

# Penetration Testing Report

**Full Name:** Aashutosh Thakur

**Program:** HCS - Penetration Testing Internship Week-2

**Date:** 22/02/2025

## Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week 2 Labs**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

### 1. Objective

The objective of the assessment was to uncover vulnerabilities in the **Week 2 Labs** and provide a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

### 2. Scope

This section defines the scope and boundaries of the project.

<b>Application Name</b>	Insecure Direct Object References, SQL injection
-------------------------	--

### 3. Summary

Outlined is a Black Box Application Security assessment for the **Week 2 Labs**.

**Total number of Sub-labs: 16 Sub-labs**

High	Medium	Low
5	6	5

**High** - 5

**Medium** - 6

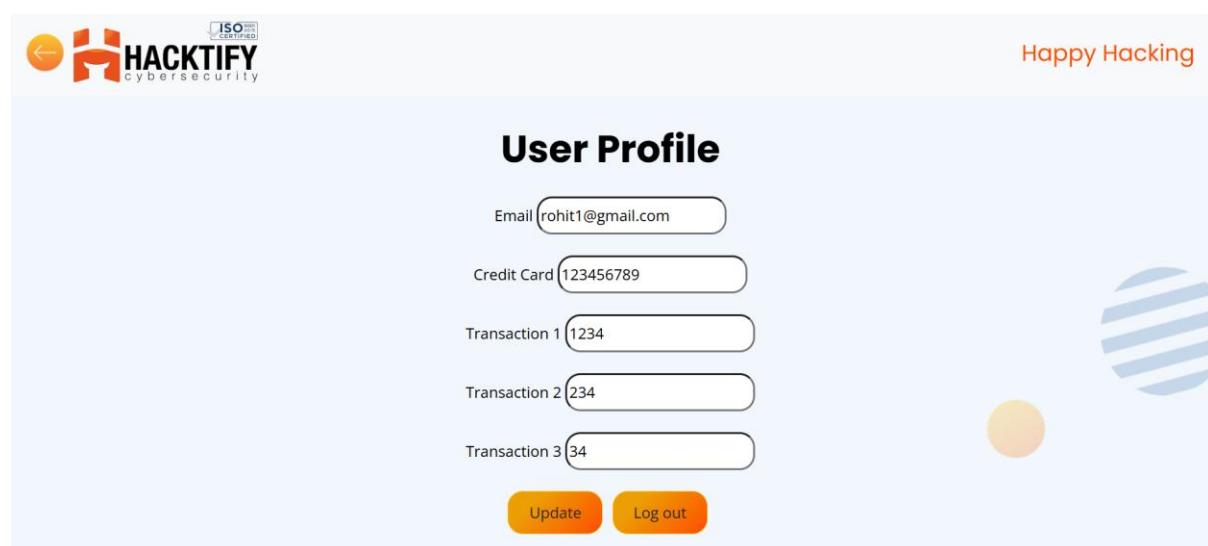
## 1. Insecure Direct Object References

### 1.1. Give me my amount!!

Reference	Risk Rating
Give me my amount!!	Low
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
IDOR allows unauthorized access to data by altering internal object references (e.g., user IDs or file names) in URLs or request parameters.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/idor_lab/lab_1/profile.php?id=7">https://labs.hackify.in/HTML/idor_lab/lab_1/profile.php?id=7</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix IDOR can lead to data leakage, privacy breaches, compliance violations, and financial losses.	
<b>Suggested Countermeasures</b>	
Implement authorization checks, use indirect references, validate inputs, and follow the principle of least privilege.	
<b>References</b>	
<a href="Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html">Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



The screenshot shows a user profile page from the Hackify cybersecurity platform. At the top left is the Hackify logo with an orange 'H' icon and the text 'HACKIFY cybersecurity'. At the top right is the text 'Happy Hacking'. The main title 'User Profile' is centered at the top. Below it, there are five input fields: 'Email' containing 'rohit1@gmail.com', 'Credit Card' containing '123456789', 'Transaction 1' containing '1234', 'Transaction 2' containing '234', and 'Transaction 3' containing '34'. At the bottom are two orange buttons: 'Update' and 'Log out'. The background features abstract blue and yellow circular patterns.

## 1.2. Stop polluting my params!

Reference	Risk Rating
Stop polluting my params!	Medium
Tools Used	
Manual code review and request analysis.	
Vulnerability Description	
IDOR allows unauthorized access to data by altering internal object references (e.g., user IDs or file names) in URLs or request parameters.	
How It Was Discovered	
Manual analysis by manipulating URL parameters and observing responses.	
Vulnerable URLs	
<a href="https://labs.hackify.in/HTML/idor_lab/lab_2/profile.php?id=1131">https://labs.hackify.in/HTML/idor_lab/lab_2/profile.php?id=1131</a>	
Consequences of not Fixing the Issue	
Failure to fix IDOR can lead to data leakage, privacy breaches, compliance violations, and financial losses.	
Suggested Countermeasures	
Implement authorization checks, use indirect references, validate inputs, and follow the principle of least privilege.	
References	
<a href="Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html">Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows a user profile page from the Hackify cybersecurity platform. At the top left is the Hackify logo with an orange 'H' icon and the text 'HACKIFY cybersecurity'. At the top right is the text 'Happy Hacking'. The main title 'User Profile' is centered above a form. The form contains three input fields: 'Username' with the value 'lef@gmail.com', 'First Name' with the value 'ab', and 'Last Name' with the value 'cd'. Below the form are two orange buttons: 'Update' on the left and 'Log out' on the right. The background of the page features abstract blue and white circular and striped graphics.

## 1.3. Someone changed my Password !

Reference	Risk Rating
Someone changed my Password !	High
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
IDOR allows unauthorized access to data by altering internal object references (e.g., user IDs or file names) in URLs or request parameters.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hacktify.in/HTML/idor_lab/lab_3/profile.php">https://labs.hacktify.in/HTML/idor_lab/lab_3/profile.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix IDOR can lead to data leakage, privacy breaches, compliance violations, and financial losses.	
<b>Suggested Countermeasures</b>	
Implement authorization checks, use indirect references, validate inputs, and follow the principle of least privilege.	
<b>References</b>	
<a href="Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html">Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows the 'User Profile' page from the Hacktify cybersecurity platform. At the top left is the Hacktify logo with an orange 'H' icon and the text 'HACKTIFY cybersecurity'. To the right is the text 'Happy Hacking'. In the center, the heading 'User Profile' is displayed. Below it are three input fields: 'Username' containing 'lucky', 'Email' containing 'abc@gmail.com', and 'Name' containing 'lakshya'. At the bottom of the profile section are two buttons: 'Change Password' and 'Log out'. To the right of the profile section, there are two decorative circular icons: one yellow and one blue with horizontal stripes.

## 1.4. Change your methods!

Reference	Risk Rating
Change your methods!	Medium
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
IDOR allows unauthorized access to data by altering internal object references (e.g., user IDs or file names) in URLs or request parameters.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/idor_lab/lab_4/profile.php?id=1734">https://labs.hackify.in/HTML/idor_lab/lab_4/profile.php?id=1734</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix IDOR can lead to data leakage, privacy breaches, compliance violations, and financial losses.	
<b>Suggested Countermeasures</b>	
Implement authorization checks, use indirect references, validate inputs, and follow the principle of least privilege.	
<b>References</b>	
<a href="Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html">Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows a "User Profile" form from the Hackify cybersecurity platform. The form includes fields for "Username", "First Name", and "Last Name", each with a corresponding input box. Below the input boxes are two buttons: "Update" and "Log out". The "Update" button is orange, while "Log out" is grey. The Hackify logo is visible in the top left corner, and the text "Happy Hacking" is in the top right corner. The background features abstract circular patterns in blue and yellow.

## 2. SQL Injection

### 2.1. Strings & Errors Part 1!

Reference	Risk Rating
Strings & Errors Part 1!	Low
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_1/lab_1.php">https://labs.hackify.in/HTML/sql_injection/lab_1/lab_1.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



## 2.2. Strings & Errors Part 2!

Reference	Risk Rating
Strings & Errors Part 2!	Low
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_2/lab_2.php?id=1%27%20UNION%20SELECT%20username,%20password%20FROM%20users%20--">https://labs.hackify.in/HTML/sql_injection/lab_2/lab_2.php?id=1%27%20UNION%20SELECT%20username,%20password%20FROM%20users%20--</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="Https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



## 2.3. Strings & Errors Part 3!

Reference	Risk Rating
Strings & Errors Part 3!	Low
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_3/lab_3.php?id=1%27%20UNION%20SELECT%20username,%20password%20FROM%20users%20--">https://labs.hackify.in/HTML/sql_injection/lab_3/lab_3.php?id=1%27%20UNION%20SELECT%20username,%20password%20FROM%20users%20--</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="Https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



## 2.4. Let's Trick `em!

Reference	Risk Rating
Let's Trick `em!	High
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_4/lab_4.php">https://labs.hackify.in/HTML/sql_injection/lab_4/lab_4.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows the 'Admin Login' page of a web application. At the top, there is a logo for 'HACKIFY cybersecurity' and an ISO 27001 certification badge. Below the logo, there are two input fields: 'Email:' followed by a placeholder 'Enter Email' and 'Password:' followed by a placeholder 'Enter Password'. A large orange 'Login' button is centered below the password field. At the bottom of the page, a message box displays the following text: 'Query Unsuccessful: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'A='A'--' LIMIT 0,1' at line 1'. This indicates that a SQL injection attack was successful.

## 2.5. Booleans and Blind!

Reference	Risk Rating
Booleans and Blinds	High
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_5/lab_5.php?id=1%20AND%201=1%20--">https://labs.hackify.in/HTML/sql_injection/lab_5/lab_5.php?id=1%20AND%201=1%20--</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

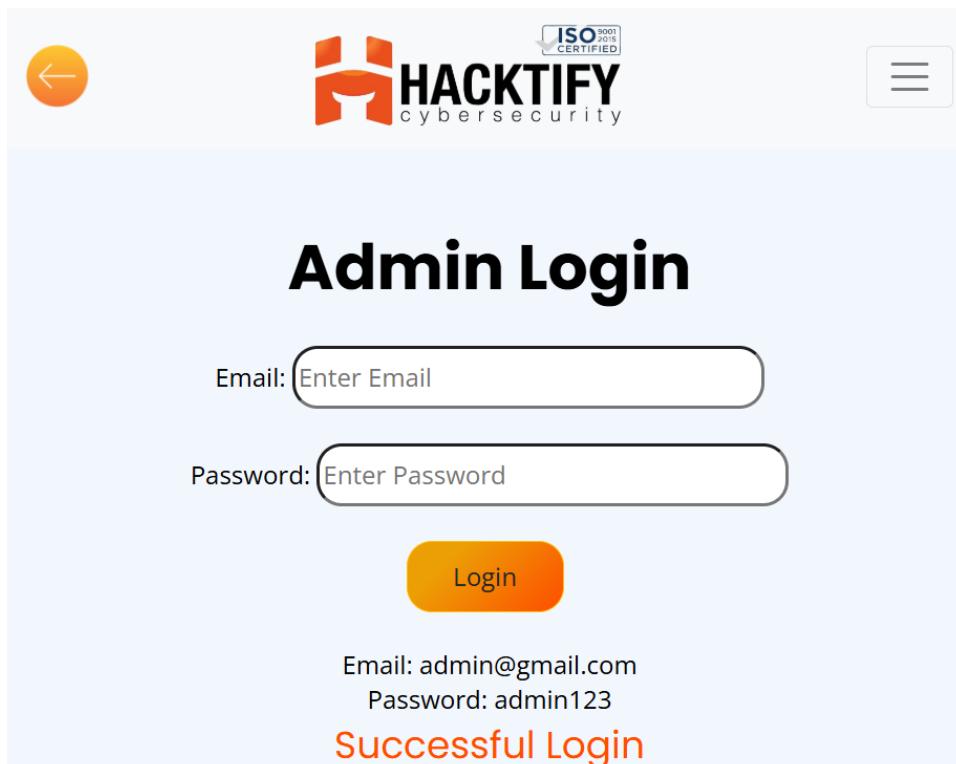


## 2.6. Error Based: Tricked

Reference	Risk Rating
Error Based: Tricked	Medium
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_6/lab_6.php">https://labs.hackify.in/HTML/sql_injection/lab_6/lab_6.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



## 2.7. Errors and Post!

Reference	Risk Rating
Errors and Post!	Low
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_7/lab_7.php">https://labs.hackify.in/HTML/sql_injection/lab_7/lab_7.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows the 'Admin Login' page of a web application. At the top, there is a logo for 'HACKIFY cybersecurity' with an ISO 9001 2015 CERTIFIED badge. Below the logo, there are two input fields: 'Email:' followed by a placeholder 'Enter Email' and 'Password:' followed by a placeholder 'Enter Password'. A large orange 'Login' button is centered below the password field. At the bottom of the page, a red error message is displayed: 'Query Unsuccessful: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '1' LIMIT 0,1' at line 1'.

## 2.8. User Agents lead us!

Reference	Risk Rating
User Agents lead us!	High
<b>Tools Used</b>	
Brup Suite and Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters, Brup Suite and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_8/lab_8.php">https://labs.hackify.in/HTML/sql_injection/lab_8/lab_8.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows an "Admin Login" interface. It features two input fields: "Email: Enter Email" and "Password: Enter Password", both with placeholder text. Below these is an orange "Login" button. At the bottom of the page, the text "Your IP ADDRESS is: 185.220.103.8" is displayed, followed by a red success message "Successful Login". At the very bottom, a user agent string is shown: "Your User Agent is: Mozilla/5.0 (X11; Linux x86\_64; rv:128.0) Gecko/20100101 Firefox/128.0 OR '1'='1".

## 2.9. Referer lead us!

Reference	Risk Rating
Referer lead us!	Medium
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_9/lab_9.php">https://labs.hackify.in/HTML/sql_injection/lab_9/lab_9.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows an "Admin Login" interface. It features two input fields: "Email: Enter Email" and "Password: Enter Password", both with placeholder text. Below these is an orange "Login" button. Underneath the form, the text "Your IP ADDRESS is: 192.42.116.218" is displayed. At the bottom, the message "Successful Login" is shown in red, along with the text "Your User Agent is: '1' OR '1'='1'".

## 2.10. Oh Cookies!

Reference	Risk Rating
Oh Cookies!	High
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_10/lab_10.php">https://labs.hackify.in/HTML/sql_injection/lab_10/lab_10.php</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

The screenshot shows a web page from the Hackify cybersecurity lab. At the top left is the Hackify logo with an orange 'H' and the text 'HACKIFY cybersecurity'. At the top right is the text 'Happy Hacking'. Below the header, there is a message about the user agent and IP address. In the center, there is a section titled 'DELETE YOUR COOKIE OR WAIT FOR IT TO EXPIRE' containing a SQL injection payload. To the right, there are decorative icons of a globe and a sun. At the bottom, there is a button labeled 'Delete Your Cookie!'

Your USER AGENT is Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36

Your IP ADDRESS is 137.59.92.42

DELETE YOUR COOKIE OR WAIT FOR IT TO EXPIRE

YOUR COOKIE: username: ' union SELECT version(),user(),database()#T and expires: Sat 22 Feb 2025 - 09:43:36

Your Login username:  
bugbzhiiy\_bughunter@localhost  
Your Password: bugbzhiiy\_sql  
Your ID: 10.6.20-MariaDB-cll-lve-log

Delete Your Cookie!

## 2.11. WAF's are injected!

Reference	Risk Rating
WAF's are injected!	Medium
Tools Used	
Manual code review and request analysis.	
Vulnerability Description	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
How It Was Discovered	
Manual analysis by manipulating URL parameters and observing responses.	
Vulnerable URLs	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_11/lab_11.php?id=1&amp;?id=0%27+union+select+1,@@version,database()--+">https://labs.hackify.in/HTML/sql_injection/lab_11/lab_11.php?id=1&amp;?id=0%27+union+select+1,@@version,database()--+</a>	
Consequences of not Fixing the Issue	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
Suggested Countermeasures	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
References	
<a href="Https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



## 2.12. WAF's are injected Part 2!

Reference	Risk Rating
WAF's are injected Part 2!	Medium
<b>Tools Used</b>	
Manual code review and request analysis.	
<b>Vulnerability Description</b>	
SQL Injection occurs when an attacker is able to insert or manipulate SQL queries in user input fields (like forms or URLs), allowing them to execute arbitrary SQL commands on the database.	
<b>How It Was Discovered</b>	
Manual analysis by manipulating URL parameters and observing responses.	
<b>Vulnerable URLs</b>	
<a href="https://labs.hackify.in/HTML/sql_injection/lab_12/lab_12.php?id=1&amp;?id=0%27+union+select+1,@@version,database()--+">https://labs.hackify.in/HTML/sql_injection/lab_12/lab_12.php?id=1&amp;?id=0%27+union+select+1,@@version,database()--+</a>	
<b>Consequences of not Fixing the Issue</b>	
Failure to fix SQL injection can lead to unauthorized data access, data modification, deletion, authentication bypass, and potential full system compromise.	
<b>Suggested Countermeasures</b>	
Use prepared statements with parameterized queries, validate and sanitize all user inputs, implement least privilege for database access, and disable detailed error messages in production environments.	
<b>References</b>	
<a href="Https://owasp.org/www-community/attacks/SQL_Injection">Https://owasp.org/www-community/attacks/SQL_Injection</a>	

## Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



**NOTES:**

- Everything mentioned inside {} has to be changed based on your week, labs and sub-labs.
- If you have 2 labs in same week you need to mention that, if not ignore those mentions for lab 2.
- Here it is given with 2 Sub-labs vulnerability, you need to add all the sub-labs based on your labs.
- Don't forget to add the screenshot of the vulnerability in the proof of concept.
- This NOTE session is only for your reference, don't forget to delete this in the report you submit.