Project title:
SQL INJECTION ATTACK ON TODAY'S TECHNOLOGY USINNG MYSQL DATABASE QUEIRIES

Prepared by **Guided by**

Pratiksha Deshmukh **Zakir Hussai**

Index

1 . Sql Injection :

SQL Injection (SQLi) is a type of cyberattack where an attacker inserts malicious SQL code into a query that is executed by a database.
This vulnerability allows attackers to execute arbitrary SQL statements when user input is not properly verified or sanitized. Among the main goals of SQL Injection attacks are the following:

Extracting Sensitive Data: gaining access to personal data including credit card numbers, passwords, and usernames.
Modifying Data: Adding, removing, or changing database entries.
Acquiring unlawful Access: Increasing privileges to carry out commands at the system level or obtain unlawful access.Entering restricted regions of an online program by circumventing authentication.
Bypassing Authentication: Gaining access to restricted areas of a web application

SQL Injection attacks can occur through various inputs such as user forms, URL parameters, and cookies, and they can exploit vulnerabilities in both simple and complex SQL queries

Types of SQL Injection attacks:
1. Classic SQLi: Injecting malicious SQL code directly into a web application's database
2. Blind SQLi: Injecting malicious SQL code that returns no data, but still allows the attacker to infer information.
3. Time-based SQLi: Injecting malicious SQL code that takes advantage of the time it takes for the database to respond.
4. Out-of-band SQLi: Injecting malicious SQL code that sends data to an external server.

To prevent SQL Injection attacks, it's essential to:
1. Use prepared statements: Separate SQL code from user input.
2. Parameterize queries: Use parameters instead of concatenating user input.
3. Validate user input: Sanitize and validate user input to prevent malicious code.
4. Limit database privileges: Restrict database access to only necessary actions.

Remember, SQL Injection attacks can have severe consequences, including data breaches and unauthorized access to sensitive information

2.Databases used in project

- **User_Authentication**: Manages user credentials and authentication details.
- **Product_Information**: Stores information related to products, their categories, and reviews.

3.Tables Used in Each Database

User_Authentication

- **users**: Stores user information (id, username, password, email).
- **user_roles**: Contains role definitions for users (id, role_name, description).
- **login_attempts**: Records login attempts including success and failure (id, username, attempt_date, success).
- **user_sessions**: Tracks user session details (id, user_id, session_start, session_end).
- **password_resets**: Manages password reset requests (id, user_id, reset_date, reset_token).

Product_Information

- **products**: Holds product details (id, product_name, description, price).
- **categories**: Contains category information for products (id, category_name, description).
- **product_categories**: Maps products to their categories (id, product_id, category_id, category_type).
- **product_reviews**: Stores reviews for products (id, product_id, review_date, rating).
- **product_images**: Manages images associated with products (id, product_id, image_url).

Employee_Information

- **employees**: Contains employee details (id, employee_name, department, salary).
- **departments**: Lists department information (id, department_name, description).
- **employee_roles**: Defines employee roles (id, role_name, description)
- **Employee_Information**: Contains details about employees, their roles, and departments.

4. Queries Identified by the network infra security team

These queries are designed to test for potential SQL injection vulnerabilities:

- **Retrieve all user information**:

  SELECT * FROM users;

- **Retrieve product details along with their categories and reviews**:

  SELECT p.*, c.category_name,
  r.rating FROM products p JOIN product_categories pc
  ON p.id = pc.product_id JOIN categories c
  ON pc.category_id = c.id JOIN product_reviews r ON p.id = r.product_id;

- **Retrieve employee details by department**:

  SELECT e.id, e.employee_name, d.department_name, e.salary FROM employees e JOIN departments d ON e.department = d.id;

5.Final Goal of This Project

The final goal of this SQL Injection project is to:

- **Identify and Demonstrate Vulnerabilities**: Show how SQL injection vulnerabilities can be exploited in the provided databases and tables.
- **Assess and Improve Security**: Analyze the impact of SQL injection attacks on the databases and implement measures to secure the systems against such attacks.
- **Educate and Raise Awareness**: Provide insights and examples to help understand the risks associated with SQL injection and promote best practices for securing applications and databases.
- **Develop Mitigation Strategies**: Implement solutions such as input validation, parameterized queries, and prepared statements to prevent SQL injection vulnerabilities.

By achieving these goals, the project aims to enhance database security and protect against potential threats posed by SQL injection attacks.