https://www.acunetix.com/websitesecurity/sql-injection/

https://www.imperva.com/learn/application-security/sql-injection-sqli/

https://www.guru99.com/learn-sql-injection-with-practical-example.html

https://outpost24.com/blog/SQL-injections-cyberattacks

https://plagiarismdetector.net/

------------------------------------------------------------------------------------------------------------------------------------------

The attacker poisons dynamic SQL statements in the SQL Injection Attack to comment on some components of the declaration or to add a condition that will always be valid. The attacker uses the design faults to exploit SQL statements by implementing malicious SQL code is poorly designed web applications.

Usually, SQL injection happens when input is taken from a user, such as their username, and in such fields, the user will provide a SQL statement instead of a name id that you will run on your database unknowingly. The attacker will execute malicious SQL statements in SQL Injection (SQLI) attack and attempts to control a database server behind a web application. And SQL Injection vulnerabilities can also be used to bypass security measures of application.

Vulnerability like SQLI can influence any website or web application using a SQL database such as MySQL, Oracle, SQL Server, or others. SQLI is a common attack vector that uses malicious SQL code for backend database manipulation to access confidential and sen0sitive information like customer information, personal data, trade secrets, intellectual property, and more.SQL Injection attack are one of the oldest, the most common and dangerous Vulnerability in the Web application.

An attacker must first discover vulnerable user inputs within the web page or web application to perform a SQL Injection attack and such user input is used as a target to pass an SQL query to a web page or web application for performing SQL Injection attack. SQL injection also termed SQLI. Input content can be created by the attacker. Such content is often referred to as a malicious payload and is the main component of the attack. After the attacker sends this content, the database executes malicious SQL commands. The malicious queries can be inserted by the attacker via a web form or by attaching them directly to the end of the URL or HTTP headers.

SQL is a query language for managing data stored in relational databases. And it can be used to access, edit, and delete data. Many websites and web applications manage all the data in SQL databases. You can also use SQL commands to execute operating system commands in some instances. An effective SQL Injection attack can, therefore, have very severe implications like [2].

Attackers can use SQL Injection to identify other user's credentials in the database. These users can then be impersonated by the attacker. The impersonated user can be an administrator with all the privileges of the database also [2].

SQL allows you to select and display information in the database. An SQL Injection vulnerability could give the attacker full access to all information on a database server. [2]

SQL also allows you to change information and add new information to a database. For instance, an attacker could use SQL Injection in a financial application to change balance, void transactions, or transfer cash to their account[2].

To delete documents from a database, you can use SQL, even to drop tables also. Even if database backups are made by the administrator, data deletion could influence the accessibility of the application until the database is restored. Backups may not also contain the latest information [2].

In some database servers, you can use the database server to access the working system. This may be accidental or deliberate. In such a case, an attacker might use an initial vector of SQL Injection and then attack the internal network behind a firewall [2].

------------------------------------------------------------------------------------------------------------------------------------------

Types of SQL injection

In-band SQLi

The attacker utilizes the same communication channel to launch their assaults and collect outcomes. The simplicity and effectiveness of In-band SQLI make it one of the most popular SQLi attack kinds. This technique has two sub-variations [3]:

Error-based SQLi— The attacker executes activities that cause error messages to be generated by the database. The attacker may use the data supplied by the error messages to collect information about the database structure [3].

Union-based SQLi—This method uses the UNION SQL operator to fuse various select statements generated by the database to obtain a single HTTP response. This result may include information that the attacker can leverage [3].

Inferential (Blind) SQLi

The attacker sends payloads to the server and observes the server's response and behavior to know more about its database structure Because the data is not transferred from the website database to the attacker machine, this method is called blind SQLi, so the attacker can not see much information after an attack in in-band attack[3]

Blind SQL injections depend on the server's response and behavior patterns, to perform these types of attacks typically consume time but can be just as damaging. The following can be categorized as blind SQL injection:

Boolean— that attacker sends a SQL query to the database prompting the application to return the result. The result depends on whether the request is true or false. Based on the result, the HTTP response data will change or remain unchanged. The attacker can then work out if a true or false outcome has been produced by the message.

Time-based—the attacker will send a SQL request to the database, which will cause the database to wait (in seconds for a period) before it responds. From the time the database takes to respond, the attacker can see if a request is true or false. An HTTP response will be produced immediately or after a waiting period based on the result. Thus, if the message they used returned true or false, the attacker can work out without depending on database information [3].

Out-of-band SQLi

Only when certain features are enabled on the database server used by the web application the attacker can perform this type of attack. This type of attack is used mainly as an alternative to the SQLi methods in-band and inferential.

Out - of-band SQLi is conducted if the attacker is unable to use the same channel to start the attack and collect data, or if a server is too slow or unstable to perform such activities. These methods rely on the server's ability to generate DNS or HTTP requests for information transfer to an attacker [3].

---------------------------------------------------------------------------------------------------------------------

Example:

Username = request.post ['username']

password = request.post ['password']

// Statement vulnerable to SQL injection

SQL = "SELECT userid FROM users WHERE username='" + username + "' AND password='" + password + "'"

// execute statements

db.exec(SQL)

The above example is vulnerable to SQL injection because the database server will interpret as a command whatever the user enters in the form. For example, by setting the password field to ' or 1=1, an attacker could bypass this form. The following looks like a SQL statement.

The following is what the SQL statement would look like.

SELECT id FROM users WHERE username='foo' AND password='pass' OR 1=1

From the above statement, we can see that the user's input has changed the statement's functionality. Now, the value of the ID column is being returned if the submitted username is equal to foo, and the password is equal to pass, or if 1is equal to 1 (which will always be the case).

With this statement, only the username has to match the value in the database since the password condition can either match the value in the database or validate it if 1=1. With this trick, for any customer whose username is known, the intruder can bypass the authentication system of the website.

An intruder may even comment on the remainder of the declaration to further regulate the SQL declaration. An intruder can, for instance, use the double-dash (--) notation to comment on the rest of the declaration:

SELECT id FROM users WHERE username='username' --' AND password=bar'

The highlighted portion of the above declaration, or after the double-dash, will be pointed out and thus not regarded during execution. This will allow an attacker to bypass authentication once again

--------------------------------------------------------------------------------------------------------------------

How to Prevent against SQL Injection Attacks

Do Not use dynamic SQL Quires

Avoid placing user-provided input directly into SQL statements. Prefer prepared statements and parameterized queries[1], which are much safer. Stored procedures are also usually safer than dynamic SQL.

Sanitize user-provided inputs

Properly escape the characters and should Verify that the type of data submitted matches the type expected.

Don't leave sensitive data in plaintext: Encrypt private/confidential data stored in a database. This also provides a further level of protection if the attacker successfully enters into the system.

Restrict the rights and privileges of the database by Reducing the user's capabilities to the bare minimum. This will restrict what an intruder or attacker can do if they succeed in gaining access.

Avoid displaying common database errors that help Attackers obtain information to execute additional attacks on the database so that we need to show custom error messages instead of standard error messages.

Use a Web Application Firewall (WAF) for web applications that access databases

This protects web-based applications. It can help to identify SQL injection attempts. It can also help prevent SQL injection attempts from reaching the application (and therefore the database) based on the configuration.

[1] SQL injection prevention cheat sheet"https://www.Index.php/SQL\_InjectionPreventionCheatSheet DefenseOption1:PreparedStatements.28withParameterizedQueries.29".

[2] What is SQL injection and how to prevent it "https://www.acunetix.com/websitesecurity/sql-injection/?"

[3] What is SQL injection "https://www.imperva.com/learn/application-security/sql-injection-sqli/?"

[4] SQL injection tutorials" https://www.guru99.com/learn-sql-injection-with-practical-example.html"