Penetration Testing Report

Full Name: Rohit Kumar
Program: HCS - Penetration Testing Internship Week-2
Date: 24-Feb-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.

Application Name	**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: 13 Sub-labs

High	Medium	Low
3	4	6

High - **Number of Sub-labs with hard difficulty level**

Medium - **Number of Sub-labs with Medium difficulty level**

Low - **Number of Sub-labs with Easy difficulty level**

1. Insecure Direct Object References

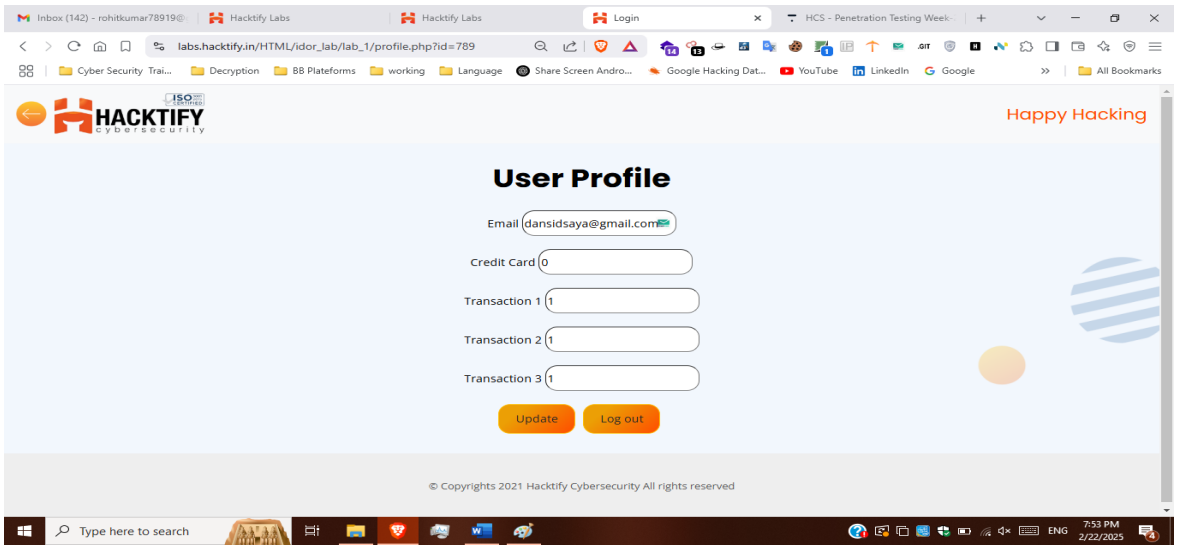
1.1. Give me my amount!!

Reference	Risk Rating
Give me my amount!!	**Low**
Tools Used	
Brave Browser	
Vulnerability Description	
An insecure direct object reference (IDOR) is an access control vulnerability where invalidated user input can be used for unauthorized access to resources or operations. It occurs when an attacker gains direct access by using user-supplied input to an object that has no authorization to access	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=788	
Consequences of not Fixing the Issue	
the attacker get hold of the sensitive user details, document and other useful information which he should not have access to.	
Suggested Countermeasures	
Use Web Application Firewalls, Input Validation and Sanitization, Do not expose internal object identifiers (like database keys or file names).	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to Reproduce:
Step 1: Click Register and enter details, then login with detail.
Step 2: after login change the value id=788 to 789 in URL
https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=788

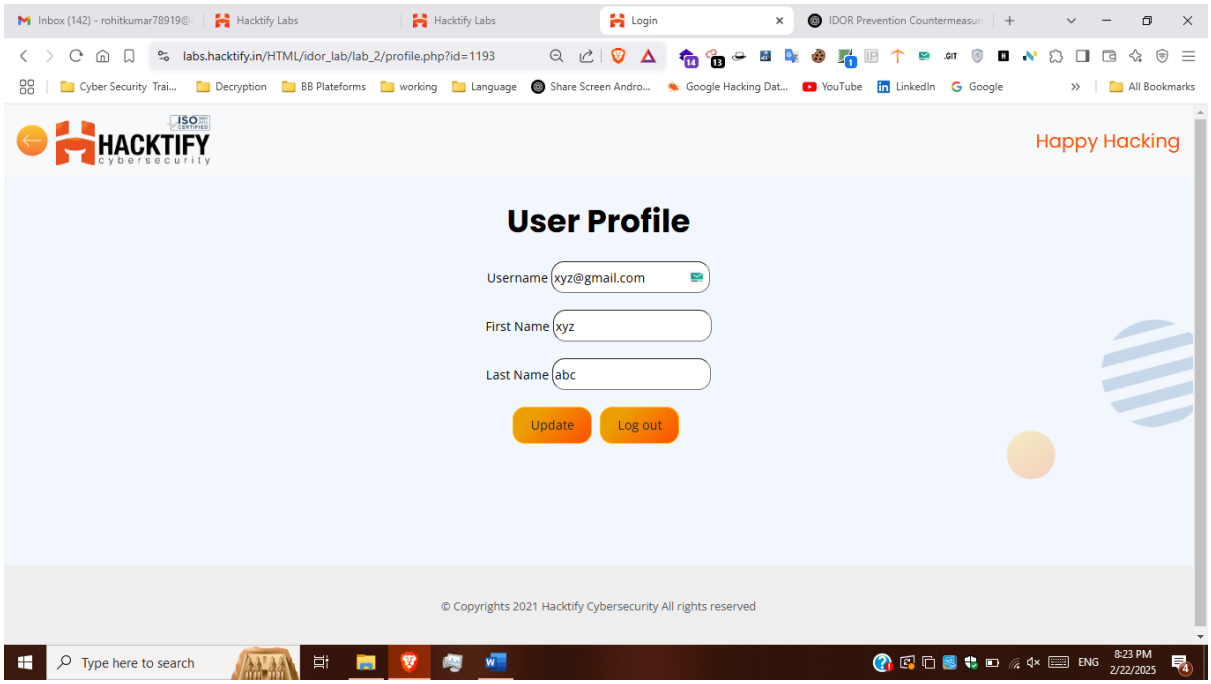


1.2. Stop polluting my params!

Reference	Risk Rating
Stop polluting my params!	**Medium**
Tools Used	
Brave Browser	
Vulnerability Description	
An insecure direct object reference (IDOR) is an access control vulnerability where invalidated user input can be used for unauthorized access to resources or operations. It occurs when an attacker gains direct access by using user-supplied input to an object that has no authorization to access	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1196	
Consequences of not Fixing the Issue	
the attacker get hold of the sensitive user details, document and other useful information which he should not have access to.	
Suggested Countermeasures	
Use Web Application Firewalls, Input Validation and Sanitization, Do not expose internal object identifiers (like database keys or file names).	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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



2. SQL Injection

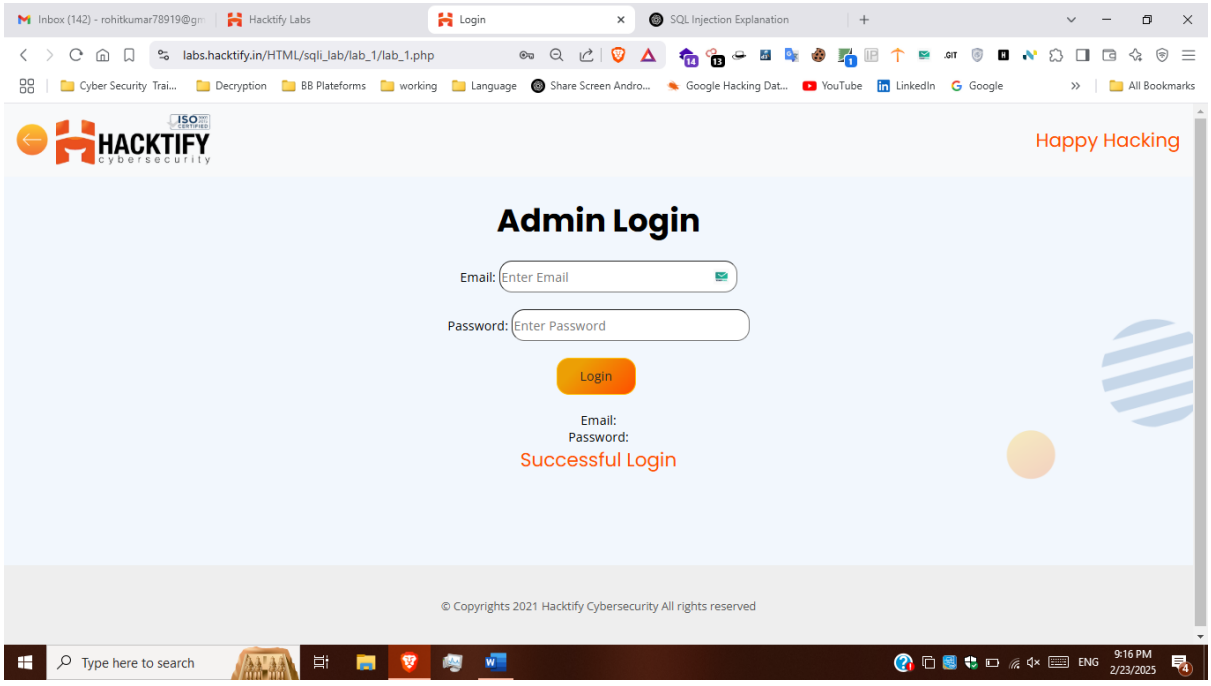
2.1. Strings & Errors Part 1!

Reference	Risk Rating
Strings & Errors Part 1!	**Low**
Tools Used	
Brave Browser, SQL injection Payload (' OR '1'='1)	
Vulnerability Description	
If the application directly constructs SQL queries using these inputs (' OR '1'='1) without proper sanitization. If the user enters ' OR '1'='1 as the username and password. The condition '1'='1' is always true, so this query will return all users from the users table, potentially granting unauthorized access to the system	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
Attackers can log in without valid credentials	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries. For example, allow only specific characters for inputs like usernames and email	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to Produce: Use payload '' OR 1=1 --' in login form



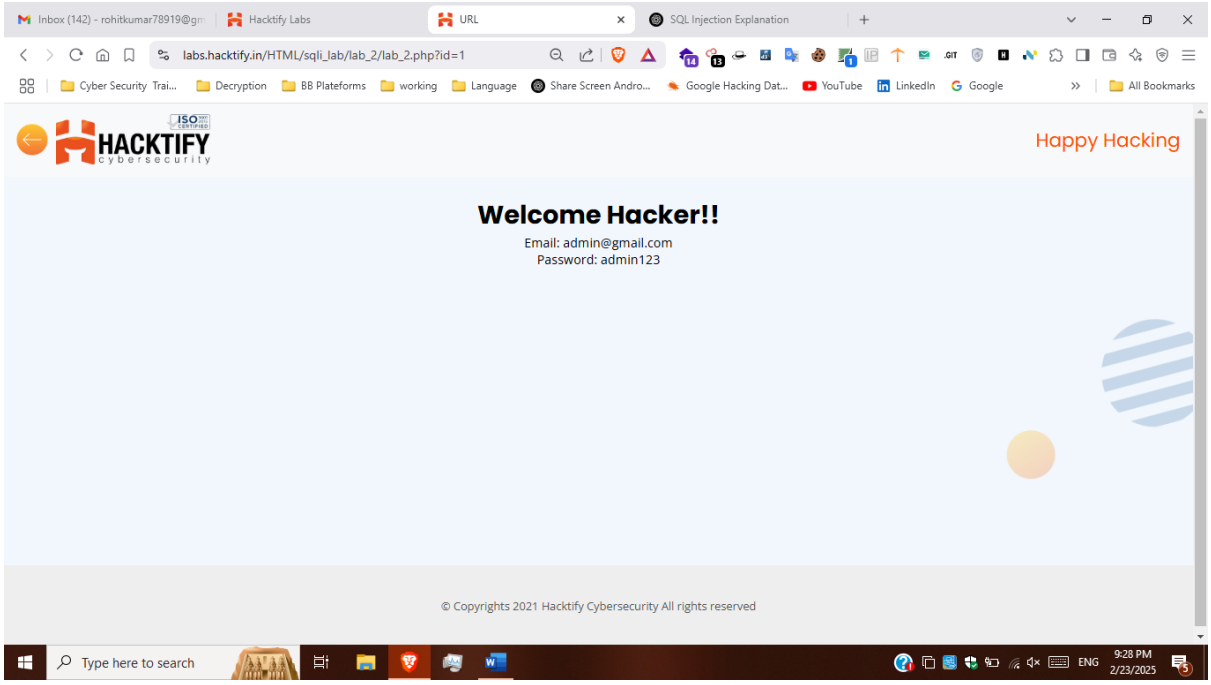
2.2. Strings & Errors Part 2!

Reference	Risk Rating
Strings & Errors Part 2!	Low
Tools Used	
Brave Browser, SQL Injection Payloads	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php	
Consequences of not Fixing the Issue	
Attackers can view sensitive data such as usernames, passwords, email addresses, etc.	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to produce:
Modifying URL Values in the Browser (Manually) that is
https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php?id=1



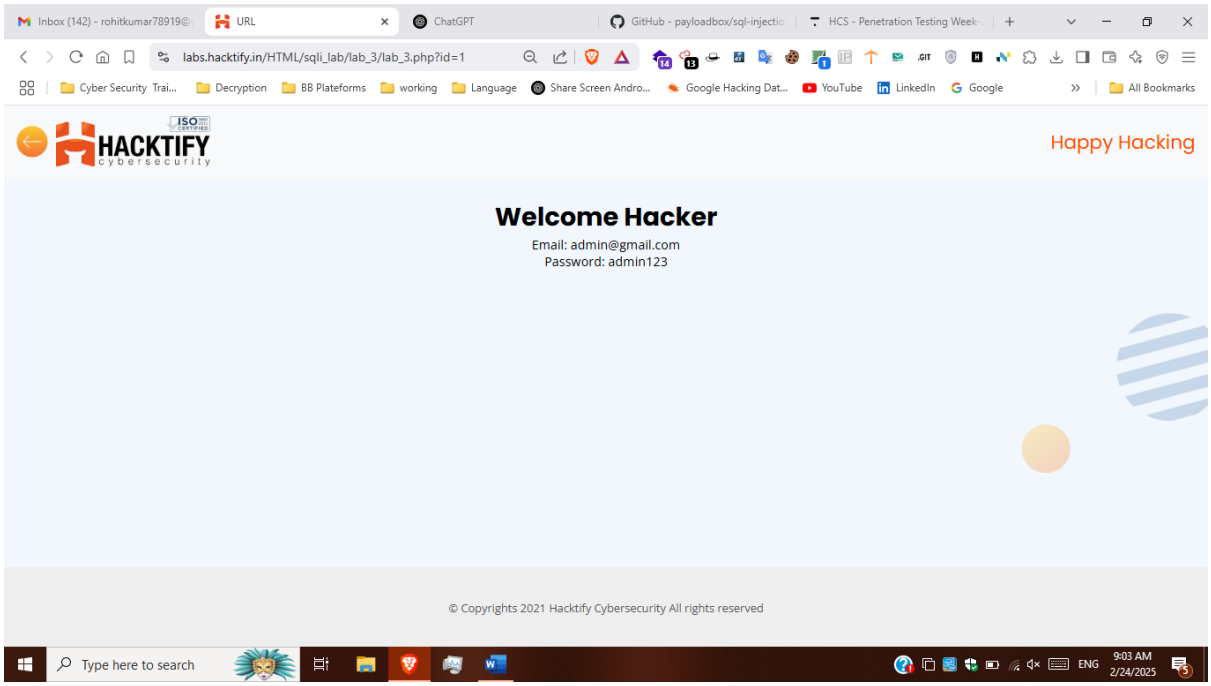
2.3. Strings & Errors Part 3!

Reference	Risk Rating
Strings & Errors Part 3!	Low
Tools Used	
Brave Browser, SQL Injection Payloads	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php	
Consequences of not Fixing the Issue	
Attackers can view sensitive data such as usernames, passwords, email addresses, etc.	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to produce:
Modifying URL Values in the Browser (Manually) that is
https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php?id=1



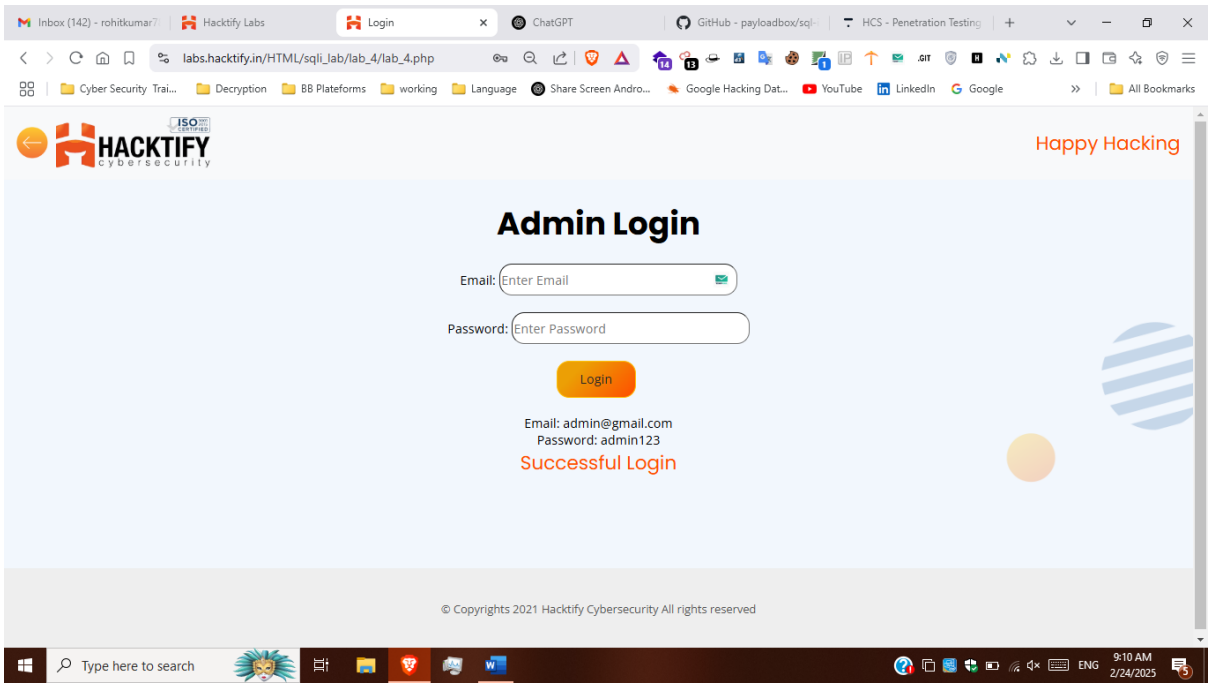
2.4. Let's Trick 'em!

Reference	Risk Rating
Let's Trick 'em!	**Medium**
Tools Used	
Brave Browser, SQL Injection Payloads (' OR ' '=')	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_4/lab_4.php	
Consequences of not Fixing the Issue	
Attackers can log in without valid credentials	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries. For example, allow only specific characters for inputs like usernames and email	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to Produce: Use payload ' OR ' '=' in login form



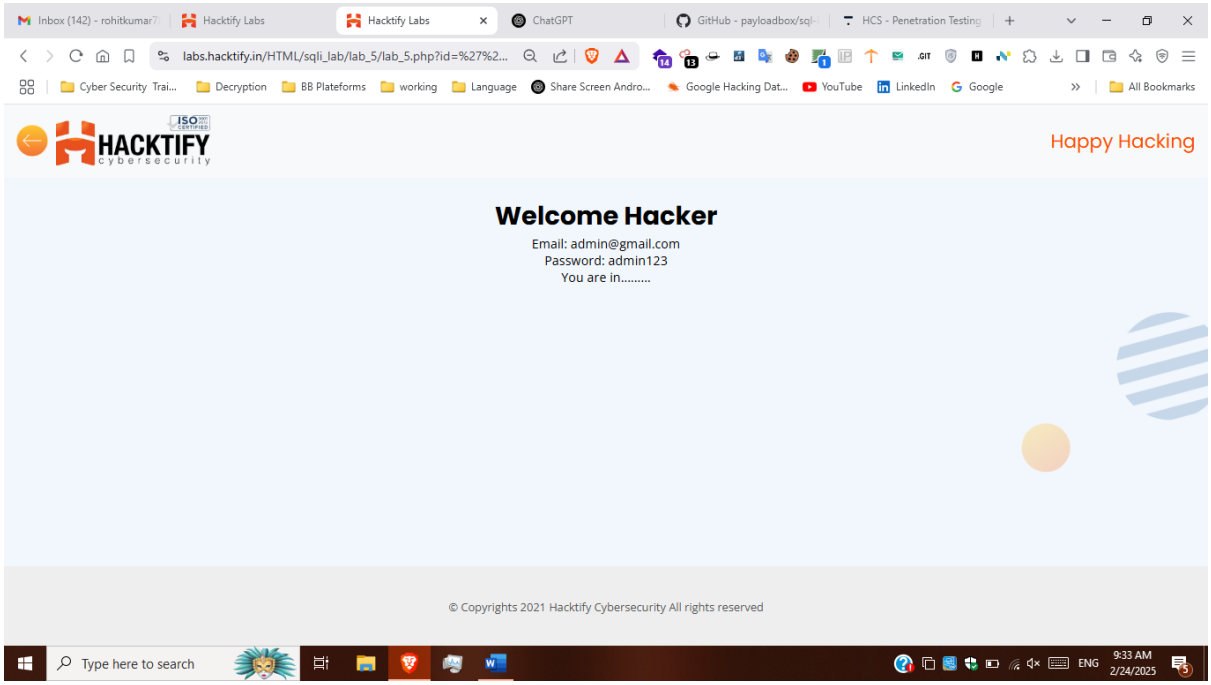
2.5. Booleans and Blind!

Reference	Risk Rating
Booleans and Blind!	High
Tools Used	
Brave Browser, SQL Injection Payloads (' OR '1)	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php	
Consequences of not Fixing the Issue	
Attackers can view sensitive data such as usernames, passwords, email addresses, etc.	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to produce:
Modifying URL Values in the Browser (Manually) that is
https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php?id= ' OR '1



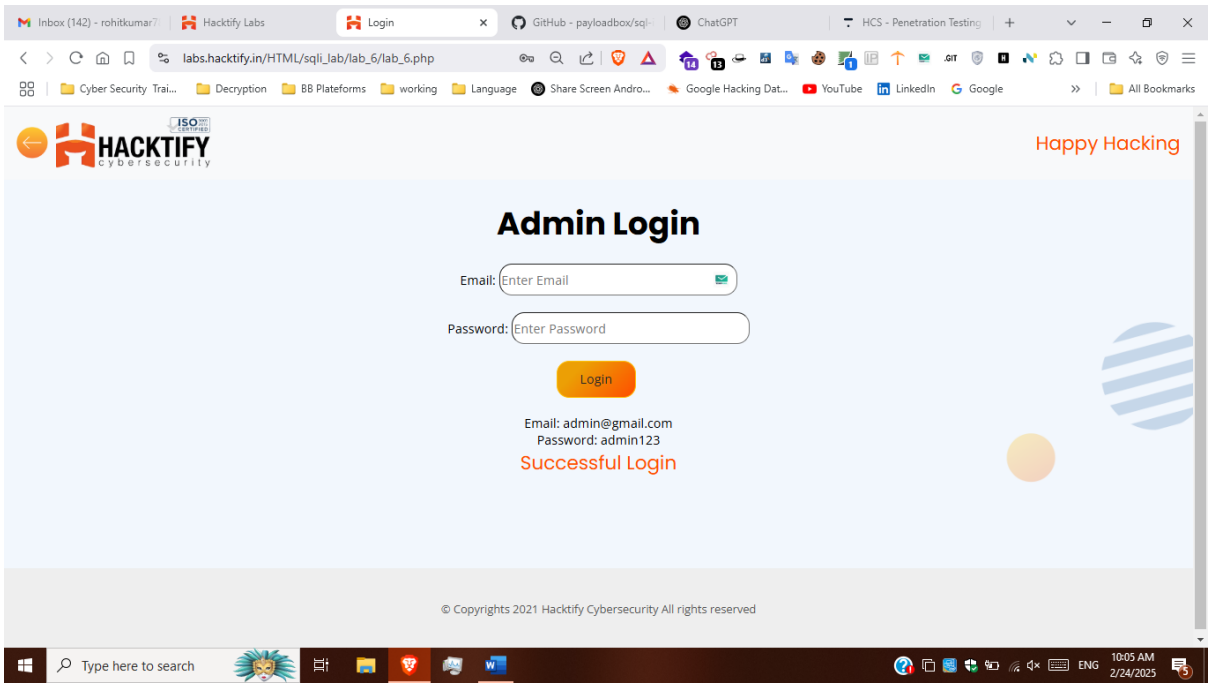
2.6. Error Based : Tricked

Reference	Risk Rating
Error Based : Tricked	Medium
Tools Used	
Brave Browser, SQL Injection Payloads {**admin") or ("1"="1**}	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_6/lab_6.php	
Consequences of not Fixing the Issue	
Attackers can log in without valid credentials	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries. For example, allow only specific characters for inputs like usernames and email	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA, https://github.com/payloadbox/sql-injection-payload-list	

Proof of Concept

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

Steps to Produce: Use payload admin") or ("1"="1 in login form



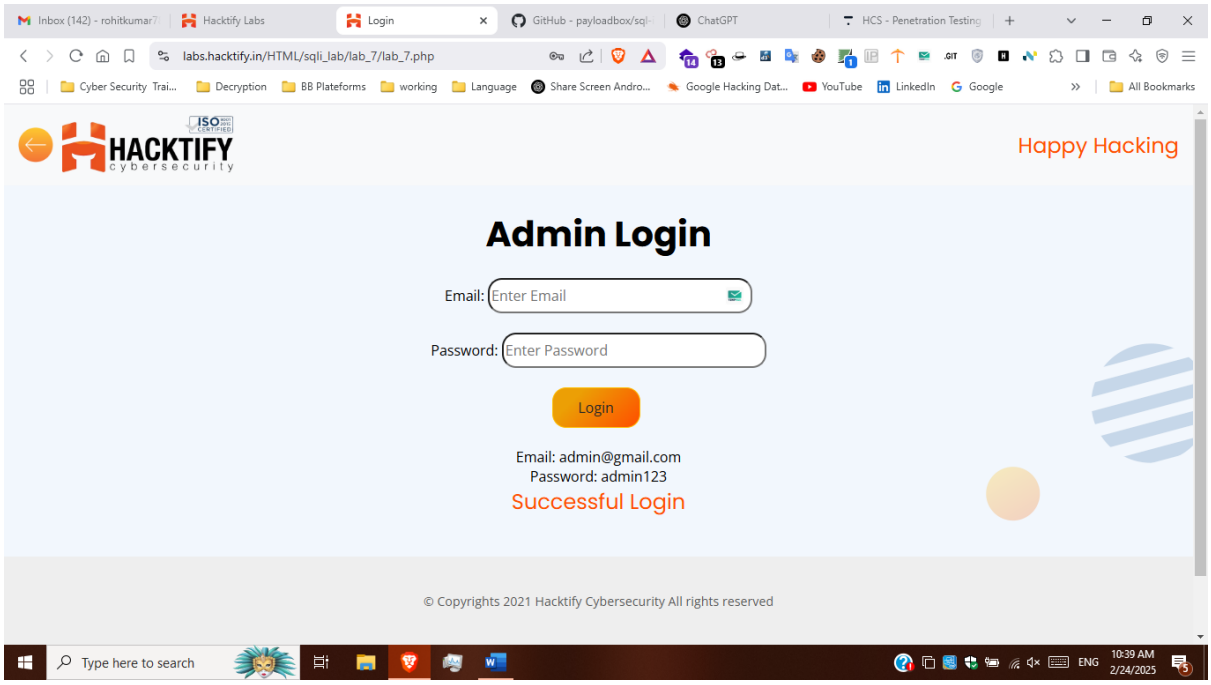
2.7. Errors and Post!

Reference	Risk Rating
Errors and Post!	Low
Tools Used	
Brave Browser, SQL Injection Payloads {admin' or '1'='1}	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_7/lab_7.php	
Consequences of not Fixing the Issue	
Attackers can view sensitive data such as usernames, passwords, email addresses, etc.	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to Produce: Use payload admin' or '1'='1 in login form



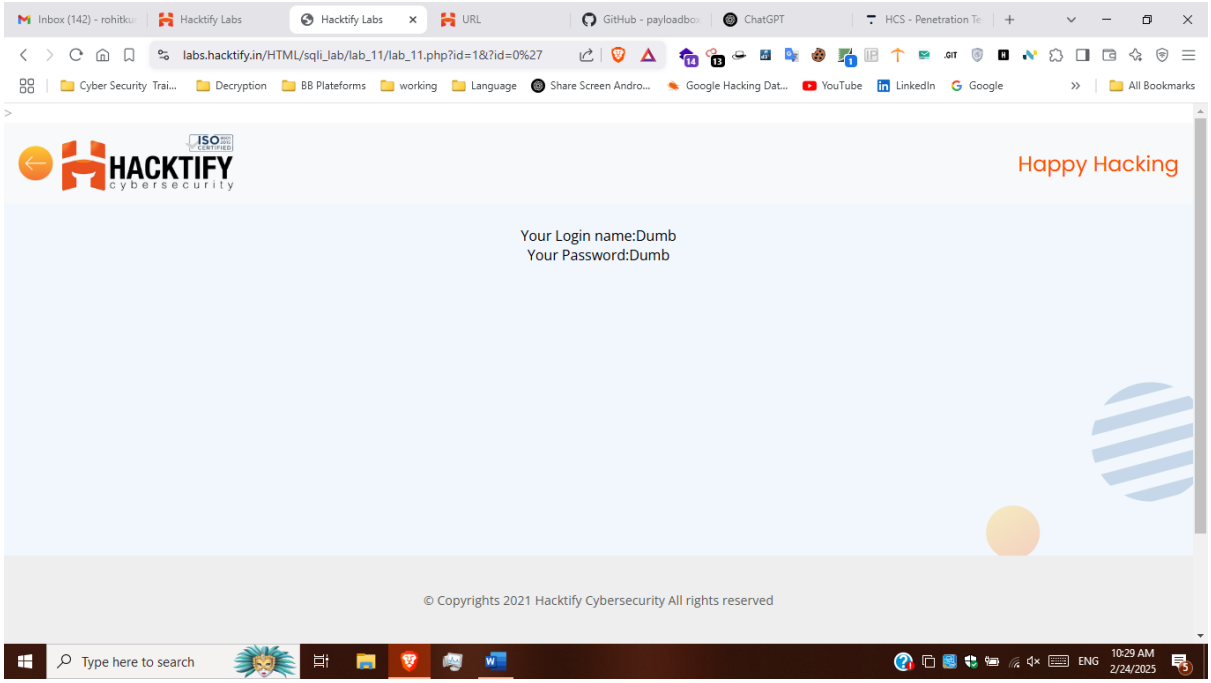
2.8. WAF's are injected!

Reference	Risk Rating
WAF's are injected!	High
Tools Used	
Brave Browser, SQL Injection Payloads	
Vulnerability Description	
SQL injection works by taking advantage of unsanitized user input in a web application. Attackers can inject SQL code into forms, query parameters, or URL parameters that the web application uses to interact with the database	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_11/lab_11.php	
Consequences of not Fixing the Issue	
Attackers can view sensitive data such as usernames, passwords, email addresses, etc.	
Suggested Countermeasures	
Always validate and sanitize user inputs before using them in queries	
References	
Week 2 YouTube Link: https://www.youtube.com/watch?v=pFDHFjxWIxA	

Proof of Concept

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

Steps to produce:
Modifying URL Values in the Browser (Manually) that is
https://labs.hacktify.in/HTML/sqli_lab/lab_11/lab_11.php?id=1&?id=0%27



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.**