SQL injection

#Session_Task_13 by Sukhveer singh

1. SQL –

SQL stands for Structured Query Language. It's a programming language used to manage and manipulate databases. Think of it as a way to communicate with databases to perform tasks like storing, retrieving, updating, and deleting data.

2. Example –

Let's say you have a table in a database called "Students" that stores information about students such as their names, ages, and grades.
To retrieve the names of all students from the table, you would write a SQL query like this:

```
SELECT name FROM Students;
```

This query tells the database to select the "name" column from the "Students" table. After running this query, you would get a list of all the student names stored in the database.

Similarly, you can use SQL to perform various other operations like adding new data, updating existing data, and deleting data from tables in the database. It's a powerful tool for managing and organizing large amounts of information.

3. Most important commands in sql –

I. **SELECT:** - This command is used to retrieve data from a database. You specify which columns you want to retrieve data from, and optionally, you can specify conditions to filter the rows returned.

Example:

SELECT * FROM table_name

Or

SELECT column1, column2 FROM table_name WHERE condition;

II. **INSERT INTO:** This command is used to add new rows of data into a table in the database.

INSERT INTO table_name (column1, column2) VALUES (value1, value2);

III. **UPDATE:** This command is used to modify existing data in a table.

UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;

IV. **DELETE:** This command is used to remove rows from a table based on specified conditions.

DELETE FROM table_name WHERE condition;

V. **CREATE TABLE:** This command is used to create a new table in the database. You specify the table name and define the columns along with their data types.

CREATE TABLE table_name (

```
    column1 datatype,
    column2 datatype,
        ...
);
```

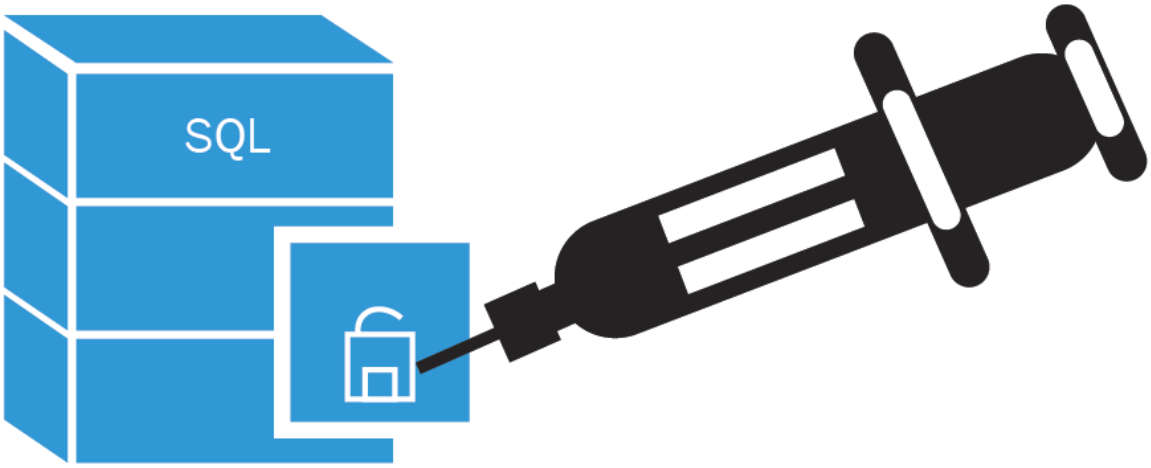
VI. **ALTER TABLE:** This command is used to modify an existing table in the database. You can add, modify, or delete columns in an existing table.

```
ALTER TABLE table_name ADD column_name datatype;
```

VII. **DROP TABLE:** This command is used to delete an entire table along with all its data from the database.

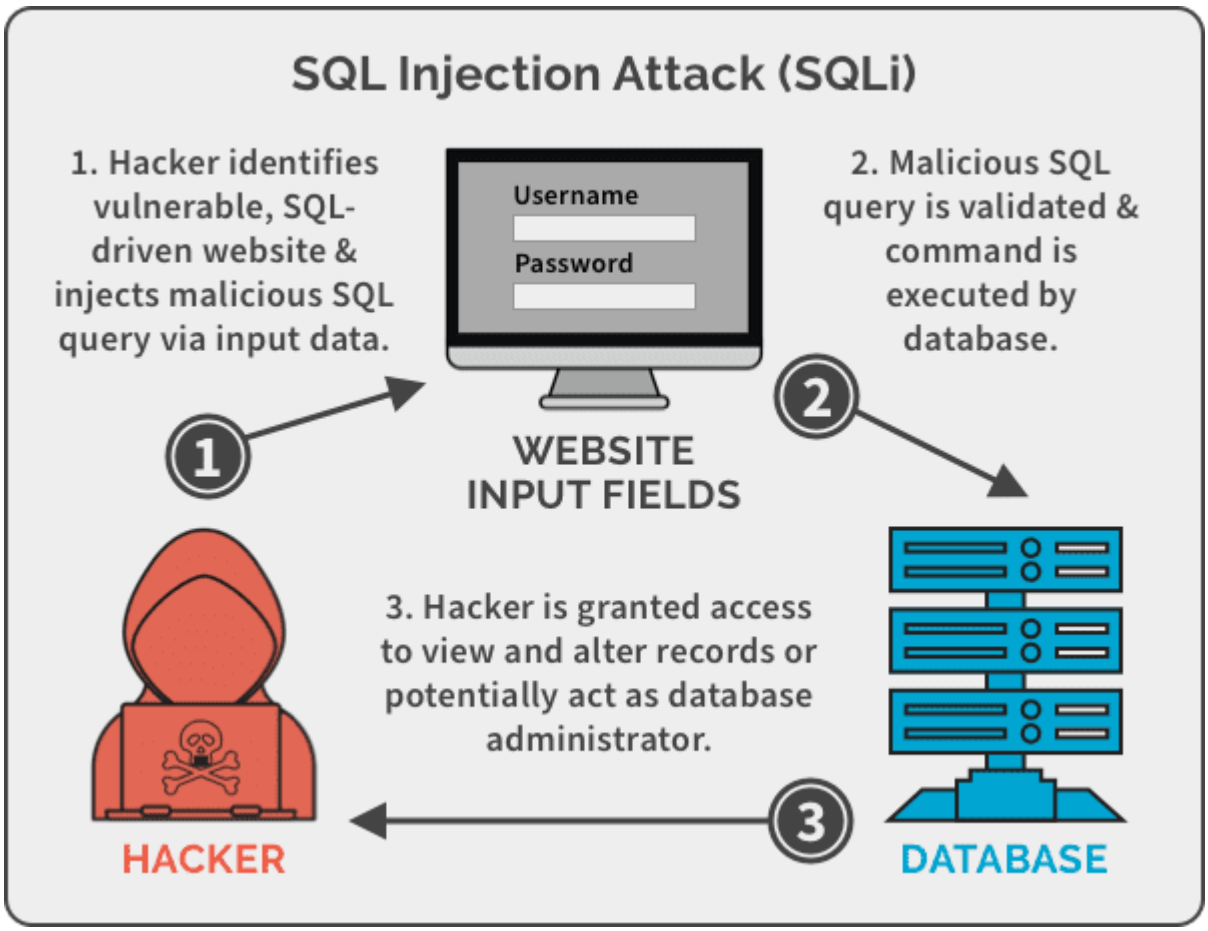
```
DROP TABLE table_name;
```

4. Sql injection –



SQL Injection Attack is a type of cyber attack that exploits vulnerabilities in a website or application's input forms to inject malicious SQL code into the backend database. This attack occurs when an attacker inserts SQL commands into input fields, such as login forms or search queries, with the intention of manipulating the database or gaining unauthorized access to sensitive information.

5. How it works: -



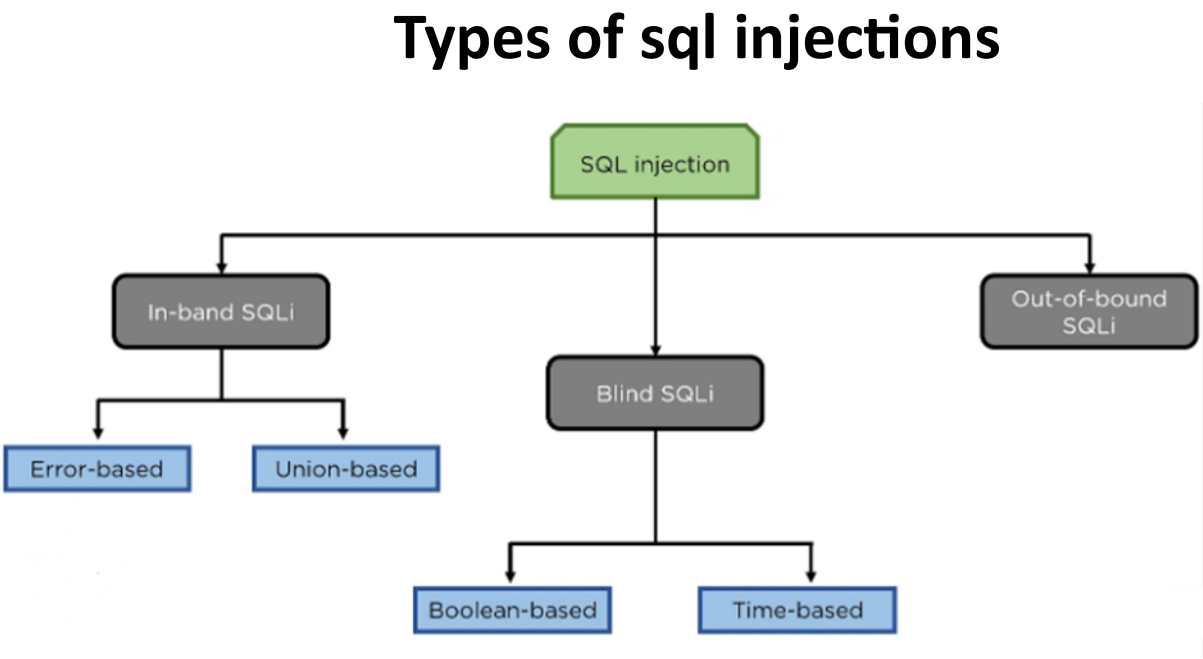
- **Injection Point:** The attacker identifies input fields on a website or application where user input is directly included in SQL queries without proper validation or sanitization.

- **Malicious Payload:** The attacker then inserts malicious SQL commands into these input fields. For example, they might enter 'OR 1=1' or '; DROP TABLE users;' into a login form.

- **Execution:** When the application processes the input, it doesn't differentiate between legitimate user input and the injected SQL

commands. As a result, the injected SQL code is executed by the database server.

- **Impact:** Depending on the attacker's intentions, the consequences of a successful SQL injection attack can vary. They could retrieve sensitive data from the database, modify or delete data, escalate privileges, or even take control of the entire system.



6. In-band SQL Injection: 2 types –

1) Error-Based SQL Injection: -

Error-based SQL injection occurs when the attacker exploits error messages generated by the database server to extract information from the database.

The attacker deliberately injects SQL queries that trigger error messages containing valuable details about the database structure or data.

- **Example: -**

  Let's say there's a vulnerable login form on a website. When a user enters their username and password, the website runs a SQL query to check if the provided credentials match any in the database. An attacker might enter a malicious input like ' OR 1=1; -- in the username field. This input could cause the SQL query to become syntactically incorrect and trigger an error. The error message returned by the server might reveal useful information, such as table names or column names, helping the attacker further exploit the vulnerability.

2) **Union-Based SQL Injection: -**

Union-based SQL injection involves the attacker using the UNION operator to combine the results of two or more SELECT statements into a single result set.

The attacker crafts SQL queries that retrieve data from other tables in the database and return the results alongside the original query results.

- **Example: -**

  The attacker may input something like ' UNION SELECT username, password FROM admin_users; -- in the username field. This input manipulates the SQL query to retrieve usernames and passwords from the "admin_users" table (assuming such a table exists). By combining the results with the original query, the attacker can potentially gain unauthorized access to the system or extract sensitive information.

7. Inferential SQL Injection:2 types : -

- **Boolean-Based SQL Injection:**

  Attacker infers database information by observing changes in application behavior when injecting conditions that are either true or false.

  Example: Injecting apple' AND 1=1; -- into a search query and noting differences in response time or displayed content.

- **Time-Based SQL Injection:**

  Attacker infers database information by introducing time delays in injected queries and observing variations in application response time.

  Example: Injecting apple' AND SLEEP(5); -- into a search query and noting the delay in the application's response.

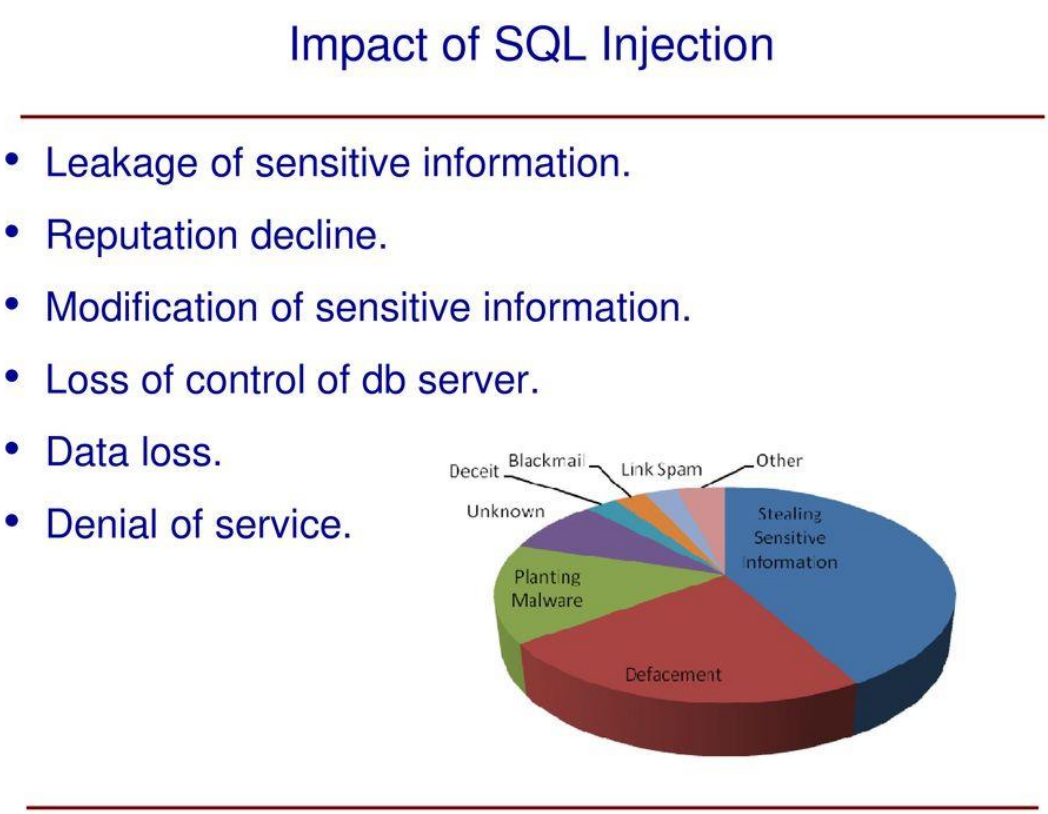
8. Out-Band SQL injection –

Out-of-band SQL Injection is a type of SQL injection attack where the attacker does not directly receive the results of the injected SQL queries. Instead, the attacker uses an alternative communication channel to extract data from the database. This could involve techniques such as making DNS or HTTP requests to an external server controlled by the attacker.

In simpler terms, in out-of-band SQL injection, the attacker doesn't immediately see the results of their actions within the application itself. Instead, they use other means to indirectly gather the information they're

after. This could involve techniques like triggering the application to send data to a server under the attacker's control.

9. Impact of SQL injection : -



i. **Data Leakage:** SQLi attacks can lead to unauthorized access to sensitive data stored in databases. Attackers can extract, modify, or delete sensitive information such as user credentials, personal details, financial records, and proprietary business data.

ii. **Data Manipulation:** Attackers can modify or delete data within the database, causing data integrity issues. This can result in incorrect or corrupted data, leading to operational disruptions, financial losses, and damage to an organization's reputation.

iii. **Application Compromise:** SQLi attacks can compromise the security of the entire application, allowing attackers to gain unauthorized access to administrative functionalities, upload malicious files, or execute arbitrary commands on the underlying server.

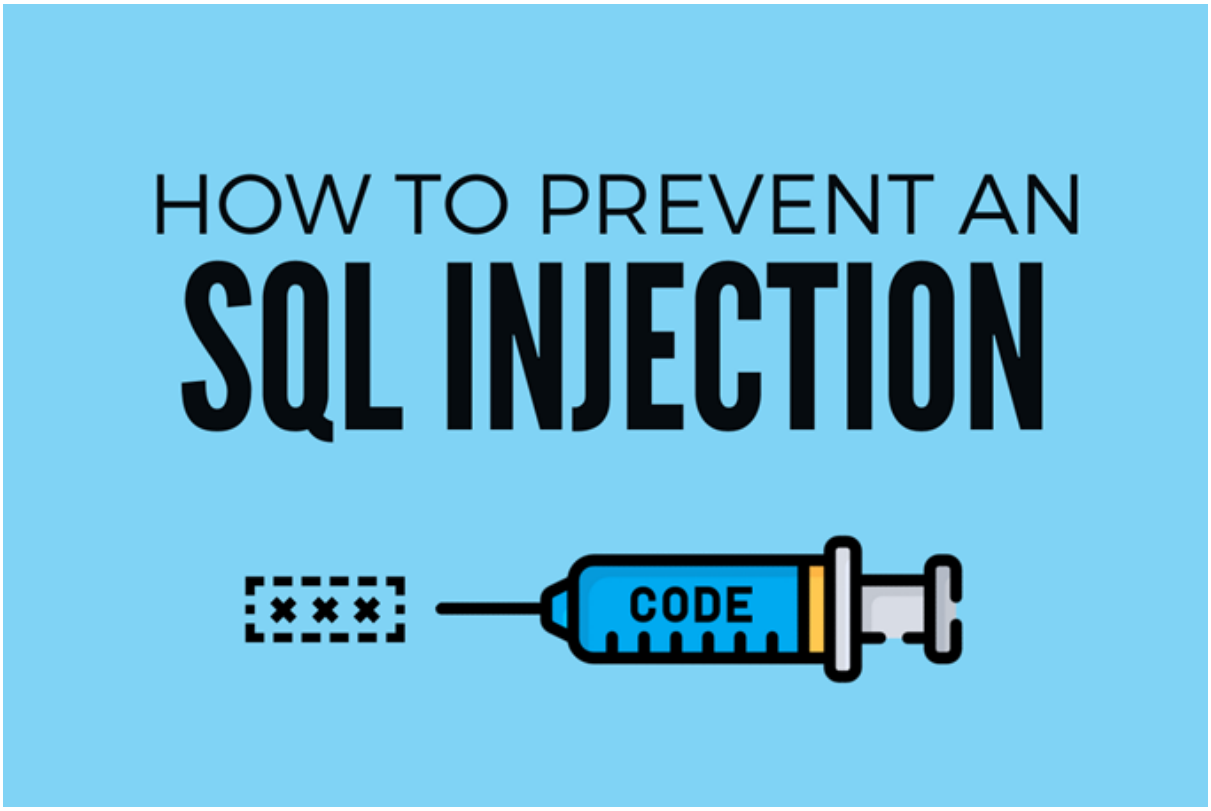
iv. **Financial Losses:** Organizations may suffer financial losses due to SQLi attacks, including costs associated with data recovery, regulatory fines for data breaches, legal expenses, and loss of business opportunities due to reputational damage.

v. **Reputation Damage**: SQLi attacks can tarnish an organization's reputation and erode trust among customers, partners, and stakeholders. News of a data breach resulting from SQLi can lead to loss of customers, decreased revenue, and long-term damage to brand image.

vi. **Regulatory Consequences:** Organizations may face regulatory consequences and legal liabilities for failing to protect sensitive data from SQLi attacks. This includes penalties, fines, and legal action under data protection laws such as GDPR, HIPAA, or PCI DSS.

vii. **Operational Disruption**: SQLi attacks can disrupt normal business operations by causing downtime, loss of access to critical systems, and delays in delivering products or services. This can result in financial losses and damage to customer relationships.

10. How to prevent SQL injection : -



i. **Input Validation and Sanitization:** Validate and sanitize all user-supplied input to ensure that it conforms to expected formats and does not contain malicious SQL code. Use server-side input validation routines and parameterized queries to prevent attackers from injecting SQL commands.

ii. **Use Prepared Statements and Parameterized Queries:** Utilize prepared statements and parameterized queries provided by database APIs instead of dynamically constructing SQL queries with user input. Parameterized queries separate SQL logic from data, preventing SQLi attacks.

iii. **Least Privilege Principle:** Assign the least privilege necessary to application accounts and database users. Restrict access rights and privileges to ensure that each user or application has access only to the data and functionality required to perform its intended tasks.

iv. **Secure Coding Practices:** Follow secure coding practices, such as avoiding dynamic SQL queries, using stored procedures, escaping special characters, and limiting the use of dynamic SQL generation methods.

v. **Database Hardening:** Implement security best practices for database management systems, such as keeping database software up-to-date with patches and security updates, disabling unnecessary database features and services, and enforcing strong authentication and access controls.

vi. **Web Application Firewalls (WAF):** Deploy WAFs to inspect and filter incoming web traffic for suspicious patterns and known SQLi attack signatures. Configure WAFs to block or log malicious requests and provide additional layers of defense against SQLi attacks.

vii. **Security Testing and Code Reviews:** Conduct regular security testing, vulnerability assessments, and code reviews to identify and remediate SQLi vulnerabilities in applications and databases. Use automated scanning tools and manual testing techniques to detect and mitigate security flaws.

viii. **Security Awareness Training:** Educate developers, administrators, and other personnel about the risks of SQL Injection attacks and best practices for secure coding, input validation, and data sanitization. Promote a culture of security awareness and encourage proactive threat mitigation efforts.

ix. **Monitoring and Logging:** Implement robust logging and monitoring mechanisms to track and analyze database activity, application behavior, and potential SQLi attack patterns. Monitor access logs, error logs, and database audit trails for signs of unauthorized access or suspicious activities.

x. **Incident Response Plan:** Develop and maintain an incident response plan to effectively respond to and mitigate SQL Injection attacks. Establish procedures for incident detection, containment, eradication, and recovery to minimize the impact of security breaches.

11. References –

I. https://www.w3schools.com/sql/sql_intro.asp
II. https://www.javatpoint.com/dbms-sql-command
III. https://www.crowdstrike.com/cybersecurity-101/sql-injection/
IV. https://www.geeksforgeeks.org/sql-injection/
V. https://www.acunetix.com/websitesecurity/sql-injection2/
VI. https://www.geeksforgeeks.org/types-of-sql-injection-sqli/
VII. https://www.acunetix.com/websitesecurity/sql-injection/#:~:text=How%20to%20Prevent%20an%20SQL,inputs%20such%20as%20login%20forms.
VIII. https://www.indusface.com/blog/how-to-stop-sql-injection/
IX. https://owasp.org/www-community/attacks/SQL_Injection#:~:text=SQL%20injection%20attacks%20allow%20attackers,administrators%20of%20the%20database%20server.