Penetration Testing Report

Full Name: Anumandla Dhanush
Program: HCPT
Date: 08/03/2024

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	labs.hacktify.in - Cross-Site Request Forgery labs.hacktify.in - Server-Side Request Forgery

3. Summary

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

Total number of Sub-labs: 15 Sub-labs

High	Medium	Low
3	10	2

High - **Number of Sub-labs with hard difficulty level**

Medium - **Number of Sub-labs with Medium difficulty level**

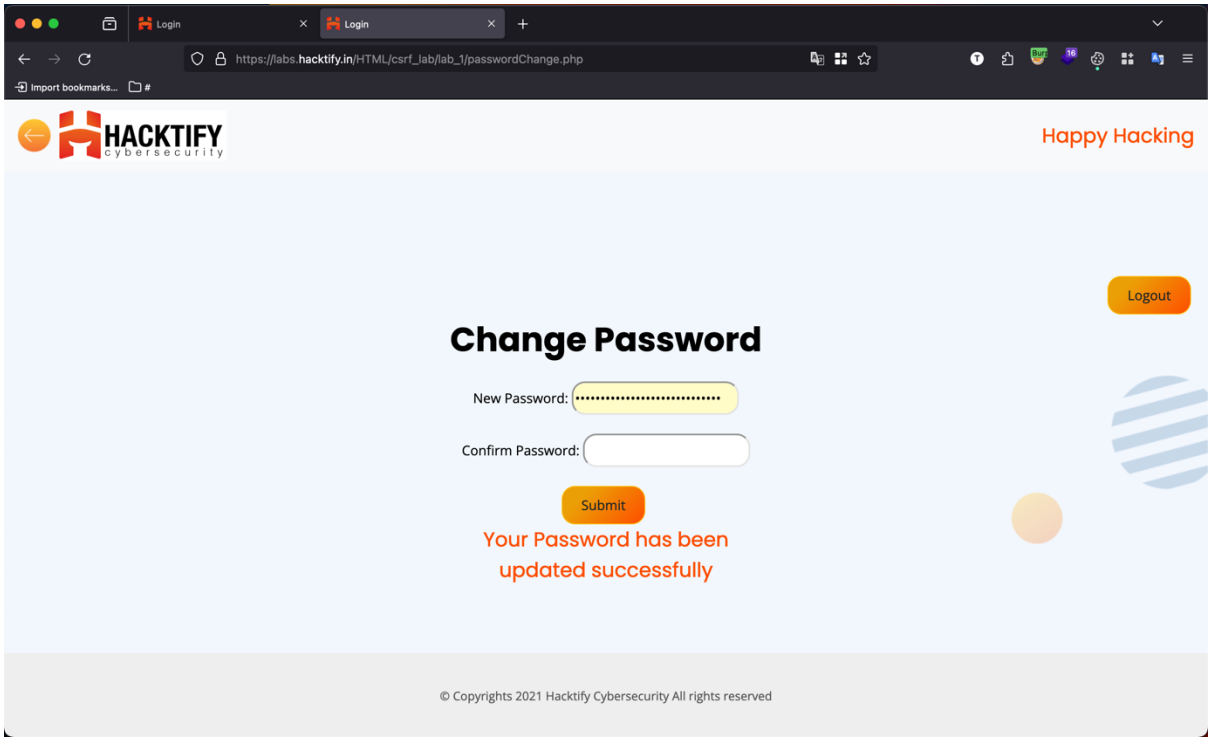
Low - **Number of Sub-labs with Easy difficulty level**

1. Cross-Site Request Forgery

1.1. Eassyy CSRF

Reference	Risk Rating
Eassyy CSRF	**Medium**
Tools Used	
Web Browser, BurpSuite	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user's browser into making unauthorized requests on a trusted site, potentially leading to unintended actions or data manipulation on behalf of the user without their consent. It often exploits the trust a website has in a user's browser by forcing them to perform actions without their explicit approval.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/csrf_lab/lab_1/login.php 2. Create 2 accounts (say victim and attacker) 3. Login to attacker account, Click on “Change Password” button. 4. Enter the new password and capture the request in BurpSuite. 5. Copy the request and paste it on https://hacktify.in/hacktify-csrf-poc-generator/ 6. Copy the CSRF PoC HTML and save it with .html extension. 7. Now, log into victim account. Open the html file in browser. 8. The file gets loaded and the password of victim account gets changed.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_1/passwordChange.php	
Consequences of not Fixing the Issue	
Failure to fix Cross-Site Request Forgery (CSRF) vulnerabilities can result in unauthorized actions performed on behalf of users, leading to account hijacking, data manipulation, and potential financial or reputational damage to both users and the affected website.	
Suggested Countermeasures	
1. Implement anti-CSRF tokens: Introduce unique, unpredictable tokens in web forms to validate legitimate requests, making it challenging for attackers to forge malicious requests. 2. Utilize SameSite cookie attribute: Set the SameSite attribute on cookies to restrict cross-site requests, reducing the risk of CSRF attacks by preventing unauthorized access to sensitive user data.	
References	
1. https://owasp.org/www-community/attacks/csrf 2. https://portswigger.net/web-security/csrf	

Proof of Concept

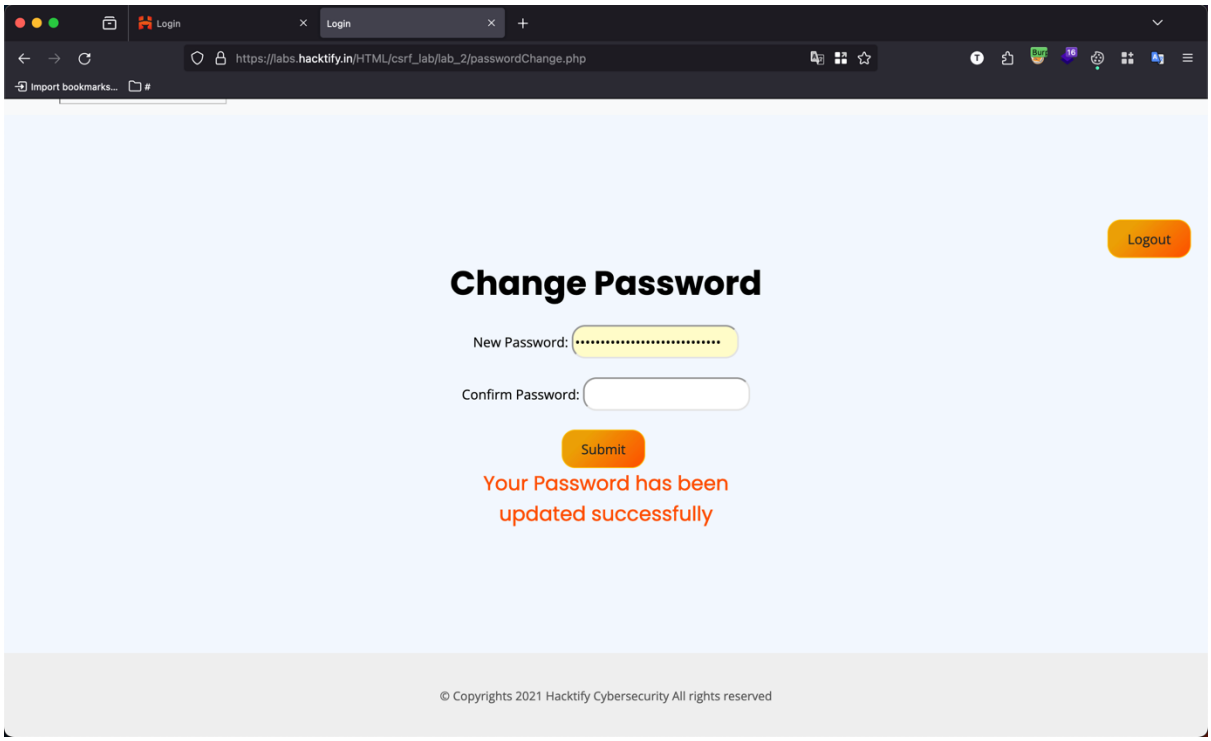


1.2. Always Validate Tokens

Reference	Risk Rating
Always Validate Tokens	**Medium**
Tools Used	
Web Browser, BurpSuite	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user's browser into making unauthorized requests on a trusted site, potentially leading to unintended actions or data manipulation on behalf of the user without their consent. It often exploits the trust a website has in a user's browser by forcing them to perform actions without their explicit approval.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/csrf_lab/lab_2/login.php 2. Create 2 accounts (say victim and attacker) 3. Login to attacker account, Click on “Change Password” button. 4. Enter the new password and capture the request in BurpSuite. 5. Copy the request and paste it on https://hacktify.in/hacktify-csrf-poc-generator/ 6. Copy the CSRF PoC HTML and save it with .html extension. 7. Now, log into victim account. Open the html file in browser. 8. The file gets loaded and the password of victim account gets changed.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_2/passwordChange.php	
Consequences of not Fixing the Issue	
Failure to fix Cross-Site Request Forgery (CSRF) vulnerabilities can result in unauthorized actions performed on behalf of users, leading to account hijacking, data manipulation, and potential financial or reputational damage to both users and the affected website.	

Suggested Countermeasures
1. Implement anti-CSRF tokens: Introduce unique, unpredictable tokens in web forms to validate legitimate requests, making it challenging for attackers to forge malicious requests. 2. Utilize SameSite cookie attribute: Set the SameSite attribute on cookies to restrict cross-site requests, reducing the risk of CSRF attacks by preventing unauthorized access to sensitive user data.
References
1. https://owasp.org/www-community/attacks/csrf 2. https://portswigger.net/web-security/csrf

Proof of Concept

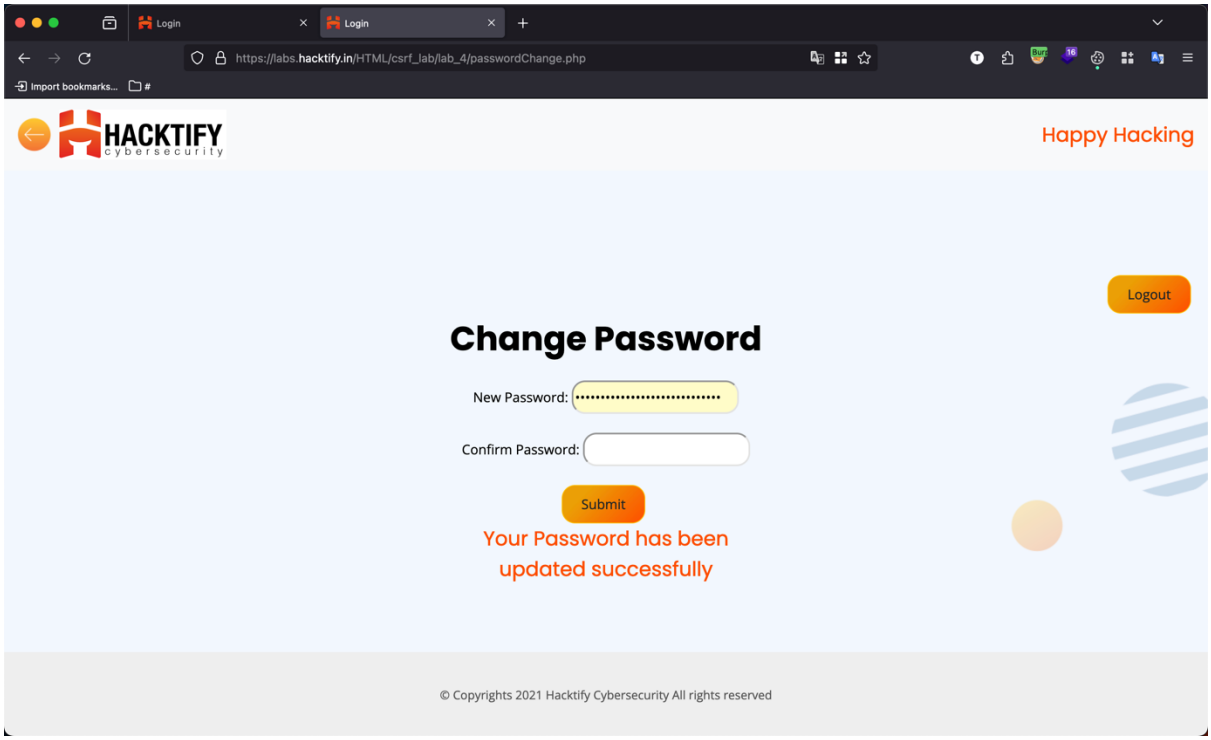


1.3. I hate when someone uses my tokens!

Reference	Risk Rating
I hate when someone uses my tokens!	**Medium**
Tools Used	
Web Browser, BurpSuite	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user's browser into making unauthorized requests on a trusted site, potentially leading to unintended actions or data manipulation on behalf of the user without their consent. It often exploits the trust a website has in a user's browser by forcing them to perform actions without their explicit approval.	
How It Was Discovered	

Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/csrf_lab/lab_4/login.php 2. Create 2 accounts (say victim and attacker) 3. Login to attacker account, Click on “Change Password” button. 4. Enter the new password and capture the request in BurpSuite. 5. Copy the request and paste it on https://hacktify.in/hacktify-csrf-poc-generator/ 6. Copy the CSRF PoC HTML and save it with .html extension. 7. Now, log into victim account. Open the html file in browser. 8. The file gets loaded and the password of victim account gets changed.
Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_4/passwordChange.php
Consequences of not Fixing the Issue
Failure to fix Cross-Site Request Forgery (CSRF) vulnerabilities can result in unauthorized actions performed on behalf of users, leading to account hijacking, data manipulation, and potential financial or reputational damage to both users and the affected website.
Suggested Countermeasures
1. Implement anti-CSRF tokens: Introduce unique, unpredictable tokens in web forms to validate legitimate requests, making it challenging for attackers to forge malicious requests. 2. Utilize SameSite cookie attribute: Set the SameSite attribute on cookies to restrict cross-site requests, reducing the risk of CSRF attacks by preventing unauthorized access to sensitive user data.
References
1. https://owasp.org/www-community/attacks/csrf 2. https://portswigger.net/web-security/csrf

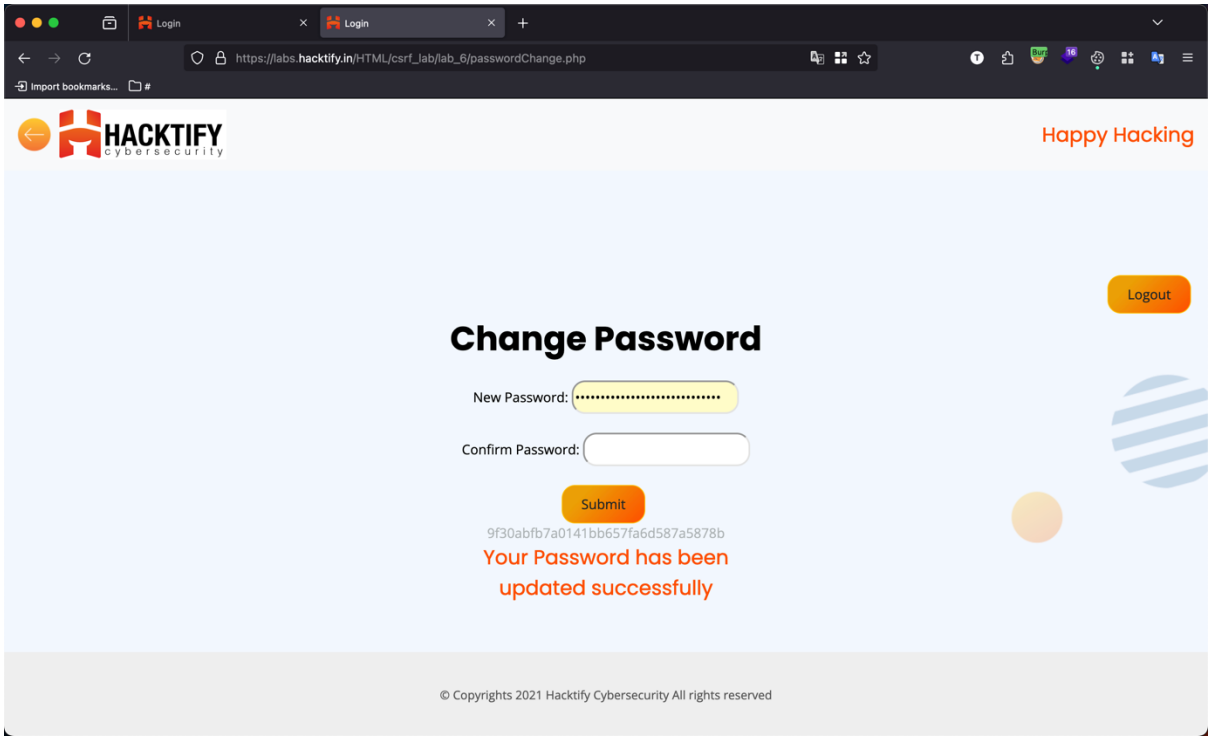
Proof of Concept



1.4. GET Me or POST ME

Reference	Risk Rating
GET Me or POST ME	**Medium**
Tools Used	
Web Browser, BurpSuite	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user's browser into making unauthorized requests on a trusted site, potentially leading to unintended actions or data manipulation on behalf of the user without their consent. It often exploits the trust a website has in a user's browser by forcing them to perform actions without their explicit approval.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/csrf_lab/lab_6/login.php 2. Create 2 accounts (say victim and attacker) 3. Login to attacker account, Click on “Change Password” button. 4. Enter the new password and capture the request in BurpSuite. 5. Copy the request and paste it on https://hacktify.in/hacktify-csrf-poc-generator/ 6. Copy the CSRF PoC HTML and save it with .html extension. 7. Now, log into victim account. Open the html file in browser. 8. The file gets loaded and the password of victim account gets changed.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_6/passwordChange.php	
Consequences of not Fixing the Issue	
Failure to fix Cross-Site Request Forgery (CSRF) vulnerabilities can result in unauthorized actions performed on behalf of users, leading to account hijacking, data manipulation, and potential financial or reputational damage to both users and the affected website.	
Suggested Countermeasures	
1. Implement anti-CSRF tokens: Introduce unique, unpredictable tokens in web forms to validate legitimate requests, making it challenging for attackers to forge malicious requests. 2. Utilize SameSite cookie attribute: Set the SameSite attribute on cookies to restrict cross-site requests, reducing the risk of CSRF attacks by preventing unauthorized access to sensitive user data.	
References	
1. https://owasp.org/www-community/attacks/csrf 2. https://portswigger.net/web-security/csrf	

Proof of Concept

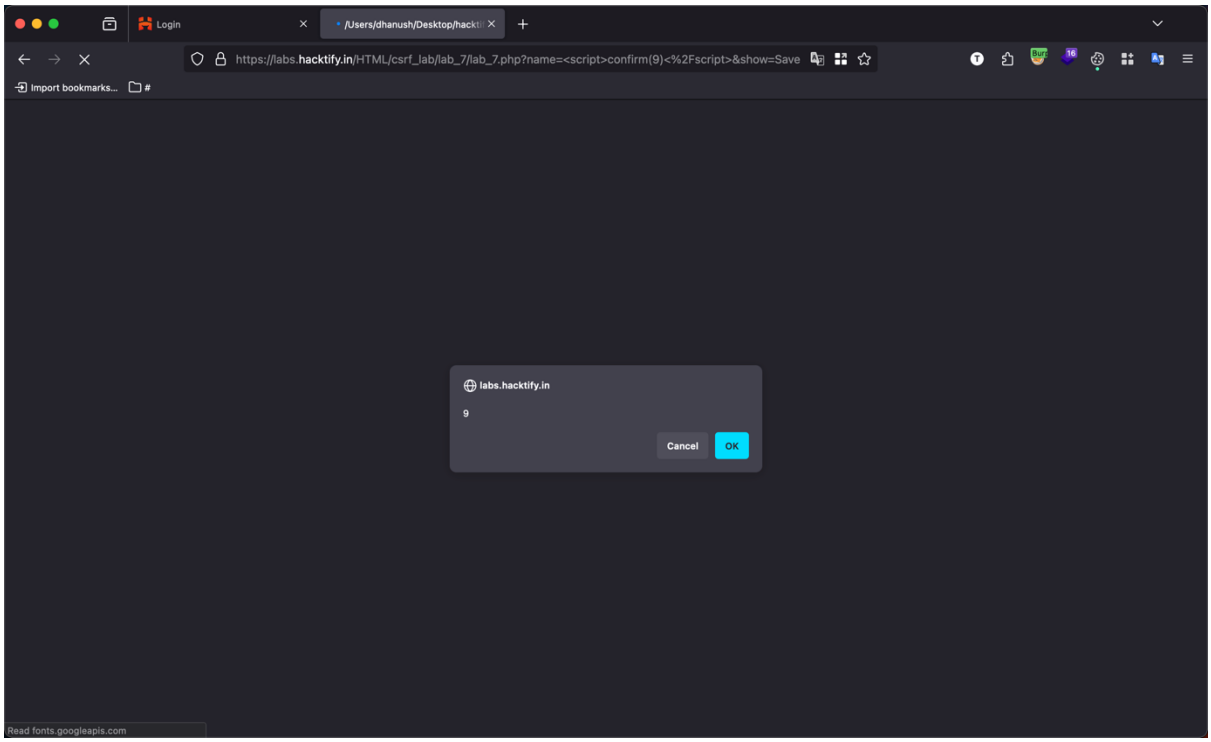


1.5. XSS the saviour

Reference	Risk Rating
XSS the saviour	**Low**
Tools Used	
Web Browser, BurpSuite	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user's browser into making unauthorized requests on a trusted site, potentially leading to unintended actions or data manipulation on behalf of the user without their consent. It often exploits the trust a website has in a user's browser by forcing them to perform actions without their explicit approval.	
How It Was Discovered	
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/csrf_lab/lab_7/login.php 2. Create 2 accounts (say victim and attacker) 3. Login to attacker account, add the XSS payload: <script>confirm(9)</script> 4. Capture the request in BurpSuite. 5. Copy the request and paste it on https://hacktify.in/hacktify-csrf-poc-generator/ 6. Copy the CSRF PoC HTML and save it with .html extension. 7. Now, log into victim account. Open the html file in browser. 8. The file gets loaded and the XSS payload gets executed.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_7/lab_7.php?name=%3Cscript%3Econfirm%289%29%3C%2Fscript%3E&show=Save	
Consequences of not Fixing the Issue	

Failure to fix Cross-Site Request Forgery (CSRF) vulnerabilities can result in unauthorized actions performed on behalf of users, leading to account hijacking, data manipulation, and potential financial or reputational damage to both users and the affected website.
Suggested Countermeasures
1. Implement anti-CSRF tokens: Introduce unique, unpredictable tokens in web forms to validate legitimate requests, making it challenging for attackers to forge malicious requests. 2. Utilize SameSite cookie attribute: Set the SameSite attribute on cookies to restrict cross-site requests, reducing the risk of CSRF attacks by preventing unauthorized access to sensitive user data.
References
1. https://owasp.org/www-community/attacks/csrf 2. https://portswigger.net/web-security/csrf

Proof of Concept

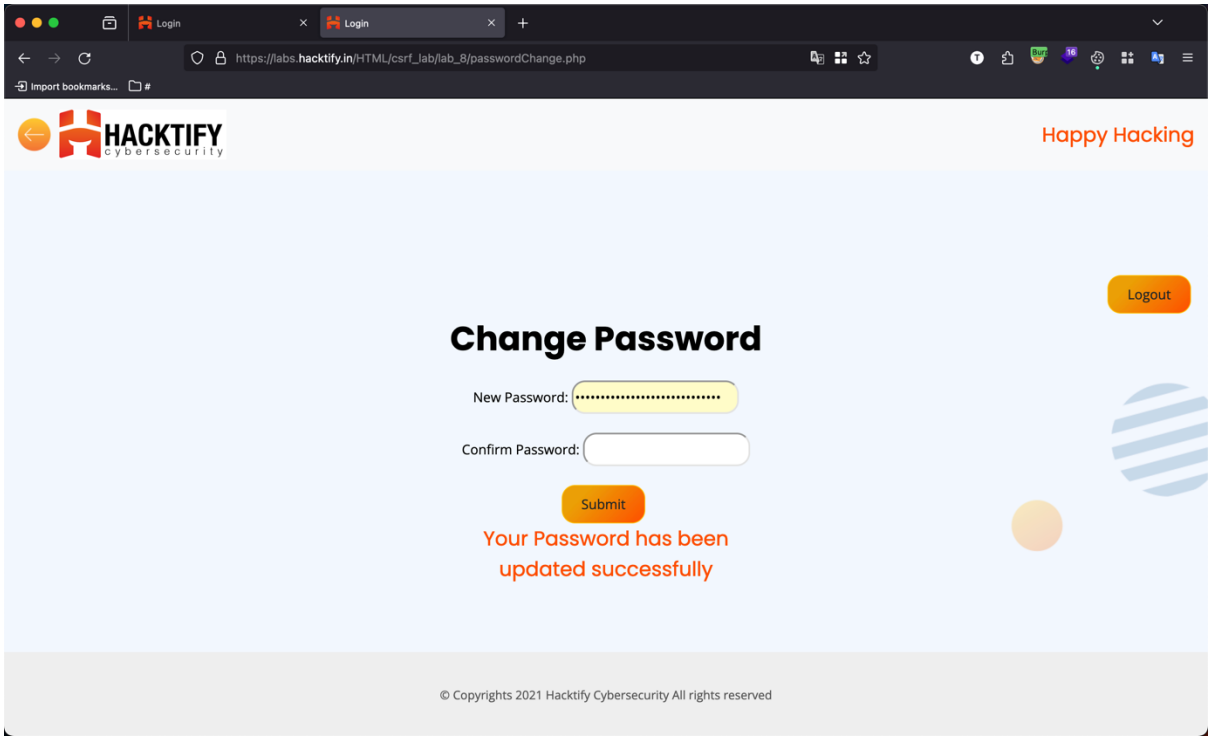


1.6. rm -rf token

Reference	Risk Rating
rm -rf token	**Low**
Tools Used	
Web Browser, BurpSuite	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user's browser into making unauthorized requests on a trusted site, potentially leading to unintended actions	

or data manipulation on behalf of the user without their consent. It often exploits the trust a website has in a user's browser by forcing them to perform actions without their explicit approval.
How It Was Discovered
Manual Analysis: 1. Go to https://labs.hacktify.in/HTML/csrf_lab/lab_8/login.php 2. Create 2 accounts (say victim and attacker) 3. Login to attacker account, Click on “Change Password” button. 4. Enter the new password and capture the request in BurpSuite. 5. Copy the request and paste it on https://hacktify.in/hacktify-csrf-poc-generator/ 6. Copy the CSRF PoC HTML and save it with .html extension. 7. Now, log into victim account. Open the html file in browser. 8. The file gets loaded and the password of victim account gets changed.
Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_8/passwordChange.php
Consequences of not Fixing the Issue
Failure to fix Cross-Site Request Forgery (CSRF) vulnerabilities can result in unauthorized actions performed on behalf of users, leading to account hijacking, data manipulation, and potential financial or reputational damage to both users and the affected website.
Suggested Countermeasures
1. Implement anti-CSRF tokens: Introduce unique, unpredictable tokens in web forms to validate legitimate requests, making it challenging for attackers to forge malicious requests. 2. Utilize SameSite cookie attribute: Set the SameSite attribute on cookies to restrict cross-site requests, reducing the risk of CSRF attacks by preventing unauthorized access to sensitive user data.
References
1. https://owasp.org/www-community/attacks/csrf 2. https://portswigger.net/web-security/csrf

Proof of Concept



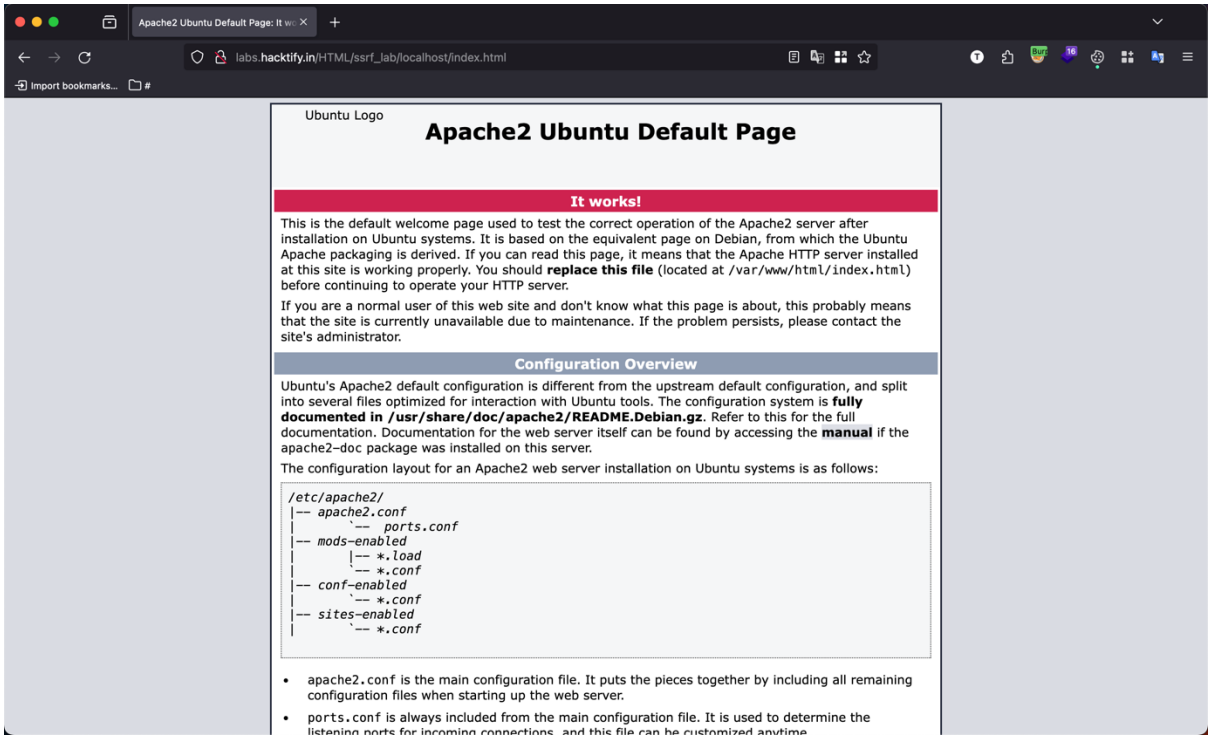
2. Server-Side Request Forgery

2.1. Get The 127.0.0.1

Reference	Risk Rating
Get The 127.0.0.1	**Medium**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_1/lab_1.php 2. There you will see an input field (Enter URL:) 3. Enter payload: 127.0.0.1:80 and click on “submit”. 4. You will be directed to localhost.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/ssrf_lab/lab_1/lab_1.php?url=127.0.0.1%3A80	
Consequences of not Fixing the Issue	
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.	

Suggested Countermeasures
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.
References
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf

Proof of Concept

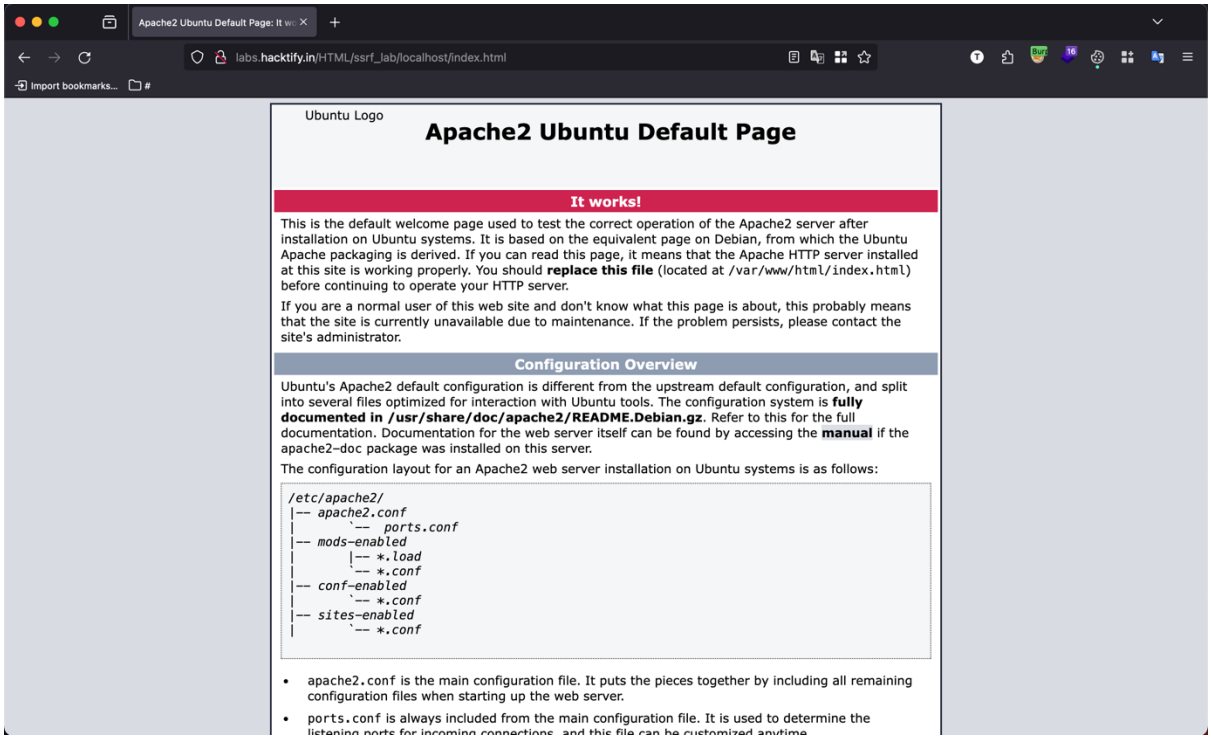


2.2. http(s)? Nevermind!!

Reference	Risk Rating
http(s)? Nevermind!!	**Medium**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	

How It Was Discovered
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_2/lab_2.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://localhost:80 and click on “submit”. 4. You will be directed to localhost.
Vulnerable URLs
http://labs.hacktify.in/HTML/ssrf_lab/lab_2/lab_2.php?url=http%3A%2F%2F127.0.0.1%3A80
Consequences of not Fixing the Issue
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.
Suggested Countermeasures
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.
References
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf

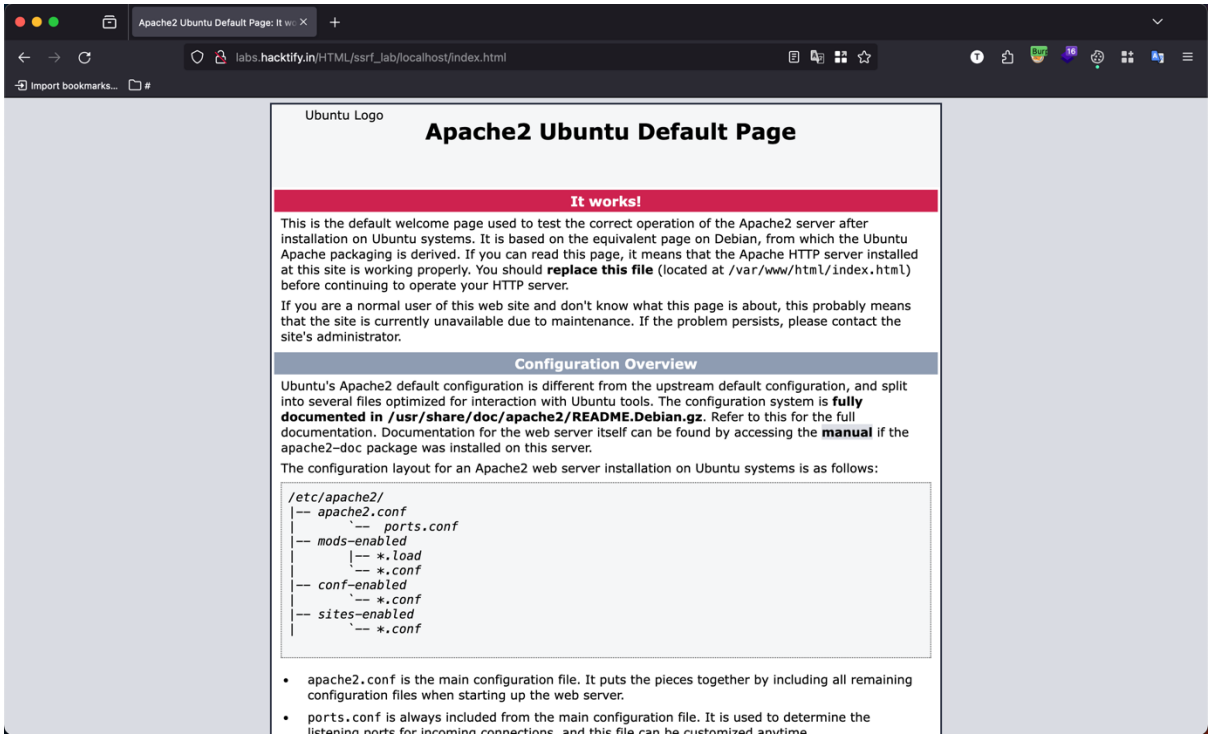
Proof of Concept



2.3. ":" The saviour!

Reference	Risk Rating
":" The saviour!	**Medium**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_3/lab_3.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://[::]:80 and click on “submit”. 4. You will be directed to localhost.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/ssrf_lab/lab_3/lab_3.php	
Consequences of not Fixing the Issue	
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.	
Suggested Countermeasures	
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.	
References	
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf	

Proof of Concept

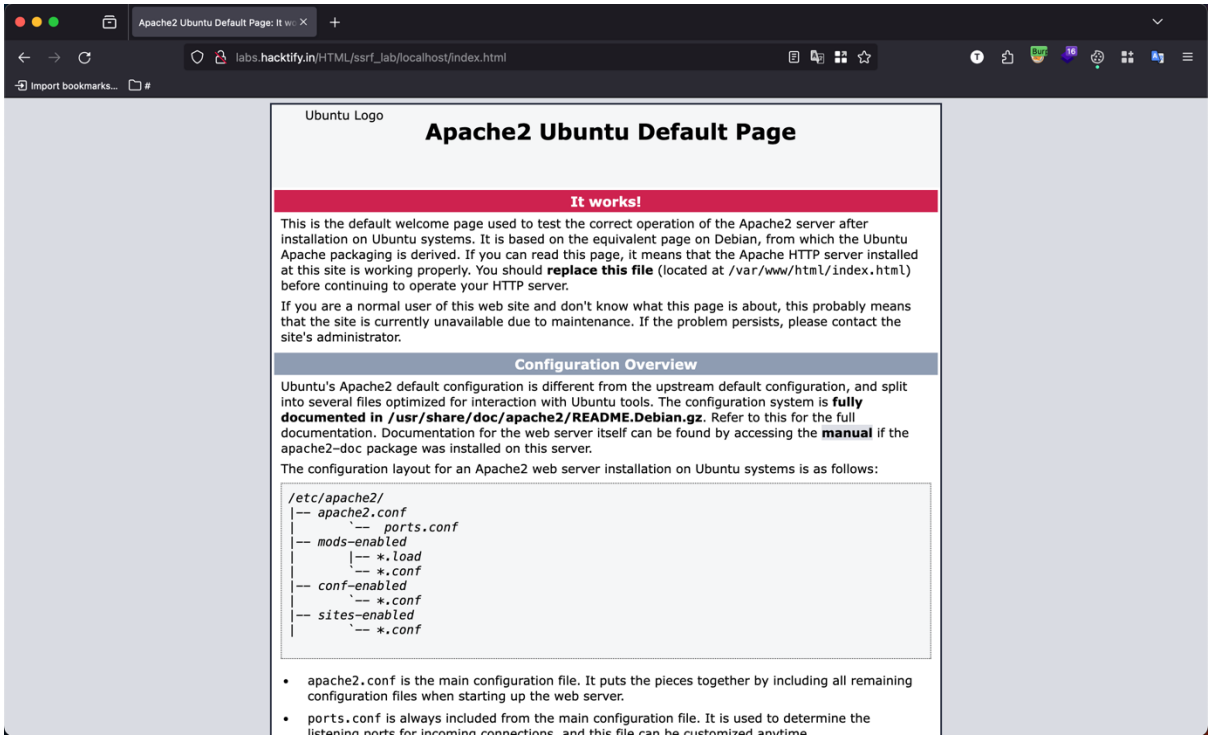


2.4. Messed up Domain!

Reference	Risk Rating
Messed up Domain!	**Medium**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_4/lab_4.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://customer1.app.localhost.my.company.127.0.0.1.nip.io 4. Then click on “submit”. 4. You will be directed to localhost.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/ssrf_lab/lab_4/lab_4.php	
Consequences of not Fixing the Issue	
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.	
Suggested Countermeasures	

1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.
References
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf

Proof of Concept

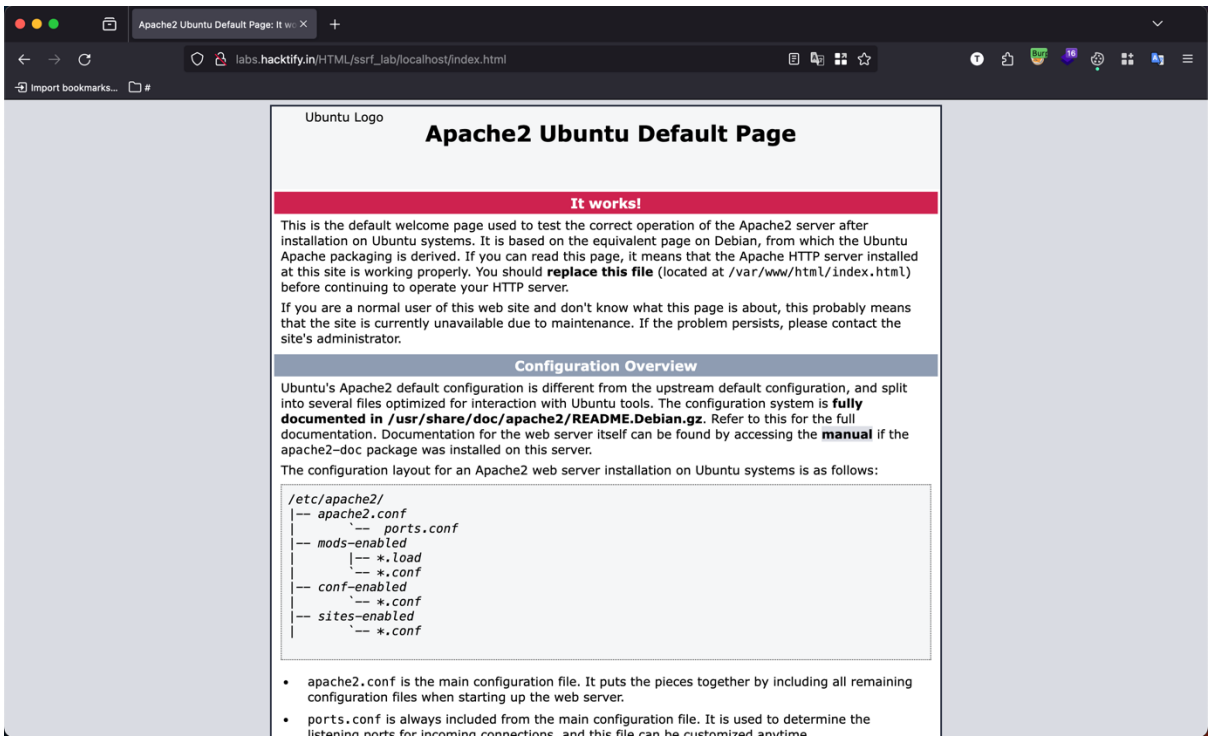


2.5. Decimal IP

Reference	Risk Rating
Decimal IP	**Medium**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	

Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_5/lab_5.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://2130706433 and click on “submit”. 4. You will be directed to localhost.
Vulnerable URLs
http://labs.hacktify.in/HTML/ssrf_lab/lab_5/lab_5.php
Consequences of not Fixing the Issue
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.
Suggested Countermeasures
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.
References
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf

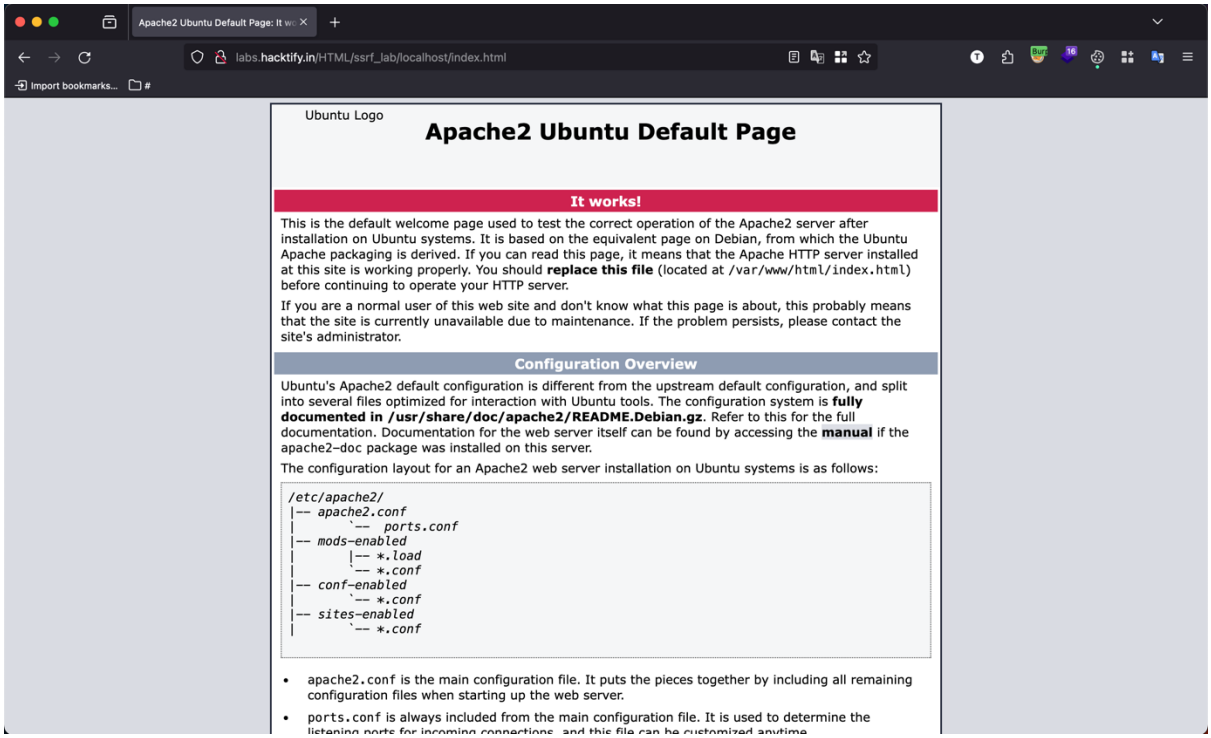
Proof of Concept



2.6. Short-Hand IP address

Reference	Risk Rating
Short-Hand IP address	**Medium**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_6/lab_6.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://127.1 and click on “submit”. 4. You will be directed to localhost.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/ssrf_lab/lab_6/lab_6.php	
Consequences of not Fixing the Issue	
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.	
Suggested Countermeasures	
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.	
References	
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf	

Proof of Concept

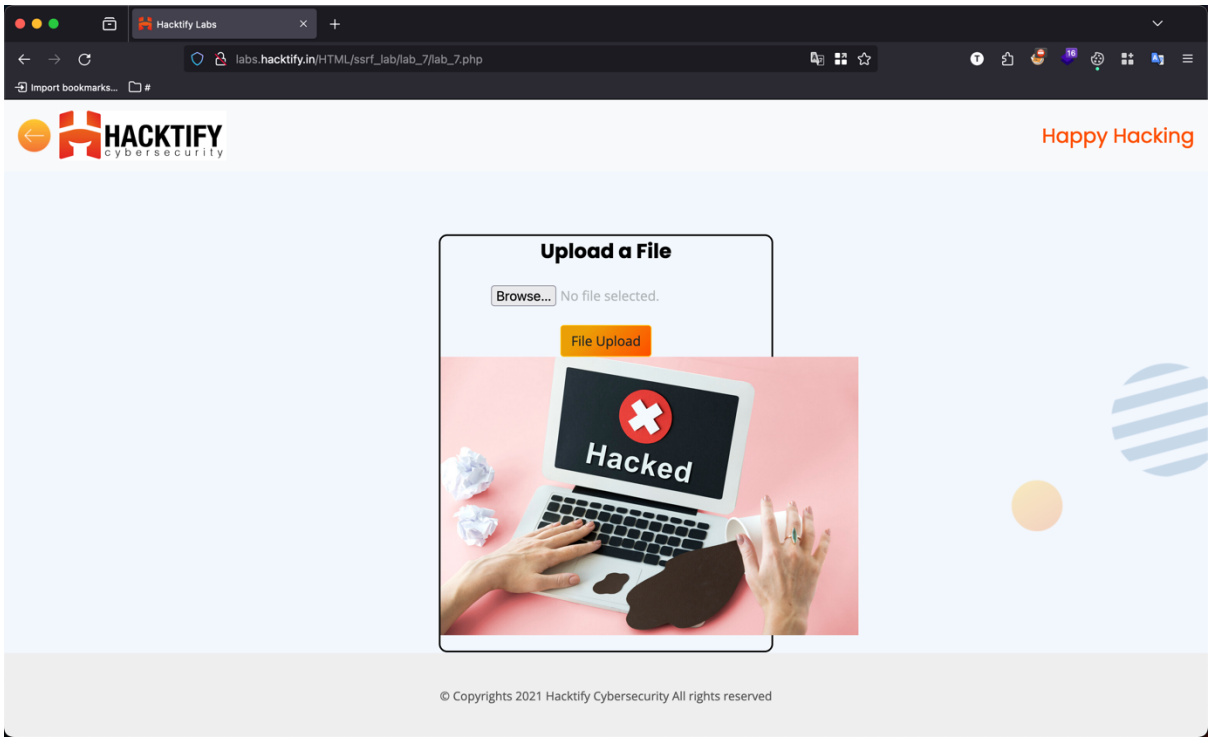


2.7. File Upload to SSRF!

Reference	Risk Rating
File Upload to SSRF!	**High**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_7/lab_7.php 2. There you will see a File Upload functionality. 3. Create a html file with an iframe leading to a picture. 4. Upload the file and click on “File Upload”. 5. The image in the html file gets loaded confirming vulnerability.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/ssrf_lab/lab_7/lab_7.php	
Consequences of not Fixing the Issue	
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.	
Suggested Countermeasures	

1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.
References
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf

Proof of Concept

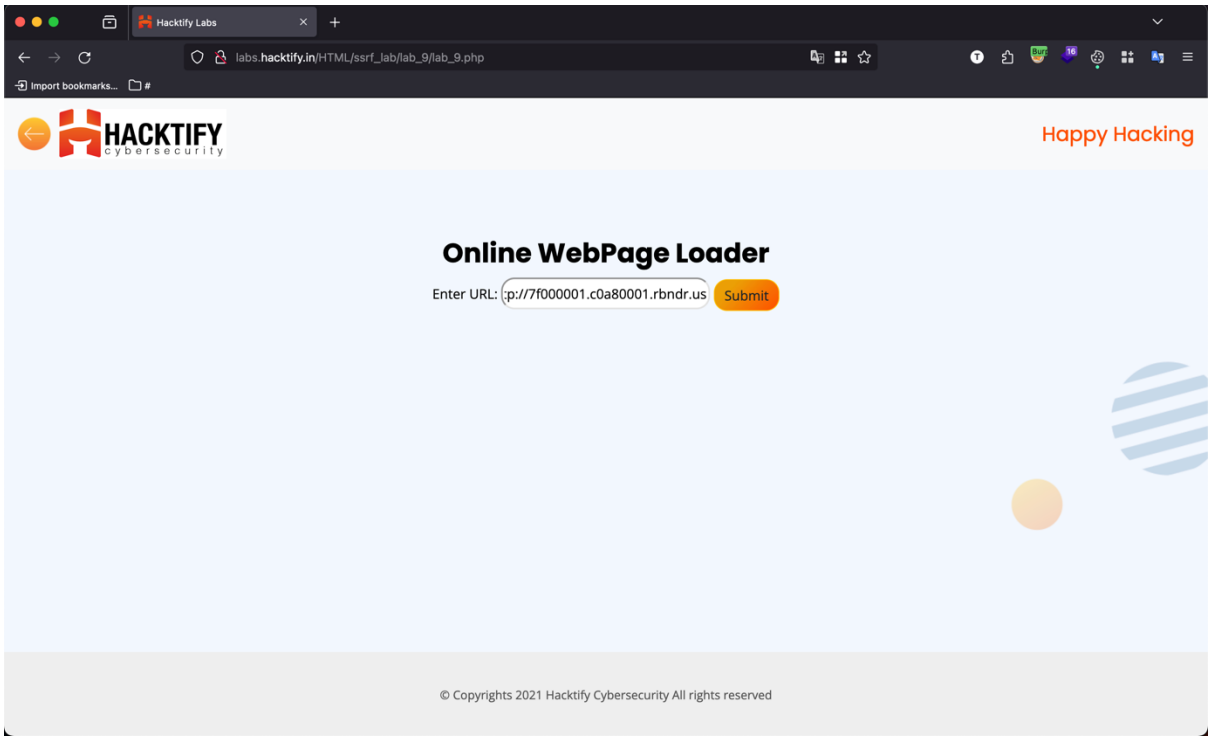


2.8. SSRF with DNS Rebinding

Reference	Risk Rating
SSRF with DNS Rebinding	**High**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	

Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_9/lab_9.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://7f000001.c0a80001.rbndr.us and click on “submit”. 4. You will be directed to localhost.
Vulnerable URLs
http://labs.hacktify.in/HTML/ssrf_lab/lab_9/lab_9.php
Consequences of not Fixing the Issue
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.
Suggested Countermeasures
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.
References
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf

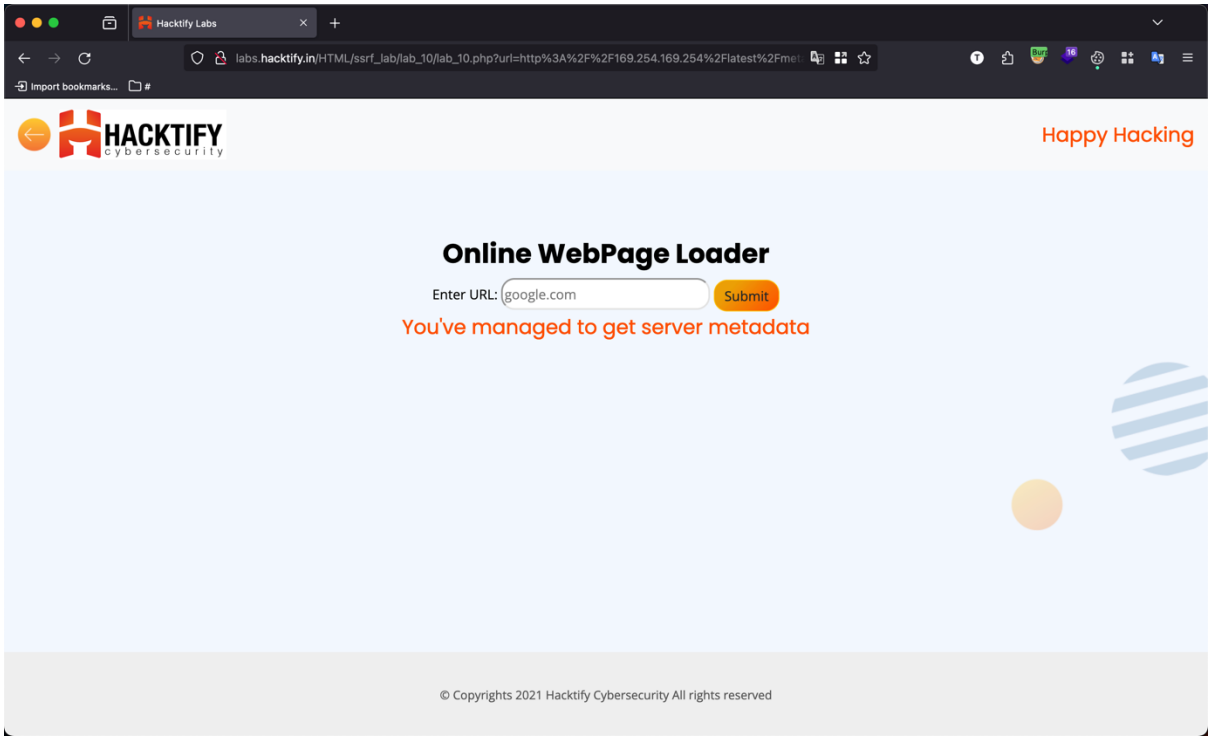
Proof of Concept



2.9. Look an SSRF on Cloud!

Reference	Risk Rating
Look an SSRF on Cloud!	**High**
Tools Used	
Web Browser	
Vulnerability Description	
Server-Side Request Forgery (SSRF) is a web security vulnerability where an attacker can influence or make unauthorized requests to internal resources by manipulating the server's ability to make HTTP requests, potentially leading to data exposure, service disruption, or remote code execution on the server. It occurs when an application allows an attacker to control or influence the server's requests to other domains.	
How It Was Discovered	
Manual Analysis: 1. Go to http://labs.hacktify.in/HTML/ssrf_lab/lab_10/lab_10.php 2. There you will see an input field (Enter URL:) 3. Enter payload: http://169.254.169.254/latest/meta-data/iam/security-credentials/ 4. Click on “submit”. 5. A message is displayed confirming SSRF vulnerability.	
Vulnerable URLs	
http://labs.hacktify.in/HTML/ssrf_lab/lab_10/lab_10.php	
Consequences of not Fixing the Issue	
Failure to address Server-Side Request Forgery (SSRF) vulnerabilities can result in unauthorized access to internal systems, data exposure, and potential exploitation of internal services, leading to severe security breaches, data leaks, and compromise of sensitive information within an organization.	
Suggested Countermeasures	
1. Input validation and whitelisting: Implement thorough input validation to ensure that user-supplied URLs are restricted to a predefined set of allowed domains, preventing attackers from manipulating requests to unauthorized destinations. 2. Network-level controls: Utilize firewalls and network-level access controls to restrict the server's ability to make outbound requests to internal resources or external services, minimizing the risk of unauthorized access and data exposure.	
References	
1. https://owasp.org/www-community/attacks/Server_Side_Request_Forgery 2. https://portswigger.net/web-security/ssrf	

Proof of Concept


Hacktify Labs
labs.hacktify.in/HTML/ssrf_lab/lab_10/lab_10.php?url=http%3A%2F%2F169.254.169.254%2Flatest%2Fmet
Import bookmarks...
HACKTIFY
cybersecurity
Happy Hacking
Online WebPage Loader
Enter URL: google.com
Submit
You've managed to get server metadata
© Copyrights 2021 Hacktify Cybersecurity All rights reserved