Penetration Testing Report

Full Name: Anumandla Dhanush
Program: HCS - Penetration Testing Internship Week-2
Date:

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	**Labs.hacktify.in - Cross Site Scripting** **Labs.hacktify.in - Insecure Direct Object References**

3. Summary

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

Total number of Sub-labs: 15 Sub-labs

High	Medium	Low
7	6	2

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

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

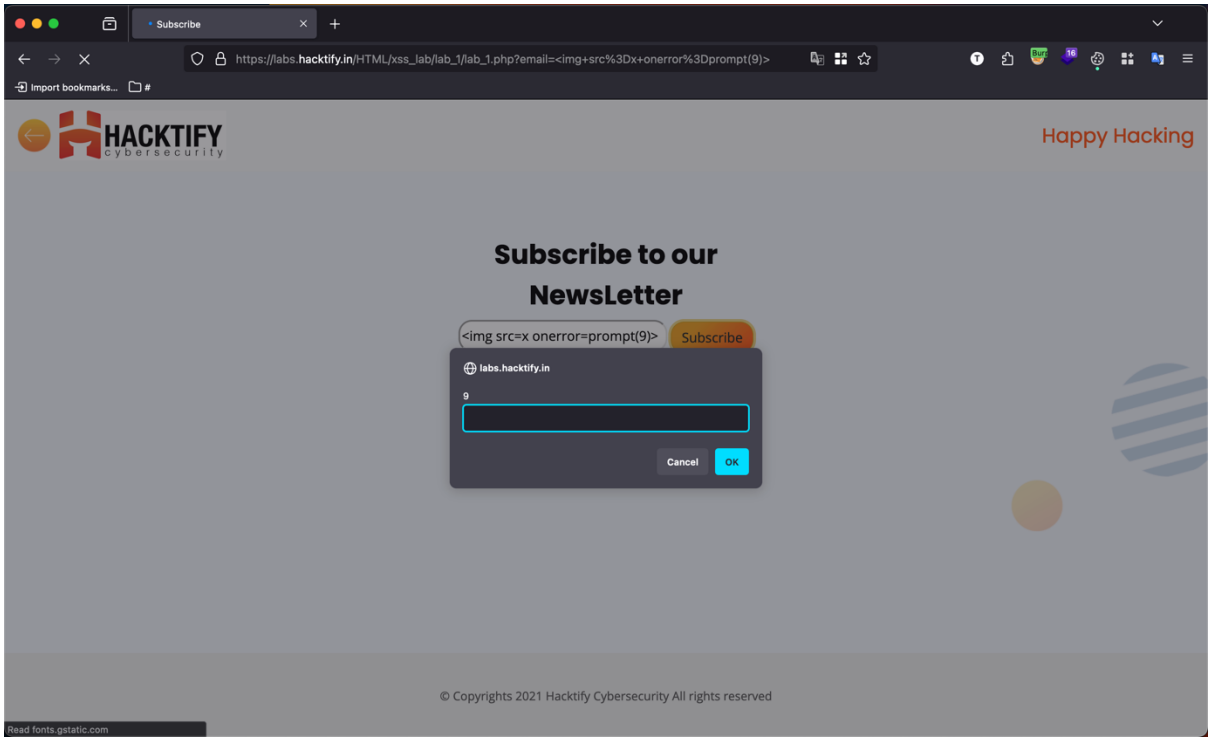
Low - **Number of Sub-labs with Easy difficulty level**

1. Cross Site Scripting

1.1. Let's Do IT!

Reference	Risk Rating
Let's Do IT!	**Low**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php 2. There you will see Newsletter subscription (input text field). 3. Enter the payload: <img src=x onerror=prompt(9)> in the input field and click on “Subscribe”. 4. The payload gets executed and displayed on the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.	
References	
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting	

Proof of Concept

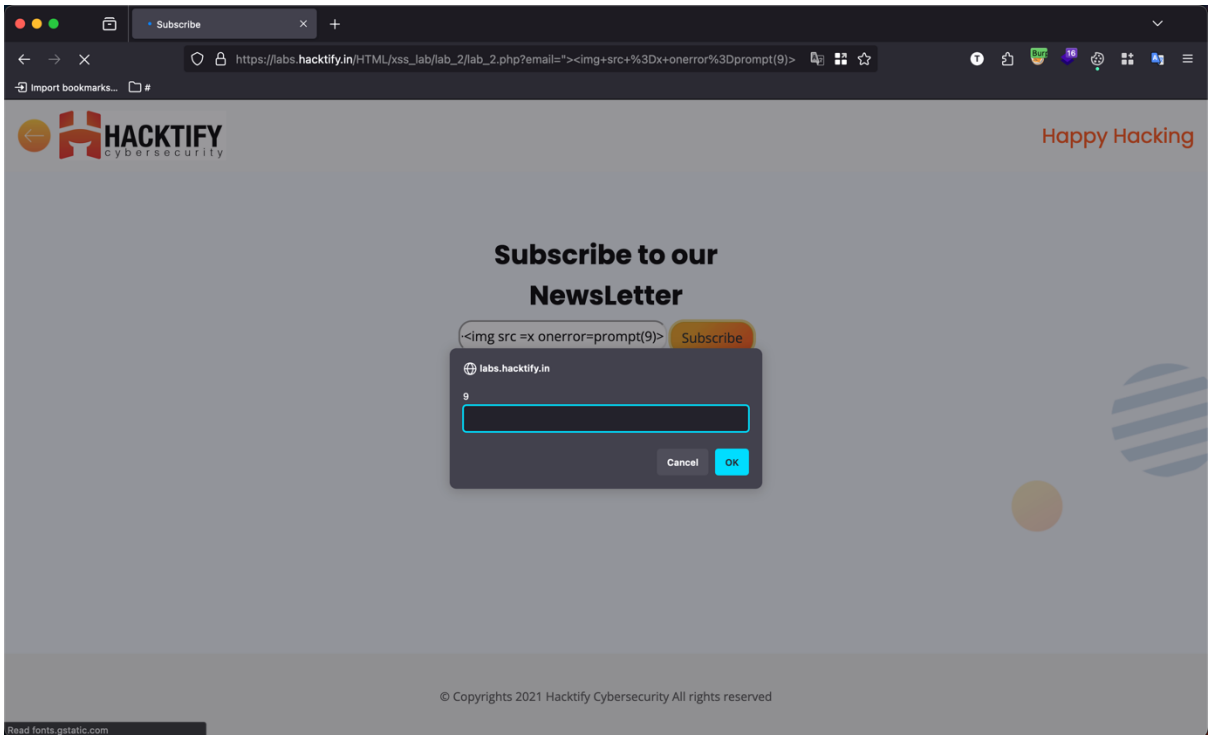


1.2. Balancing is Important in Life!

Reference	Risk Rating
Let's Do IT!	**Low**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php 2. There you will see Newsletter subscription (input text field). 3. Enter the payload: "><img src=x onerror=prompt(9)> in the input field and click on “Subscribe”. 4. The payload gets executed and displayed on the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application.	

2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

Proof of Concept

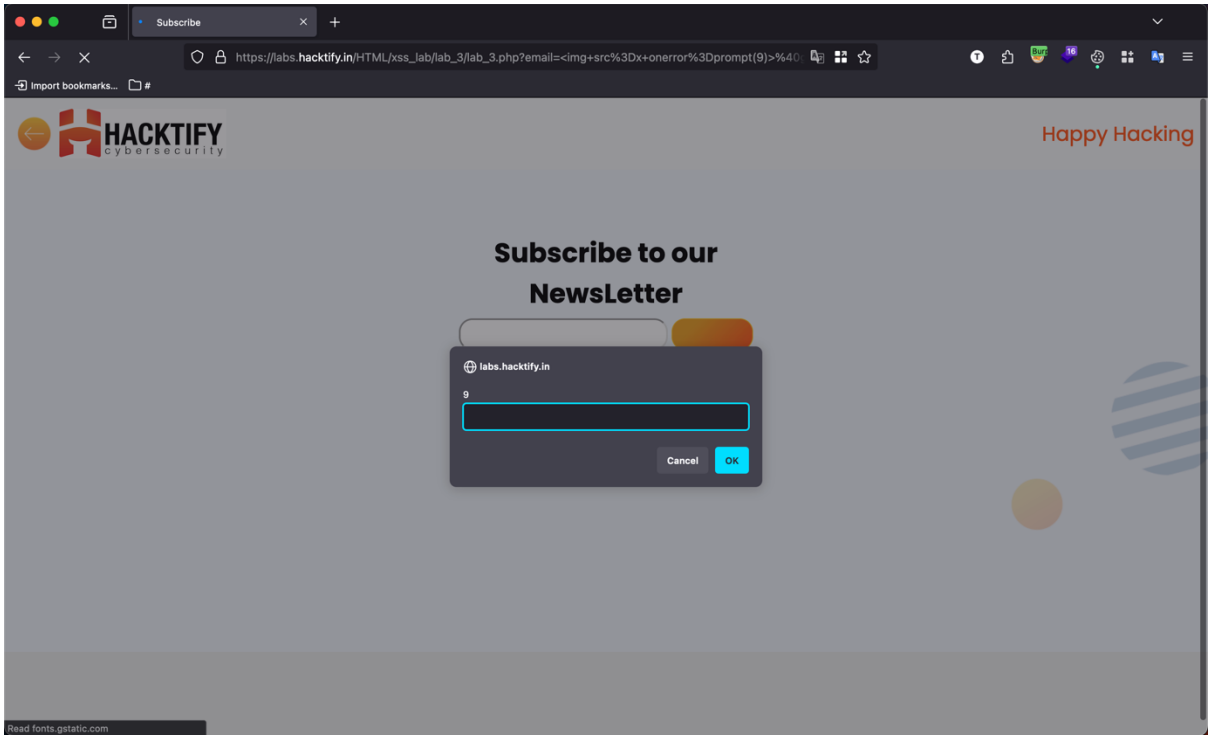


1.3. XSS is everywhere!

Reference	Risk Rating
XSS is everywhere!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php 2. There you will see Newsletter subscription (input text field). 3. Enter the payload: <img src=x onerror=prompt(9)>@gmail.com in the input field. 4. Then click on “Subscribe”. 5. The payload gets executed and displayed on the web page.	

Vulnerable URLs
https://labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php
Consequences of not Fixing the Issue
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.
Suggested Countermeasures
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

Proof of Concept

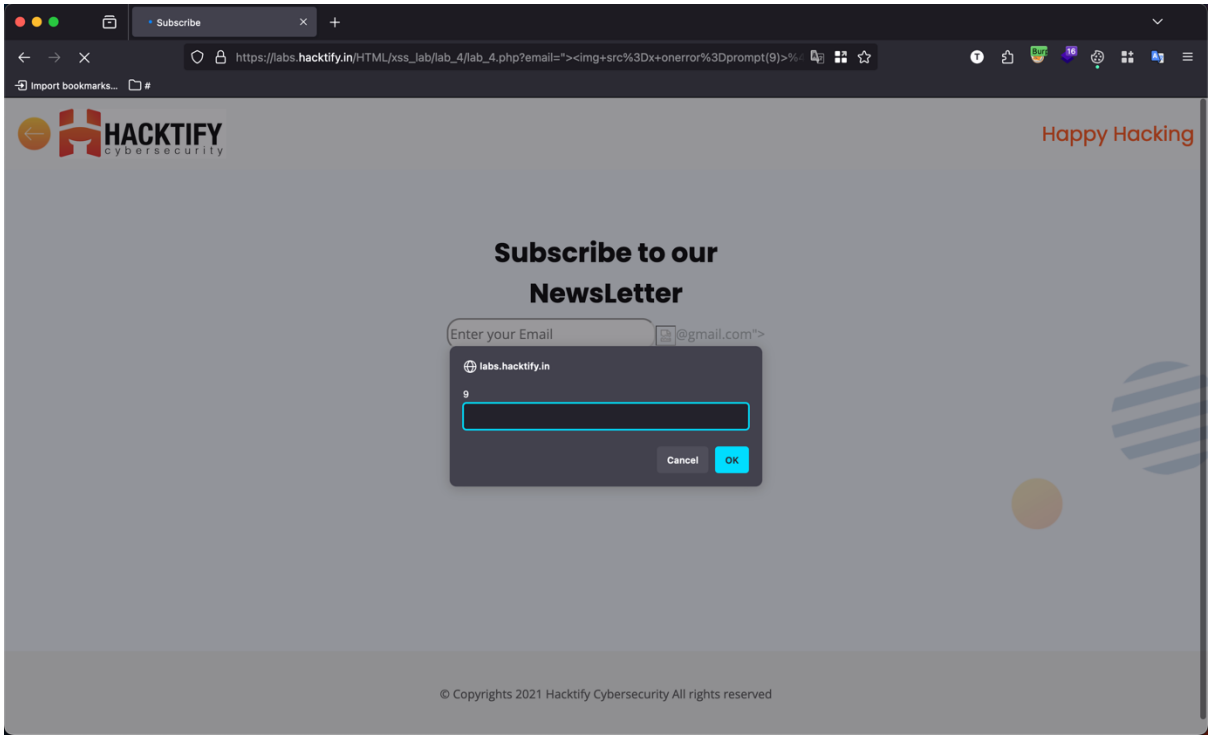


1.4. Alternatives are must!

Reference	Risk Rating
Alternatives are must!	**Medium**
Tools Used	
Web Browser - Firefox	

Vulnerability Description
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.
How It Was Discovered
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php 2. There you will see Newsletter subscription (input text field). 3. Enter the payload: "><img src=x onerror=prompt(9)>@gmail.com in the input field. 4. Then click on “Subscribe”. 5. The payload gets executed and displayed on the web page.
Vulnerable URLs
https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php
Consequences of not Fixing the Issue
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.
Suggested Countermeasures
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

Proof of Concept

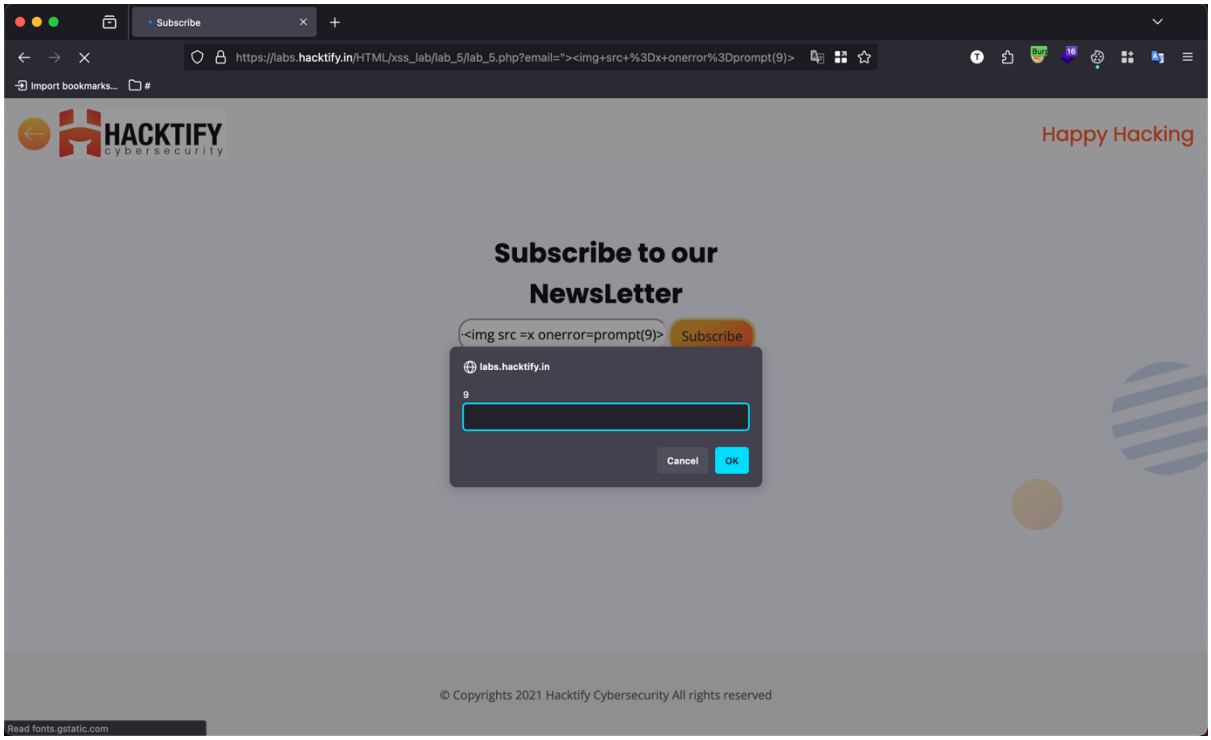


1.5. Developer hates scripts!

Reference	Risk Rating
Developer hates scripts!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_5/lab_5.php 2. There you will see Newsletter subscription (input text field). 3. Enter the payload: "><img src=x onerror=prompt(9)> in the input field and click on “Subscribe”. 4. The payload gets executed and displayed on the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_5/lab_5.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application.	

2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

Proof of Concept

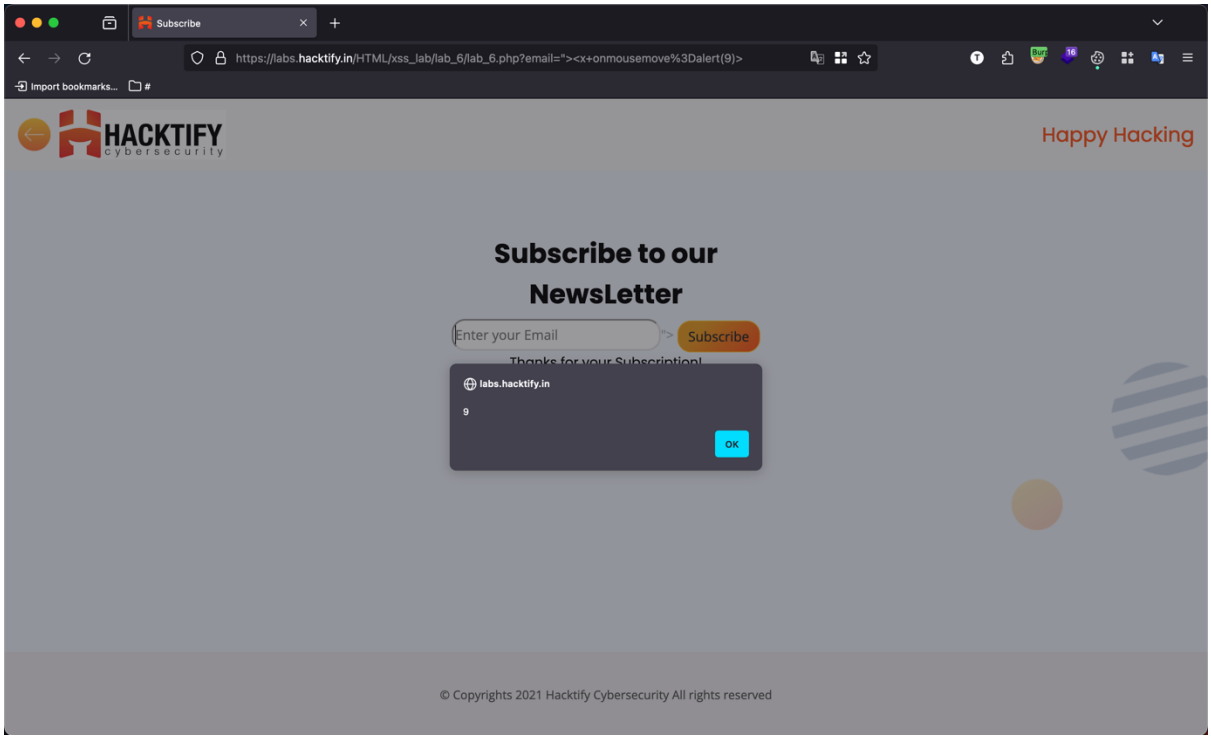


1.6. Change the Variation!

Reference	Risk Rating
Change the Variation!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_6/lab_6.php 2. There you will see Newsletter subscription (input text field). 3. Enter the payload: "><x onmousemove=alert(9)> in the input field and click on “Subscribe”.	

4. The payload gets executed whenever mousepointer move towards the “Subscribe” button and displayed on the web page.
Vulnerable URLs
https://labs.hacktify.in/HTML/xss_lab/lab_6/lab_6.php
Consequences of not Fixing the Issue
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.
Suggested Countermeasures
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

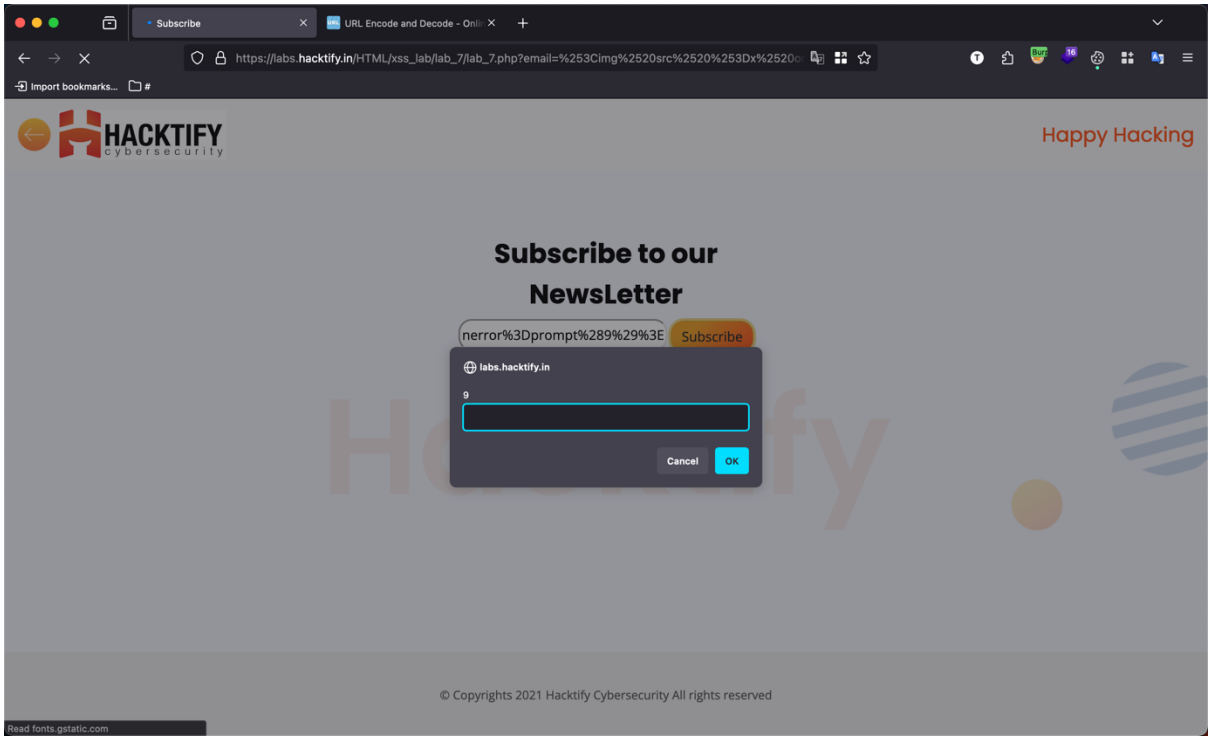
Proof of Concept



1.7. Encoding is the key?

Reference	Risk Rating
Encoding is the key?	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php 2. There you will see Newsletter subscription (input text field). 3. Webpage is filtering symbols and special characters when payloads are feed to application. 4. Go to https://www.urlencoder.org/ 5. paste the normal payload: <img src =x onerror=prompt(9)> and click on “Encode”. 6. Copy the encoded payload: %3Cimg%20src%20%3Dx%20onerror%3Dprompt%289%29%3E 7. Paste it in the input field and click on “Subscribe”. 8. The payload gets executed and displayed on the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.	
References	
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting	

Proof of Concept

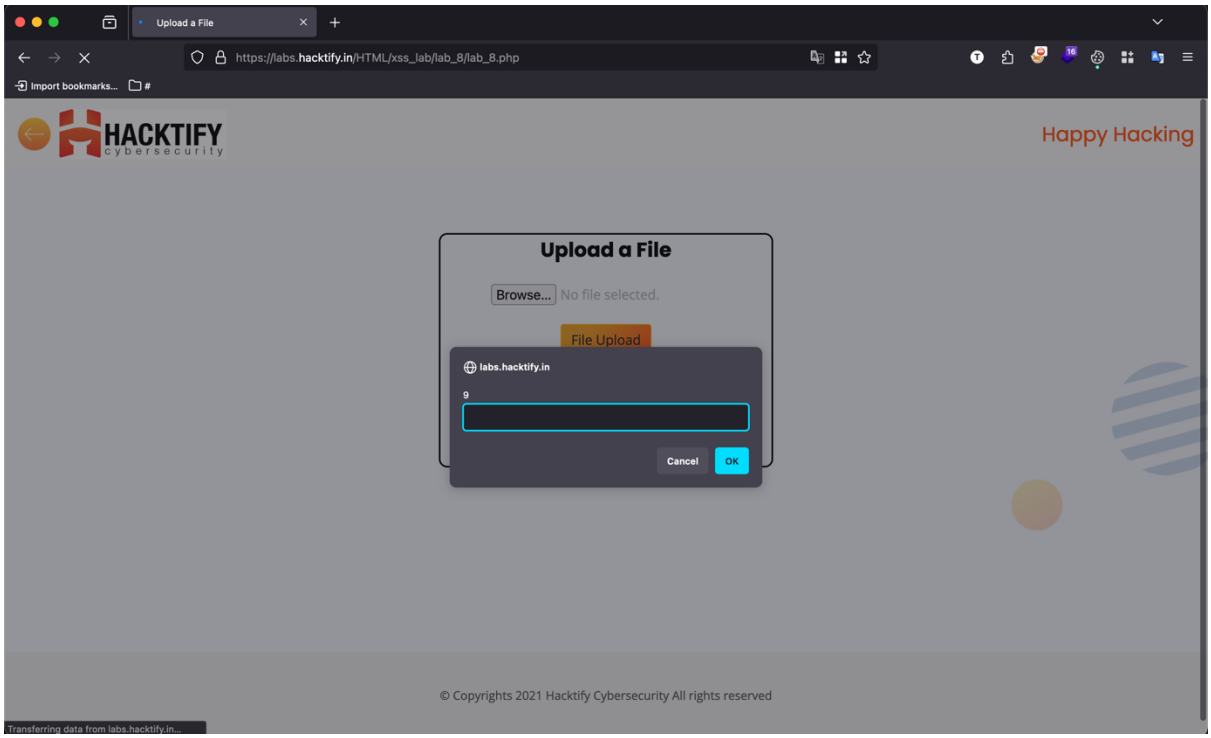


1.8. XSS with File Upload (file name)

Reference	Risk Rating
XSS with File Upload (file name)	**High**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php 2. There is a file upload option. 3. Create a file and rename it as <img src =x onerror=prompt(9)>.txt ({payload}.txt) 4. Click on “Browse” button and upload the file. 5. Click on “File Upload”. 6. The payload gets executed and displayed on the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application.	

2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

Proof of Concept

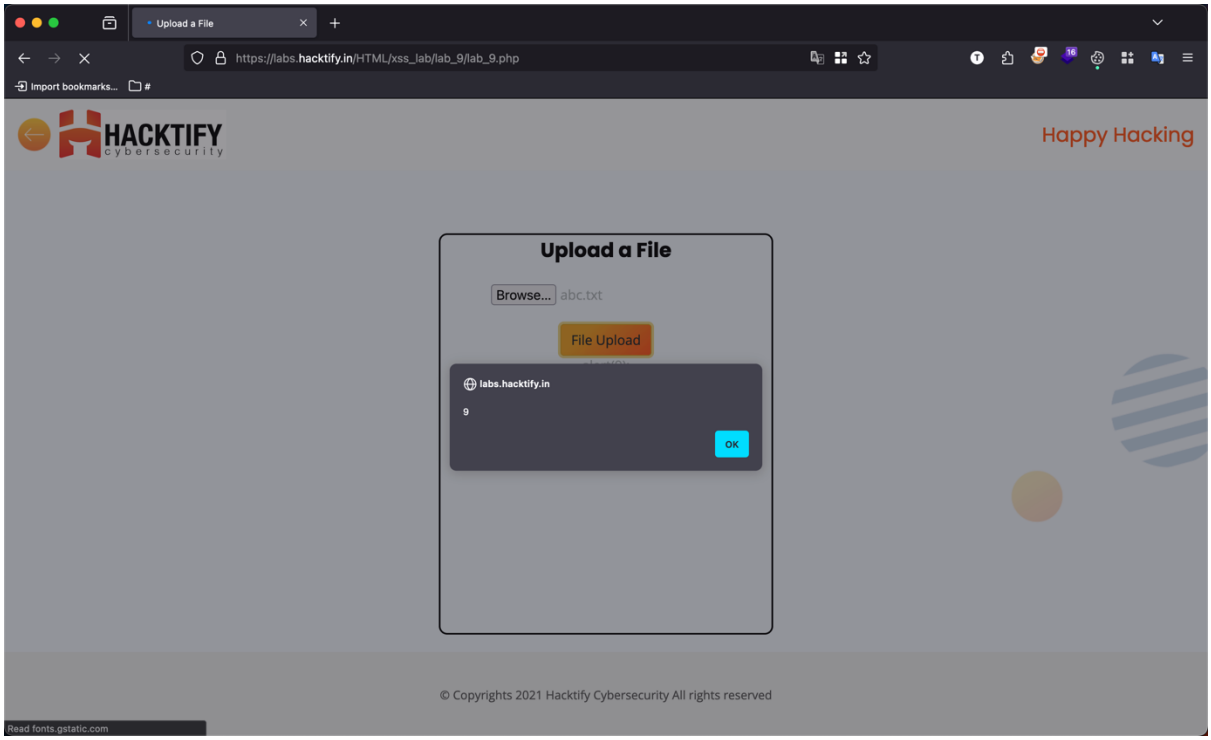


1.9. XSS with File Upload (File Content)

Reference	Risk Rating
XSS with File Upload (File Content)	**High**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_9/lab_9.php 2. There is a file upload option. 3. Create a file(abc.txt) and paste the payload: <script>alert(9)</script> 4. Save the file.	

5. Click on “Browse” button and upload the file. 6. Click on “File Upload”. 7. The payload gets executed and displayed on the web page.
Vulnerable URLs
https://labs.hacktify.in/HTML/xss_lab/lab_9/lab_9.php
Consequences of not Fixing the Issue
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.
Suggested Countermeasures
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

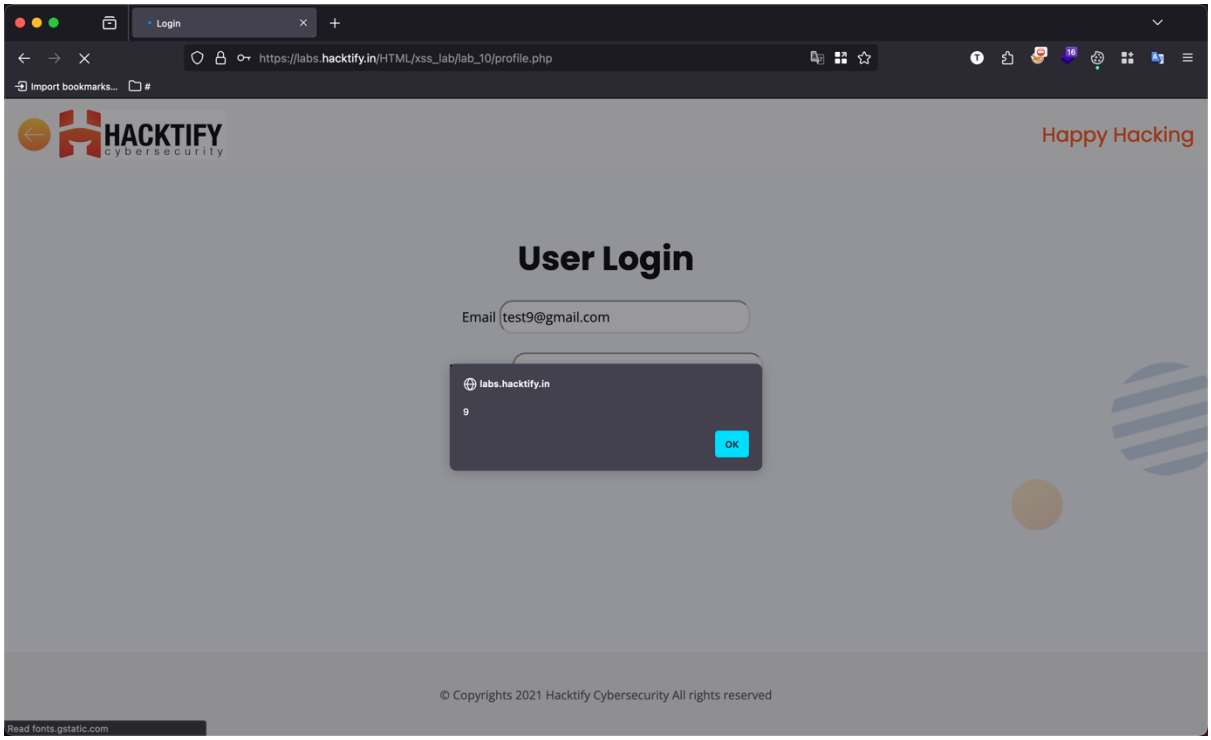
Proof of Concept



1.10. Stored Everywhere!

Reference	Risk Rating
Stored Everywhere!	**High**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_10/lab_10.php 2. There you will see login and register functionality. 3. Click on register; Paste the payload: "><script>alert(9)</script> in the input fields 'First Name' and 'Last Name' and register an account. 4. Now Click on login; Enter the credentials and click "login". 5. The payload gets executed in https://labs.hacktify.in/HTML/xss_lab/lab_10/profile.php	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_10/profile.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application. 2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.	
References	
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting	

Proof of Concept

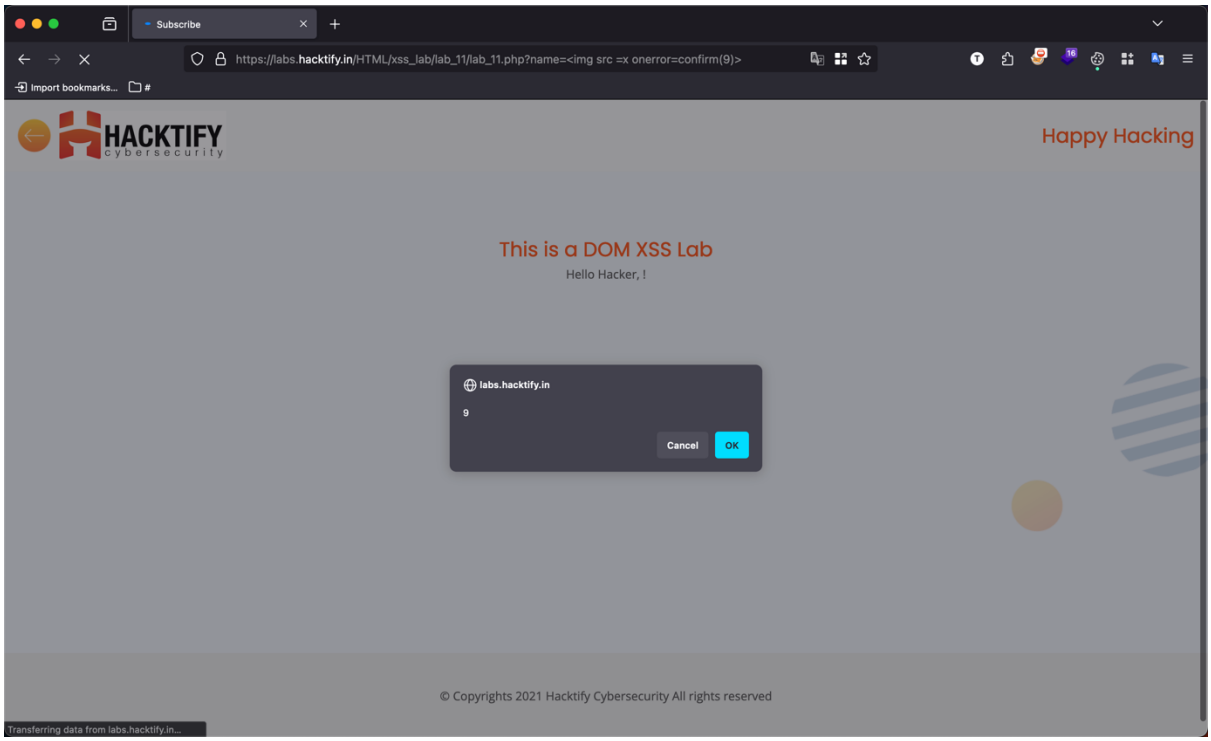


1.11. DOM's are love!

Reference	Risk Rating
DOM's are love!	**High**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web pages viewed by other users, potentially leading to the unauthorized theft of sensitive data, session hijacking, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php 2. Add the payload: <img src =x onerror=confirm(9)> with parameter 'name' at the end of the URL. 3. The final URL should be https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php?name=<img src =x onerror=confirm(9)> 4. Enter the crafted URL in the browser's search box and hit Enter. 5. The Payload gets executed and displayed on the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php	
Consequences of not Fixing the Issue	
Failure to address Cross-Site Scripting (XSS) leaves the web application vulnerable to the theft of sensitive user information, such as login credentials and personal data. Additionally, it enables attackers to execute malicious scripts in the context of other users, leading to potential account compromise and unauthorized actions within the application.	
Suggested Countermeasures	
1. Implement input validation by validating and sanitizing user inputs to prevent the injection of malicious scripts into the web application.	

2. Utilize output encoding techniques, such as HTML entity encoding, to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of Cross-Site Scripting vulnerabilities.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html 2. https://portswigger.net/web-security/cross-site-scripting

Proof of Concept



2. Insecure Direct Object References

2.1. Give me my amount!!

Reference	Risk Rating
Give me my amount!!	**High**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a web security vulnerability where an attacker exploits inadequate access controls, accessing unauthorized resources or manipulating object references directly, potentially leading to unauthorized data exposure, modification, or deletion within the web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/idor_lab/lab_1/lab_1.php	

2. There you will see Bank User Login.
3. Click on “Register” and create an account.
4. Later, Log into the account.
5. You will be directed to https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=136
6. As we can see there is an “id” generated for the account.
7. Change the value of “id” parameter in the URL to random number and hit Enter.
8. You get accessed to another account profile(victim) disclosing sensitive user information.

Vulnerable URLs

https://labs.hacktify.in/HTML/idor_lab/lab_1/lab_1.php

Consequences of not Fixing the Issue

Failure to address Insecure Direct Object References (IDOR) can result in unauthorized access to sensitive data, allowing attackers to view, modify, or delete confidential information. This may lead to data breaches, privacy violations, and compromise the overall security and integrity of the web application.

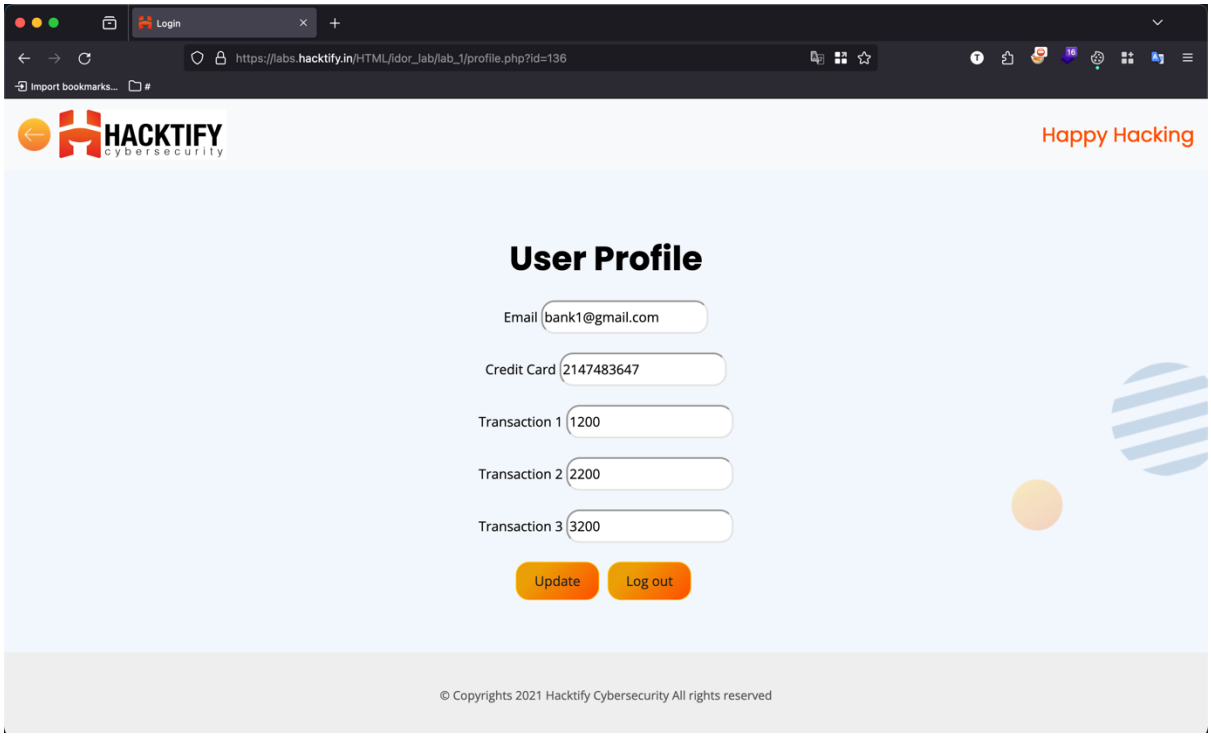
Suggested Countermeasures

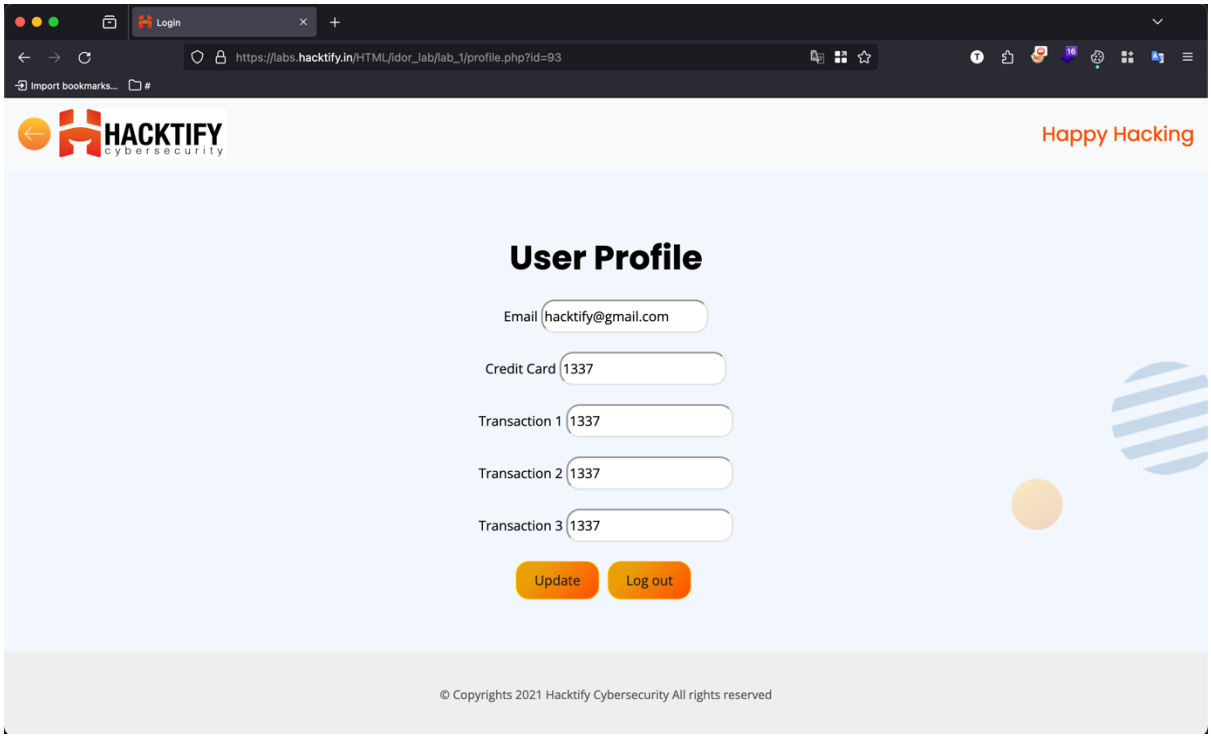
1. Implement proper access controls and authorization mechanisms to ensure that users can only access resources for which they have explicit permissions.
2. Use indirect references, such as unique identifiers or tokens, instead of relying on user-controlled input or sequential identifiers, to mitigate the risk of Insecure Direct Object References (IDOR) vulnerabilities.

References

1. https://owasp.org/www-community/attacks/Insecure_Direct_Object_References
2. https://portswigger.net/web-security/access-control/idor

Proof of Concept



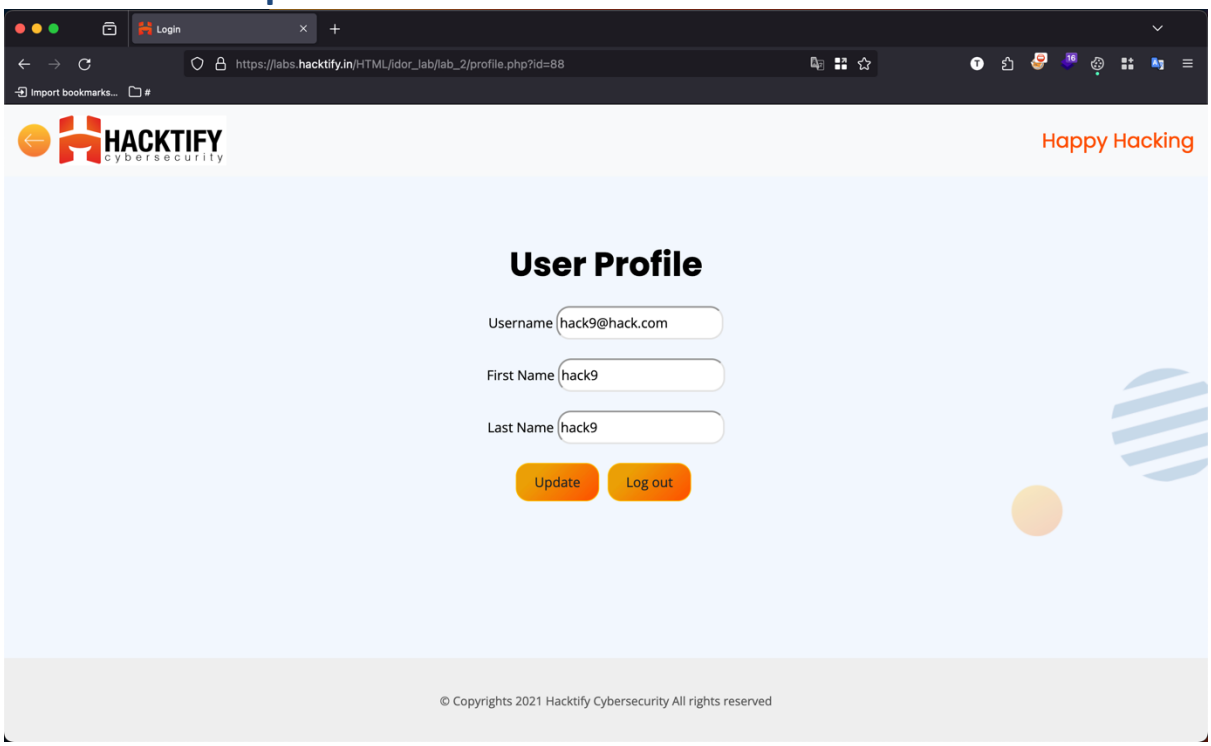


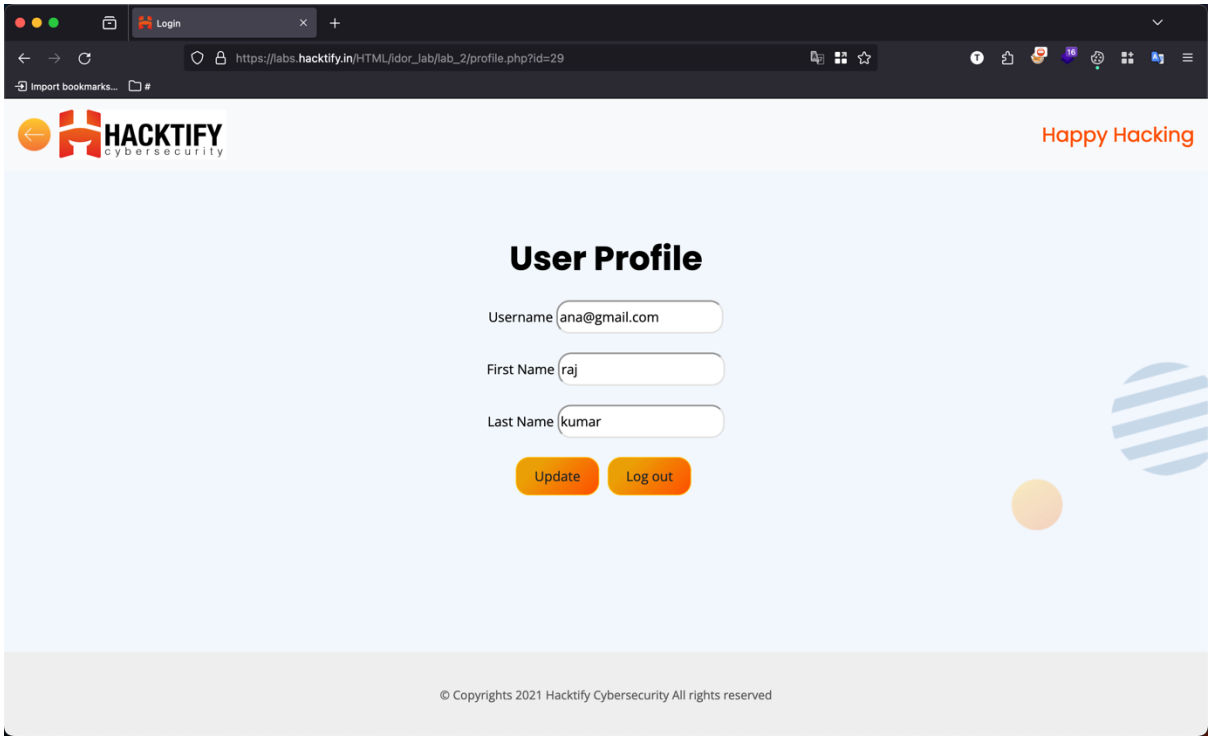
2.2. Stop polluting my params!

Reference	Risk Rating
Stop polluting my params!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a web security vulnerability where an attacker exploits inadequate access controls, accessing unauthorized resources or manipulating object references directly, potentially leading to unauthorized data exposure, modification, or deletion within the web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/idor_lab/lab_2/lab_2.php 2. There you will see User Login. 3. Click on “Register” and create an account. 4. Later, Log into the account. 5. You will be directed to https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=88 6. As we can see there is an “id” generated for the account. 7. Change the value of “id” parameter in the URL to random number and hit Enter. 8. You get accessed to another account profile(victim) disclosing sensitive user information.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_2/lab_2.php	
Consequences of not Fixing the Issue	
Failure to address Insecure Direct Object References (IDOR) can result in unauthorized access to sensitive data, allowing attackers to view, modify, or delete confidential information. This may lead to	

data breaches, privacy violations, and compromise the overall security and integrity of the web application.
Suggested Countermeasures
1. Implement proper access controls and authorization mechanisms to ensure that users can only access resources for which they have explicit permissions. 2. Use indirect references, such as unique identifiers or tokens, instead of relying on user-controlled input or sequential identifiers, to mitigate the risk of Insecure Direct Object References (IDOR) vulnerabilities.
References
1. https://owasp.org/www-community/attacks/Insecure_Direct_Object_References 2. https://portswigger.net/web-security/access-control/idor

Proof of Concept



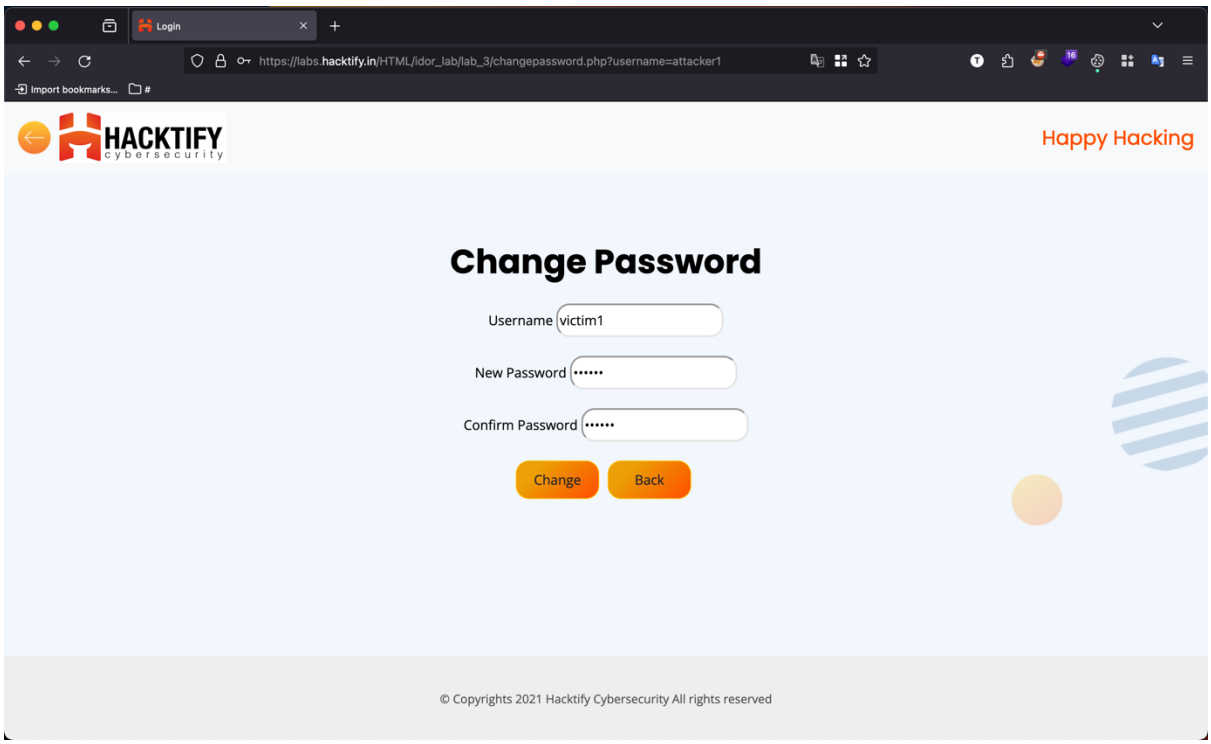


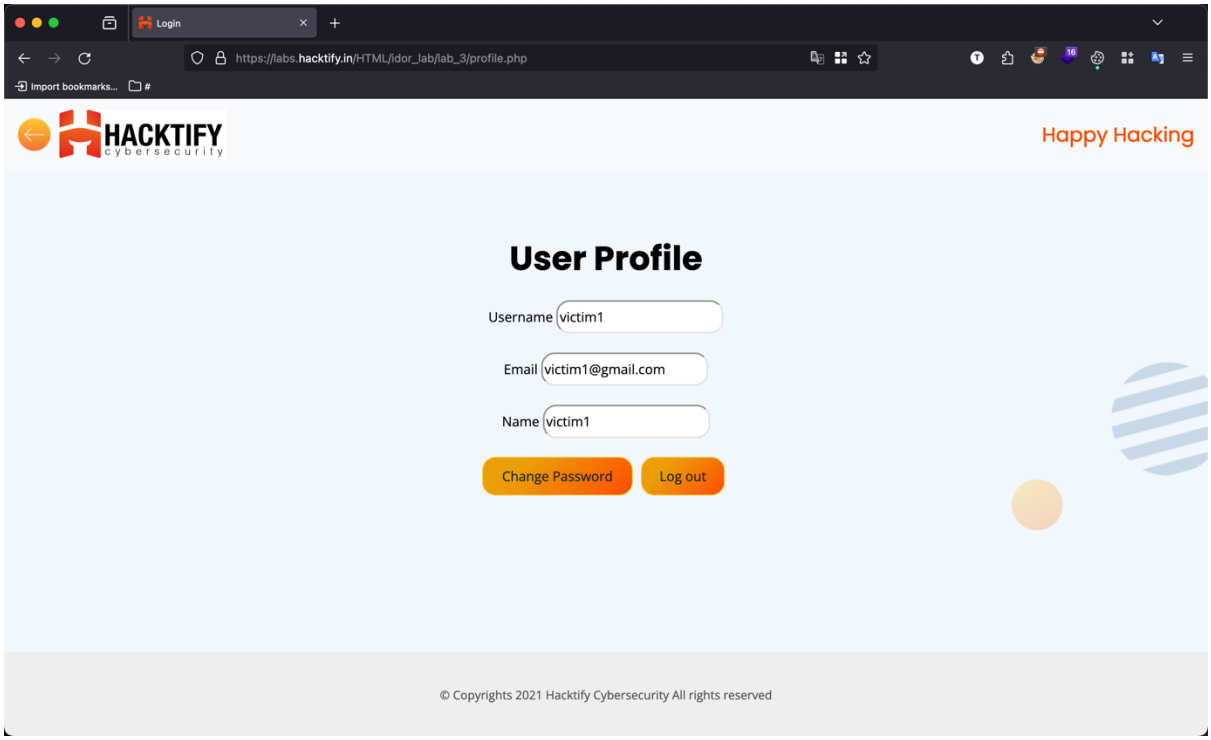
2.3. Someone changed my Password!

Reference	Risk Rating
Someone changed my Password!	**High**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a web security vulnerability where an attacker exploits inadequate access controls, accessing unauthorized resources or manipulating object references directly, potentially leading to unauthorized data exposure, modification, or deletion within the web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/idor_lab/lab_3/lab_3.php 2. There you will see User Login. 3. Click on “Register” and create 2 accounts (attacker, victim). 4. Later, Log into the attacker account. 5. Click on “Change Password”. You will be directed to https://labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=attacker1 6. Here, change the username to ‘victim’ username and update the password. 7. The ‘victim’ account password is changed without proper authentication and validation.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_3/lab_3.php	
Consequences of not Fixing the Issue	
Failure to address Insecure Direct Object References (IDOR) can result in unauthorized access to sensitive data, allowing attackers to view, modify, or delete confidential information. This may lead to data breaches, privacy violations, and compromise the overall security and integrity of the web application.	

Suggested Countermeasures
1. Implement proper access controls and authorization mechanisms to ensure that users can only access resources for which they have explicit permissions. 2. Use indirect references, such as unique identifiers or tokens, instead of relying on user-controlled input or sequential identifiers, to mitigate the risk of Insecure Direct Object References (IDOR) vulnerabilities.
References
1. https://owasp.org/www-community/attacks/Insecure_Direct_Object_References 2. https://portswigger.net/web-security/access-control/idor

Proof of Concept



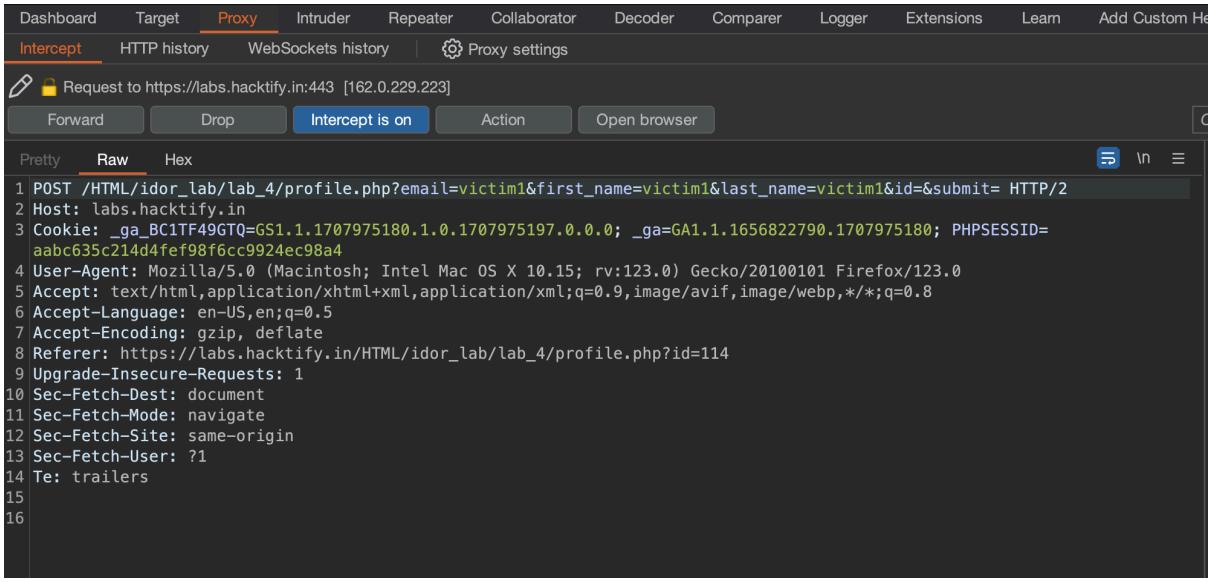


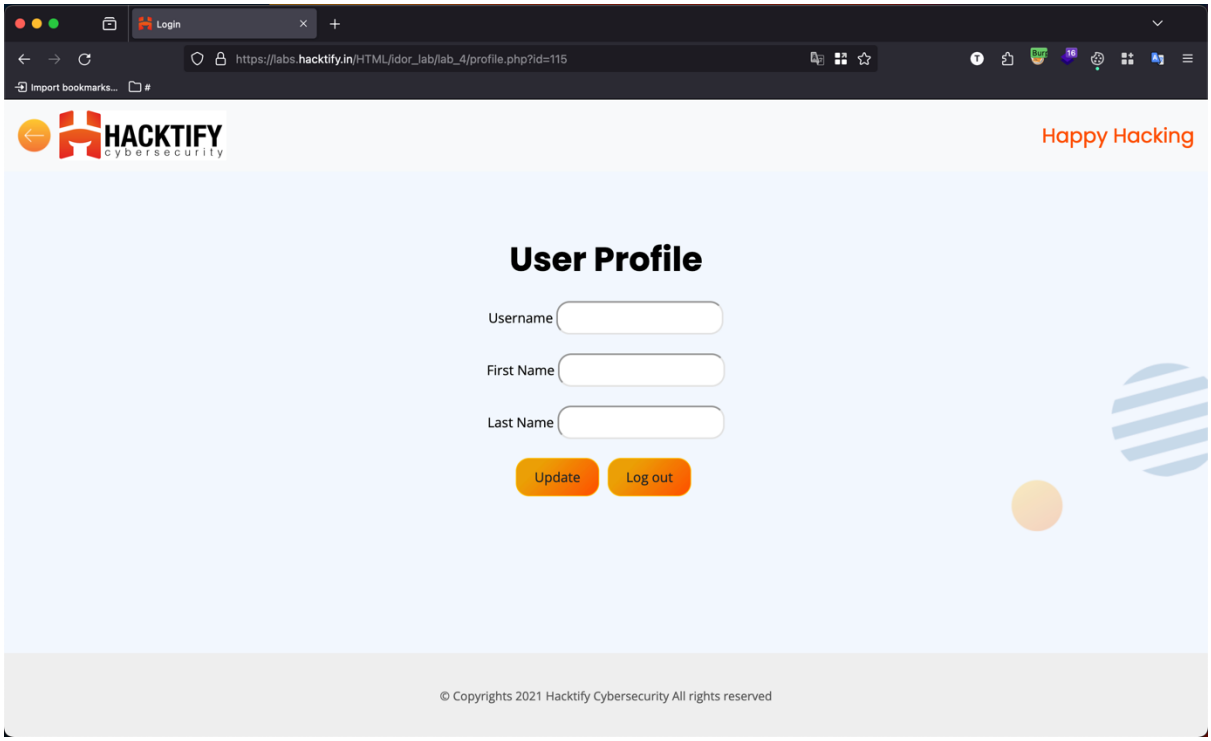
2.4. Change your methods!

Reference	Risk Rating
Change your methods!	**High**
Tools Used	
Web Browser - Firefox, BurpSuite	
Vulnerability Description	
Insecure Direct Object References (IDOR) is a web security vulnerability where an attacker exploits inadequate access controls, accessing unauthorized resources or manipulating object references directly, potentially leading to unauthorized data exposure, modification, or deletion within the web application.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/idor_lab/lab_4/lab_4.php 2. There you will see User Login. 3. Click on “Register” and create 2 accounts (attacker, victim). 4. Later, Log into the attacker account. 5. You will be directed to https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=114 6. In the User Profile, you have 3 input fields. Fill them with random strings and make sure to enter victim username in ‘username’ field. 7. Capture the request on BurpSuite, change the method form ‘GET’ to ‘POST’ and click on forward. 8. The User Profile of the victim is changed through attacker account.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/idor_lab/lab_4/lab_4.php	
Consequences of not Fixing the Issue	
Failure to address Insecure Direct Object References (IDOR) can result in unauthorized access to sensitive data, allowing attackers to view, modify, or delete confidential information. This may lead to	

data breaches, privacy violations, and compromise the overall security and integrity of the web application.
Suggested Countermeasures
3. Implement proper access controls and authorization mechanisms to ensure that users can only access resources for which they have explicit permissions. 4. Use indirect references, such as unique identifiers or tokens, instead of relying on user-controlled input or sequential identifiers, to mitigate the risk of Insecure Direct Object References (IDOR) vulnerabilities.
References
3. https://owasp.org/www-community/attacks/Insecure_Direct_Object_References 4. https://portswigger.net/web-security/access-control/idor

Proof of Concept




Login
https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=115
Import bookmarks...
HACKTIFY
cybersecurity
Happy Hacking
User Profile
Username
First Name
Last Name
Update
Log out
© Copyrights 2021 Hacktify Cybersecurity All rights reserved