Penetration Testing Report

Full Name: MILIND PATEL
Program: HCPT
Date: 19 February 2025

Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week 1 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 1 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	Black Box Application HTML Injection, Cross Site Scripting

3. Summary

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

Total number of Sub-labs: 17 Sub-labs

High	Medium	Low
4	3	10

High	-	**4 Sub-lab with high difficulty level**
Medium	-	**3 Sub-labs with medium difficulty level**
Low	-	**10 Sub-labs with low difficulty level**

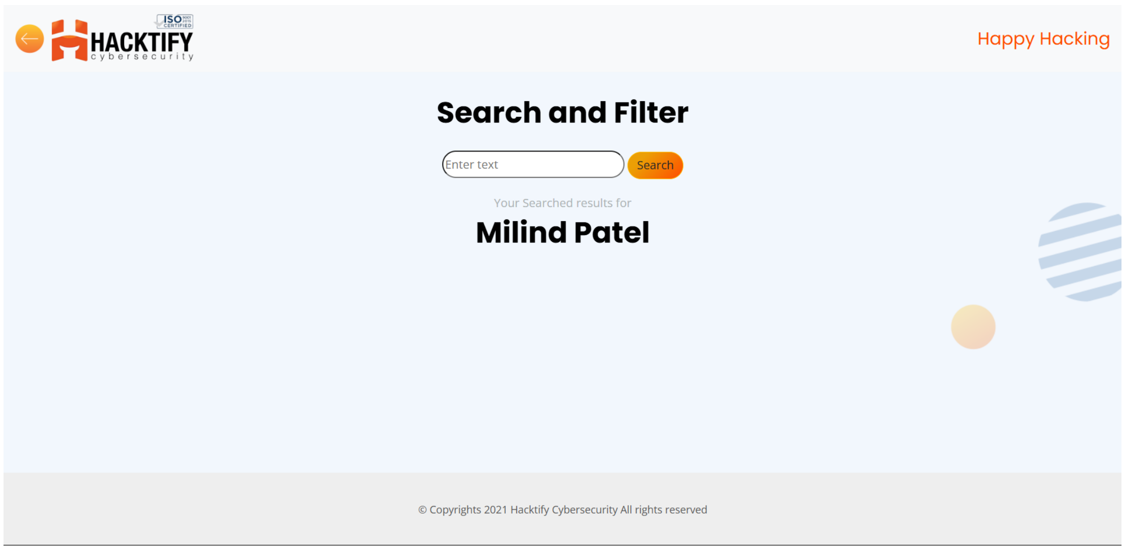
1. HTML Injection Labs

1.1. HTML's are easy!

Reference	**Risk Rating**
Sub-lab-1: HTML's are easy!	**Low**
Tools Used	
Browser "Inspect" and "View Page Source" are used to find vulnerability.	
Vulnerability Description	
HTML injection is a type of web security vulnerability that allows an attacker to inject malicious HTML code into a webpage. Attacker can exploit HTML code of website and steal the crucial data of users like login credentials or attacker can redirect user to its own malicious website.	
How It Was Discovered	
Automated Tools: View Page Source, Inspect	
Vulnerable URLs	
labs.hacktify.in/HTML/html_lab/lab_1/html_injection_1.php	
Consequences of not Fixing the Issue	
1. If the vulnerability is not fixed, it allows an attacker to put his own malicious code in website which leads to stealing of user's data.	
Suggested Countermeasures	
1. Use encoding to script. 2. Deploy Content Security Policy (CSP) to restrict script sources. 3. Regularly check and do vulnerability testing.	
References	
https://portswigger.net/web-security/sql-injection https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



Search and Filter

Your Searched results for

Milind Patel

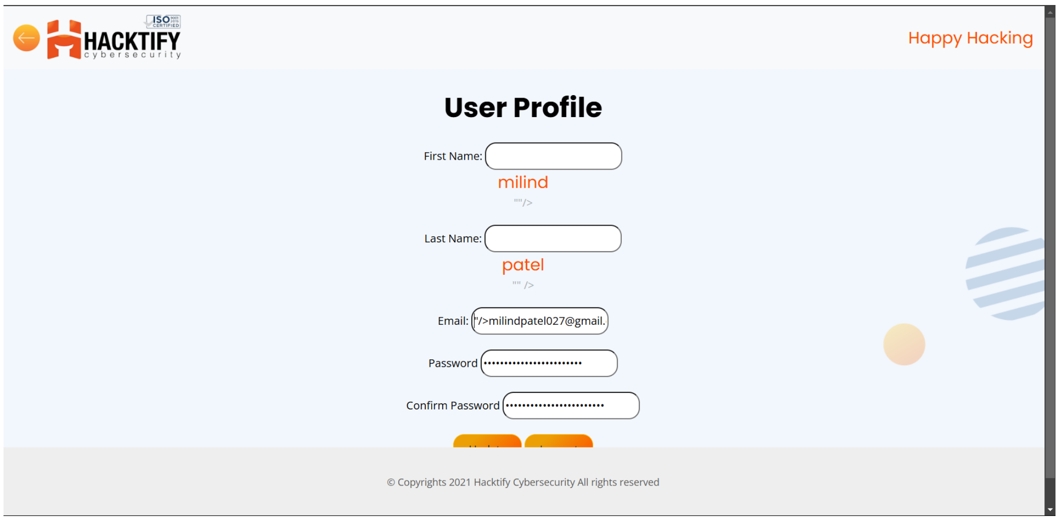
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.2. Let me store them!

Reference	Risk Rating
Sub-lab-2: Let me store them!	**Low**
Tools Used	
Scripts are used.	
Vulnerability Description	
On the login page, user data is stored. However, when we enter scripts into the text area, they get executed, which should not happen.	
How It Was Discovered	
Manual Analysis: Used scripts to identify vulnerability	
Vulnerable URLs	
labs.hacktify.in/HTML/html_lab/lab_2/profile.php	
Consequences of not Fixing the Issue	
1. Unauthorized user can get access to website by passing the security measures.	
Suggested Countermeasures	
1. Put some validations at text area. 2. Routine checks for vulnerability. 3. Validate page to do not accept script at user side.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/sql-injection	

Proof of Concept

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



User Profile

First Name:

milind

""/>

Last Name:

patel

"" />

Email: "/>milindpatel027@gmail.

Password ••••••••••••••••••••••••

Confirm Password ••••••••••••••••••••••••

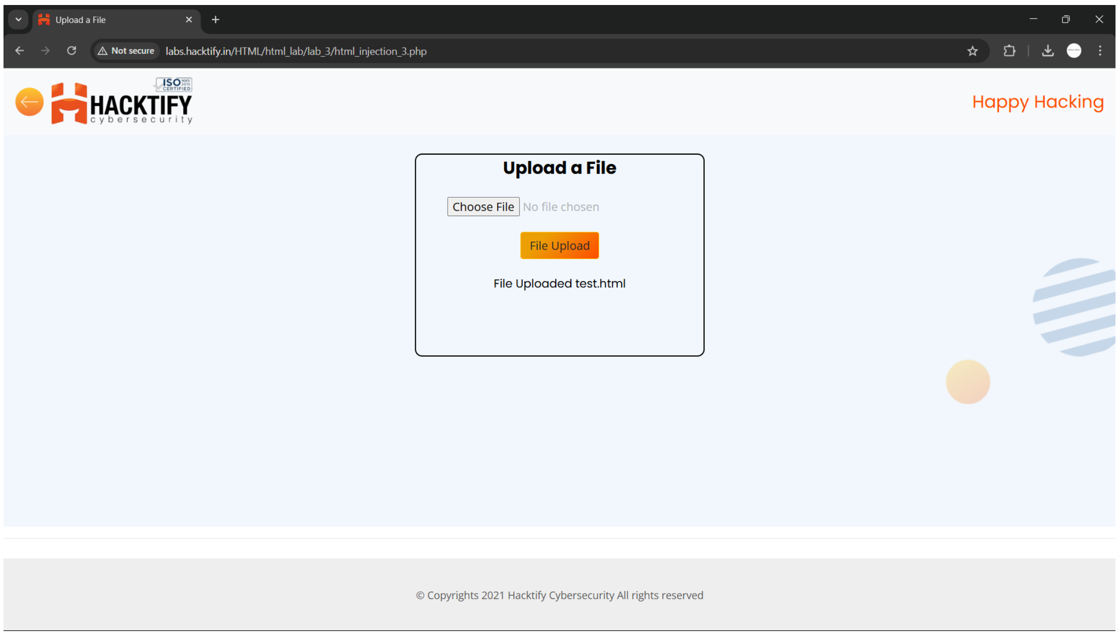
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.3. File names are also vulnerable!

Reference	Risk Rating
Sub-lab-3: File names are also vulnerable!	**Low**
Tools Used	
Browser ”inspect” tool, Scripts	
Vulnerability Description	
Attacker can change file name and manipulate files in server or change files with his own files.	
How It Was Discovered	
Automated Tools: Inspect tool of browser	
Vulnerable URLs	
labs.hacktify.in/HTML/html_lab/lab_3/html_injection_3.php	
Consequences of not Fixing the Issue	
1. XSS Attacks 2. Phishing Attacks 3. Attacker can change file extensions.	
Suggested Countermeasures	
1. Validate the server to prevent file manipulation. 2. Verify that no malicious files were uploaded based on their filenames. 3. Validate file extensions to allow only safe file types.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

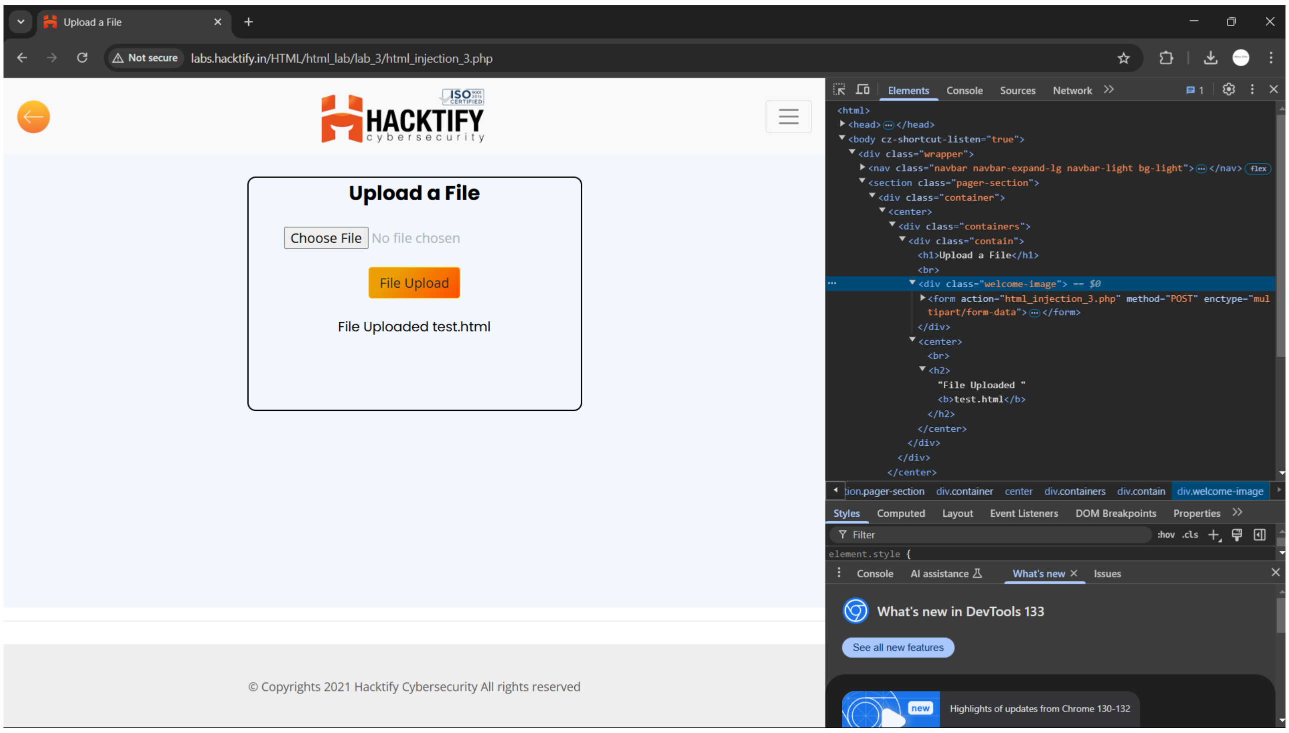
Proof of Concept

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



File Uploaded test.html

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



Upload a File

File Uploaded test.html

```
<html>
 <head> … </head>
 <body cz-shortcut-listen="true">
  <div class="wrapper">
   <nav class="navbar navbar-expand-lg navbar-light bg-light"> … </nav> flex
   <section class="pager-section">
    <div class="container">
     <center>
      <div class="containers">
       <div class="contain">
        <h1>Upload a File</h1>
        <br>
        <div class="welcome-image"> == $0
         <form action="html_injection_3.php" method="POST" enctype="mul
         tipart/form-data"> … </form>
        </div>
        <center>
         <br>
         <h2>
          "File Uploaded "
          <b>test.html</b>
         </h2>
        </center>
       </div>
      </div>
     </center>
```

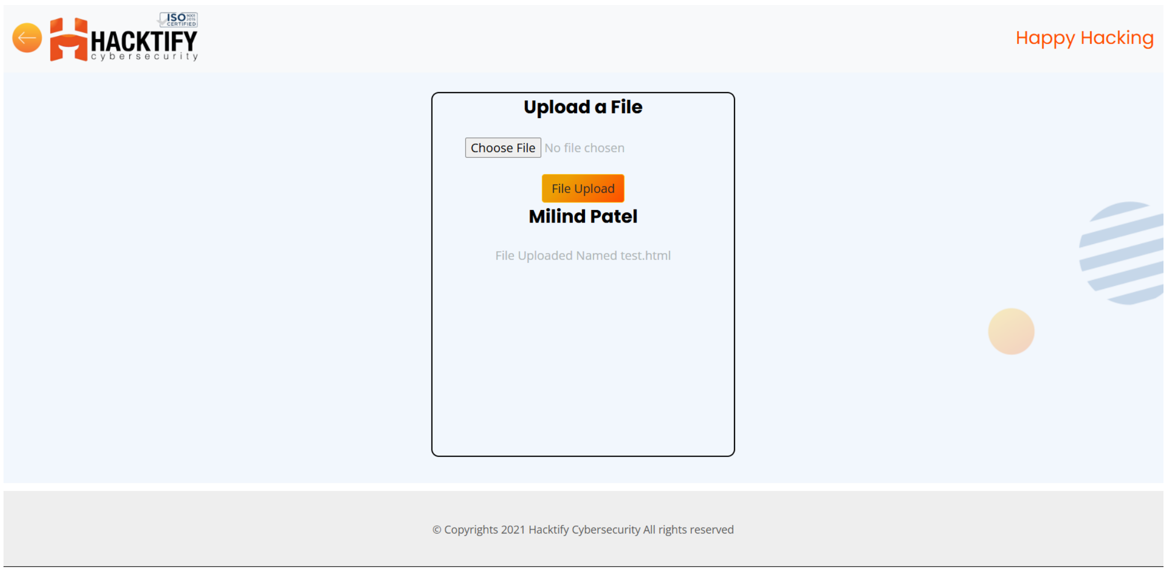
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.4. File Content and HTML Injection a perfect pair!

Reference	Risk Rating
Sub-lab-4: File Content and HTML Injection a perfect pair!	**Low**
Tools Used	
Burp Suite, Scripts	
Vulnerability Description	
This vulnerability allows users to view file contents without needing a proper application to open them. If left unpatched, it can lead to data breaches by exposing sensitive information. When a file is uploaded, its content is displayed directly or any embedded scripts may execute, posing a significant security risk.	
How It Was Discovered	
Automated Tools: Burp Suite, Script	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_4/html_injection_4.php	
Consequences of not Fixing the Issue	
1. Data breach as attacker can see content of files simply by uploading files. 2. Phishing attacks can be done.	
Suggested Countermeasures	
1. Sanitize the files uploaded. 2. Use Content Security Policy (CSP).	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

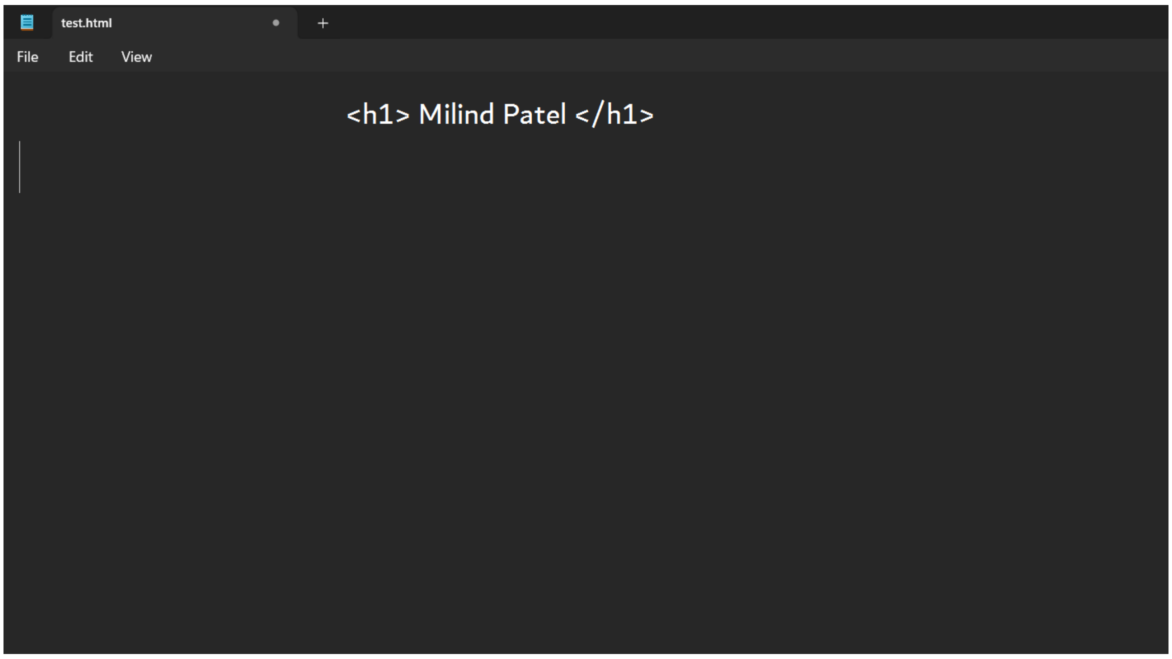
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.



Milind Patel

File Uploaded Named test.html

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



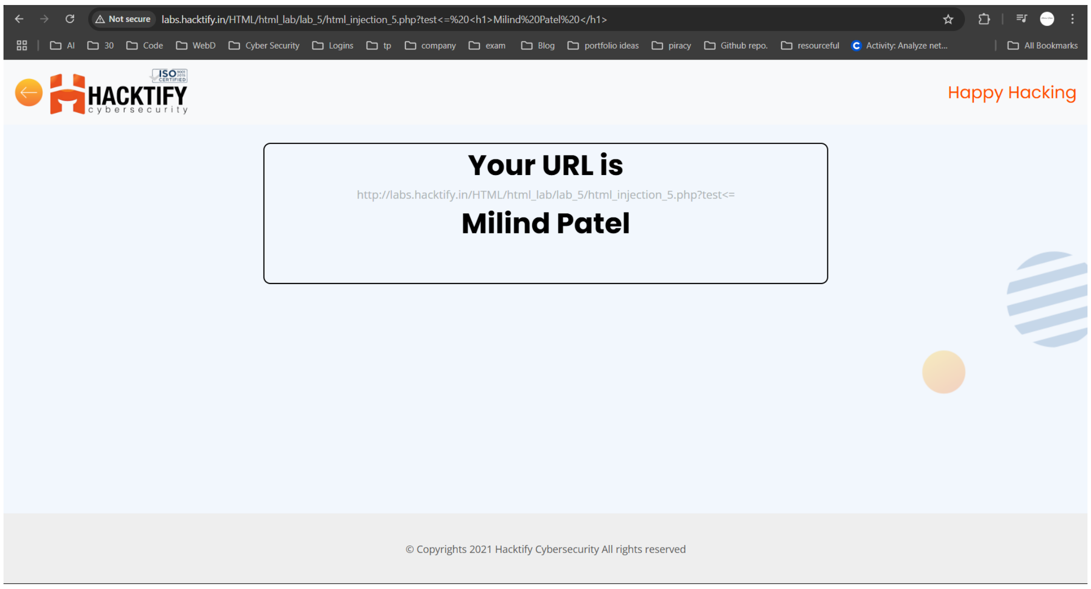
```
<h1> Milind Patel </h1>
```

1.5. Injecting HTML using URL

Reference	Risk Rating
Sub-lab-5: Injecting HTML using URL	**Medium**
Tools Used	
HTML, JavaScripts Scripts are used	
Vulnerability Description	
It occurs when user or attacker put code snippet or scripts directly to URL and it runs into webpage. We can put simple HTML line to URL and it shows on webpage.	
How It Was Discovered	
Manual Analysis: Putting or Changing html, javscripts scripts on URL	
Vulnerable URLs	
http://labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php?Sample=<h1>Hello%20Community</h1>	
Consequences of not Fixing the Issue	
1. False Information on Webpage: An attacker could inject false or misleading information onto the webpage, potentially damaging the credibility of the site and misleading users. 2. Illegitimate Login Page: An attacker might create a fake login page to steal user credentials, leading to unauthorized access and potential data breaches.	
Suggested Countermeasures	
1. Encode the URL. 2. Implement CSP. 3. Use HTTPS for more secure	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



Not secure labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php?test<=%20<h1>Milind%20Patel%20</h1>

Your URL is

http://labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php?test<=

Milind Patel

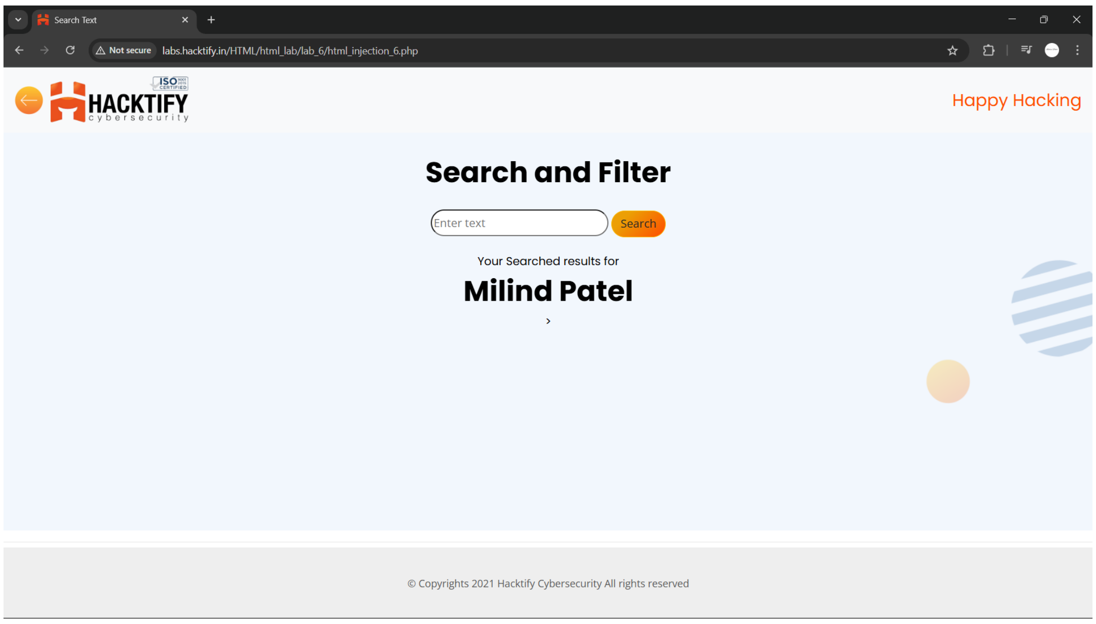
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.6. Encode It!

Reference	Risk Rating
Sub-lab-6: Encode It!	**High**
Tools Used	
URL Encoder	
Vulnerability Description	
This vulnerability shows that URLs can be a threat too as the attacker can use malicious codes embedded as URL and inject it into the webpages.	
How It Was Discovered	
Automated Tools: URL Encoder	
Vulnerable URLs	
http%3A%2F%2Flabs.hacktify.in%2FHTML%2Fhtml_lab%2Flab_5%2Fhtml_injection_5.php%3FSample%3D%3Ch1%3EHello%2520Community%3C%2Fh1%3E%0A%0A	
Consequences of not Fixing the Issue	
1. Attacker can tricks to login in fake webpage and steal login credentials. 2. Can redirect to fake webpage.	
Suggested Countermeasures	
1. Encode the URL. 2. Encode the content of File. 3. Use HTTPS rather than HTTP.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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


Search Text
Not secure
labs.hacktify.in/HTML/html_lab/lab_6/html_injection_6.php
HACKTIFY
Happy Hacking
Search and Filter
Enter text
Search
Your Searched results for
Milind Patel
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

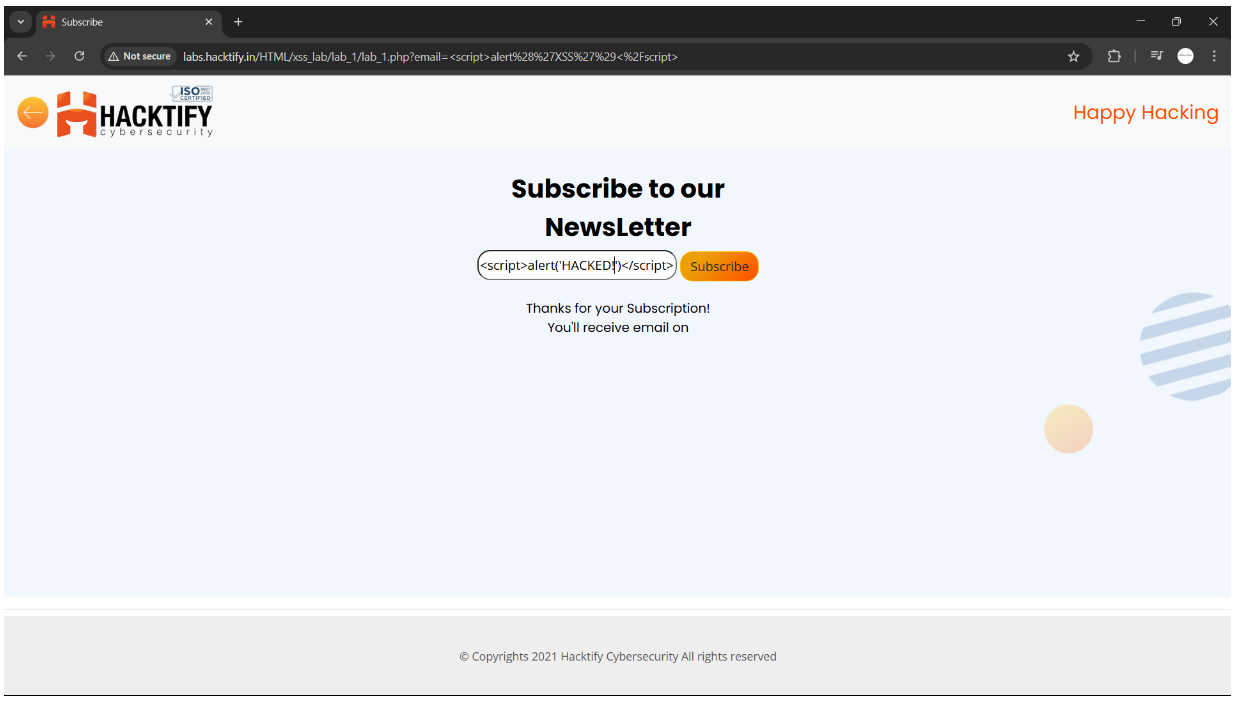
2. Cross-Site Scripting Labs

2.1. Let’s do it!

Reference	Risk Rating
Sub-lab-1: Let’s do it!	**Low**
Tools Used	
Burp Suite, Acunetix, JavaScript	
Vulnerability Description	
XSS occurs when attacker injects an malicious JavaScript code to webpage or web application, and it runs on target browser.	
How It Was Discovered	
Manual Analysis: Putting JavaScript	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=<script>alert%28”Test+XSS”%29<%2Fscript>	
Consequences of not Fixing the Issue	
1. Using JavaScript directly to webpage can helps to steal cookie containing user data. 2. Can inject fake HTML webpage.	
Suggested Countermeasures	
1. Use secure headers. 2. Validate the input areas. 3. Input CSP. 4. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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

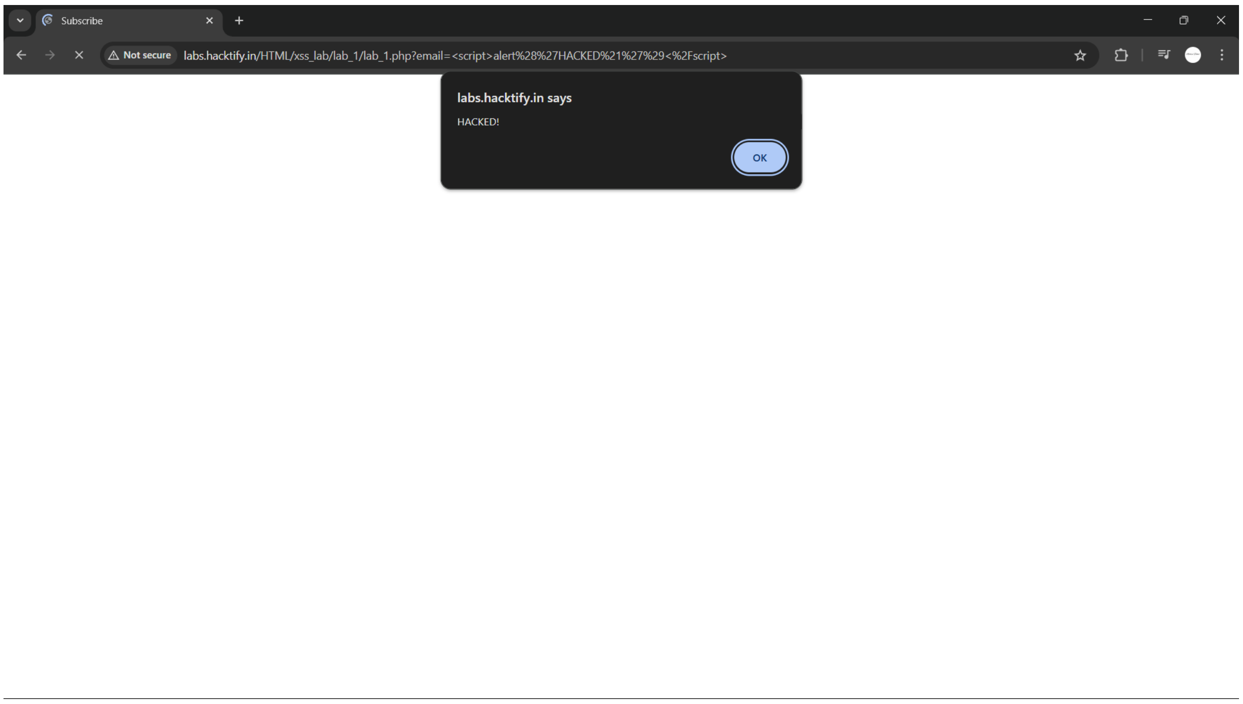

labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=<script>alert%28%27XSS%27%29<%2Fscript>

Subscribe to our NewsLetter

<script>alert('HACKED!')</script> Subscribe

Thanks for your Subscription!
You'll receive email on

© Copyrights 2021 Hacktify Cybersecurity All rights reserved


labs.hacktify.in/HTML/xss_lab/lab_1/lab_1.php?email=<script>alert%28%27HACKED%21%27%29<%2Fscript>

labs.hacktify.in says

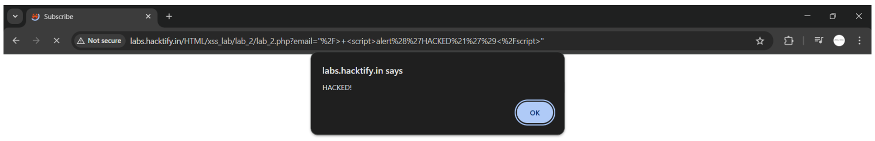
HACKED!

2.2. Balancing is important in life!

Reference	Risk Rating
Sub-lab-2: Balancing is important in life!	**Low**
Tools Used	
Burp Suite, Acunetix, JavaScript	
Vulnerability Description	
XSS occurs when attacker injects an malicious JavaScript code to webpage or web application, and it runs on target browser. We can use the concept of Social Engineering in attacking to trick an user or target using XSS.	
How It Was Discovered	
Manual Analysis: JavaScript	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email=%2F><script>alert%28"Compromised"%29<%2Fscript>	
Consequences of not Fixing the Issue	
1. Cookie Theft: Attackers can use JavaScript to steal cookies that contain user data, leading to unauthorized access to user accounts and sensitive information. 2. Fake HTML Injection: Attackers can inject fraudulent HTML pages to capture login credentials, such as usernames and passwords, compromising user security and privacy.	
Suggested Countermeasures	
1. Use secure headers. 2. Validate the input areas. 3. Input CSP. 4. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



labs.hacktify.in/HTML/xss_lab/lab_2/lab_2.php?email="%2F>+<script>alert%28%27HACKED%21%27%29<%2Fscript>"

labs.hacktify.in says

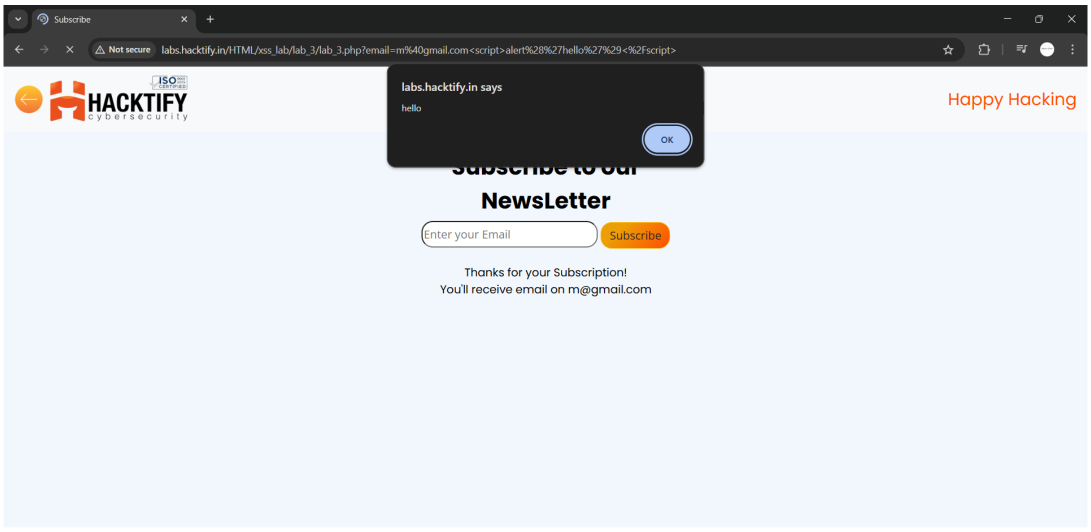
HACKED!

2.3. XSS is everywhere!

Reference	Risk Rating
Sub-lab-3: XSS is everywhere!	**Low**
Tools Used	
Burp Suite, Acunetix, JavaScript	
Vulnerability Description	
XSS occurs when attacker injects an malicious JavaScript code to webpage or web application, and it runs on target browser. We can use the concept of Social Engineering in attacking to trick an user or target using XSS. We can also implement this vulnerability through URL.	
How It Was Discovered	
Manual Analysis: By putting JavaScript into URL	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_3/lab_3.php?email=test%40<script>alert%28%27XSS%27%29<%2Fscript>	
Consequences of not Fixing the Issue	
1. Using JavaScript directly to webpage can helps to steal cookie containing user data. 2. Can inject fake HTML webpage to steal login ID's and Password.	
Suggested Countermeasures	
1. Using HTTPS to secure URL. 2. Validate the input areas. 3. Input CSP. 4. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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


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

labs.hacktify.in says

hello

Subscribe to our NewsLetter

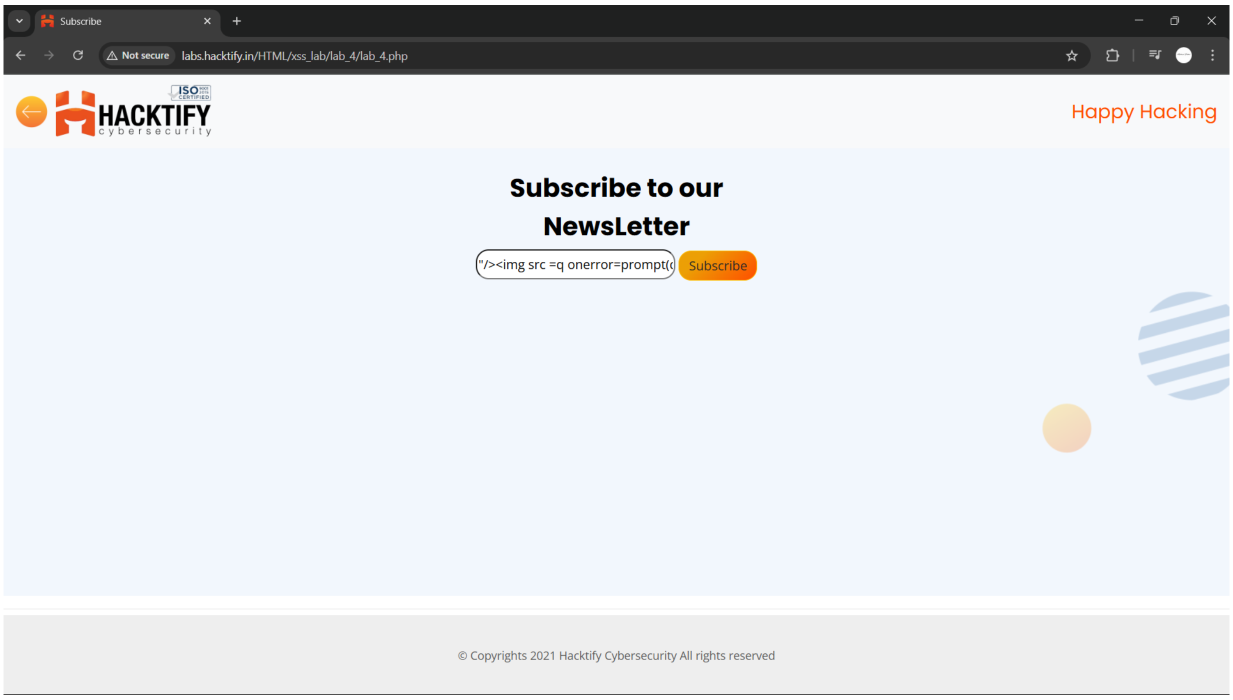
Thanks for your Subscription!
You'll receive email on m@gmail.com

2.4. Alternatives are must!

Reference	**Risk Rating**
Sub-lab-4: Alternatives are must!	**Medium**
Tools Used	
Burp Suite, Acunetix, JavaScript	
Vulnerability Description	
XSS occurs when attacker injects an malicious JavaScript code to webpage or web application, and it runs on target browser. We can use different Payloads to attack this vulnerability. We can also implement this vulnerability through URL.	
How It Was Discovered	
Manual Analysis: JavaScript/ Different Payloads	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php?email=%22%3E%3Ca+href%3D%22javascript%3Aalert%28%27Hacked %27%29%22%3EClick+Me%3C%2Fa%3E	
Consequences of not Fixing the Issue	
1. Using JavaScript directly to webpage can helps to steal cookie containing user data. 2. Can inject fake HTML webpage to steal login ID's and Password. 3. Using different Payloads can do variation in attack or target information which we required.	
Suggested Countermeasures	
1. Using HTTPS to secure URL. 2. Validate the input areas. 3. Input CSP. 4. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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

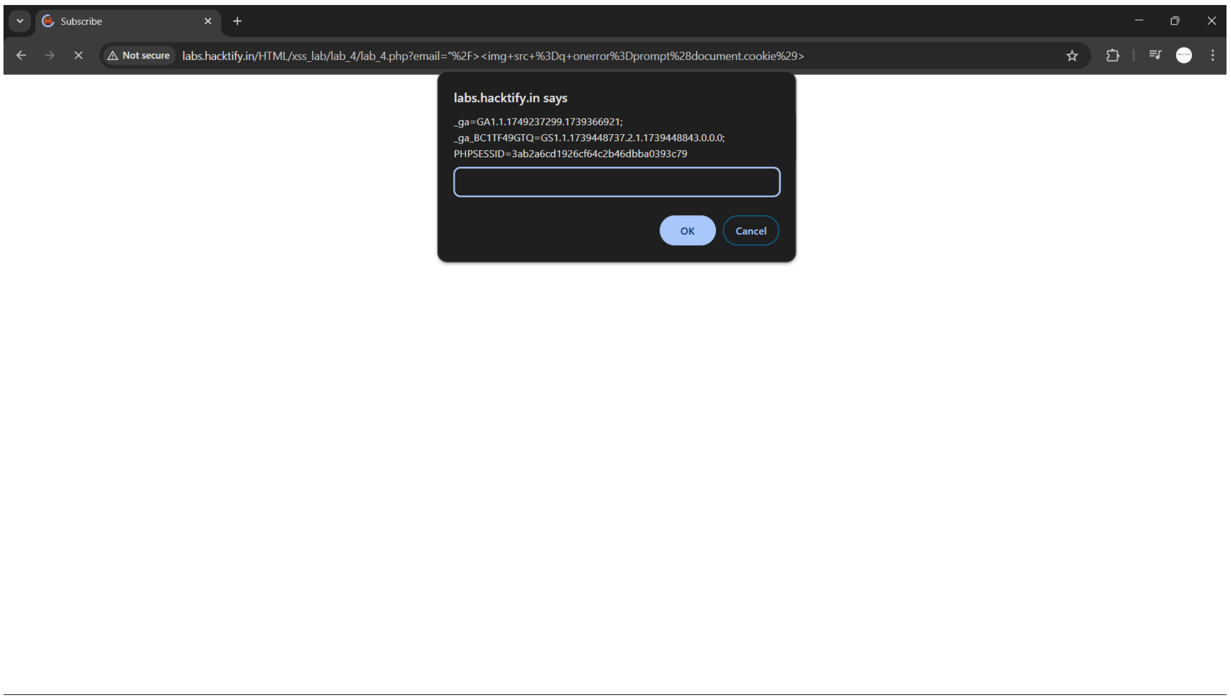


```
labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php
```

Subscribe to our NewsLetter

```
"/><img src =q onerror=prompt(d
```

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



```
labs.hacktify.in/HTML/xss_lab/lab_4/lab_4.php?email="%2F><img+src+%3Dq+onerror%3Dprompt%28document.cookie%29>
```

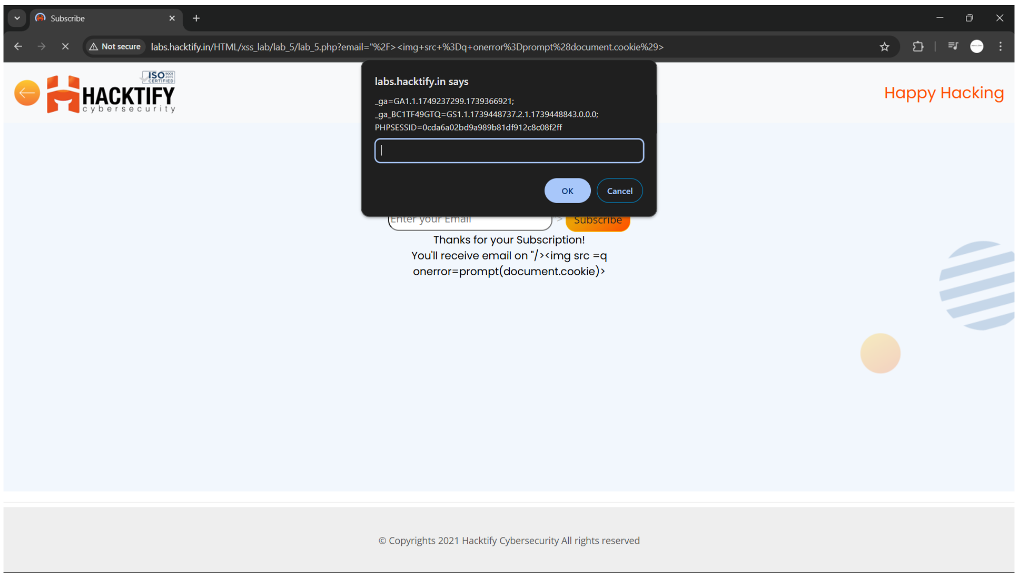
```
labs.hacktify.in says
_ga=GA1.1.1749237299.1739366921;
_ga_BC1TF49GTQ=GS1.1.1739448737.2.1.1739448843.0.0.0;
PHPSESSID=3ab2a6cd1926cf64c2b46dbba0393c79
```

2.5. Developer hates scripts!

Reference	**Risk Rating**
Sub-lab-5: Developer hates scripts!	**High**
Tools Used	
JavaScript	
Vulnerability Description	
Different Payloads can be used to violate this vulnerability. Implementation of this vulnerability can be done through URLs . Several times Developer puts validations on webpage so to by-pass that we use Payloads which are small and dangers to webpage or web application.	
How It Was Discovered	
Manual Analysis: JavaScript	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_5/lab_5.php?email=”%2F><img+src%3Dx+onerror%3Dalert%28%27XSS%27%29>	
Consequences of not Fixing the Issue	
1. Using JavaScript directly to webpage can helps to steal cookie containing user data. 2. Can inject fake HTML webpage to steal login ID’s and Password. 3. Using different Payloads can do variation in attack or target information which we required.	
Suggested Countermeasures	
1. Validate the input areas. 2. Input CSP. 3. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



labs.hacktify.in/HTML/xss_lab/lab_5/lab_5.php?email="%2F><img+src+%3Dq+onerror%3Dprompt%28document.cookie%29>

labs.hacktify.in says

```
_ga=GA1.1.1749237299.1739366921;
_ga_BC1TF49GTQ=GS1.1.1739448737.2.1.1739448843.0.0.0;
PHPSESSID=0cda6a02bd9a989b81df912c8c08f2ff
```

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

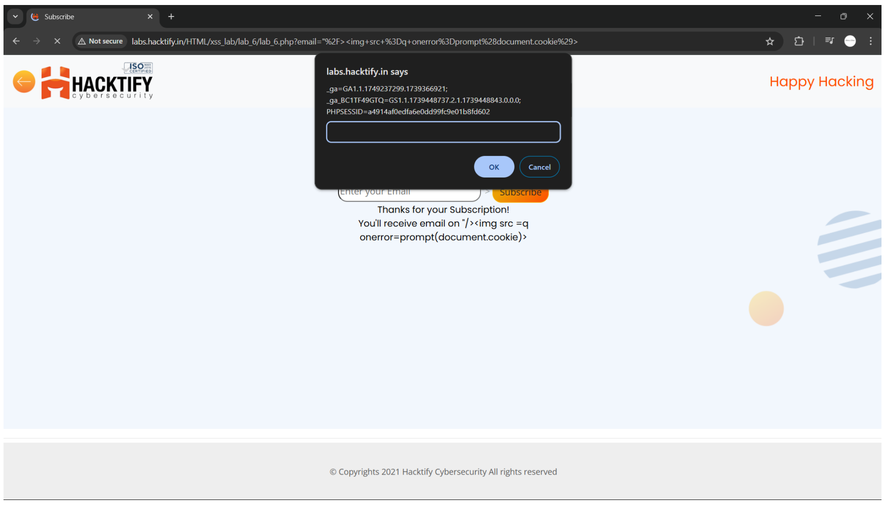
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.6. Change the Variation!

Reference	Risk Rating
Sub-lab-6: Change the Variation!	**High**
Tools Used	
JavaScript, Acunetix	
Vulnerability Description	
Different Payloads can be used to violate this vulnerability. Implementation of this vulnerability can be done through URLs . Several times Developer puts validations on webpage so to by-pass that we use Payloads which are small and dangers to webpage or web application.	
How It Was Discovered	
Automated Tools: Burp Suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_6/lab_6.php?email=test%22%2F%3E%3Cimg+src%3Dq+onerror%3Dprompt%28document.cookie%29%3E%40example.com	
Consequences of not Fixing the Issue	
1. Using JavaScript directly to webpage can helps to steal cookie containing user data. 2. Can inject fake HTML webpage to steal login ID's and Password. 3. Using different Payloads can do variation in attack or target information which we required.	
Suggested Countermeasures	
1. Validate the input areas. 2. Input CSP. 3. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



labs.hacktify.in/HTML/xss_lab/lab_6/lab_6.php?email="%2F><img+src+%3Dq+onerror%3Dprompt%28document.cookie%29>

labs.hacktify.in says

_ga=GA1.1.1749237299.1739366921;
_ga_BC1TF49GTQ=GS1.1.1739448737.2.1.1739448843.0.0.0;
PHPSESSID=a4914af0edfa6e0dd99fc9e01b8fd602

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

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.7. Encoding is the key?

Reference	Risk Rating
Sub-lab-7: Encoding is the key?	**Medium**
Tools Used	
JavaScript, Acunetix, URL Encoder.	
Vulnerability Description	
XSS occurs when attacker injects an malicious JavaScript code to webpage or web application, and it runs on target browser. We can use different Payloads to attack this vulnerability. We can also implement this vulnerability through URL.	
How It Was Discovered	
Manual Analysis: JavaScript	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_7/lab_7.php?email=%253Cscript%253Ealert%2528document.cookie%2529%253C%252Fscript%253E	
Consequences of not Fixing the Issue	
1. Using JavaScript directly to webpage can helps to steal cookie containing user data. 2. Can inject fake HTML webpage to steal login ID's and Password. 3. Using different Payloads can do variation in attack or target information which we required.	
Suggested Countermeasures	
1. Validate the input areas. 2. Input CSP. 3. Encoding.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

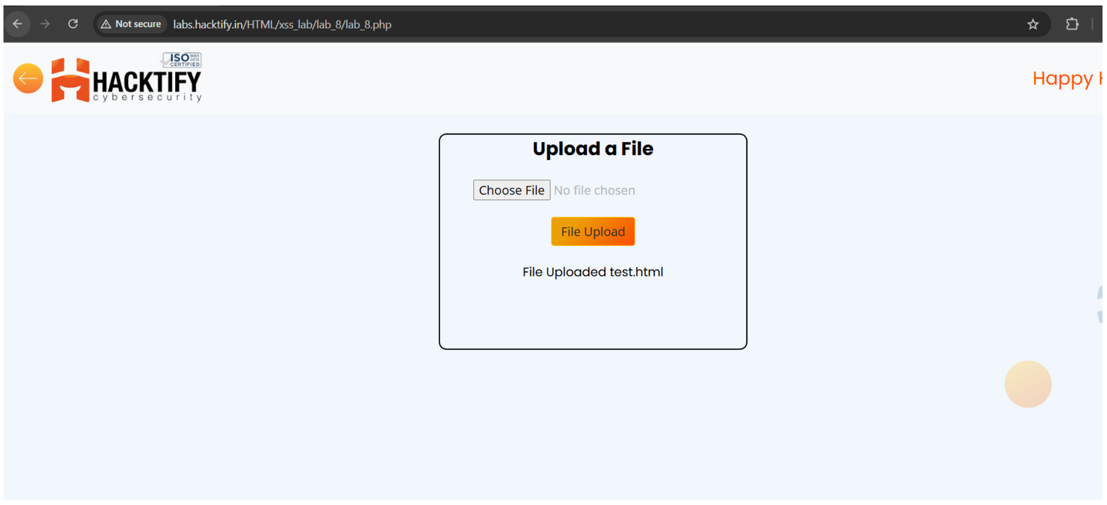
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

2.8. XSS with File Upload(lab8.html)

Reference	Risk Rating
Sub-lab-8: XSS with File Upload	**Low**
Tools Used	
File containing JavaScript codes, Burp Suite	
Vulnerability Description	
It occurs when an attacker uploads an malicious JavaScript code file and when user tries to opens or preview it in the browser the code in it executed and leads to steal data or downloading malicious software into user's computer. Using Burp Suite we can see file interacting with server.	
How It Was Discovered	
Manual Analysis: By checking if the file is directly accessible after uploading.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php	
Consequences of not Fixing the Issue	
1. Can automatically install malicious software without user's permission. 2. Can run malicious code as soon as user opens file. 3. Can corrupt server.	
Suggested Countermeasures	
1. Validate file type to upload. 2. Implementing CSP. 3. Not saving file directly to server. 4. Block direct execution of uploaded files.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

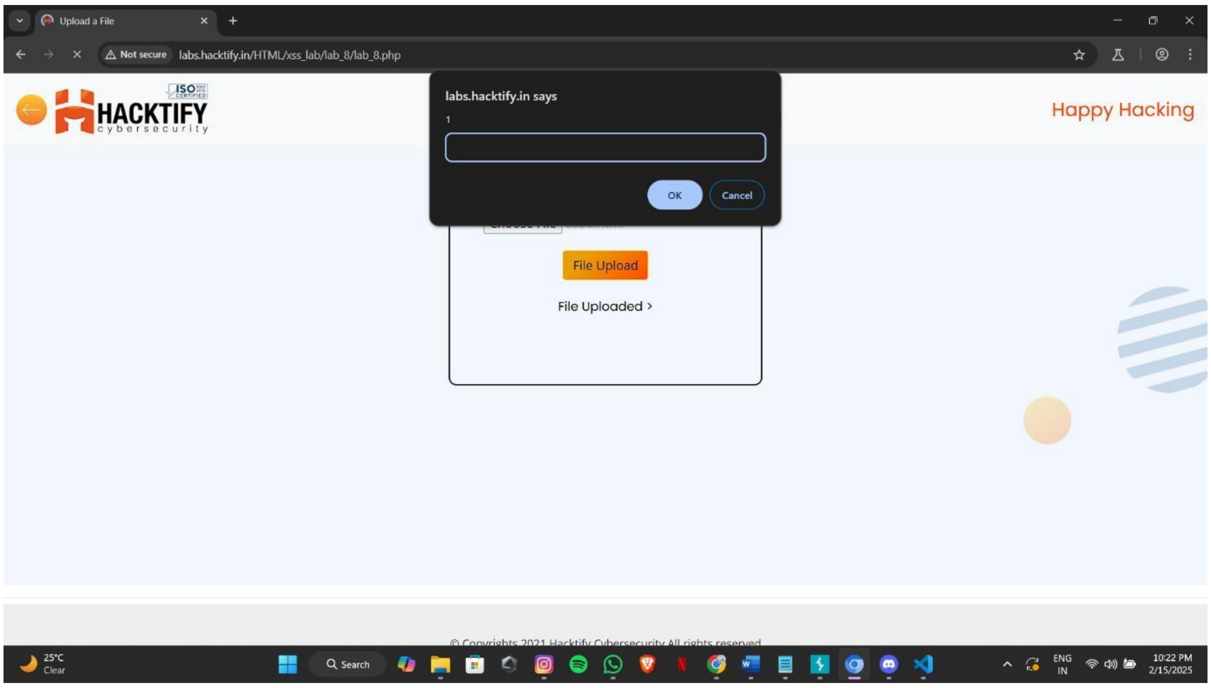
Proof of Concept

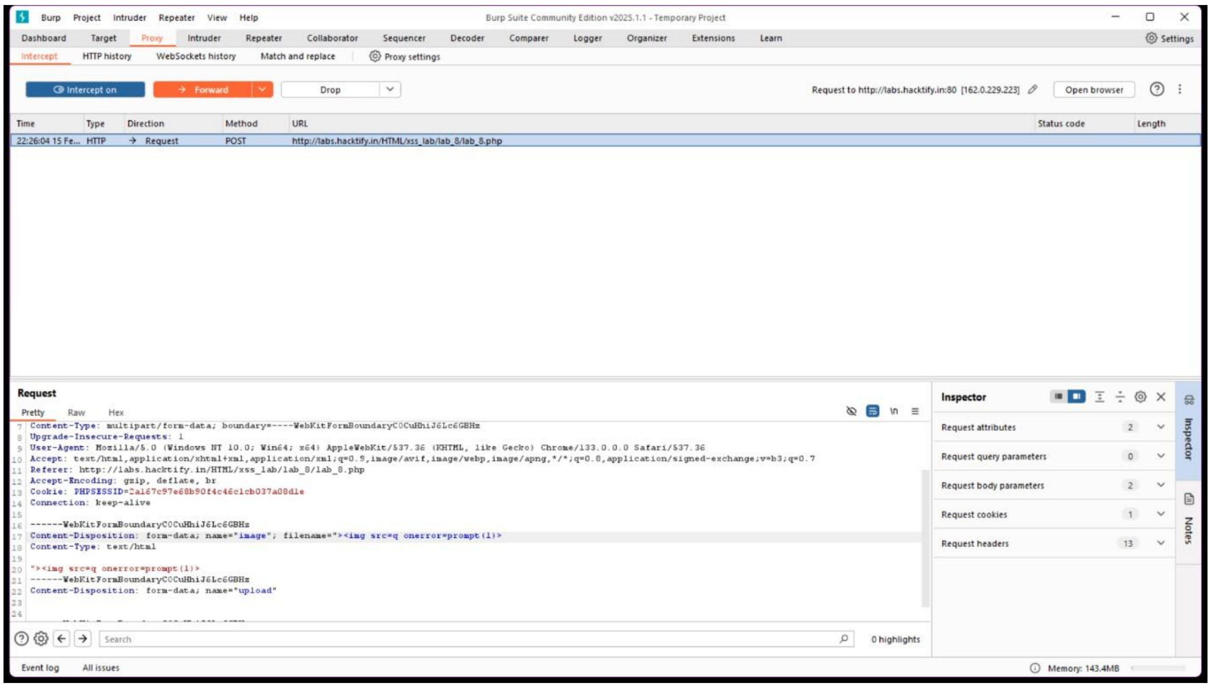
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



Upload a File

File Uploaded test.html


Upload a File
Not secure labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php
labs.hacktify.in says
1
OK
Cancel
HACKTIFY cybersecurity
Happy Hacking
File Upload
File Uploaded >
© Copyrights 2021 Hacktify Cybersecurity All rights reserved


Burp Suite Community Edition v2025.1.1 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://labs.hacktify.in:80 [162.0.229.223]
Open browser
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 2
Request cookies 1
Request headers 13

Time	Type	Direction	Method	URL	Status code	Length
22:26:04 15 Fe...	HTTP	→ Request	POST	http://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php		

Request

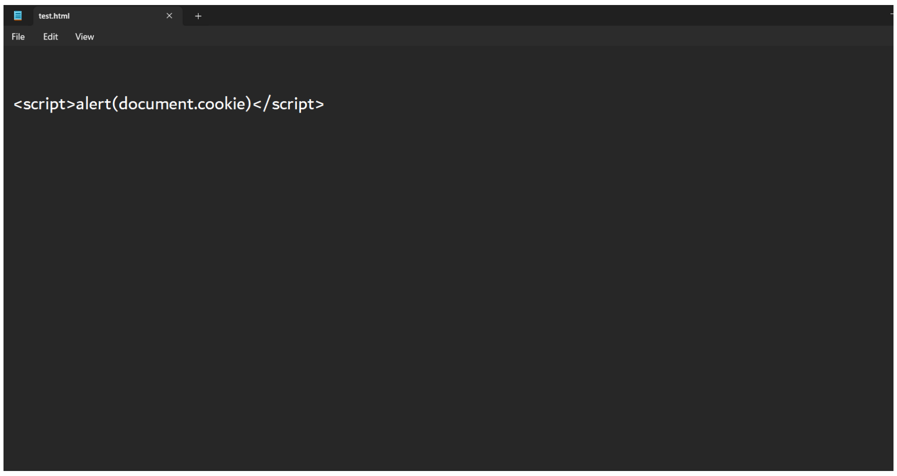
```
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryC0CuHhiJ6Lc6GBHz
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://labs.hacktify.in/HTML/xss_lab/lab_8/lab_8.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=2a167c97e68b90f4c46c1cb037a08d1e
Connection: keep-alive

------WebKitFormBoundaryC0CuHhiJ6Lc6GBHz
Content-Disposition: form-data; name="image"; filename="><img src=q onerror=prompt(1)>
Content-Type: text/html

"><img src=q onerror=prompt(1)>
------WebKitFormBoundaryC0CuHhiJ6Lc6GBHz
Content-Disposition: form-data; name="upload"
```

2.9. XSS with File Content(<script>alert(document.cookie)</script>)

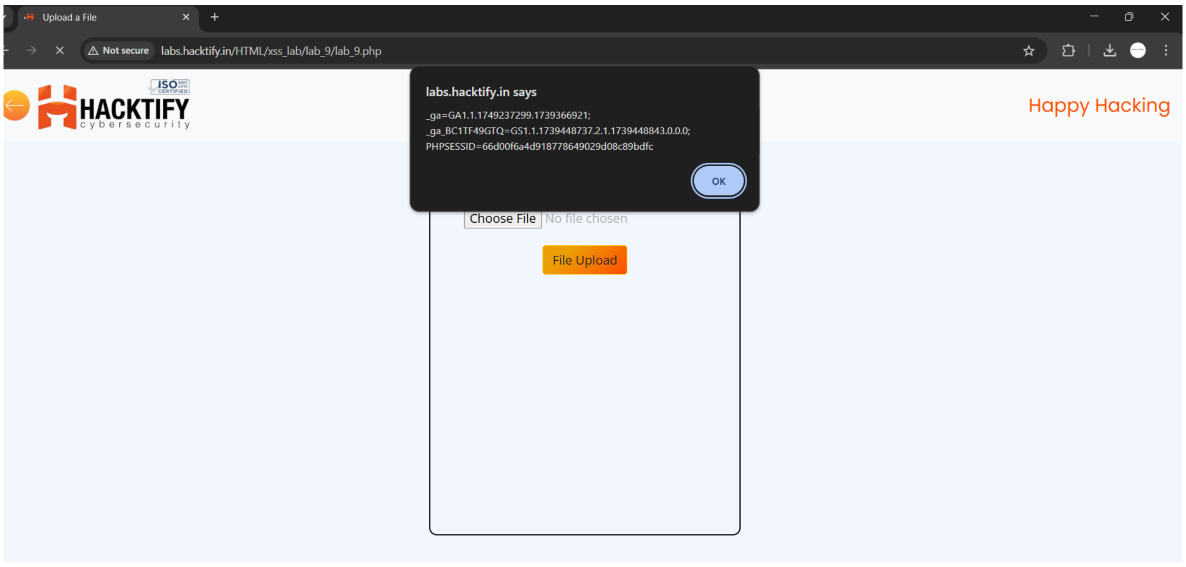
Reference	**Risk Rating**
Sub-lab-9: XSS with File Content	**Low**
Tools Used	
File containing JavaScript codes, Burp Suite	
Vulnerability Description	
It occurs when an attacker uploads an malicious JavaScript code file and when user tries to opens or preview it in the browser the code in it executed and leads to steal data or downloading malicious software into user's computer. Using Burp Suite we can see file interacting with server.	
How It Was Discovered	
Manual Analysis: By checking if the file is directly accessible after uploading.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_9/lab_9.php	
Consequences of not Fixing the Issue	
1. Can automatically install malicious software without user's permission. 2. Can run malicious code as soon as user opens file. 3. Can corrupt server. 4. Can affect multiple users at a time.	
Suggested Countermeasures	
1. Validate file type to upload. 2. Encode the content. 3. Implementing CSP. 4. Not saving file directly to server. 5. Block direct execution of uploaded files.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



```
<script>alert(document.cookie)</script>
```



labs.hacktify.in says

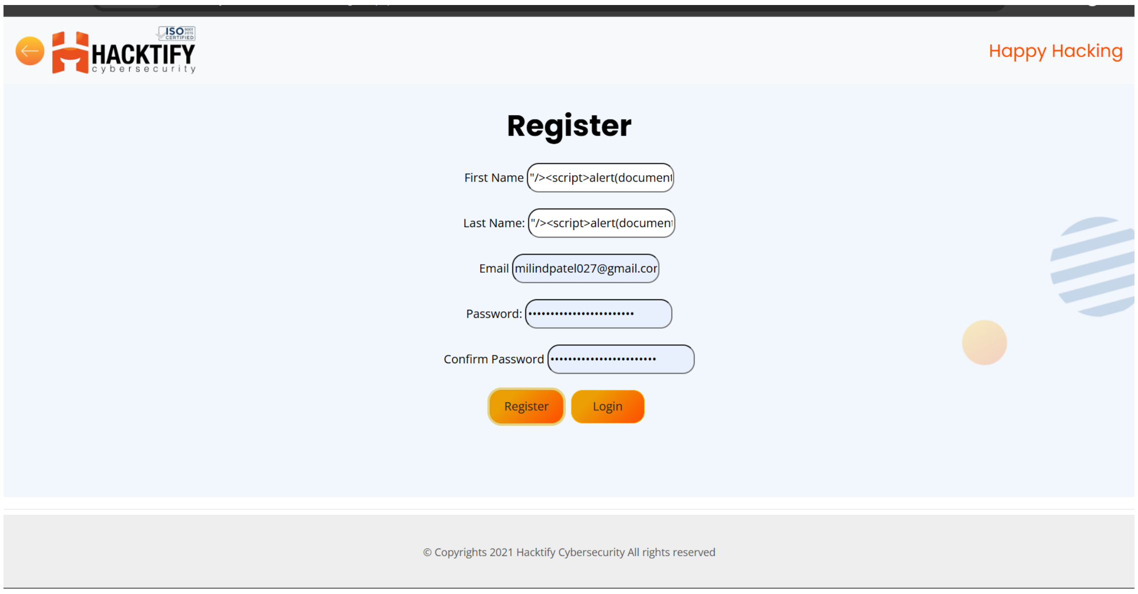
```
_ga=GA1.1.1749237299.1739366921;
_ga_BC1TF49GTQ=GS1.1.1739448737.2.1.1739448843.0.0.0;
PHPSESSID=66d00f6a4d918778649029d08c89bdfc
```

2.10. Stored Everywhere!

Reference	Risk Rating
Sub-lab-10: Stored Everywhere!	**Low**
Tools Used	
File containing JavaScript codes, Burp Suite	
Vulnerability Description	
Stored XSS is one of the most dangerous type of XSS, it gets permantly stored in website database and execute whenever user open or preview that file or content.	
How It Was Discovered	
Manual Analysis: Writing Script at every input places on webpage.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_10/profile.php	
Consequences of not Fixing the Issue	
1. Can automatically install malicious software without user's permission. 2. Can run malicious code as soon as user opens file. 3. Can corrupt server. 4. Can affect multiple users at a time.	
Suggested Countermeasures	
1. Validate file type to upload. 2. Implementing CSP. 3. Not saving file directly to server. 4. Block direct execution of uploaded files.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

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



First Name "/><script>alert(document

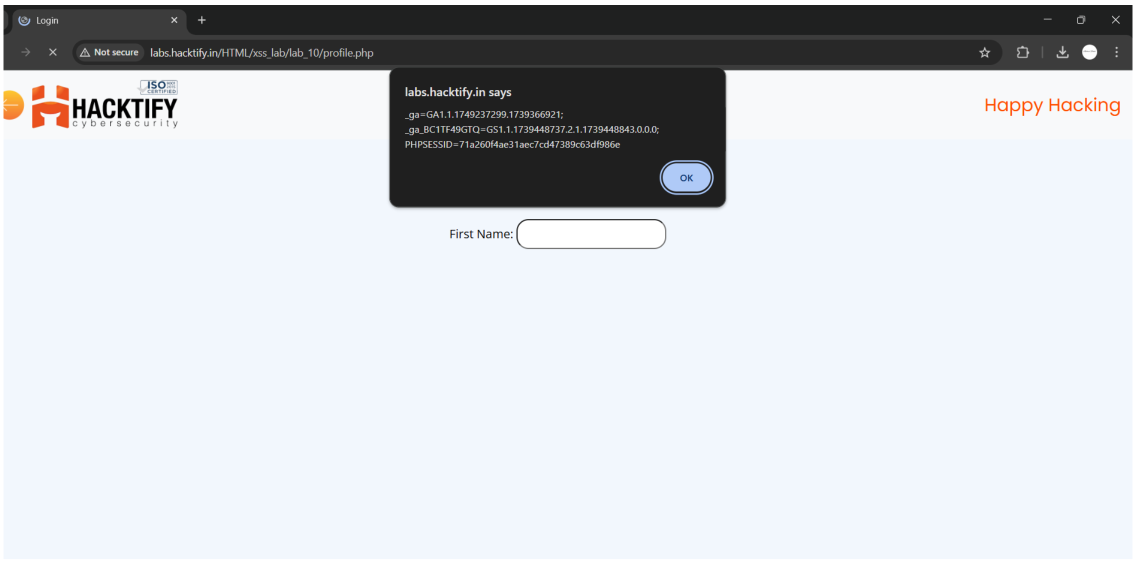
Last Name: "/><script>alert(documen

Email milindpatel027@gmail.cor

Password: ••••••••••••••••••••••

Confirm Password ••••••••••••••••••••••

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



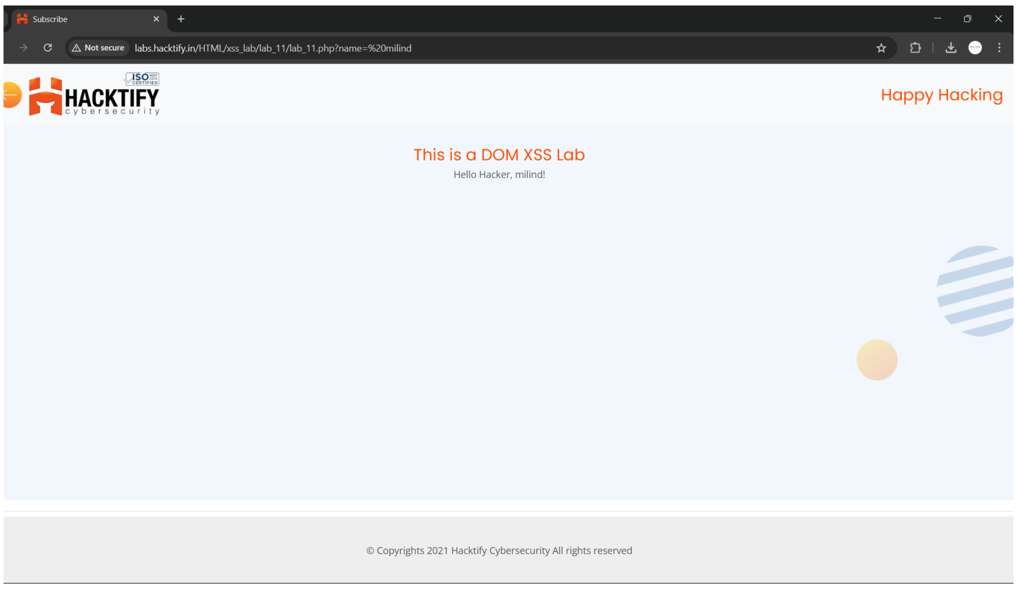
```
labs.hacktify.in says

_ga=GA1.1.1749237299.1739366921;
_ga_BC1TF49GTQ=GS1.1.1739448737.2.1.1739448843.0.0.0;
PHPSESSID=71a260f4ae31aec7cd47389c63df986e
```

2.11. DOM's are love!

Reference	**Risk Rating**
Sub-lab-11: DOM's are love!	**High**
Tools Used	
Burp Suite, File containing JavaScript codes.	
Vulnerability Description	
DOM-based XSS occurs when JavaScript manipulates the webpage directly at user browser without touching or interacting with sever.	
How It Was Discovered	
Automated Tools / Manual Analysis	
Vulnerable URLs	
https://labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php?name=%3Cimage%20src=q%20onerror=prompt(document.cookie)%3E	
Consequences of not Fixing the Issue	
1. Can automatically install malicious software without user's permission. 2. Can run malicious code as soon as user opens file. 3. Keylogging. 4. Steal passwords, login credentials.	
Suggested Countermeasures	
1. Validate file type to upload. 2. Implementing CSP. 3. Block direct installation of any file. 4. Using good Anti-Virus Software	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection	

Proof of Concept

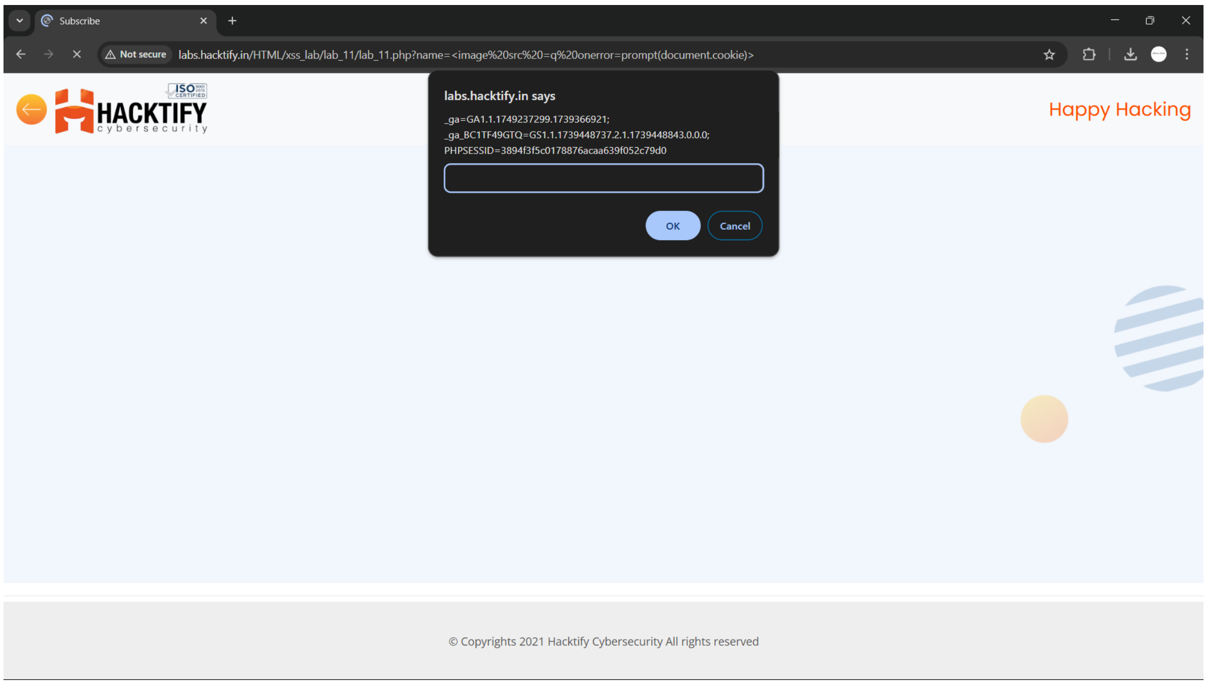
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab


Subscribe
Not secure
labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php?name=%20milind
HACKTIFY
cybersecurity
Happy Hacking

This is a DOM XSS Lab

Hello Hacker, milind!

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



labs.hacktify.in/HTML/xss_lab/lab_11/lab_11.php?name=<image%20src%20=q%20onerror=prompt(document.cookie)>

labs.hacktify.in says

```
_ga=GA1.1.1749237299.1739366921;
_ga_BC1TF49GTQ=GS1.1.1739448737.2.1.1739448843.0.0.0;
PHPSESSID=3894f3f5c0178876acaa639f052c79d0
```

© Copyrights 2021 Hacktify Cybersecurity All rights reserved