Penetration Testing Report

Full Name: Aurelia Anthony
Program: HCPT
Date: 20/02/2025

Introduction

This report 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 Reference and SQL Injections**

3. Summary

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

Total number of Sub-labs: 16 Sub-labs

High	Medium	Low
Five (5)	Six (6)	Five (5)

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

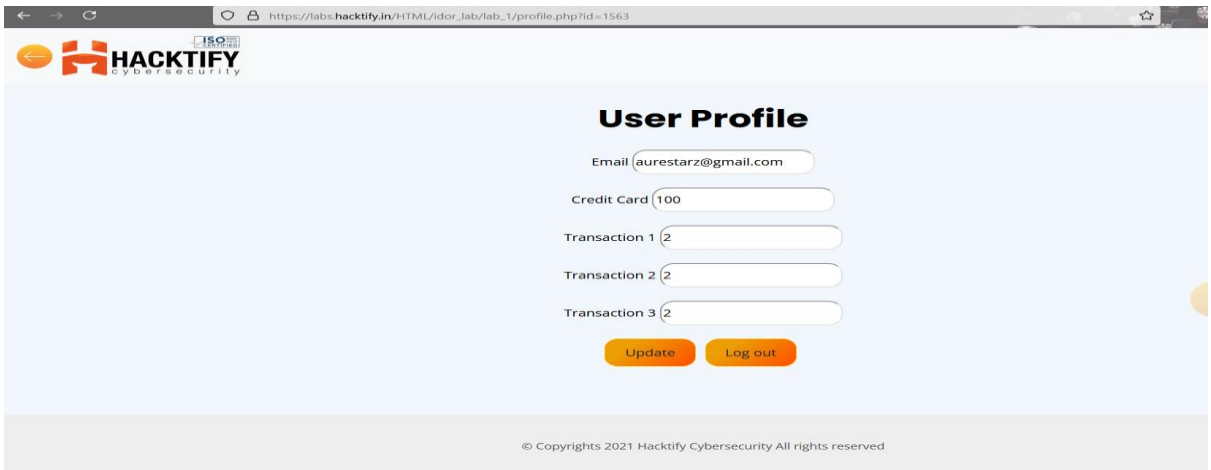
1. Insecure Direct Object References Lab

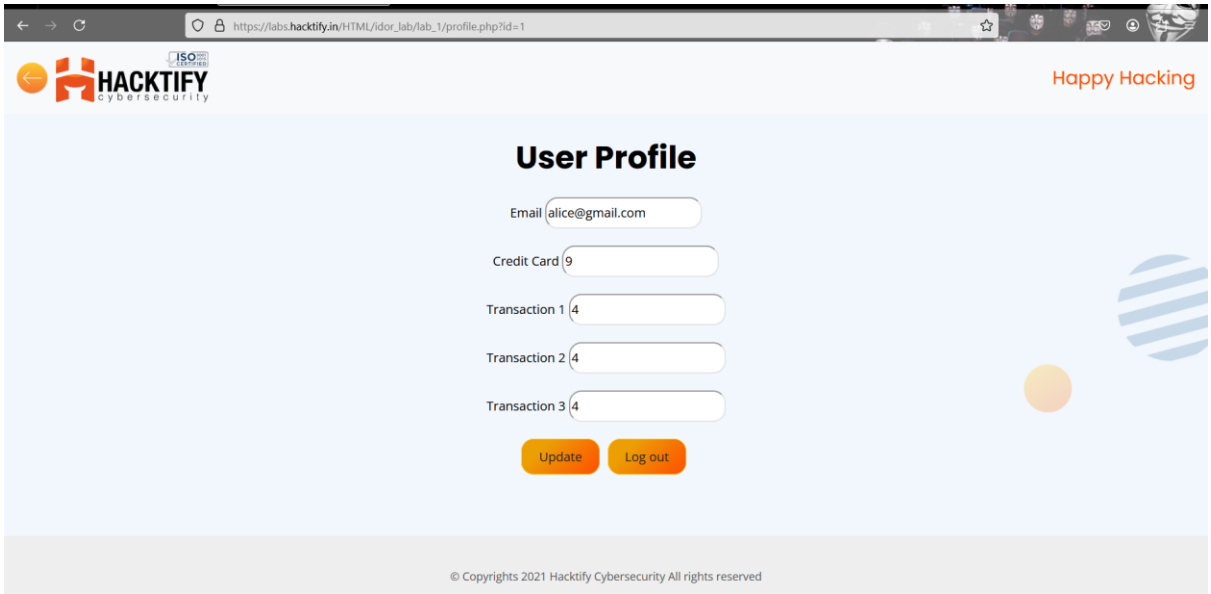
1.1. Give me my money!!

Reference	Risk Rating
Give me my money!!	**Low**
Tools Used	
ID payload	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a type of security vulnerability that occurs when an application provides direct access to objects based on user-supplied input. This allows attackers to bypass authorization and access resources directly by manipulating the input, such as changing the value of a parameter in a URL or form.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=1	
Consequences of not Fixing the Issue	
Attackers can access sensitive information such as personal data, financial records, or confidential documents by manipulating object references	
Suggested Countermeasures	
Use indirect references (e.g., mapping user input to internal objects) to avoid exposing direct object references	
References	
https://portswigger.net/web-security/access-control/idor	

Proof of Concept

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





1.2. Stop pulling my params!

Reference	Risk Rating
Stop pulling my params!	Low
Tools Used	
ID payloads	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a type of security vulnerability that occurs when an application provides direct access to objects based on user-supplied input.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1	
Consequences of not Fixing the Issue	
Attackers can modify or delete sensitive data by exploiting IDOR vulnerabilities.	
Suggested Countermeasures	
Implement proper authorization checks to ensure users can only access objects they are permitted to.	
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

https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1065

HACKTIFY cybersecurity

User Profile

Username aurestarz@gmail.com

First Name Aurelia

Last Name Anthony

Update Log out

https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=1

HACKTIFY cybersecurity

User Profile

Username alice@gmail.com

First Name alice

Last Name xyz

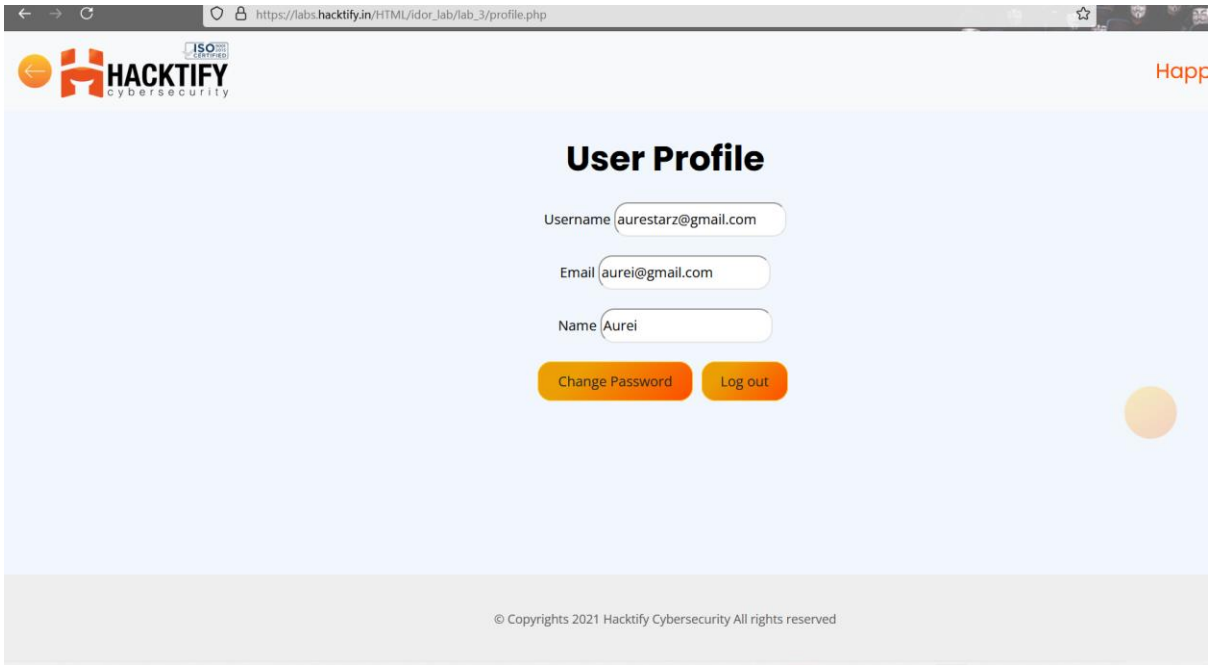
Update Log out

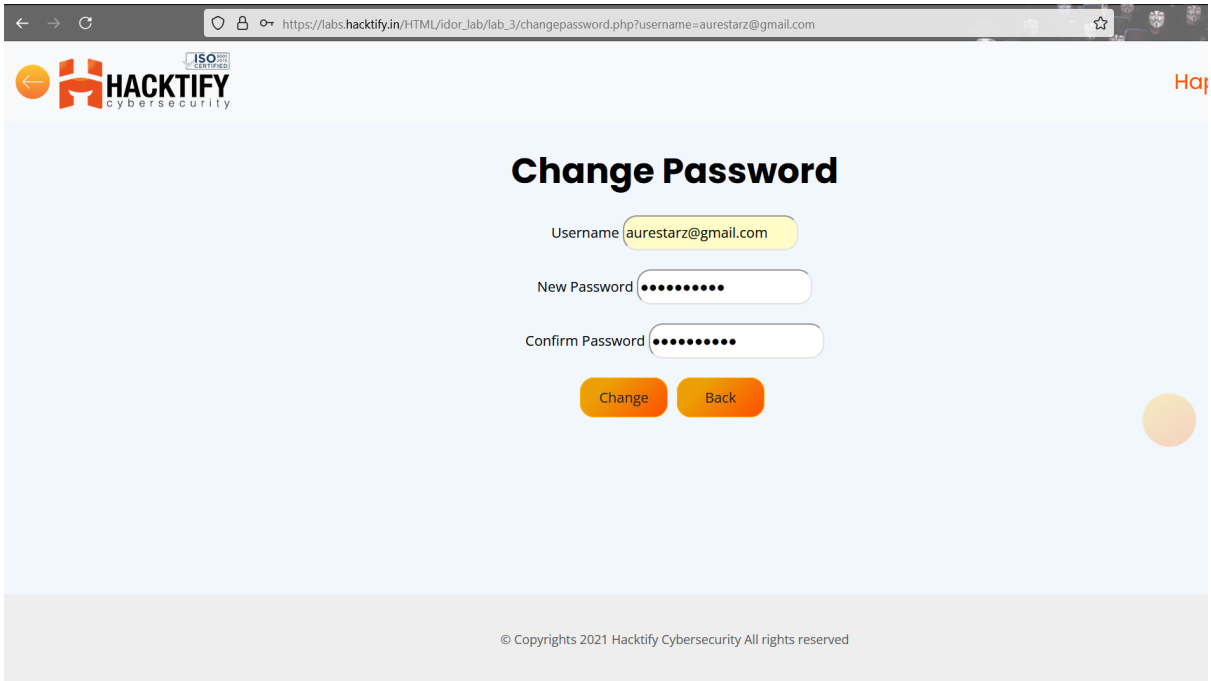
1.3. Someone changed my password

Reference	Risk Rating
Someone changed my password	Low
Tools Used	
ID payload	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a type of security vulnerability that occurs when an application provides direct access to objects based on user-supplied input.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=aurestarz@gmail.com	
Consequences of not Fixing the Issue	
Attackers can gain access to administrative or higher-privileged accounts by manipulating object references.	
Suggested Countermeasures	
Validate and sanitize all user inputs to prevent manipulation.	
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





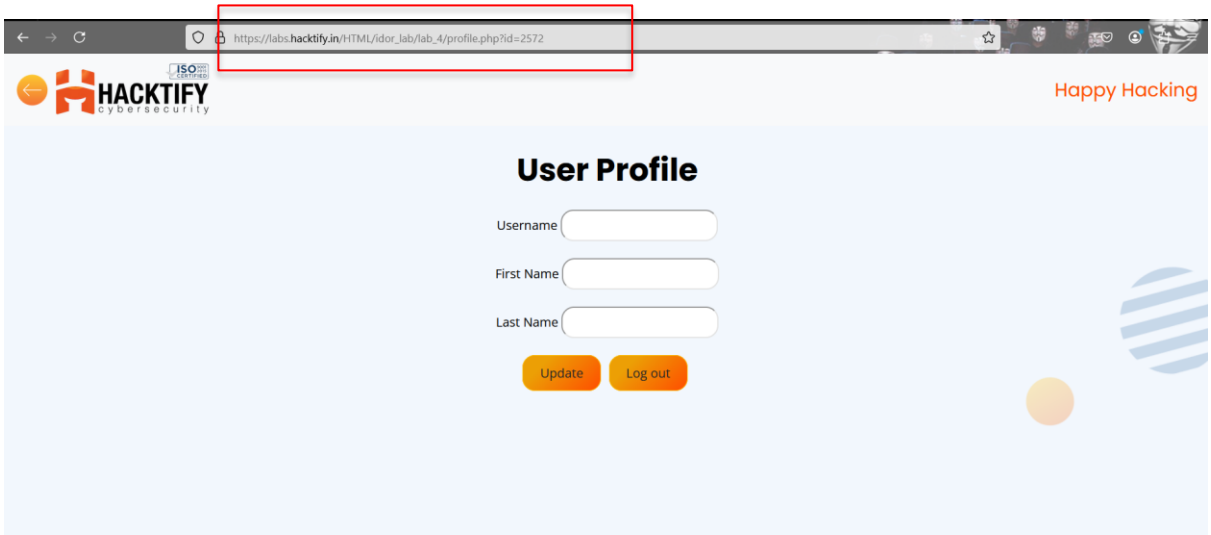
1.4. Change your methods!

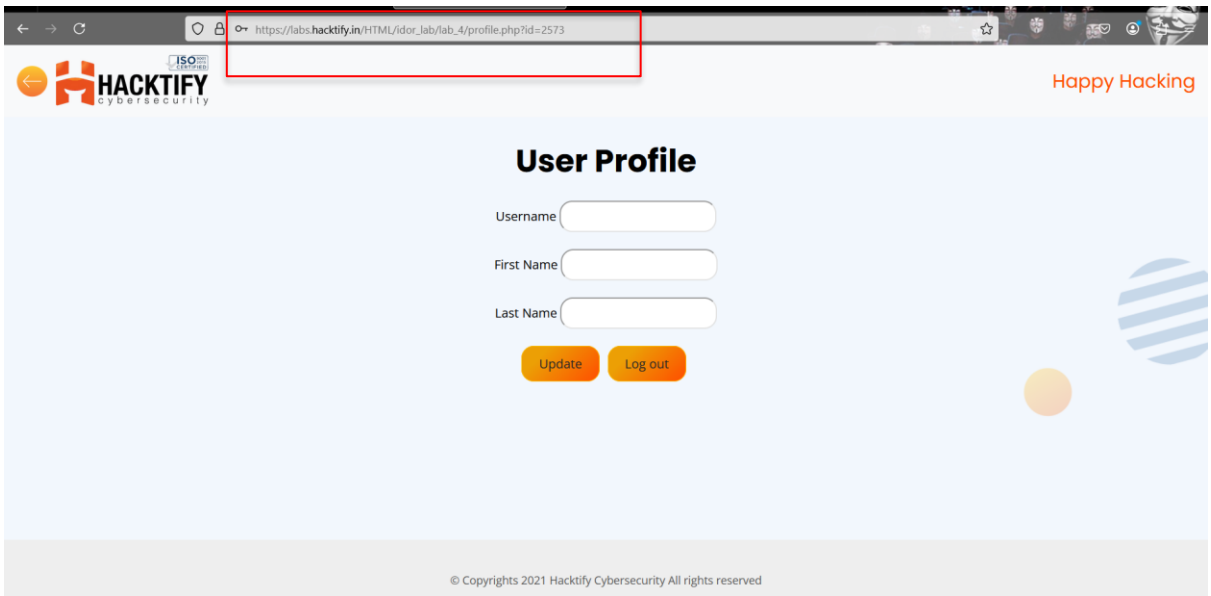
Reference	Risk Rating
Change your methods!	High
Tools Used	
ID payload	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a type of security vulnerability that occurs when an application provides direct access to objects based on user-supplied input.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=2573 https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=2572	
Consequences of not Fixing the Issue	
Attackers can gain access to administrative accounts by manipulating object references.	
Suggested Countermeasures	
Implement proper authorization checks to ensure users can only access objects they are permitted to.	
References	
https://portswigger.net/web-security/access-control/idor	

Proof of Concept

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

- Create two accounts.
- Note the difference in the id parameter.
- Enter the other account's id and update data. If you try to enter the same account, you will find that the log in is no longer working because it was updated with another account.





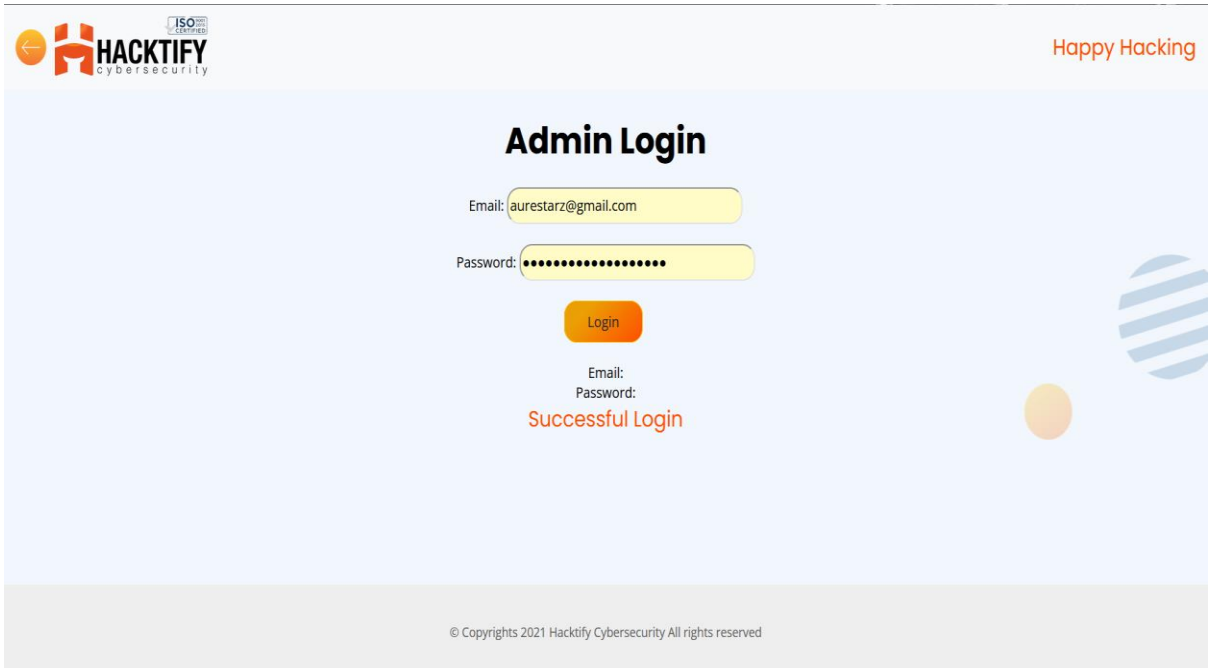
2. SQL INJECTION

2.1. Strings & Errors Part 1!

Reference	Risk Rating
Strings & Errors Part 1!	Low
Tools Used	
SQL Payload	
Vulnerability Description	
SQL Injection is a type of security vulnerability that occurs when an attacker is able to manipulate an application's SQL queries by injecting malicious SQL code. This happens when user input is not properly validated or sanitized before being included in SQL queries.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_1/lab_1.php?admin%27%20OR%201=1--	
Consequences of not Fixing the Issue	
Attackers can bypass login mechanisms and gain unauthorized access to systems.	
Suggested Countermeasures	
Validate and sanitize all user inputs to ensure they conform to expected formats.	
References	
https://portswigger.net/web-security/sql-injection	

Proof of Concept

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

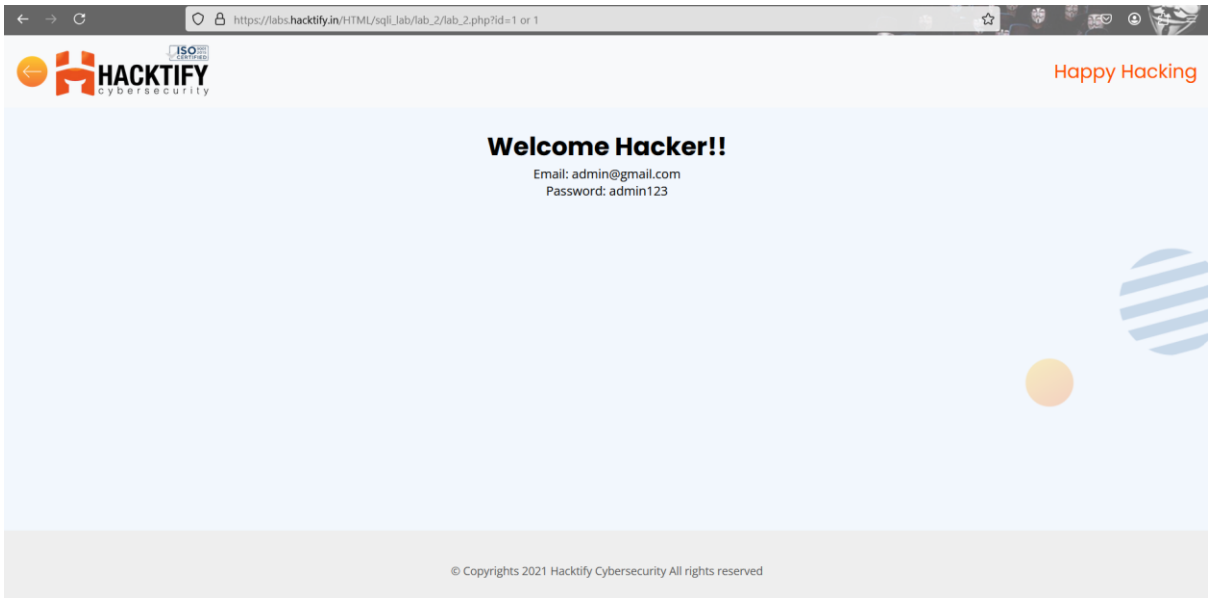


2.2. Strings & Errors Part 2!

Reference	Risk Rating
Strings & Errors Part 2!	**Low**
Tools Used	
SQL Payload	
Vulnerability Description	
SQL Injection is a type of security vulnerability that occurs when an attacker can manipulate an application's SQL queries by injecting malicious SQL code. This happens when user input is not properly validated or sanitized before being included in SQL queries.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php?id=1%20or%201	
Consequences of not Fixing the Issue	
Attackers can modify, delete, or insert data into the database, leading to data corruption or loss.	
Suggested Countermeasures	
Separate SQL code from user input using parameterized queries or prepared statements.	
References	
https://www.imperva.com/learn/application-security/sql-injection-sqli/	

Proof of Concept

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

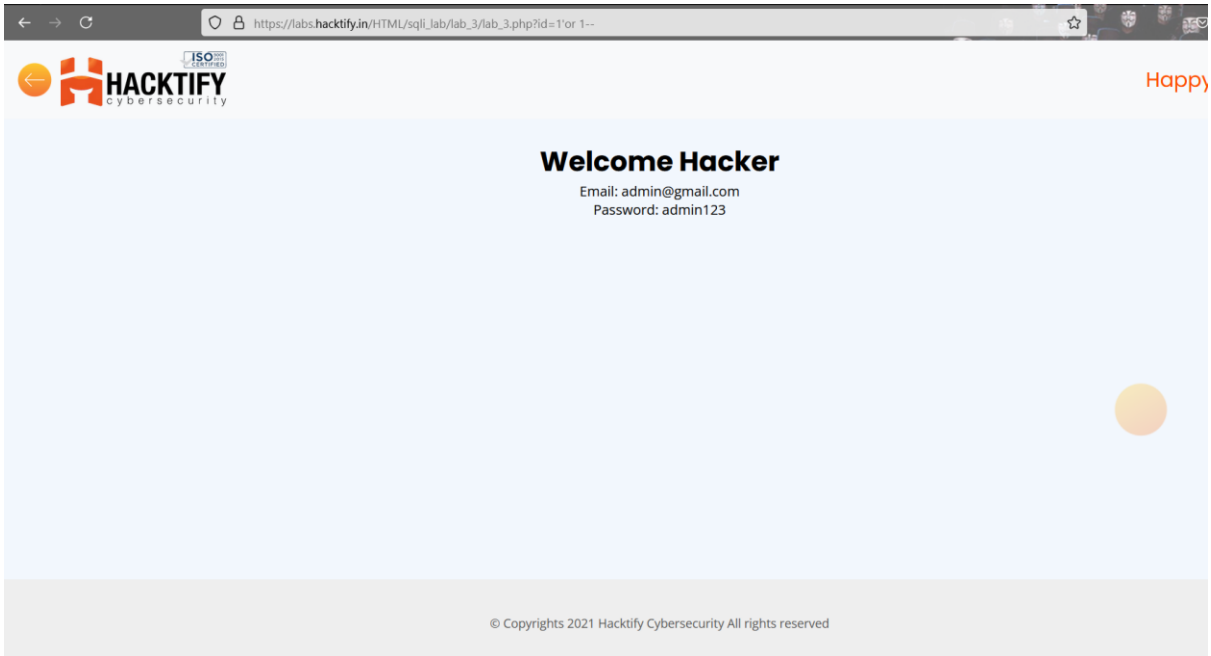


2.3. Strings & Errors Part 3!

Reference	Risk Rating
Strings & Errors Part 3!	**Medium**
Tools Used	
SQL Payload	
Vulnerability Description	
It is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php?id=1%27or%201--	
Consequences of not Fixing the Issue	
Attackers can modify, delete, or insert data into the database, leading to data corruption or loss.	
Suggested Countermeasures	
Separate SQL code from user input using parameterized queries or prepared statements.	
References	
https://www.imperva.com/learn/application-security/sql-injection-sqli/	

Proof of Concept

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

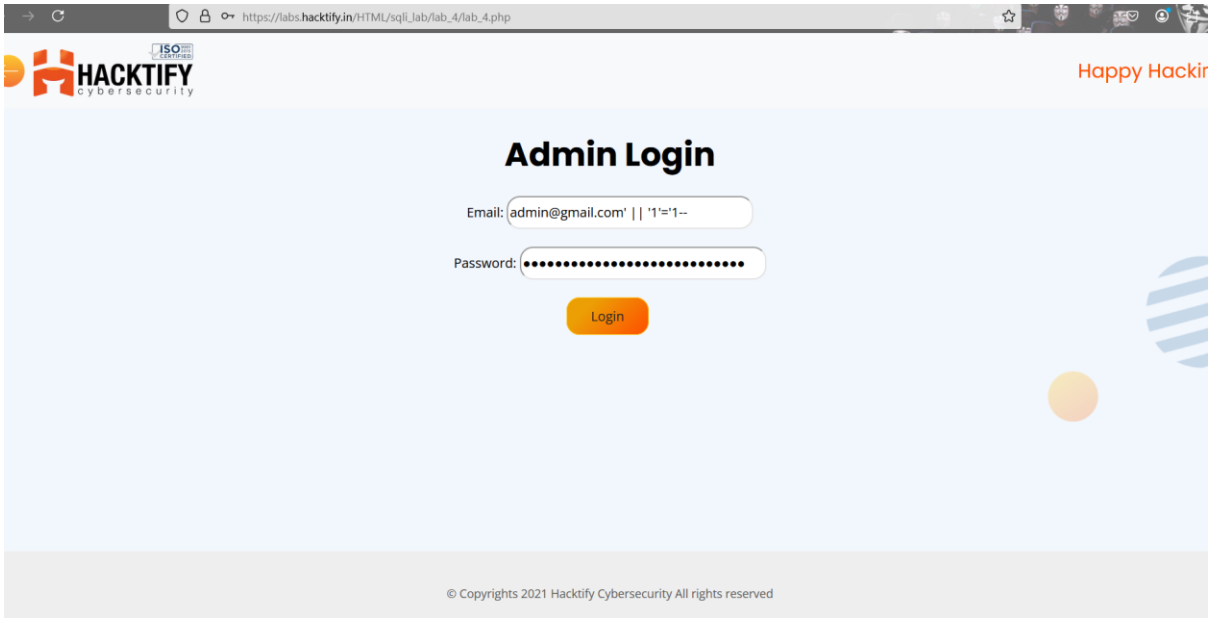


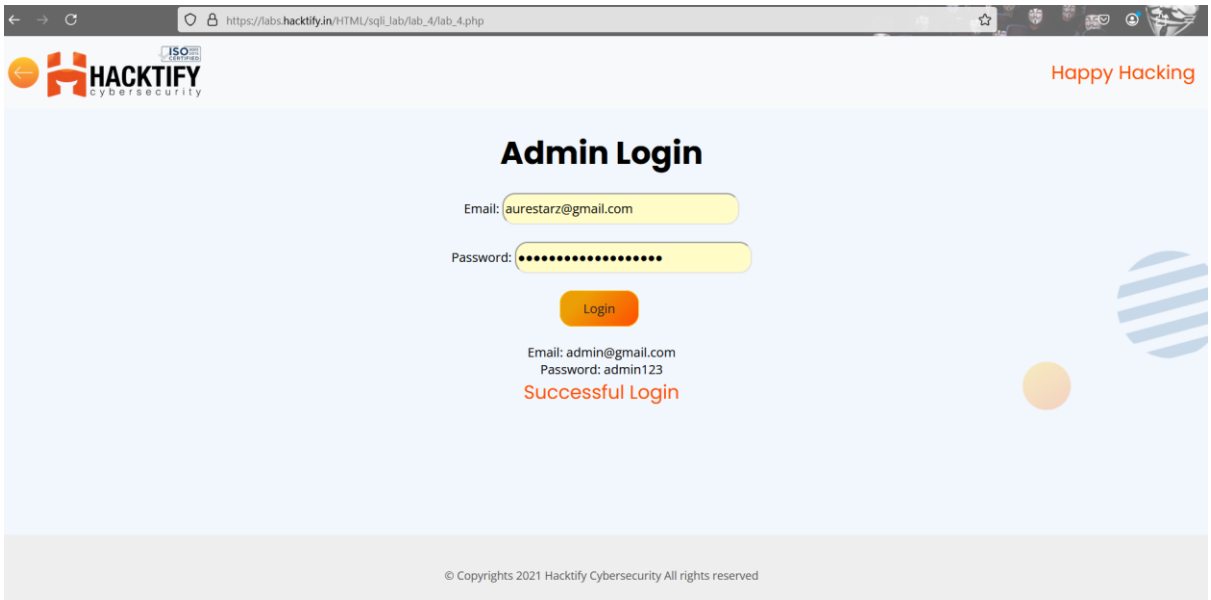
2.4. Trick 'em!

Reference	Risk Rating
Trick 'em!	**Medium**
Tools Used	
SQL payload	
Vulnerability Description	
It is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
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 retrieve sensitive data such as usernames, passwords, credit card numbers, and personal information.	
Suggested Countermeasures	
Validate and sanitize all user inputs to ensure they conform to expected formats.	
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



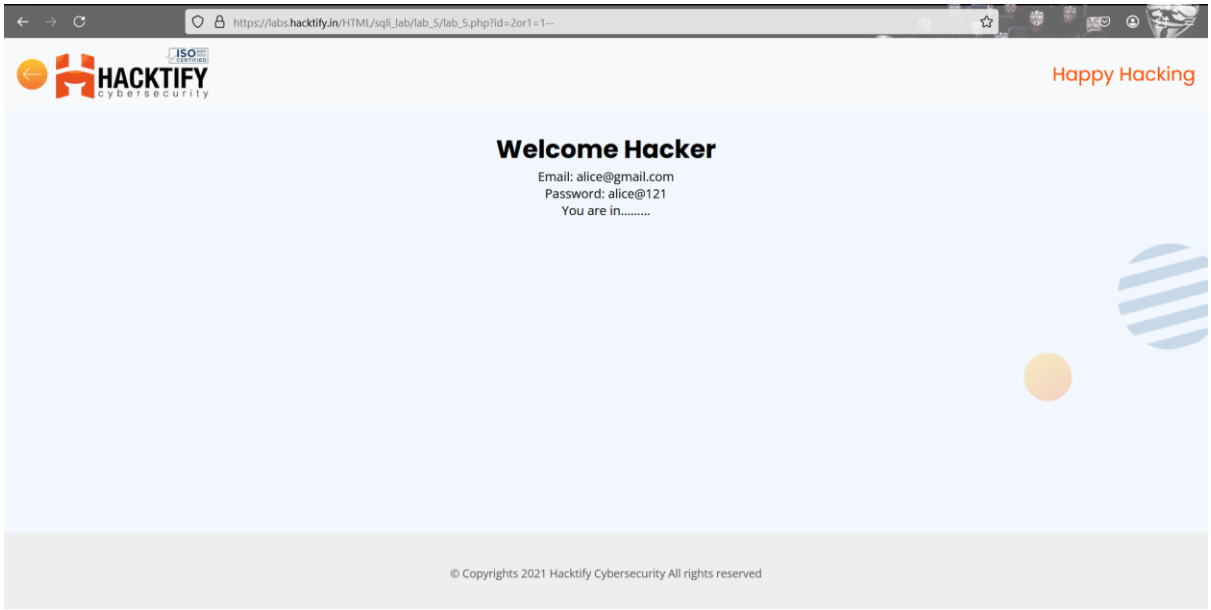


2.5. Booleans and Blind!

Reference	Risk Rating
Booleans and Blind!	**Medium**
Tools Used	
SQL payloads	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php?id=2or1=1--	
Consequences of not Fixing the Issue	
Attackers can bypass login mechanisms and gain unauthorized access to the application.	
Suggested Countermeasures	
Use stored procedures to encapsulate SQL logic and reduce the risk of injection.	
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

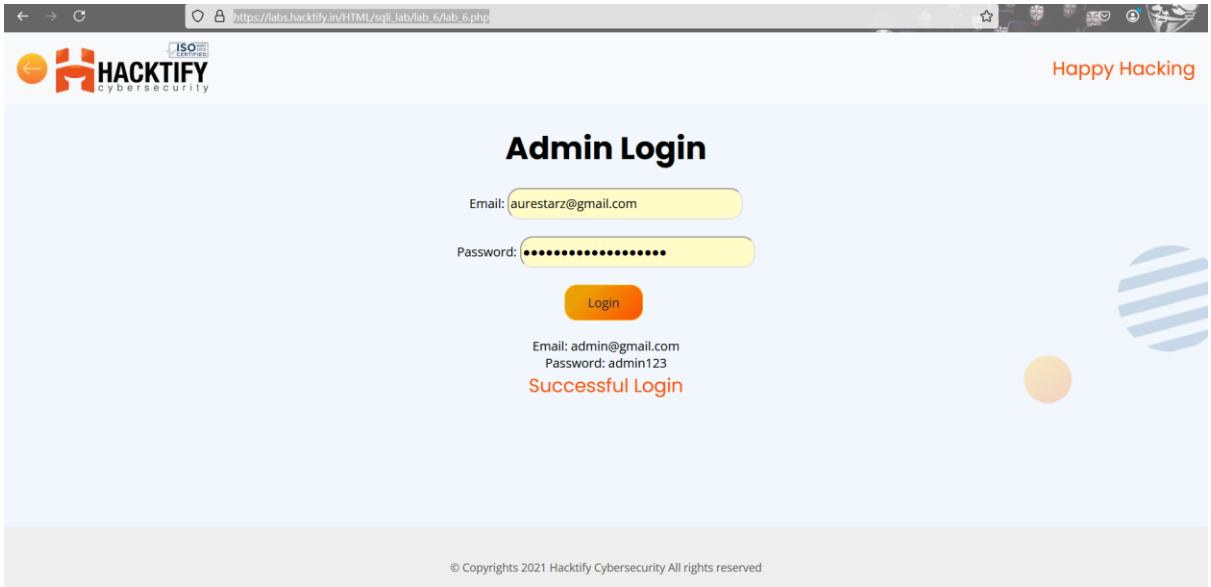


2.6. Error based: Tricked!

Reference	Risk Rating
Error based: Tricked!	**Medium**
Tools Used	
SQL payloads >> ') or ('a' = 'a and h")or("a" = "a into password section	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
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 bypass login mechanisms and gain unauthorized access to the application.	
Suggested Countermeasures	
Validate and sanitize all user inputs to ensure they conform to expected formats.	
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

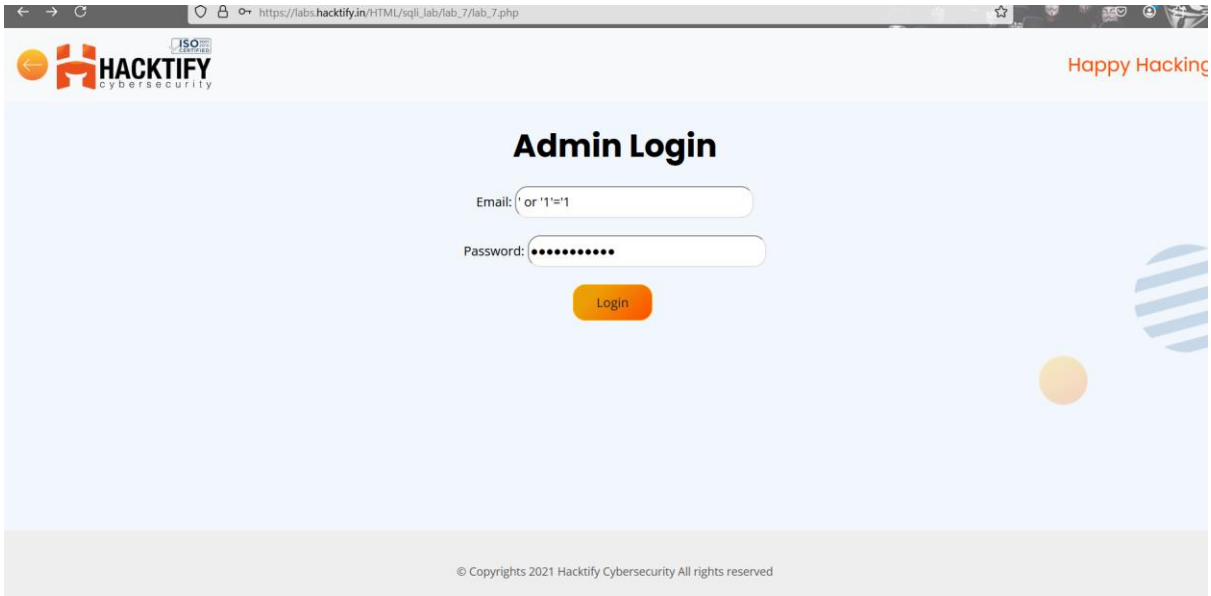


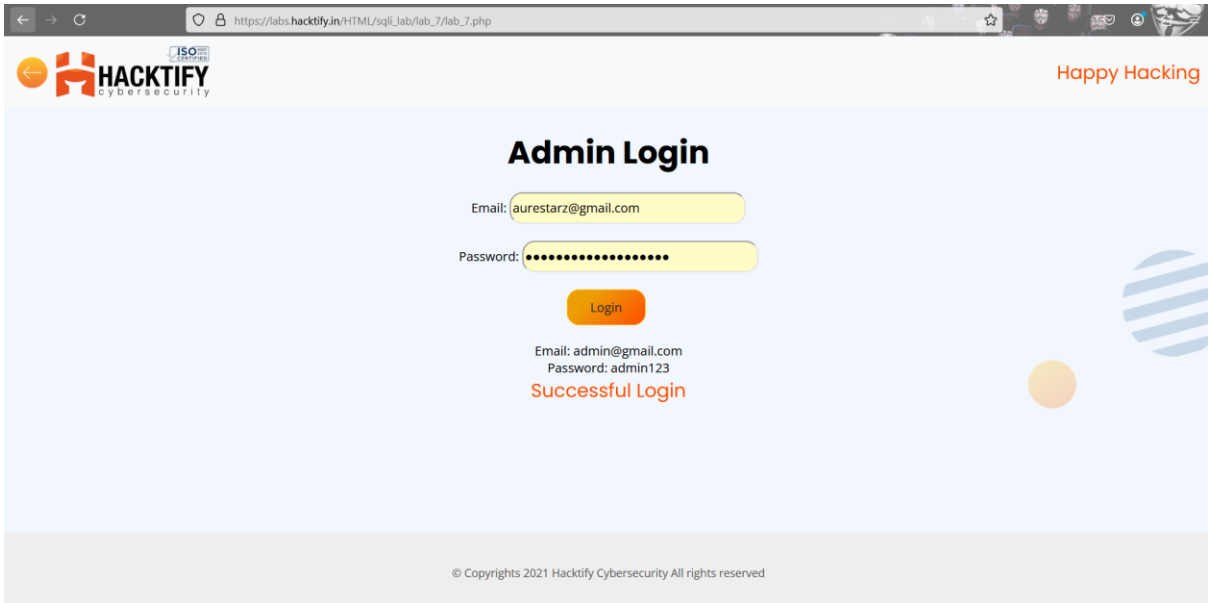
2.7. Errors and post!

Reference	Risk Rating
Errors and post!	Medium
Tools Used	
SQL payloads	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
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 inject malicious scripts to redirect users to phishing sites or download malware onto their devices.	
Suggested Countermeasures	
Deploy a Web Application Firewall to detect and block SQL injection payloads in real-time.	
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



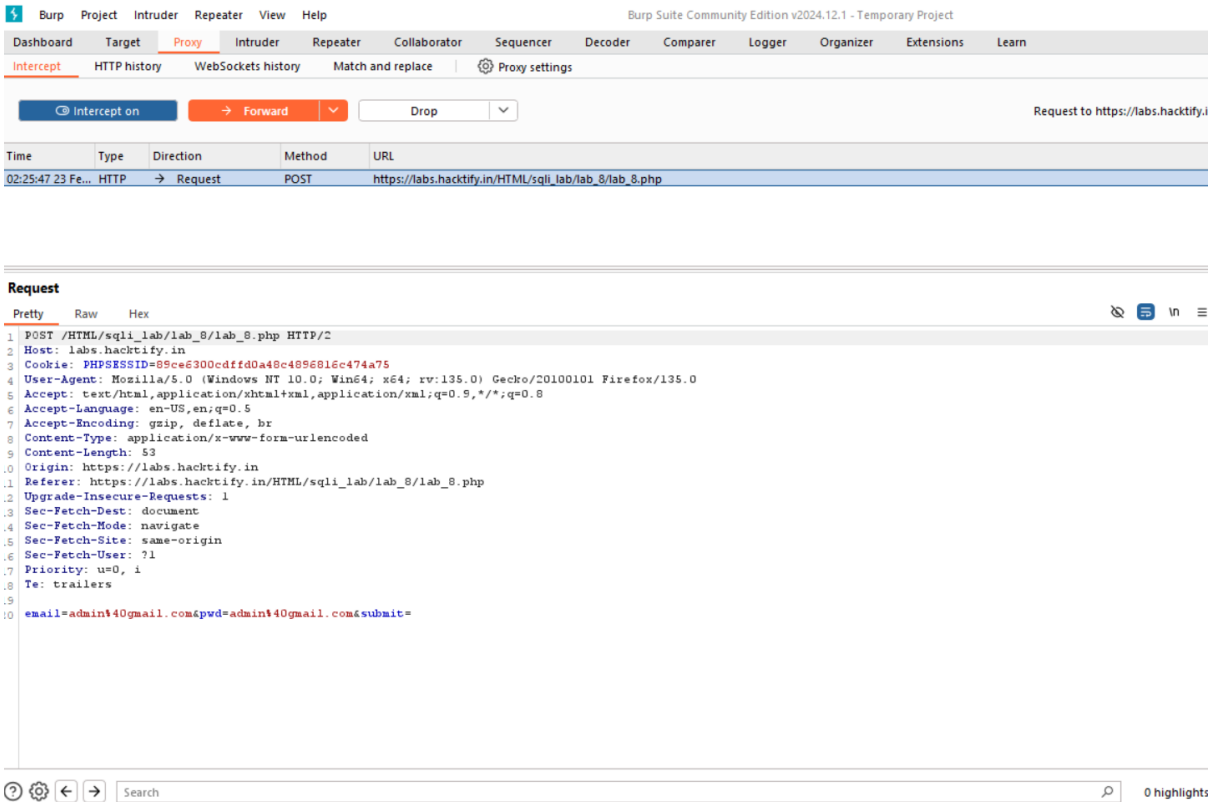


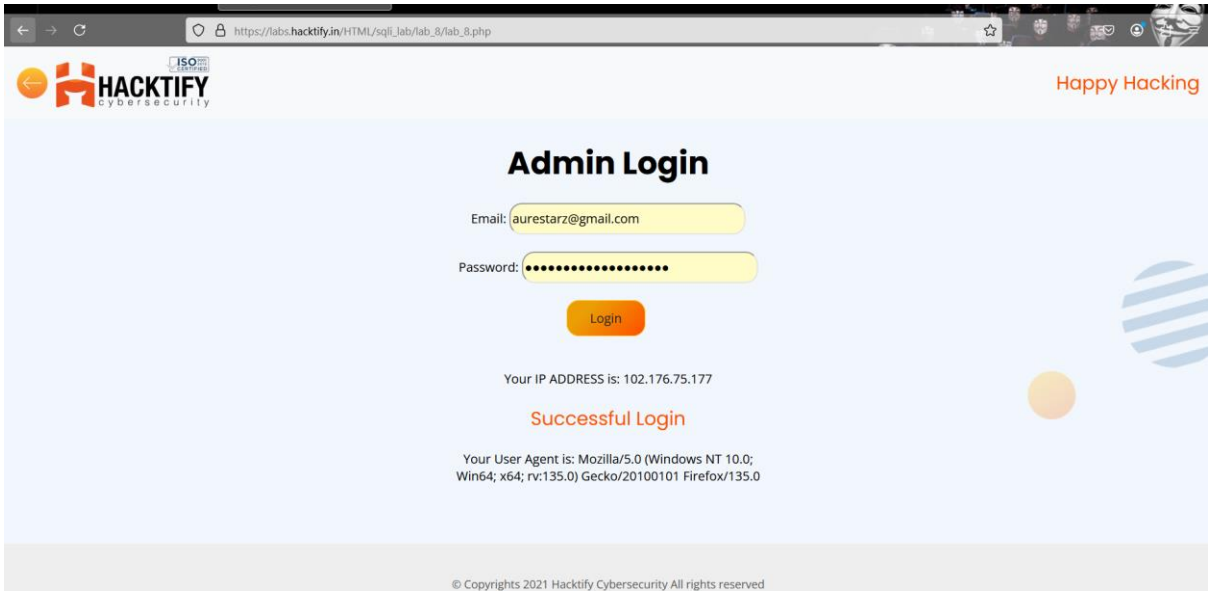
2.8. User Agents save us!

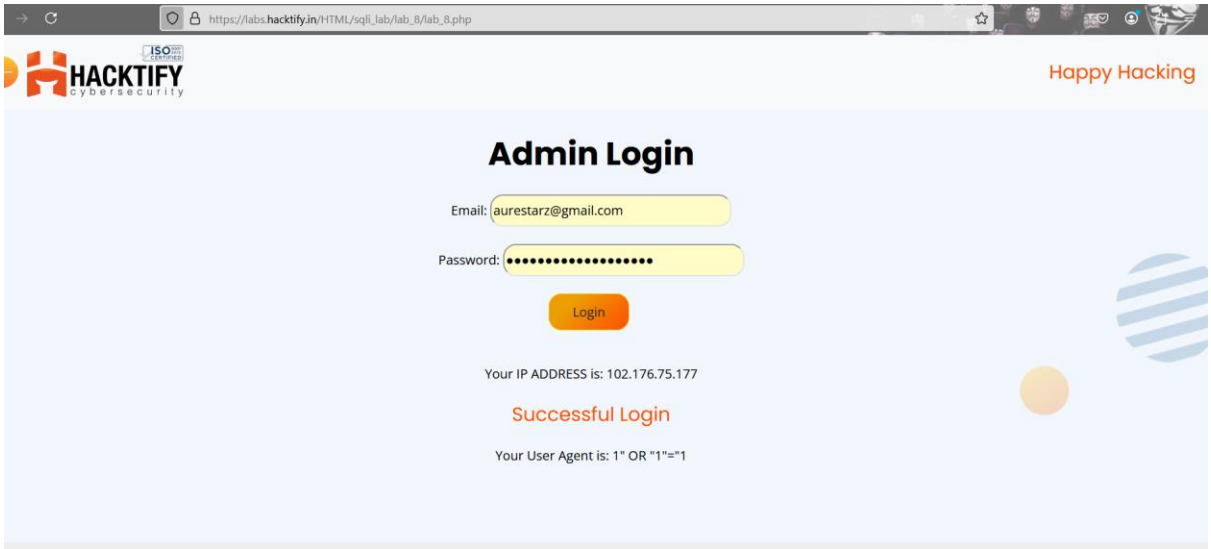
Reference	Risk Rating
User Agents save us!	**Medium**
Tools Used	
SQL payloads and burp suite	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis & Automatic Tool	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php	
Consequences of not Fixing the Issue	
Attackers can modify, delete, or insert data into the database, leading to data corruption or loss.	
Suggested Countermeasures	
Conduct regular vulnerability assessments and penetration testing to identify and fix SQL injection vulnerabilities.	
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





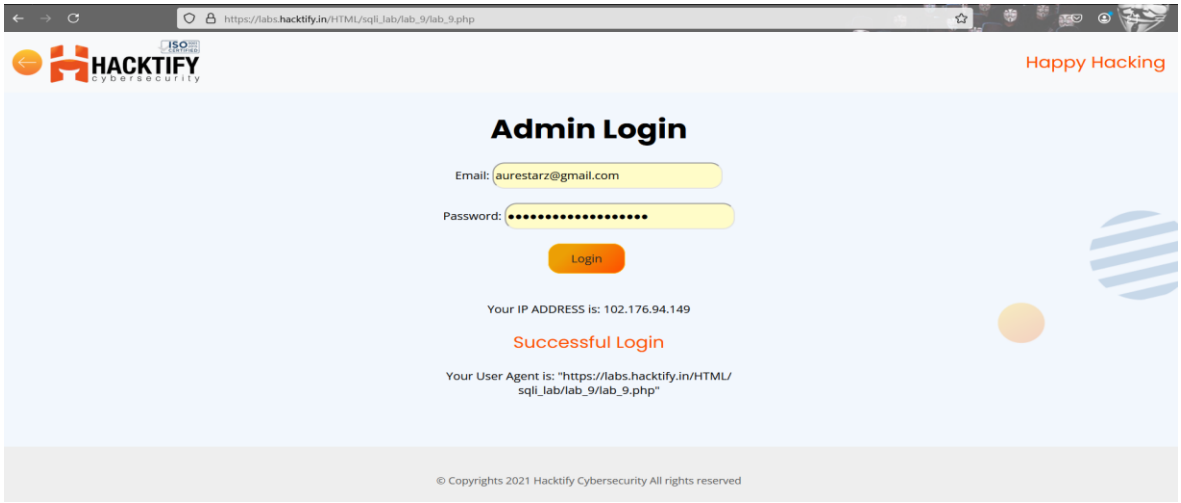


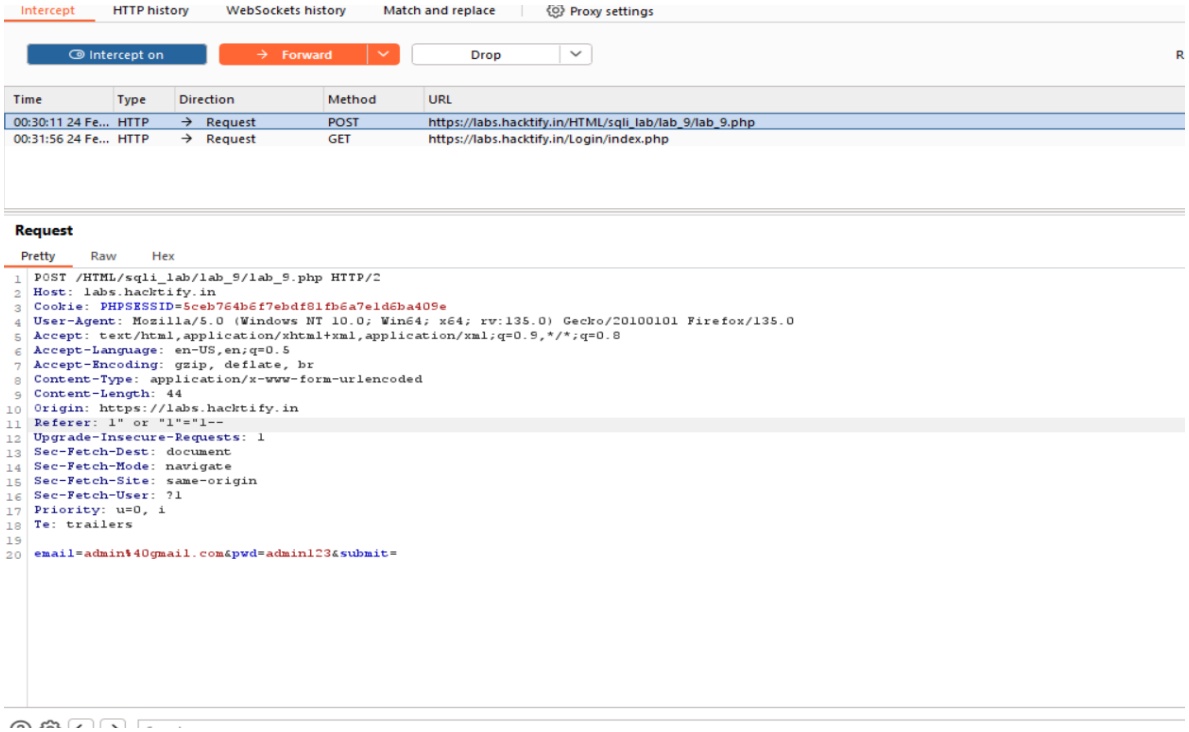
2.9. Referrer lead us!

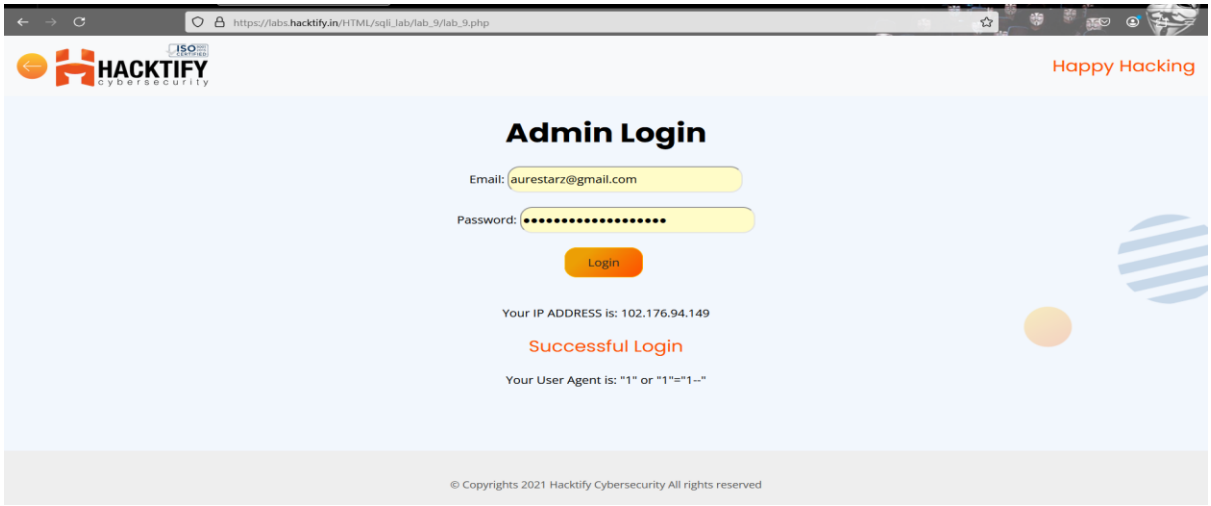
Reference	Risk Rating
Referrer lead us!	High
Tools Used	
SQL payloads and burp suite	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis and Automatic Tool	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php	
Consequences of not Fixing the Issue	
Attackers can bypass login mechanisms and gain unauthorized access to the application.	
Suggested Countermeasures	
Conduct regular vulnerability assessments and penetration testing to identify and fix SQL injection vulnerabilities.	
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





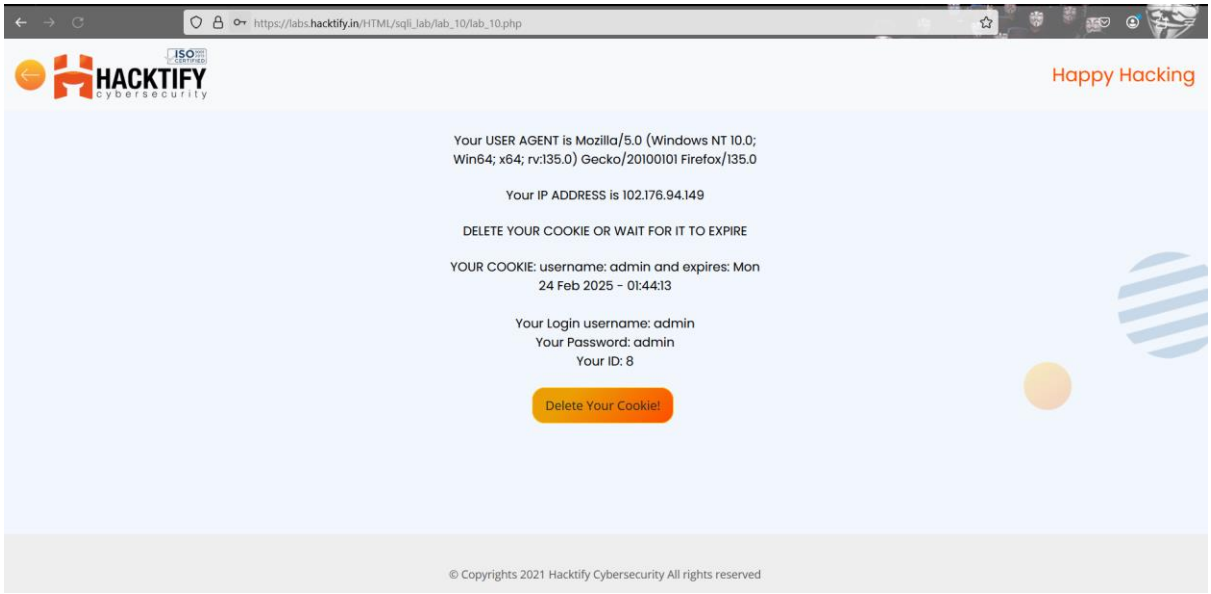


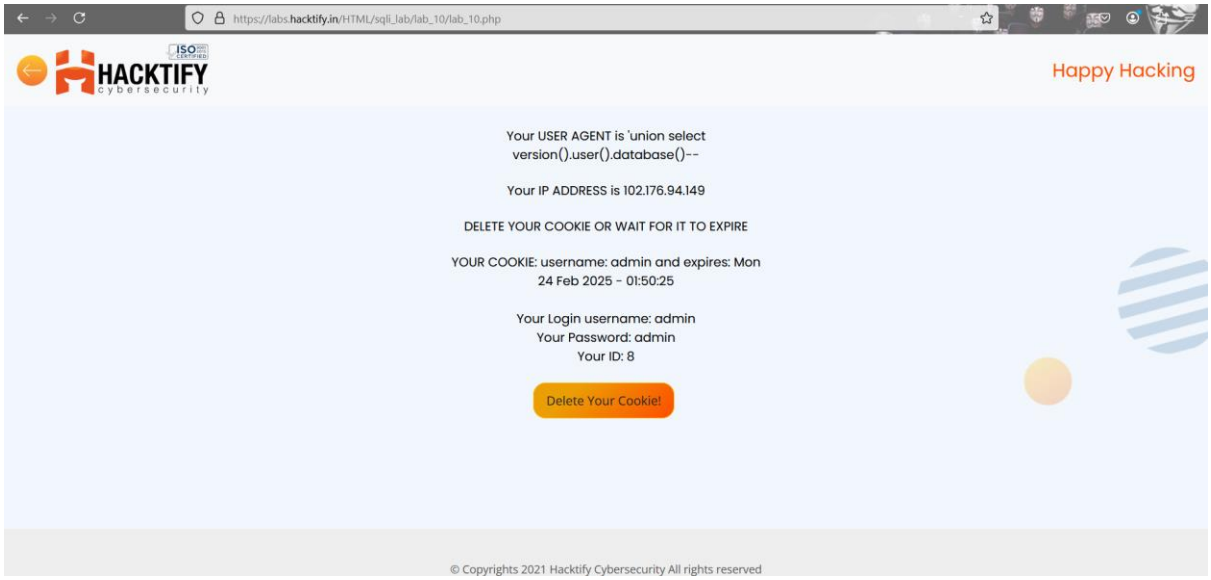
2.10. Oh Cookies!

Reference	Risk Rating
Oh Cookies!	High
Tools Used	
SQL payloads and burp suite	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis and Automatic Tool	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php	
Consequences of not Fixing the Issue	
Sensitive data can be exposed, leading to privacy violations and reputational damage.	
Suggested Countermeasures	
Conduct regular vulnerability assessments and penetration testing to identify and fix SQL injection vulnerabilities.	
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


https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php
HACKTIFY
cybersecurity
Happy Hacking
Your USER AGENT is Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101 Firefox/135.0
Your IP ADDRESS is 102.176.94.149
DELETE YOUR COOKIE OR WAIT FOR IT TO EXPIRE
YOUR COOKIE: username: admin and expires: Mon 24 Feb 2025 - 01:44:13
Your Login username: admin
Your Password: admin
Your ID: 8
Delete Your Cookie!
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

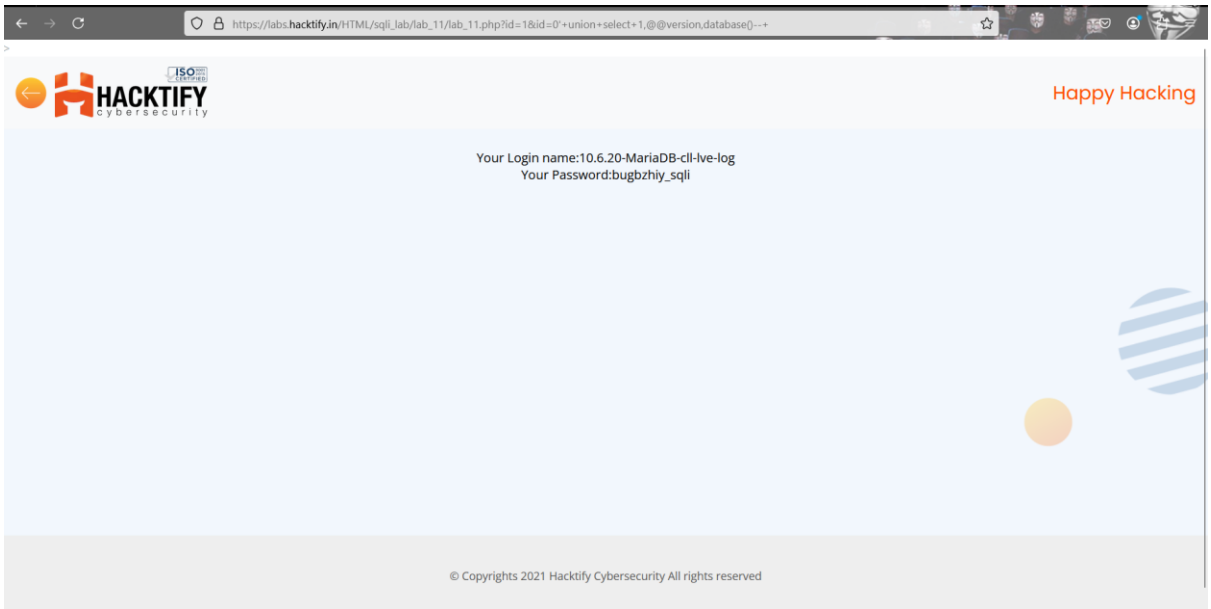

https://labs.hacktify.in/HTML/sqli_lab/lab_10/lab_10.php
HACKTIFY
cybersecurity
Happy Hacking
Your USER AGENT is 'union select version().user().database()--
Your IP ADDRESS is 102.176.94.149
DELETE YOUR COOKIE OR WAIT FOR IT TO EXPIRE
YOUR COOKIE: username: admin and expires: Mon 24 Feb 2025 - 01:50:25
Your Login username: admin
Your Password: admin
Your ID: 8
Delete Your Cookie!
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.11. WAF's are injected!

Reference	Risk Rating
WAF's are injected!	**High**
Tools Used	
SQL payloads and burp suite	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis and Automatic Tool	
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	
Sensitive data can be exposed, leading to privacy violations and reputational damage.	
Suggested Countermeasures	
Use allowlists (not blocklists) to accept only safe and expected input patterns.	
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!	**High**
Tools Used	
SQL payloads and burp suite	
Vulnerability Description	
SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.	
How It Was Discovered	
Manual Analysis and Automatic Tool	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_12/lab_12.php?id=1&id=1	
Consequences of not Fixing the Issue	
Sensitive data can be exposed, leading to privacy violations and reputational damage.	
Suggested Countermeasures	
Conduct regular vulnerability assessments and penetration testing to identify and fix SQL injection vulnerabilities.	
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