**Penetration Testing Report**

**Full Name:   
Program: HCS - Penetration Testing Internship Week-2**

**Date:02-03-2024**

**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** | **{Cross site scripting}, {Insecure direct object references }** |

**3. Summary**

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

**Total number of Sub-labs: {15} Sub-labs**

|  |  |  |
| --- | --- | --- |
| **High** | **Medium** | **Low** |
| **{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**

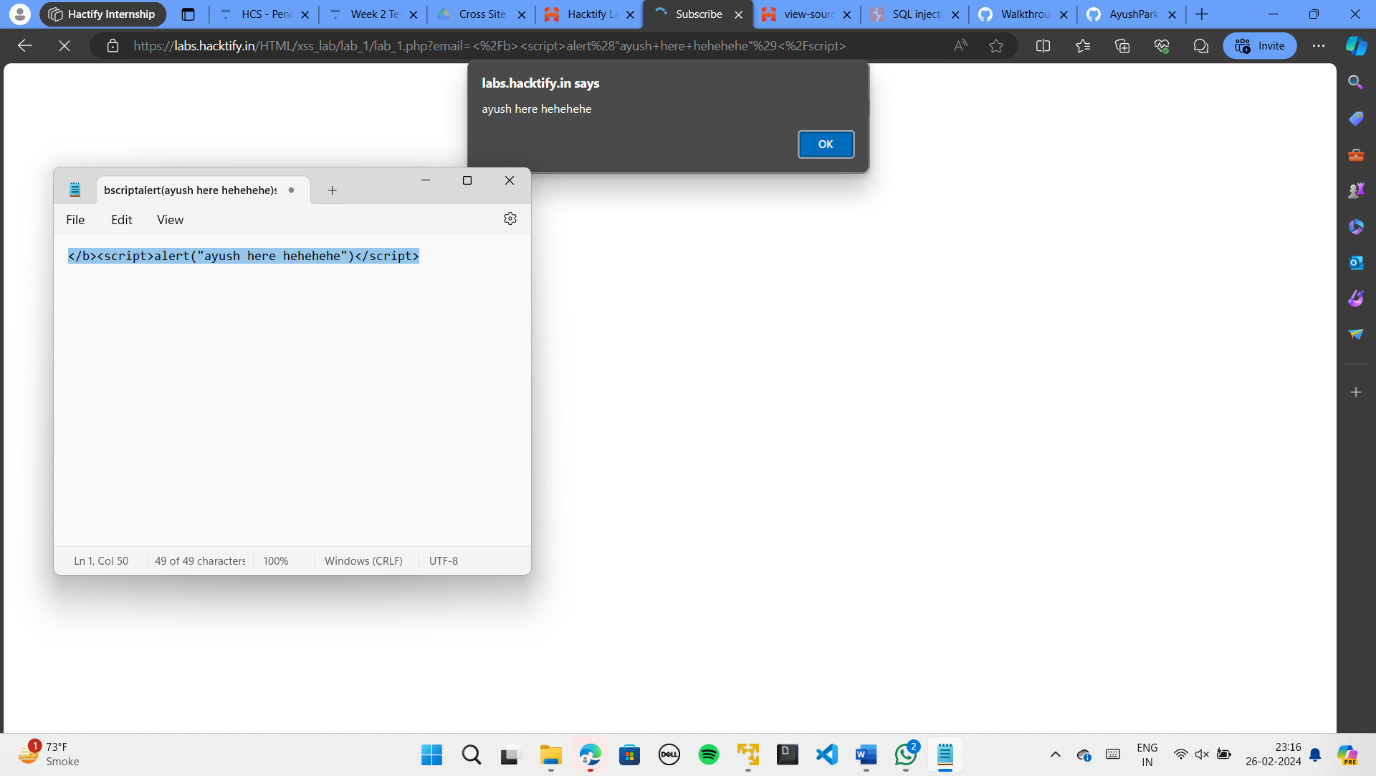
# 1. {Cross site scripting}

# 1.1. {Let’s Do It!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Let’s Do It!} | **Low** |
| **Tools Used** | |
| Use XSS payloads ,Payloads with <script> | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis by exploring input fields such as search field , text button , input text fields , etc… | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_1/lab\_1.php | |
| **Consequences of not Fixing the Issue** | |
| XSS can cause a variety of problem for the end user like –   * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| Countermeasures of XSS are;   * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# 

# Proof of Concept

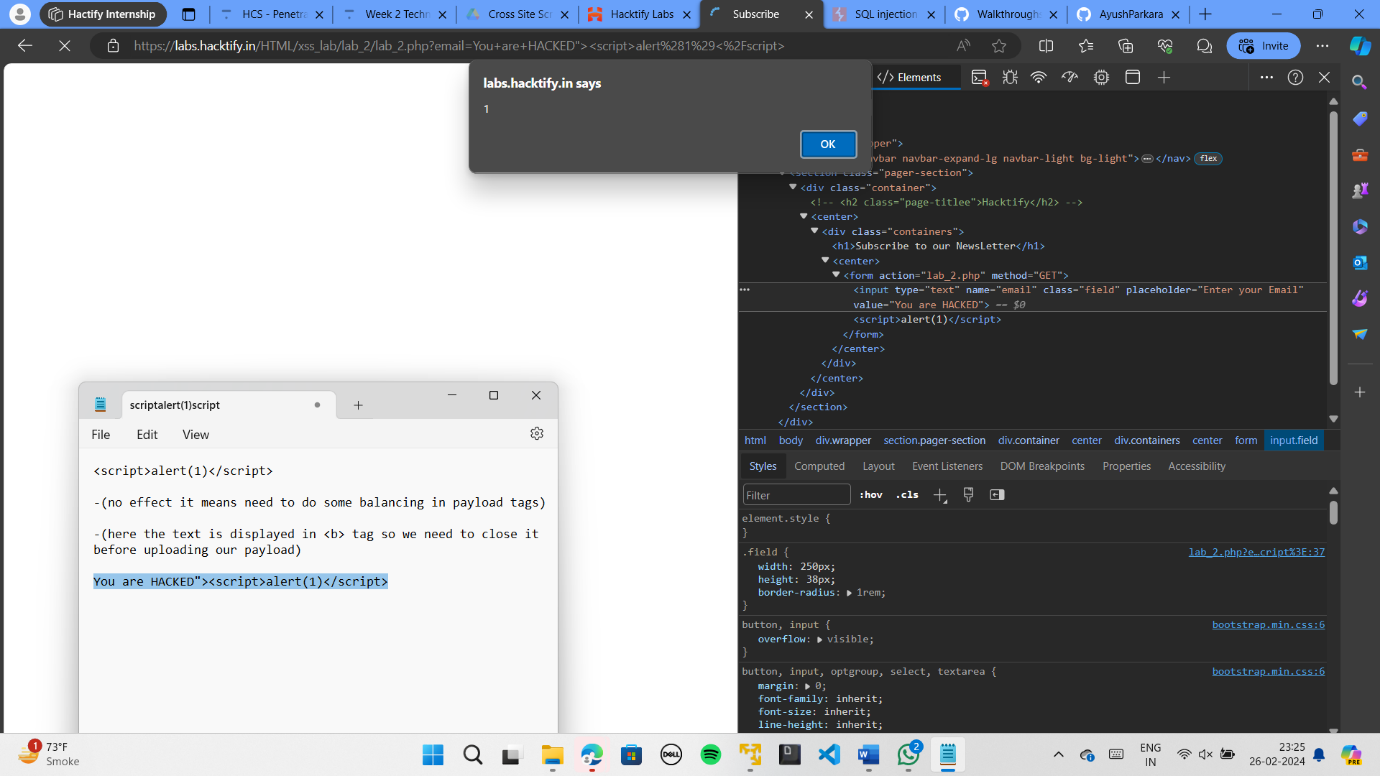


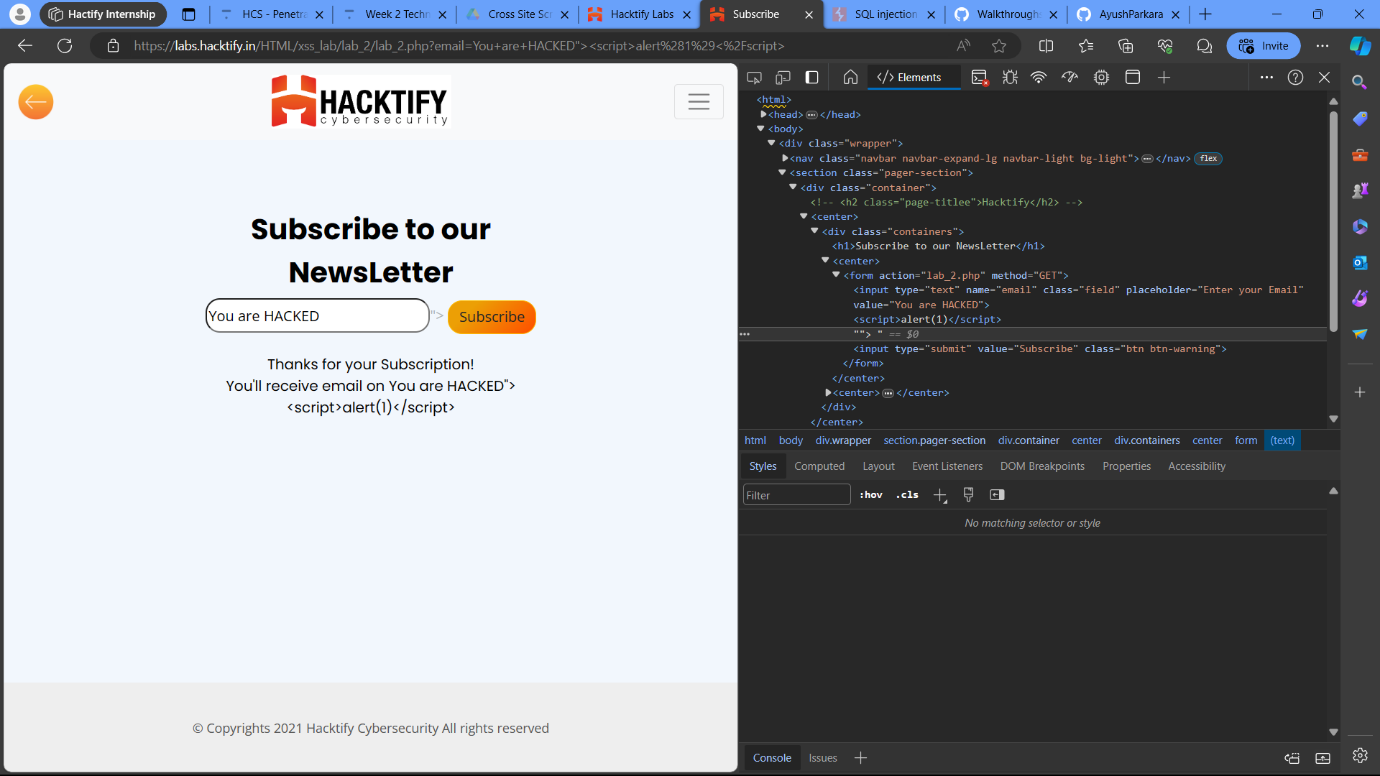
Here the site is also vulnerable to HTML Injection also.

# 1.2. {Balancing is important in life!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Balancing is important in life!} | **Low** |
| **Tools Used** | |
| XSS Payloads with <script> | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_2/lab\_2.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# Proof of Concept

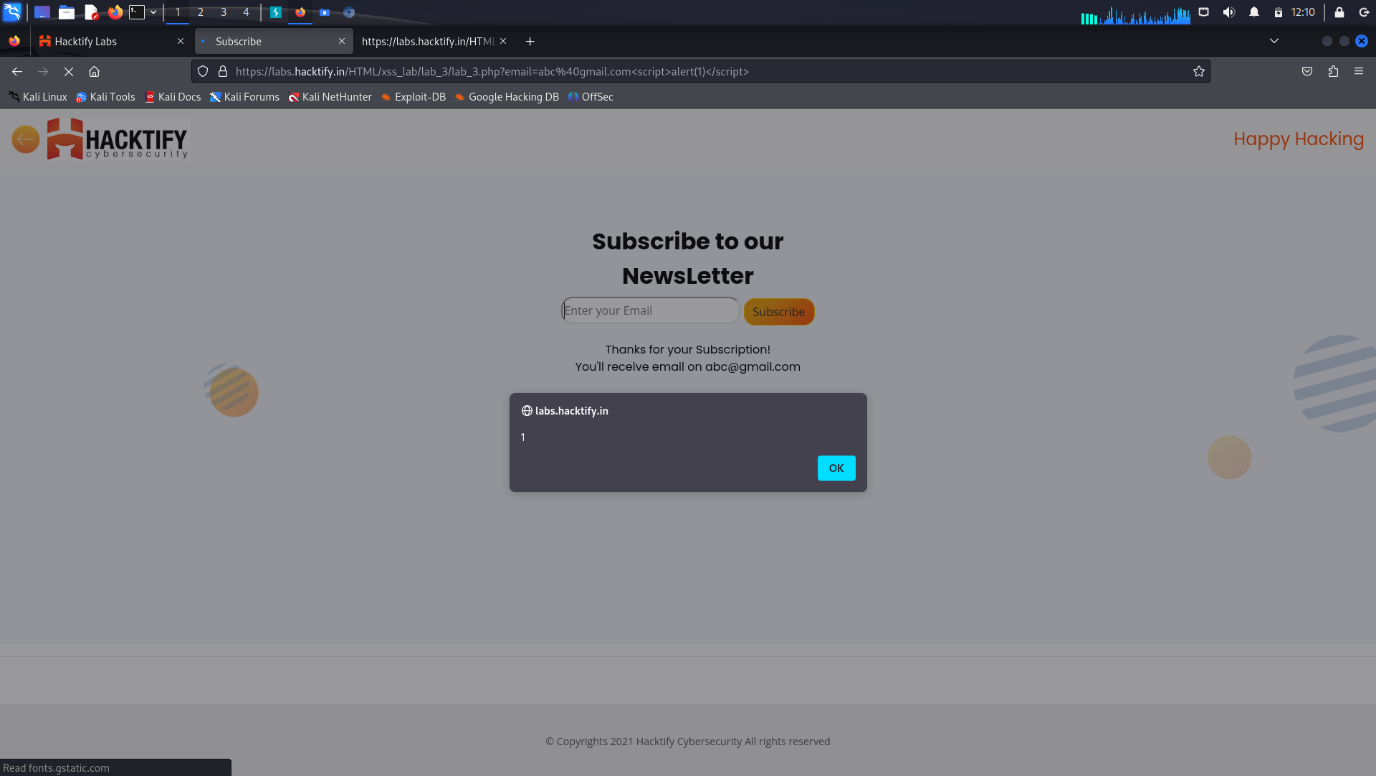




# 1.3. {XSS is everywhere!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {XSS is everywhere!} | **Low** |
| **Tools Used** | |
| XSS payload with <script> | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_3/lab\_3.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

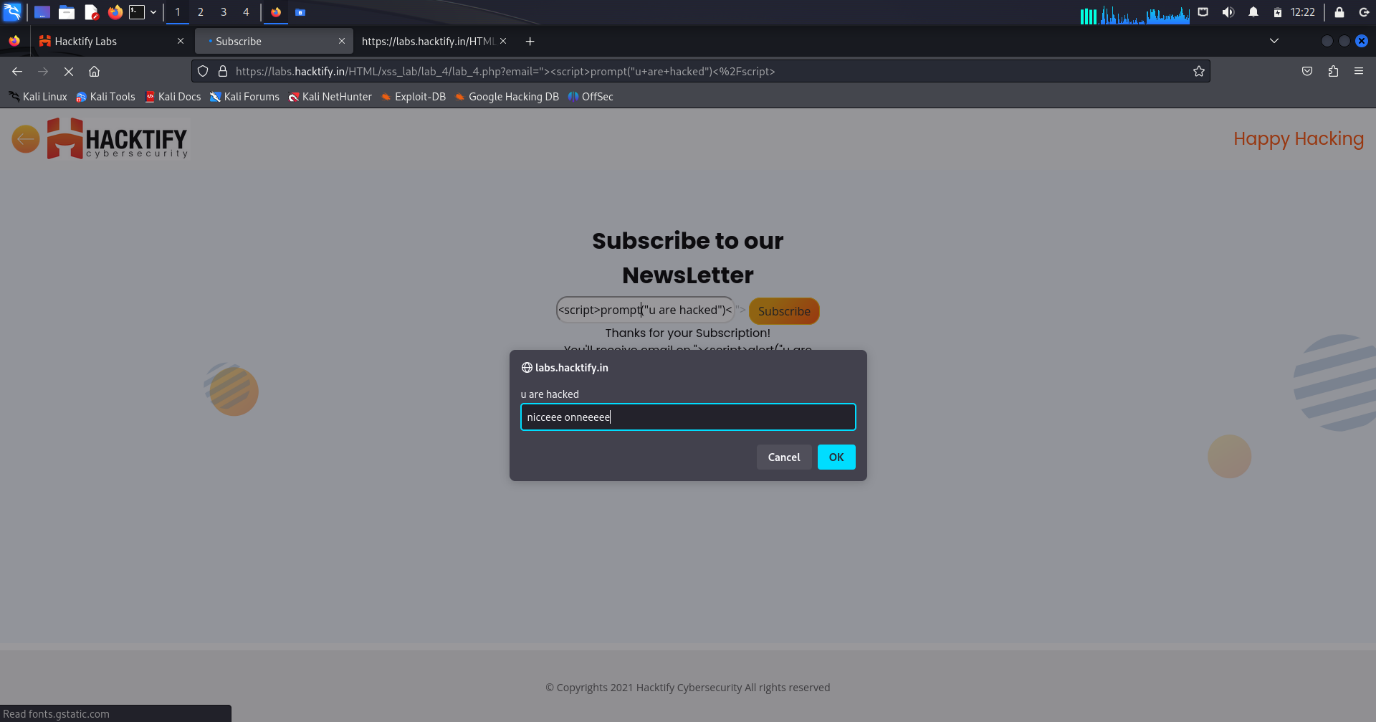
# Proof of Concept

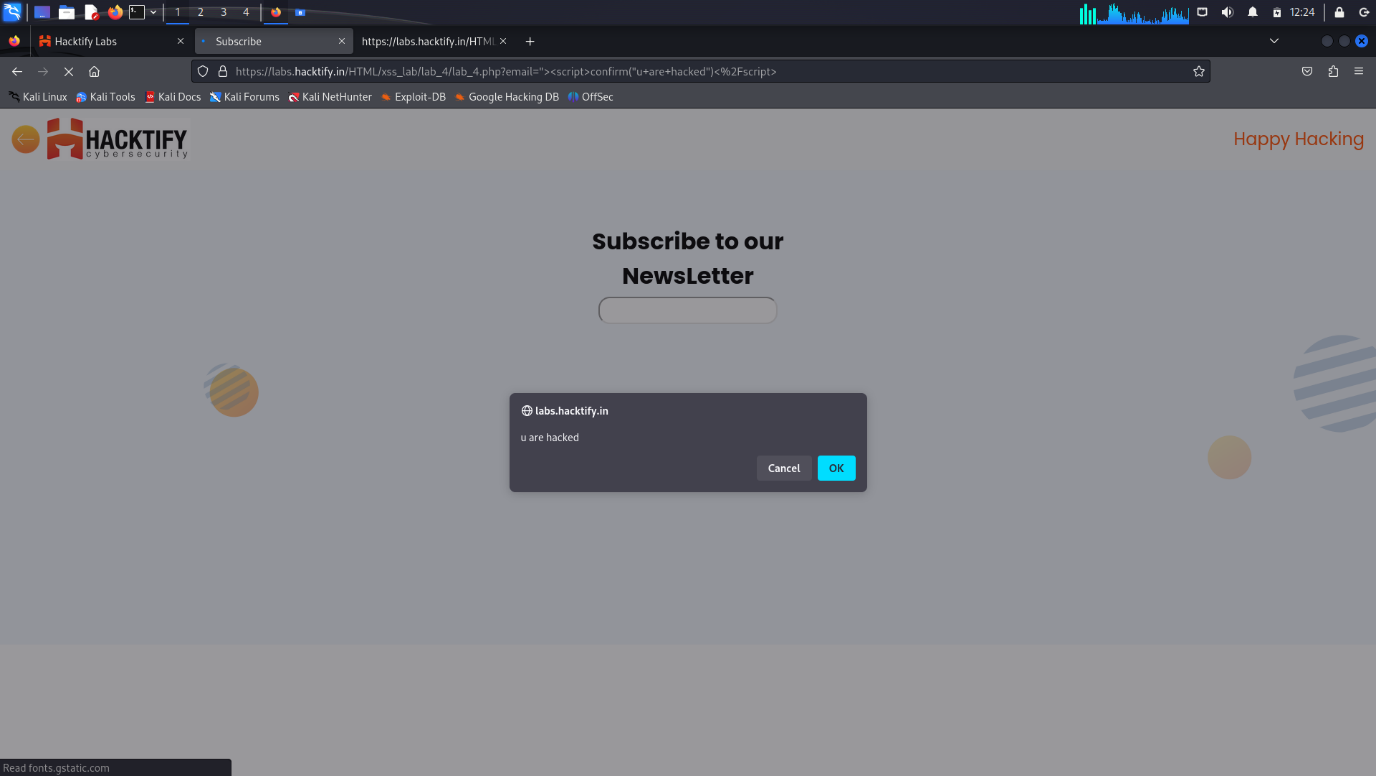


# 1.4. {Alternatives are must!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Alternatives are musst! } | **Medium** |
| **Tools Used** | |
| XSS payloads | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_4/lab\_4.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# Proof of Concept

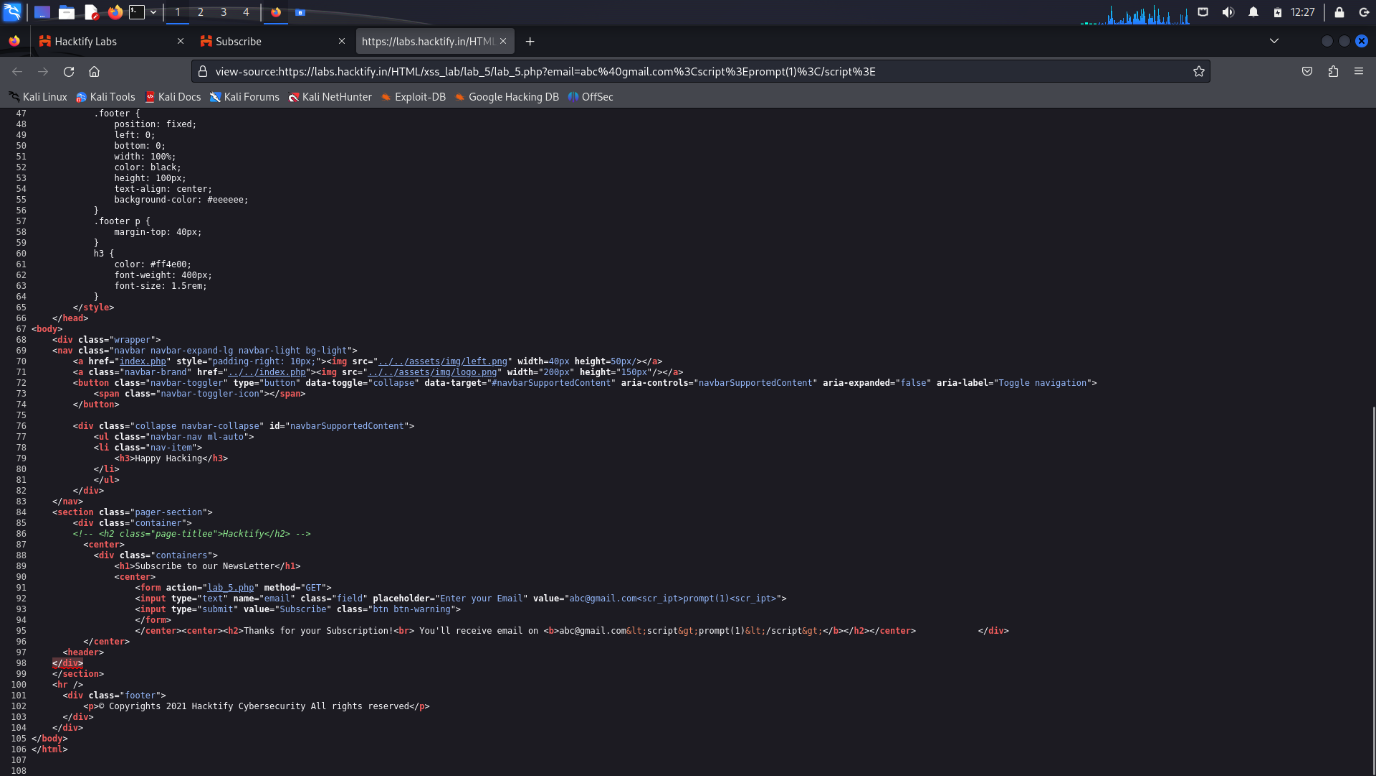


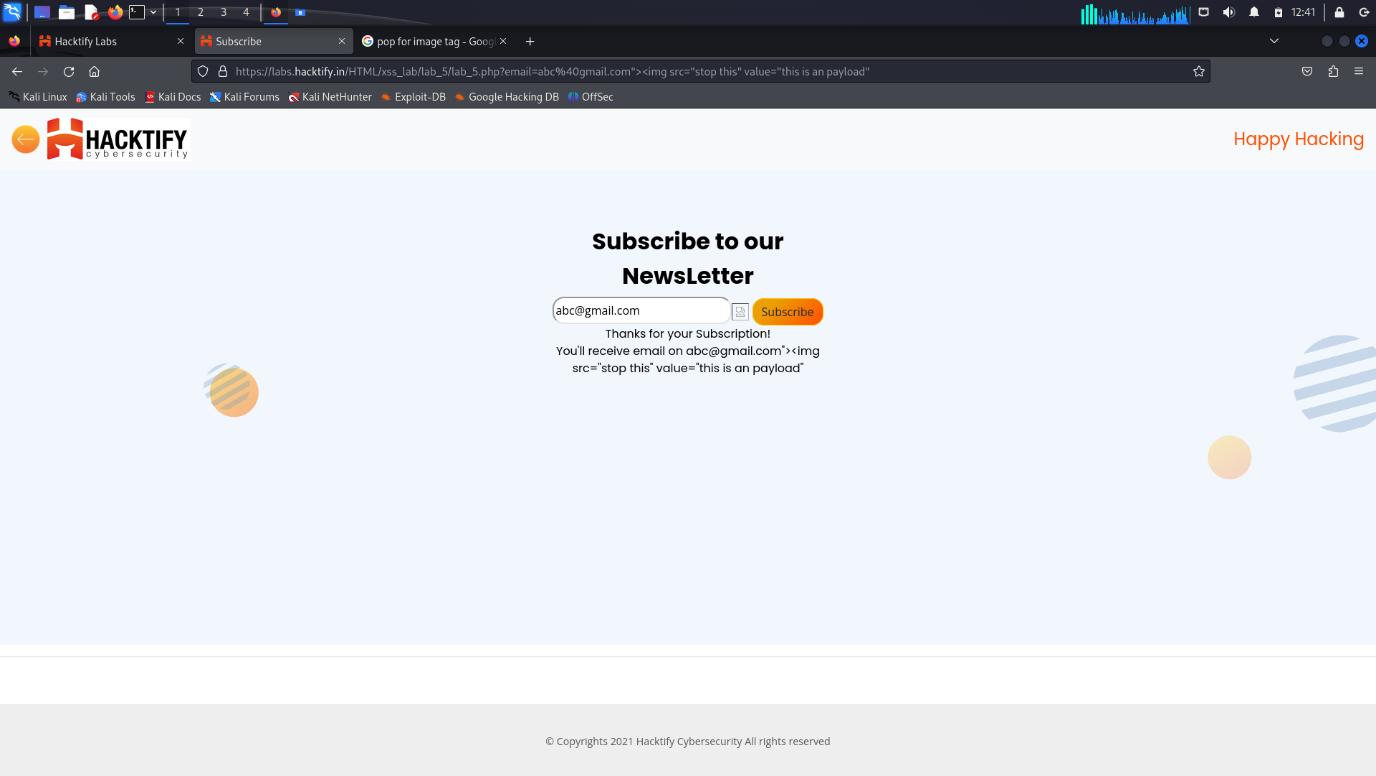


# 1.5. {Developer hates scripts!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Developer hates scripts!} | **High** |
| **Tools Used** | |
| XSS payload with <img> | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_5/lab\_5.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# Proof of Concept

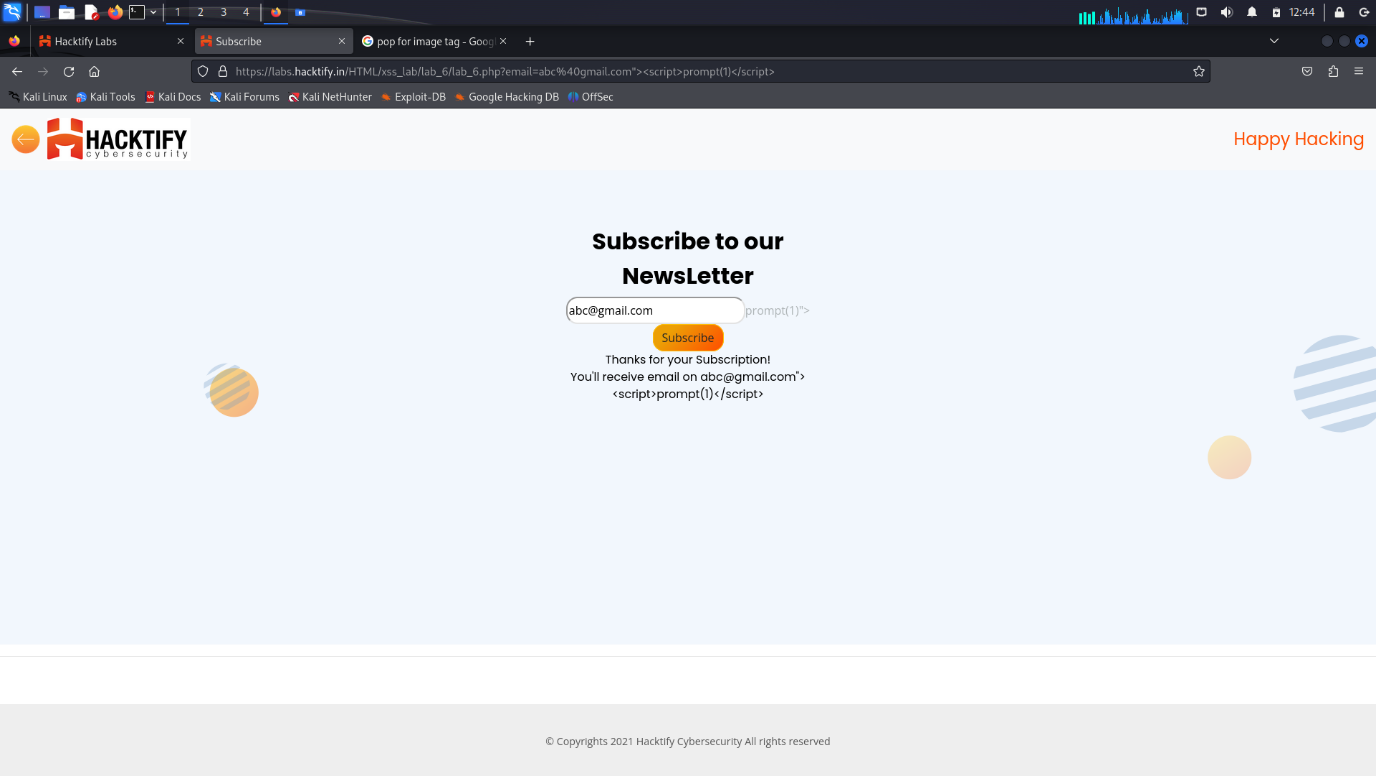


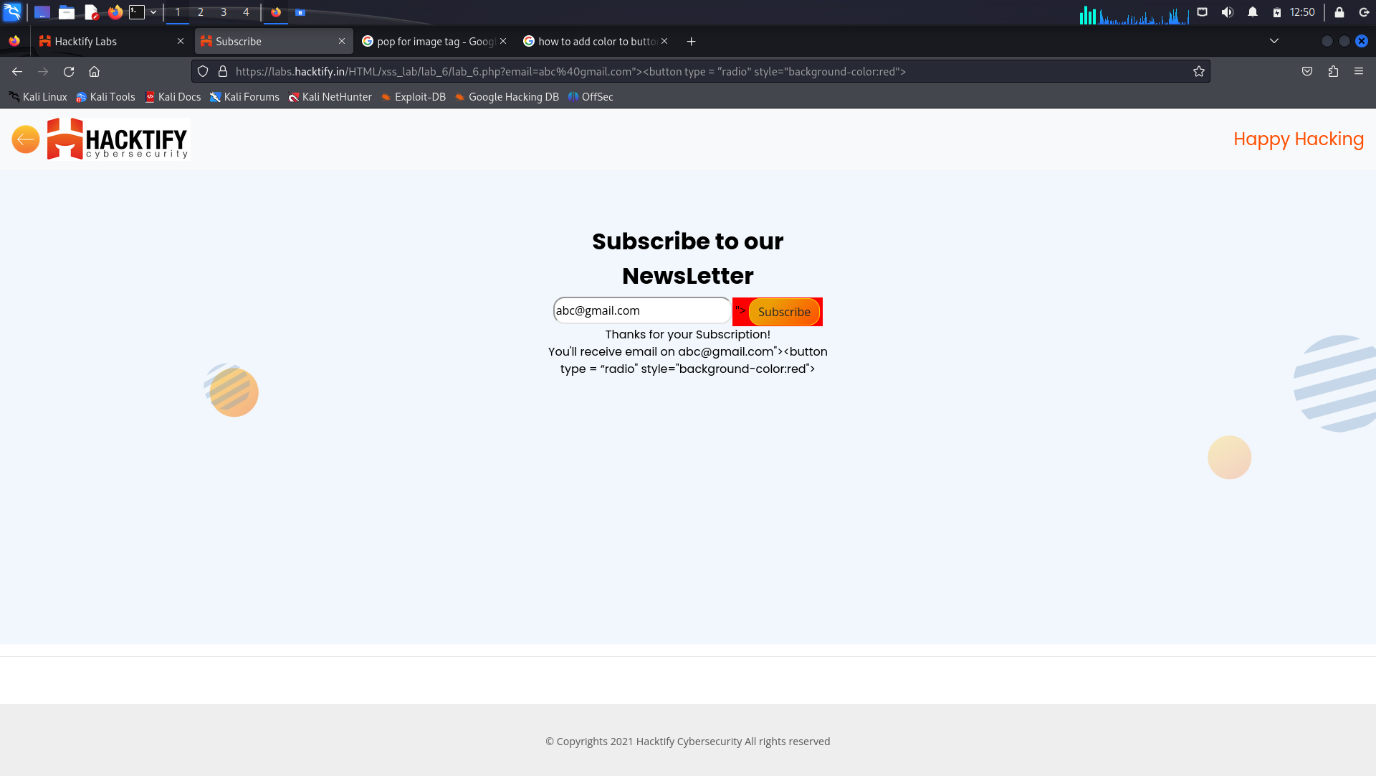


# 1.6. {change the variation!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Change the variation!} | **High** |
| **Tools Used** | |
| XSS payloads with <script> | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_6/lab\_6.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# Proof of Concept

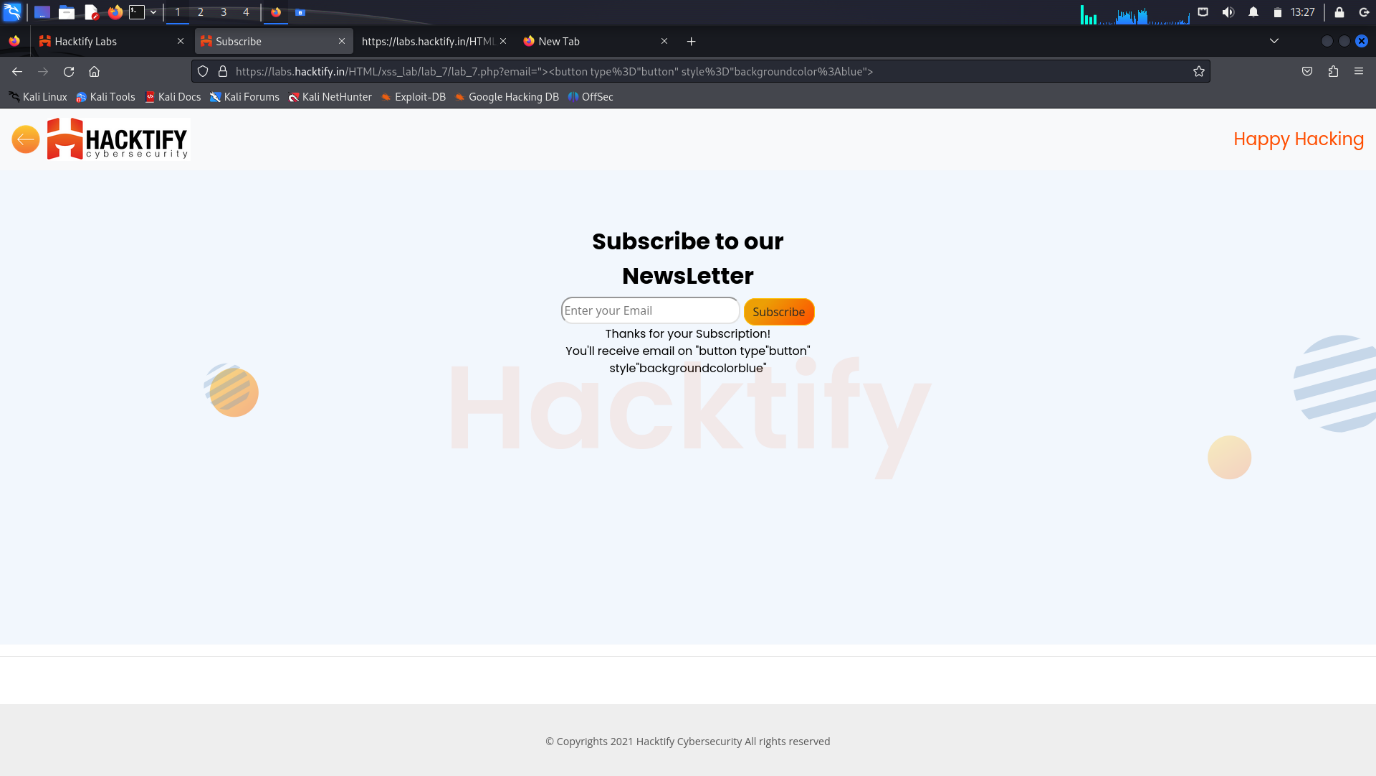


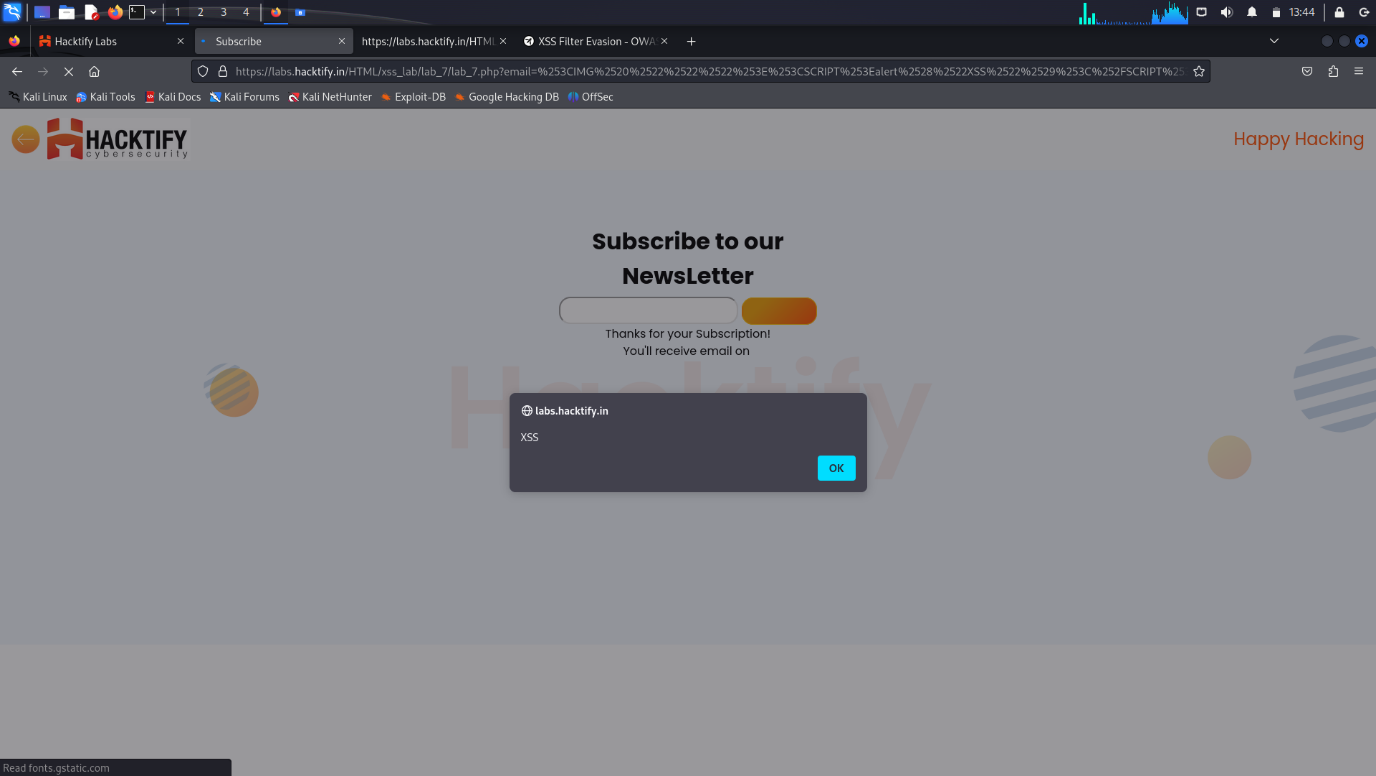


# 1.7. {Encoding is the key?}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Encoding is the key?} | **Medium** |
| **Tools Used** | |
| XSS payload with<script> , encoding payloads , … | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_7/lab\_7.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# Proof of Concept

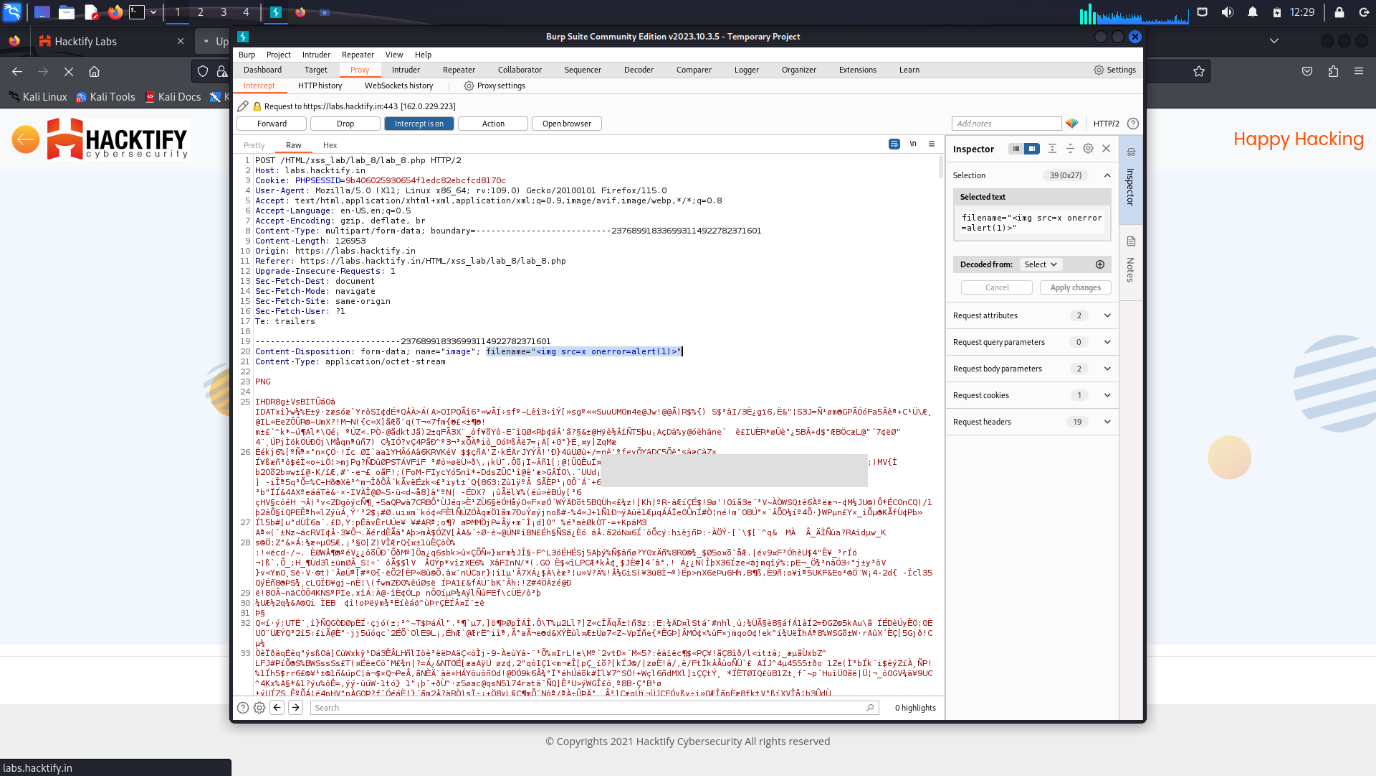


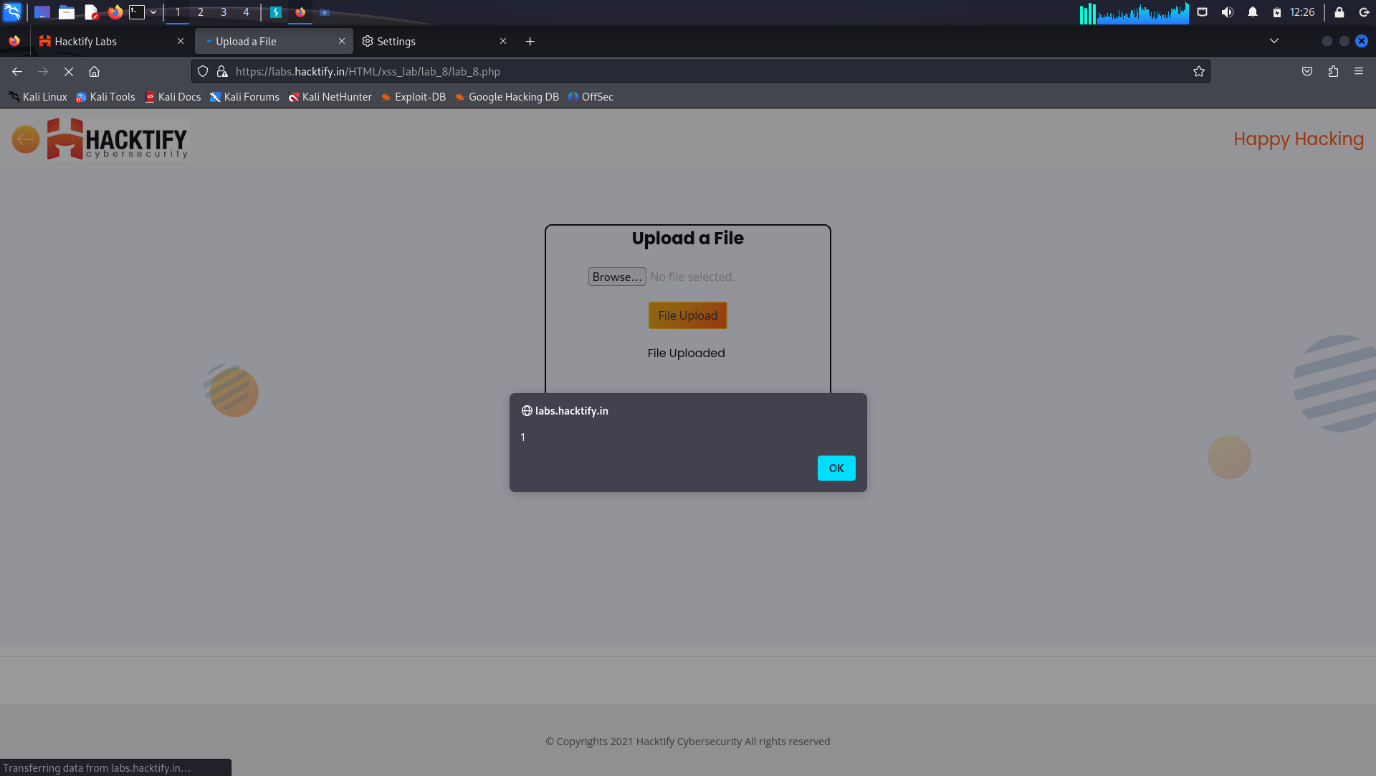


# 1.8. {XSS with file upload(File name)}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {XSS with file upload (File name)} | **Low** |
| **Tools Used** | |
| XSS payload with <img> , burpsuite | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_8/lab\_8.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

# Proof of Concept

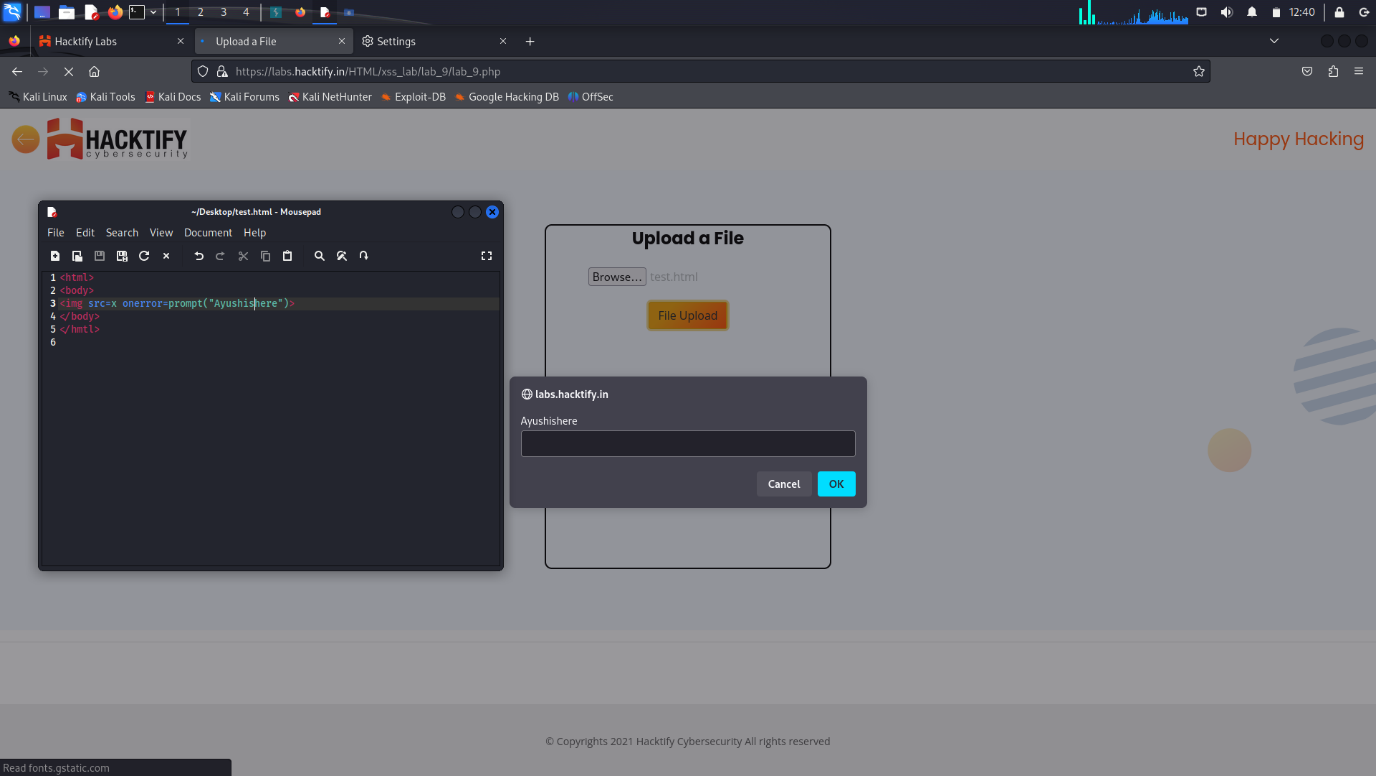




# 1.9. {XSS with file upload(file content)}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {XSS with file upload(file content) } | **Medium** |
| **Tools Used** | |
| XSS payload with <script> , burpsuite | |
| **Vulnerability Description** | |
| XSS is a web security vulnerability that allows an attacker to compromise the interactions that user have with vulnerable applications.  XSS normally allow an attacker to act as a victim user and manipulate a vulnerable website’s source code so that it returns malicious code to users. when malicious code executes inside a victim’s browser , attacker can compromise interactions with the apps by stealing session cookies, user credentials, tokens, secrets, etc… | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_9/lab\_9.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * Output encoding * Html sanitization * Input sanitization * Implement content security policy * X-XSS protection header | |
| **References** | |
| <https://owasp.org/www-community/attacks/xss/#:~:text=XSS%20Attack%20Consequences&text=XSS%20can%20cause%20a%20variety,and%20take%20over%20the%20account>.  <https://portswigger.net/web-security/cross-site-scripting>  <https://www.acunetix.com/websitesecurity/cross-site-scripting/>  <https://www.invicti.com/blog/web-security/cross-site-scripting-xss/> | |

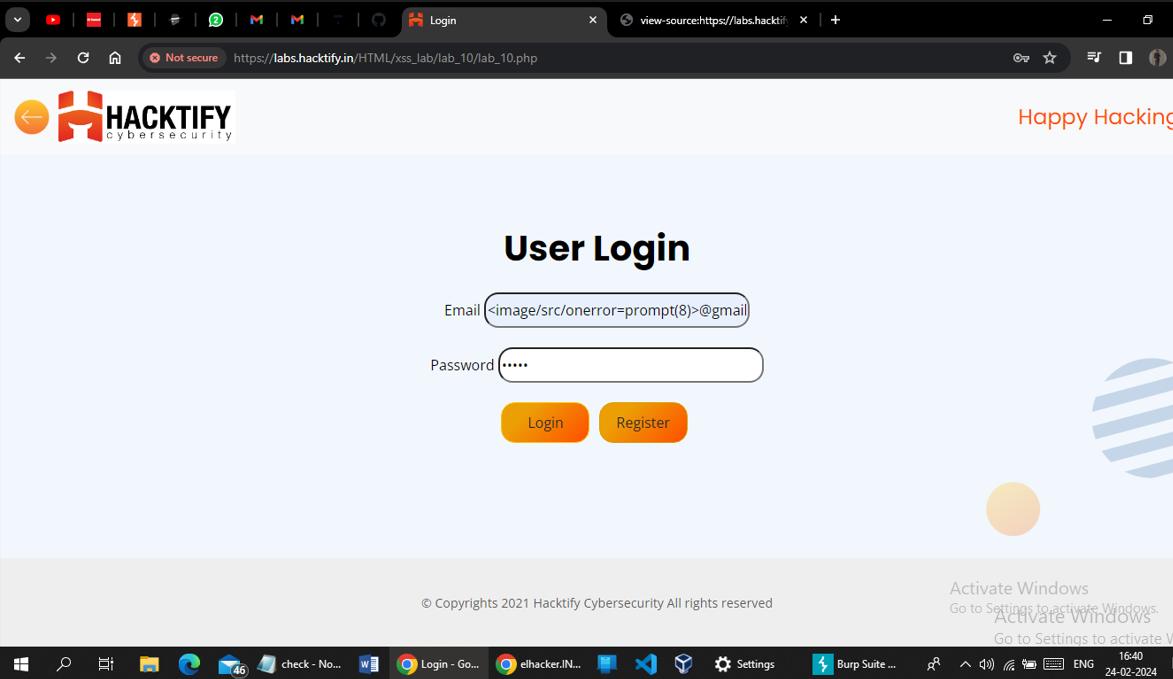
# Proof of Concept

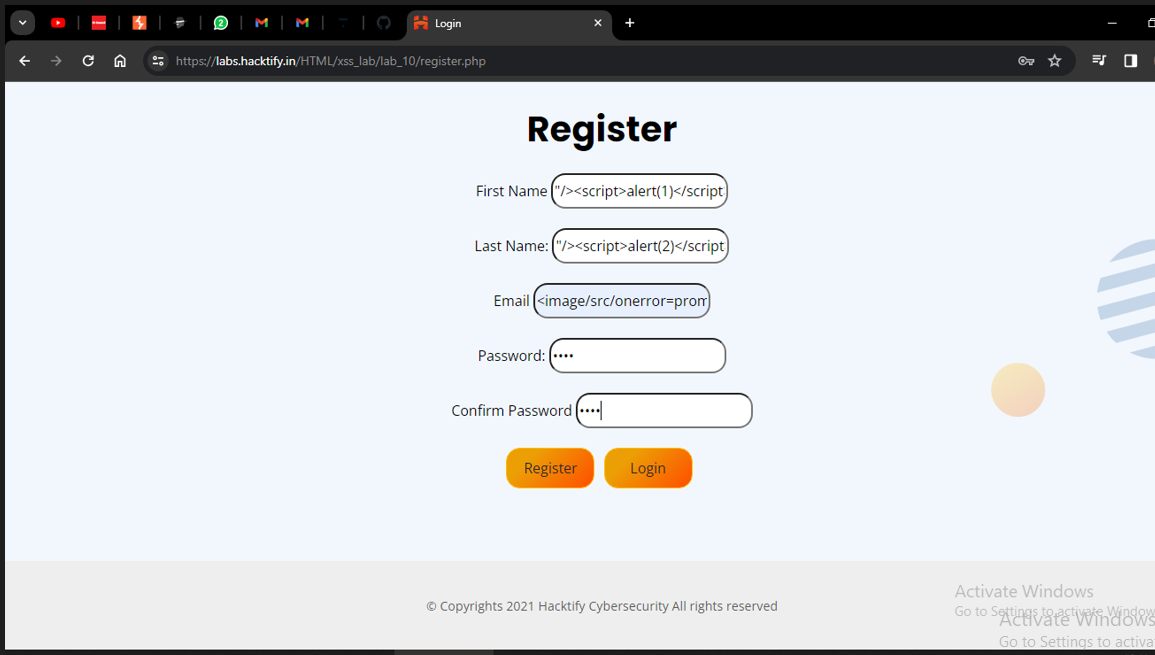


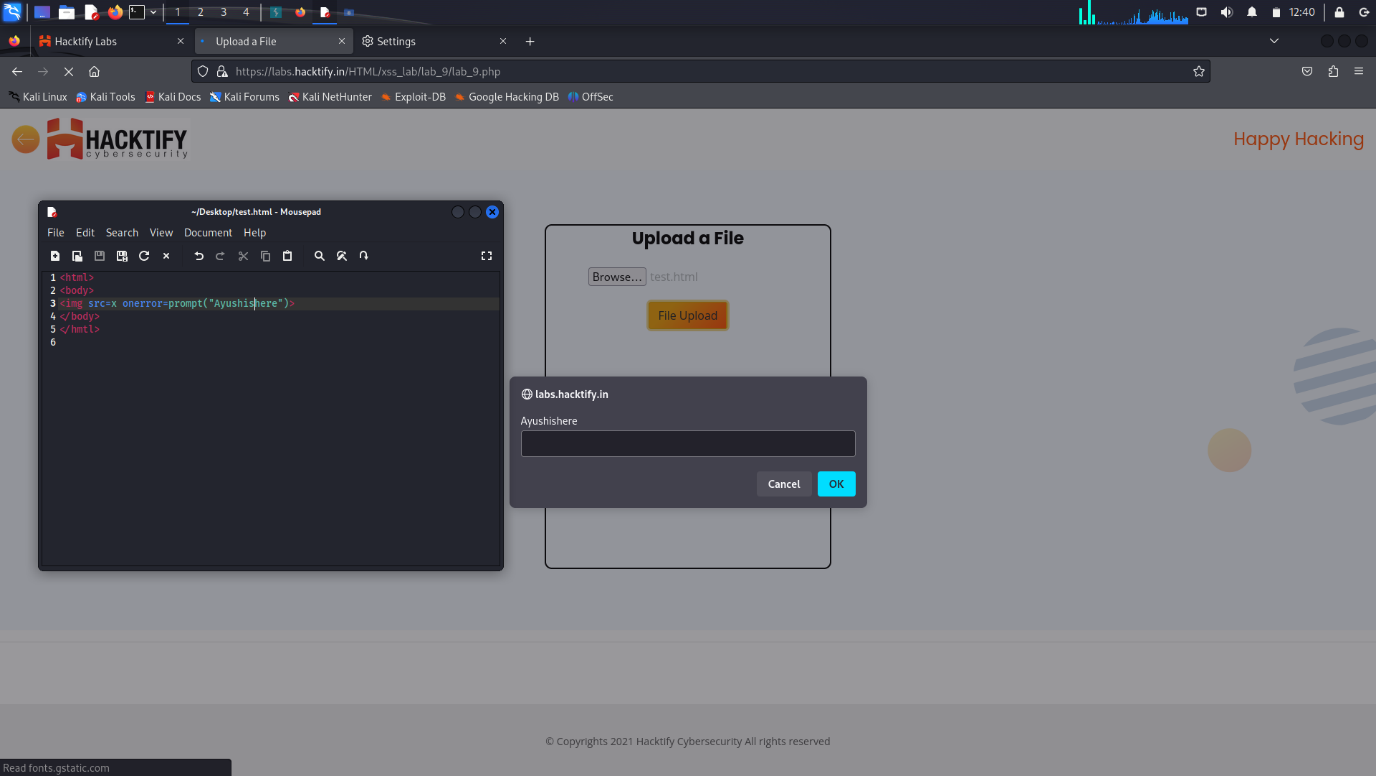
# 1.10. {Stored everywhere! }

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {stored everywhere!} | **Low** |
| **Tools Used** | |
| XSS payload | |
| **Vulnerability Description** | |
| In Stored XSS the malicious script comes from website’s database which eventually gets executed in usr’s browser. This type of XSS can be used to steal cookies of large number of users or admin as well  When unsuspecting users access the affected pages, the injected scripts are executed within their browsers, potentially leading to various harmful consequences. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_10/register.php | |
| **Consequences of not Fixing the Issue** | |
| * Account hijacking * Stealing sensitive data * Session hijacking * Financial loss * Data manipulation * Disclosure of user’s session cookie * Allow an attacker to hijack user’s session and take over the account | |
| **Suggested Countermeasures** | |
| * HTTPonly and secure cookies * Database security * Security testing * Web application firewall * Secure handling of user input * Request blocking * Choosing right framework * Mitigating damage of an XSS attack | |
| **References** | |
| <https://brightsec.com/blog/stored-xss/>  <https://portswigger.net/web-security/cross-site-scripting/stored>  <https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/>  <https://owasp.org/www-community/attacks/xss/> | |

# Proof of Concept



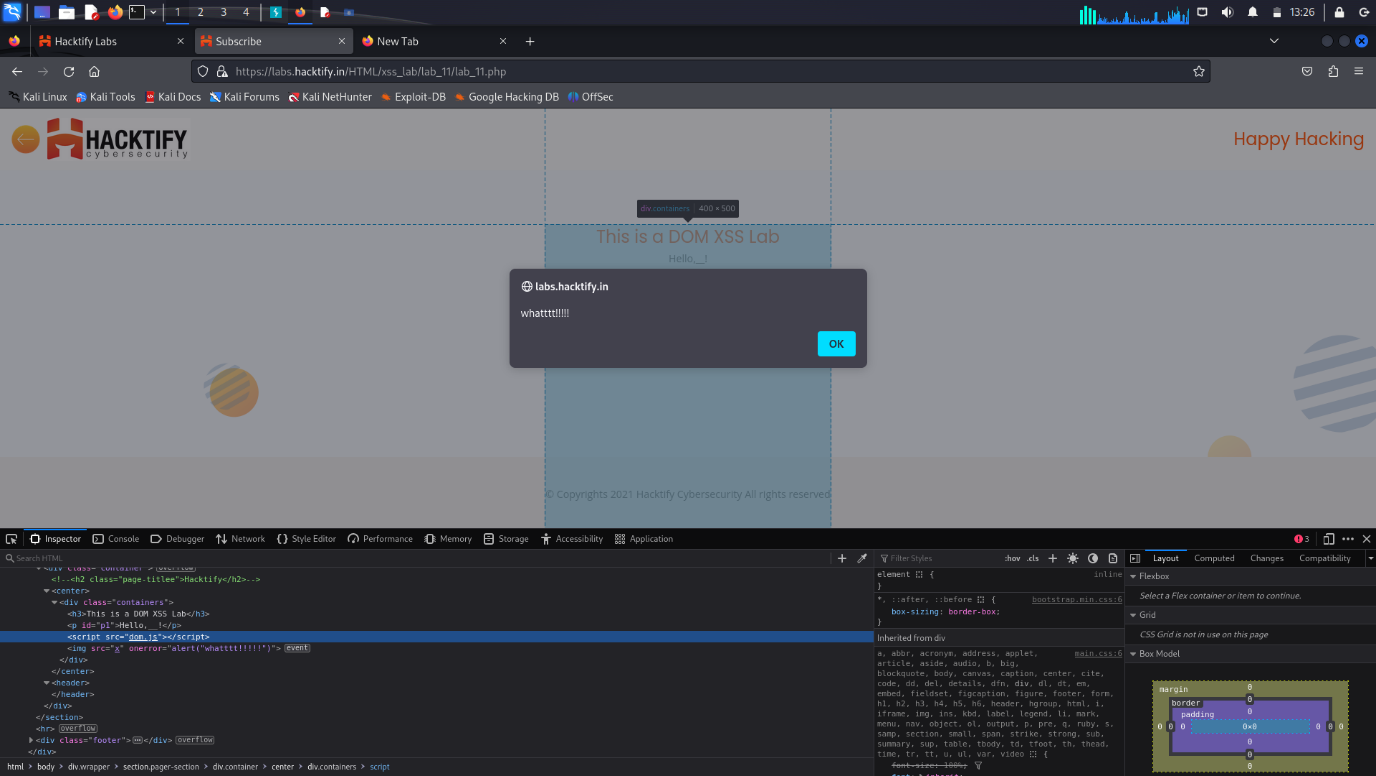


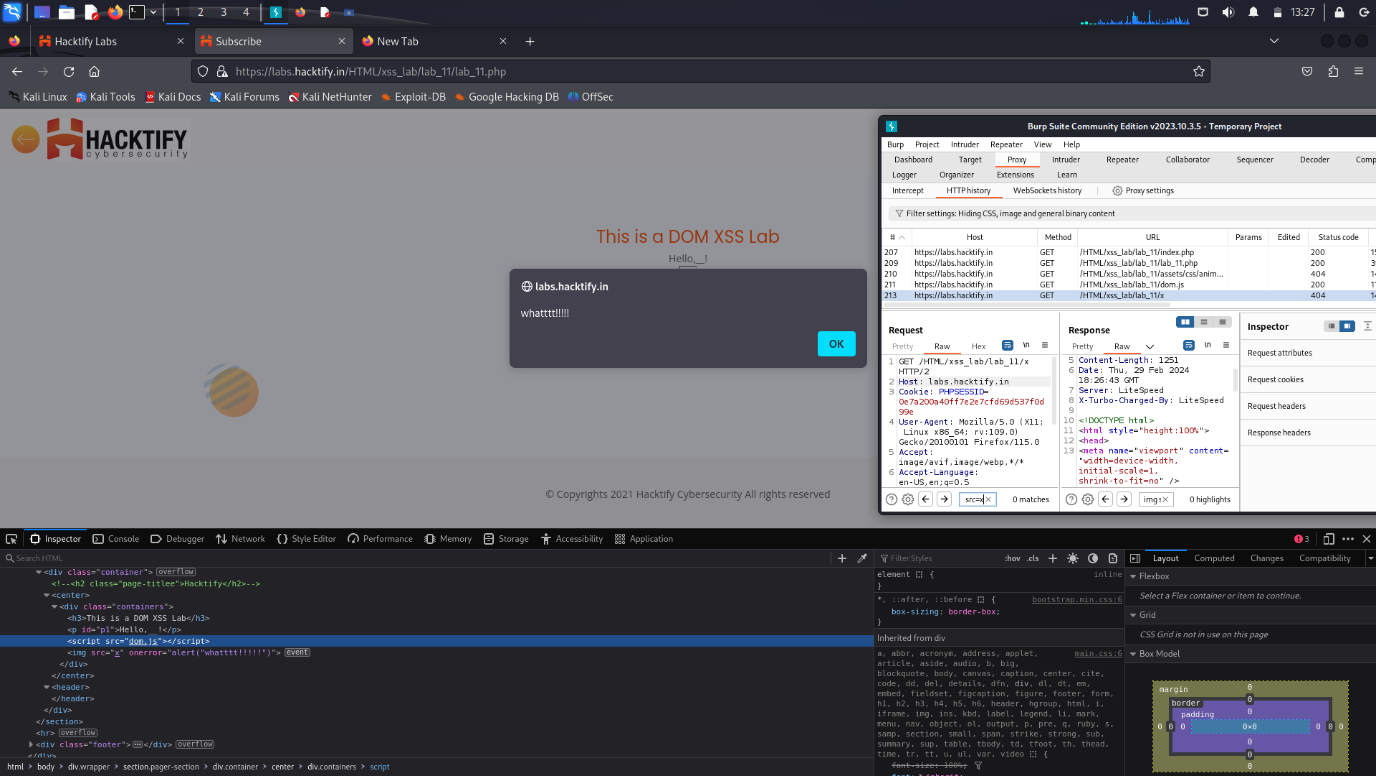


# 1.11. {DOM’S are love!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {DOM’S are love!} | **High** |
| **Tools Used** | |
| XSS payload with <script> , <img>, etc… | |
| **Vulnerability Description** | |
| DOM based vulnerability exists in client-side code rather than server-side code.  In DOM based XSS , malicious user input goes inside the source and comes out of the sink | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/xss\_lab/lab\_11/lab\_11.php | |
| **Consequences of not Fixing the Issue** | |
| * Data theft * Session hijacking * Phishing attacks * Client-side redirection * Account compromise | |
| **Suggested Countermeasures** | |
| * Input validation * Output encoding * Content security policy * Use safe sink * Instead of inner html use innertext , textcontent * Use typical XSS protection technique * Security awareness training | |
| **References** | |
| <https://portswigger.net/web-security/cross-site-scripting/dom-based>  <https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/>  <https://owasp.org/www-community/attacks/xss/>  <https://brightsec.com/blog/xss/>  <https://portswigger.net/web-security/cross-site-scripting/dom-based#:~:text=This%20enables%20attackers%20to%20execute,causes%20execution%20of%20arbitrary%20JavaScript>. | |

# Proof of Concept





# 2. {Insecure direct object references}

# 2.1. {Give me my amount!!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Give me my amount!} | **Low** |
| **Tools Used** | |
| Payload | |
| **Vulnerability Description** | |
| An insecure direct object reference 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. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/idor\_lab/lab\_1/profile.php?id=121 | |
| **Consequences of not Fixing the Issue** | |
| * Exposure of confidential information * Authentication bypass * Alteration of data * Account takeover | |
| **Suggested Countermeasures** | |
| * Indirect reference maps * Fuzz testing * Parameter verification * Access validation | |
| **References** | |
| <https://portswigger.net/web-security/access-control/idor>  <https://www.geeksforgeeks.org/insecure-direct-object-reference-idor-vulnerability/>  <https://www.imperva.com/learn/application-security/insecure-direct-object-reference-idor/>  <https://www.eccouncil.org/cybersecurity-exchange/web-application-hacking/idor-vulnerability-detection-prevention/>  <https://en.wikipedia.org/wiki/Insecure_direct_object_reference>. | |

# Proof of Concept:

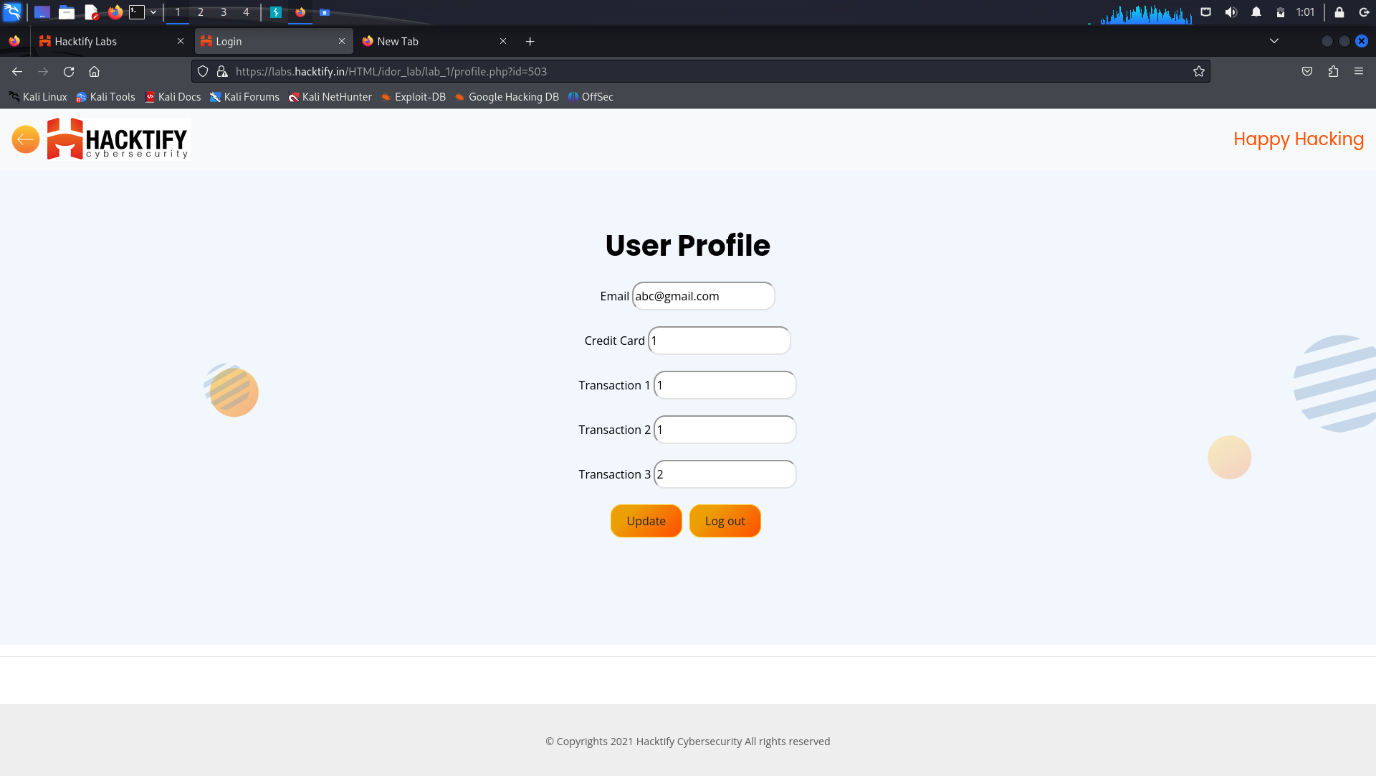
Here two acocunts were made using the register and then logged in.

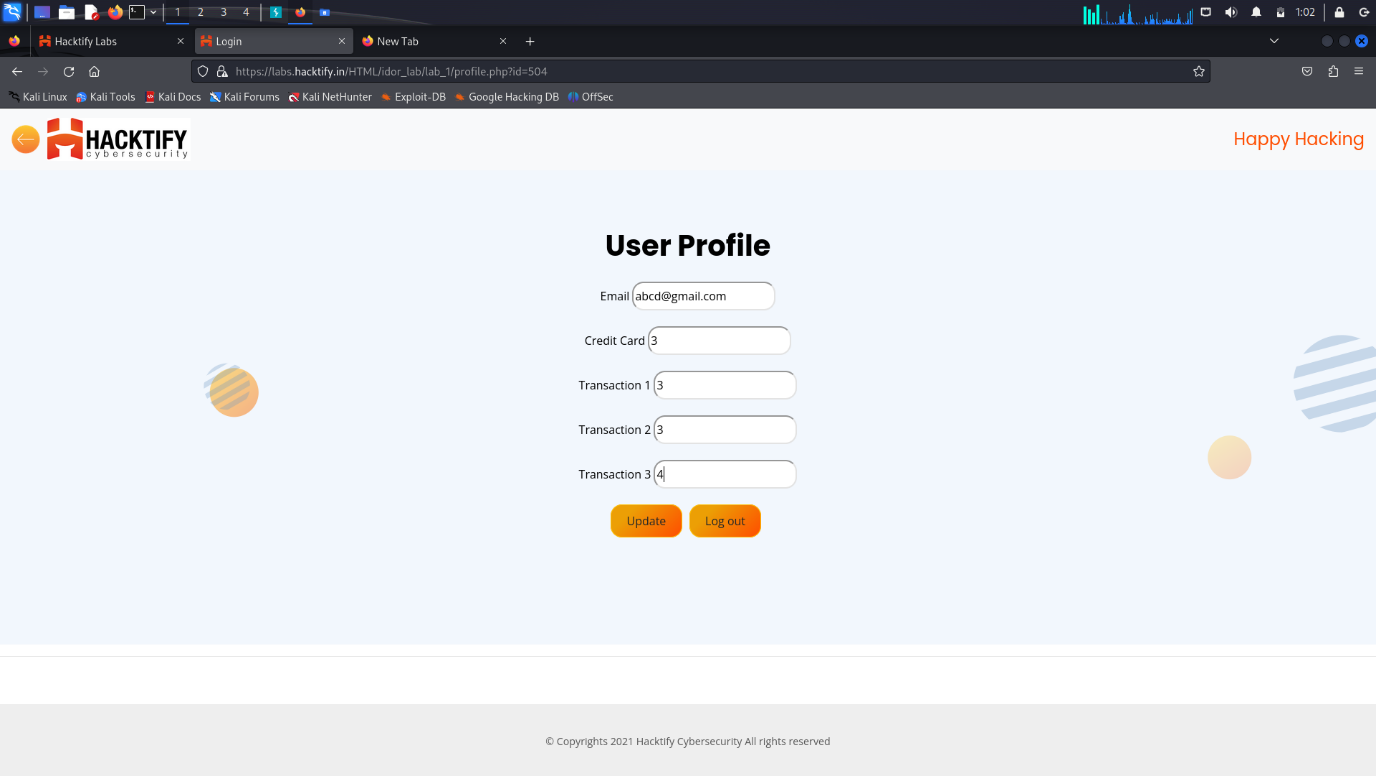
Then the user profile was visible and we can add further details of the user.

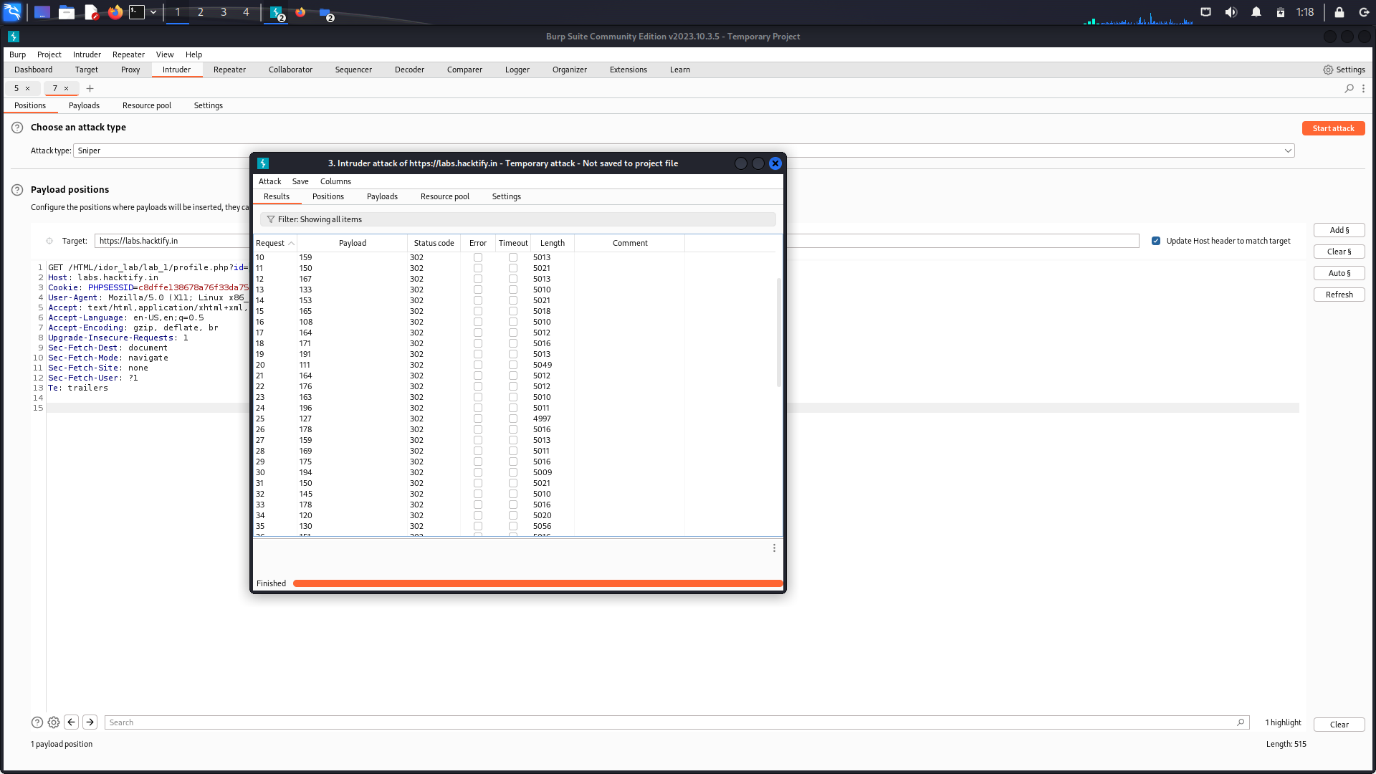
Then after doing that we can get the other user’s info which can be read by just changing the id parameter used in the url.

We have to just randomly guess the id’s and try one by one.

As we get the proper id match we will be able to get the other user’s account details and can be used by any means.







# 2.2. {Stop Polluting my params!}

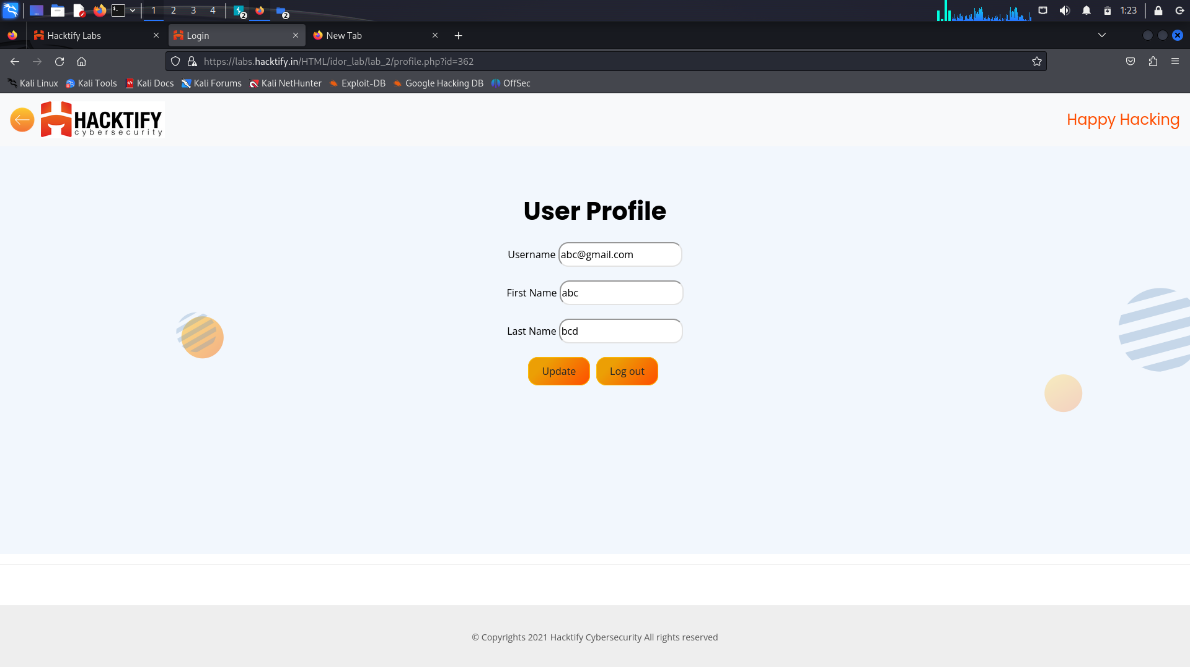
|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Stop polluting my params!} | **Medium** |
| **Tools Used** | |
| burpsuite | |
| **Vulnerability Description** | |
| An insecure direct object reference is an access control vulnerability where invalidated user input can be used for unauthorized 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. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/idor\_lab/lab\_1/profile.php?id=121 | |
| **Consequences of not Fixing the Issue** | |
| * Exposure of confidential information * Authentication bypass * Alteration of data * Account takeover | |
| **Suggested Countermeasures** | |
| * Indirect reference maps * Fuzz testing * Parameter verification * Access validation | |
| **References** | |
| <https://portswigger.net/web-security/access-control/idor>  <https://www.geeksforgeeks.org/insecure-direct-object-reference-idor-vulnerability/>  <https://www.eccouncil.org/cybersecurity-exchange/web-application-hacking/idor-vulnerability-detection-prevention/> | |

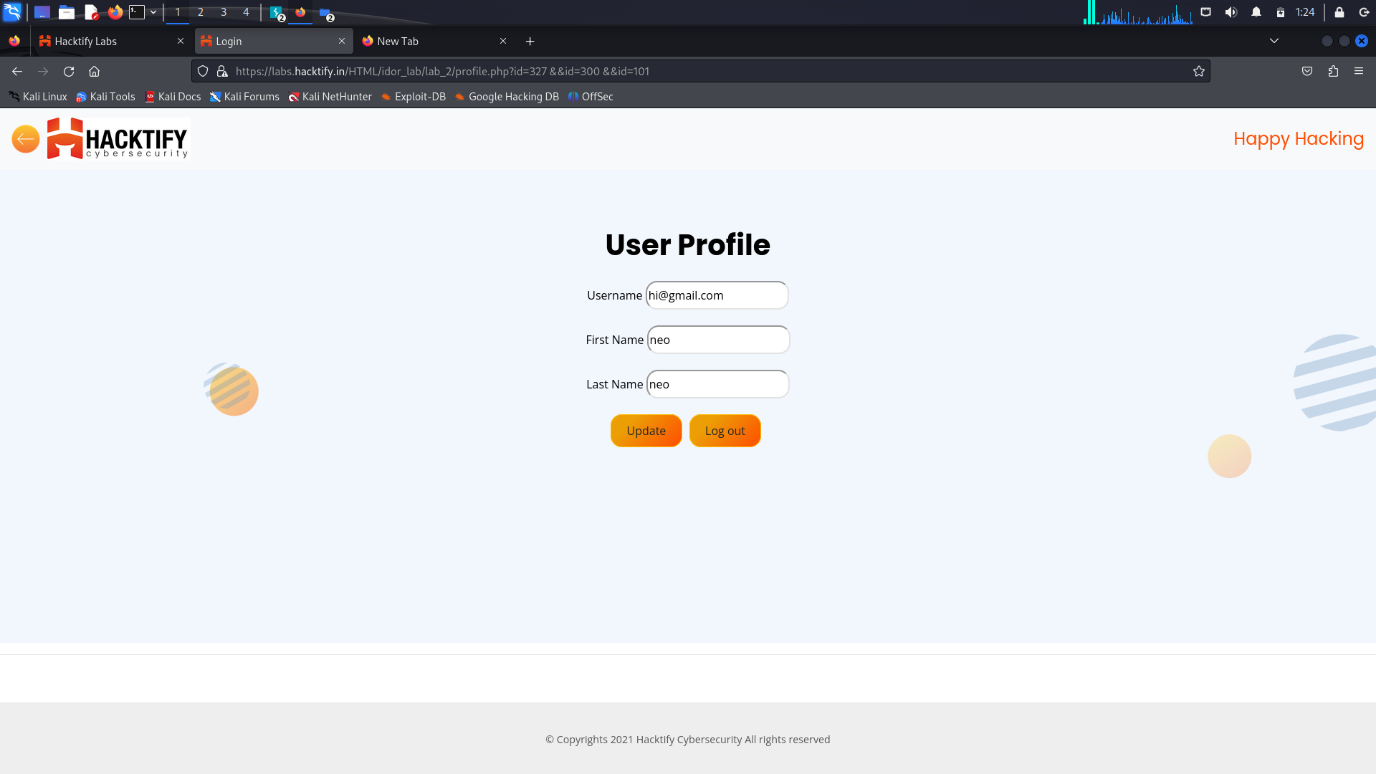
# Proof of Concept:

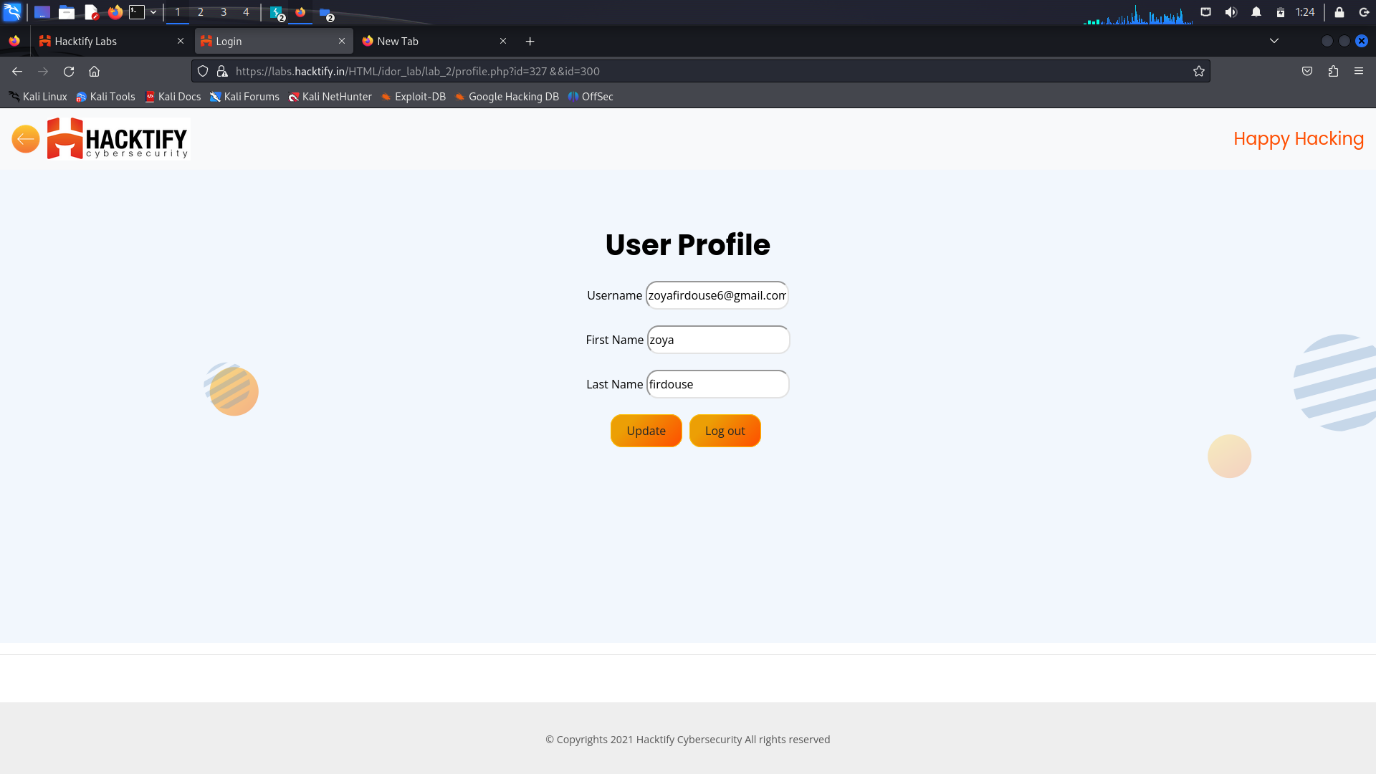
First of all account was made with username [abc@gmail.com](mailto:abc@gmail.com)

Then the same credentials were used to login

After that the id in the url were changed manually and randomly to get access to different accounts in the server.







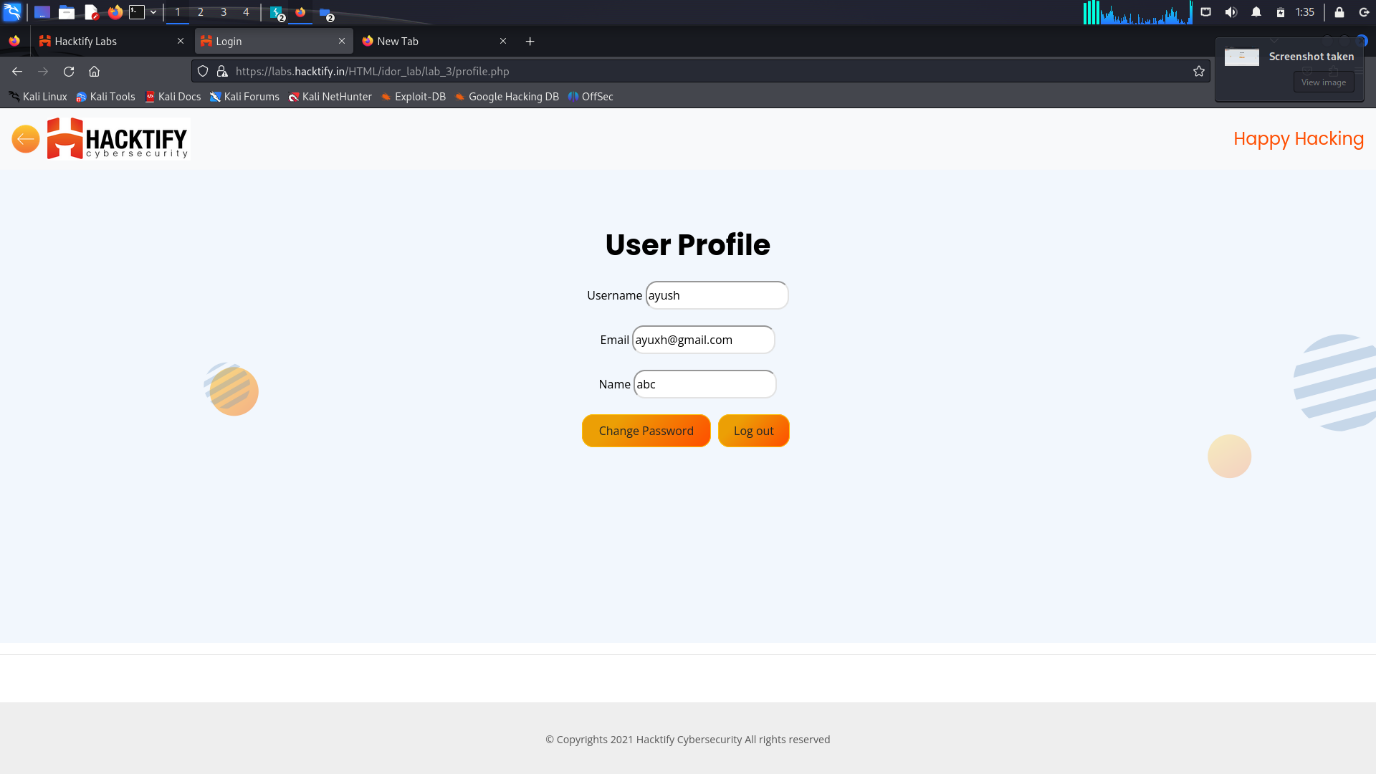
# 2.3. {someone change my password!}

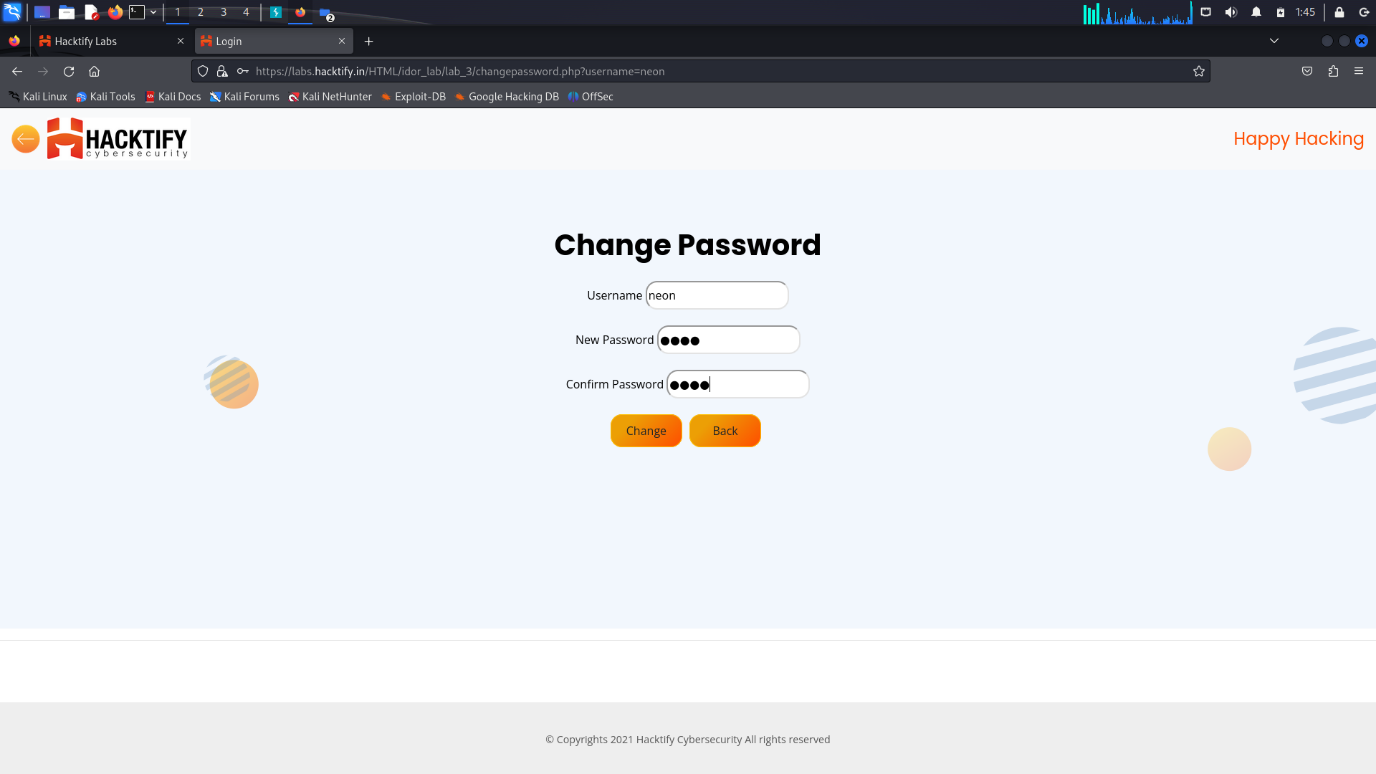
|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {Someone change my password!} | **High** |
| **Tools Used** | |
| Payload , burpsuite | |
| **Vulnerability Description** | |
| An insecure direct object reference is an access control vulnerability where invalidated user input can be used for 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. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/idor\_lab/lab\_3/changepassword.php?username=bc | |
| **Consequences of not Fixing the Issue** | |
| * Exposure of confidential information * Authentication bypass * Alteration of data * Account takeover | |
| **Suggested Countermeasures** | |
| * Indirect reference maps * Fuzz testing * Parameter verification * Access validation | |
| **References** | |
| <https://portswigger.net/web-security/access-control/idor>  <https://www.geeksforgeeks.org/insecure-direct-object-reference-idor-vulnerability/>  <https://www.eccouncil.org/cybersecurity-exchange/web-application-hacking/idor-vulnerability-detection-prevention/> | |

# Proof of Concept

The first steps were same.

The only difference is that with the idor method we can also use html tags like ‘&&’ to change the user and can update the credentials of the other user without their permission.





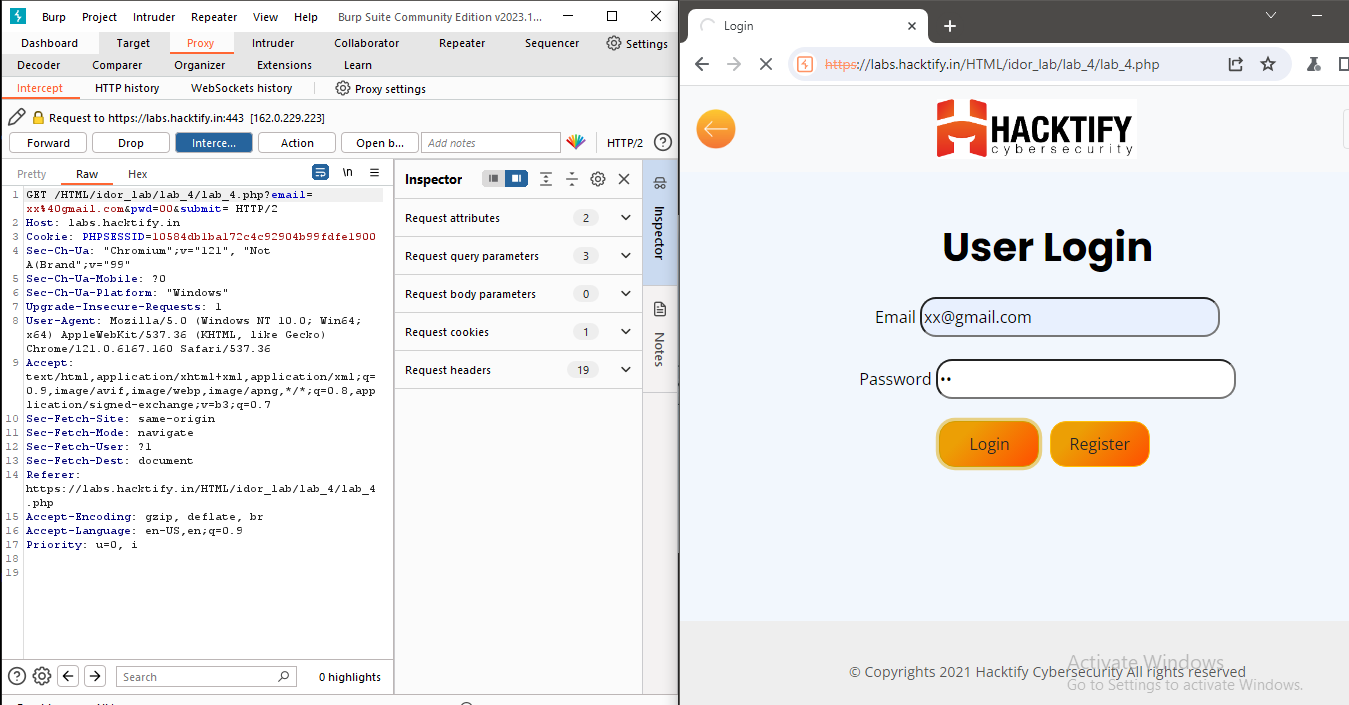
# 2.4. {Change your Methods!}

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| {change your methods!} | **Medium** |
| **Tools Used** | |
| burpsuite | |
| **Vulnerability Description** | |
| An insecure direct object reference 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. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://labs.hacktify.in/HTML/idor\_lab/lab\_4/profile.php?id=237 | |
| **Consequences of not Fixing the Issue** | |
| * Exposure of confidential information * Authentication bypass * Alteration of data * Account takeover | |
| **Suggested Countermeasures** | |
| * Indirect reference maps * Fuzz testing * Parameter verification * Access validation | |
| **References** | |
| <https://portswigger.net/web-security/access-control/idor>  <https://www.geeksforgeeks.org/insecure-direct-object-reference-idor-vulnerability/>  <https://www.eccouncil.org/cybersecurity-exchange/web-application-hacking/idor-vulnerability-detection-prevention/> | |

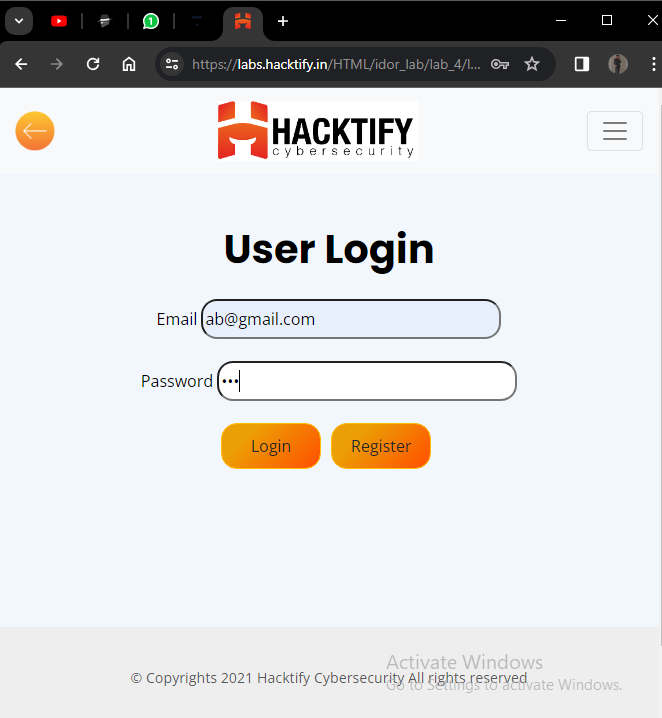
# Proof of Concept

**Firstly I have created an account with email:** [**XX@gmail.com**](mailto:XX@gmail.com) **& pass:00 .**

**Then I try to login into account and I successfully login.**



**Second I have created another account with email:** [**ab@gmail.com**](mailto:ab@gmail.com) **& pass:123 .**



intercept the request

