Penetration Testing Report

Full Name	**:**	**Suresh Babu B**
Program	**:**	**HCPT**
Date	**:**	**17-02-2025**

Introduction

This document provides a detailed Proof of Concept (PoC) for HTML Injection and Cross-Site Scripting (XSS) vulnerabilities encountered in the Hacktify.in Labs training platform. The document includes observed vulnerabilities, exploitation techniques, payloads, and mitigation strategies.

1.Objective

The objective of this assessment is to identify and exploit HTML Injection and Cross-Site Scripting (XSS) vulnerabilities in the target application. These vulnerabilities can allow attackers to manipulate the web page’s content or execute malicious scripts in a user’s browser, leading to data theft, phishing attacks, and other security risks.

2.Scope

- Modules Tested: HTML input fields, query parameters, and reflected content
- Testing Type: Black-box and Gray-box testing
- Vulnerability Types:
    - HTML Injection
    - DOM-based XSS
    - Reflected XSS

Hacktify.in	**HTML Injection, XSS Injection**

3.Summary

During the security assessment, multiple injection points were identified that allowed malicious input to be reflected in the application's DOM. The vulnerabilities could be exploited to execute JavaScript, steal session cookies, deface the webpage, or redirect users to phishing sites.

Total number of Sub-labs: 17 Sub-lab

High	Medium	Low
04	**05**	**08**

High	-	**Four Sub-labs with hard difficulty level.**
Medium	-	**Five of Sub-labs with Medium difficulty level.**
Low	-	**Nine of Sub-labs with Easy difficulty level.**

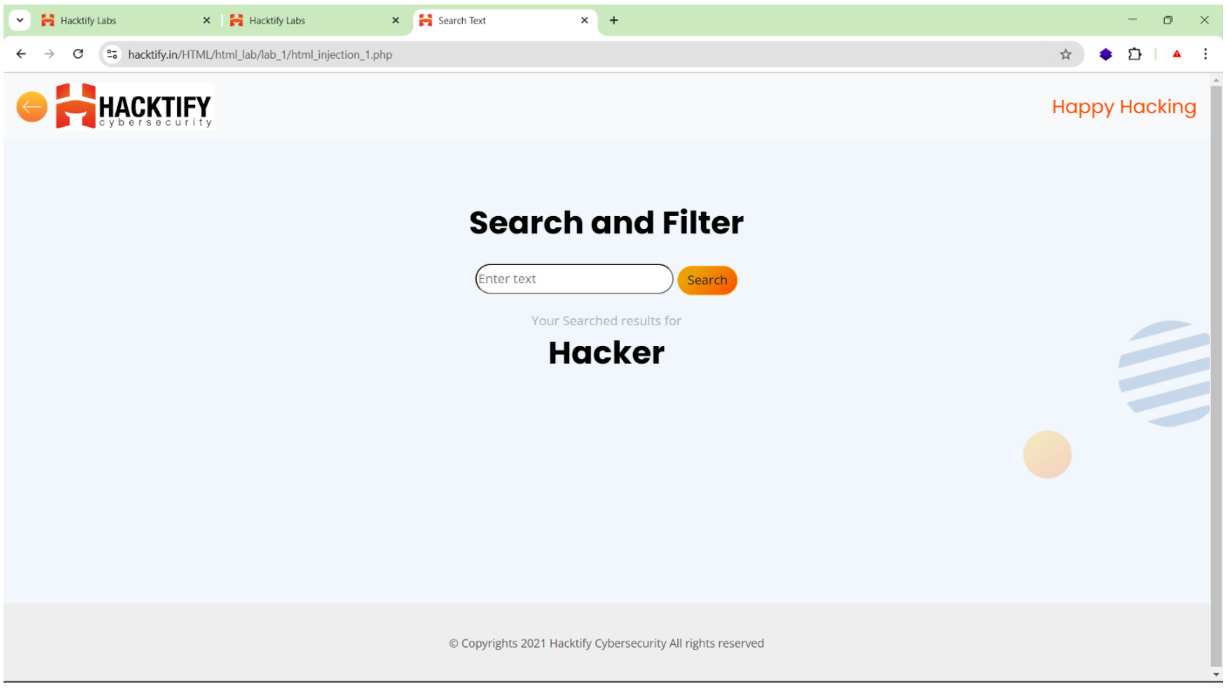
1. HTML Injection

1.1. HTML's Are Easy!

Reference	**Risk Rating**
HTML's Are Easy !	**Low**
Tools Used	
Burp Suite	
Vulnerability Description	
HTML Injection allows attackers to manipulate web page content.	
How It Was Discovered	
Manual testing identified an input field rendering HTML tags.	
Vulnerable URLs	
https://hacktify.in/HTML/html_lab/lab_1/html_injection_1.php	

Consequences of not Fixing the Issue
Attackers can deface the webpage, perform phishing, or inject malicious links
Suggested Countermeasures
Sanitize user input by encoding HTML entities.
References
OWASP HTML Injection Guide

Proof of Concept

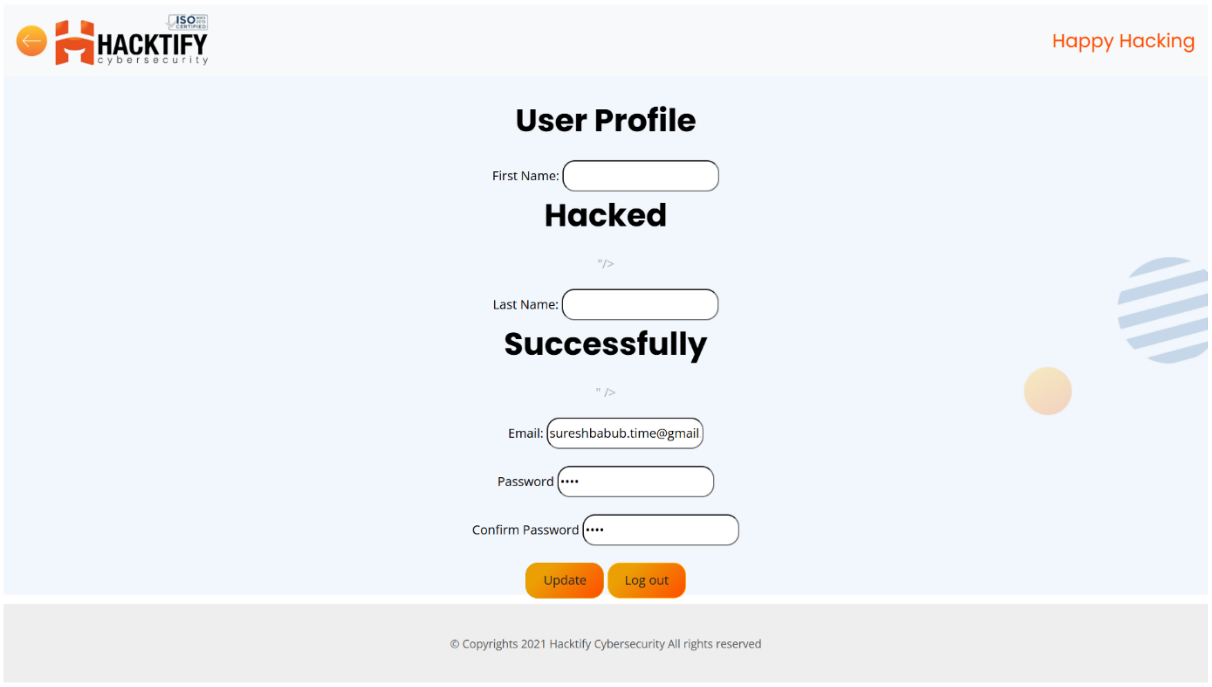

Hacktify Labs
Hacktify Labs
Search Text
hacktify.in/HTML/html_lab/lab_1/html_injection_1.php
HACKTIFY cybersecurity
Happy Hacking
Search and Filter
Enter text
Search
Your Searched results for
Hacker
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.2. Let Me Store Them!

Reference	Risk Rating
Let Me Store Them!	**Low**
Tools Used	
Burp Suite	

Vulnerability Description
HTML Injection enables attackers to modify displayed content on a webpage
How It Was Discovered
Identified through manual testing by injecting HTML tags into input fields.
Vulnerable URLs
https://hacktify.in/HTML/html_lab/lab_2/html_injection_2.php
Consequences of not Fixing the Issue
Can lead to UI manipulation, phishing attempts, or misleading information display.
Suggested Countermeasures
Implement input validation and encode user-supplied data.
References
OWASP HTML Injection Guide

Proof of Concept


Happy Hacking

User Profile

First Name:

Hacked

"/>

Last Name:

Successfully

" />

Email: sureshbabub.time@gmail

Password ••••

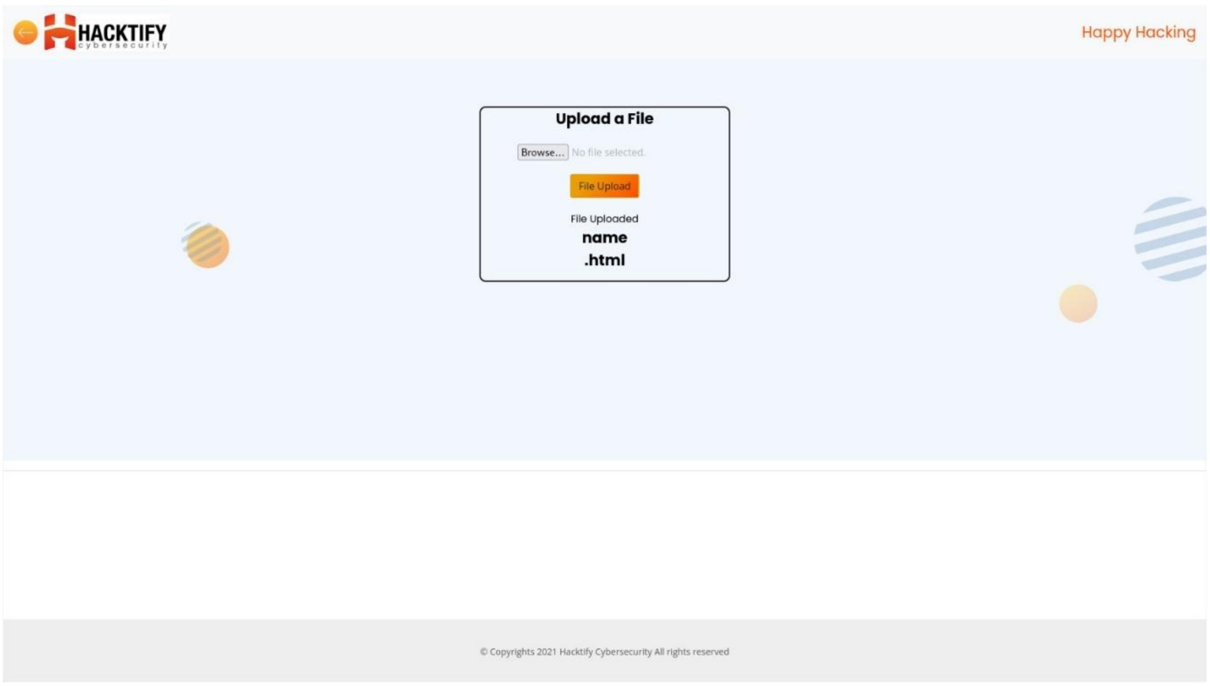
Confirm Password ••••

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.3. File Names Are Also Vulnerable!

Reference	**Risk Rating**
File Names Are Also Vulnerable!	**Low**
Tools Used	
Burp Suite, Web Browser	
Vulnerability Description	
The application allows users to upload HTML files without proper validation, leading to stored HTML injection.	
How It Was Discovered	
By manually uploading a crafted .html file containing injected HTML tags and observing the rendered output.	
Vulnerable URLs	
https://hacktify.in/HTML/html_lab/lab_3/html_injection_3.php	
Consequences of not Fixing the Issue	
Attackers can execute **stored HTML injection**, misleading users or tricking them into phishing attacks.	
Suggested Countermeasures	
-Restrict file types to non-executable formats like .png, .jpg, or .pdf. -Implement **server-side validation** to reject .html, .js, or other potentially harmful files. -Enforce **Content Security Policy (CSP)** to mitigate content injection risks.	
References	
OWASP Unrestricted File Upload Guide	

Proof of Concept

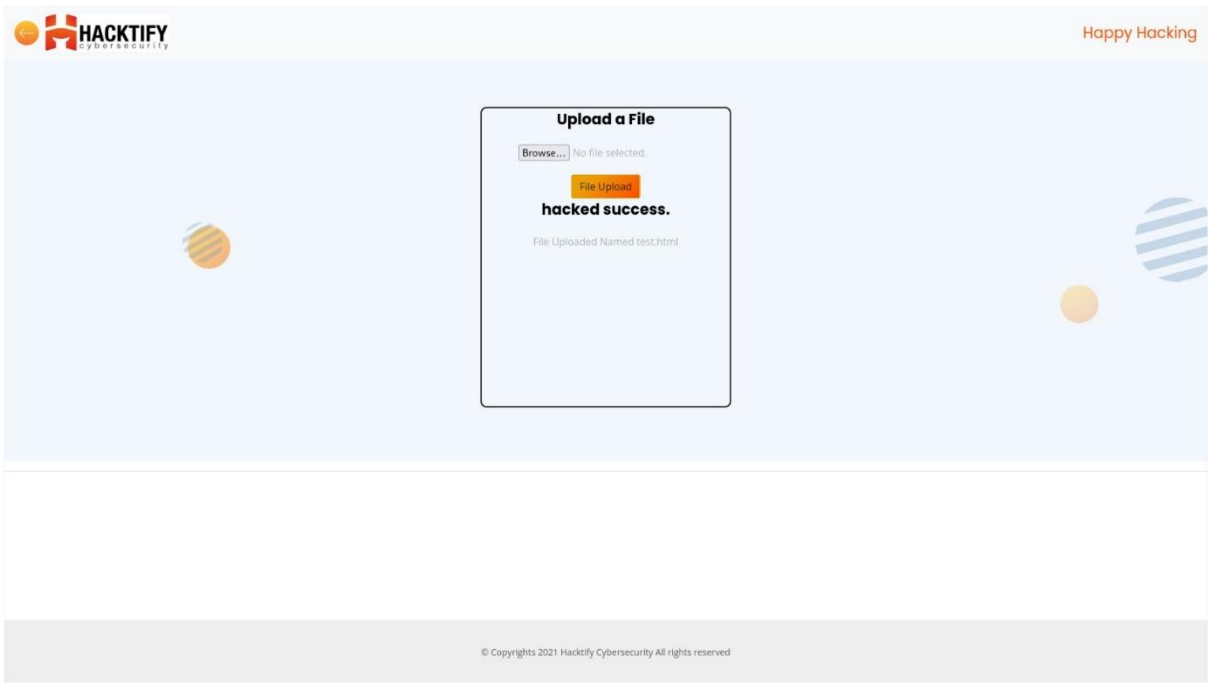


1.4. File Content And HTML Injection A Perfect Pair!

Reference	**Risk Rating**
File Content And HTML Injection A Perfect Pair!	**Medium**
Tools Used	
Burp Suite, Web Browser	
Vulnerability Description	
The application allows uploading an HTML file without restrictions, leading to potential stored HTML injection.	
How It Was Discovered	
A manually crafted test.html file containing injected HTML was uploaded and successfully executed when accessed	

Vulnerable URLs
https://hacktify.in/HTML/html_lab/lab_4/html_injection_4.php
Consequences of not Fixing the Issue
-Stored HTML injection may mislead users with fake content. -Attackers could exploit this for phishing or social engineering attacks.
Suggested Countermeasures
-Restrict file types to only necessary formats (e.g., .png, .jpg, .pdf). -Validate and sanitize uploaded files on the server-side. -Enforce **Content-Disposition: attachment** to prevent execution.
References
OWASP Unrestricted File Upload Guide

Proof of Concept



Upload a File

No file selected.

hacked success.

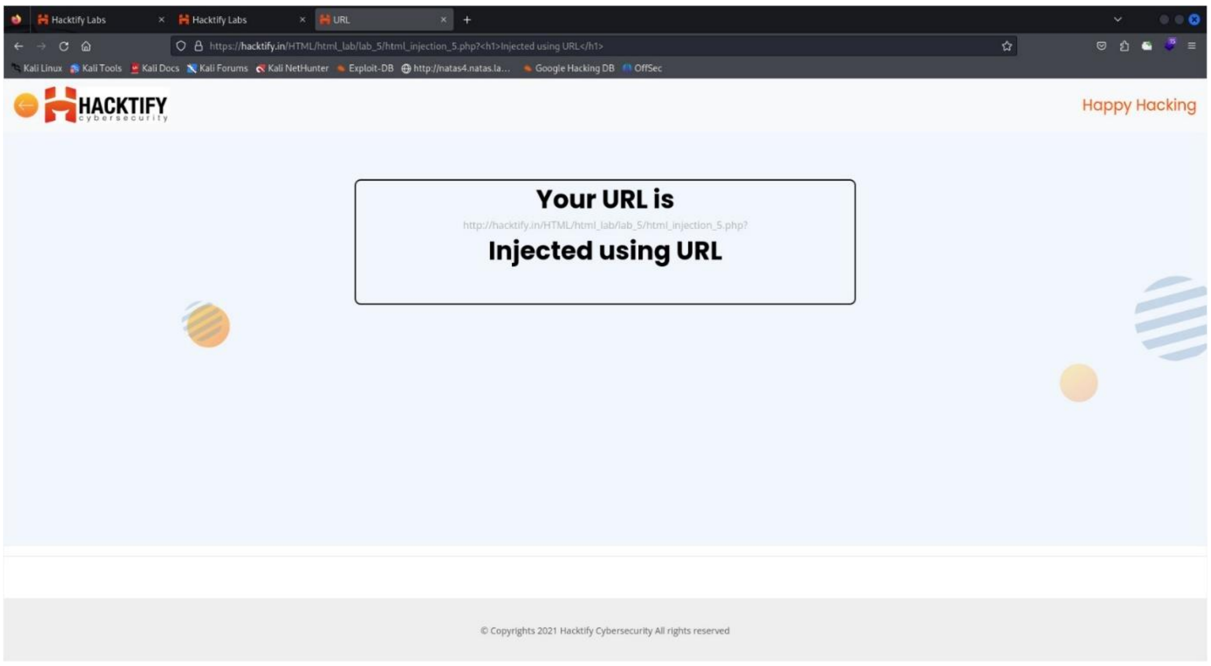
File Uploaded Named test.html

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.5. Injecting HTML Using URL

Reference	**Risk Rating**
Injecting HTML Using URL	**Medium**
Tools Used	
Burp Suite	
Vulnerability Description	
The application is vulnerable to **HTML injection**, allowing attackers to manipulate page content by injecting HTML through URL parameters.	
How It Was Discovered	
Manually tested by injecting HTML elements into a parameterized URL, which was reflected on the webpage.	
Vulnerable URLs	
https://hacktify.in/HTML/html_lab/lab_5/html_injection_5.php?<h1>name</h1>	
Consequences of not Fixing the Issue	
-Attackers can alter page content, misleading users. -Could be used for phishing or social engineering attacks.	
Suggested Countermeasures	
-Sanitize user input before rendering. -Use proper encoding to prevent direct HTML execution. -Implement Content Security Policy (CSP) to mitigate injected content risks.	
References	
OWASP HTML Injection Guide	

Proof of Concept



https://hacktify.in/HTML/html_lab/lab_5/html_injection_5.php?<h1>Injected using URL</h1>

Your URL is

http://hacktify.in/HTML/html_lab/lab_5/html_injection_5.php?

Injected using URL

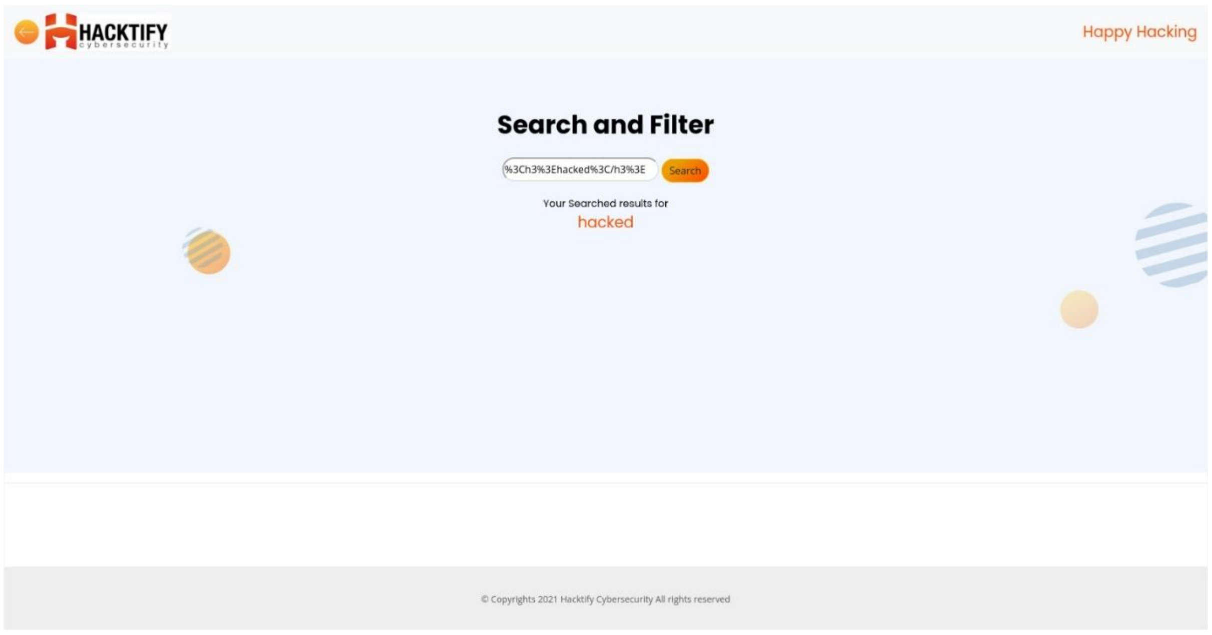
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.6. Encoding It!

Reference	**Risk Rating**
Encoding it!	**High**
Tools Used	
Burp Suite, Web Browser	
Vulnerability Description	
The search functionality is vulnerable to **HTML injection**, allowing attackers to insert HTML tags, which are rendered on the webpage instead of being treated as plain text.	
How It Was Discovered	
Tested by entering encoded HTML tags (%3Ch3%3Ehacked%3C/h3%3E) into the search bar and observing their execution on the page.	
Vulnerable URLs	
https://hacktify.in/HTML/html_lab/lab_6/html_injection_6.php	

Consequences of not Fixing the Issue
-Attackers can manipulate displayed content. -Could lead to phishing attacks, misleading information, or defacement.
Suggested Countermeasures
-Implement input validation to allow only expected characters. -Encode output before displaying user-generated input. -Use security headers like **Content Security Policy (CSP)**.
References
OWASP HTML Injection Guide

Proof of Concept



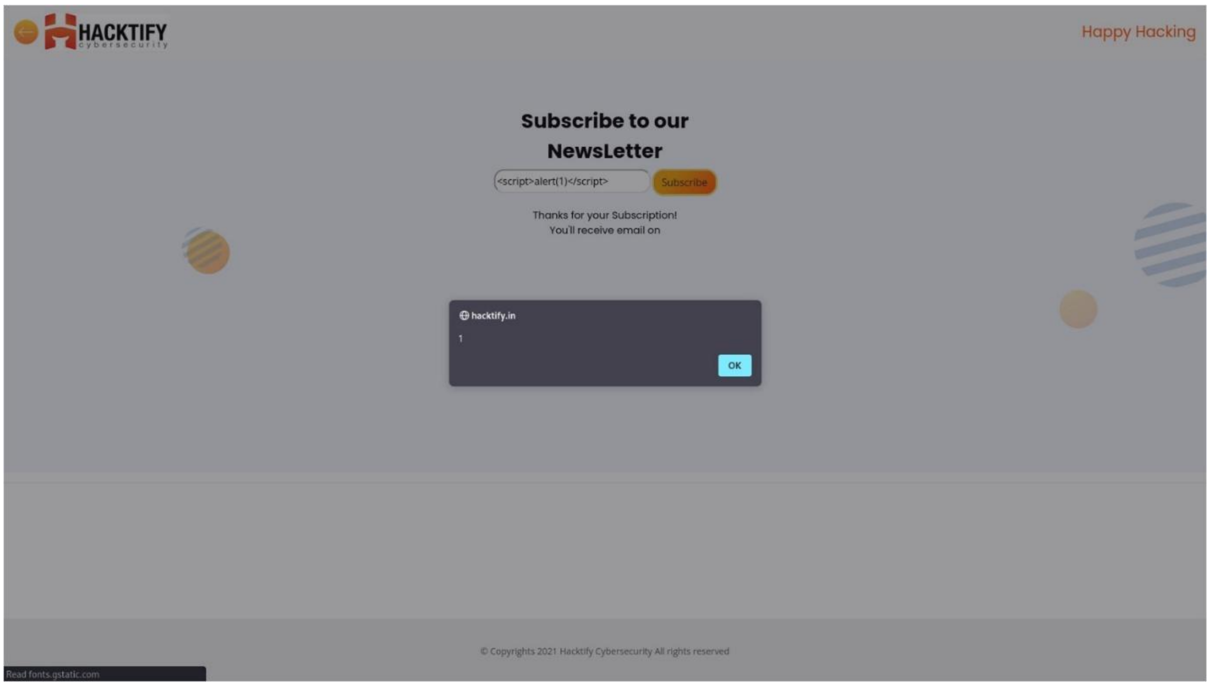
2. Cross Site Scripting

2.1. Let's Do It!

Reference	**Risk Rating**
Let's Do It!	**Low**

Tools Used
Burp Suite
Vulnerability Description
The search functionality does not properly sanitize user input, allowing **Cross-Site Scripting (XSS)** attacks. Injected scripts execute in the victim's browser, potentially leading to session hijacking or phishing.
How It Was Discovered
Tested by entering a simple XSS payload (<script>alert(1)</script>) in the search bar and observing its execution in the browser.
Vulnerable URLs
https://hacktify.in/HTML/xss_lab/lab_1/lab_1.php
Consequences of not Fixing the Issue
-Malicious scripts can execute in users' browsers. -Attackers can steal session cookies, impersonate users, or redirect them to phishing sites.
Suggested Countermeasures
-Implement **input validation** to restrict special characters. -Encode user-generated content before displaying it. -Use **Content Security Policy (CSP)** to block inline scripts.
References
OWASP XSS Prevention Cheat Sheet

Proof of Concept



Subscribe to our NewsLetter

<script>alert(1)</script>

Thanks for your Subscription!
You'll receive email on

hacktify.in

1

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

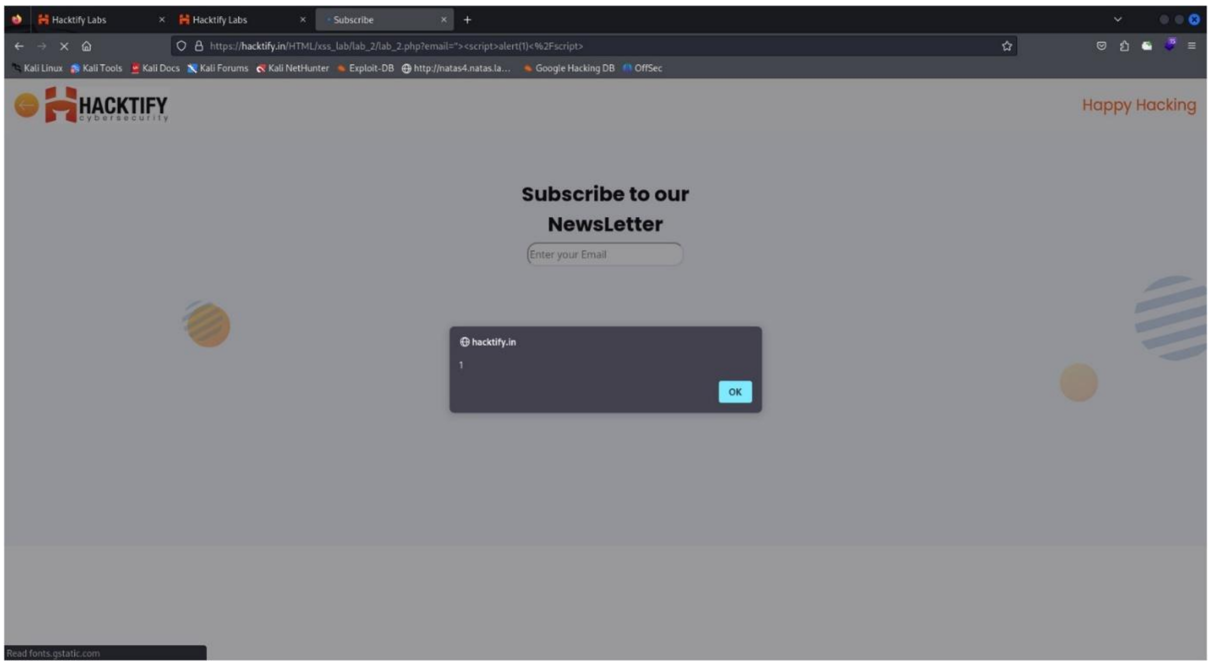
Read fonts.gstatic.com

2.2. Balancing Is Important In Life!

Reference	**Risk Rating**
Balancing Is Important In Life!	**Low**
Tools Used	
Burp Suite	
Vulnerability Description	
The search bar fails to properly sanitize input when passing the **email parameter**, allowing an attacker to inject JavaScript. The injected script executes in the victim's browser, leading to **Reflected XSS**.	
How It Was Discovered	
Manually tested by injecting "><script>alert(1)</script> into the **email** parameter in the search bar and observing JavaScript execution.	
Vulnerable URLs	

https://hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email=%22%3E%3Cscript%3Ealert%281%29%3C%2Fscript%3E
Consequences of not Fixing the Issue
-Attackers can execute malicious JavaScript in users' browsers. -Can lead to **session hijacking, cookie theft, or phishing attacks**. -Attackers may redirect users to malicious sites or keylog their inputs.
Suggested Countermeasures
-Sanitize user inputs and encode special characters. -Implement **output encoding** (e.g., htmlspecialchars() in PHP). -Use **Content Security Policy (CSP)** to block inline JavaScript execution.
References
OWASP XSS Prevention Cheat Sheet

Proof of Concept

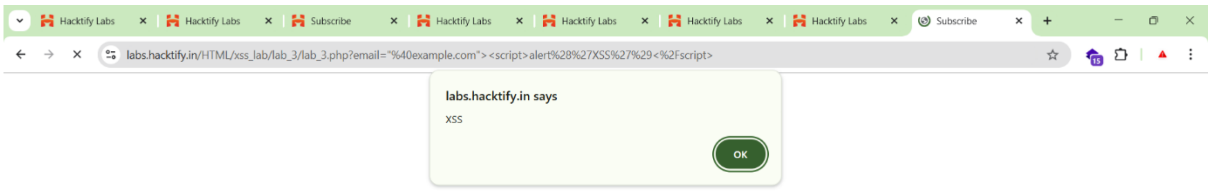


https://hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email="><script>alert(1)<%2Fscript>

2.3. XSS Is Everywhere!

Reference	**Risk Rating**
XSS Is Everywhere!	**Low**
Tools Used	
Burp Suite, Web Browser	
Vulnerability Description	
The application does not properly validate and sanitize user input in the **email parameter** of the search bar. By injecting JavaScript after breaking the expected email format, an attacker can execute **Reflected XSS** in the victim's browser.	
How It Was Discovered	
Manually tested by injecting the payload "@example.com"><script>alert('XSS')</script> into the **email** parameter and observing script execution.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_3/lab_3.php?email=%22%40example.com%22%3E%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E	
Consequences of not Fixing the Issue	
-Attackers can execute JavaScript in the victim's browser. -Users may be tricked into revealing sensitive data through phishing attacks. -Attackers can steal session cookies and impersonate users.	
Suggested Countermeasures	
Validate input: Ensure only valid email formats are accepted. **Encode output:** Use htmlspecialchars() or similar encoding methods. **Implement CSP (Content Security Policy):** Block inline JavaScript execution.	
References	
OWASP XSS Prevention Cheat Sheet	

Proof of Concept


labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php?email="%40example.com"><script>alert%28%27XSS%27%29<%2Fscript>

labs.hacktify.in says

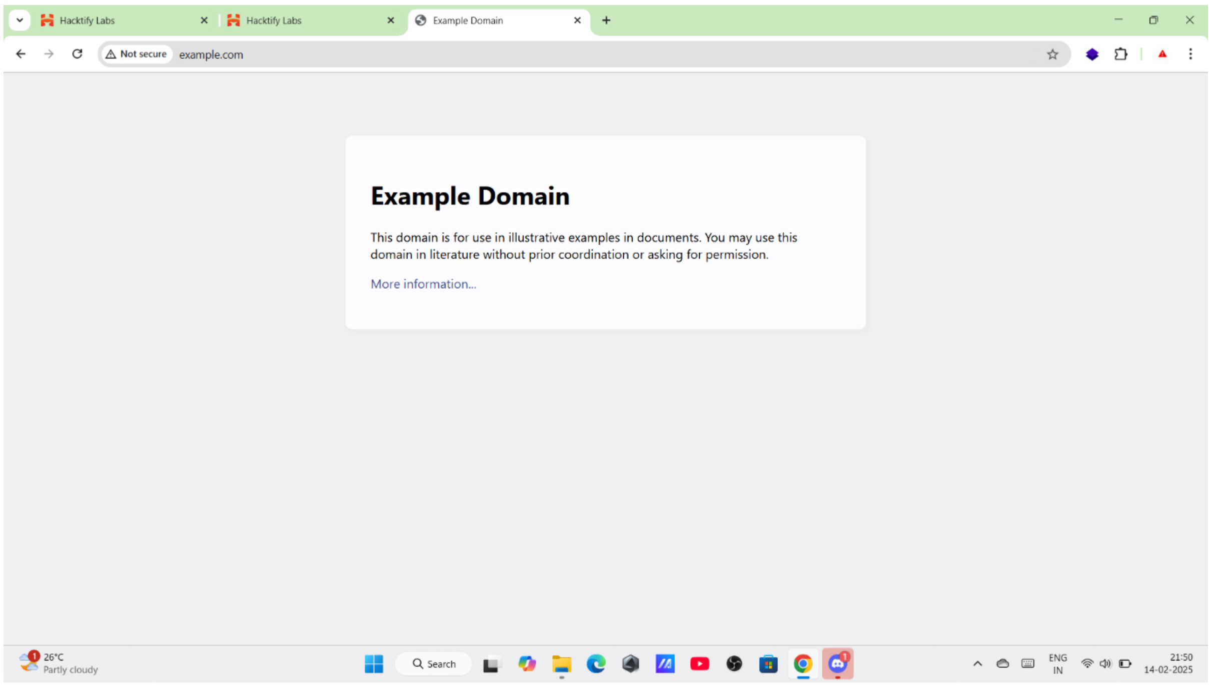
XSS

2.4. Alternatives Are Must!

Reference	**Risk Rating**
Alternates Are Must!	**Medium**
Tools Used	
Burp Suite, Web Browser	
Vulnerability Description	
The application allows **unsanitized user input** in an input field, making it vulnerable to **Reflected XSS**. By injecting a JavaScript payload, an attacker can force victims to be redirected to a malicious website.	
How It Was Discovered	
Manually tested by injecting the payload "><script>window.location='http://example.com/'</script> into an input field. Upon submission, the browser redirected to the specified URL.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_4/lab_4.php	

Consequences of not Fixing the Issue
Phishing attacks: Users can be redirected to fake login pages. **Data theft**: Sensitive information can be harvested through social engineering. **Session hijacking**: Attackers may steal cookies to impersonate users.
Suggested Countermeasures
Input validation: Reject special characters like < > " '. **Output encoding: Encode user input before rendering in the DOM.** **CSP implementation: Restrict inline JavaScript execution.**
References
OWASP XSS Prevention Cheat Sheet

Proof of Concept



Example Domain

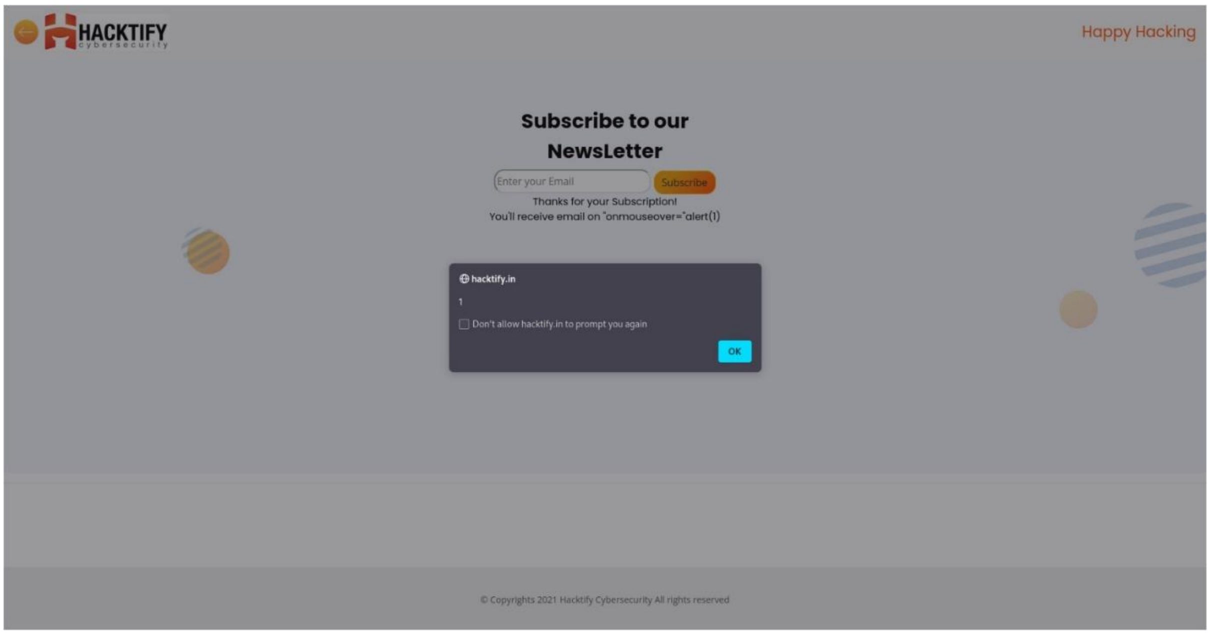
This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

2.5. Developer Hates Scripts!

Reference	Risk Rating
Developer Hates Scripts!	**Hard**
Tools Used	
Burp Suite	
Vulnerability Description	
The application does not properly sanitize input in the **search bar**, allowing attackers to inject **event-based XSS payloads**. This exploit triggers JavaScript execution when a user hovers over the injected element.	
How It Was Discovered	
Tested by injecting the **onmouseover** event into the search bar. When the result is displayed, hovering over it executed JavaScript.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_5/lab_5.php	
Consequences of not Fixing the Issue	
-Attackers can execute JavaScript in users' browsers. -Can be used for stealing cookies or session hijacking. -Users may be tricked into performing unintended actions.	
Suggested Countermeasures	
Input validation: Restrict special characters and JavaScript event handlers. **Output encoding:** Ensure user input is rendered as text, not executable code. **CSP headers:** Block inline script execution with Content-Security-Policy.	
References	
OWASP XSS Prevention Cheat Sheet	

Proof of Concept



Subscribe to our NewsLetter

Thanks for your Subscription!
You'll receive email on "onmouseover="alert(1)

hacktify.in
1
Don't allow hacktify.in to prompt you again

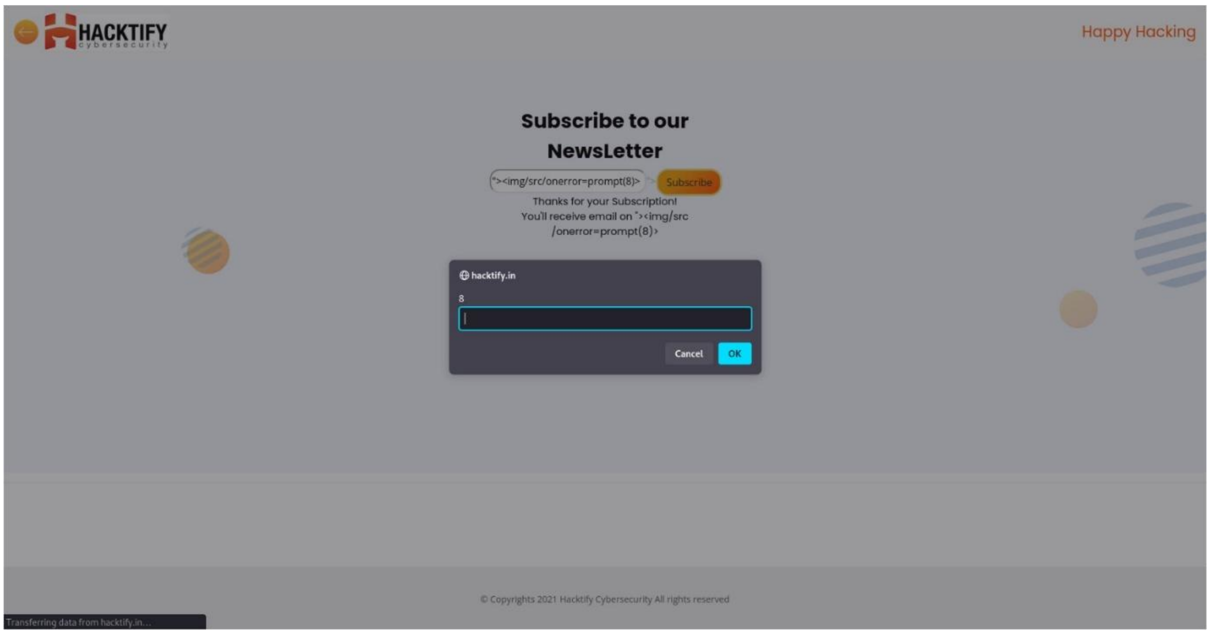
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.6. Change The Variation!

Reference	**Risk Rating**
Change The Variation!	**Hard**
Tools Used	
Burp Suite	
Vulnerability Description	
The application does not properly sanitize user input in the **input field**, allowing **image-based XSS injection**. The injected payload triggers JavaScript execution via the onerror event when the browser fails to load the image.	
How It Was Discovered	
Tested by injecting an image tag with an **onerror** event handler into the input field. Upon submission, the JavaScript executed as expected.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_6/lab_6.php	

Consequences of not Fixing the Issue
-Attackers can execute malicious JavaScript in the victim’s browser. -May lead to **session hijacking, data theft, or phishing attacks**. -Can be used to inject fake UI elements for social engineering.
Suggested Countermeasures
Input validation: Reject <script>, <img>, and other risky tags. **Output encoding:** Ensure user input is properly escaped when displayed. **Content Security Policy (CSP):** Restrict inline JavaScript execution.**.**
References
OWASP XSS Prevention Cheat Sheet

Proof of Concept


HACKTIFY cybersecurity
Happy Hacking
Subscribe to our NewsLetter
"><img/src/onerror=prompt(8)>
Subscribe
hacktify.in
8
Cancel
OK
© Copyrights 2021 Hacktify Cybersecurity All rights reserved
Transferring data from hacktify.in...

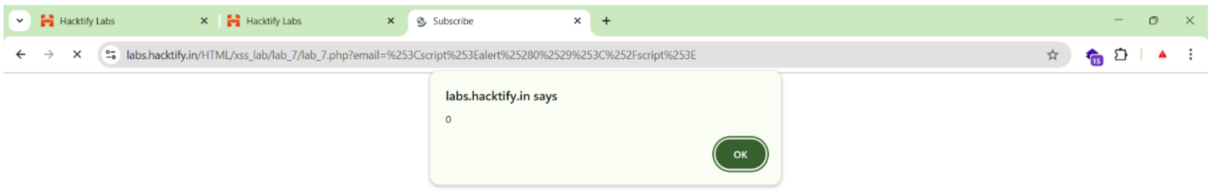
Thanks for your Subscription!
You'll receive email on "><img/src/onerror=prompt(8)>

2.7. Encoding Is The Key!

Reference	Risk Rating
Encoding Is The Key!	**Medium**

Tools Used
Burp Suite, Web Browser
Vulnerability Description
The application does not properly sanitize **URL-encoded input**, allowing an attacker to inject a **percent-encoded XSS payload**. When processed, the application decodes the input and executes the injected JavaScript.
How It Was Discovered
Tested by injecting a **URL-encoded XSS payload** (%3Cscript%3Ealert%280%29%3C%2Fscript%3E) into the input field. Upon submission, the browser executed the alert function.
Vulnerable URLs
https://hacktify.in/HTML/xss_lab/lab_7/lab_7.php
Consequences of not Fixing the Issue
-Attackers can **bypass standard filtering** by using encoded payloads. -**Stored or reflected XSS** may execute on the victim’s browser. -Can be used for **cookie theft, session hijacking, or phishing attacks**.
Suggested Countermeasures
Input validation: Restrict and sanitize user input before processing. **Output encoding:** Use functions like htmlspecialchars() to prevent script execution. **Implement CSP (Content Security Policy):** Block execution of inline scripts.
References
OWASP XSS Prevention Cheat Sheet

Proof of Concept



labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php?email=%253Cscript%253Ealert%25280%2529%253C%252Fscript%253E

labs.hacktify.in says

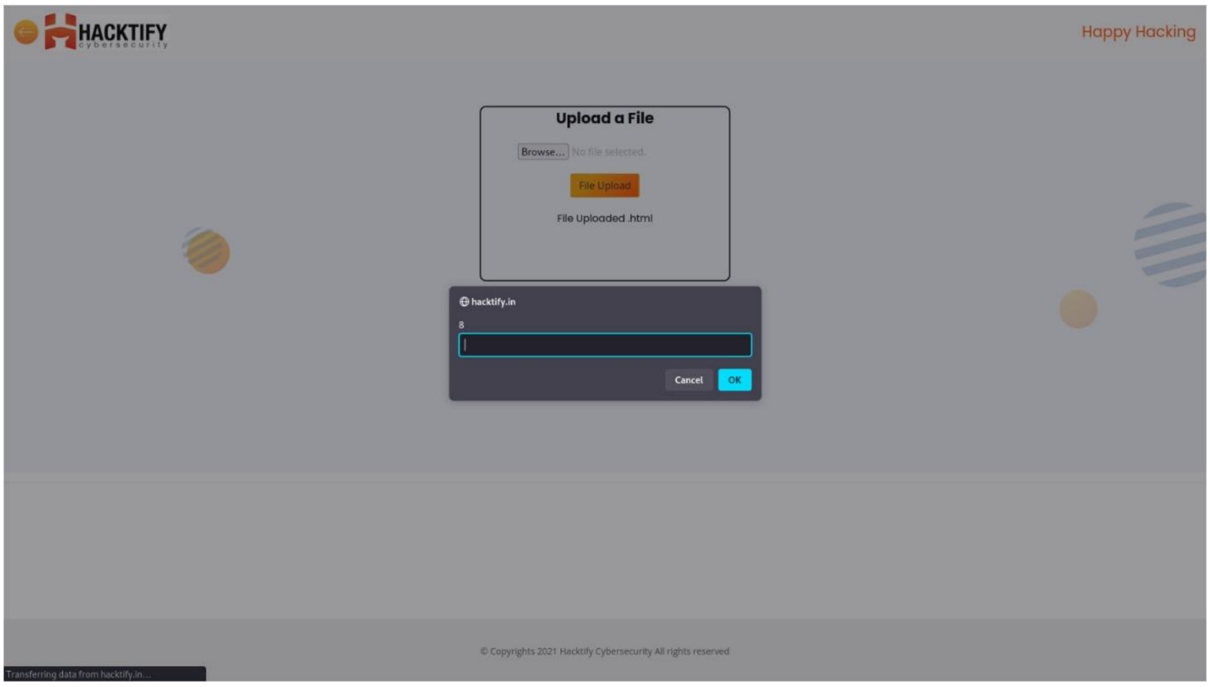
0

2.8. XSS With File Upload(File Name)

Reference	**Risk Rating**
XSS With File Upload (File Name)	**Low**
Tools Used	
Burp Suite	
Vulnerability Description	
The application allows unrestricted file uploads, permitting attackers to upload an **HTML file** containing malicious JavaScript. When a victim accesses the uploaded file, the JavaScript executes in their browser, leading to **Stored XSS**.	
How It Was Discovered	
By uploading an HTML file containing an **<img> tag with an onerror event**, which executes JavaScript when the browser fails to load the image.	
Vulnerable URLs	

https://hacktify.in/HTML/xss_lab/lab_8/lab_8.php
Consequences of not Fixing the Issue
Stored XSS: Every user accessing the uploaded file will execute the malicious script. **Session hijacking**: Attackers can steal cookies and impersonate users. **Phishing**: Users can be redirected to malicious sites.
Suggested Countermeasures
Restrict file types: Only allow safe extensions like .png, .jpg, and .pdf. **Validate file contents**: Use **MIME type validation** to ensure the uploaded file matches its expected type. **Deny execution**: Serve uploaded files with Content-Disposition: attachment headers to prevent execution.
References
OWASP Unrestricted File Upload Guide

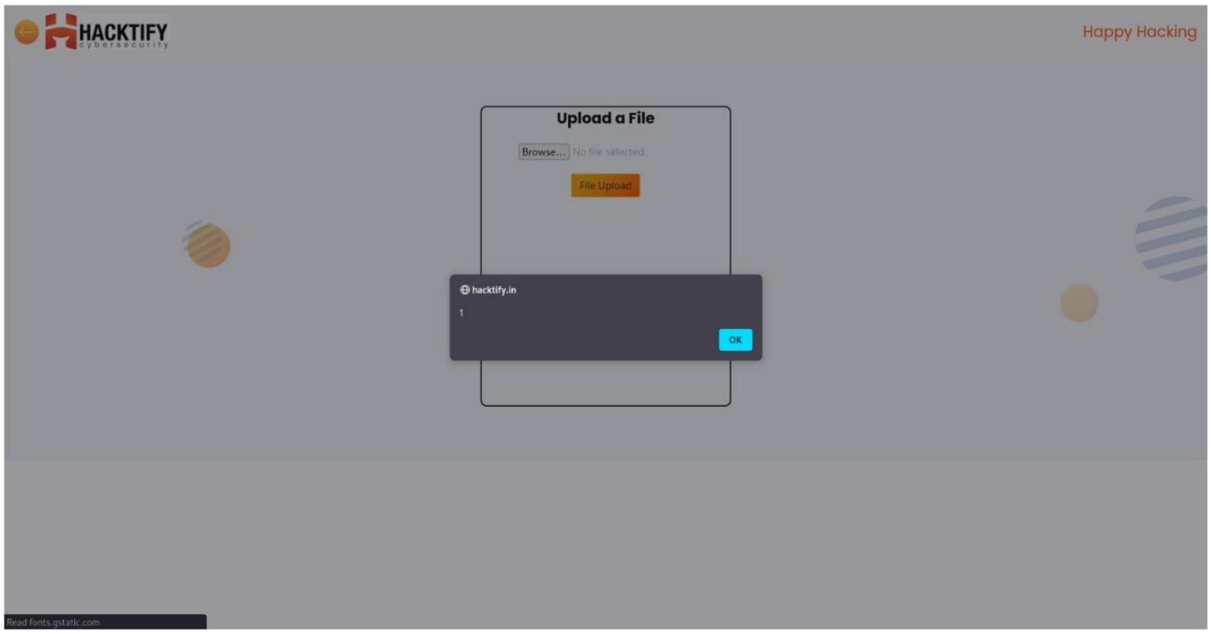
Proof of Concept


HACKTIFY
cybersecurity
Happy Hacking
Upload a File
Browse... No file selected
File Upload
File Uploaded .html
hacktify.in
8
Cancel
OK
© Copyrights 2021 Hacktify Cybersecurity All rights reserved
Transferring data from hacktify.in...

2.9. XSS With File Upload (File Content)

Reference	**Risk Rating**
XSS With File Upload (File Content)	**Medium**
Tools Used	
Burp Suite	
Vulnerability Description	
The application allows unrestricted **.txt file** uploads but fails to sanitize file content. If the server renders text files as **HTML instead of plain text**, an attacker can inject JavaScript into a .txt file, leading to **Stored XSS** when a victim views the file in a browser.	
How It Was Discovered	
By uploading a .txt file containing an XSS payload. When accessed in a browser, the JavaScript executed instead of being displayed as plain text.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_9/lab_9.php	
Consequences of not Fixing the Issue	
Stored XSS: The script executes every time a user opens the file. **Session hijacking**: Attackers can steal cookies and impersonate users. **Phishing**: Users may be tricked into executing malicious scripts.	
Suggested Countermeasures	
Restrict file types: Prevent .txt files from being served as HTML. **Set correct MIME types**: Ensure text files are served with Content-Type: text/plain instead of text/html. **Sanitize user uploads**: Escape <script> and other dangerous HTML tags in uploaded files.	
References	
OWASP Unrestricted File Upload Guide	

Proof of Concept

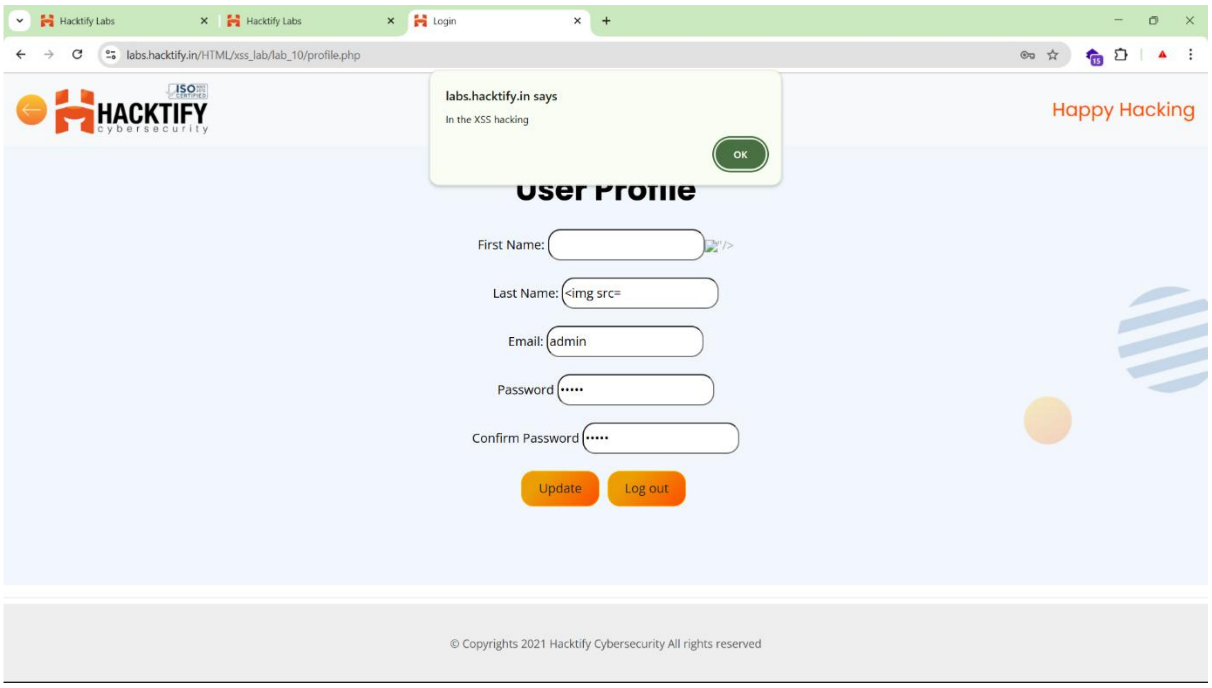


2.10. Stored Everywhere!

Reference	**Risk Rating**
Stored Everywhere!	**Low**
Tools Used	
Burp Suite	
Vulnerability Description	
The application **fails to sanitize user input** during registration, allowing **Stored XSS**. When a user registers with a malicious payload, the script gets stored in the database and executes when viewing the profile page.	
How It Was Discovered	
Tested by injecting a **JavaScript payload** in the registration form. Upon accessing the profile page, the script executed, confirming **Stored XSS**.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_10/lab_10.php	

Consequences of not Fixing the Issue
Persistent attack vector: The malicious script remains stored and executes every time the profile page is viewed. **Session hijacking**: Attackers can steal user cookies and impersonate victims. **Phishing & social engineering**: Users can be tricked into revealing sensitive information.
Suggested Countermeasures
Input validation: Reject < > " ' & characters in user input. **Output encoding**: Use htmlspecialchars() to encode stored data before rendering. **Content Security Policy (CSP)**: Restrict inline JavaScript execution.
References
OWASP XSS Prevention Cheat Sheet

Proof of Concept



labs.hacktify.in/HTML/xss_lab/lab_10/profile.php

labs.hacktify.in says

In the XSS hacking

OK

User Profile

First Name: '/>

Last Name: <img src=

Email: admin

Password •••••

Confirm Password •••••

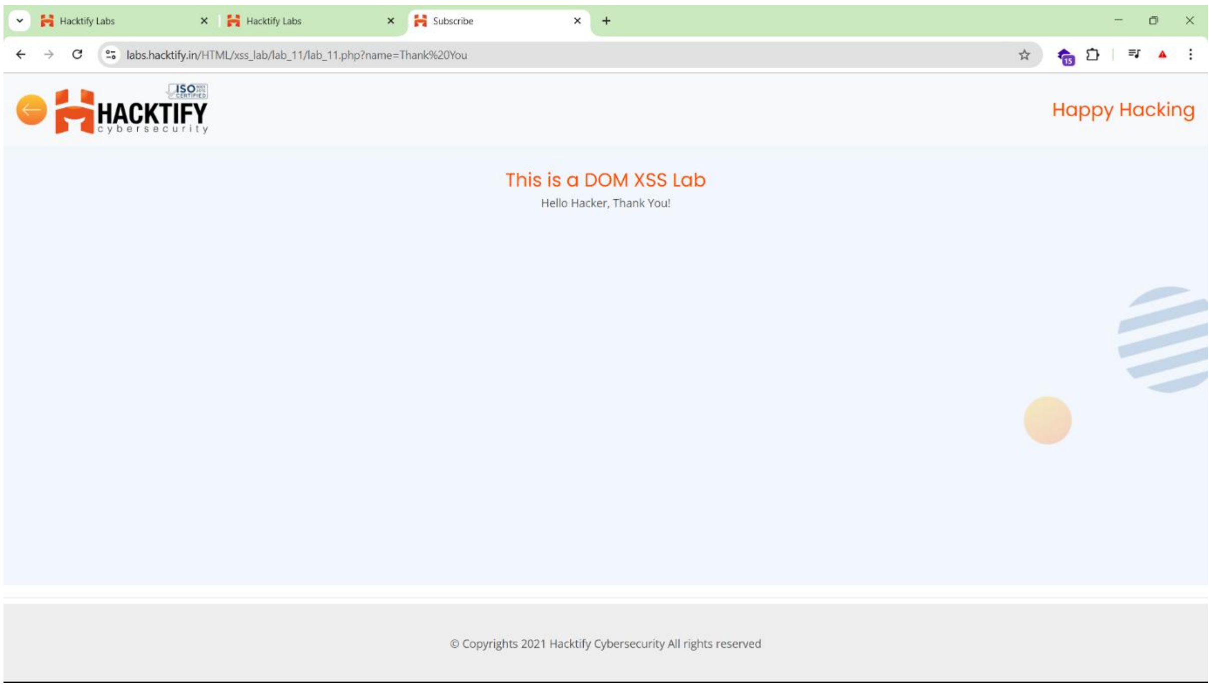
Update Log out

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.11. DOM’s Are Love!

Reference	**Risk Rating**
DOM’s Are Love!	**Hard**
Tools Used	
Burp Suite	
Vulnerability Description	
The **name parameter** in dom.js is **directly used in the DOM** without proper sanitization, making the application vulnerable to **DOM-Based XSS**. This allows attackers to inject and execute JavaScript within the victim's browser when they visit a crafted URL.	
How It Was Discovered	
By viewing the **page source**, it was observed that the name parameter is dynamically handled in dom.js. Injecting a **URL-encoded script payload** resulted in execution within the DOM.	
Vulnerable URLs	
https://hacktify.in/HTML/xss_lab/lab_11/lab_11.php?name=%3Cscript%3Ealert(1)%3C/script%3E	
Consequences of not Fixing the Issue	
Client-side script execution: Allows attackers to execute arbitrary JavaScript. **Session hijacking**: Attackers can steal cookies using document.cookie. **Phishing attacks**: Users can be redirected to malicious sites.	
Suggested Countermeasures	
Sanitize input: Use JavaScript sanitization libraries like DOMPurify. **Avoid innerHTML**: Use textContent instead to prevent script execution. **Implement CSP (Content Security Policy)**: Restrict inline JavaScript execution.	
References	
OWASP DOM-Based XSS Prevention Guide	

Proof of Concept



This is a DOM XSS Lab

Hello Hacker, Thank You!

© Copyrights 2021 Hacktify Cybersecurity All rights reserved