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.

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: 16 Sub-labs

High	Medium	Low
5	6	5

High - **5**

Medium - **6**

Low - 5

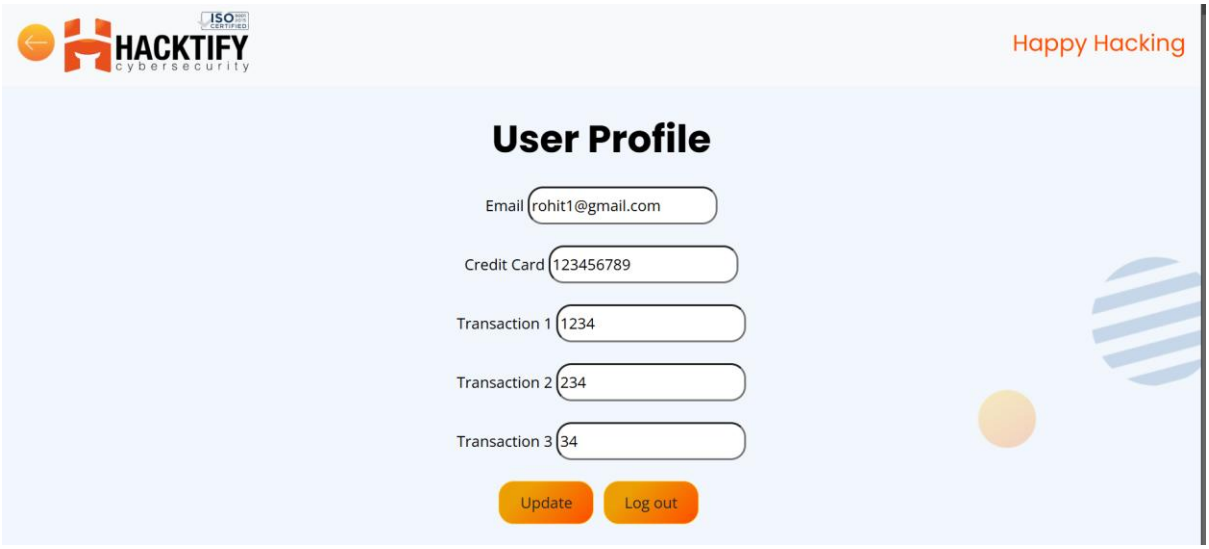
1. Insecure Direct Object References

1.1. Give me my amount!!

Reference	Risk Rating
Give me my amount!!	Low
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	
https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=7	
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	
Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html	

Proof of Concept

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

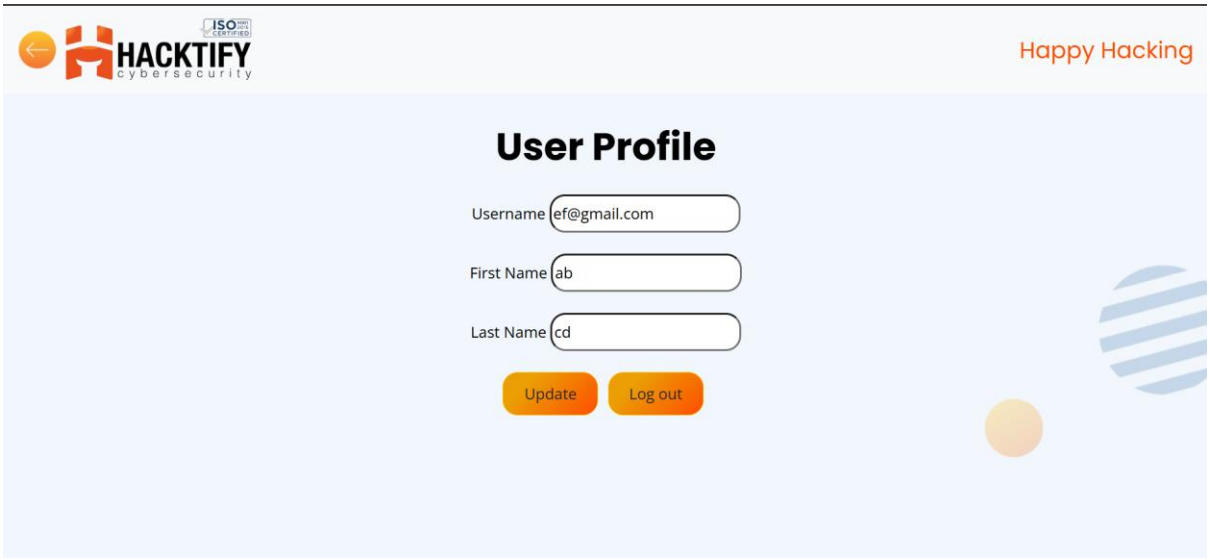


1.2. Stop polluting my params!

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	
https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1131	
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	
Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html	

Proof of Concept

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

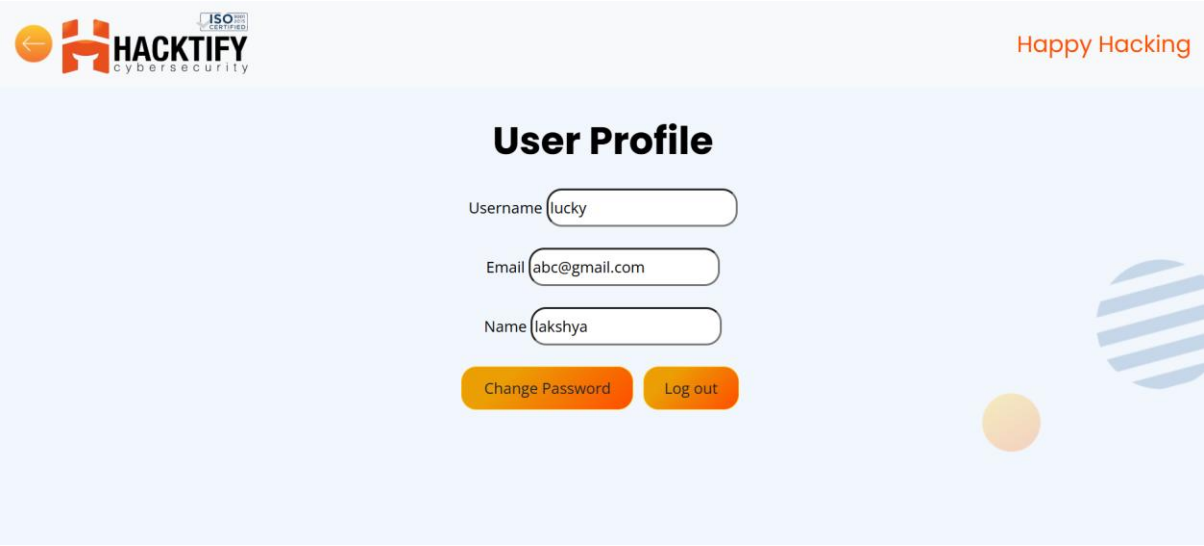


1.3. Someone changed my Password !

Reference	Risk Rating
Someone changed my Password !	High
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	
https://labs.hacktify.in/HTML/idor_lab/lab_3/profile.php	
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	
Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html	

Proof of Concept

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

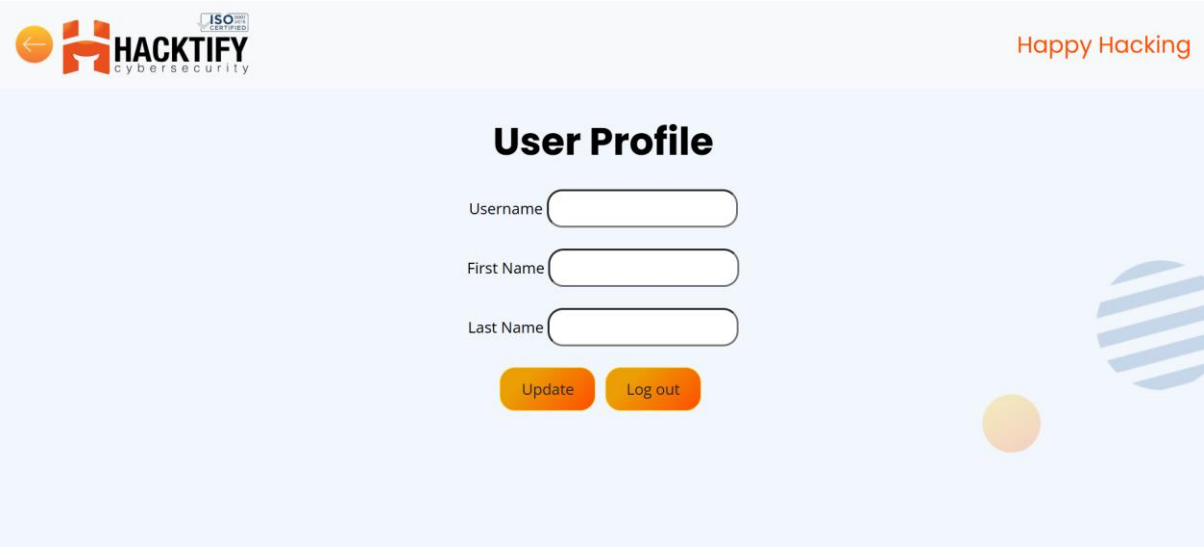


1.4. Change your methods!

Reference	Risk Rating
Change your methods!	**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	
https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=1734	
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	
Https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html	

Proof of Concept

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



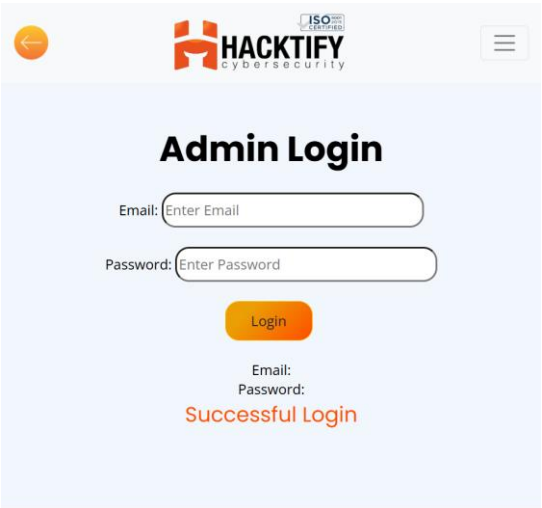
2. SQL Injection

2.1. Strings & Errors Part 1!

Reference	Risk Rating
Strings & Errors Part 1!	**Low**
Tools Used	
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_1/lab_1.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

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**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php?id=1%27%20UNION%20SELECT%20username,%20password%20FROM%20users%20--	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

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**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php?id=1%27%20UNION%20SELECT%20username,%20password%20FROM%20users%20--	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

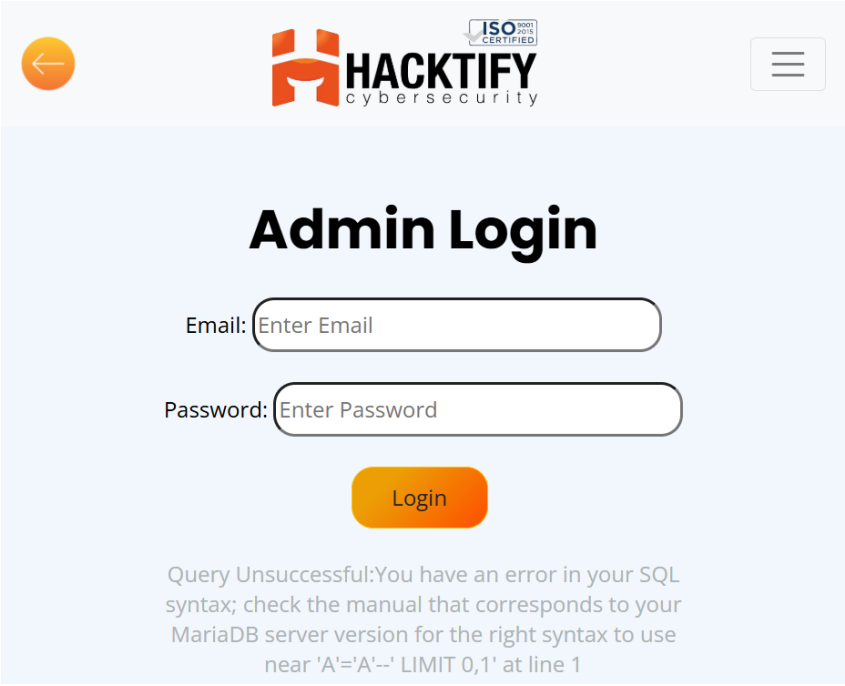
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

2.4. Let's Trick `em!

Reference	Risk Rating
Let's Trick `em!	**High**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_4/lab_4.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

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



2.5. Booleans and Blind!

Reference	Risk Rating
Booleans and Blinds	**High**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php?id=1%20AND%201=1%20--	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

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**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_6/lab_6.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

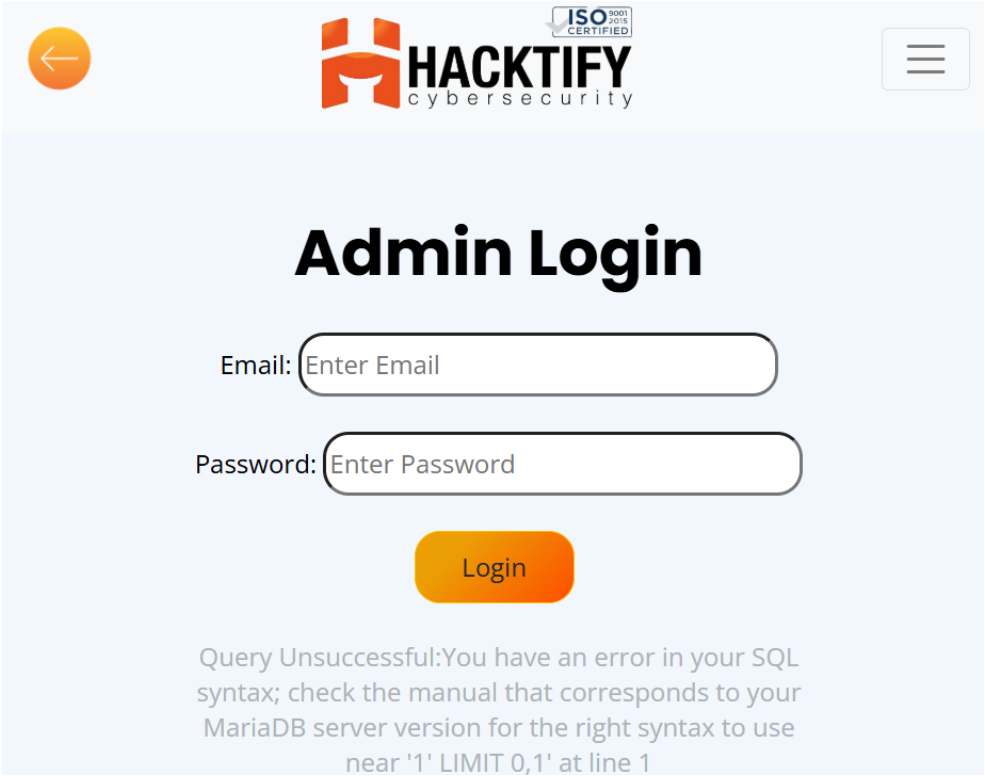
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

2.7. Errors and Post!

Reference	Risk Rating
Errors and Post!	**Low**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_7/lab_7.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

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

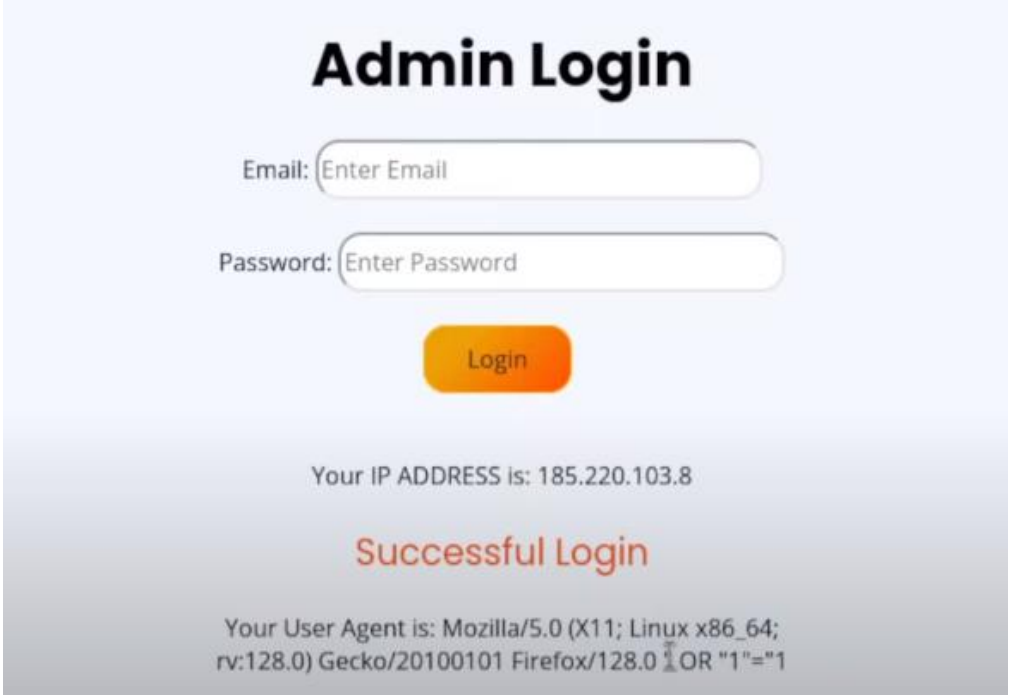


2.8. User Agents lead us!

Reference	Risk Rating
User Agents lead us!	**High**
Tools Used	
Brup Suite and 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, Brup Suite and observing responses.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

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



2.9. Referer lead us!

Reference	Risk Rating
Referer lead us!	**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	
https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

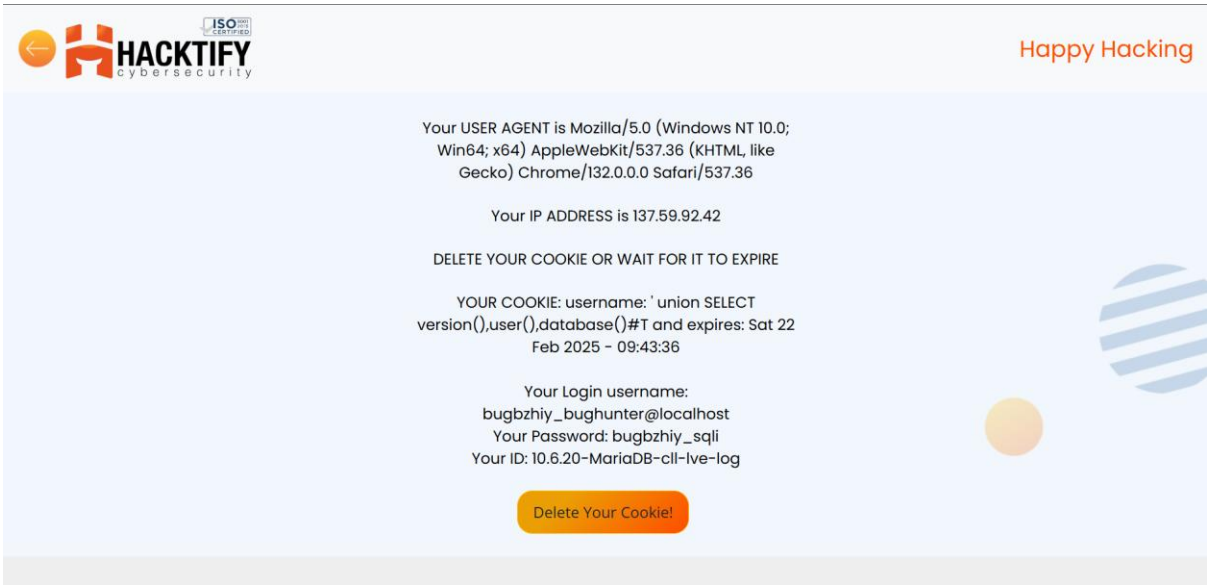
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

2.10. Oh Cookies!

Reference	Risk Rating
Oh Cookies!	**High**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

Proof of Concept

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



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	
https://labs.hacktify.in/HTML/sqli_lab/lab_11/lab_11.php?id=1&?id=0%27+union+select+1,@@version,database()--+	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

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**
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	
https://labs.hacktify.in/HTML/sqli_lab/lab_12/lab_12.php?id=1&?id=0%27+union+select+1,@@version,database()--+	
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	
Https://owasp.org/www-community/attacks/SQL_Injection	

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.**