**Penetration Testing Report**

**Full Name: Saurav Pundir  
Program: HCPT**

**Date: 03-03-2025**

**Introduction**

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week 3 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 3 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 Request Forgery, Cross-Origin Resource Sharing** |

**3. Summary**

Outlined is a Black Box Application Security assessment for the **Week 3 Labs**.

**Total number of Sub-labs: 13 Sub-labs**

|  |  |  |
| --- | --- | --- |
| **High** | **Medium** | **Low** |
| **4** | **4** | **5** |

**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**

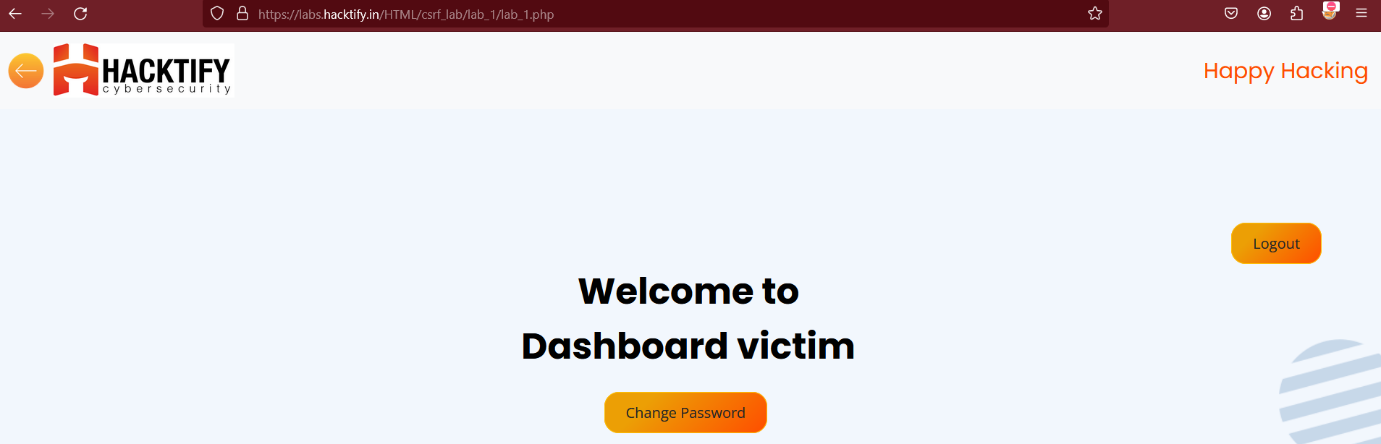
# 1. Cross-Site Request Forgery

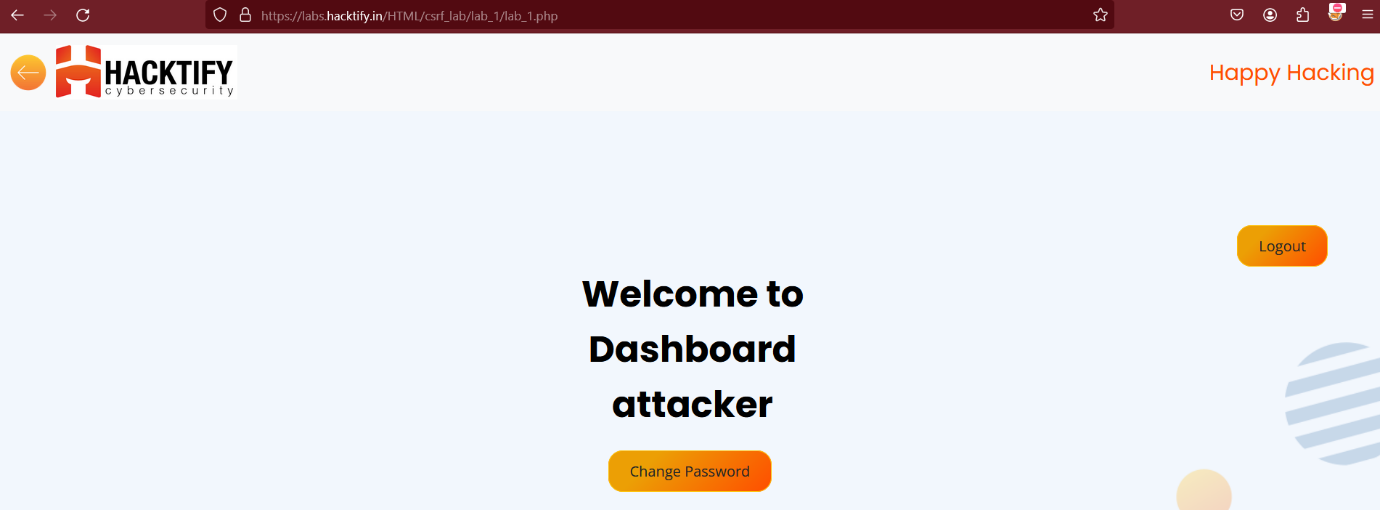
# 1.1. Eassyy CSRF

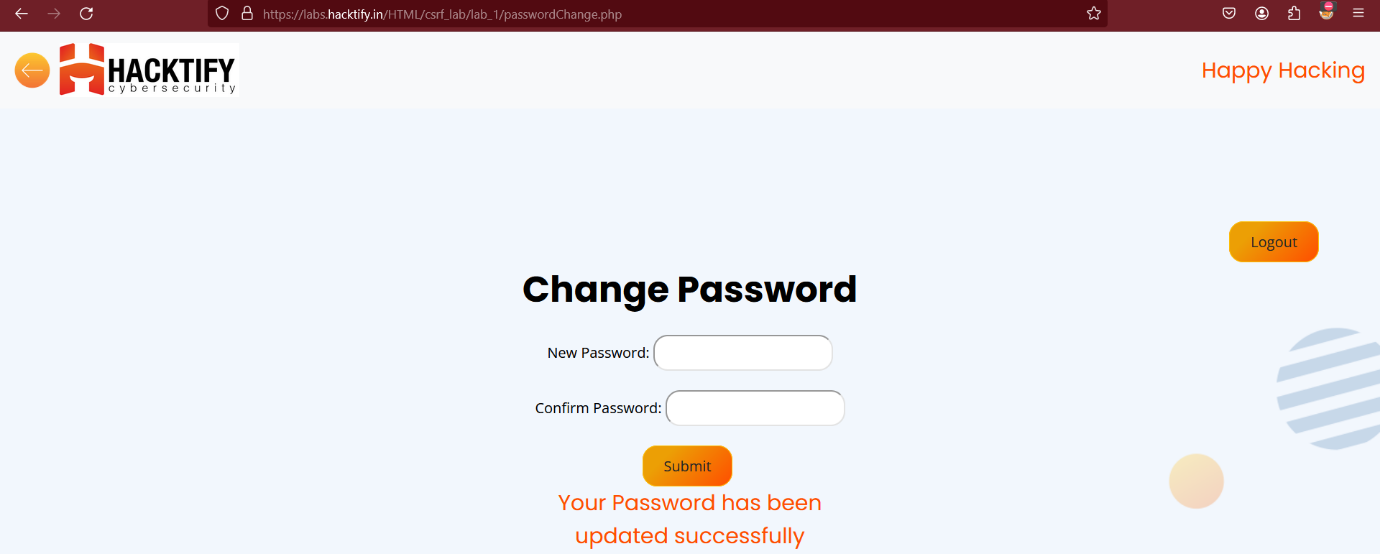
|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| Eassy CSRF | **Low** |
| **Tools Used** | |
| Burp Suite and POC Generator | |
| **Vulnerability Description** | |
| CSRF (Cross-Site Request Forgery) tricks a logged-in user into unknowingly performing unwanted actions on a trusted website, like changing account settings or making transactions, by exploiting their active session. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/csrf_lab/lab_1/login.php> | |
| **Consequences of not Fixing the Issue** | |
| If unpatched, CSRF can allow attackers to perform unauthorized actions on behalf of authenticated users, leading to account takeovers, financial fraud, data modifications, or privilege escalations. This can severely impact user security and trust. | |
| **Suggested Countermeasures** | |
| Implement anti-CSRF tokens in requests, enforce the SameSite cookie attribute to prevent unauthorized cross-site requests, validate request origins using Referrer headers, and restrict sensitive actions to POST methods with proper authentication. | |
| **References** | |
| <https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%5FRequest%5FForgery%5FPrevention%5FCheat%5FSheet.html>  <https://medium.com/@L1lith/cross-site-request-forgery-csrf-cheat-sheet-a4b3d07ed860> | |

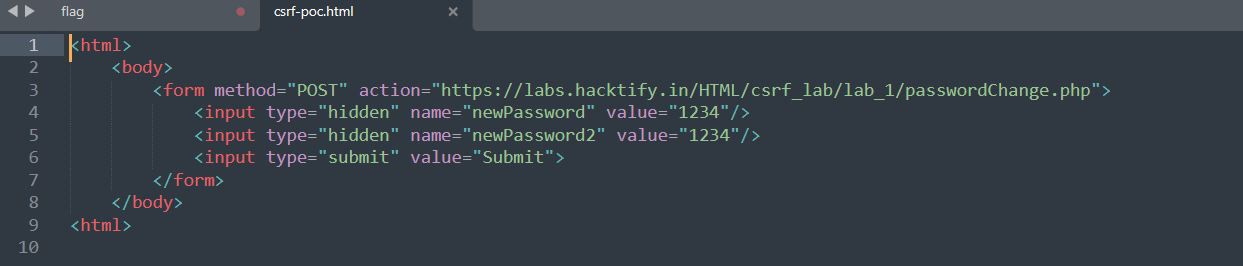
# 

# Proof of Concept



****

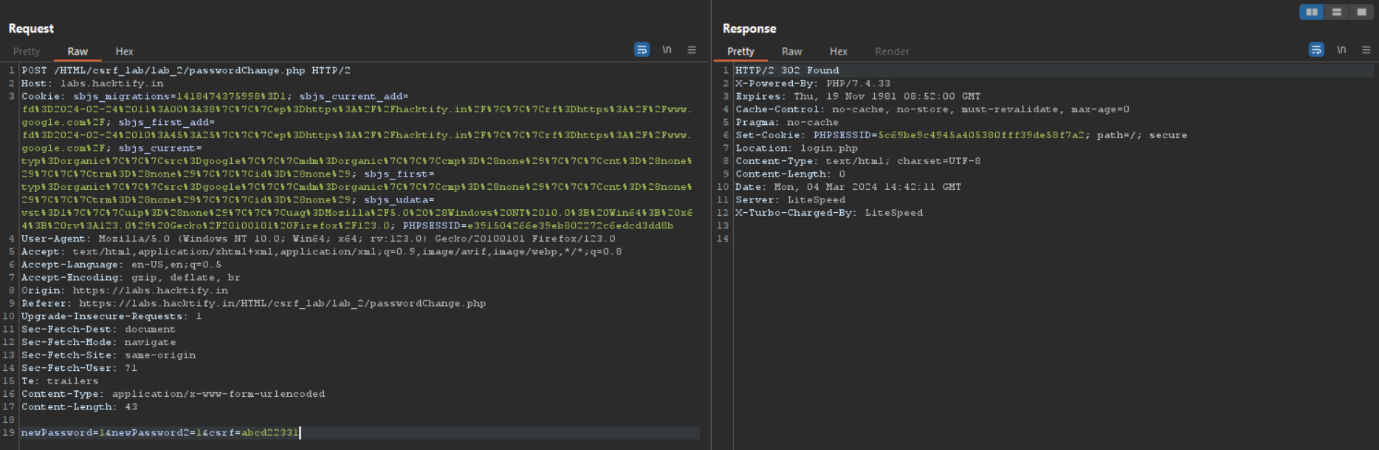
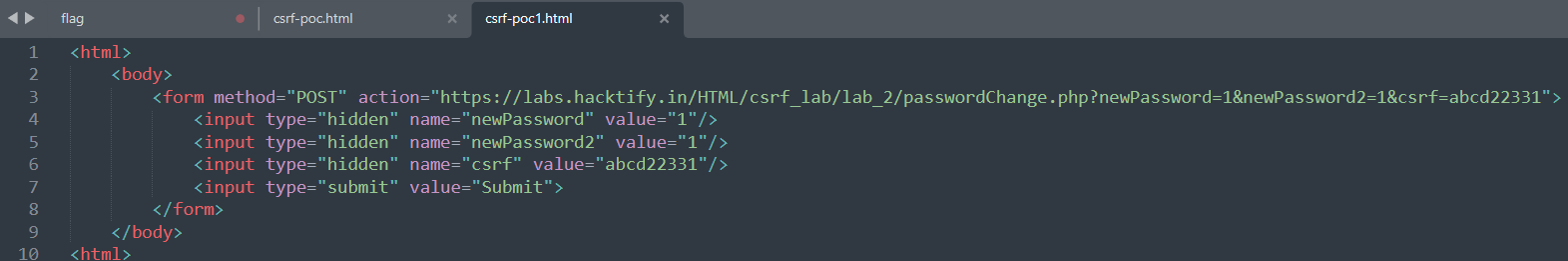
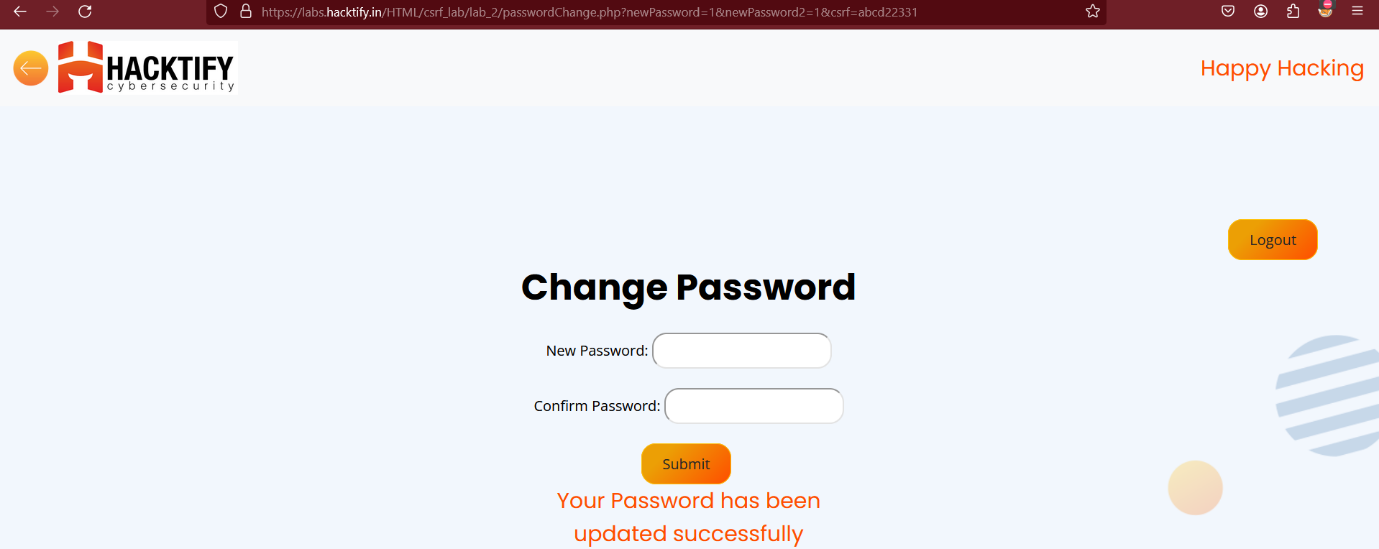
****

****

**1.2.** **Always Validate Tokens**

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| Always Validate Tokens | **Medium** |
| **Tools Used** | |
| Burp Suite and POC Generator | |
| **Vulnerability Description** | |
| CSRF (Cross-Site Request Forgery) tricks a logged-in user into unknowingly performing unwanted actions on a trusted website, like changing account settings or making transactions, by exploiting their active session. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/csrf_lab/lab_2/login.php> | |
| **Consequences of not Fixing the Issue** | |
| If unpatched, CSRF can allow attackers to perform unauthorized actions on behalf of authenticated users, leading to account takeovers, financial fraud, data modifications, or privilege escalations. This can severely impact user security and trust. | |
| **Suggested Countermeasures** | |
| Implement anti-CSRF tokens in requests, enforce the SameSite cookie attribute to prevent unauthorized cross-site requests, validate request origins using Referrer headers, and restrict sensitive actions to POST methods with proper authentication. | |
| **References** | |
| <https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%5FRequest%5FForgery%5FPrevention%5FCheat%5FSheet.html>  <https://medium.com/@L1lith/cross-site-request-forgery-csrf-cheat-sheet-a4b3d07ed860> | |

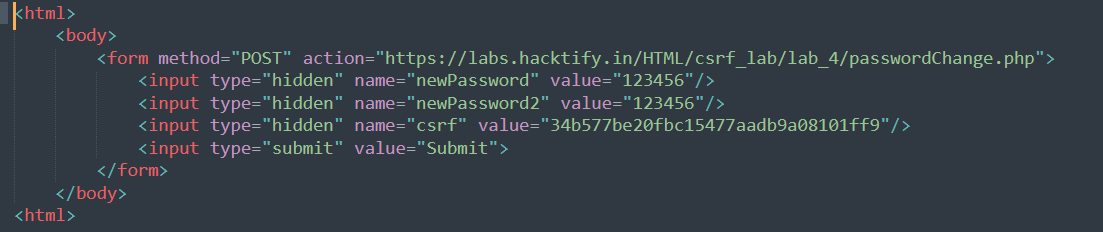
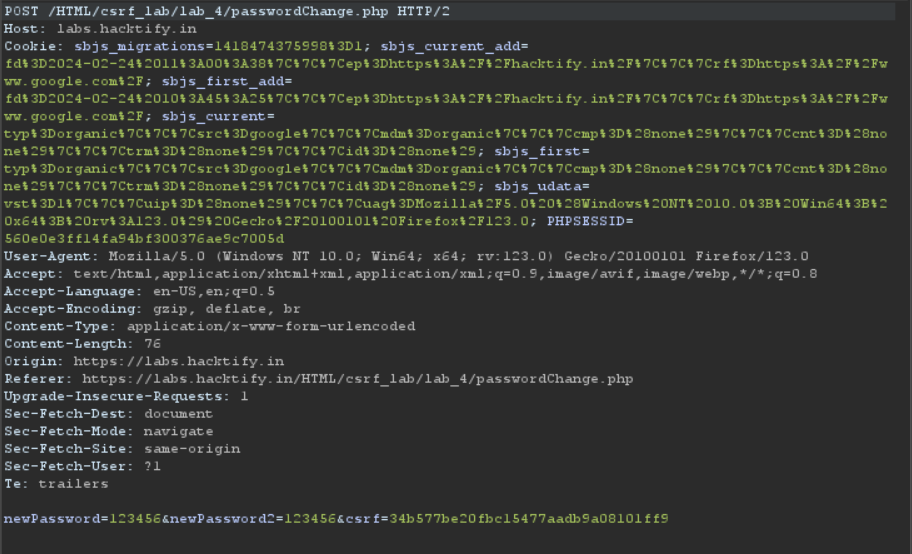
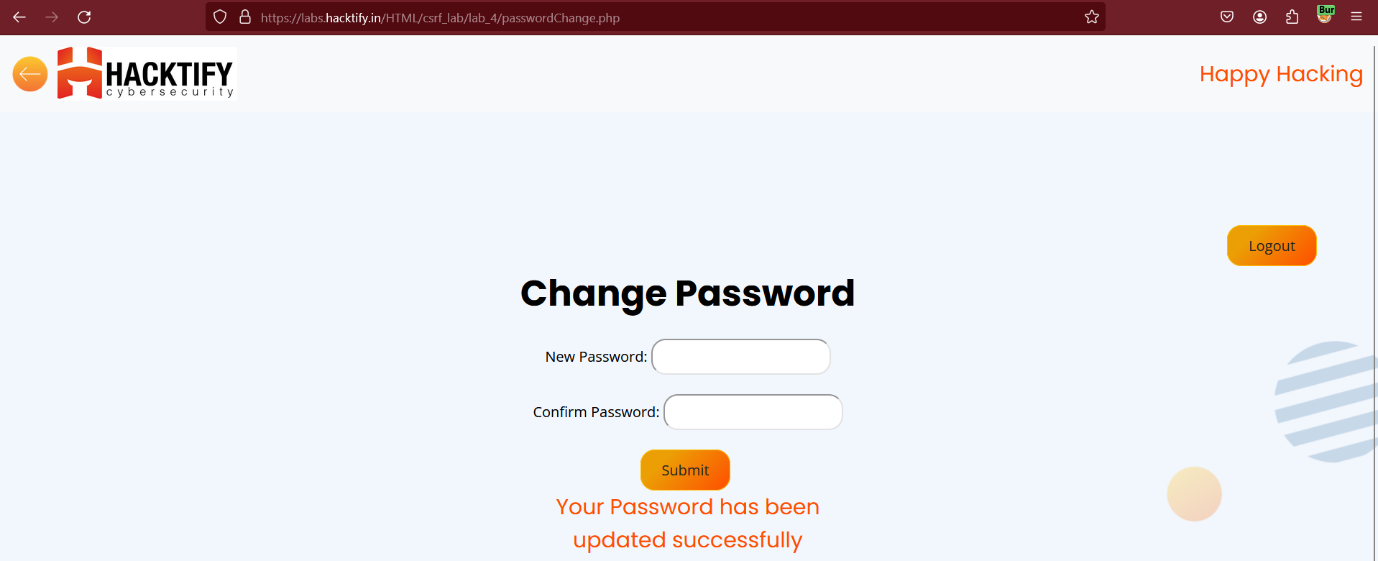
# Proof of Concept



**1.3. I Hate When Someone Uses My Tokens!**

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| I Hate When Someone Uses My Tokens! | **Medium** |
| **Tools Used** | |
| Burp Suite and POC Generator | |
| **Vulnerability Description** | |
| CSRF (Cross-Site Request Forgery) tricks a logged-in user into unknowingly performing unwanted actions on a trusted website, like changing account settings or making transactions, by exploiting their active session. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/csrf_lab/lab_4/login.php> | |
| **Consequences of not Fixing the Issue** | |
| If unpatched, CSRF can allow attackers to perform unauthorized actions on behalf of authenticated users, leading to account takeovers, financial fraud, data modifications, or privilege escalations. This can severely impact user security and trust. | |
| **Suggested Countermeasures** | |
| Implement anti-CSRF tokens in requests, enforce the SameSite cookie attribute to prevent unauthorized cross-site requests, validate request origins using Referrer headers, and restrict sensitive actions to POST methods with proper authentication. | |
| **References** | |
| <https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%5FRequest%5FForgery%5FPrevention%5FCheat%5FSheet.html>  <https://medium.com/@L1lith/cross-site-request-forgery-csrf-cheat-sheet-a4b3d07ed860> | |

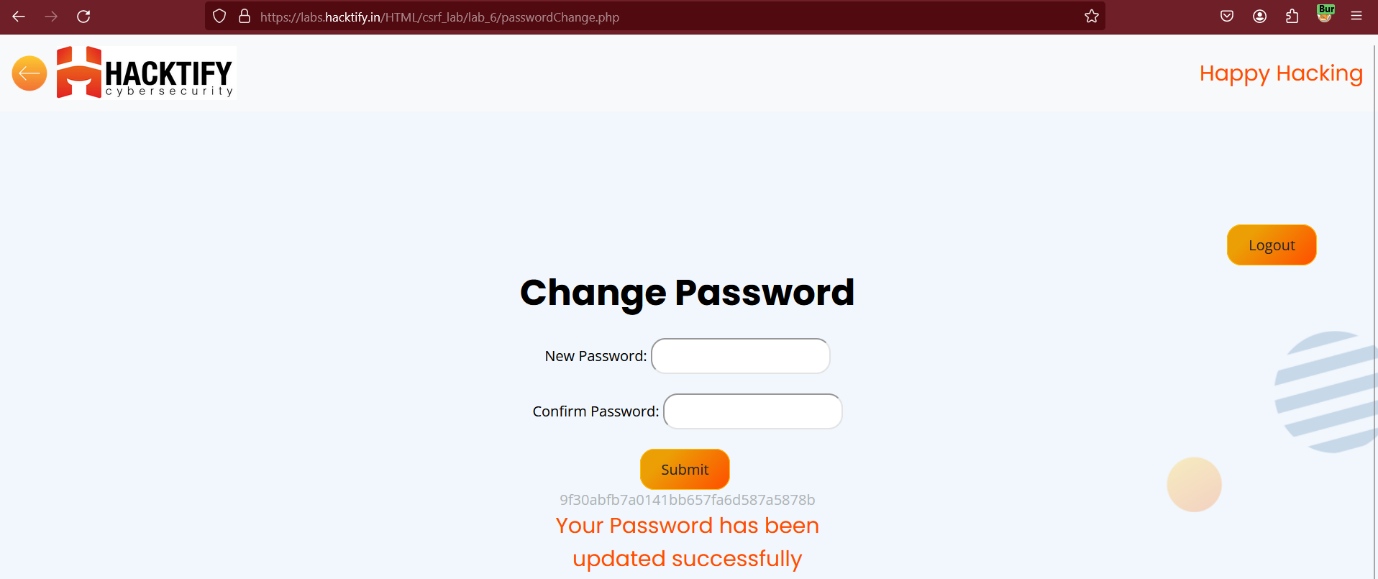
# Proof of Concept

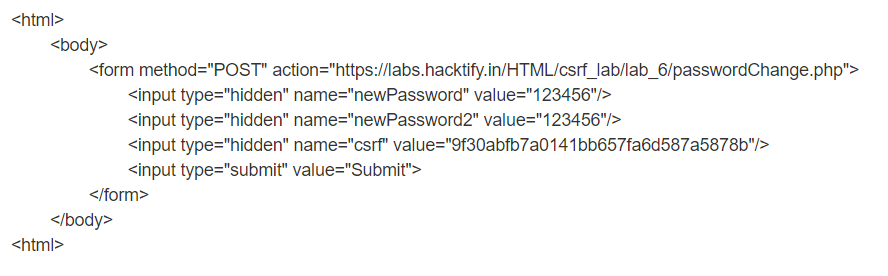


**1.4. GET Me or POST Me**

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| GET Me or POST Me | **Low** |
| **Tools Used** | |
| Burp Suite and POC Generator | |
| **Vulnerability Description** | |
| CSRF (Cross-Site Request Forgery) tricks a logged-in user into unknowingly performing unwanted actions on a trusted website, like changing account settings or making transactions, by exploiting their active session. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/csrf_lab/lab_6/login.php> | |
| **Consequences of not Fixing the Issue** | |
| If unpatched, CSRF can allow attackers to perform unauthorized actions on behalf of authenticated users, leading to account takeovers, financial fraud, data modifications, or privilege escalations. This can severely impact user security and trust. | |
| **Suggested Countermeasures** | |
| Implement anti-CSRF tokens in requests, enforce the SameSite cookie attribute to prevent unauthorized cross-site requests, validate request origins using Referrer headers, and restrict sensitive actions to POST methods with proper authentication. | |
| **References** | |
| <https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%5FRequest%5FForgery%5FPrevention%5FCheat%5FSheet.html>  <https://medium.com/@L1lith/cross-site-request-forgery-csrf-cheat-sheet-a4b3d07ed860> | |

# Proof of Concept

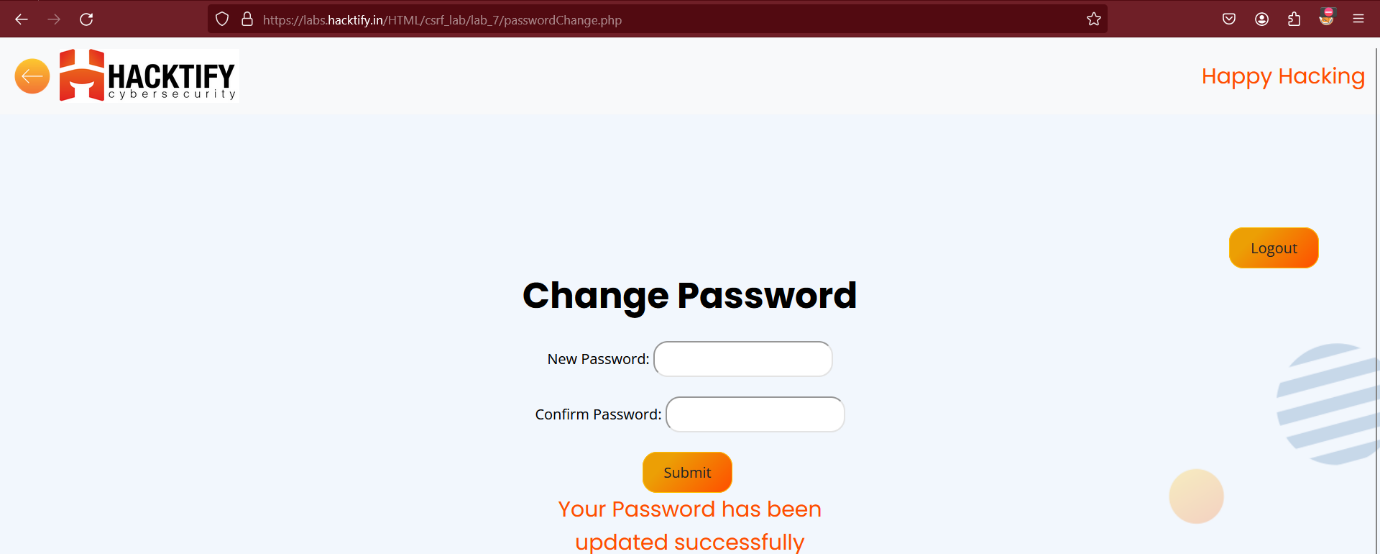
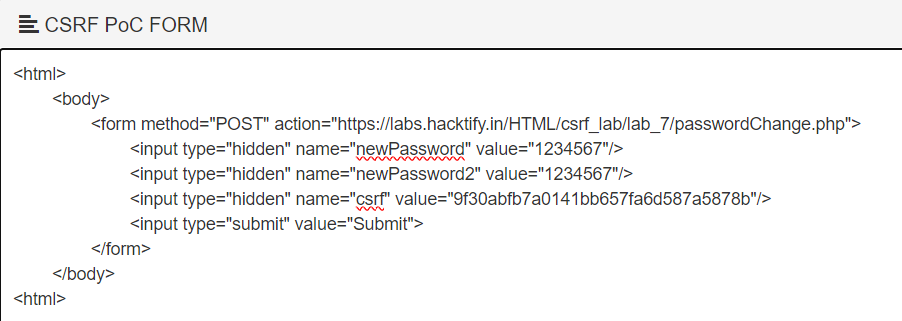
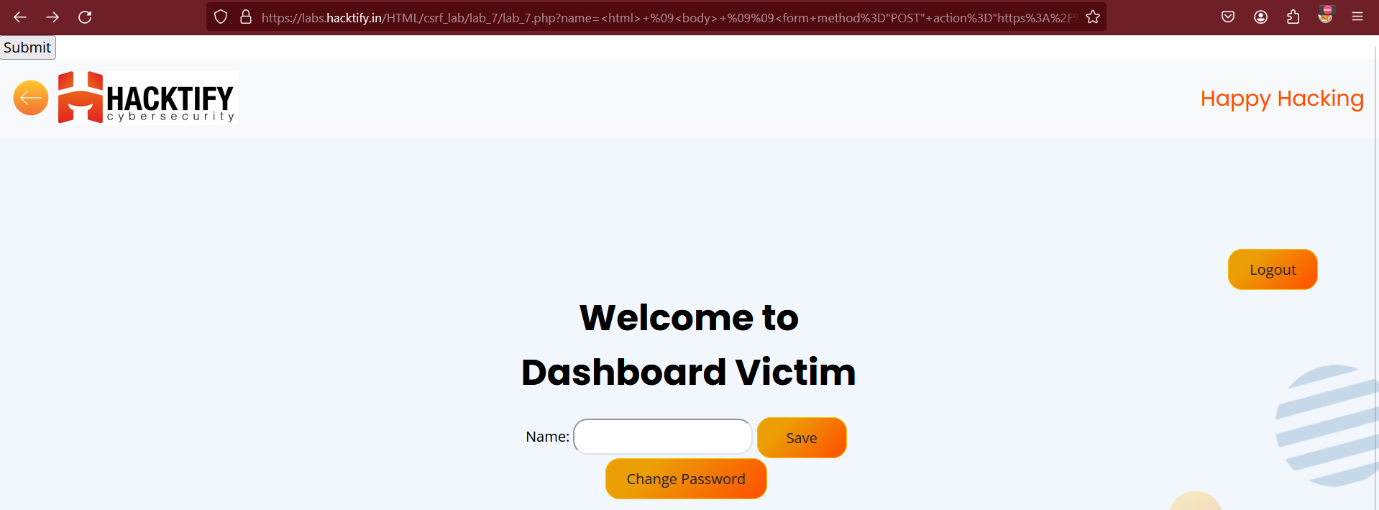




**1.5. XSS The Saviour**

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| XSS The Saviour | **High** |
| **Tools Used** | |
| Burp Suite and POC Generator | |
| **Vulnerability Description** | |
| CSRF (Cross-Site Request Forgery) tricks a logged-in user into unknowingly performing unwanted actions on a trusted website, like changing account settings or making transactions, by exploiting their active session. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/csrf_lab/lab_7/login.php> | |
| **Consequences of not Fixing the Issue** | |
| If unpatched, CSRF can allow attackers to perform unauthorized actions on behalf of authenticated users, leading to account takeovers, financial fraud, data modifications, or privilege escalations. This can severely impact user security and trust. | |
| **Suggested Countermeasures** | |
| Implement anti-CSRF tokens in requests, enforce the SameSite cookie attribute to prevent unauthorized cross-site requests, validate request origins using Referrer headers, and restrict sensitive actions to POST methods with proper authentication. | |
| **References** | |
| <https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%5FRequest%5FForgery%5FPrevention%5FCheat%5FSheet.html>  <https://medium.com/@L1lith/cross-site-request-forgery-csrf-cheat-sheet-a4b3d07ed860> | |

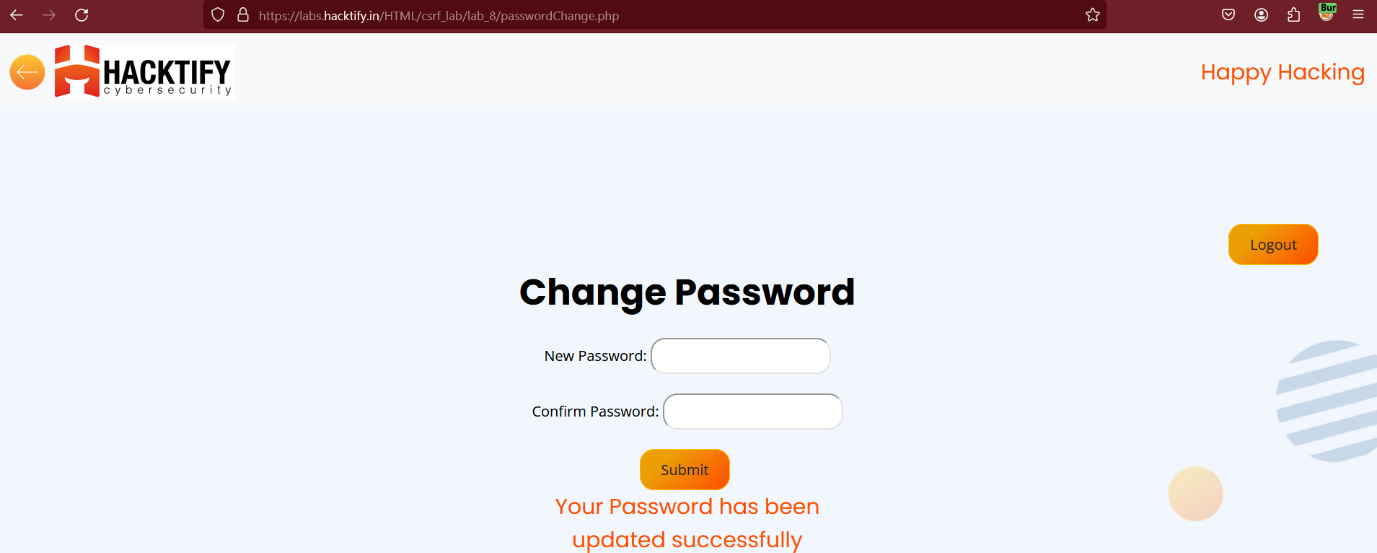
# Proof of Concept

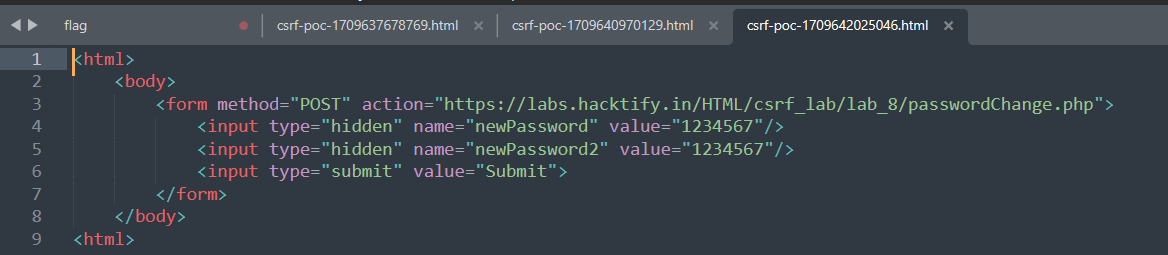


**1.6. Rm -Rf Token**

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| Rm -Rf Token | **High** |
| **Tools Used** | |
| Burp Suite and POC Generator | |
| **Vulnerability Description** | |
| CSRF (Cross-Site Request Forgery) tricks a logged-in user into unknowingly performing unwanted actions on a trusted website, like changing account settings or making transactions, by exploiting their active session. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/csrf_lab/lab_8/login.php> | |
| **Consequences of not Fixing the Issue** | |
| If unpatched, CSRF can allow attackers to perform unauthorized actions on behalf of authenticated users, leading to account takeovers, financial fraud, data modifications, or privilege escalations. This can severely impact user security and trust. | |
| **Suggested Countermeasures** | |
| Implement anti-CSRF tokens in requests, enforce the SameSite cookie attribute to prevent unauthorized cross-site requests, validate request origins using Referrer headers, and restrict sensitive actions to POST methods with proper authentication. | |
| **References** | |
| <https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%5FRequest%5FForgery%5FPrevention%5FCheat%5FSheet.html>  <https://medium.com/@L1lith/cross-site-request-forgery-csrf-cheat-sheet-a4b3d07ed860> | |

# Proof of Concept





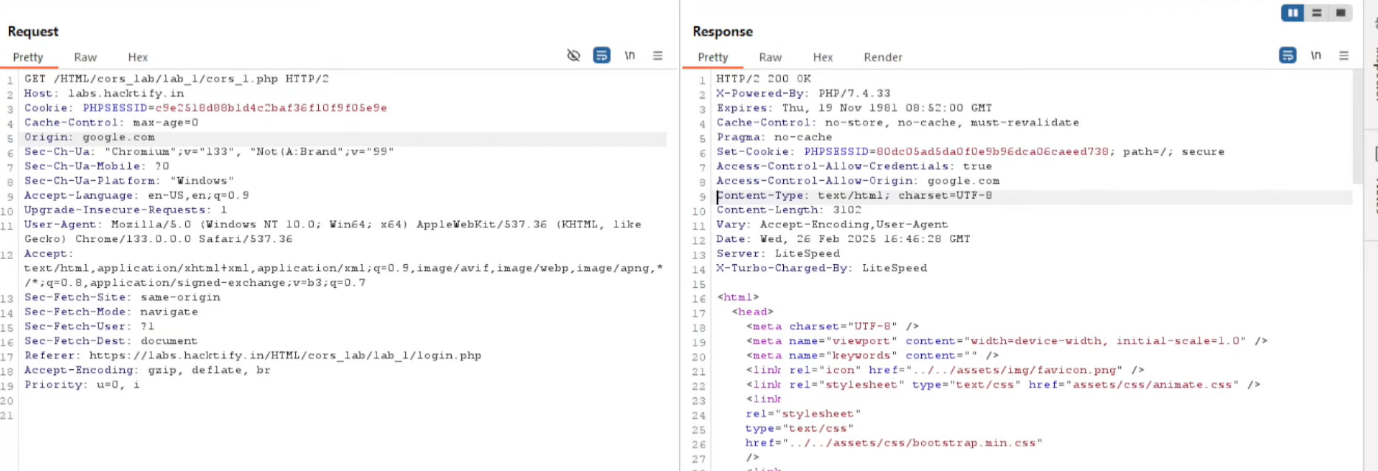
# 2. Cross Origin Resource Sharing

# 2.1. CORS with Arbitrary Origin

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| CORS with Arbitrary Origin | **Low** |
| **Tools Used** | |
| Burpsuite | |
| **Vulnerability Description** | |
| CORS controls cross-origin resource access. If misconfigured (e.g., Access-Control-Allow-Origin: \*), attackers can exploit it to steal sensitive data or perform unauthorized actions on behalf of authenticated users. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/cors_lab/lab_1/login.php> | |
| **Consequences of not Fixing the Issue** | |
| Attackers can exploit these vulnerabilities to steal sensitive data or perform unauthorized actions on behalf of users. | |
| **Suggested Countermeasures** | |
| Configure the Access-Control-Allow-Origin header to specify only trusted domains, avoiding the use of wildcards (\*).  Implement server-side validation to ensure that the Origin header matches expected values before allowing cross-origin requests. | |
| **References** | |
| <https://www.tenable.com/blog/understanding-cross-origin-resource-sharing-vulnerabilities?utm_source=chatgpt.com>  <https://portswigger.net/web-security/cors#what-is-cors-cross-origin-resource-sharing> | |

# 

# Proof of Concept



# 2.2. CORS with Null Origin

|  |  |
| --- | --- |
| **Reference** | **Risk Rating** |
| CORS with Null Origin | **Low** |
| **Tools Used** | |
| Burpsuite | |
| **Vulnerability Description** | |
| CORS controls cross-origin resource access. If misconfigured (e.g., Access-Control-Allow-Origin: \*), attackers can exploit it to steal sensitive data or perform unauthorized actions on behalf of authenticated users. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://labs.hacktify.in/HTML/cors_lab/lab_1/login.php> | |
| **Consequences of not Fixing the Issue** | |
| Attackers can exploit these vulnerabilities to steal sensitive data or perform unauthorized actions on behalf of users. | |
| **Suggested Countermeasures** | |
| Configure the Access-Control-Allow-Origin header to specify only trusted domains, avoiding the use of wildcards (\*).  Implement server-side validation to ensure that the Origin header matches expected values before allowing cross-origin requests. | |
| **References** | |
| <https://www.tenable.com/blog/understanding-cross-origin-resource-sharing-vulnerabilities?utm_source=chatgpt.com>  <https://portswigger.net/web-security/cors#what-is-cors-cross-origin-resource-sharing> | |

# Proof of Concept

