: blind injection and timing attacks.

1) Blind Injection: In this type of injection the attack is applied on well secured databases which do not return any usable feedback or descriptive error messages. The attack is normally created in the style of true false statement. After finding the vulnerable parameter, the attacker injects various conditions (that he wants to know whether they are true or false) through query and carefully observe the situation. If statement evaluates to true, the page continues to function normally. If false, the page behaves significantly different from the normal functioning. This type of injection is called Blind Injection [9].

2) Timing Attacks:In this type of attack an attacker design a conditional statement and inject through the vulnerable parameter and gather information based on time delays in the response of the database. In the following code :
http://www.abc.com/product.php?product_id=100 AND IF(version() like '5%', sleep(15), 'false'))-- Here an attacker checks whether the system is using a MySQL version is 5.x or not, making the server to delay the answer

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

in 15 seconds (the attacker can increase the delay's time) [9].

IV. RELATED WORK

The techniques reviewed can cover a subset of the vulnerabilities of SQL Injections. Important work on SQL injection attacks is presented as follows.

A. Roichman and Gudes's Scheme

This scheme proposed a fine-grained access control to the web databases of information systems of an organization. This scheme developed a new method based on fine-grained access control mechanism [10]. The access to the database of web application is supervised and monitored by the built-in database access control. This is preventionto the vulnerability of the SQL session traceability.

B. Shaukat Ali et al.'s Scheme

This schemeproposed the hash value based approach to improve the user authentication mechanism to protect against SQL injection attacks [11]. They used hash values of user name and password. The SQL Injection Protector for Authentication (SQLIPA) prototype was developed in order to test the SQL injection attack strings. The user name and password hash values were created and calculated at runtime for the first time when the particular user registered itself.

C. Thomas et al.'s Scheme

This scheme proposed an automated prepared statement generation algorithm to remove SQL injection vulnerabilities in web applications [12]. The scheme implemented in this research work using four open source projects namely: (i) Net-trust, (ii) Itrust, (iii) WebGoat, and (iv) Roller. On the basis of the empirical results, their prepared statement codes were able to successfully replace 94% of the SQL injection vulnerabilities in four open source projects.

D. SAFELI

This scheme dealt with the Static Analysis Framework in order to detect SQL Injection Vulnerabilities in web applications [13]. The aim of the framework is to identifying the SQL Injection attacks during the compile-time. This static analysis tool has two main superiority are: firstly, it does a White-box Static Analysis and secondly, it uses a Hybrid-Constraint Solver. If we consider the White-box we found the Static Analysis, the proposed approach considered the byte-code and dealt mainly with strings. While on the other hand, the Hybrid-Constraint Solver implemented the methods to an efficient string analysis tool which is able to deal with Boolean, integer and string variables.

E. Removing SQL Query Attribute Values

In this scheme the Authors proposed an technique to detect SQL injection attacks in web applications based on static and dynamic analysis [14]. This method removes the attribute values of SQL queries at runtime (dynamic method) and compares these values with the SQL queries analyzed in advance (static method) to detect the SQL injection attack in web applications. When run the application each dynamical generated query is compared or performed XOR operation with fixed query if it results zero, then that particular query allowed to the database and if it results to non-zero then that query reported as abnormal query stop sending to the database of information system of an organization.

V. PROPOSED SCHEME

We propose anauthentication scheme for protecting web applications against SQL injection attacks using hash function algorithm which would prevent these attacks effectively. In this study, a prototype website/model was developed for testing SQL injection attacks. Two levels of authentication schemes- data validation and hash function mechanism were developed for securing web applications. In this proposed technique, three approaches for data validation, and an approach to hash values of username and password in hash function based authentication were tested.

A. Prevention of SQLIA using Data Validation

For validation of data, we propose the three approaches (Escape single quotes, Reject bad input, and Accept only good input) for prevention of SQLIA as follows.

1) escape single quotes:

```
functionescape  ( input )
input = replace(input, “”,””)
escape = input
end function
```

2) Reject input that is known to be bad:

```
functionvalidate  string ( input )
known_bad = array( "select", "insert", "update", "delete",
    "drop", "--", """ )
validate_string = true
 for i = lbound( known_bad ) to ubound( known_bad ) if (
 instr( 1, input, known_bad(i), vbtextcompare ) !=0 ) then
validate_string = false
exit function
end if
next
end function
```

3)Accept only input that is known to be good:

```
functionvalidatepassword( input )
good_password_ch="abcdefghijklmnopqrstuvwxyzABCD
EFGHIJKLMNOPQRSTUVWXYZ0123456789"
    validatepassword = true
    for i = 1 to len( input )
    c = mid( input, i, 1 )
        if ( InStr( good_password_ch, c ) = 0 ) then
    validatepassword = false
    exit function
end if
next
end function
```

B. Preventionof SQLIA using Hash Functions

In the proposed approach, an authentication scheme based on Hash function algorithm was developedfor protecting web applications against SQL injection attacks. A prototype website/model was developed using MS Visual

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

Studio 2008 and SQL Server Management Studio Express 2005 for testing different SQL injection attack strings. Two extra columns were added in the login database, one for hash values of username and other for hash values of password. When the user gets itself registered with the web application, it selects its username and password. At the same time, hash value of username and password were computed at the coding side and stored in the login table with username and password. Whenever user log-in to the web application, hash value of username and password were matched at the backend and user was allowed to access the data. If SQL Injection attack string was entered for logging into the database, its hash value did not match with the hash values stored in the table and hence attacker could not access the database. During the authentication of user, the SQL query with hash parameters was used. Hence, if a user tries the injection to the query, and our proposed methodology is working with SQL query, it automatically detects the injections as the potentially harmful content and rejects the values.

Therefore, the attacker could not 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 not possible for the attacker/hacker to bypass theauthentication process through the general SQL injectiontechniques.

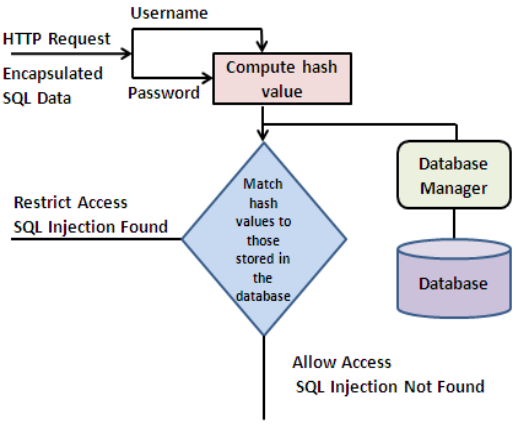

Fig 1. Proposed Hash scheme for preventing SQLIA

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 is dynamic at runtime.

VI. RESULT

The developed prototype website/model with authentication schemes was tested by entering all possible SQL injection strings and it was found that none of the string was able to penetrate. So, the attacker could not get the unauthorized access to the backend database of web application and could not bypass the authentication.

VII. CONCLUSION

Despite of many approaches and frameworks were developed and implemented in many interactive web applications, SQL Injection prevails as one of the top-10 vulnerabilities and threatens to online businesses targeting the backend databases. In this research, a technique for protecting authentication of web applications against SQL Injection attacks using hash function algorithm has been developed and testedwith different SQL injection attack strings. Hash values of user name and password have improved the authentication process with minimum overhead. This technique is quite useful and secure in protecting authentication of web applications against SQL Injection Attack. The technique 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 for prevention of SQL injections. Still, we need to improve our approach so that, it can prevent the backend databasesandweb applications from all kind of SQL injection attacks.

VIII. FUTURE SCOPE

In the present era of online world, web applications must provide full security and assurance to the users. During the review of research work, we found that in certain cases, these approaches were not found to be effective. Hence, these approaches were become not useful and could not able to detect the injections to prevent them. The proposed technique can only protect authentication mechanism of web applications. Rest of the SQL Injection techniques cannot be prevented using this technique. So, in future, we will try to improve the technique by making it fully secure and efficient for other types of SQL injection attacks also. Then, this technique will be able to prevent SQL Injection Attacks completely.

IX. ACKNOWLEDGMENT

I am sincerely thankful to **Er. Sandeep Jain,** Head and staff of Department of Computer science and Engg, Doon Valley Institute of Engg. & Technology, Karnal. We also wish to thank all the anonymous reviewers for their valuable suggestions, who helped in improving the manuscript.

REFERENCES

[1] S. W. Boyd and A. D. Keromytis, "SQLRand: Preventing SQL injection attacks," in Proc. of ACNS, 2004.
[2] R. Thenmozhi, M. Priyadharshini, V. VidhyaLakshmi, K. Abirami "Vulnerability Management in Web Applications" http://www.ciitresearch.org/dl/index.php/dmke/article/view/DMKE042013007
[3] Top 10 2013-A1-Injection, available at: http://www.owasp.org/index.php/Top_10_2013-A1-Injection, last accessed 11 June, 2013.
[4] Bojken, A. Shqiponja, A. Marin, and Xh. Aleksander,"Protection of Personal Data in Information Systems",International Journal of Computer Science, Vol. 10, No. 2,July 2013, ISSN (Online): 1694-0784.
[5] Malicious SQL Injection: an introduction homepage on hackmac. [Online]. Available: http://www.hackmac.org/tutorials/malicious-sql-injection-an-introduction/ , 2013.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

[6] Amit Kukreti ”SQL Injection Attacks homepage on codeproject” [Online].Available:http://www.codeproject.com/Articles/11020/SQLinjection-attacks/ , 2005.

[7] 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.

[8] Atefeh Tajpour ,Suhaimi Ibrahim, Mohammad Sharifi “Web Application Security by SQL Injection DetectionTools” IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012.

[9] Mayank Namdev, FehreenHasan, GauravShrivastav “Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA”International Journal of Advanced Research in Computer Science and Software Engineering,Volume 2, Issue 7, July 2012.

[10] Roichman, A., Gudes, E. “ Fine - grained Access Control to Web Databases”.In: Proc.Of 12th SACMAT Symposium, France, 2007.

[11] Ali, S., Shahzad, S.K., and Javed, H., SQLIPA: An Authentication Mechanism Against SQL Injection. European Journal of ScientificResearch, Vol. 38, No. 4, pp. 604-611, 2009.

[12] S. Thomas, L. Williams, and T. Xie, “On automated preparedstatement generation to remove SQL injection vulnerabilities”. Information and Software Technology, Elsevier 51, 589–598, 2009.

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

[14] I. Lee , S. Jeong, S. Yeoc, J. Moond, “A novel method for SQL injection attack detection based on removing SQL query attribute”, Journal of Mathematical and Computer Modeling, Elsevier 2011.