International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 01 | Jan 2021 www.irjet.net p-ISSN: 2395-0072

SQL INJECTION: ATTACK & MITIGATION

Vishnu Menon

Student, Dept. of Information Technology, Keraleeya Samajam (REGD.) Model College Maharashtra, India

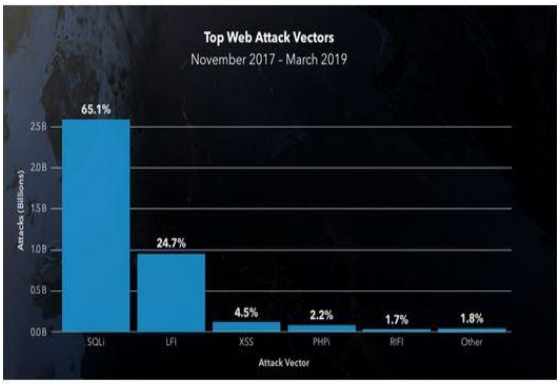
---

ABSTRACT: - *Nowadays, all around the world, SQL Injection Attack (SQLI) is considered to be the serious security threat over the Internet for many web-based applications. Organizations use various kinds of Web Applications for various business purposes. These organizations use SQL Language in MySQL, PostgreSQL to store information into the database. SQL Injection is flexible, hence powerful. SQL has become a center of attention of hackers due to its massive usage in almost every web application. They (Hackers) take advantage of poorly coded Web Applications to breach the database. They make use of SQL Query through unauthorized user input into the legitimate SQL Statement to gain unauthorized access to the database and gather confidential information. In this paper, we have tried to present different types of SQL Injection Attacks (SQLIA), it's detection and prevention methods used.*

Keywords: - *SQL Injection, OWASP Top-Ten, DDoS Attack, Authentication Bypass, XSS, Attack Types, Mitigation*

1. INTRODUCTION: -

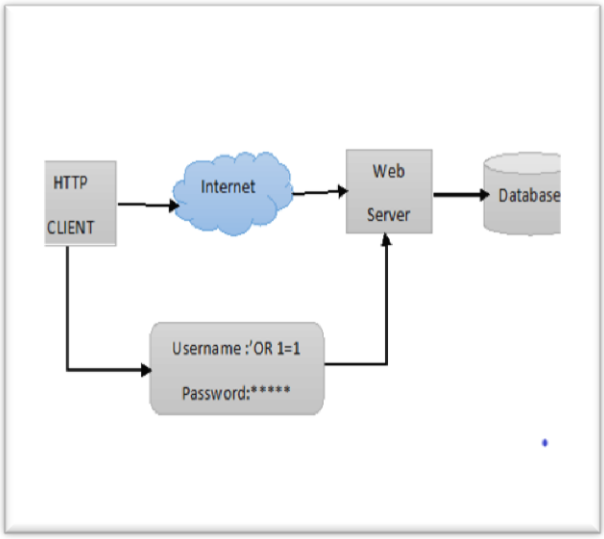
SQL Injection is one of the top-most vulnerabilities in OWASP Top-Ten Website Vulnerability as compared to rest of the website flaws. According to the survey conducted, it has been observed that from the year 2017 to 2019 SQL Injection has mostly taken place than rest of the website Attacks.



This attack takes advantage of un-sanitized input flaws to pass SQL command through web application for execution by the database. This is a basic attack where the attacker gains unauthorized access to the database or to access the confidential files directly from the database. It is not a web server or database flaw, but is an issue with the web applications. Using SQL Injection following types of attacks can be implemented: -

- **Authentication Bypass: -** Gaining unauthorized access and later achieving administrative privileges to the web application without providing valid username and password.
- **Information Disclosure:** - Confidential information can be obtained that is stored in the database.
- **Compromised Data Integrity:** - Webpage Defacement, inserting malicious content into webpages or altering contents of a database can be achieved from this kind of attack.
- **Compromised availability of Data:** - This attack can be implemented to delete database information, logs, and audit information that is stored in the database.
- **Remote Code Execution:** - This attack can be used to compromise the host Operating System.

WORKING OF SQL INJECTION ATTACK: -



SQL Injection consists of various types.

2. BACKGROUND STUDY: -

SQL Injection vulnerability has been known for twenty-one (21) years and still is still the most dangerous vulnerability. This vulnerability was first discovered "Rain Forest Puppy" pseudonym of Jeff Forristal in December 1998 in the issue of Phrack Magazine that described a Microsoft SQL Server that retrieved sensitive data through the use of commands in normal user inputs like 'name' and 'phone number'.

3. TYPES OF SQL INJECTION ATTACKS: -

There are 2 main types of SQL Injection Attacks, viz. Classical SQL Injection and Modern SQL Injection. Modern SQL Injection is considered to be more dangerous. Different types of SQL Injections are discussed below.

A. Classical SQL Injection: -

Various types of Classical SQL Injections are as follows: -

- **Piggy Backed Queries: -**

Purpose of Attack: - Information Retrieval, Denial of Service

In this form of attack, the attacker (or Hacker) tries to "piggy back" the query with the original query in the input field which is present in the web application. The word piggy-backed indicates as "on or as if on the back of another". During the execution of the legitimate query, the second malicious or harmful query also gets executed immediately after the first query to perform the injection attack. This is known as menacing (threatening) attack as it fully exploits the database.

- **Stored Procedures: -**

Purpose of Attack: - Authentication Bypass, Denial of Service

In this form of attack, any malicious user can input any malicious data in the username and password fields. The simple command that is entered by the malicious user can corrupt the database and can lead to disruption of services.

- **Union Query: -**

This form of attack uses the union operator (UNION) when inserting the SQL Query. Two or more SQL queries are joined together with the Union Operator. The first query is a normal SQL query which is appended by the malicious SQL statement by the Union Operator. Using this operator, the attacker can bypass the prevention and detection mechanism of the system. The attacker can add "--"to make the database ignore the condition followed by it as this becomes the comment for the SQL Parser.

- **Alternative Encoding: -**

In this form of attack, the attacker changes the SQL Injection pattern so that it goes undetected from common detection and prevention techniques. The attacker makes use of Hexadecimal, Unicode, ASCII code and Octal code in SQL statement to make it undetectable.

B. Advanced SQL Injection: -

Advanced SQL Injections are of various types.

- **Blind SQL Injection Attack: -**

The attacker uses this kind of SQL Injection attack when the web application is vulnerable to the SQL Injection but the error messages are not visible to the attacker. The Blind SQL Injection is used when the attacker tries to exploit the web application and instead of displaying the SQL error message, custom page is displayed. A True or False condition is posed to the database by an attacker to check if the web application is vulnerable to SQL Injection attack.

- **Compounded SQL Injection Attack: -**

Due to rapid increase of detection and prevention method of SQL Injection Attack, attackers developed a technique called Compounded SQL Injection Attack. In this form of attack, there exist a mixture of SQL Injection Attack and other Web Application Attacks which can be detailed as below.

(1) SQL Injection+DDoS Attack: -

DDoS (Distributed Denial of Service) Attack is used by an attacker in order to hang the server and to make the legitimate user unable to access the resources of the Web Application. The greater the number of rows and columns in the database, more the chances for the database to be vulnerable to SQL DDoS Attack.

(2) SQL Injection+DNS Hijacking: -

Data Transfer using Blind SQL Injection Attack is usually slower. Hence the attacker introduced another method i.e. DNS Hijacking in which the attacker modifies the DNS entries by exploiting the Domain Registers. This method is much faster and less noisy as compared to Blind SQL Injection Attack. This Attack can be conducted in two ways; first by injecting the malware or DNS Trojan software in the system or workstation; and second by hacking and modifying a particular website so that the user gets redirected to a fake website.

(3) SQL Injection+XSS Attack: -

In this type of attack, SQL Injection is the medium for setting up the attack, the rest of the attack is performed using XSS (Cross Site Scripting) Attack. They are also known as the "Third Wave Attacks" as they are not the same as the old way of attacks; instead they are the commands to hide from Network Monitoring Devices. These types of attacks are mainly used for data extraction.

(4) SQL Injection+Insufficient Authentication: -

The compounded attack is followed by Insufficient Authentication where the user or the administrator of the site is a novice. The security policy has not been properly implemented where the application fails to identify the user's location, services or application. This type of website can allow an attacker to bypass the authentication and stealing the confidential data without actually verifying the identity of the user. To make this attack successful, first identify whether the website has insufficient authentication vulnerability present and if it does have, SQL Injection Attack can be implemented.

(5) SQL Injection Attack using Cross Domain Policies of Rich Internet Application: -

Cross-Domain Policy is an XML file which permits web client to handle data in multiple domains and is used to define the list of RIA hosting domains that are allowed to collect or retrieve information from Content Providers Domain. Almost every web application use Adobe Flash Player and Microsoft Silverlight for improving and increasing the user interaction with the websites. The use of such applications is vulnerable to SQL Injection and other kinds of attacks as proper care has not been taken while programming the code. Improper use and misinterpretation of Cross-Domain Policy can give rise to vulnerability in Rich Internet Application. When the legitimate website was being attacked by Asprox Injection String, Firefox or Internet Explorer and Javascript code was used for SQL Injection Attack.

4. EVASION USING DIFFERENT TYPES OF SQL INJECTION AND ITS MITIGATION: -

We have discussed above the various types of SQL Injection Attacks. Below are the methods on how the attackers evade using these techniques.

A. Classical SQL Injection: -

(1) Piggy Backed Queries: -

EVASION TECHNIQUE: -

select emp_info from employee where uid=" admin" AND pass='123'**; DELETE from employee where empname='Austin';**

On executing the first query the interpreter notices the semicolon (;) and begin executing the second query with the first statement. The second query is malicious as it will delete all the data of the employee named "Austin".

MITIGATION: -

These attacks can be mitigated by determining the legitimate SQL Statement with the help of correct validation or to introduce new detection techniques. These kinds of attack can be mitigated using Static Analysis.

(2) Stored Procedure: -

EVASION TECHNIQUE: -

create procedure emp_info @username varchar2 @pass varchar2 @emp_id int AS BEGIN EXEC ('SELECT employee_info from employee_table where username=' "+@username " ' and password=' "+@pass " ' GO

The mentioned code is vulnerable to SQL Injection Attack as a malicious user can enter the malicious data in the input fields of Username and Password. Command entered by the user can destroy whole database and can also lead to service destruction.

MITIGATION: -

To prevent the data of the user from being compromised it is advised not to store any sensitive information in stored procedures, as it does not have the most important security features.

(3) Union Query: -

EVASION TECHNIQUE: -

Select * from employee where id='100' UNION select * from payment_details where user='admin'--'and password='pass'

The example above shows two queries that is being executed. The first query is a normal query. The second query is malicious and the text following (--) which becomes a comment for the SQL Parser. The attacker takes an advantage of this and performs an attack on the web application or website.

MITIGATION: -

Input Validation.

(4) Alternative Encoding: -

EVASION TECHNIQUE: -

select * from employee where uname="and pass=";exec(char(0x73687574646j776e))'

In the above command, the char () function and ASCII hexadecimal encoding have been used. The char () function will return the actual characters of the hexadecimal character encoding. The encoded string will be translated to the "shutdown" command and it will be executed by the database.

MITIGATION: -

1) Whenever one requires the data from the web-based forms, ensure that you use POST () instead of GET () as the GET () appends the data with the URL which is not a good security practice.
2) Whenever the data is submitted to the server always ensure that some amount of the data must only be accepted at the particular interval and it must fall under the specified bounds (maximum and minimum lengths) and also contains specified contents.

B. Advanced SQL Injection: -

(1) Blind SQL Injection: -

'; if (select count(username) from users where username = 'Administrator' AND substring (password, 1, 1) > 'p') = 1 wait for delay '0:0: {delay}'—

The above command retrieves the data the way it is described, by systematically testing one character at a time.

MITIGATION: -

1) The countermeasure of this kind of attack is adopt secure coding practices, that is independent of the language.
2) Avoid the use of Dynamic SQL.
3) Usage of Stored Procedures.
4) Usage of Vulnerability Scanner and performing regular scan is recommended.

(2) Compounded SQL Injection Attack: -

(A) SQL Injection+DDoS Attack: -

select table1 from (select decode(encode(convert(compress(post) using latin1), concat (post, post, post, post)), sha1(concat(post, post, post,post))) as table1 from table_1)a--

The above-mentioned code is used to perform the DDoS Attack on the website.

If we know only particular column is only vulnerable then we will try to inject the code to those columns which is found to be vulnerable. Let us consider that the 3rd column is vulnerable and we will inject the payload to the website as follows.

http://certifiedhacker.com/hack.php?id=1' union select 1,2,table1,4 from (select decode(encode(convert(compress(post) using latin1, des_encrypt(concat(post,post,post,post),8)),des_encrypt(sha1(concat(post,post,post,post)),9)) as table1 from table_1)a--

MITIGATION: -

The Cluster Analysis Methodology is used to identify the DDoS Attack that is being performed on the website and this can easily identify the type of attack on the system.

(B) SQL Injection+DNS Hijacking: -

do_dns_lookup ((select top 1 password from employee) +'. inse6140.net');

The select statement is used to retrieve the password hash that interests the hacker or attacker followed by the domain name('inse6140.net') that is to be controlled. This is done with the help of DNS Hijacking. DNS lookup has to be performed at last. The packet sniffer has to be run on the name server for the domain and wait for the DNS Record containing our hash.

MITIGATION: -

1) DNS Hijacking has been prevented by not downloading the free utilities from the websites as they mostly contain vulnerabilities.
2) DNS Rebinding tries to capture the router settings of the client or user.
3) DNS Hijacking can be prevented by the use of Session Shield i.e. Light Weight Client-Side Protection Mechanism.
4) Sqlmap is used to protect against SQL Injection+DNS Attack and has the feature of the DNS exfiltration and many other command-line designed for DNS prevention and detection.

(C) SQL Injection+XSS Attack: -

print "<html>"

print "<h1>Most recent comment</h1>"

print database.latestComment

<iframe src=http://evil.com/xss.html>

print "</html>"

The above command will get executed and the XSS Attack is performed with the MySQL Injection. The script above will try to connect to the database of the website, hence it's a difficult and complicated task. If the connection to the database is successful the attacker or a hacker will have full access to the database but through the client-side language.

MITIGATION: -

1) Ardilla Tool uses Taint Based Approaches and Static Analysis Techniques for filters and sanitization method for detection of vulnerability.
2) Noxes Tool can also be used for the detection and prevention of XSS and SQL Injection Attack.
3) Cookie Stealing through XSS for SQL Injection Attack can be prevented by implementing Dynamic Cookie Rewriting Technique.

(D) SQL Injection Attack using Cross Domain Policies of Rich Internet Application: -

<allow-access-from domain="*.sub1.domainX.com"/>

<allow-access-from domain="*.domainY.com"/>

<allow-access-from domain="*"/>

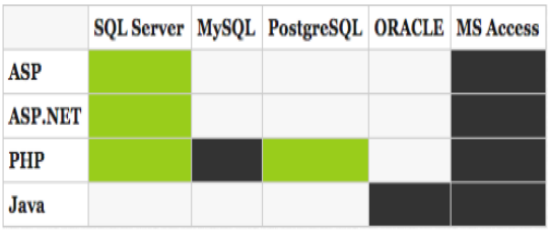
The above code is an example of Weak Code for the Cross-Domain Policy.

MITIGATION: -

1) FLASHOVER which uses static as well as dynamic code analysis to protect SQL Injection Cross Domain Policies.
2) DEMACRO which detects the malicious cross domain requests and tries to de-authenticate them.

LANGUAGES / DATABASE STACKED QUERY TABLE: -

Green : Supported; **Dark Gray:** Not Supported; **Light Gray:** Unknown



Pattern	String Pattern	Expected Results	
		Secure	Insecure
1	'OR" ='	Login Failed	Login Successful
2	0'or'1'='1	Login Failed	Login Successful
3	1'or'1'='1	Login Failed	Login Successful
4	'OR'1'='1'	Login Failed	Login Successful
5	"'or 1=1--"	Login Failed	Login Successful
6	OR'1'='1'"	Login Failed	Login Successful
7	Emp_id='x'AND emp_name IS NULL	Attack identified	Columns retrieved
8	SELECT * FROM emp; DROP TABLE emp;	Attack Identified	Table employee deleted
9	SELECT TABLE_NAME FROM INFORMATION_SCHEMA TABLE	Attack Identified	Table name in the database

5. EXPERIMENTAL RESULTS: -

We have performed basic SQL Injection queries proposed by us on unprotected interface. Below mentioned table shows us the result of the analysis we have performed.

6. CONCLUSION: -

In our paper, we have tried to explain various kinds of SQL Injection Attacks. They are the typical form of attacks which are done on web applications or websites. These kinds of attack are quite complex to understand. We have discussed detection and prevention techniques which are very limited as very smaller number of researches are carried out on this attack because it overcomes previous prevention and detection methods. Even the proper coding of the web applications or websites can be overcome easily. The attacker or hacker who have good knowledge of these kinds of attacks can destroy the web application which in turn will impact the business of an organization.

7. ACKNOWLEDGEMENT: -

Through this research paper, I have shared enough knowledge on how a security flaws in the design implementation of database could compromise the sensitive data of the customers.

I would like to thank my professors to help explore and write a research paper on this particular topic.

I would like to give my gratitude to those who devoted their precious time to help me complete my research without whom it would not have been possible.

8. REFERENCES: -

1. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
2. https://portswigger.net/web-security/sql-injection
3. CEH v10 Guide Book