SQL Injection Application

Index

1. Project Overview

This application demonstrates SQL Injection vulnerabilities caused by unsafe string-based SQL query construction using untrusted user input.

2. Learning Outcomes

- Understand SQL Injection attacks
- Exploit vulnerable authentication logic
- Apply prepared statements to prevent SQL Injection

3. Scope

In scope: SQL Injection exploitation and mitigation.
Out of scope: ORM abstractions, DB hardening.

4. Tech Stack & Versions

• Web Framework (Express / Flask)
• SQL Database (MySQL / PostgreSQL / SQLite)

5. Folder Structure

```
SQLInjection/
├── app
├── db
├── routes/auth
├── config
├── package.json / requirements.txt
└── README.md
```

6. High-Level Architecture

```
Client → Web API → SQL Database
```

7. Vulnerability Demonstration Flow

```
1. User input concatenated into SQL
2. Query logic altered
3. Authentication bypass
```

8. Attack Scenarios

```
' OR '1'='1
```

9. Secure Coding Fix

```
• Prepared statements
• Parameterized queries
• Never concatenate user input
```

10. API Contract

```
POST /login
```

11. How to Run

```
npm install

npm start
```

12. Common Mistakes

- Building SQL using string concatenation
- Trusting client-side validation

13. Debugging Techniques

- Log generated SQL queries
- Enable DB debug logging

Appendix A: Source Code

package.json

```
{
  "name": "node-mysql-injection-demo",
  "version": "1.0.0",
  "main": "server.js",
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^3.9.0"
  }
}
```

public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>SQL Injection Demo</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>

  <div class="container">
    <h1>SQL Injection Demo</h1>
    <div class="card">
      <label>Username</label>
      <input type="text" id="username" placeholder="Enter username" />

      <label>Password</label>
      <input type="text" id="password" placeholder="Enter password" />

      <div class="buttons">
        <button class="danger" onclick="vulnerableLogin()">
          Login (Vulnerable)
        </button>

        <button class="safe" onclick="safeLogin()">
          Login (Safe)
        </button>
      </div>

      <pre id="output"></pre>
    </div>

   </div>

  <script src="script.js"></script>
</body>
</html>
```

public/script.js

```
async function vulnerableLogin() {
  await sendRequest("/login-vulnerable");
}

async function safeLogin() {
  await sendRequest("/login-secure");
}

async function sendRequest(url) {
  const username = document.getElementById("username").value;
  const password = document.getElementById("password").value;
  const output = document.getElementById("output");

  output.textContent = "Sending request...";

  try {
    const res = await fetch(url, {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({ username, password })
    });

    const data = await res.json();
    output.textContent = JSON.stringify(data, null, 2);
  } catch (err) {
    output.textContent = "Error: " + err.message;
  }
}
```

public/styles.css

```
body {
  font-family: Arial, sans-serif;
  background: #f4f6f8;
}

.container {
  max-width: 600px;
  margin: 50px auto;
}

h1 {
  text-align: center;
}

.subtitle {
  text-align: center;
```

```
  color: #555;
}

.card {
  background: white;
  padding: 25px;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
}

label {
  display: block;
  margin-top: 15px;
  font-weight: bold;
}

input {
  width: 100%;
  padding: 10px;
  margin-top: 5px;
  font-size: 16px;
}

.buttons {
  margin-top: 20px;
  display: flex;
  gap: 10px;
}

button {
  flex: 1;
  padding: 12px;
  font-size: 16px;
  cursor: pointer;
  border: none;
  border-radius: 4px;
}

button.danger {
  background: #d9534f;
  color: white;
}

button.safe {
  background: #5cb85c;
  color: white;
}

pre {
  margin-top: 20px;
  background: #111;
  color: #0f0;
  padding: 15px;
  min-height: 80px;
  overflow-x: auto;
```

```
}
```

readme.txt

```
Step 1: Insert a test user (mandatory)

Create DB:
CREATE DATABASE injection_demo;
USE injection_demo;
CREATE TABLE users(id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(50),
password VARCHAR(50));
INSERT INTO users VALUES (1,'admin','admin123');

Step 2: Install and Start
npm i
npm start


Step 3: Demo scenarios

Open browser
http://localhost:3000

Step 3.1 : Normal login
Username: admin
Password: admin123

Button     Result
Vulnerable    Success
Secure   Success

Injection attack
Username: admin' OR '1'='1
Password: anything

Button     Result
Vulnerable    Login Successful
Secure     Invalid Input
```

server.js

```
const express = require("express");
const mysql = require("mysql2");

const app = express();
app.use(express.json());
app.use(express.static("public"));

const db = mysql.createConnection({
  host: "localhost",
  user: "root",
```

```
  password: "root",
  database: "injection_demo"
});

db.connect();

// Vulnerable Login
app.post("/login-vulnerable", (req, res) => {
  const { username, password } = req.body;

  const query =
    "SELECT * FROM users WHERE username = '" +
    username +
    "' AND password = '" +
    password +
    "'";

  db.query(query, (err, results) => {
    if (err) {
      console.error("SQL Error (vulnerable):", err.message);
      return res.json({
        success: false,
        error: "SQL error occurred",
        details: err.message
      });
    }

    if (results.length > 0) {
      return res.json({
        success: true,
        message: "Logged in (vulnerable)",
        results
      });
    }

    res.json({
      success: false,
      message: "Invalid credentials"
    });
  });
});

// Secure Login (Prepared Statement)
app.post("/login-secure", (req, res) => {
  const { username, password } = req.body;

  const query =
    "SELECT * FROM users WHERE username = ? AND password = ?";

  db.execute(query, [username, password], (err, results) => {
    if (err) {
      console.error("SQL Error (safe):", err.message);
      return res.json({
        success: false,
        error: "SQL error"
```

```
      });
    }

    if (results.length > 0) {
      return res.json({
        success: true,
        message: "Logged in (safe)"
      });
    }

    res.json({
      success: false,
      message: "Invalid credentials"
    });
  });
});

app.listen(3000, () => console.log("Server running on port 3000"));
```