Penetration Testing Report

Full Name: Anumandla Dhanush
Program: HCPT
Date: 20/02/2024

Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week 1 Lab**. 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 1 Lab** 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 - HTML Injection** **Labs.hacktify.in - Clickjacking**

3. Summary

Outlined is a Black Box Application Security assessment for the **Week 1 Lab**.

Total number of Sub-labs: 8 Sub-labs

High	Medium	Low
0	5	3

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

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

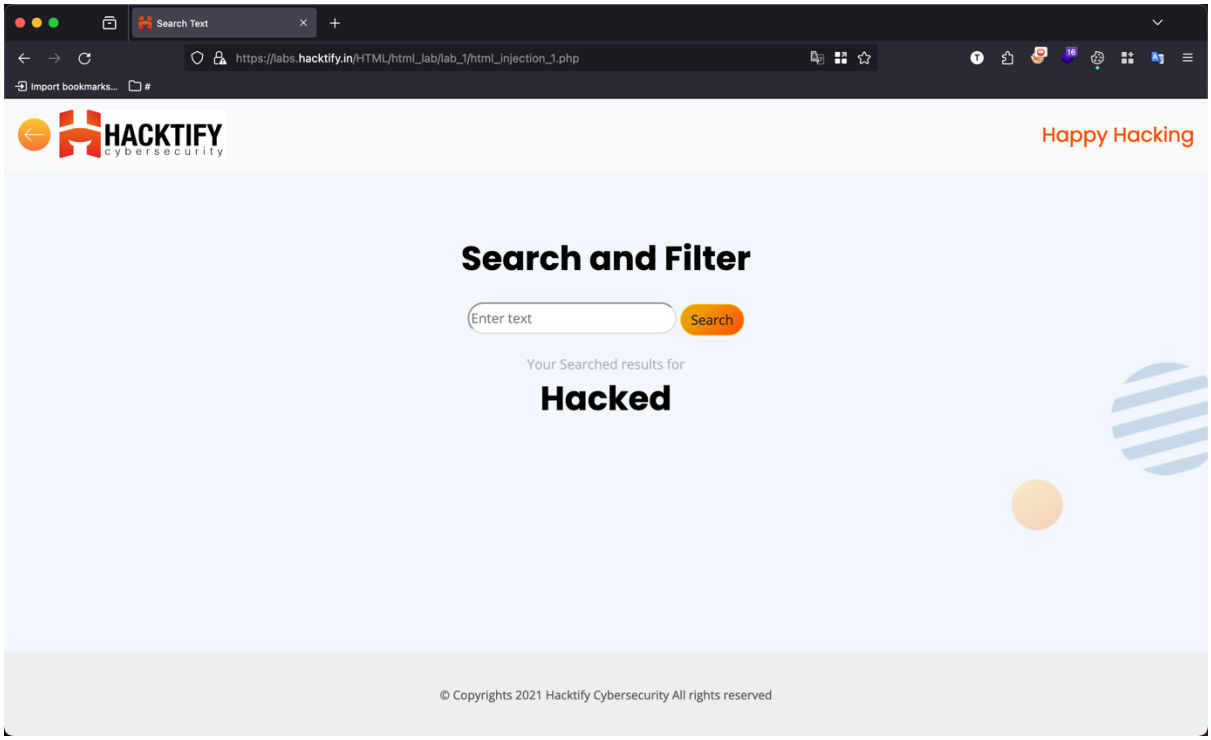
Low - **Number of Sub-labs with Easy difficulty level**

1. HTML Injection

1.1. HTML's are easy!

Reference	Risk Rating
HTML's are easy!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
HTML Injection is a web security vulnerability where an attacker injects malicious HTML code into a webpage, often through user inputs, leading to the unauthorized manipulation of content, potential execution of scripts, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Open https://labs.hacktify.in/HTML/html_lab/lab_1/html_injection_1.php 2. There is an input field (search box) 3. Enter the payload : Hacked ; Then click on “search” button. 4. The HTML Injection has executed and displayed on the webpage.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_1/html_injection_1.php	
Consequences of not Fixing the Issue	
Failure to address HTML Injection can result in the manipulation of webpage content, potential theft of sensitive user data, execution of malicious scripts, and the compromise of user trust, posing significant security risks to the affected web application.	
Suggested Countermeasures	
1. Implement strict input validation to sanitize user inputs, ensuring that any user-provided data is properly validated and does not contain malicious HTML code. 2. Use output encoding to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of HTML Injection vulnerabilities.	
References	
1. https://owasp.org/www-community/attacks/HTML_Injection 2.https://portswigger.net/web-security/injection/html-injection	

Proof of Concept

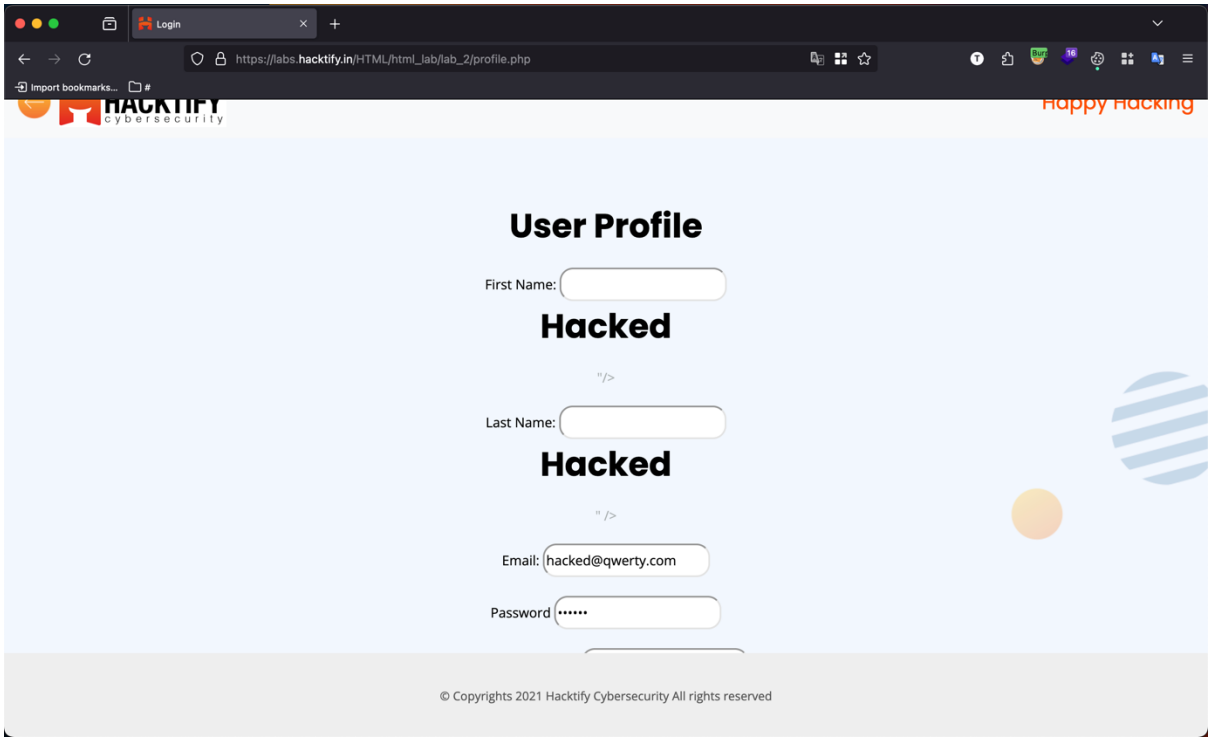


1.2. Let me Store them!

Reference	Risk Rating
Let me Store them!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
HTML Injection is a web security vulnerability where an attacker injects malicious HTML code into a webpage, often through user inputs, leading to the unauthorized manipulation of content, potential execution of scripts, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Open https://labs.hacktify.in/HTML/html_lab/lab_2/html_injection_2.php 2. There is a login and register functionality; register an account by clicking on “register”. 3. Here, the “First Name” and “Last Name” input fields are vulnerable. 4. Enter the Payload: "> Hacked in both the fields and click on register. 5. Login using the email and password, the HTML Injection payloads gets executed.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_2/html_injection_2.php	
Consequences of not Fixing the Issue	
Failure to address HTML Injection can result in the manipulation of webpage content, potential theft of sensitive user data, execution of malicious scripts, and the compromise of user trust, posing significant security risks to the affected web application.	
Suggested Countermeasures	
1. Implement strict input validation to sanitize user inputs, ensuring that any user-provided data is properly validated and does not contain malicious HTML code. 2. Use output encoding to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of HTML Injection vulnerabilities.	

References
1. https://owasp.org/www-community/attacks/HTML_Injection 2.https://portswigger.net/web-security/injection/html-injection

Proof of Concept

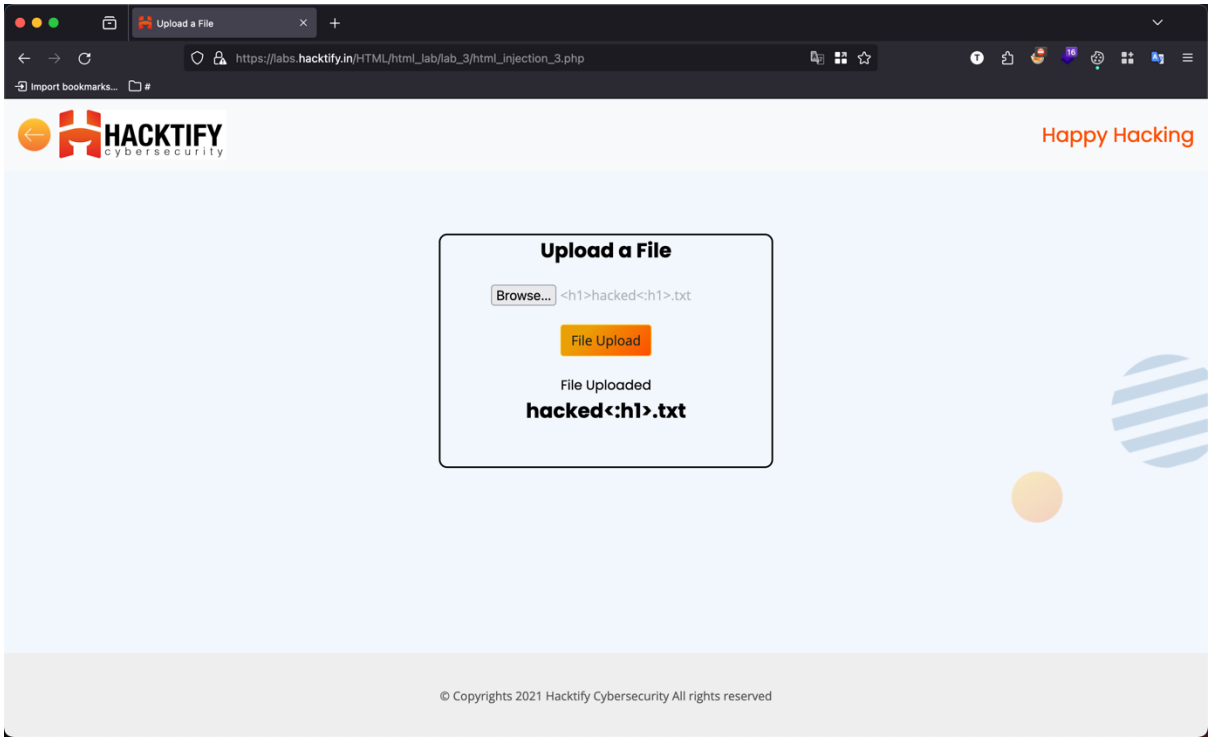


1.3. File Names are also vulnerable!

Reference	Risk Rating
File Names are also vulnerable!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
HTML Injection is a web security vulnerability where an attacker injects malicious HTML code into a webpage, often through user inputs, leading to the unauthorized manipulation of content, potential execution of scripts, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Open https://labs.hacktify.in/HTML/html_lab/lab_3/html_injection_3.php 2. There is a File upload functionality; Click on “Browse”. 3. Create a file and rename it with the payload: hacked (example: { hacked }.txt) 4. Upload the file and click on “File Upload” button. 5. The payload get executed and displayed in the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_3/html_injection_3.php	

Consequences of not Fixing the Issue
Failure to address HTML Injection can result in the manipulation of webpage content, potential theft of sensitive user data, execution of malicious scripts, and the compromise of user trust, posing significant security risks to the affected web application.
Suggested Countermeasures
1. Implement strict input validation to sanitize user inputs, ensuring that any user-provided data is properly validated and does not contain malicious HTML code. 2. Use output encoding to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of HTML Injection vulnerabilities.
References
1. https://owasp.org/www-community/attacks/HTML_Injection 2.https://portswigger.net/web-security/injection/html-injection

Proof of Concept

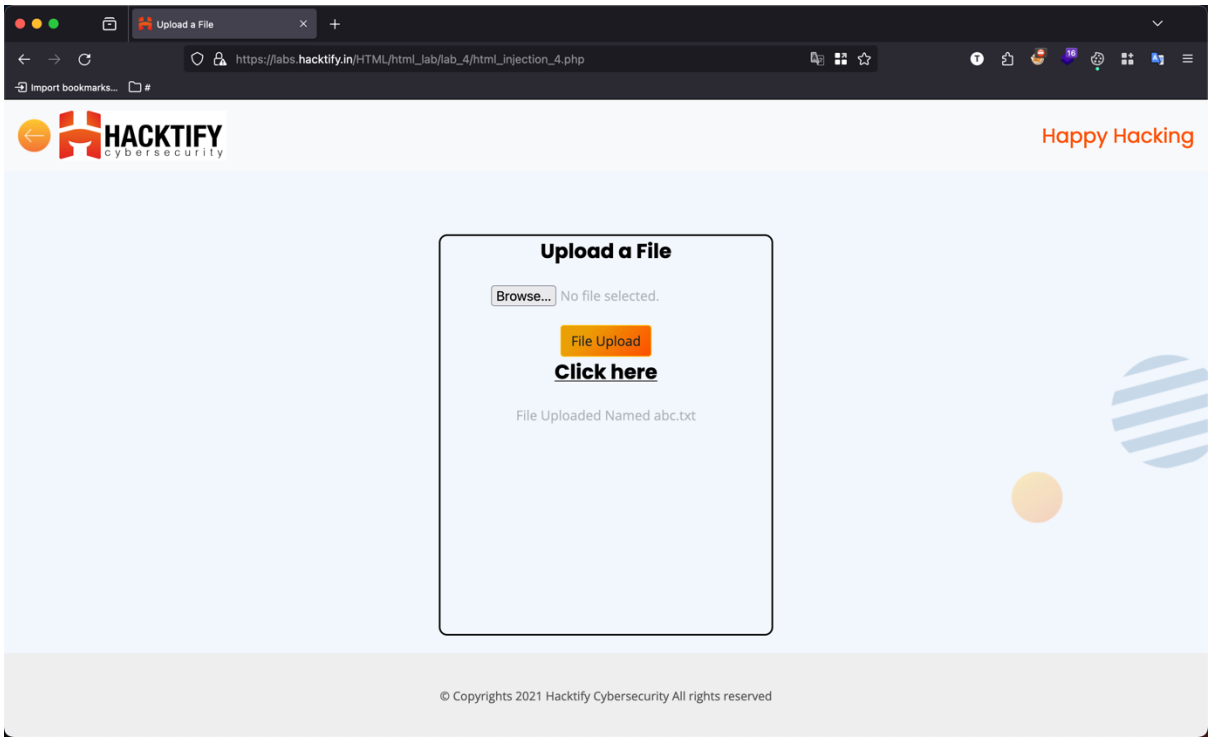


1.4. File Content and HTML Injection a perfect pair!

Reference	Risk Rating
File Content and HTML Injection a perfect pair!	**Medium**
Tools Used	
Web Browser - Firefox	

Vulnerability Description
HTML Injection is a web security vulnerability where an attacker injects malicious HTML code into a webpage, often through user inputs, leading to the unauthorized manipulation of content, potential execution of scripts, and the compromise of user interactions with the affected web application.
How It Was Discovered
Manual Analysis: 1. Open https://labs.hacktify.in/HTML/html_lab/lab_4/html_injection_4.php 2. There is a File upload functionality; create a file(like abc.txt) 3. Copy the payload: <u>Click here</u> and paste it in the file(abc.txt) 4. Click on “Browse” and select the file(abc.txt) containing the payload inside. 5. Click on “File Upload”. The page gets loaded and executes the payload. 6. Clicking on the payloads output (“Click Here”) will be redirected to other site.
Vulnerable URLs
https://labs.hacktify.in/HTML/html_lab/lab_4/html_injection_4.php
Consequences of not Fixing the Issue
Failure to address HTML Injection can result in the manipulation of webpage content, potential theft of sensitive user data, execution of malicious scripts, and the compromise of user trust, posing significant security risks to the affected web application.
Suggested Countermeasures
1. Implement strict input validation to sanitize user inputs, ensuring that any user-provided data is properly validated and does not contain malicious HTML code. 2. Use output encoding to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of HTML Injection vulnerabilities.
References
1. https://owasp.org/www-community/attacks/HTML_Injection 2.https://portswigger.net/web-security/injection/html-injection

Proof of Concept

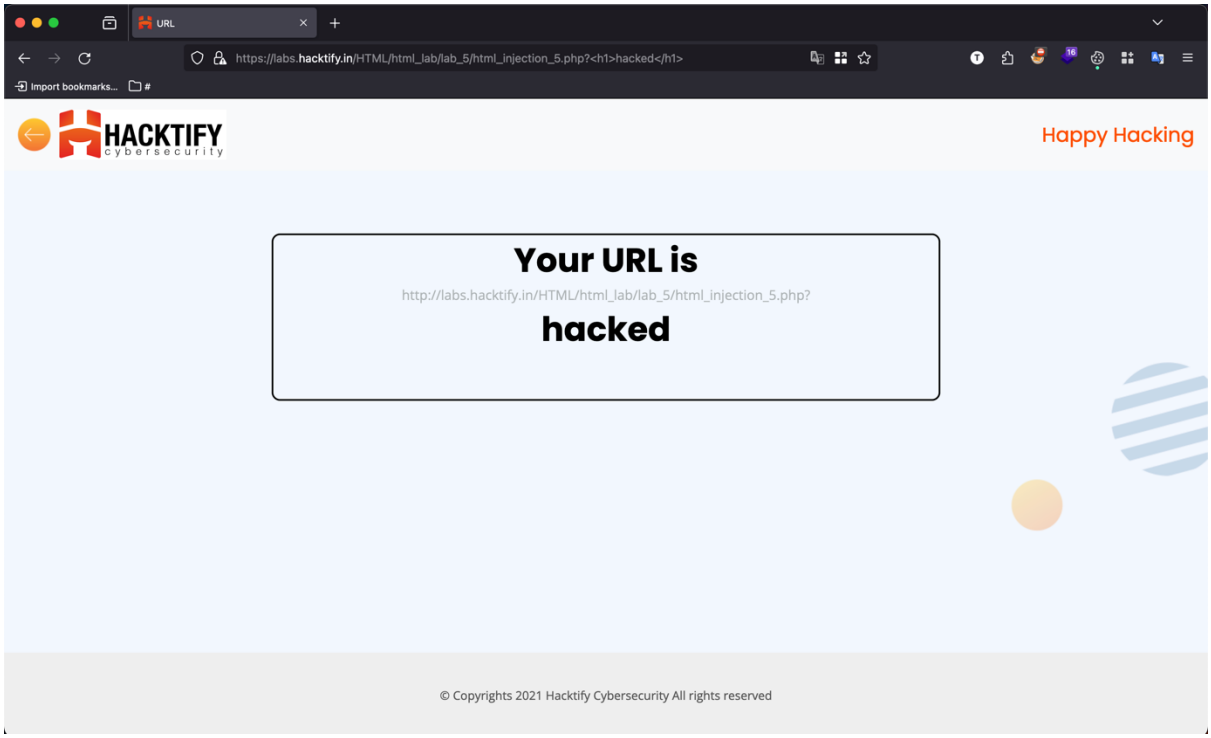


1.5. Injecting HTML using URL

Reference	Risk Rating
Injecting HTML using URL	**Low**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
HTML Injection is a web security vulnerability where an attacker injects malicious HTML code into a webpage, often through user inputs, leading to the unauthorized manipulation of content, potential execution of scripts, and the compromise of user interactions with the affected web application.	
How It Was Discovered	
Manual Analysis: 1. Open https://labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php 2. Copy the URL from the browser. 3. Add the payload: ? hacked at the end of the URL. 4. Paste the crafted URL in the browser search bar and hit “Enter”. 5. The payload gets executed and displayed in the web page.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php	
Consequences of not Fixing the Issue	
Failure to address HTML Injection can result in the manipulation of webpage content, potential theft of sensitive user data, execution of malicious scripts, and the compromise of user trust, posing significant security risks to the affected web application.	
Suggested Countermeasures	

1. Implement strict input validation to sanitize user inputs, ensuring that any user-provided data is properly validated and does not contain malicious HTML code. 2. Use output encoding to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of HTML Injection vulnerabilities.
References
1. https://owasp.org/www-community/attacks/HTML_Injection 2.https://portswigger.net/web-security/injection/html-injection

Proof of Concept

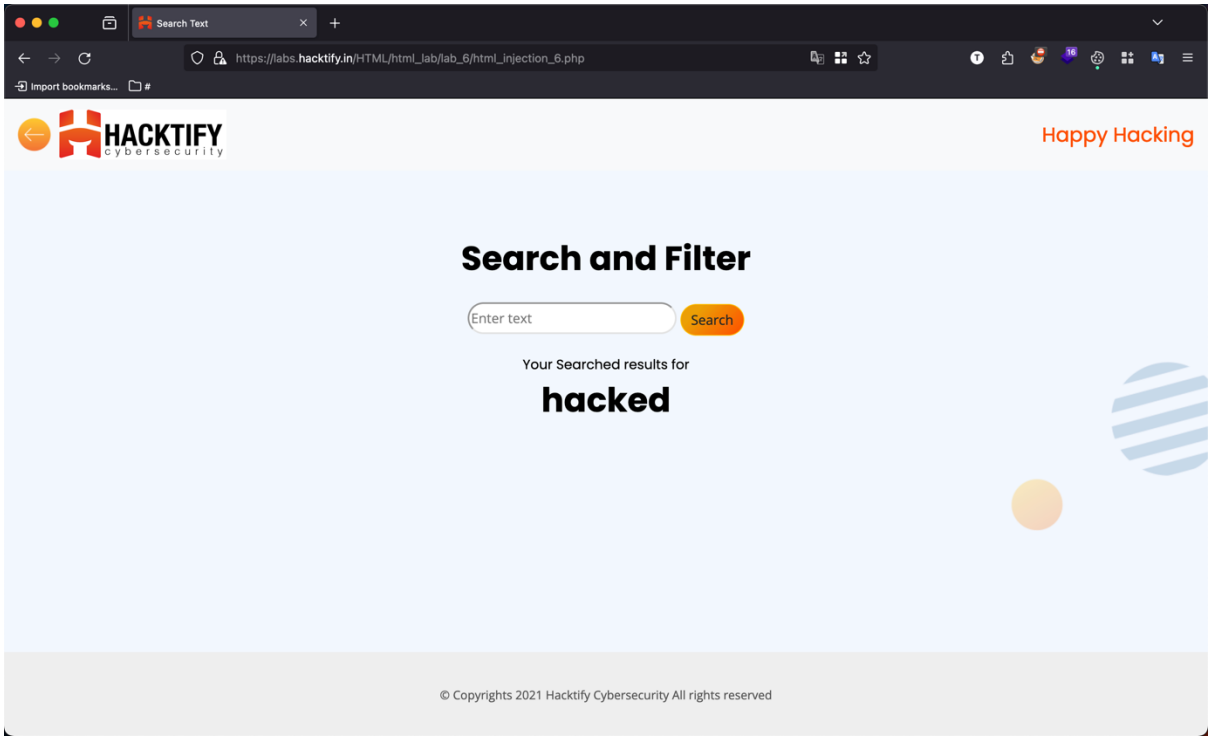


1.6. Encode IT!

Reference	Risk Rating
Encode IT!	**Medium**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
HTML Injection is a web security vulnerability where an attacker injects malicious HTML code into a webpage, often through user inputs, leading to the unauthorized manipulation of content, potential execution of scripts, and the compromise of user interactions with the affected web application.	
How It Was Discovered	

Manual Analysis: 1. Open https://labs.hacktify.in/HTML/html_lab/lab_6/html_injection_6.php 2. There is an input search field. 3. Enter the payload: %3ch1%3ehacked%3c/h1%3e and click on “search”. 4. The web application is filtering <,> tags. Hence, we URL Encode them. 5. The payload gets executed bypassing the filters and displayed on the web page.
Vulnerable URLs
https://labs.hacktify.in/HTML/html_lab/lab_6/html_injection_6.php
Consequences of not Fixing the Issue
Failure to address HTML Injection can result in the manipulation of webpage content, potential theft of sensitive user data, execution of malicious scripts, and the compromise of user trust, posing significant security risks to the affected web application.
Suggested Countermeasures
1. Implement strict input validation to sanitize user inputs, ensuring that any user-provided data is properly validated and does not contain malicious HTML code. 2. Use output encoding to sanitize dynamic content before rendering it in web pages, preventing the execution of injected scripts and reducing the risk of HTML Injection vulnerabilities.
References
1. https://owasp.org/www-community/attacks/HTML_Injection 2.https://portswigger.net/web-security/injection/html-injection

Proof of Concept

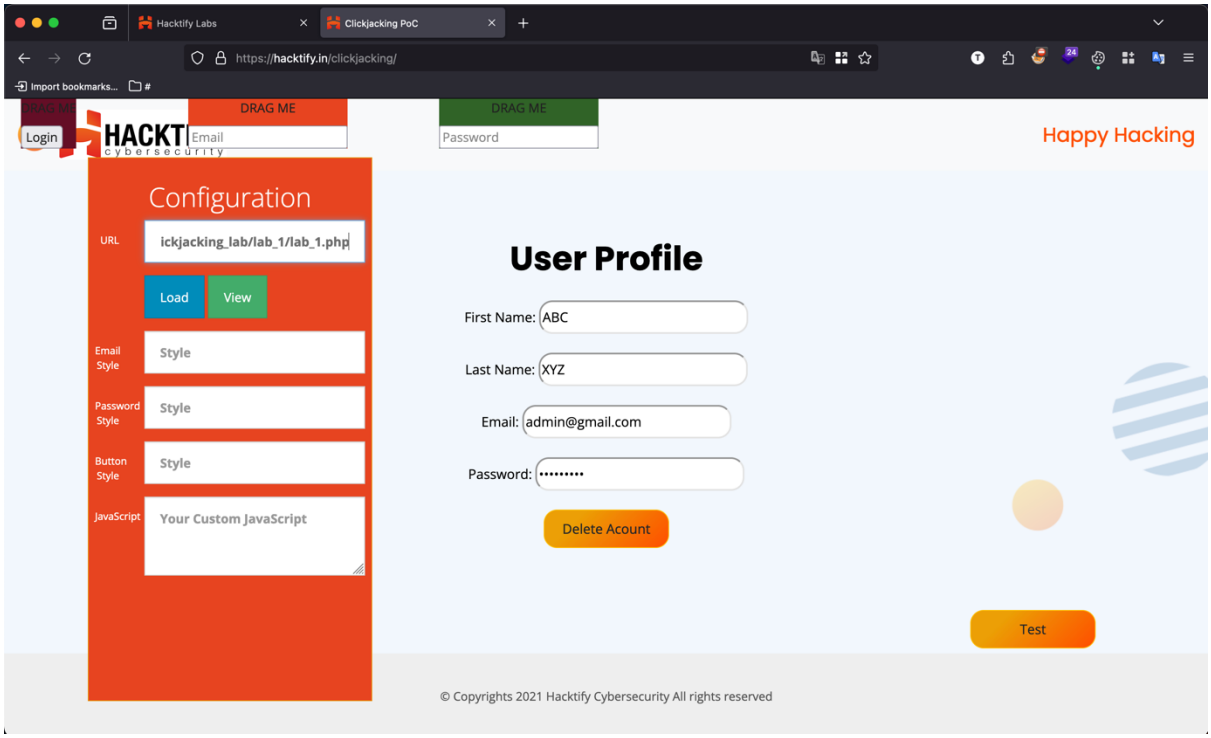


2. Clickjacking

2.1. Let's Hijack!

Reference	Risk Rating
Let's Hijack!	**Low**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Clickjacking is a cybersecurity attack where an attacker tricks a user into unknowingly clicking on a disguised or invisible element, often through the overlay of a legitimate webpage, to perform actions without the user's consent. This technique is employed to hijack clicks and potentially exploit user interactions for malicious purposes.	
How It Was Discovered	
Manual Analysis: 1. Open https://hacktify.in/clickjacking 2. In the URL input field enter the target website link that we want to test for Clickjacking vulnerability; which is https://labs.hacktify.in/HTML/clickjacking_lab/lab_1/lab_1.php 3. Then click on Load. The target website gets loaded into the iframe confirming the Clickjacking vulnerability.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/clickjacking_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
Failure to address Clickjacking can result in unauthorized actions performed by users unknowingly, leading to potential data theft, account manipulation, or unintended interactions with sensitive functionalities, posing significant risks to the security and integrity of the web application.	
Suggested Countermeasures	
1. Implement framebusting techniques, such as utilizing the X-Frame-Options header, to prevent the web page from being embedded within iframes, thereby mitigating Clickjacking risks. 2. Utilize Content Security Policy (CSP) to define and control the sources from which content can be loaded, reducing the risk of malicious embedding and providing an additional layer of defense against Clickjacking attacks.	
References	
1. https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html 2. https://portswigger.net/web-security/clickjacking	

Proof of Concept



2.2. Re-Hijack!

Reference	Risk Rating
Re-Hijack!	**Low**
Tools Used	
Web Browser - Firefox	
Vulnerability Description	
Clickjacking is a cybersecurity attack where an attacker tricks a user into unknowingly clicking on a disguised or invisible element, often through the overlay of a legitimate webpage, to perform actions without the user's consent. This technique is employed to hijack clicks and potentially exploit user interactions for malicious purposes.	
How It Was Discovered	
Manual Analysis: 1. Open https://hacktify.in/clickjacking 2. In the URL input field enter the target website link that we want to test for Clickjacking vulnerability; which is https://labs.hacktify.in/HTML/clickjacking_lab/lab_2/lab_2.php 3. Then click on Load. The target website gets loaded into the iframe confirming the Clickjacking vulnerability.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/clickjacking_lab/lab_2/lab_2.php	
Consequences of not Fixing the Issue	

Failure to address Clickjacking can result in unauthorized actions performed by users unknowingly, leading to potential data theft, account manipulation, or unintended interactions with sensitive functionalities, posing significant risks to the security and integrity of the web application.
Suggested Countermeasures
1. Implement framebusting techniques, such as utilizing the X-Frame-Options header, to prevent the web page from being embedded within iframes, thereby mitigating Clickjacking risks. 2. Utilize Content Security Policy (CSP) to define and control the sources from which content can be loaded, reducing the risk of malicious embedding and providing an additional layer of defense against Clickjacking attacks.
References
1. https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html 2. https://portswigger.net/web-security/clickjacking

Proof of Concept