Penetration Testing Report

Full Name : Rachael Ayomide Fanifosi
Program : HCS - Penetration Testing Internship Week-2
Date : 23/02/2025

Introduction

This report document hereby describes the proceedings and results of a SQL and IDOR lab assessment conducted against the **Week {2} Labs**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

I. 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.

II. Scope

This section defines the scope and boundaries of the project.

Application Name	{Lab 1 -SQL injection} {Lab 2 –Insecure Direct Object Reference}

III. Summary

Outlined is an (Injection) Security assessment for the **Week {2} Labs**.

Total number of Sub-labs: 16

High	Medium	Low
4	7	5

High - Number of Sub-lab with high difficulty level

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

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

1. SQL Injection

1.1. Strings & Errors Part 1!

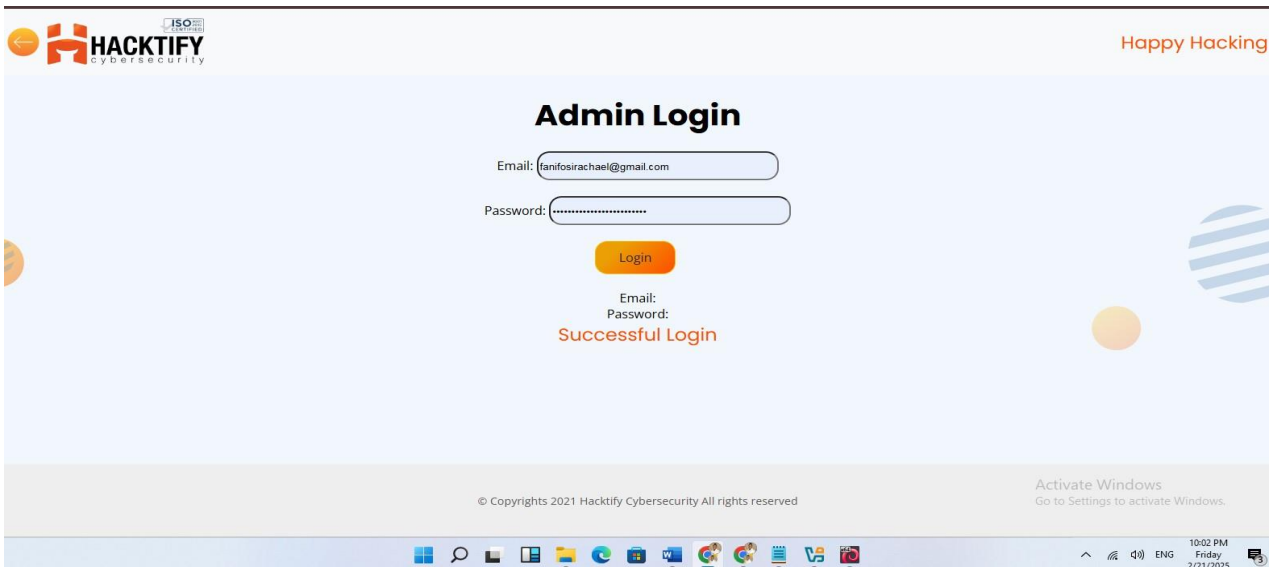
Reference	Risk Rating
Sub-lab-1: **Strings & Errors Part 1!**	**Low**
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code Email: ' OR 1=1 – Password: 'OR' '1'='1' – into the search function, which was then executed by the website.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage	
Suggested Countermeasures	
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems	

References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: Email: ‘ OR 1=1 –
Password: ‘OR’ ‘1’=’1’ –



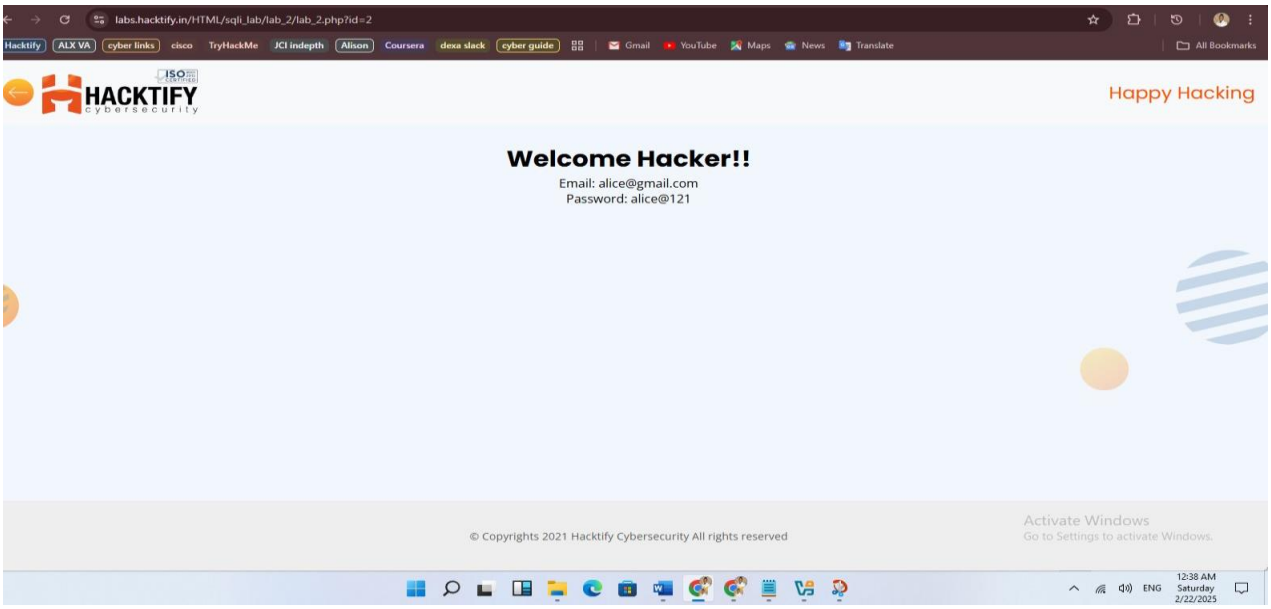
1.2. Strings & Errors Part 2!

Reference	Risk Rating
Sub-lab-2: Strings & Errors Part 2!	Low
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	

SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.
How It Was Discovered
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by adding the SQL code ?id=2 into the URL parameter, which was then executed by the website.
Vulnerable URLs
https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php?id=2
Consequences of not Fixing the Issue
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.
Payload: ?id=2



1.3. Strings & Errors Part 3!

Reference	Risk Rating
Sub-lab-3: **Strings & Errors Part 3!**	**Low**

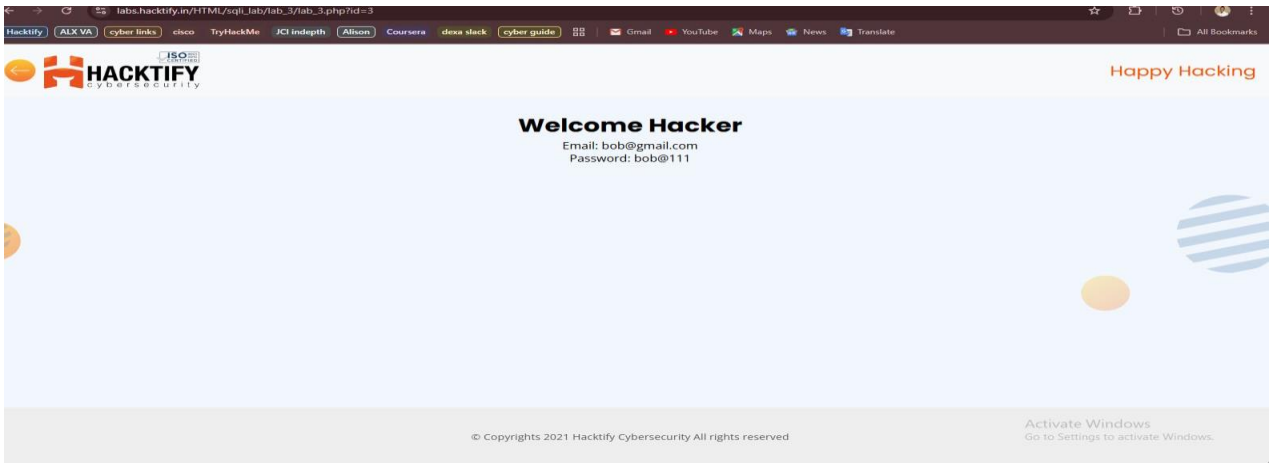
Tools Used
Browser(Google Chrome browser), manual testing
Vulnerability Description
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.
How It Was Discovered
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by adding the SQL code ?id=3 into the URL parameter, which was then executed by the website.
Vulnerable URLs
https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php?id=3
Consequences of not Fixing the Issue
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage

Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: ?id=3



1.4. Let's Trick 'em!

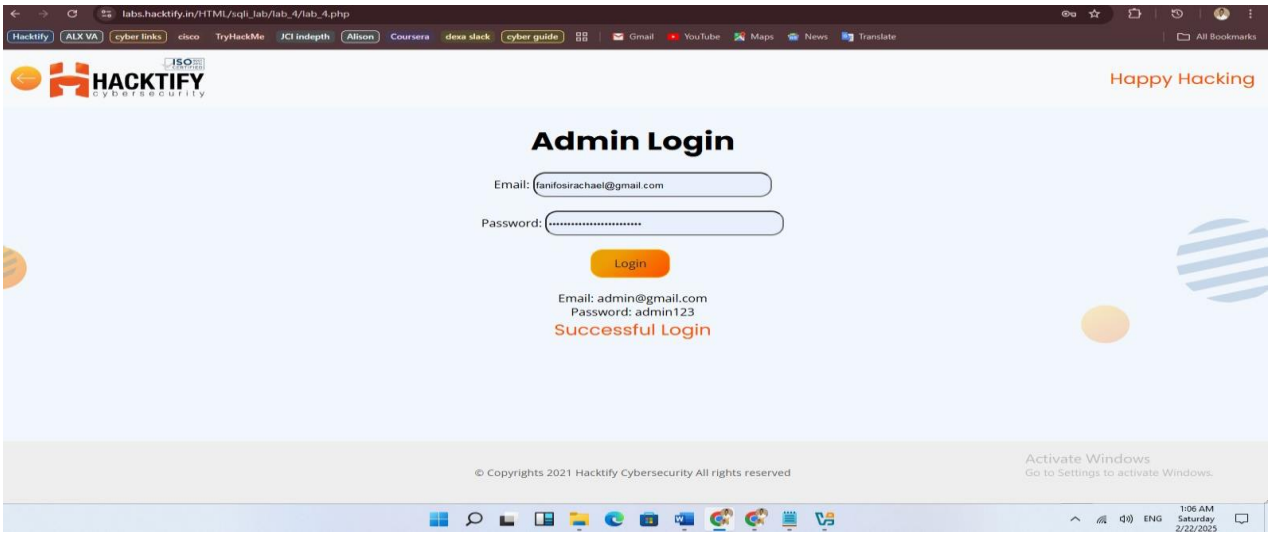
Reference	Risk Rating
Sub-lab-4: **Let's Trick 'em!**	**Medium**
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	

SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.
How It Was Discovered
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code Email: ' OR 'a'='a' – Password: 1' || '1'= '1 → into the search function, which was then executed by the website.
Vulnerable URLs
https://labs.hacktify.in/HTML/sqli_lab/lab_4/lab_4.php
Consequences of not Fixing the Issue
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: Email: ' OR 'a'='a' –
Password: 1' || '1'= '1 →



1.5. Boolean and Blind

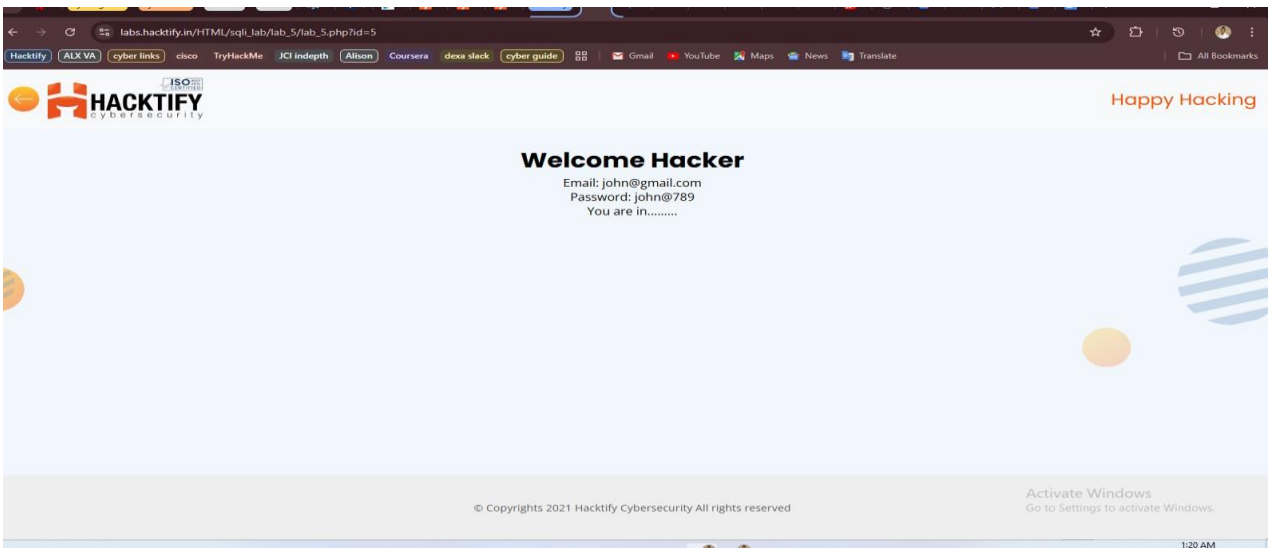
Reference	Risk Rating
Sub-lab-5: **Boolean and Blind**	**Hard**
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php?id=5	
Consequences of not Fixing the Issue	
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage	
Suggested Countermeasures	

Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: ?id=5



1.6. Error Based: Tricked

Reference	**Risk Rating**
Sub-lab-6: Error Based: Tricked	**Medium**
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	

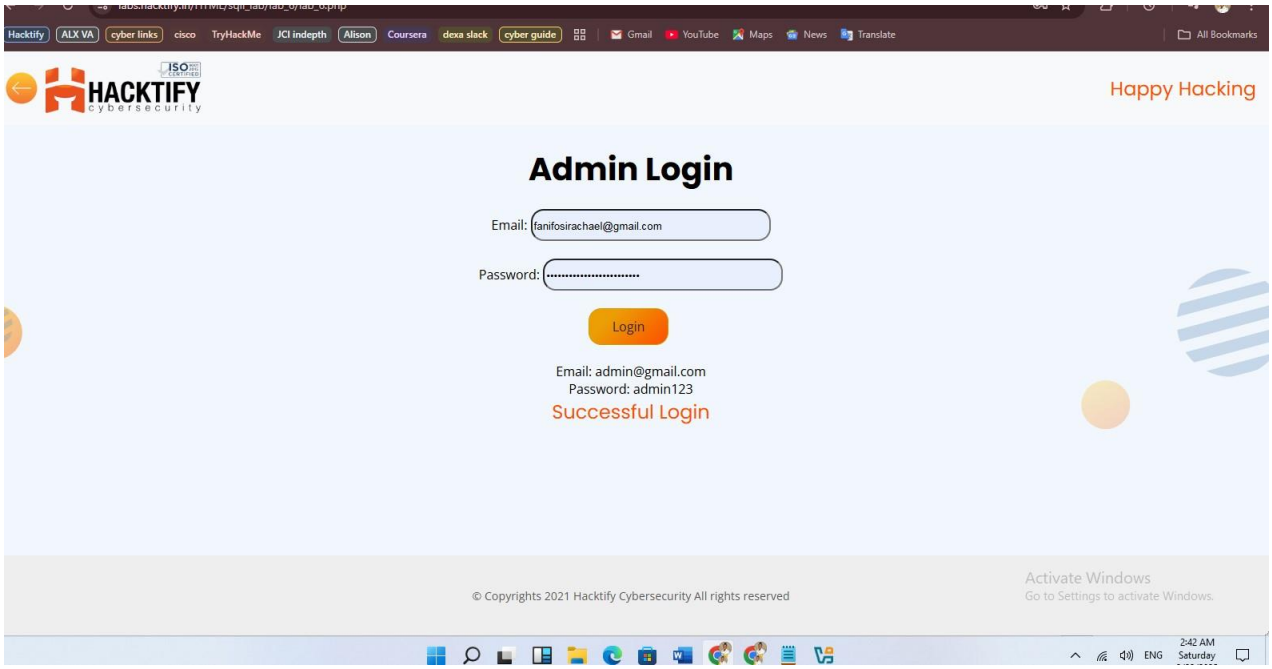
How It Was Discovered
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.
Vulnerable URLs
https://labs.hacktify.in/HTML/sqli_lab/lab_6/lab_6.php
Consequences of not Fixing the Issue
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: Email: ") or ("1")=("1

Password: ") or ("1")=("1



1.7. Errors and Post!

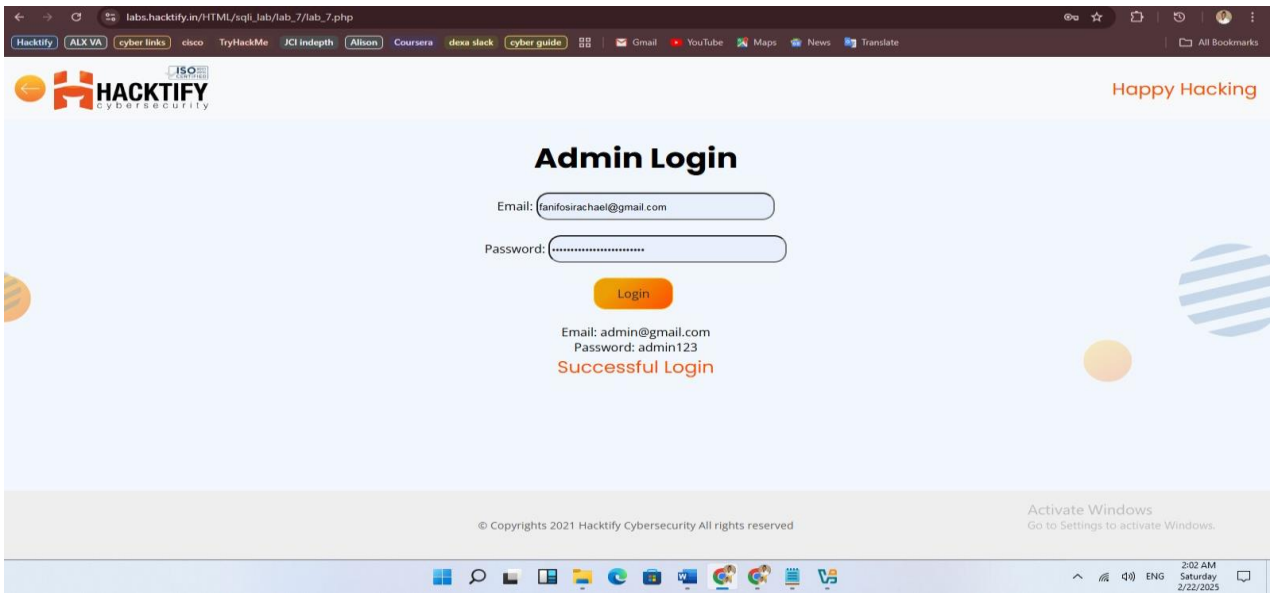
Reference	Risk Rating
Sub-lab-7: Errors and Post!	Low
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_7/lab_7.php	
Consequences of not Fixing the Issue	

Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: Email: ' OR 1=1--
Password: ‘



1.8. User Agent Lead Us!

Reference	Risk Rating

Sub-lab-8: User Agent Lead Us!	**Hard**
Tools Used	
Browser(Google Chrome browser), burpsuite,manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php	
Consequences of not Fixing the Issue	
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage	
Suggested Countermeasures	
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems	
References	
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/	

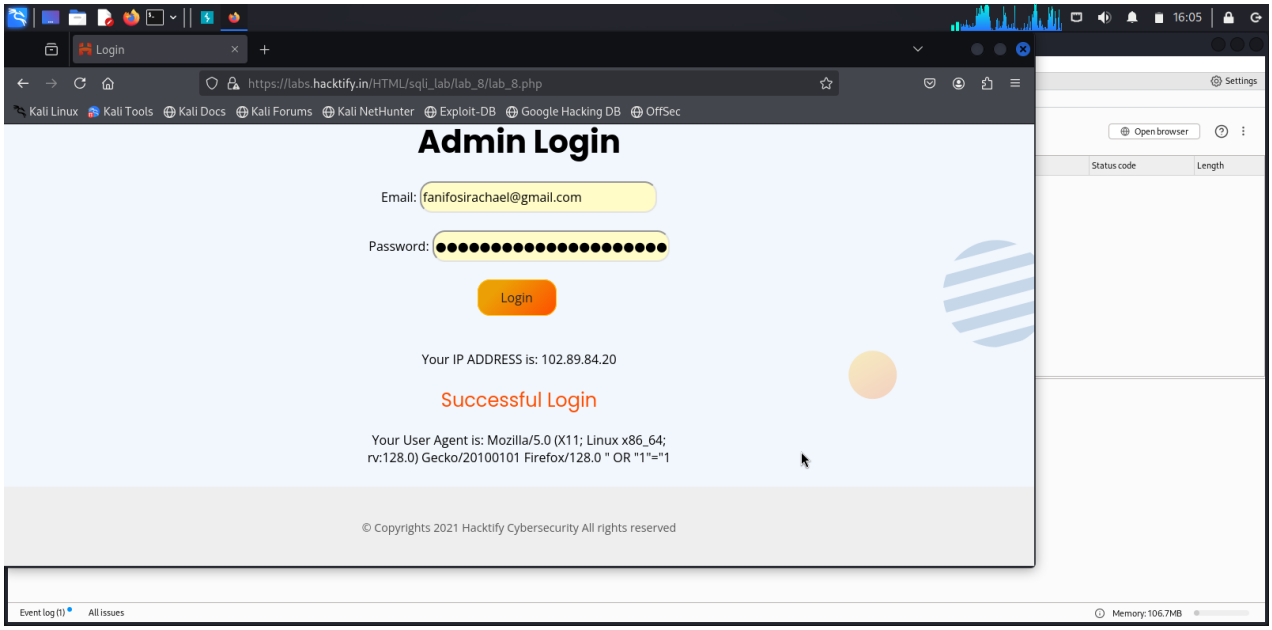
Proof of Concept

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

Payload: email:admin@gmail.com

Password: admin123

Then, using burpsuite to intercept the data, get through to user agent and modify the parameter using " OR "1'="1



1.9. Referer Lead Us!

Reference	Risk Rating
Sub-lab-9: Referer Lead Us!	Medium
Tools Used	
Browser(Google Chrome browser), burpsuite, manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.	

Vulnerable URLs
https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php
Consequences of not Fixing the Issue
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

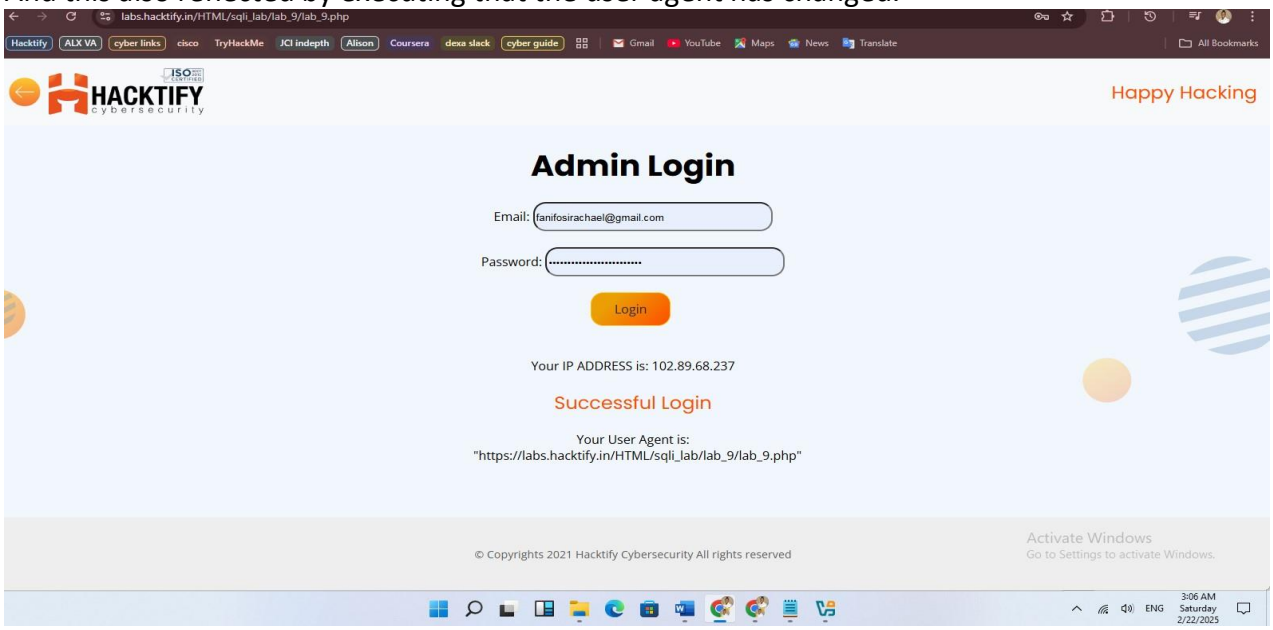
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.

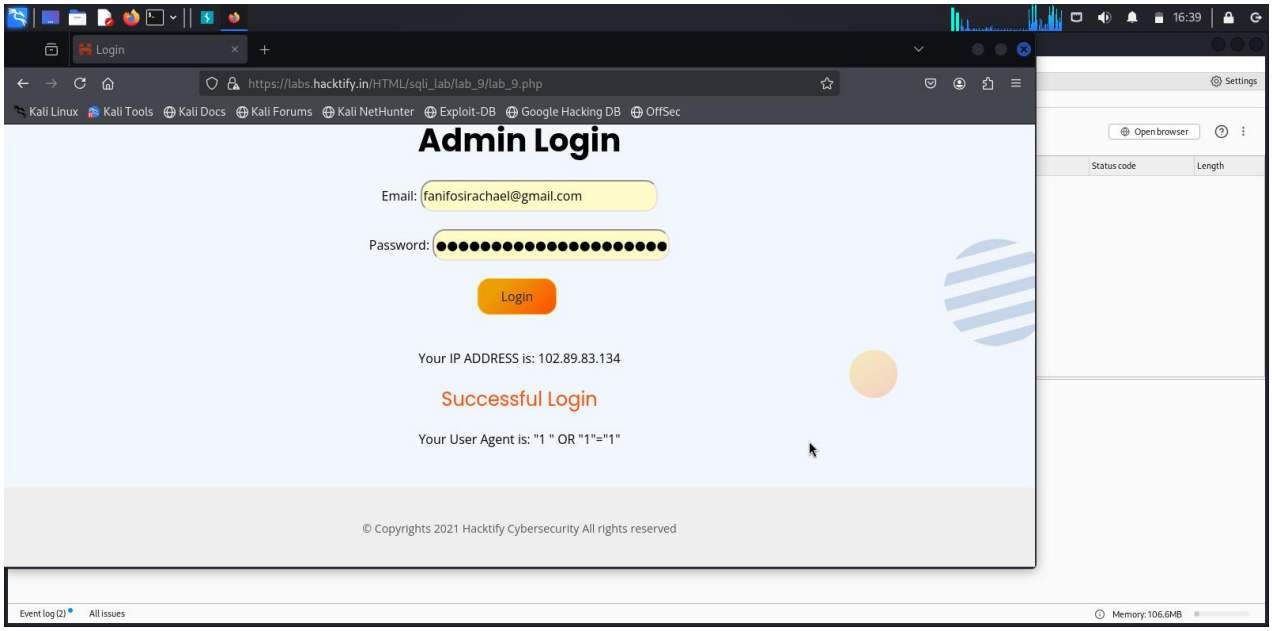
Payload: email: admin@gmail.com

Password: admin123

To login, then use burpsuite to intercept and modify the parameters of the Referer using “1 OR “1”=”1

And this also reflected by executing that the user agent has changed.





1.10. Oh Cookies!

Reference	Risk Rating
Sub-lab-10: Oh Cookies!	Hard
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php	
Consequences of not Fixing the Issue	

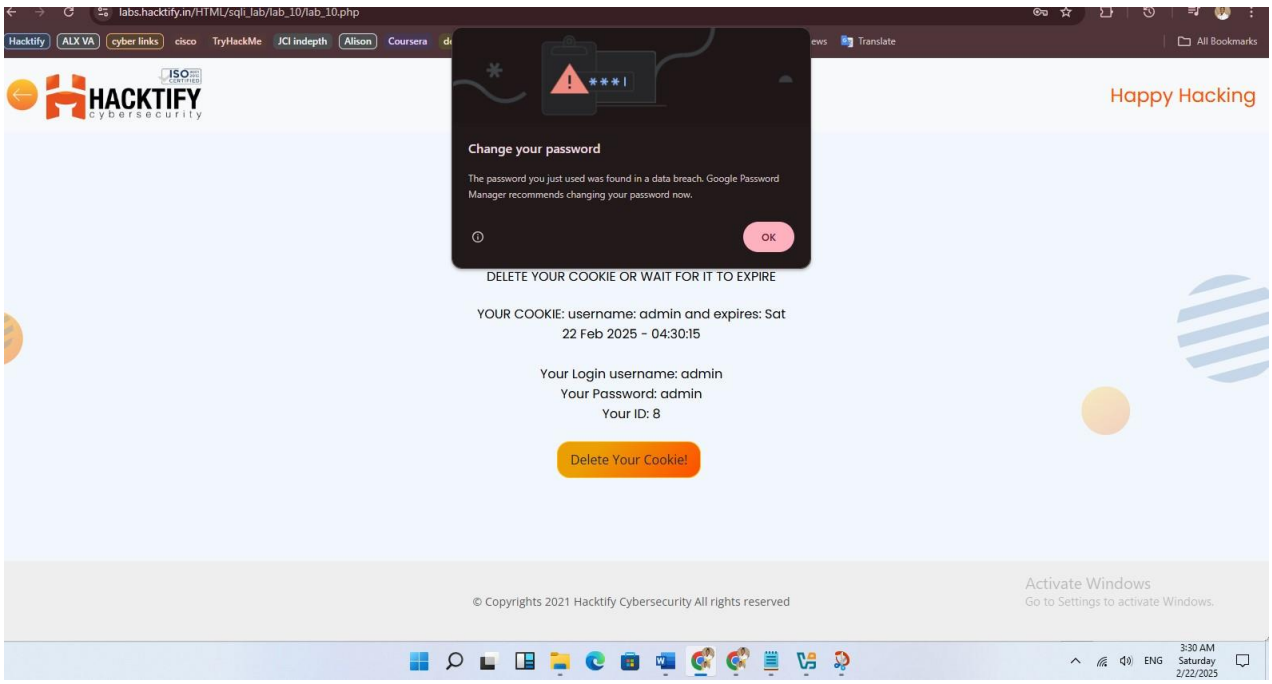
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/s

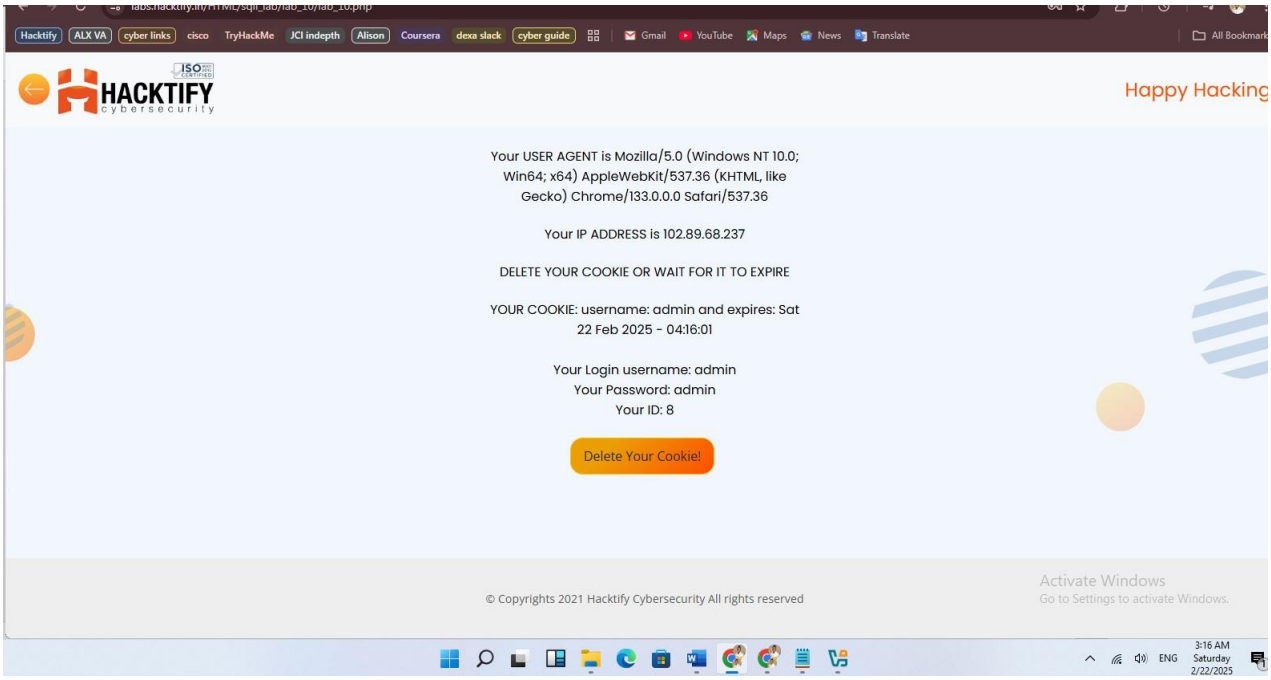
Proof of Concept

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

Payload: username: admin

Password: admin





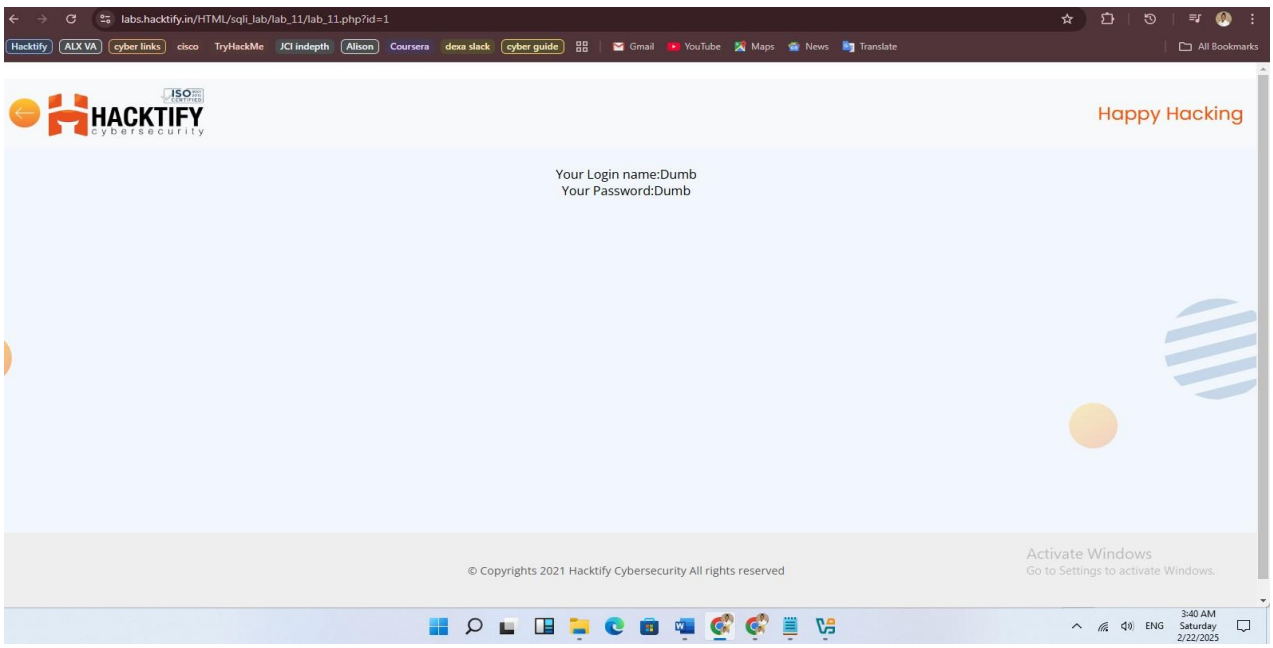
1.11. WAF's are Injected!

Reference	Risk Rating
Sub-lab-11: WAF's are Injected!	Hard
Tools Used	
Browser(Google Chrome browser), manual testing	
Vulnerability Description	
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.	
How It Was Discovered	
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_11/lab_11.php?id=1	
Consequences of not Fixing the Issue	

Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.
Payload: ?id=1



1.12. WAF's are Injected Part 2!

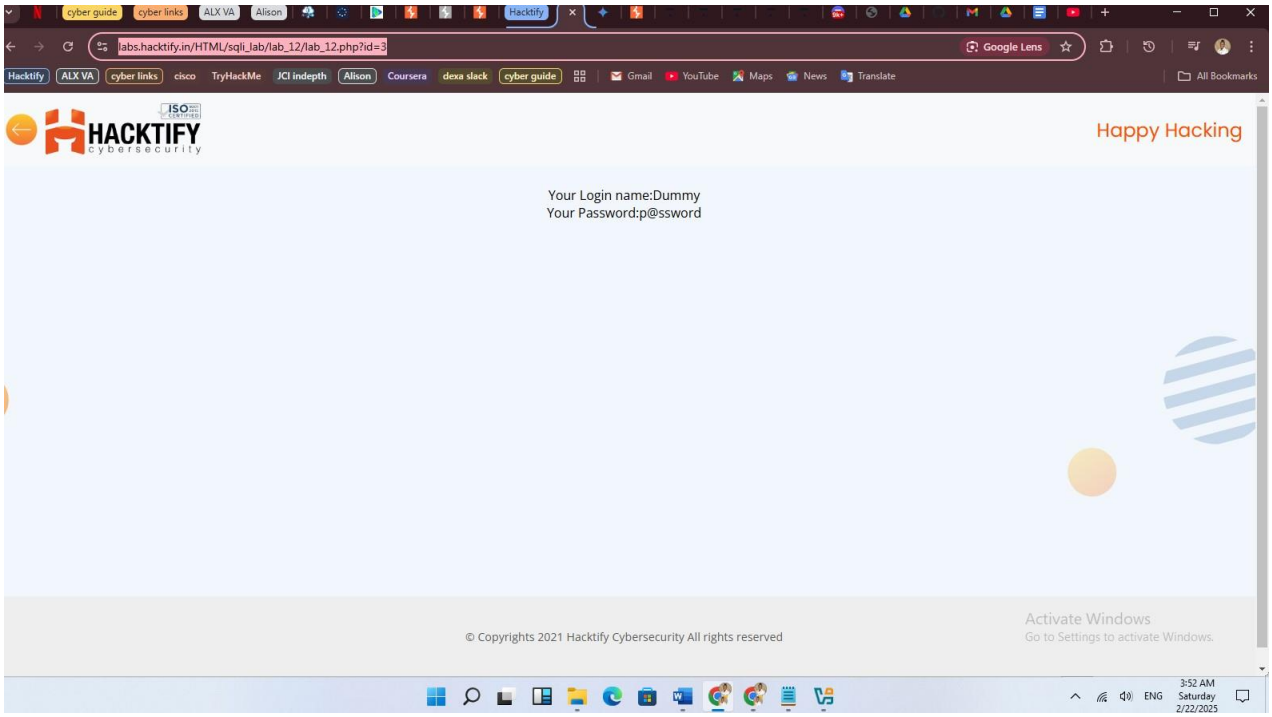
Reference	Risk Rating
Sub-lab-12: WAF's are Injected Part 2!	**Medium**

Tools Used
Browser(Google Chrome browser), manual testing
Vulnerability Description
SQL injection is a web application security vulnerability that allows attackers to inject malicious SQL code into a database, potentially leading to unauthorised data access, modification, or deletion.
How It Was Discovered
It was discovered through Manual testing of the website's search function. During testing, it was found that the website allowed user-inputted SQL injection to be executed. This was done by inputting the SQL code into the search function, which was then executed by the website.
Vulnerable URLs
https://labs.hacktify.in/HTML/sqli_lab/lab_12/lab_12.php?id=3
Consequences of not Fixing the Issue
Unauthorized data access, modification, or deletion Data breaches and leaks System compromise and takeover Financial losses and reputational damage
Suggested Countermeasures
Use prepared statements and parameterized queries Validate and sanitize user input Limit database privilege and access Regularly update and patch software and frameworks Implement web application firewalls, and Intrusion detection systems
References
https://portswigger.net/web-security/sql-injection https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

Proof of Concept

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

Payload: ?id=3



2. Insecure Direct Object Reference

2.1. Give me my amount!!

Reference	Risk Rating
Sub-lab-1: Give me my amount!!	Low
Tools Used	
Google Chrome Browser	
Vulnerability Description	
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. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=235	

Consequences of not Fixing the Issue
Unauthorized data access Sensitive information exposure Data breaches Account takeover Fraudulent activities Data manipulation
Suggested Countermeasures
Developers should avoid displaying private object references such as keys or file names. Validation of parameters should be properly implemented. Verification of all the referenced objects should be checked. Tokens should be generated in such a way that it can only be mapped to the user and is not public. Ensure that queries are scoped to the owner of the resource. Avoid things like using UUIDs (Universally unique identifier) over Sequential IDs as UUIDs often let IDOR vulnerabilities go undetected.
References
IDOR by Port Swigger: https://portswigger.net/web-security/access-control/idor IDOR by OWASP :https://cheat sheetseries .owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html How to find IDOR's by Bugcrowd: https://www.bugcrowd.com/blog/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/

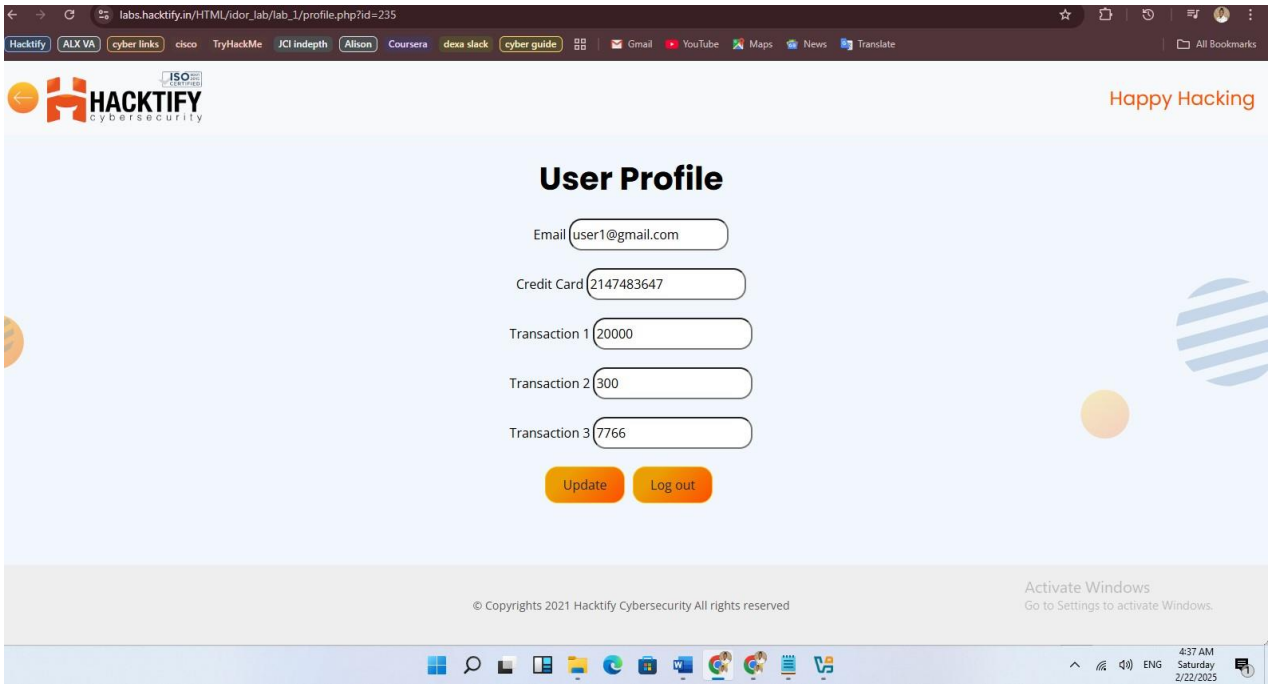
Proof of Concept

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

Payload: Registered an account - abcd@gmail.com

Login with it, and had access to my user profile

i changed the id from 23 to 235



2.2. Stop Polluting my param

Reference	Risk Rating
Sub-lab-2: Stop Polluting my param	medium
Tools Used	
Google Chrome Browser	
Vulnerability Description	
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. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	
labs.hacktify.in/HTML/idor_lab/lab2/profile.php?id=45	
Consequences of not Fixing the Issue	

Unauthorized data access
Sensitive information exposure
Data breaches
Account takeover
Fraudulent activities
Data manipulation

Suggested Countermeasures

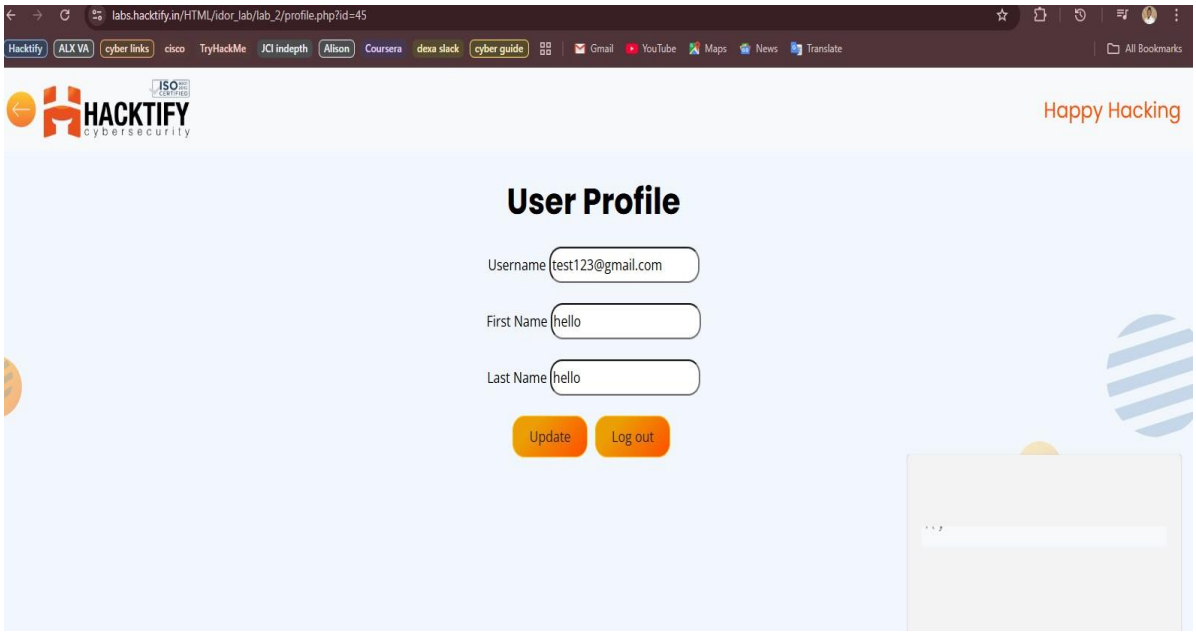
Developers should avoid displaying private object references such as keys or file names.
Validation of parameters should be properly implemented.
Verification of all the referenced objects should be checked.
Tokens should be generated in such a way that it can only be mapped to the user and is not public.
Ensure that queries are scoped to the owner of the resource.
Avoid things like using UUIDs (Universally unique identifier) over Sequential IDs as UUIDs often let IDOR vulnerabilities go undetected.

References

IDOR by Port Swigger: https://portswigger.net/web-security/access-control/idor
IDOR by OWASP :https://cheat sheetseries .owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
How to find IDOR's by Bugcrowd: https://www.bugcrowd.com/blog/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab
Payload: registered an account - waq@gmaill.com
Login, to the new account
Changed the username id parameter to 45 from 1395
Had access to the account.



2.3. Someone Changed my Password

Reference	Risk Rating
Sub-lab-3: Someone changed my password	**medium**
Tools Used	
Google Chrome Browser	
Vulnerability Description	
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. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_3/profile.php https://labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=john	
Consequences of not Fixing the Issue	

Unauthorized data access
Sensitive information exposure
Data breaches
Account takeover
Fraudulent activities
Data manipulation

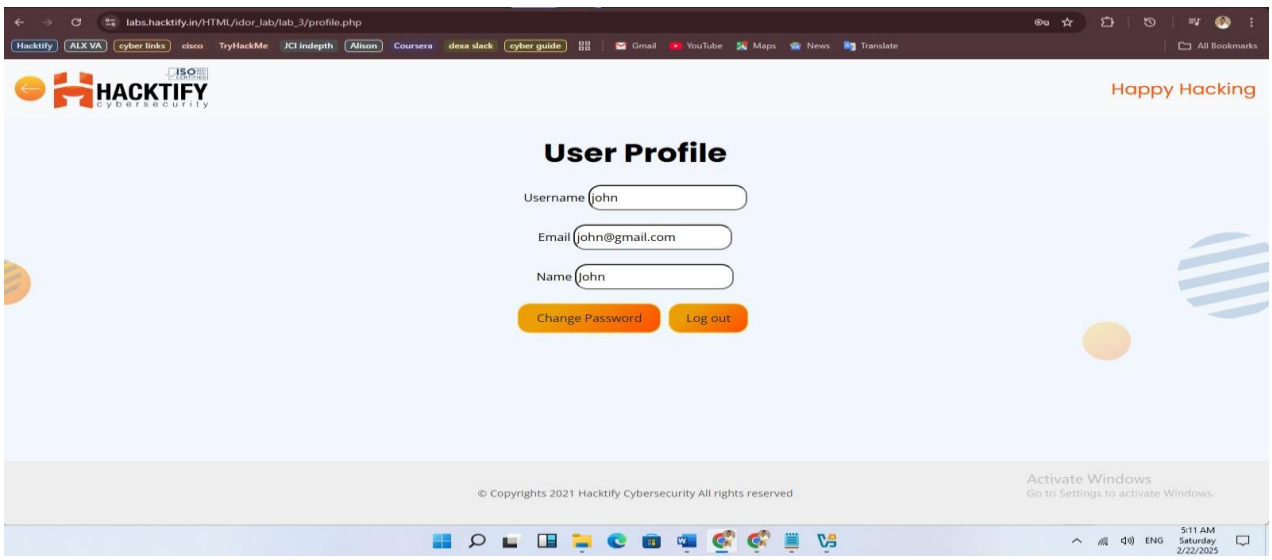
Suggested Countermeasures

Developers should avoid displaying private object references such as keys or file names.
Validation of parameters should be properly implemented.
Verification of all the referenced objects should be checked.
Tokens should be generated in such a way that it can only be mapped to the user and is not public.
Ensure that queries are scoped to the owner of the resource.
Avoid things like using UUIDs (Universally unique identifier) over Sequential IDs as UUIDs often let IDOR vulnerabilities go undetected.

References

IDOR by Port Swigger: https://portswigger.net/web-security/access-control/idor
IDOR by OWASP :https://cheat sheetseries .owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
How to find IDOR's by Bugcrowd: https://www.bugcrowd.com/blog/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/

Proof of Concept
Payload: registered an account - waq@gmaill.com
Login, to the new account
Changed the username id parameter to john from fray
Had access to the account and proceeded to change the password.



2.4. Changed your methods!

Reference	Risk Rating
Sub-lab-4: Change your methods!	Medium

Tools Used

Google Chrome Browser

Vulnerability Description

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. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability.

How It Was Discovered

Manual testing

Vulnerable URLs

https://labs.hacktify.in/HTML/idor_lab/lab_4/lab_4.php?email=qaz%40gmail.com&pwd=12345&submit=

Consequences of not Fixing the Issue

Unauthorized data access
Sensitive information exposure
Data breaches
Account takeover
Fraudulent activities
Data manipulation

Suggested Countermeasures

Developers should avoid displaying private object references such as keys or file names.
Validation of parameters should be properly implemented.
Verification of all the referenced objects should be checked.
Tokens should be generated in such a way that it can only be mapped to the user and is not public.
Ensure that queries are scoped to the owner of the resource.
Avoid things like using UUIDs (Universally unique identifier) over Sequential IDs as UUIDs often let IDOR vulnerabilities go undetected.

References
IDOR by Port Swigger: https://portswigger.net/web-security/access-control/idor IDOR by OWASP :https://cheat sheetseries .owasp.org/cheatsheets/Insecure _Direct_Object _Reference_Prevention_Cheat_Sheet.html How to find IDOR's by Bugcrowd: https://www.bugcrowd.com/blog/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/

Proof of Concept

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

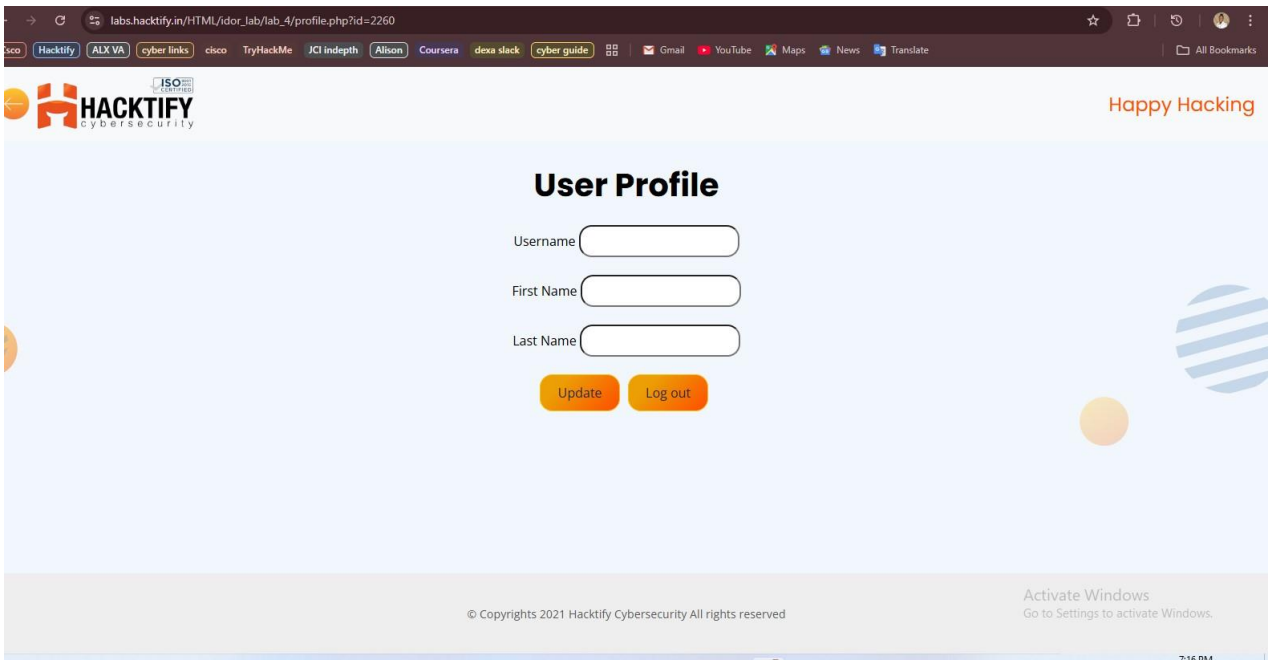
Payload: registered with zo@gmail.com, looking at the parameters, the id issued was 2260

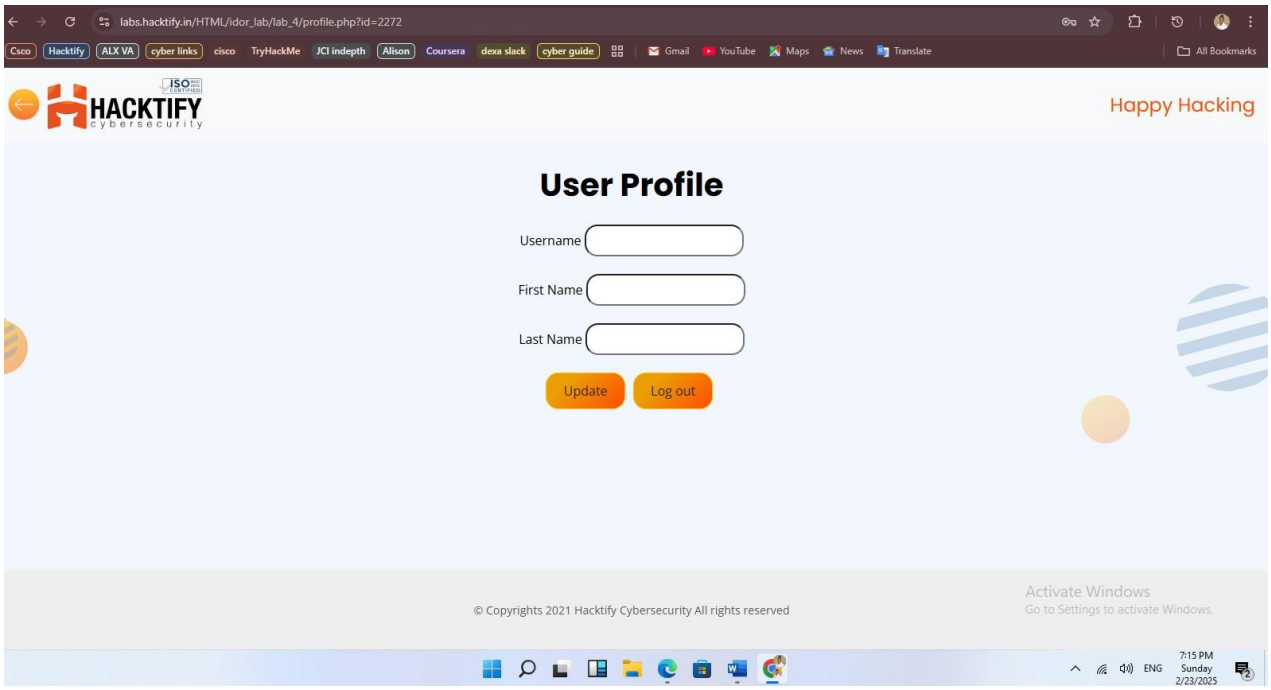
Changed the id to another number, and it changed

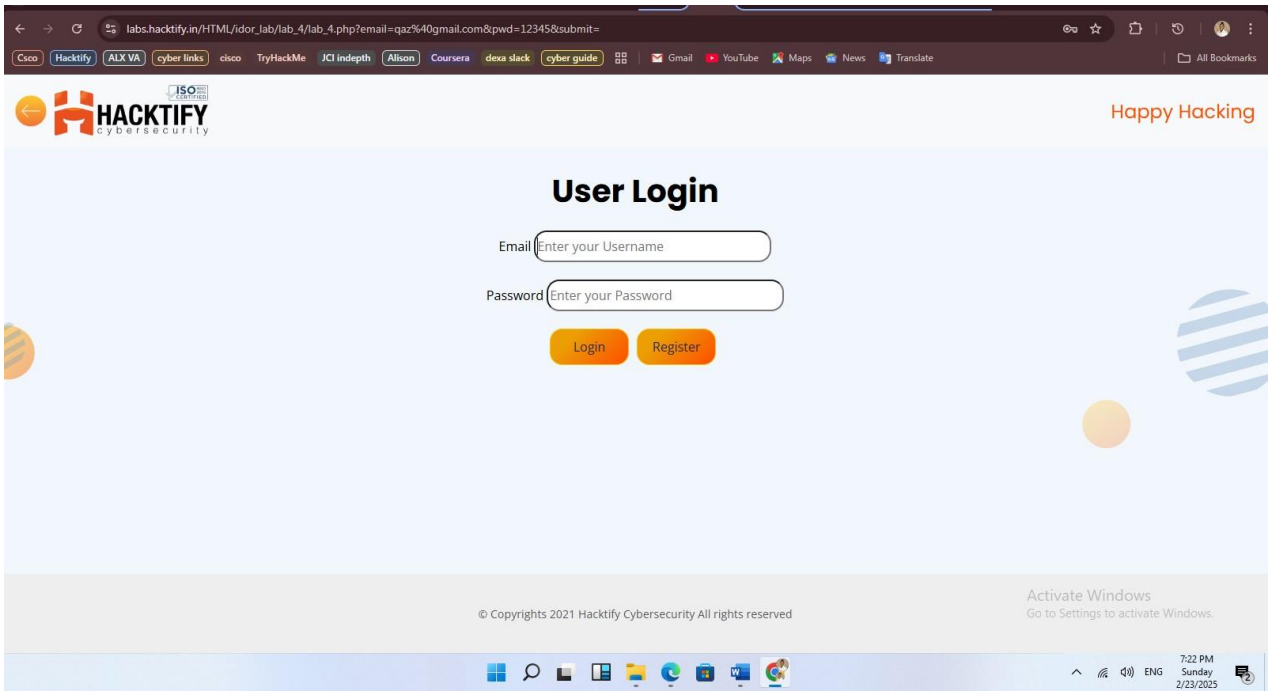
Register with another email qaz@gmail.com, the id issued was 2297

Changed the id to 2260, proceeded to change the username, first and last name before updating it.

Then i tried loggin with the updated version of the changed parameter.




labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=2272
HACKTIFY
cybersecurity
Happy Hacking
User Profile
Username
First Name
Last Name
Update
Log out
© Copyrights 2021 Hacktify Cybersecurity All rights reserved
Activate Windows
Go to Settings to activate Windows.


labs.hacktify.in/HTML/idor_lab/lab_4/lab_4.php?email=qaz%40gmail.com&pwd=12345&submit=
HACKTIFY
cybersecurity
Happy Hacking
User Login
Email
Enter your Username
Password
Enter your Password
Login
Register
© Copyrights 2021 Hacktify Cybersecurity All rights reserved
Activate Windows
Go to Settings to activate Windows.