International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

SQLIA – Detection and Prevention

Prachit Raut[1], Vedang Gharat [2]

[1]*Student, Department of Computer Engineering, Padmabhushan Vasant Dada Patil Pratishthan's College of Engineering, Mumbai, Maharashtra, India*
[2]*Student, Department of Information Technology, Universal College of Engineering, Vasai*

Abstract - *In present era, cyber-attacks are used to steal and manipulate huge volume of assets from different lines of businesses. So, it is necessary to protect our important assets from such attacks. SQL Injection Attack has been one of the top most vulnerabilities in OWASP's Top 10 list of security vulnerabilities. SQL Injection compromises confidentiality, availability and integrity of the data. Therefore, it is mandatory to protect user data from such attack. In SQL Injection attack the attacker intentionally uses harmful characters, and inputs it in such a way that it alters the collection of user assets and gets the desired data. This research paper is prepared in such a fashion that it gives a complete analysis about SQL Injection, SQL Injection types, Detection methods and preventive measures for this attack.*

Key Words: **SQL Injection, Stored, Union, Select, Prevention, Detection, Architecture, SQL Queries, Token, Inferential.**

1. INTRODUCTION

Now-a-days web applications are prevalent and has become essential for every individual. There are n number of security contraventions that takes place every day one of them is SQL Injection attack. SQL is abbreviated as Structured Query Language [1]. The main purpose of this language is to communicate with the database in order to manipulate the data. SQL Injection is commonly referred as SQLI, it is an attack in which an unauthorized user tries to attack the system or a network using a malignant SQL queries in order to get the access of the database and manipulate the data which is not meant to be accessed. Unfiltered SQL queries are the prime reasons for this attack. Various SQLI attack techniques implemented by attackers are combinations of SQL statements. Today, more than 1 million websites are being infected because of SQLI attack. There are different types of SQLI attacks and each one uses different manner to attack the website. The classic SQLI attacks were easy to block so to not manipulate the data form the databases. But the combination of SQLI attack and XSS attacks makes the data extraction and manipulation of the data from the database much easier [3]. Various techniques were discussed to surpass SQL Injections. One of them was to implement avert code with conventional encoding and decoding techniques. Even today defensive codes are implemented but they are not that strong enough to protect the data from SQLI attacks. Although there are many ways to prohibit SQLI vulnerabilities the main problem all of these is that their application is difficult to implement. These methods are likely to have human errors and are not applied as automated techniques. The rest of the paper is organized as follows:

2. Insights of SQLI attack
3. Types of SQLI Attacks
4. Risks related to SQLI attack
5. Ways to attack,
6. Attack Prevention Techniques
7. Conclusion
8. References

2. INSIGHTS OF SQLI ATTACK

SQLI Attack is considered to be severe type of attack affecting confidentiality, integrity and availability of the data. This attack adds harmful SQL code to a web form input box to gain access or manipulate the data in the database [2]. By using this weakness an attacker can send his commands directly to the database and destroy the complete functionality of an application. Attacker can use this attack to execute system level commands that may result in denial of the service to the application by the system.

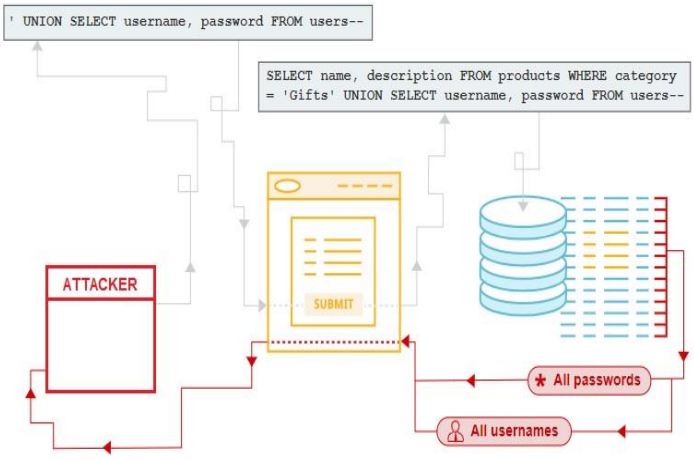

Fig 1. Explanation of a simple SQLI attack

3. TYPES OF SQLI ATTACKS

SQLI attacks can be performed in various ways. Attacker keeps a track of system's behavior and then selects the best suitable attack for a specific system or a network. The attack is very much common with ASP and PHP applications as it

uses much older functional interface which are easily vulnerable to SQLI Attacks and which makes attacker to penetrate through them easily and manipulation becomes less difficult.

Following are the types of SQLI attacks [4].

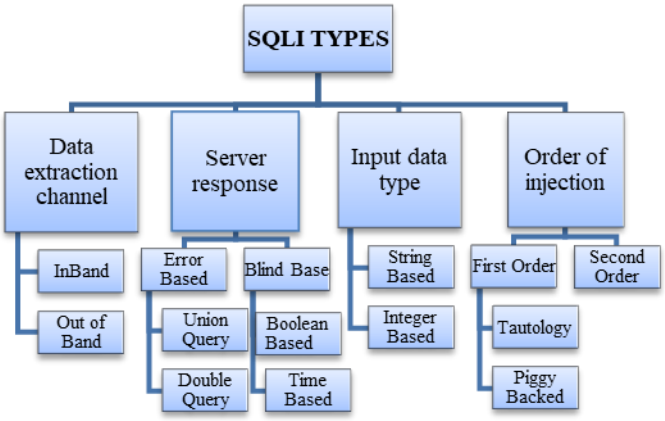

Fig 2. Types of SQLI attack

1) InBand SQLI Attack: In an InBand SQLI Attack the attacker uses similar communication channel for attacking and stealing the data from that system. It is most commonly used and easy SQLI attack, also known as Classic SQLI Attack. Typically, in this attack an attacker sends the payload through GET or POST HTTP request and in return the data will be inserted in the page which is returned by the server.
2) Out of Band SQLI Attack: In this attack the attacker does not use same communication channel, instead it depends on the features that are enabled in the server's database that is being used by the web application. Out-of-band method offers the attacker a substitute to time-based method if the server's response is unstable [14]. Out of band SQLI technique depends on database server to make DNS or HTTP requests to deliver data to the attacker. Such is the case with Oracle's database UTL_HTTP package which is used to send HTTP requests from a SQL server which the hacker controls.
3) Piggy Backing Attack: In this attack the attacker "Piggy Backs" the query with the genuine query as input in a web-application.[5]
The purpose of this attack is retrieval of information from the database and Denial of service to the genuine user. Piggy Backing can be defined as "on the back of other". In this a query works as normal query and it is piggy backed by another harmful query, when the query is executed both of them gets executed which makes the database vulnerable exploitation becomes easy. For example: select customer_details from accounts where login_id="administrator" AND pass='1234'; DELETE FROM accounts WHERE Customer_Name="John";
After execution of the first query the interpreter sees the ';' Semi Colon and executes the second query with the first query. The second query is harmful and so it will delete the all the data of the customer 'John'.

4) Blind Base (Boolean) Attack: Blind base SQLI is an inferential SQLI which depends on sending a SQL query to the database which impels the application to return different output deciding whether the query returns a TRUE or FALSE result. Depending on the result the data with HTTP response might change or be the same. In such situation the attacker might interfere if the result returned is true or false even if no data is
returned from the database [16]. The attack usually slow especially on huge databases as the attacker will need to extract a database, character by character [6].
5) Union Query SQLI Attack: This attack uses Union operator (U) in the SQL Query. The SQL queries are joined using this Union operator. While firing the query the first statement is a normal query after which the harmful query is adjoined to it with a Union operator. The attacker must design a SELECT statement similar to the Original query. For this a valid table should be known to the attacker, also it is necessary to know the number of column and rows in first query and their data type.
It bypasses the prevention and detection mechanism of the system [13].

```
SELECT * FROM userTable WHERE
user_Id='1111' UNION SELECT * FROM
memberTable WHERE member_Id='admin' --'
AND password='1234';
```

Fig 3. Union query injection attack.

6) Blind Base (Time) Attack: It is an Inferential Injection technique that depends on sending an SQL query to the database and it forces the database to wait for some specific time (in seconds) before returning. The response time indicates to the attacker whether the outcome of the query is True or False. After the result is returned, an HTTP response will be returned with a delay (in Seconds) or might returned immediately. This allows the attacker to conclude if the payload used returned true or false, even if no data was returned from the database [9].
For example: http://example.com/items.php?id=2 this url will send query to database to SELECT title, description, body FROM items WHERE ID = 2, after this attacker might fire a query which will return true and other with false this will makes it easier for the attacker to distinguish between the two results and to easily manipulate the data .
7) Error Based SQLI Attack: An Error Based SQLI Attack is useful when the attacker is unable to exploit the vulnerability in the database using other techniques. In this attack the database is forced to perform some operation which will result in an error and then the attacker tries to gain some information from the error messaged displayed [8].
8) Tautology: In Tautology based injection the attacker injects one or more conditional statements which are always true and it is most commonly used for bypassing the login pages and extracting the data from the database. Attacker

injects the query in the input field, the query is transformed into tautology which causes all the rows which are aimed in the query to return.

For Example: Bypassing the login page:

SELECT name from users where username='a' OR '1' = '1' AND password='a' OR '1' = '1'.

The code injected (OR 1=1) transforms the entire WHERE clause in a tautology. The database evaluates the condition in the statement and if the condition turns out to be true it checks the rows in the query and returns the specific rows or all of them. Above mentioned query will return all the possible entities from the database.

9) Compounded SQLI Attack: Compound spoofing also known as content injection is a combination of Content Spoofing and SQL Injection. Content spoofing mainly focuses on injecting harmful content to the websites and is basically a client-side attack [12]. Whereas the SQL Injection attack mainly focuses to target the databases in which attacker crafts the malicious code in such a fashion that it would be treated as a legit query and is a server-side attack which provides ability to the attacker for unauthorized access to database.

```
http://victim-site/product.php?=1 and 0 union select
1,"</td><td></td><td></td></tr></table><h1>There is some
probplem</h1><h2>Please login again to continue the operation</h2><form
method= 'POST' action='http://attacker-site/login.php'>Username: <input
type='text' name='username' /><br />Password: <input type='password'
name='password' /><br /><input type='submit' value='Login' /></form><!--
",3,4
```

Fig 4. Injection of HTML code via vulnerable column

Type	Objective	SQL query
Union	Data extraction	Select * from stu_data where stuid='' **union** select * from details -- and pwd='b';
Tautologies	Authentication bypass	Select * from stu_data where stuid='vxyz' and pwd ='b' or **'4'='4'**
Piggy Backing	Dataset Extraction	Select Rno FROM Stu WHERE login = 'abc' AND pass = ''**; DROP table St --'**
Incorrect queries	Identification of injectable parameters	SELECT * FROM students WHERE username = **'aaa'"** AND password =
Inference	Determining Database Schema	SELECT name, email FROM members WHERE id=5; **IF SYSTEM_USER='sa' SELECT 1/0 ELSE SELECT 5**

4. RISKS RELATED TO SQLI ATTACK

SQLI attacks are harmful and risks related to it encourages hackers to attack the database. The main out-comes of this attack are as follows:

1) Loss of confidentiality: Databases contains highly sensitive
data such as Phone numbers, Bank Account numbers, etc. Attack on database can cause misuse of such data and which results in loss of confidentiality.

2) Loss of Integrity: After attacking the database the attacker might manipulate the data by modifying or deleting data and such manipulation of data can cause let loss of data integrity.

3)Loss of Authentication: If the attacker successfully bypasses the login page and gets into the database then the authenticity is lost as unauthorize user gets successfully in the database.

5. WAYS TO ATTACK

There are many reasons which makes the system prone to attacks. Here are some of the many reasons which cause loopholes in a system which makes in vulnerable.

1) Tempering of URL: URL tempering is one of the widely used common way to access data. Detection is comparatively tough, also it requires much skills. An URL might contain sensitive information like Primary Key IDs, any entity names, such information can be used to gain access to data. Tempering or modifying URL can be achieved by attaching harmful strings to the URL. SQL Queries can also be used in the URL to check if the website is vulnerable.

2) Improper input validation: Incorrect input validation is the main source of SQLI attacks. So, to avoid such vulnerabilities avert coding must be done. Encoding of the input must be done as attacker mostly uses the meta characters into the query which can confuse the SQL parser and illuminate the user inputs as SQL tokens so restricting the use of such meta characters will terminate the use of such characters. Solution to this is encryption of user queries in such a fashion that all the meat characters are encrypted and are illuminated as general characters to the database system.

3) Identify input sources: One must check all the input sources to their application as there may be many possible ways to input the data to the application. These input sources can be a gateway
for the vulnerabilities to get in the database [17].

4) Use of automated tools: Automated tools have made it easier to exploit and manipulate data. Tools such as SQL Map, Whitewidow, DSSS, Blisqy, etc. are used. These tools hold various payloads in them to detect and exploit SQL vulnerabilities.

6. ATTACK PREVENTION TECHNIQUES

1) Sanitizing and validating user inputs: Validation of user inputs is necessary to check if the given user data as input is acceptable for whatever user intends to use it for. Another reason could be if the user gives wrong input then it should prompt the user about invalid input, rather than giving access to the information. To avoid this, we can use an PDO (PHP Data Objects). PDO is an interface for accessing the database, it also supports 12 different database drivers.

Working of PDO Statements:

A) Prepare a query with null values as placeholders with either a question mark or a variable name with a colon preceding it for each value.
B) Attach the values to the placeholders and execute the query.
Now, to prevent password from cracking, the php.ini file should look like this**:** *display_errors = Off and log_errors = On.* This will all the error individually gather in the error log instead of popping it up or printing it. But if an attacker gets the logs it would be harmful. So, we would use a *try/catch OR set_exception_handler* while doing *error_log($e->getMessage()).*

2) Using SQL Parameters: SQL Parameters does check for type and length validation. Validation is done by both user and client side. If the values appeared to be outside the range it would trigger an error. The code below elaborates the use of parameter collections:

- *SqlDataAdapter myCommand = new SqlDataAdapter("UserLogin", conn);*
- *myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;*
- *SqlParameter parm = myCommand. SelectCommand.Parameters.Add ("@cust_id", SqlDbType.VarChar, 15);*
- *parm.Value = Login.Text;*

In this example above, the @cust_id is treated as a literal value instead of as executable code. The value is first checked for length and type and if the value of @cust_id does not fulfill with the given length and type Value, an exception will pop.

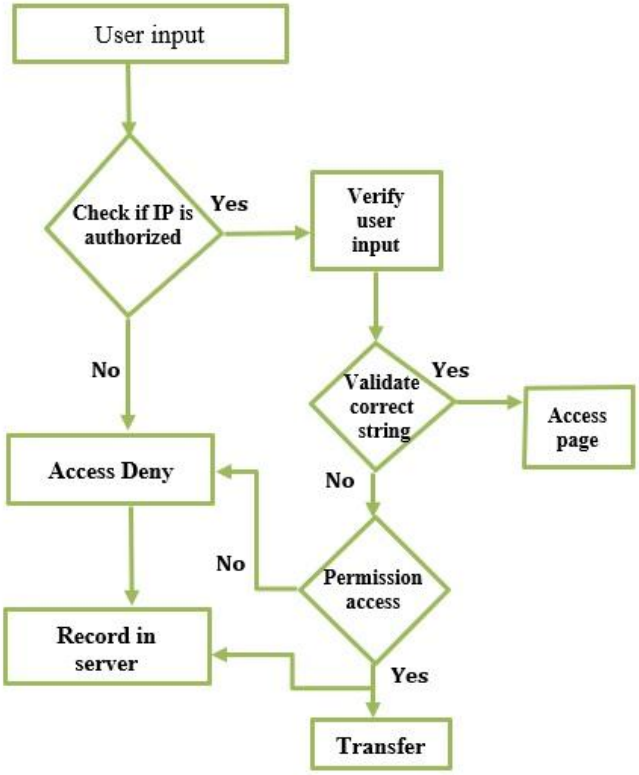

Fig 5. Defense Mechanism

This defense mechanism is mainly designed to prevent SQLIA's. The defense mechanism primarily checks for input information and also detects web address bar for information, mainly sensitive character detection. First the server side checks for the IP address authenticity. Access to the server is denied for the users with unverified input values and if the input value matches SQL defense mechanism rule, only then the user is allowed to access the server. At last the server checks and verifies input values for user to give him privileges. If the user exceeds beyond the given number of permissions the system sends the alert message to the administrator and the user is blocked instantly. Besides this, the server also monitors the activity simultaneously and records the attack when all the verification done is proved as invalid.

7. CONCLUSION

In this paper we've discussed SQL injection attack with its principle, attack execution process, prevention and detection techniques. As SQL injection attack is mainly performed on databases and application development and so for the vast majority of firewall are unable to bypass this attack successfully.

Although the database server is updated, the scripting languages used are still vulnerable itself. But as the SQL technology is continuously evolving the loopholes and threat continues to the databases. Hence the prevention and detection methods should also need to evolve. More attention towards server and databases configuration and

more sanitization and filtration of the inputs must be considered in order to prevent attack. Also, while developing a web application one must built it with security in mind and each should be thoroughly tested for SQL Injection attack vulnerabilities.

8. REFERENCES

[1] Etienne Janot, Pavol Zavarsky "Preventing SQL Injections in Online Applications" OWASP Application security conference May 2018, Ghent, Belgium
[2] XuePing-Chen "SQL injection attack and guard technical research" 2011 Published by Elsevier Ltd
[3] Rubidha Devi.D, R.Venkatesan, Raghuraman.K "A study on SQL injection techniques" IJPT| Dec-2016 | Vol. 8 | Issue No.4 | 22405-22415
[4] "Stephanie Reetz" SQL Injection Technical MS-ISAC White Paper , May 2017
[5] Subhranil Som , Sapna Sinha , Ritu Kataria, " Study on sql injection attacks: mode, detection and prevention" International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 8, ISSN No. 2455-2143, Pages 23-29
Published Online June - July 2016 in IJEAST (http://www.ijeast.com)
[6] Zainab S. Alwan, Manal F.Younis "Detection and Prevention of SQL Injection Attack: A Survey" IJCSMC, Vol. 6, Issue. 8, August 2017
[7] 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
[8] Nanhay Singh, Khushal Singh, Ram Shringar Raw ("Analysis of Detection and Prevention of Various SQL Injection Attacks on Web Applications" www.ijais.org
[9] Nikita Patel, Fahim Mohammed, Santosh Soni "SQL Injection Attacks: Techniques and Protection Mechanisms"
[10] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks" ACM Transactions on Information and System Security March 2010 Article No.: 14
[11] Kirti Randhe, Vishal Mogal, "Security Engine for prevention of SQL Injection and CSS Attacks using DataSanitization Technique"s , Vol. 3, Issue 6, June 2015
[12] G. Kontaxis, D. Antoniades, I. Polakis, and E. P. Markatos "Anempirical study on the security of cross-domain policies in rich internet applications" EUROSEC '11: Proceedings of the Fourth European Workshop on System Security April 2011 Article No.: 7
[13] Nabeel Salih Ali, Abd Samad Shibghatullah "Protection Web Applications using Real-Time Technique to Detect Structured Query Language Injection Attacks"International Journal of Computer Applications Foundation of Computer Science (FCS), NY, USA Volume 149 - Number 6, 2016
[14] Sonam Panda, Ramani "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
[15] Chris Anley "Advanced SQL Injection in SQL Server Applications" An NGSSoftware Insight Security Research (NISR) Publication ©2002 Next Generation Security Software Ltd
[16] "OWASP Top Ten"
https://owasp.org/www-project-top-ten/
[17] S V Athawale, M A Pund "An Improved Network-based Intrusion Detection System for Virtual Private Networks" Journal of Information and Computing Science, Vol. 13, No. 2, 2018