Web Penetration Testing Report

Full Name: Geshom Lukoshi
Program: HCPT
Date: 3rd March, 2025

Introduction

This report document hereby describes the proceedings and results of the web site vulnerabilities assessment and conducted against the **Hacktify Cyber Security Internship 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 web site of **Week 3 Labs** and provide a final security assessment report comprising vulnerabilities found and their PoC's, remediation strategies and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

2. Scope

The scope of this penetration testing will cover **(02) labs** by **Hacktify Cyber Security**. The tested attacks are the web application flaws **Cross-Origin Resource Sharing (CORS)** and **Cross-Site Request Forgery (CSRF)**. The testing will include identifying vulnerabilities in these labs, ways in which these flaws can be taken advantage off to exploit the vulnerable web applications. The testing will focus on detecting the **CORS** and **CSRF** vulnerabilities that could lead to unauthorized access and actions on the web applications by unauthorized users **(attackers)**. The injection point entries such as URL parameter, login forms and manipulation of the source code will be assessed to identify potential avenues for malicious code insertion (links). The boundaries of this penetration testing labs exclude the testing of network infrastructures of the web application. The results will be provided with recommendations for mitigation to enhance the web application security posture.

Application Name	Lab 1 – Cross-Origin Resources Sharing (CORS) Lab 2 – Cross-Site Request Forgery (CSRF)

3. Summary

Outlined is the Hacktify web site Security assessment for the **Week 3 Labs**.

Total number of Sub-labs: (12) Sub-labs

High	Medium	Low
5	3	4

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**

Lab 1 Cross-Site Resource Sharing

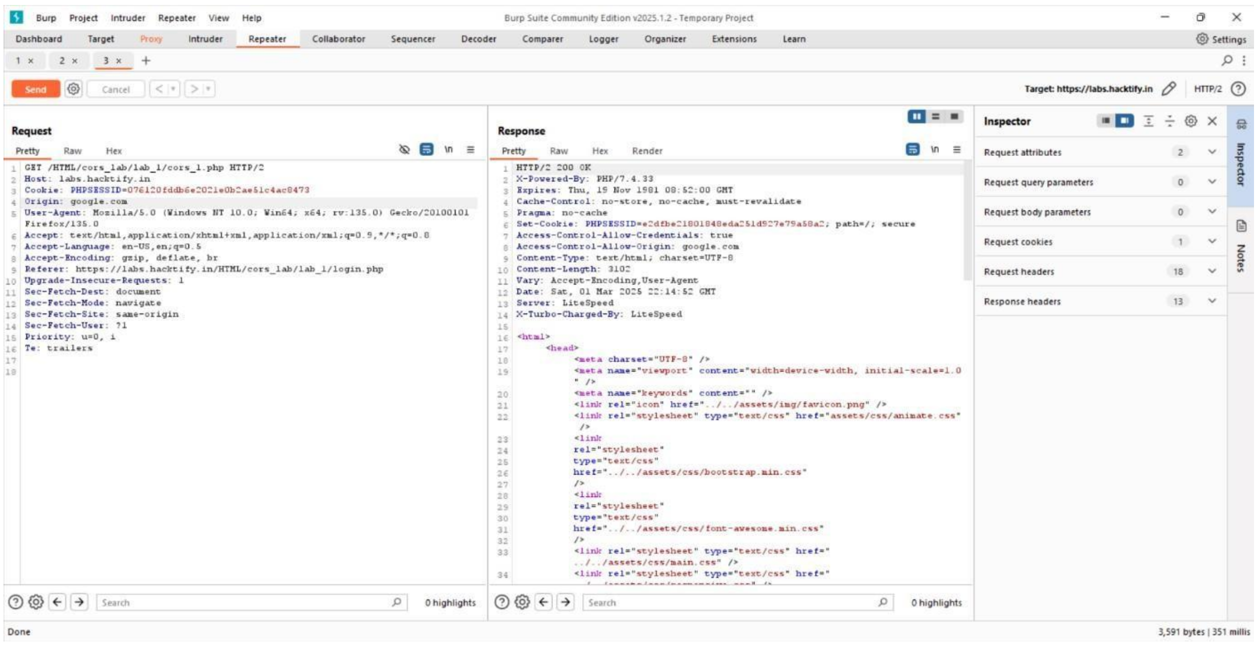
1.1. Sub-lab-1 CORS with Arbitrary origin!

Reference	**Risk Rating**
Sub-lab-1: CORS with Arbitrary origin	**Low**
Tools Used	
Browser / Burp suite	
Vulnerability Description	
Cross-Origin Resource Sharing (CORS), this type of web site misconfiguration that can leave the application at a high risk of compromises resulting in an impact on the confidentiality and integrity of data by allowing third-party sites to carry out privileged requests through the website's authenticated users such as retrieving user setting information or saved payment card data. On the other hand, the risk is low for applications that deal with public data and require that resources are sent to other origins. The configuration could be expected behaviour and it would need to be up to the penetration tester to identify the appropriate risk and the organization to understand and mitigate, or accept the risk.	
How It Was Discovered	
Burp Suite used to find the vulnerability.	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
Not patching a Cross-Origin Resource Sharing (CORS) vulnerability can have significant consequences on the application, some which are: 1. **Cross-site scripting (XSS):** An attacker can exploit a CORS vulnerability to launch XSS attacks, allowing them to inject malicious code into a user's browser.	

2. **Data exposure:** A CORS vulnerability can allow an attacker to access sensitive data from other origins, potentially leading to data breaches or exposure of confidential information. 3. **CSRF attacks:** An attacker can exploit a CORS vulnerability to launch CSRF attacks, allowing them to perform unauthorized actions on behalf of a legitimate user. 4. **Reputational damage:** A CORS vulnerability can damage your organization's reputation, eroding customer trust and confidence. 5. **Financial losses:** A successful attack exploiting a CORS vulnerability can result in financial losses due to data breaches, intellectual property theft, or other malicious activities.
Suggested Countermeasures
1. Restrict Access-Control-Allow-Origin 2. Do not use Access-Control-Allow-Origin: for requests that require authentication. Instead, maintain an allowlist of trusted domains. 1. Disable Access-Control-Allow-Credentials Unless Necessary 2. Setting Access-Control-Allow-Credentials: true allows session cookies to be sent in cross-origin requests, increasing the risk of session hijacking. 3. If credentials are required, limit the allowed origins strictly. 1. Validate and Sanitize Origins Properly 2. Avoid dynamically reflecting the Origin header in Access-Control-Allow-Origin without strict validation. Ensure only trusted domains are accepted.
References
https://www.packetlabs.net/posts/cross-origin-resource-sharing-cors/

Proof of Concept

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


Burp Suite Community Edition v2025.1.2 - Temporary Project
Repeater
Send
Target: https://labs.hacktify.in
HTTP/2
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 1
Request headers 18
Response headers 13
3,591 bytes | 351 millis

Request

```
GET /HTML/cors_lab/lab_1/cors_1.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=076120fddb6e2021e0b2ae51c4ac8473
Origin: google.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101
Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/cors_lab/lab_1/login.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=e2dfbe21801848eda251d927e79a58a2; path=/; secure
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: google.com
Content-Type: text/html; charset=UTF-8
Content-Length: 3102
Vary: Accept-Encoding,User-Agent
Date: Sat, 01 Mar 2025 22:14:52 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

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

1.2. Sub-lab-2 CORS with null origin!

Reference	**Risk Rating**
Sub-lab-2: CORS with null origin!	**Low**
Tools Used	
Browser burp suite	
Vulnerability Description	
Cross-Origin Resource Sharing (CORS) is a security mechanism that controls how web pages in one domain can request resources from another domain. A CORS misconfiguration occurs when a server incorrectly allows unauthorized cross-origin requests, potentially exposing sensitive data to attackers.	
How It Was Discovered	
Automated Tools Burp suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_2/login.ph	
Consequences of not Fixing the Issue	
CORS contains two main components that when misconfigured can pose a significant risk to any web application. The two components are: • **Access-Control-Allow-Origin – (ACAO)** allows for two-way interaction by third-party websites. This can be an issue for requests that modify or pull sensitive data. • **Access-Control-Allow-Credentials** is where third-party websites can carry out privileged actions. Think of this as an attacker conducting changes that only you, the authenticated user, should be able to.	

The types of misconfigurations can vary depending on the deployment. Below are the most common corresponding risks:

1. Compliance issues: Failure to patch a CORS vulnerability can lead to non-compliance with regulatory requirements, resulting in fines and penalties.
2. System downtime: A CORS vulnerability can cause system downtime, impacting business operations and customer service.
3. Resource intensive: Responding to a CORS vulnerability can be resource-intensive, requiring significant time and effort from your security team.
4. Legal liabilities: Your organization may be held liable for damages resulting from a CORS vulnerability, including legal fees, settlements, and judgments.
5. Data compromise: A user's data may be compromised due to a CORS vulnerability, potentially leading to identity theft, financial losses, or other malicious activities.

Suggested Countermeasures

CORS vulnerabilities arise primarily as misconfigurations. Prevention is therefore a configuration problem. The following is a list of some effective defences against CORS attack:

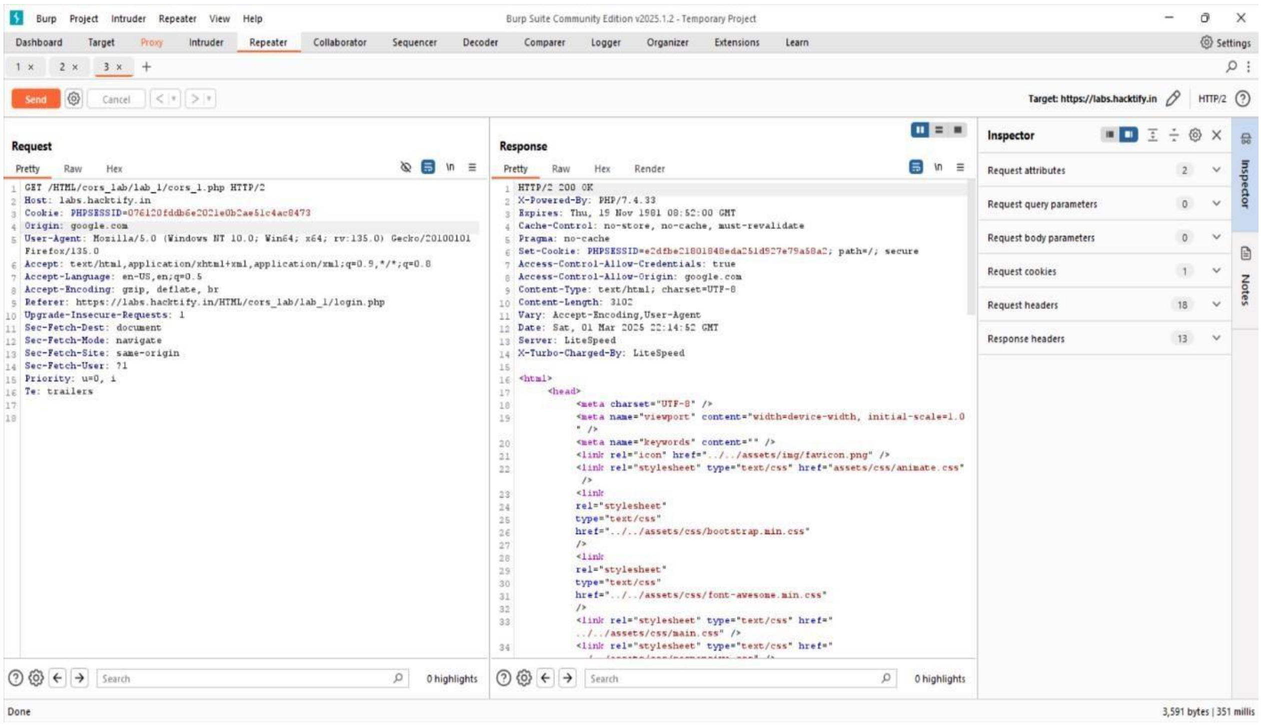
1. Proper configuration of cross-origin requests
2. Only allow trusted sites
3. Avoid whitelisting null
4. Avoid wildcards in internal networks
5. CORS is not a substitute for server-side security policies

References

https://portswigger.net/web-security/all-labs#cross-origin-resource-sharing-cors
https://portswigger.net/web-security/cors https://owasp.org/www-project-secure-headers/#cors https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Proof of Concept

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



```
GET /HTML/cors_lab/lab_1/cors_1.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=076120fddb6e2021e0b2ae51c4ac8473
Origin: google.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101
Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/cors_lab/lab_1/login.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=e2dfbe21801848eda251d927e79a58a2; path=/; secure
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: google.com
Content-Type: text/html; charset=UTF-8
Content-Length: 3102
Vary: Accept-Encoding,User-Agent
Date: Sat, 01 Mar 2025 22:14:52 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

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

Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.1.2 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Send
Cancel
Target: https://labs.hacktify.in
HTTP/2
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 1
Request headers 18
Response headers 13
Notes
Search
0 highlights
Done
3,591 bytes | 351 millis

1.3. Sub-lab-3 CORS with prefix match

Reference	**Risk Rating**
Sub-lab-3: CORS with prefix match	**Medium**
Tools Used	
Browser/ Burp suite	
Vulnerability Description	
Cross-origin resource sharing (CORS) vulnerability, it is flaw in a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the same-origin policy (SOP). However, it also provides potential for cross-domain attacks, if a website's CORS policy is poorly configured and implemented. CORS is not a protection against cross-origin attacks such as cross-site request forgery (CSRF).	
How It Was Discovered	
Automated tool burp suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_3/login.php	
Consequences of not Fixing the Issue	
If CORS (Cross-Origin Resource Sharing) vulnerability are left unpatched, they can have severe consequences like: 1. Cross-site scripting (XSS)	

2. Data exposure 3. Reputational damage 4. Compliance issues 5. Legal liabilities 6. Account compromise
Suggested Countermeasures
CORS vulnerabilities arise primarily as misconfigurations. Prevention is therefore a configuration problem. The following describes some effective defences against CORS attack: 1. Proper configuration of cross-origin requests 2. Only allow trusted sites 3. Avoid whitelisting null 4. Avoid wildcards in internal networks 5. CORS is not a substitute for server-side security policies
References
https://portswigger.net https://portswigger.net/web-security/cors https://owasp.org/www-project-secure-headers/#cors https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Proof of Concept

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

1.4. Sub-lab-4 CORS with suffix match!

Reference	**Risk Rating**
Sub-lab-4: CORS with suffix match!	**medium**
Tools Used	
Burp suite	
Vulnerability Description	
The application has a Cross-Origin Resource Sharing (CORS) misconfiguration, allowing an attacker- controlled domain (attacker.com.hacktify.in) to interact with the server as an authenticated user	
How It Was Discovered	
Automated tool	
Vulnerable URLs	

https://labs.hacktify.in/HTML/cors_lab/lab_4/login.php

Consequences of not Fixing the Issue

If CORS (Cross-Origin Resource Sharing) vulnerability are left unpatched they can lead to big issues to an organization like:

1. **Security Consequences**
   - Cross-site scripting (XSS)
   - CSRF attacks: An attacker can exploit a CORS vulnerability to launch CSRF attacks, allowing them to perform unauthorized actions on behalf of a legitimate user.

2. **Business Consequences**
   - Financial losses
   - Compliance issues

3. **Operational Consequences**
   - Resource intensive
   - Legal liabilities

4. **User Consequences**
   - Malware infections
   - Account compromise

Suggested Countermeasures

CORS vulnerabilities arise primarily as misconfigurations. Prevention is therefore a configuration problem. The following describes some effective defences against CORS attack:

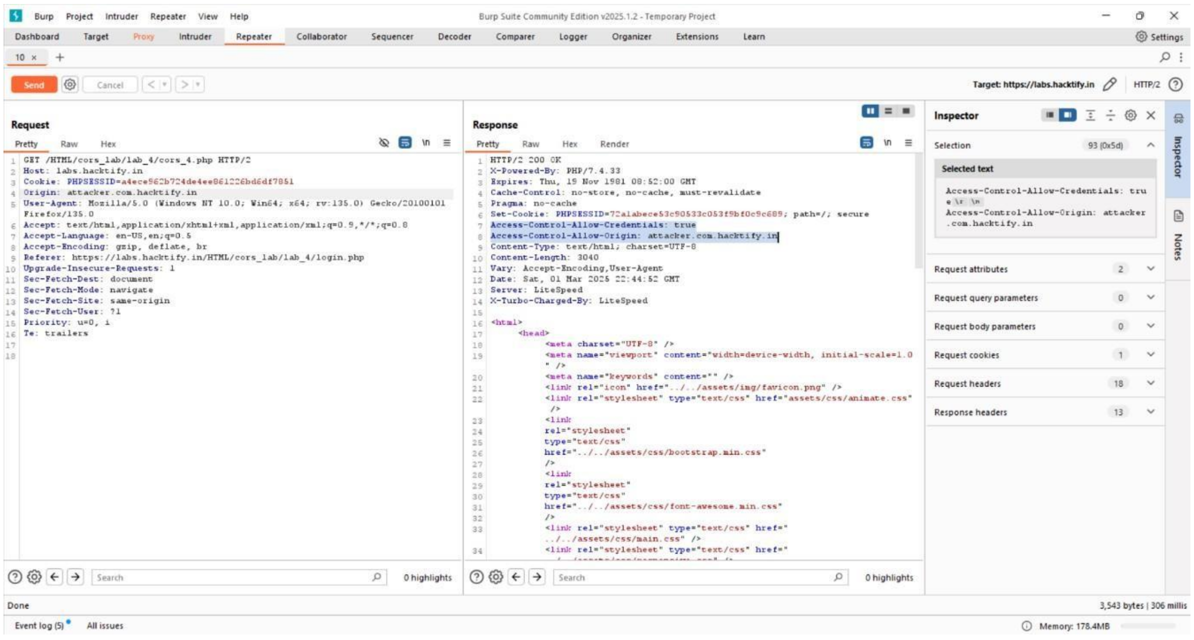
1) Proper configuration of cross-origin requests
2) Only allow trusted sites
3) Avoid whitelisting null
4) Avoid wildcards in internal networks
5) CORS is not a substitute for server-side security policies

References

https://portswigger.net/web-security/cors https://owasp.org/www-project-secure-headers/#cors https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Proof of Concept

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



```
GET /HTML/cors_lab/lab_4/cors_4.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=a4ece962b724de4ee861226bd6df7851
Origin: attacker.com.hacktify.in
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101
Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/cors_lab/lab_4/login.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=72a1abece53c90533c053f9bf0c9c689; path=/; secure
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: attacker.com.hacktify.in
Content-Type: text/html; charset=UTF-8
Content-Length: 3040
Vary: Accept-Encoding,User-Agent
Date: Sat, 01 Mar 2025 22:44:52 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

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

Selected text

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: attacker
.com.hacktify.in
```

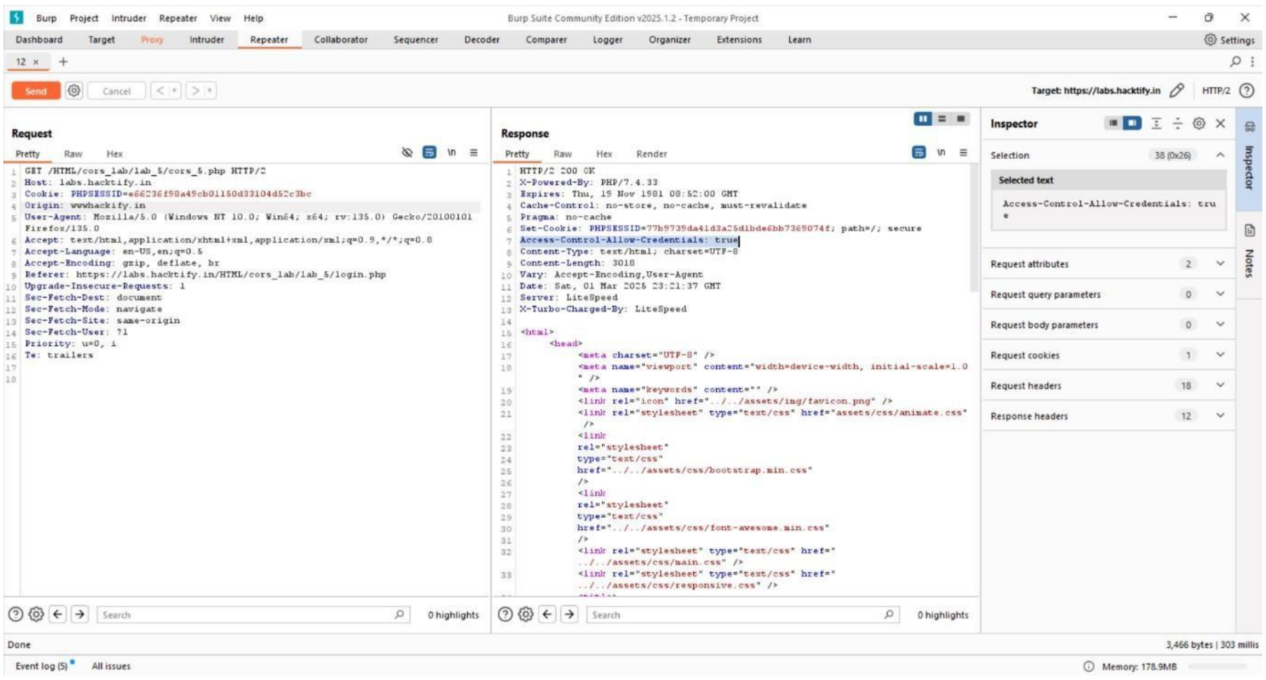
1.5. Sub-lab-5 CORS with Escape dot!

Reference	**Risk Rating**
Sub-lab-5: CORS with suffix match!	**High**
Tools Used	
Manual analysis “Browser”	
Vulnerability Description	
During testing, it was observed that the server responded with the Access-Control-Allow-Credentials: true header, indicating that the application permits credentialed requests from cross-origin sources. Additionally, modifying the Origin header (e.g., using wwwhacktify.in instead of www.hacktify.in) resulted in a successful response, suggesting weak or improper origin validation. This misconfiguration could allow an attacker to exploit a vulnerable endpoint by hosting a malicious website and tricking authenticated users into making unauthorized requests, potentially leading to session hijacking, data theft, or unauthorized actions.	
How It Was Discovered	
Automated tool	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_5/login.php	
Consequences of not Fixing the Issue	
If CORS (Cross-Origin Resource Sharing) vulnerability are not accurately patched or not just fixed at all, they can negatively impact the organization in ways like: 1. Cross-site scripting (XSS 2. Data exposure	

3. Reputational damage 4. Compliance issues 5. System downtime 6. Data compromise 7. Account compromise
Suggested Countermeasures
CORS vulnerabilities arise primarily as misconfigurations. Prevention is therefore a factor The following describes some effective defences against CORS attack: 1. Proper configuration of cross-origin requests 2. Only allow trusted sites 3. Avoid whitelisting null 4. Avoid wildcards in internal networks 5. CORS is not a substitute for server-side security policies
References
https://portswigger.net/web-security/file-upload https://www.vaadata.com/blog/sql-injections-principles-impacts-exploitations-security-best-practices/

Proof of Concept

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



Request

```
GET /HTML/cors_lab/lab_5/cors_5.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=e66236f98a49cb01150d33104d52c3bc
Origin: wwhackify.in
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101
Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/cors_lab/lab_5/login.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=77b9739da41d3a25d1bde6bb7369074f; path=/; secure
Access-Control-Allow-Credentials: true
Content-Type: text/html; charset=UTF-8
Content-Length: 3018
Vary: Accept-Encoding,User-Agent
Date: Sat, 01 Mar 2025 23:21:37 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

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

Inspector — Selected text: Access-Control-Allow-Credentials: true

Burp Suite Community Edition v2025.1.2 - Temporary Project
Repeater
Send
Target: https://labs.hacktify.in
HTTP/2
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 1
Request headers 18
Response headers 12
3,466 bytes | 303 millis

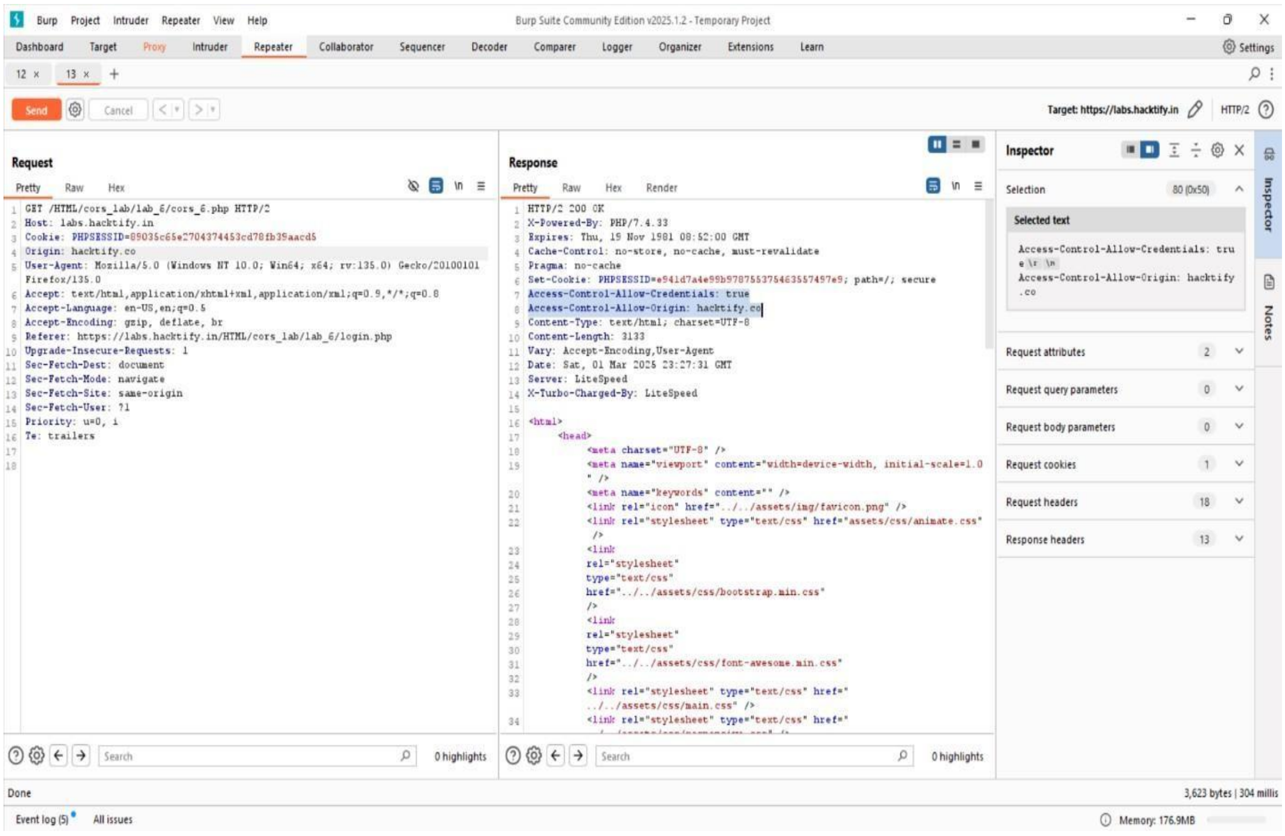
1.6. Sub-lab-6 CORS with Subscribing match!

Reference	Risk Rating
Sub-lab-6: CORS with Subscribing match!	**High**
Tools Used	
Browser/ Burp suite	
Vulnerability Description	
Cross-Origin Resource Sharing (CORS) vulnerabilities, these security flaws that occurs when a web application allows cross-domain requests without proper validation of the origin. This can lead to an attacker being able to perform unauthorized actions, such as stealing sensitive information, through malicious JavaScript code executed on the victim's browser. The root cause of this vulnerability lies in the permissive access control policies implemented by the web application. For instance, imagine that an application is designed to allow cross-domain requests from any domain. A malicious actor could craft a script to send a request to the application and then execute it from their own domain. If the application doesn't properly validate the origin of the request, it will allow the attacker's domain to access its resources. This allows the attacker to exploit vulnerabilities and potentially gain access to sensitive data.	
How It Was Discovered	
Automated tool Burp suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_6/login.php	
Consequences of not Fixing the Issue	
CORS (Cross-Origin Resource Sharing) vulnerability are web application flaws that can lead to significant consequences on the organization such as:	

1. Reputational damage 2. Compliance issues 3. System downtime 4. Legal liabilities 5. Data compromise 6. Account compromise
Suggested Countermeasures
CORS vulnerabilities arise primarily as misconfigurations. Prevention is therefore a configuration problem. The following describes some effective defences against CORS attack: 1. Proper configuration of cross-origin requests 2. Only allow trusted sites 3. Avoid whitelisting null 4. Avoid wildcards in internal networks 5. CORS is not a substitute for server-side security policies
References
https://medium.com/@tushar_rs_/cross-origin-resource-sharing-cors-vulnerability-example-and-prevention-588d299ff185#:~:text=CORS%20vulnerabilities%20pose%20a%20serious,secure%20and%20protected%20from%20exploitation.

Proof of Concept

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



Request

```
GET /HTML/cors_lab/lab_6/cors_6.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=89035c65e2704374453cd78fb39aacd5
Origin: hacktify.co
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101
Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/cors_lab/lab_6/login.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=e941d7a4e99b97875537546355749 7e9; path=/; secure
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: hacktify.co
Content-Type: text/html; charset=UTF-8
Content-Length: 3133
Vary: Accept-Encoding,User-Agent
Date: Sat, 01 Mar 2025 23:27:31 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

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

Selected text

```
Access-Control-Allow-Credentials: true\r\n
Access-Control-Allow-Origin: hacktify.co
```

1.7. Sub-lab-7 CORS with Arbitrary Subdomain!

Reference	**Risk Rating**
Sub-lab-7 CORS with Arbitrary Subdomain!	**High**
Tools Used	
Burp suite used to find the vulnerability.	
Vulnerability Description	
Cross-Origin Resource Sharing (CORS) vulnerabilities, these security flaws that occurs when a web application allows cross-domain requests without proper validation of the origin. This can lead to an attacker being able to perform unauthorized actions, such as stealing sensitive information, through malicious JavaScript code executed on the victim's browser. The root cause of this vulnerability lies in the permissive access control policies implemented by the web application. For instance, imagine that an application is designed to allow cross-domain requests from any domain. A malicious actor could craft a script to send a request to the application and then execute it from their own domain. If the application doesn't properly validate the origin of the request, it will allow the attacker's domain to access its resources. This allows the attacker to exploit vulnerabilities and potentially gain access to sensitive data.	
How It Was Discovered	
Manual Analysis “By trying all the entry points with scripts”	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_7/login.php	
Consequences of not Fixing the Issue	

Not patching a CORS (Cross-Origin Resource Sharing) vulnerability can have severe consequences such, the following are the different threats that arise from poor implementation of the Cross-Origin Resource Sharing mechanism:

1. Data exposure
2. CSRF attacks
3. Reputational damage
4. Financial losses
5. System downtime
6. Resource intensive
7. Legal liabilities

Suggested Countermeasures

Failure to Fix **Cross-Site Resource Sharing (CORS)** vulnerabilities can have a very bad impact on the web application, namely:

1. Compliance Issues
2. Business Disruption
3. Physical Security Risks
4. Regulatory compliance
5. Competitive Disadvantage

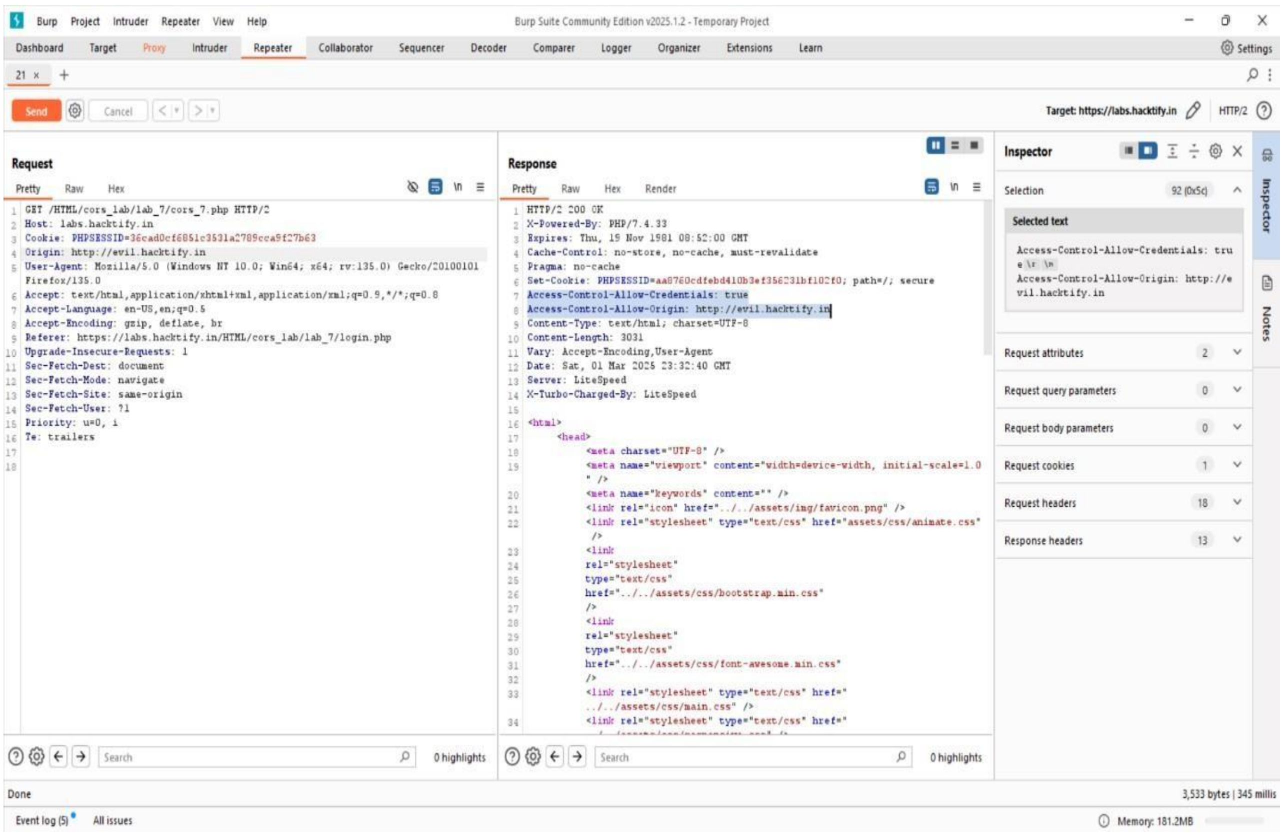
It's essential patching CORS vulnerabilities to prevent these severe impacts and protect sensitive data.

References

https://zerothreat.ai/blog/cors-explained-mitigating-cross-origin-risks
https://medium.com/@tushar_rs_/cross-origin-resource-sharing-cors-vulnerability-example-and-prevention-588d299ff185#:~:text=CORS%20vulnerabilities%20pose%20a%20serious,secure%20and%20protected%20from%20exploitation.

Proof of Concept

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



```
GET /HTML/cors_lab/lab_7/cors_7.php HTTP/2
Host: labs.hacktify.in
Cookie: PHPSESSID=36cad0cf6851c3531a2789cca9f27b63
Origin: http://evil.hacktify.in
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101
Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/cors_lab/lab_7/login.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=aa8760cdfebd410b3ef356231bf102f0; path=/; secure
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://evil.hacktify.in
Content-Type: text/html; charset=UTF-8
Content-Length: 3031
Vary: Accept-Encoding,User-Agent
Date: Sat, 01 Mar 2025 23:32:40 GMT
Server: LiteSpeed
X-Turbo-Charged-By: LiteSpeed

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

Inspector — Selected text:

```
Access-Control-Allow-Credentials: true \r \n
Access-Control-Allow-Origin: http://evil.hacktify.in
```

Burp Suite Community Edition v2025.1.2 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send
Cancel
Target: https://labs.hacktify.in
HTTP/2
Request
Response
Pretty Raw Hex Render
Inspector
Selection 92 (0x5c)
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 1
Request headers 18
Response headers 13
0 highlights
Done
3,533 bytes | 345 millis
Event log (5)
All issues
Memory: 181.2MB

2. Lab 2 Cross-Site Request Forgery (CSRF)

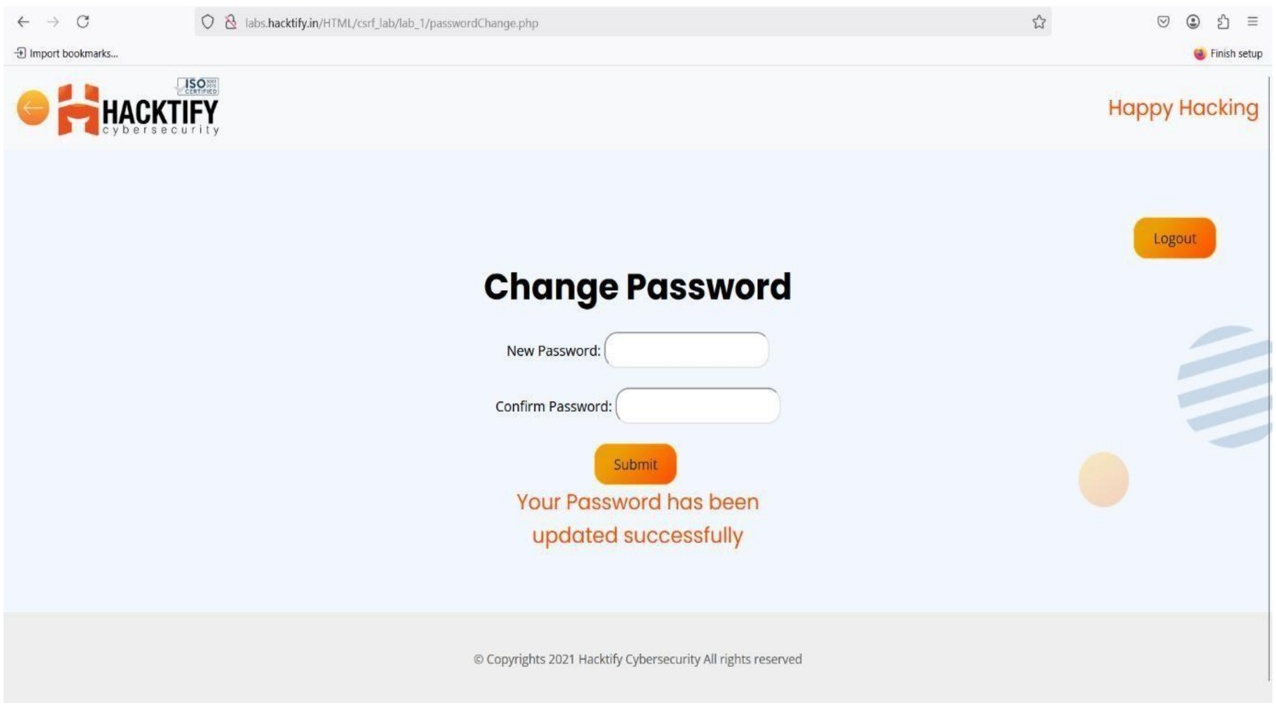
2.1. Sub-lab-1: Eassy CSRF!

Reference	**Risk Rating**
Sub-lab-1: Eassy CSRF!	**Low**
Tools Used	
https://labs.hacktify.in/HTML/csrf_lab/lab_1/login.php	
Vulnerability Description	
The Cross-Site Request Forgery (CSRF) vulnerability in the password change functionality allows an attacker to manipulate an authenticated user's session by forcing them to perform unauthorized actions without their knowledge. Since the application does not implement CSRF protection mechanisms (such as anti-CSRF tokens or SameSite cookie attributes), an attacker can craft a malicious request that automatically submits a password change request on behalf of the victim when they visit a malicious website or interact with a deceptive link.	
How It Was Discovered	
Automated Tools burp suite	

Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_1/login.php
Consequences of not Fixing the Issue
Not patching a CSRF (Cross-Site Request Forgery) vulnerability can have severe consequences, repercussions such as: 1. Unauthorized actions: An attacker can perform unauthorized actions on behalf of a legitimate user, potentially leading to data breaches, financial losses, or reputational damage. 2. Session hijacking: An attacker can hijack a user's session, allowing them to access sensitive data and perform malicious actions. 3. Data tampering: An attacker can modify or delete sensitive data, leading to data corruption or loss. 4. Financial losses: A successful CSRF attack can result in financial losses due to unauthorized transactions, data breaches, or other malicious activities. 5. Reputational damage: A CSRF vulnerability can damage your organization's reputation, eroding customer trust and confidence. 6. Compliance issues: Failure to patch a CSRF vulnerability can lead to non-compliance with regulatory requirements, resulting in fines and penalties.
Suggested Countermeasures
1. Patch Management 2. Secure Coding Practices 3. Code Refactoring 4. Security Information and Event Management (SIEM) 5. Penetration Testing
References

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/csrf_lab/lab_1/passwordChange.php

Change Password

New Password:

Confirm Password:

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

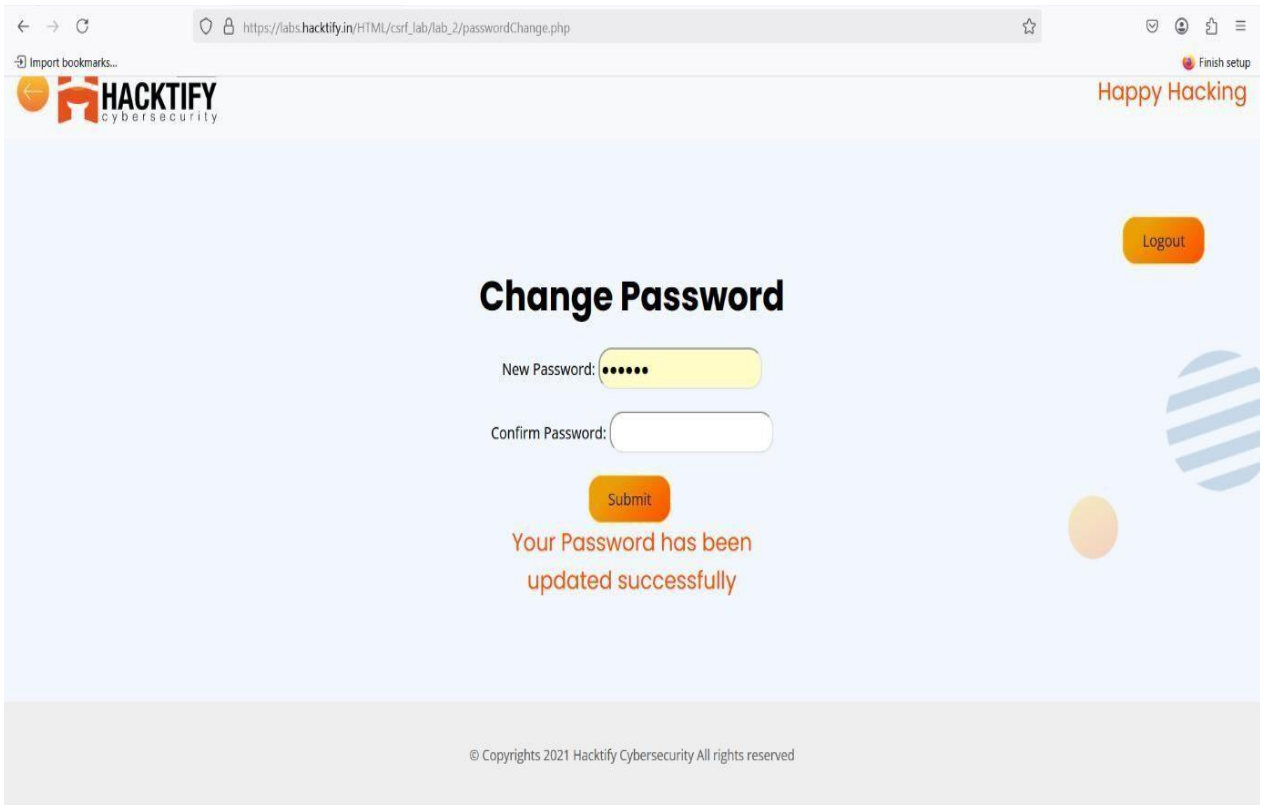
2.2. Sub-lab-2: Always validate tokens!

Reference	Risk Rating
Sub-lab-3: Always validate tokens!	**Medium**
Tools Used	
Burp suite	
Vulnerability Description	
The Cross-Site Request Forgery (CSRF) vulnerability exists in the password change functionality despite the presence of a TSRF (Token-based CSRF Protection) token. However, the application fails to properly validate the token, allowing an attacker to manipulate a logged-in user's session and perform unauthorized actions. By modifying the CSRF token value in the exploit, the attacker was still able to execute a password change request, indicating broken CSRF protection.	
How It Was Discovered	
Burp Suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_2/login.php	
Consequences of not Fixing the Issue	
Leaving the CSRF (Cross-Site Request Forgery) vulnerability unpatched can lead to dead serious issues like:	

1. Data tampering 2. Financial losses 3. Reputational damage 4. System downtime 5. Financial losses 6. Identity theft
Suggested Countermeasures
1. Patch Management 2. Secure Coding Practices 3. Code Refactoring 4. Security Information and Event Management (SIEM) 5. Penetration Testing
References
https://portswigger.net/web-security/file-upload https://www.vaadata.com/blog/sql-injections-principles-impacts-exploitations-security-best-practices/ https://portswigger.net

Proof of Concept

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


https://labs.hacktify.in/HTML/csrf_lab/lab_2/passwordChange.php

Change Password

New Password: ••••••

Confirm Password:

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.3. Sub-lab-3: I hate when someone uses my tokens!

Reference	**Risk Rating**
Sub-lab-3: I hate when someone uses my tokens!	**Medium**
Tools Used	
Burp suite	
Vulnerability Description	
The application is vulnerable to Cross-Site Request Forgery (CSRF), allowing an attacker to change a user's password without proper authorization. Despite having a CSRF token, the vulnerability persists because the application does not properly validate or enforce the token. Additionally, in this instance, the CSRF token is displayed on the screen after the password change, exposing it to potential exploitation.	
How It Was Discovered	
Automated Tools "By trying different entry points and different scripts payload"	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_6/login.ph	
Consequences of not Fixing the Issue	
Not patching a CSRF (Cross-Site Request Forgery) vulnerability can have severe consequences:	

Security Consequences

- Unauthorized actions
- Session hijacking

Business Consequences

- Reputational damage
- Compliance issues

User Consequences

- Account compromise
- Financial losses
- Identity theft

Suggested Countermeasures

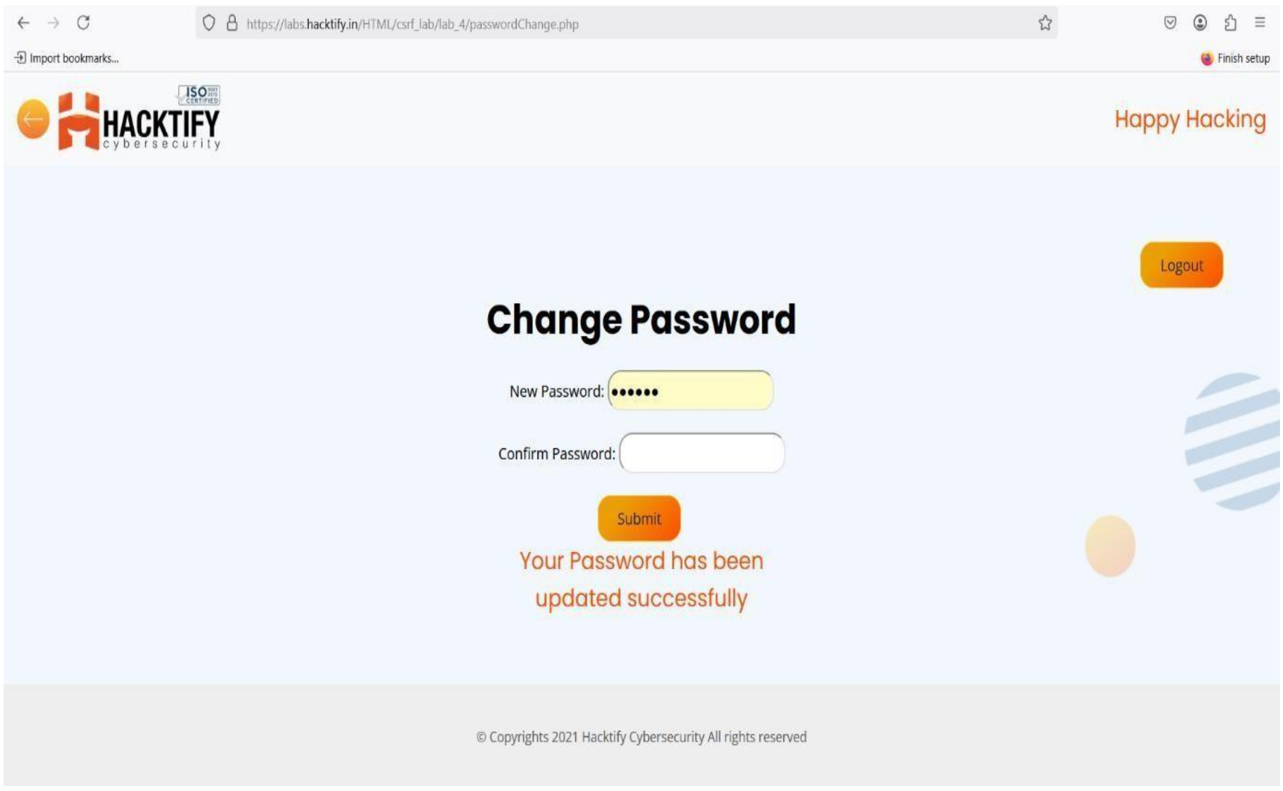
6. Patch Management
7. Secure Coding Practices
8. Code Refactoring
9. Security Information and Event Management (SIEM)
10. Penetration Testing

References

htps://www.invcti.com
https://portswigger.net
https://www.acunetix.com
https://owasp.org/www-community/attacks/csrf](https://owasp.org/www-community/attacks/csrf)
https://portswigger.net/web-security/csrf
https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Proof of Concept

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


https://labs.hacktify.in/HTML/csrf_lab/lab_4/passwordChange.php
Import bookmarks...
Finish setup
HACKTIFY cybersecurity
Happy Hacking
Logout
Submit
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

Change Password

New Password: ••••••

Confirm Password:

Your Password has been updated successfully

2.4. Sub-lab-4: Get me or post me!

Reference	**Risk Rating**
Sub-lab-4: Get me or Post me!	**low**
Tools Used	
Tools that you have used to find the vulnerability.	
Vulnerability Description	
SQL injection **(SQLi)** is a type of web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve. This might include data that belongs to other users, or any other data that the application can access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior. In some situations, an attacker can escalate a SQL injection attack to compromise the underlying server or other back-end infrastructure. It can also enable them to perform denial-of-service attacks. This type of vulnerability can have serious consequences and requires proper input validation and output encoding to mitigate the risk	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
Consequences of not Fixing the Issue	

Failing to patch CSRF (Cross-Site Request Forgery) vulnerability in the web application can have severe consequences some which are:

1. Session hijacking
2. Data tampering
3. Reputational damage
4. Compliance issues
5. Legal liabilities
6. Identity theft

Suggested Countermeasures

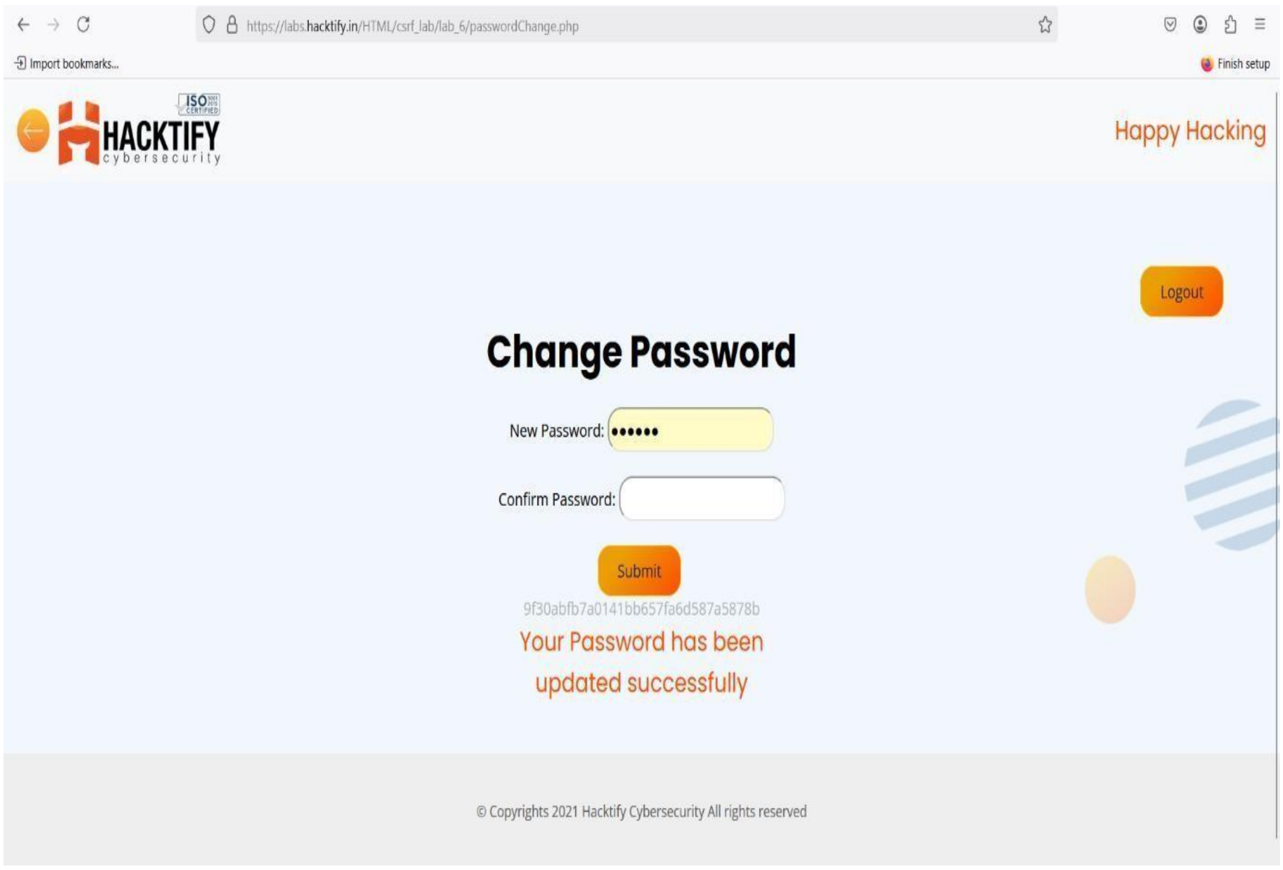
1. Implement proper CSRF token validation for every sensitive request.
2. Ensure that CSRF tokens are unique per session and request, making them unpredictable.
3. Use the SameSite attribute on cookies to prevent cross-origin requests.
4. Do not expose the CSRF token on-screen or in responses.
5. Implement strict referer header validation and custom request headers.

References

https://owasp.org/www-community/attacks/csrf https://portswigger.net/web-security/csrf https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://portswigger.net
https://www.acunetix.com

Proof of Concept

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


https://labs.hacktify.in/HTML/csrf_lab/lab_6/passwordChange.php

Change Password

New Password: ••••••

Confirm Password:

9f30abfb7a0141bb657fa6d587a5878b

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.5. Sub-lab-5 XSS the Saviour!

Reference	**Risk Rating**
Sub-lab-5: XSS the Saviour!	**High**
Tools Used	
Burp suite	
Vulnerability Description	
The application is vulnerable to Cross-Site Request Forgery (CSRF), allowing an attacker to change a user's password without proper authorization. Despite the presence of a CSRF token, the vulnerability persists because the application does not properly validate or enforce the token. Additionally, the CSRF token is displayed on-screen after the password change, exposing it to potential exploitation.	
How It Was Discovered	
Automated Tools burp suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_7/login.php	
Consequences of not Fixing the Issue	

Not patching a CSRF (Cross-Site Request Forgery) vulnerability can have severe consequences:

1. **Data tampering:** An attacker can modify or delete sensitive data, leading to data corruption or loss.
2. **Financial losses:** A successful CSRF attack can result in financial losses due to unauthorized transactions, data breaches, or other malicious activities.
3. **Reputational damage:** A CSRF vulnerability can damage your organization's reputation, eroding customer trust and confidence.
4. **Compliance issues:** Failure to patch a CSRF vulnerability can lead to non-compliance with regulatory requirements, resulting in fines and penalties.
5. **Account compromise:** A CSRF attack can result in a user's account being compromised, leading to unauthorized access to sensitive data.
6. **Identity theft:** A CSRF attack can increase the risk of identity theft, as an attacker may gain access to sensitive personal data.

Suggested Countermeasures

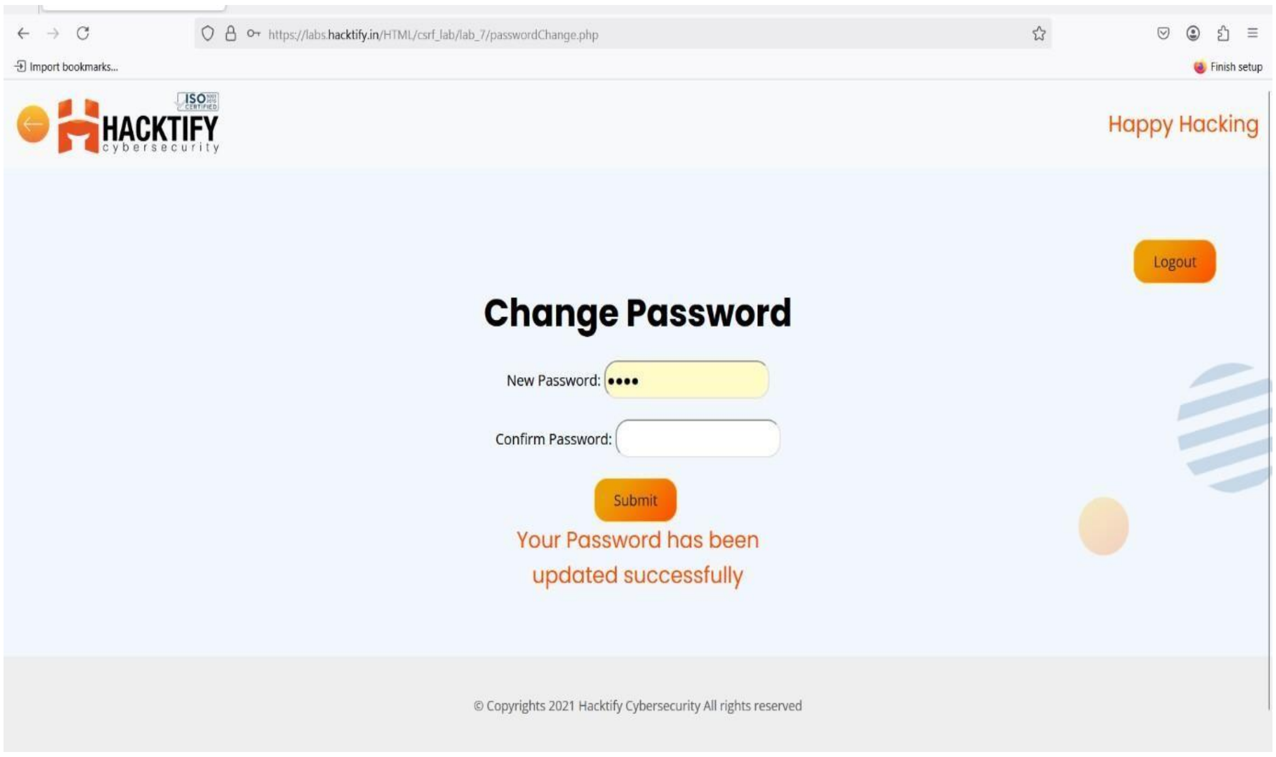
1. Implement Per-Request CSRF Tokens – Ensure each request gets a new CSRF token, making replay attacks impossible.
2. Invalidate Tokens After Use – Once a request is processed, mark the CSRF token as invalid and issue a new one.
3. Use Secure Cookie Attributes – Set HttpOnly, Secure, and SameSite=Strict for CSRF tokens to prevent access via client-side scripts.
4. Do Not Expose Tokens in Responses – CSRF tokens should only exist in HTTP headers or hidden form fields, never in URLs or responses.

References

https://owasp.org/www-community/attacks/csrf https://portswigger.net/web-security/csrf https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Proof of Concept

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


https://labs.hacktify.in/HTML/csrf_lab/lab_7/passwordChange.php

Change Password

New Password: ••••

Confirm Password:

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

2.6. Sub-lab-6 rm-rf token!

Reference	**Risk Rating**
Sub-lab-6: rm-rf token!	**High**
Tools Used	
Burp suite was used to find the vulnerability.	
Vulnerability Description	
The application is vulnerable to Cross-Site Request Forgery (CSRF), due to improper CSRF token enforcement. While a CSRF token is implemented, it is not validated for uniqueness as per request. This allows an attacker to re-use the same token multiple times, making it possible to execute repeated CSRF attacks. Additionally, if an attacker captures a valid token, they can continue exploiting it until the victim logs out or the session expires.	
How It Was Discovered	
Automated Tools Burp suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_8/login.php	
Consequences of not Fixing the Issue	
Failing to patch a CSRF (Cross-Site Request Forgery) vulnerability as an organization can lead to severe consequences such as: 1. Unauthorized actions:	

2. Session hijacking:
3. Financial losses:
4. Reputational damage:
5. Compliance issues:
6. Resource intensive:
7. Identity theft:

Suggested Countermeasures

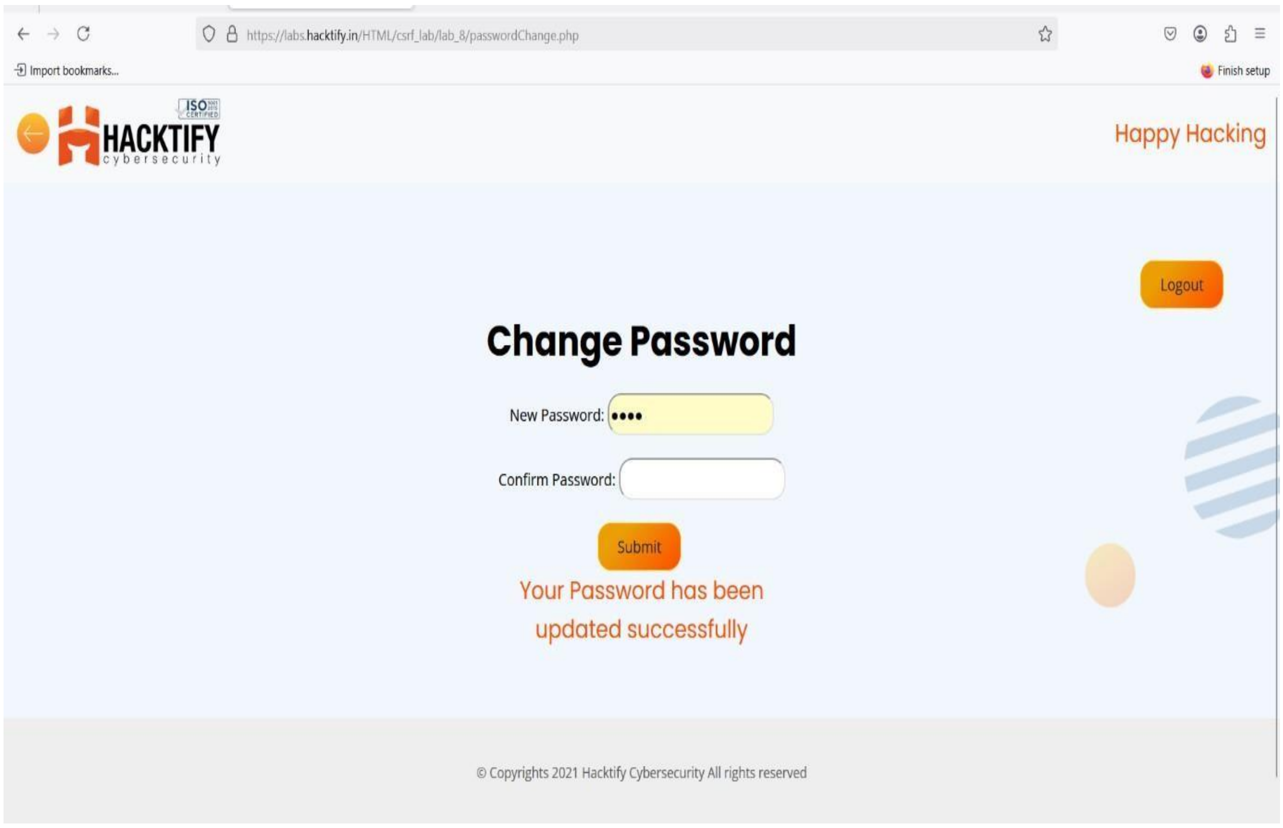
5. Implement Per-Request CSRF Tokens – Ensure each request gets a new CSRF token, making replay attacks impossible.
6. Invalidate Tokens After Use – Once a request is processed, mark the CSRF token as invalid and issue a new one.
7. Use Secure Cookie Attributes – Set HttpOnly, Secure, and SameSite=Strict for CSRF tokens to prevent access via client-side scripts.
8. Do Not Expose Tokens in Responses – CSRF tokens should only exist in HTTP headers or hidden form fields, never in URLs or responses.
9. Enforce Referer and Origin Checks – Validate Referer and Origin headers to ensure requests come from trusted sources.

References

https://owasp.org/www-community/attacks/csrf https://portswigger.net/web-security/csrf https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control

Proof of Concept

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



Change Password

New Password: ••••

Confirm Password:

Submit

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved