# Exploiting-and-Mitigating-Web-Vulnerabilities

**Problem Statement:**

**SQL injection vulnerability allowing login bypass**

**This lab contains a SQL injection vulnerability in the login function. To solve the lab, perform a SQL injection attack that logs in to the application as the administrator user**

**SQL Injection Vulnerability: Login Bypass (Administrator Account)**

**Objective:**

**To perform a SQL Injection attack by modifying the username parameter to administrator'--, allowing login as the administrator user.**

**Prerequisites:**

- **Access to a vulnerable web application's login page.**
- **Understanding of SQL queries and SQL injection techniques.**

**Steps to Perform the Attack:**

1. **Navigate to the Login Page: Open the web browser and go to the login page of the vulnerable application.**
2. **Identify Input Fields: Locate the fields for entering the username and password:**
    - **Username: [Text Box]**
    - **Password: [Text Box]**
3. **Inject SQL Payload: In the username field, input the following payload:**

**administrator'--**

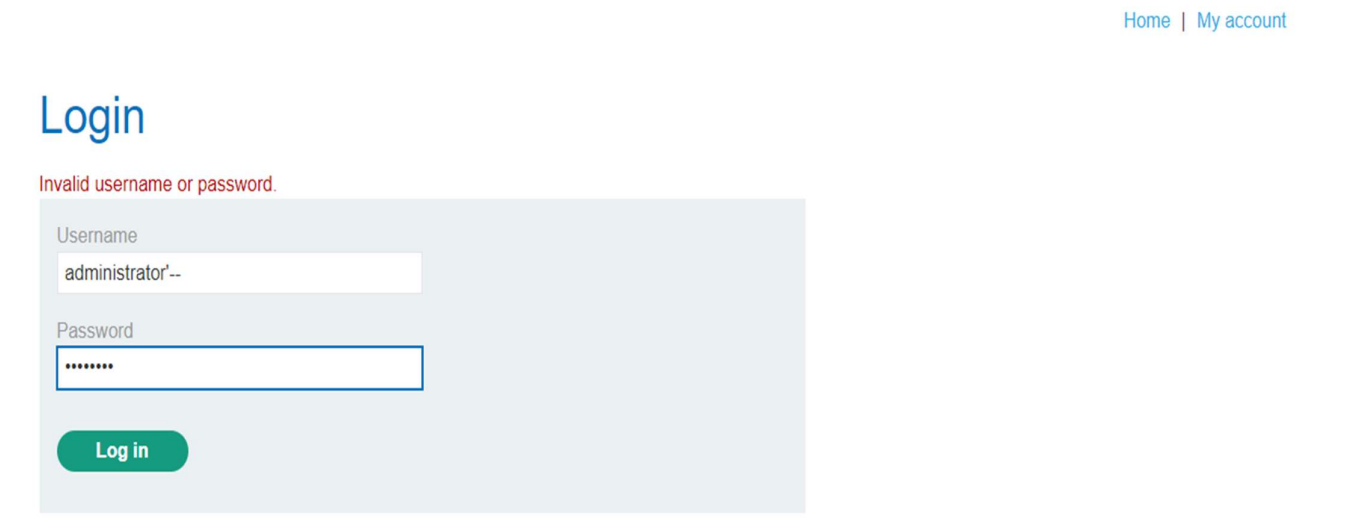
**Leave the password field empty or input anything, as it will be ignored.**

**Explanation of the Payload:**

- **administrator': This part of the input is intended to close the string in the SQL query for the username value.**
- **--: This is an SQL comment operator, which ignores the rest of the SQL query, including the password check. It effectively bypasses the need for a valid password.**

4. **Submit the Form: Click on the Login or Submit button.**
5. **Expected Outcome: If the SQL injection is successful, the query will log you in as the administrator user without needing a valid password. You should gain access to the admin functionalities of the application.**

---





**Sample Login Query Before Injection:**

**In a vulnerable application, the SQL query that processes the login might look like this:**

```
SELECT * FROM users WHERE username = 'user_input' AND password = 'user_password';
```

**Sample Login Query After Injection:**

**After injecting the payload administrator'--, the query will be modified to:**

```
SELECT * FROM users WHERE username = 'administrator'--' AND password = '';
```

**This query ignores the password condition and attempts to log in as the administrator user directly.**

**SQL Injection Prevention:**

**To avoid such vulnerabilities, it is crucial to:**

- **Use prepared statements and parameterized queries to ensure that user input does not directly affect the SQL query logic.**
- **Validate and sanitize all user inputs to prevent malicious SQL commands from being executed.**
- **Limit user privileges based on roles to reduce the impact of a successful attack.**

**Conclusion:**

**By modifying the username parameter to administrator'--, it is possible to bypass the authentication mechanism and log in as the administrator. This demonstrates how dangerous SQL injection vulnerabilities can be when input is not properly sanitized.**