Name: Avaneesh S
ID: RS_1802

SQL Injection

What is SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Some of The Most Important SQL Commands

- `SELECT` - extracts data from a database
- `UPDATE` - updates data in a database
- `DELETE` - deletes data from a database
- `INSERT INTO` - inserts new data into a database
- `CREATE DATABASE` - creates a new database
- `ALTER DATABASE` - modifies a database
- `CREATE TABLE` - creates a new table
- `ALTER TABLE` - modifies a table
- `DROP TABLE` - deletes a table
- `CREATE INDEX` - creates an index
- `DROP INDEX` - deletes an index

What is SQL Injection?

SQL injection is a technique through which attackers can execute their own malicious SQL statements generally referred to as a malicious payload. Through the malicious SQL statements, attackers can steal information from the victim's database; even worse, they may be able to make changes to the database. Our employee management web application has SQL injection vulnerabilities, which mimic the mistakes frequently made by developers.

Applications will often need dynamic SQL queries to be able to display content based on different conditions set by the user. To allow for dynamic SQL queries, developers often concatenate user input directly into the SQL statement. Without checks on the received input, string concatenation becomes the most common mistake that leads to SQL injection vulnerability. Without input sensitization, the user can make the database interpret the user input as a SQL statement instead of as data. In other words, the attacker must have access to a parameter that they can control, which goes into the SQL statement. With control of a parameter, the attacker can inject a malicious query, which will be executed by the database. If the application does not sanitize the given input from the attacker-controlled parameter, the query will be vulnerable to SQL injection attack.

Types of SQL Injections

SQL injections typically fall under three categories: In-band SQLi (Classic), Inferential SQLi (Blind) and Out-of-band SQLi. You can classify SQL injections types based on the methods they use to access backend data and their damage potential.

In-band SQLi

The attacker uses the same channel of communication to launch their attacks and to gather their results. In-band SQLi's simplicity and efficiency make it one of the most common types of SQLi attack. There are two sub-variations of this method:

- **Error-based SQLi**—the attacker performs actions that cause the database to produce error messages. The attacker can potentially use the data provided by these error messages to gather information about the structure of the database.

Eg: https://tryhackme.com/room/id=1'

- **Union-based SQLi**—this technique takes advantage of the UNION SQL operator, which fuses multiple select statements generated by the database to get a single HTTP response. This response may contain data that can be leveraged by the attacker.

Eg: https://tryhackme.com/room/id=1 UNION SELECT 1,2,3

Inferential (Blind) SQLi

The attacker sends data payloads to the server and observes the response and behavior of the server to learn more about its structure. This method is called blind SQLi because the data is not transferred from the website database to the attacker, thus the attacker cannot see information about the attack in-band.

Blind SQL injections rely on the response and behavioral patterns of the server so they are typically slower to execute but may be just as harmful. Blind SQL injections can be classified as follows:

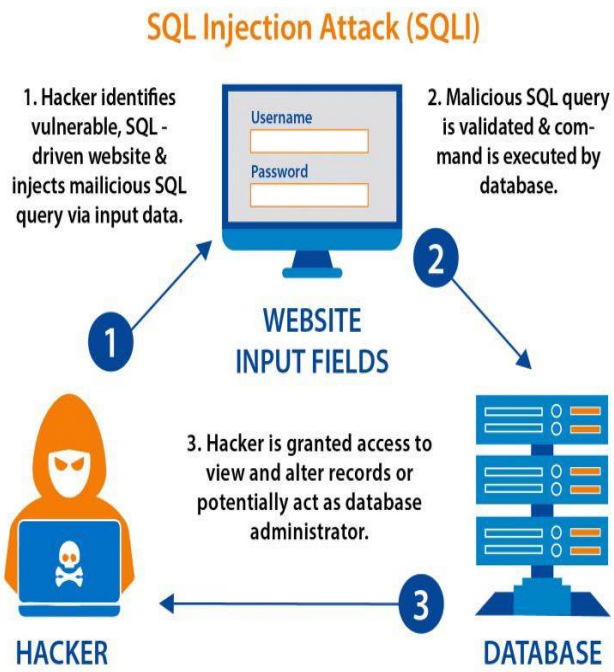
- **Boolean**—that attacker sends a SQL query to the database prompting the application to return a result. The result will vary depending on whether the query is true or false. Based on the result, the information within the HTTP response will modify or stay unchanged. The attacker can then work out if the message generated a true or false result.
- Eg: https://tryhackme.com/room/username=admin123' UNION SELECT 1,2,3 from users where username like 'a%
- **Time-based**—attacker sends a SQL query to the database, which makes the database wait (for a period in seconds) before it can react. The attacker can see from the time the database takes to respond, whether a query is true or false. Based on the result, an HTTP response will be generated instantly or after a waiting period. The attacker can thus work out if the message they used returned true or false, without relying on data from the database.
- Eg: https://tryhackme.com/room/username=admin123' UNION SELECT sleep(2),2,3 from users where username like 'a%

Out-of-band SQLi

The attacker can only carry out this form of attack when certain features are enabled on the database server used by the web application. This form of attack is primarily used as an alternative to the in-band and inferential SQLi techniques.

Out-of-band SQLi is performed when the attacker can't use the same channel to launch the attack and gather information, or when a server is too slow or unstable for these actions to be performed. These techniques count on the capacity of the server to create DNS or HTTP requests to transfer data to an attacker.

Eg: The payload contains a request which forces an HTTP request back to the hacker's machine containing data from the database.



SQLI prevention and mitigation

There are several effective ways to prevent SQLI attacks from taking place, as well as protecting against them, should they occur.

The first step is input validation (a.k.a. sanitization), which is the practice of writing code that can identify illegitimate user inputs.

While input validation should always be considered best practice, it is rarely a foolproof solution. The reality is that, in most cases, it is simply not feasible to map out all legal and illegal inputs—at least not without causing a large number of false positives, which interfere with user experience and an application's functionality.

For this reason, a web application firewall (WAF) is commonly employed to filter out SQLI, as well as other online threats. To do so, a WAF typically relies on a large, and constantly updated, list of meticulously crafted signatures that allow it to surgically weed out malicious SQL queries. Usually, such a list holds signatures to address specific attack vectors and is regularly patched to introduce blocking rules for newly discovered vulnerabilities.

Modern web application firewalls are also often integrated with other security solutions. From these, a WAF can receive additional information that further augments its security capabilities.For example, a web application firewall that encounters a suspicious, but not outright malicious input

may cross-verify it with IP data before deciding to block the request. It only blocks the input if the IP itself has a bad reputational history.

References:

https://www.w3schools.com/sql

https://www.tutorialspoint.com/sql/sql-overview.htm

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

https://tryhackme.com/room/sqlinjectionlm

https://infosecwriteups.com/sql-injection-lab-tryhackme-writeup-fcf30f846e82

https://portswigger.net/web-security/sql-injection