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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

Defenses To Protect Against SQL Injection Attacks

Neha Mishra[1], Sunita Gond[2]

Student-M-Tech, Department of Information Technology, B.U.I.T, Barkatullah University, Bhopal (M.P), India[1]

Professor**,** Department of Information Technology, B.U.I.T, Barkatullah University, Bhopal (M.P), India[2]

Abstract: Web applications are steadily increasing in our daily routines activities and continue to integrate them. Online Banking, On-line reservations, on-line shopping expect these web applications to be secure and reliable; the terror of SQL–Injection Attacks has become increasingly frequent and serious. SQL Injection Attacks are one of the topmost threats for web application security. Using SQL Injection attackers can leak confidential information: such as credit card numbers, ATM pins, User credentials from web applications and even corrupt the database. This paper presents a new technique to protect Web applications against SQL injection Attacks. SQL Injection Attacks are a class of attacks that many of these systems are highly vulnerable to, and there is no known foolproof defense against such attacks. In this paper, some predefined methods are discussed and integrated approach of encryption method with secure hashing is applied in the database to avoid attack on login phase. This combined method is applied to a system where user's information is kept and the designing of this system are done by using PHP and MYSQL.

Index –Terms*:* Database security, SQL injection attacks, Hashing, Encryption technique, Preventions.

I. INTRODUCTION

With the rising use of internet, web application vulnerability has been increasing effectively. All web applications are depended on the Internet. Example:-e-banking, admission portals, online shopping, and various government activities like online electricity bills payment etc. Since these applications are used by hundreds of people, in many cases the security level is weak, which makes them vulnerable to be attacked by external users. From time to time, the users need to interact with the backend databases through the user interfaces for various tasks such as: modify data, manipulating queries, extracting data, and so forth. For all these operations, design interface plays crucial role, the quality of which has a great bang on the security of the stored data in the database. A less secure Web application design may allow crafted injection and malicious update on the backend database. This trend can cause lots of damages and thefts of trusted users' sensitive data by unauthorized users. In the worst case, the attacker may gainful control over the Web application and totally destroy or damage the system. This is effectively achieved, in general, through SQL injection attacks on the online Web application database.

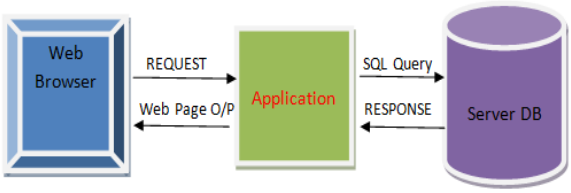

Fig.1. Web Application (REQUEST/RESPONSE) Structure

According to OWASP report released in 2012, SQL Injection attacks are top most risk/danger to Web applications [11].SQL Injection Attack occurs when adversary changes the logic, semantics or syntax of an SQL query [1]. The query which is generated dynamically based on user input, maliciously crafted with SQL keywords, operators, strings or literals, executes in the database server. The aim of the intruder for the SQL Injection Attack is to access database for which he is unauthorized [2]. So, accessing information beyond limitations intruder applies SQL Injection Attack in the form of queries which are syntactically correct [3]. It is, therefore, top important to prevent such types of attacks, and topics of research in the industry and academia. There has been significant progress in the field and a number of models have been proposed and developed to counter SQL Injection Attack, but no one have been able to assure absolute level of security in web applications, mainly due to the variety and scope of SQL Injection Attack's. One common encoding practice in today's times to avoid SQL Injection Attack is to use

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

database stored procedures as an alternative of direct SQL statements to interact with original databases in a web application. However, there are vulnerabilities in this scheme too, most notably when dynamic SQL statements are used in the stored procedures, to obtain the database objects during runtime. Our work is centered on this particular type of vulnerability in stored procedures and we expand a scheme for detection of SQL Injection Attack in scenarios where dynamic SQL statements are used. This paper is prearranged as follows: Section I show the introduction of web attack and how SQL Injection Attack is vulnerable reason of attack, section II show update about SQL Injection Attack, section III show varieties of SQL Injection Attack and in IV section show the various methods used for detecting and preventing SQL Injection Attack and in V section our projected work and in last section conclusion is present.

II. OVERVIEW OF SQL INJECTION ATTACK?

While Typing SQL keywords and control signs an intruder is able to modify the structure of SQL query developed by a Web designer. SQL Injection is a kind of web application security exposure in which an attacker is able to expose a database SQL command, which is executed by a web application. SQL Injection attacks can take place when a web application utilizes user-supplied data without proper validation or encoding as part of a command or query.

SQL injection attacks are nothing but injecting malicious queries by the hackers into the application projected queries to get the desired outputs from the database.SQL Injection allows an attacker to create, read, update, modify, or delete data stored in the back-end database.

Aspects of SQL Injection:-

- SQL injection is a software exposure that occurs when data entered by users is sent to the SQL interpreter as a piece of an SQL query.
- Attackers provide specially crafted input data to the SQL interpreter and trick the interpreter to execute unintentional orders [2]
- Attackers utilize this exposure by providing specially crafted input data to the SQL interpreter in such a way that the interpreter is unable to distinguish between the actual commands and the attacker's specially crafted data. The interpreter is tricked into executing unintended commands

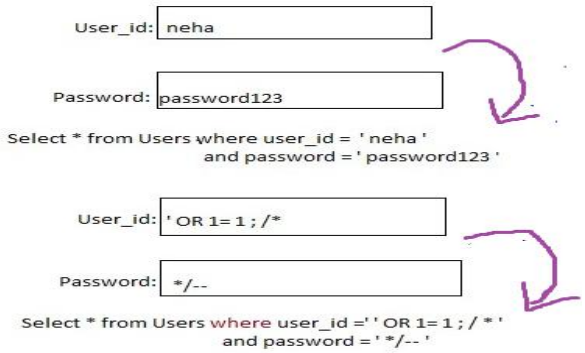

Fig 2. SQL Injection

Thus, SQL injection exploits security vulnerabilities at the database layer.

III SQL INJECTION ATTACK TYPES

There are diverse methods of attacks that depending on the goal of attacker are performed together or sequentially and its classification is given below:-

Tautologies: - This type of attack injects SQL tokens to the conditional query statement to be evaluated always true. This type of attack used to avoid authentication control and access to data by exploiting vulnerable input field which use WHERE clause. "SELECT * FROM employee WHERE userid = '211' and password ='bbb' OR '1 '='1 III as the tautology statement (1=1) has been added to the query statement so it is always true.

Illegal/Logically Incorrect Queries:- When a query is not needed, an error message is returned from the database including useful debugging information. This error messages help attacker to discover vulnerable parameters in the application and consequently database of the application. In fact attacker injects junk input or SQL tokens in query to produce syntax error, type mismatches, or logical errors by reason. In this example attacker makes a type mismatch error by injecting the following text into the *pin* input field.

1)Original
URL:http://www.arch.polimi.it/eventi/?id_nav=886
2)SQLInjection:
http:/`/www.arch.polimi.it/eventi/?id_nav=8864'
3) Error message showed:
SELECT name FROM Employee WHERE id =8864\' from the message error we can find out name of table and fields: name; Employee; id. By the gained information attacker can arrange more strict attacks [3].

Union Query*:* By this method, attackers join injected query to the safe and sound query by the word UNION and then can get data about other tables from the application. The

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

output of this attack is that the database returns a dataset that is the union of the results of the original query with the results of the injected query.
SELECT Name, Address FROM Users WHERE Id=$id by injecting the following-
Id value: *$id=1 UNION ALL SELECT creditCardNumber,1 FROM CreditCarTable.*
We will have the following query: -
SELECT Name, Address FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber, 1 FROM CreditCarTable which will join the result of the original query with all the credit card users.

Piggy-backed Queries: In the piggy-backed Query attacker tries to add on additional queries to the original query string. In this method the first query is original whereas the following queries are injected one. Here the intruders exploit database by the query delimiter, such as ";", to append extra query to the original query. With a successful attack database receives and execute a multiple different queries. In the following example, attacker inject " 0; *drop table user* " into the *pin* input field instead of logical value. Then the application would produce the query: *SELECT info FROM users WHERE login='aaa' ANDpin=0; drop table users*
Because of ";" character, database accepts both queries and executes them. The second query is illegitimate and can drop *users* table from the database. It is clear that some databases do not need special separation character in multiple distinct queries, so for detecting this type of attack, scanning for a special character is not remarkable solution.

Stored Procedure: In this technique, attacker focuses on the stored procedures which are present in the database system. Stored procedures run directly by the database engine. Stored procedure is nothing but a code and it can be vulnerable as program code. For authorized/unauthorized user the stored procedure returns true/false. As an SQL Injection Attack, intruder input "; SHUTDOWN; --" for username or password. Then the stored procedure generates the following query:
For example:-
SELECT accounts FROM users WHERE login= '1111' AND pass='1234 '; SHUTDOWN;--; this type of attack works as piggyback attack. The first original query is executed and consequently the second query which is illegitimate is executed and causes database shut down. So, it is considerable that stored procedures are as vulnerable as web application code

Inference: By this type of attack, intruders change the behavior of a database or application. There are two well-known attack techniques that are based on inference: blind injection and timing attacks.

• **Blind Injection**: At times developers hide the error details which help attackers to compromise the database. In this situation attacker face to a generic page provided by developer, instead of an error message. So the SQL Injection Attack would be very difficult but not impossible. An attacker can still steal data by asking a series of True/False questions through SQL statements.
SELECT accounts FROM users WHERE login= 'doe' and 1 =0 -- AND pass = AND pin=O SELECT accounts FROM users WHERE login= 'doe' and 1 = 1 -- AND pass = AND pin=O
If the application is secured, both queries would be unsuccessful, because of input validation. But if there is no input validation, the attacker can try the chance. First the attacker submits the first query and receives an error message because of "1 =0 ". So the attacker does not understand the error is for input validation or for logical error in query. Then the attacker submits the second query which always true. If there is no login error message, then the attacker finds the login field vulnerable to injection [2].

• **Timing Attacks**: A timing attack lets attacker gather information from a database by observing timing delays in the database's responses. This technique by using if-then statement cause the SQL engine to execute a long running query or a time delay statement depending on the logic injected. This attack is similar to blind injection and attacker can then measure the time the page takes to load to determine if the injected statement is true. This technique uses an if-then statement for injecting queries. WAITFOR is a keyword along the branches, which causes the database to delay its response by a specified time.
For example, in the following query: declare @ varchar(8000) select @ = db_nameO if (ascii(substring(@, 1, 1)) & (power(2, 0))) > 0 waitfor delay '0:0:5' Database will pause for five seconds if the first bit of the first byte of the name of the current database is 1. Then code is then injected to generate a delay in response time when the condition is true. Also, attacker can ask a series of other questions about this character. As these examples show, the information is extracted from the database using a vulnerable parameter.[2]

IV RELATED WORK

In order to detect and prevent SQL Injection attacks, filtering and other detection methods are being researched. This section explains the related work.

IndraniBalasundaram In [6]This paper proposes technique―Service-Oriented Authentication‖ is to prevent SQL–Injection Attacks in database the deployment of this technique is by appending first level Service has the functionality of Tame-card detection and Prevention. The

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

Second level Service has the functionality of Authentication Checker also dataset (the temporary storage of database) of application scripts additionally allowing seamless integration with currently-deployed systems.

[1] ID	[2] Username	[3] Password
[4] 1	[5] Neha	[6] password123
[7] 2	[8] Rishab	[9] Rish84

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

William G.J.Halfondet al.‟s Scheme- [10] - proposed an approach that works by combining static analysis and runtime monitoring of database queries. In its static part, technique uses program analysis to automatically build a model of the legitimate queries that will be generated by the application. While in the dynamic part, the technique monitors the dynamically runtime generated queries and checks them for acceptability with the statically-generated model. A query that doesn't match with the model represent potential SQL Injection Attacks and are hence prevented from executing on the database and reported

Id	Username	Password	Hash_username*	Hash_ password*
1	Neha	password123	14AB6EE730DFC50936B5 45A683A0719380B9E9E6	F5629535196ECA033BBA8 02EA1A576FBC936EB0
2	Rishab	Rish84	647C02E46B4DA59498860 C21AA4A9391DDC52F1A	8FA94ACBAC6B92269B5E 8C280ECB55E443661655A

Table 2: User table with security guidelines also contains the hash values.

Sonam Panda Approach - In [3], propose a technique where predefined methods are used and hybrid encryption method is applied in the database to stay away from attack on login phase. This applied hybrid encryption method is a integration of Advanced Encryption Standard (AES) and Rabin cryptosystem.

Ali et al.‟s Scheme - [8] adopts the hash value approach to further improve the user authentication mechanism. They use the user name and password hash values SQLIPA (SQL Injection Protector for Authentication) prototype was developed in order to test the framework. The user name and password hash values are created and calculated at runtime for the first time the particular user account is created.

V PROPOSED TECHNIQUE

In this paper we proposed a new technique for preventing Database against SQL injection attack using an Integrated Approach. During this approach, in a Login Table, two columns are created by DBA. One for username and other is used for password. Our tactic requires two more columns. One for the secure hash value of username and one for the password. The secure hash values of username and password are designed and stored in Login Table when the user's account is first time created with the web application. Whenever user desires to login to database his/her identity is verified via username, password and secure hash values. These secure hash values are generated at runtime using stored procedure when user wants to login into the database.

In database there also separate tables for AES encryption and combined approach of secure hashing on encryption. Different techniques and methods are being developed which are used to protect the database. By applying encryption technique, database attacks can be prevented. Encryption of data basically helps to change the data into a form that is not readable [1]. Without the correct key, this format can't be deciphered even if attacker hacks the information. Application of encryption in login phase makes it difficult for unauthorized users to access the database. In this paper, some predefined methods are discussed and mixture of encryption method with secure hashing is applied in the database to avoid attack on login phase. This applied integrated approach is an encryption method which is a combination of Advanced Encryption Standard (AES) and Secure Hashing technique.

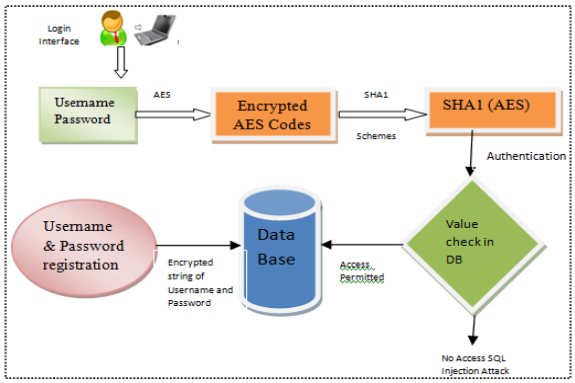

Fig.3. Proposed integrated approach architecture

VI. CONCLUSION AND FUTURE WORK

The SQL - Injection Attacks are tremendously dangerous in association to other types of Web-based attacks, for the reason that here the end result is data manipulation.SQL injection holes can be easily exploited by a technique called SQL Injection Attacks. The web-application code perfectly

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

contains a policy that allows distinguishing lawful and malicious queries. In this paper, different varieties of SQL injection attacks as well as predefined prevention methods are discussed. Then the hybrid approach of encryption is used which includes AES
encryption and Secure hashing method. The reason behind applying two layer of encryption & hashing is that it will provide more security in database. SQL query is generated and encrypted by using AES technique which is further hashed by a secure hashing technique and as we know hashed codes is a one way encryption technique thus it is not possible to decode it. This proposed integrated approach is an effort to add some more security measures to databases to avoid SQL injection attack.

In this work, we have concentrated on the particular area of SQL injection. I think that this area is in need of more research, mainly because of various reasons: - SQL injection attacks are most probable to change and new vulnerabilities will be found, collectively with new countermeasures to deal with them. As many hacking sites are existing on the web, and since attack methods are well described and circulated between hackers, we believe that information about new attack methods must be constantly surveyed and new counter measures should be developed. So, in future, I will try to develop the technique by making it capable for other varieties of SQL Injection Attacks also. Due to which, this technique will be able to prevent SQL Injection Attack totally.

VII ACKNOWLEDGMENT

I am sincerely thankful to Prof. Sunita Gond from B.U.I.T, Barkatullah University, Bhopal (M.P), for her kind support and guidance.

VIII REFERENCES

[1] Priyanka, Vijay Kumar Bohat," Detection of SQL Injection Attack and Various Prevention Strategies**",** International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-4, April 2013

[2]Sonam Panda, 1 Ramani S2, "Protection of Web Application against Sql Injection Attacks", International Journal of Modern Engineering Research (IJMER) Vol.3, Issue.1, Jan-Feb. 2013 pp-166-168 ISSN: 2249-6645

[3]Mihir Gandhi, JwalantBaria,s "SQL INJECTION Attacks in Web Application"International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013

[4]By Mayank Namdev*, Fehreen Hasan, Gaurav Shrivastav "Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA"Volume 2, Issue 7, July 2012.

[5]Shubham Srivastava1, Rajeev Ranjan Kumar Tripathi, "Attacks Due to SQL Injection & Their Prevention Method for Web-Application ", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012,3615-3618

[6] IndraniBalasundaram,E.Ramaraj,"An Authentication Scheme for Preventing SQL injection Attack using Hybrid Encryption" (ISSN 1450-216,2011, Vol.53,pp.359-368.

[7]AtefehTajpour et al. "Evaluation of SQL Injection Detection and Prevention Techniques" Second International Conference on Computational Intelligence, 2010.

[8]Shaukat Ali, Azhar Rauf, Huma Javed," SQLIPA: An Authentication Mechanism Against SQL Injection", European Journal of Scientific Research ISSN 1450-216X Vol.38 No.4 (2009), pp 604-611

[9]Sayyed Mohammad Sadegh Sajjadi and Bahare Tajalli Pour, "Study of SQL Injection Attacks and Countermeasures", International Journal of Computer and Communication Engineering, Vol. 2, No. 5, September 2013

[10] W. G. Halfond, J. Viegas, and A. Orso , "A Classification of SQL Injection Attacks and Countermeasures," in Proc. the International Symposium on Secure Software Engineering 2006.

[11] The Open Web Application Security Project, OWAP TOP 10 Projects. [Online]. Available: http://www.owasp.org/

BIOGRAPHIES

Neha Mishra received the B.E. degree in Computer Science, in 2007; and pursuing M-Tech. degree in Information Technology, from B.U.I.T, Barkatullah University Bhopal (M.P). She worked as an Assistant Professor in Computer Engineering Branch in JSPM, ICOER Wagholi, Pune. Her current research interests include Network Security.