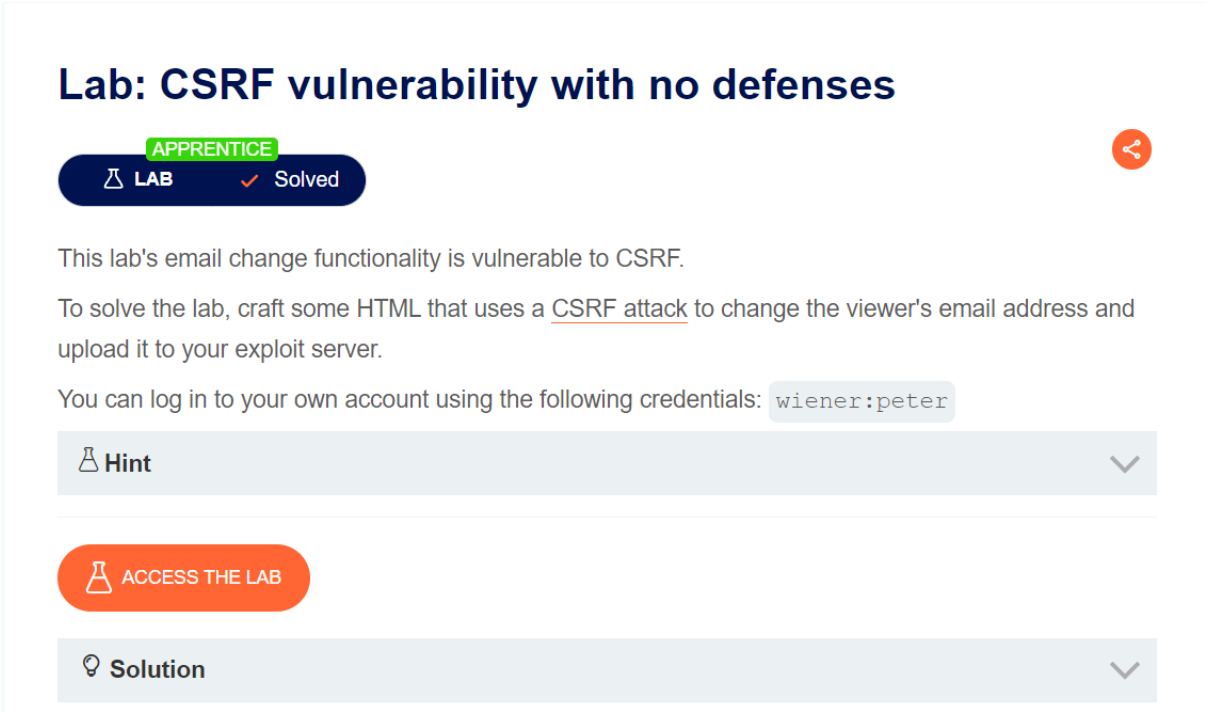
PORT SWIGGER

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.*

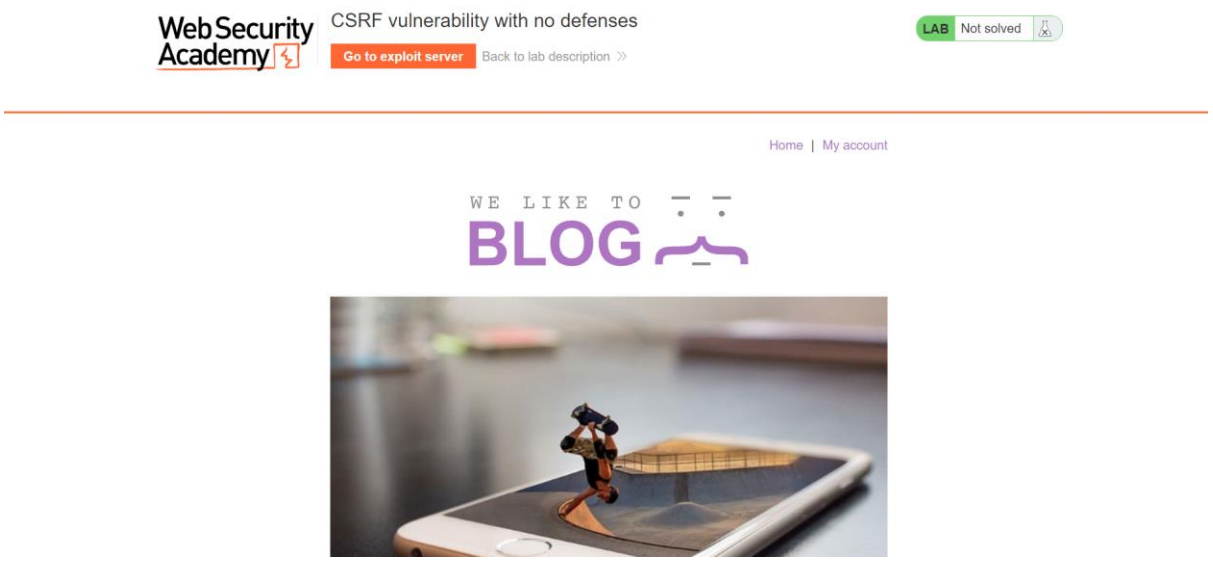
Lab 1: CSRF vulnerability with no defenses



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.

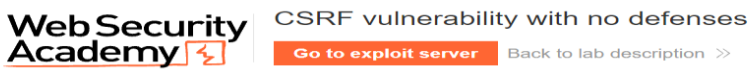
Let's access the lab,

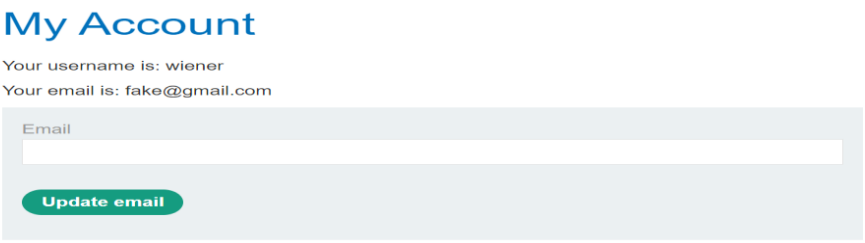


So let's login with the credential they had given, go to the **my account** option and enter username as wiener and password peter.

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

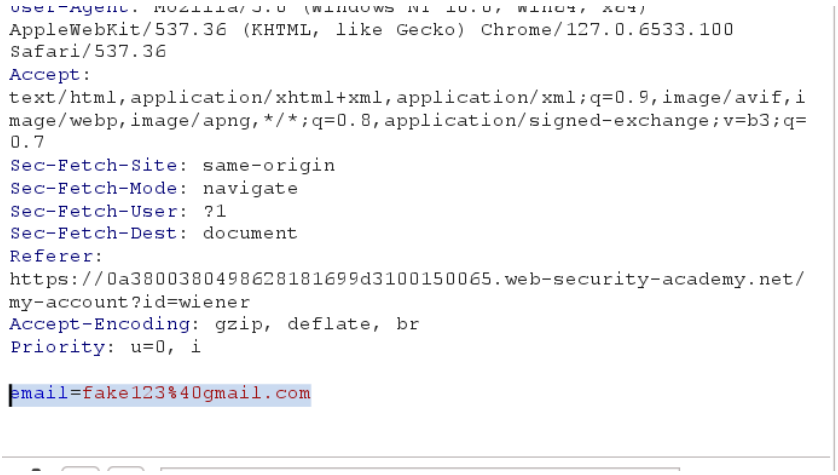
I’m entering a fake email id,





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.**



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

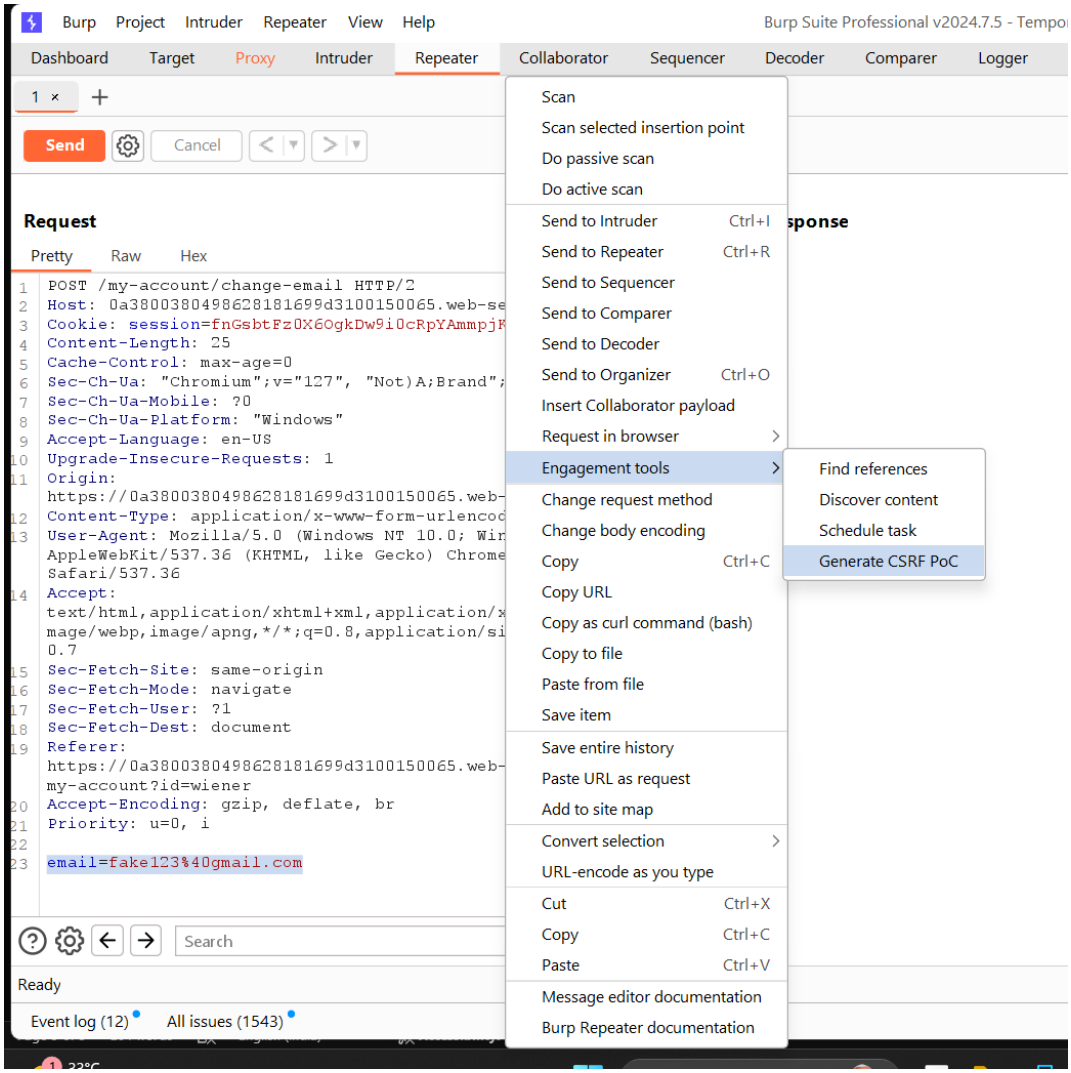
Email was updated and there was a request and a 302 Response on Burp.

This tells us that the is no defenses on switching a users 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 users email address if the given user click on the exploit servers link.

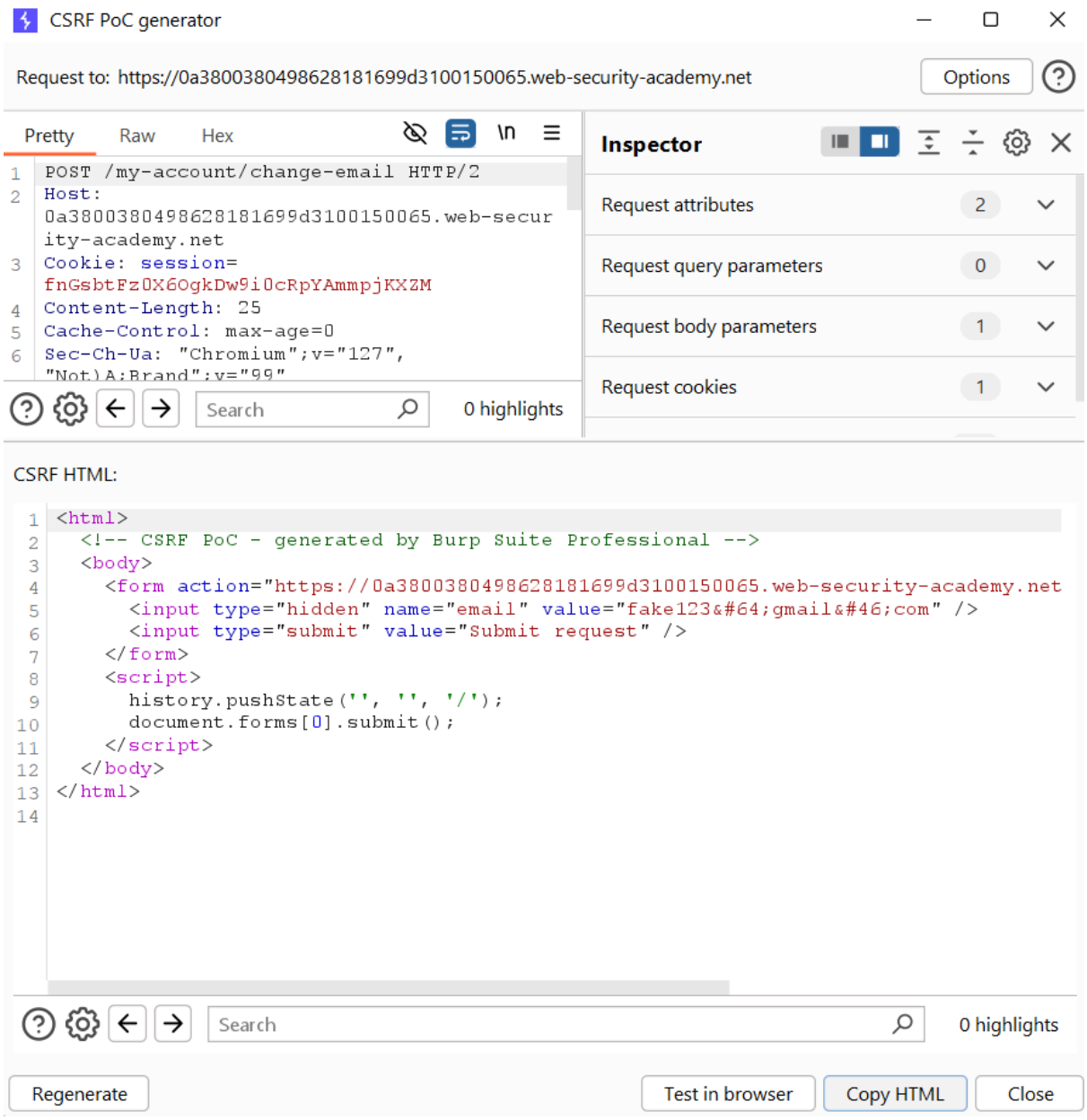
Soo in professional version there is a option engagement tool which is available to create **CSRF Poc(proof of concept).** Or it is also called to create the payload.

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



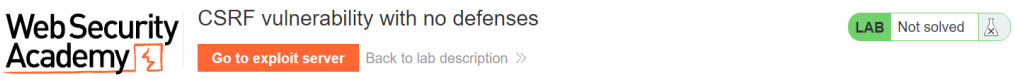
Open options in that window and make sure the “include auto-submit script” option is turned on.

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

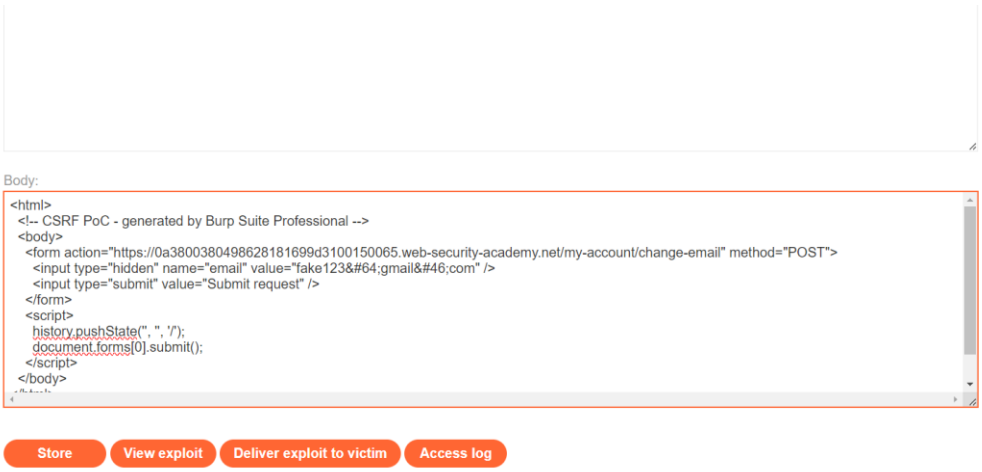


After this open our lab and there we can see a option of **“Go To Exploit Server”**

On the top.



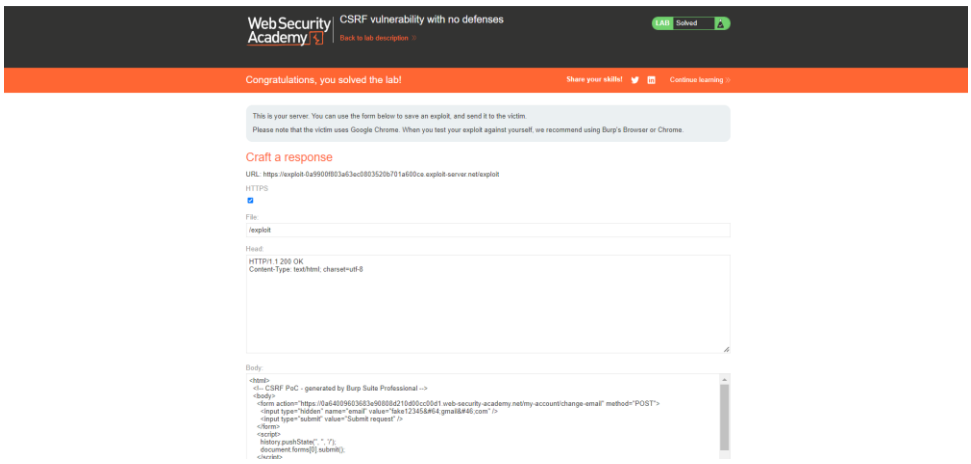
Go to exploit server and paste the copied HTML on the Body.



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

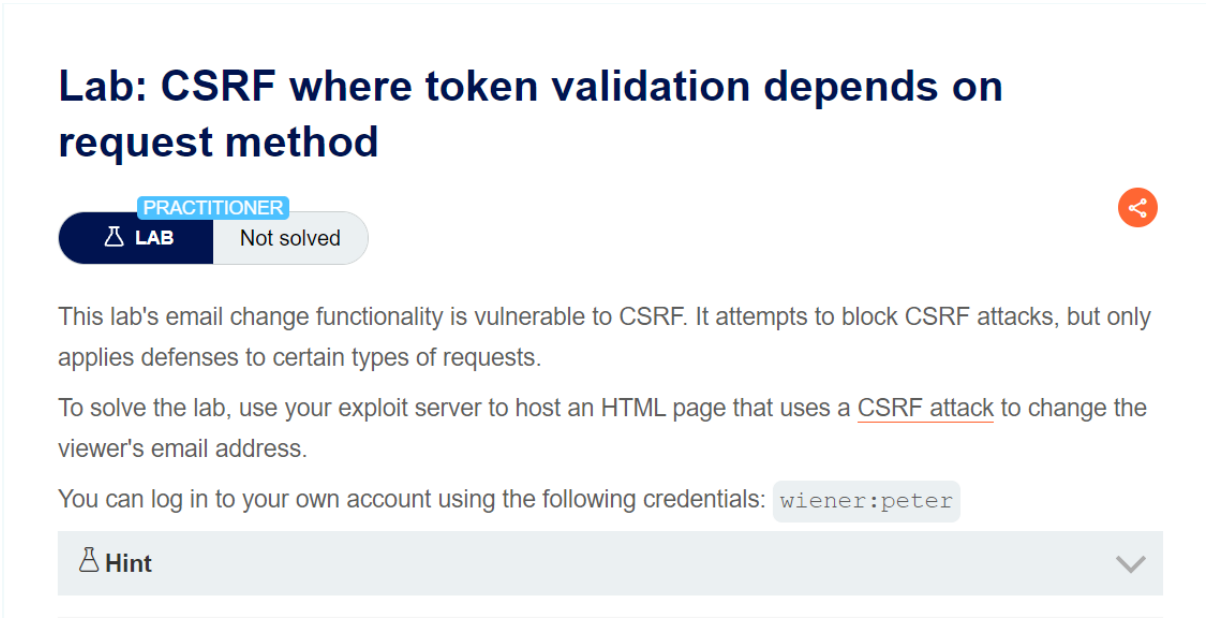
Here the exploit is delivered to the victim, before only we could see the result, now the all server got exploited.

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



So the lab is completed. 😊

Lab 2: CSRF where token validation depends on request method



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

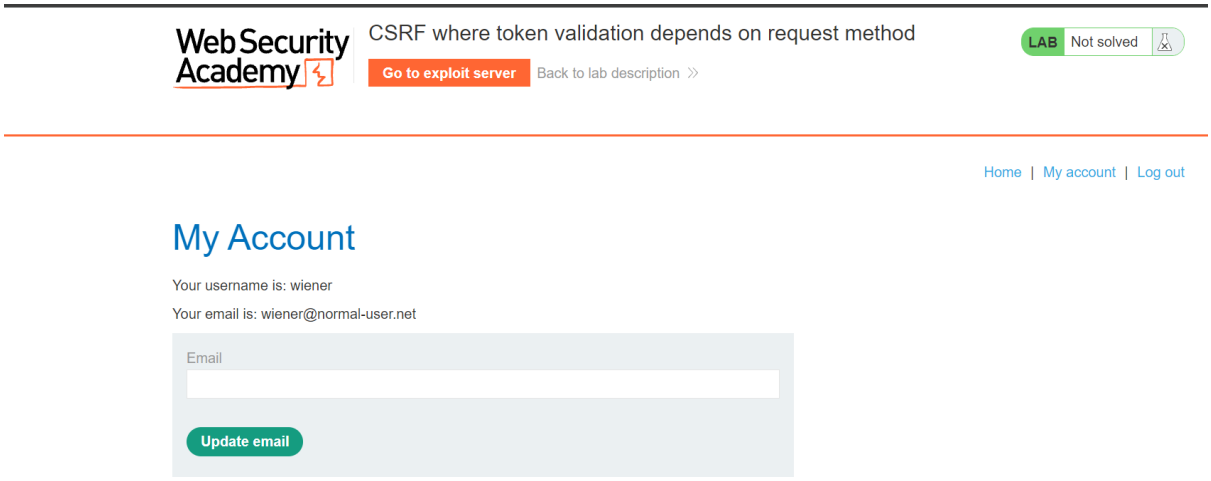
And also it shows it prevents only to certain types of requests.

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.

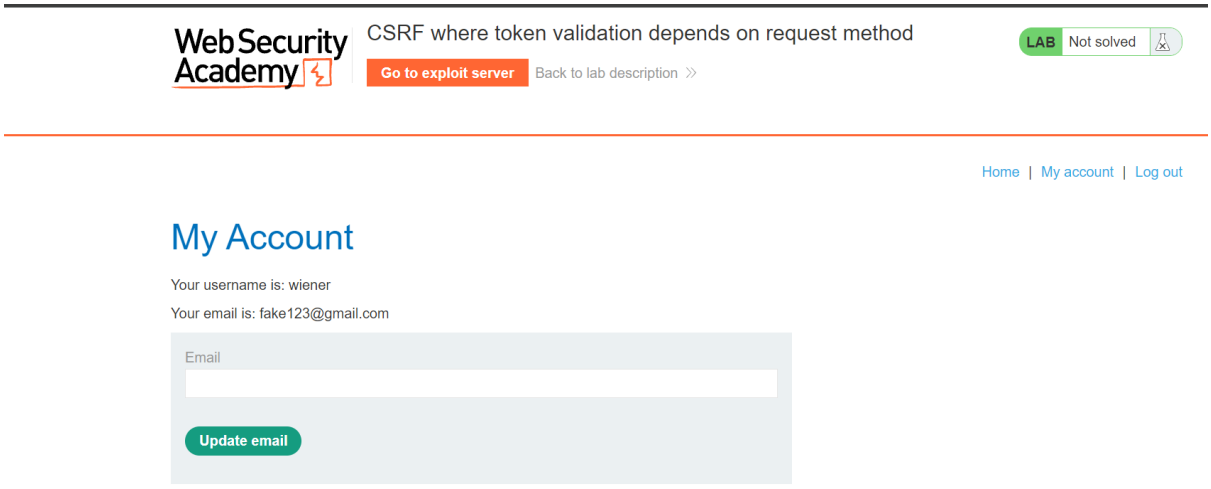
So let's see the lab.

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



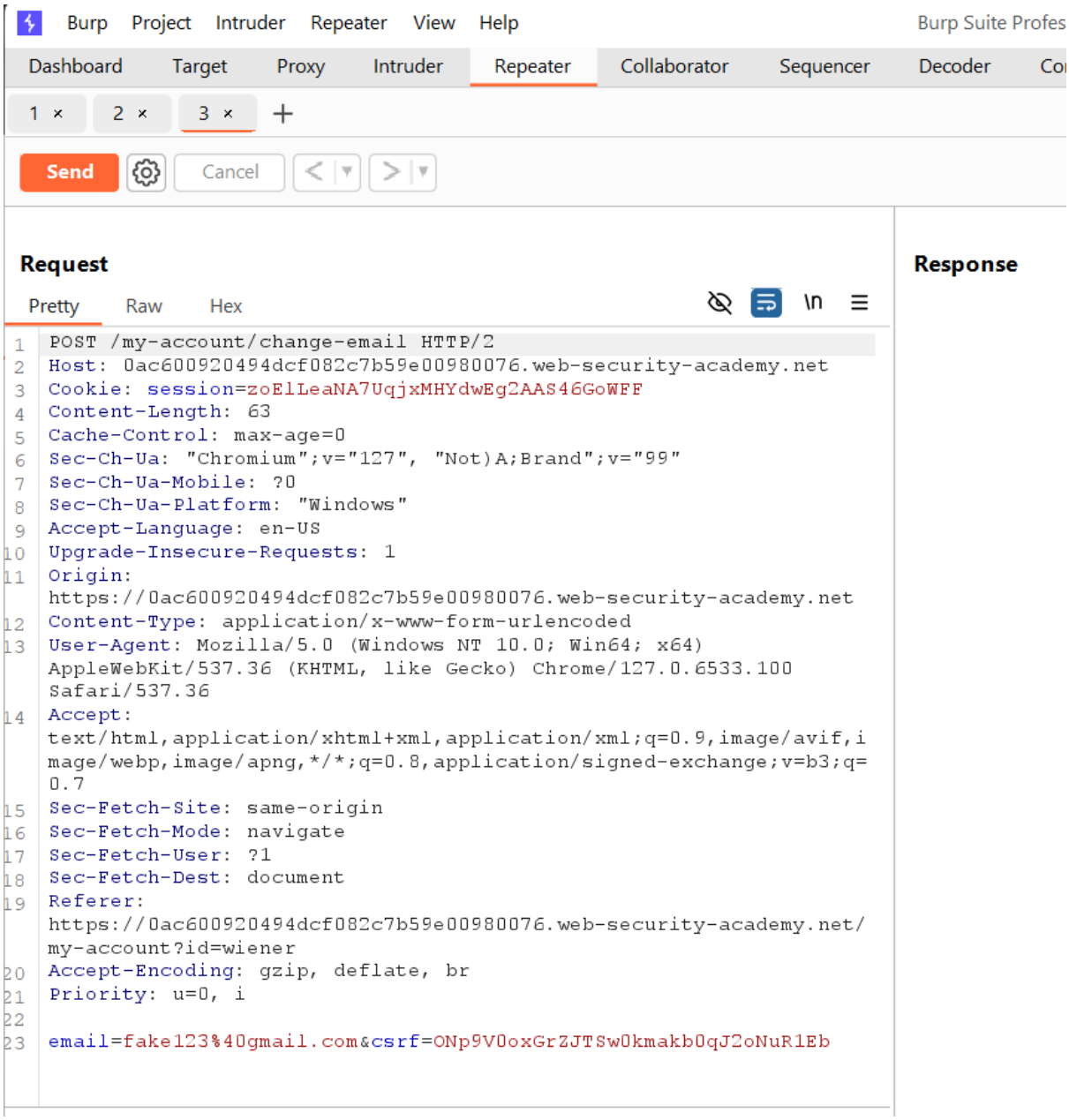
Here we can see a box to enter a email account to change, let's try entering any random email id

I'm entering a fake email id,

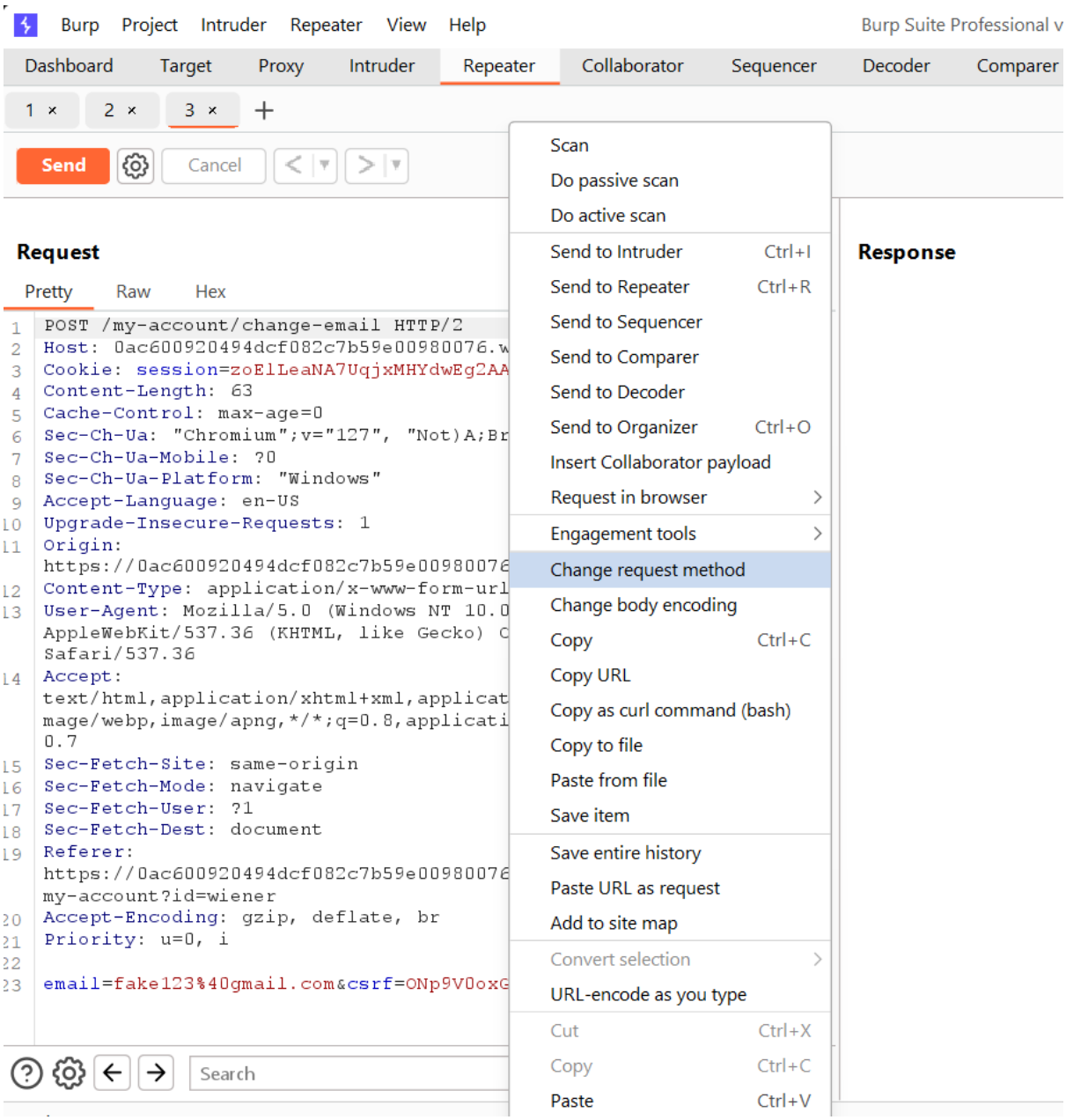


Here we can see the email is changed, but it doesn't upload to the exploit server.

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

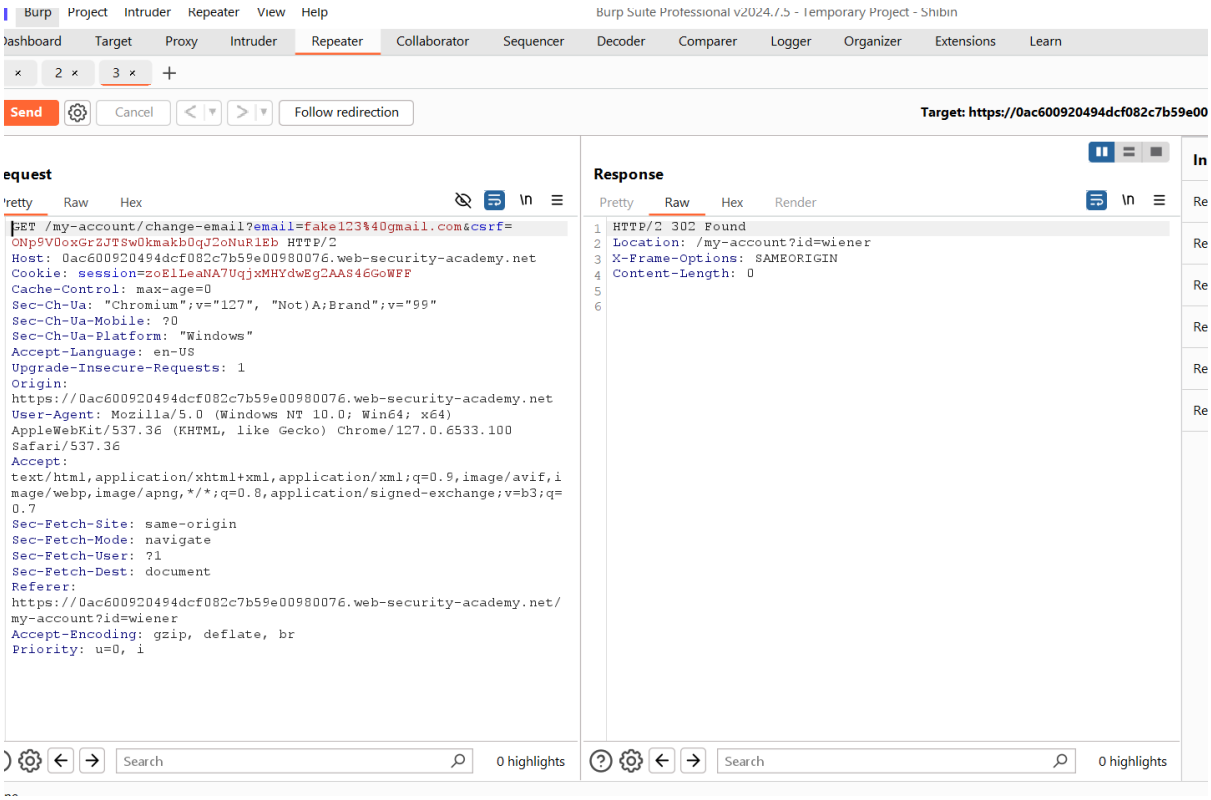


On the description it is described that this web prevents only to certain types of requests. So let's change the request.



Let’s see what is the response,

If it is 302 it explains the web is working,



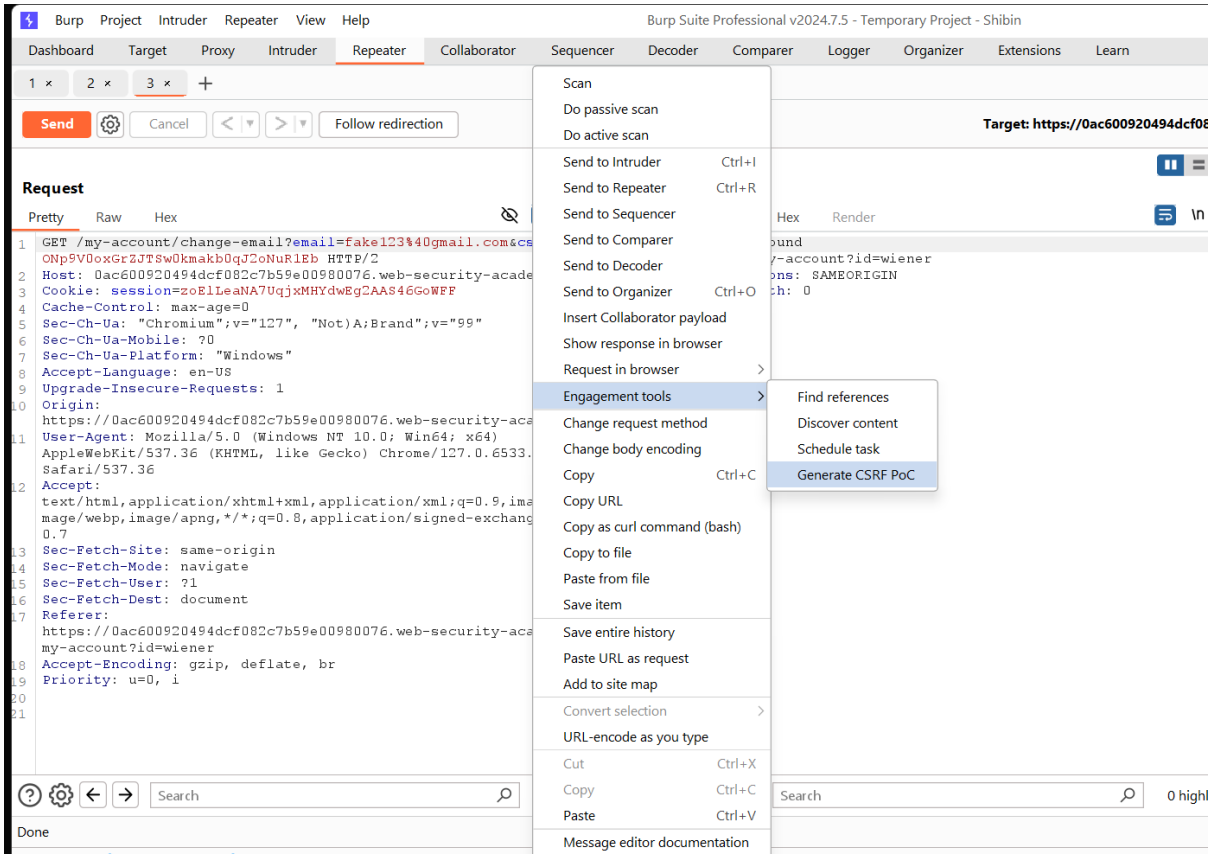
Email was updated and there was a request and a 302 Response on Burp.

This tells us that the is no defenses on switching a users 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 users email address if the given user click on the exploit servers link.

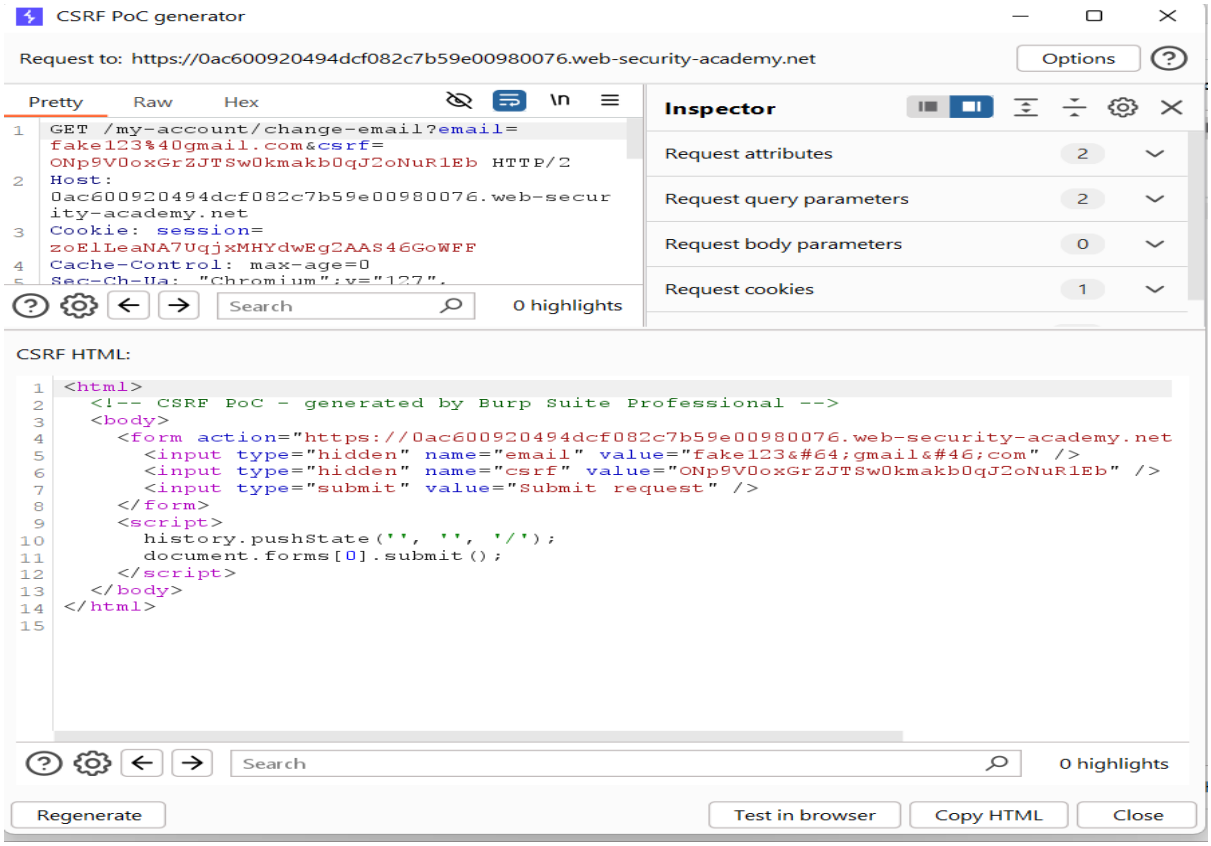
Soo in professional version there is a option engagement tool which is available to create **CSRF Poc(proof of concept).** Or it is also called to create the payload.

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



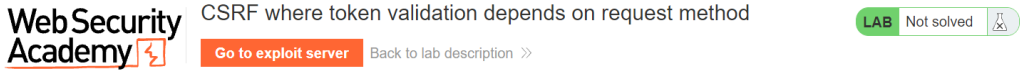
Open options in that window and make sure the “include auto-submit script” option is turned on.

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

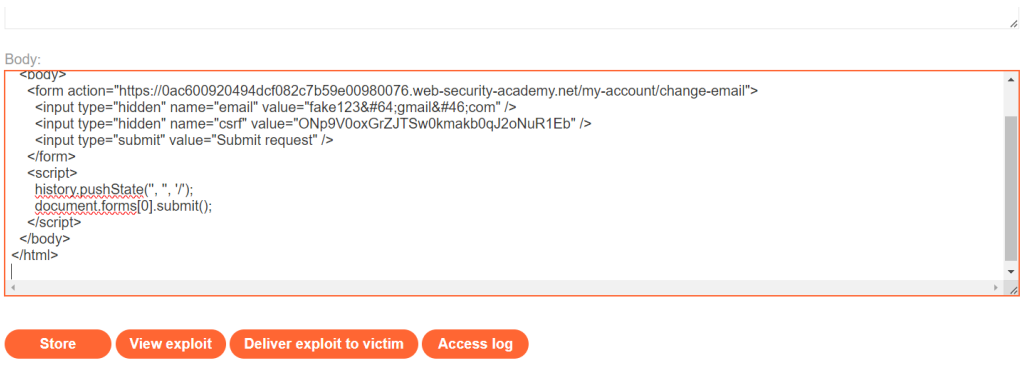


After this open our lab and there we can see a option of **“Go To Exploit Server”**

On the top.



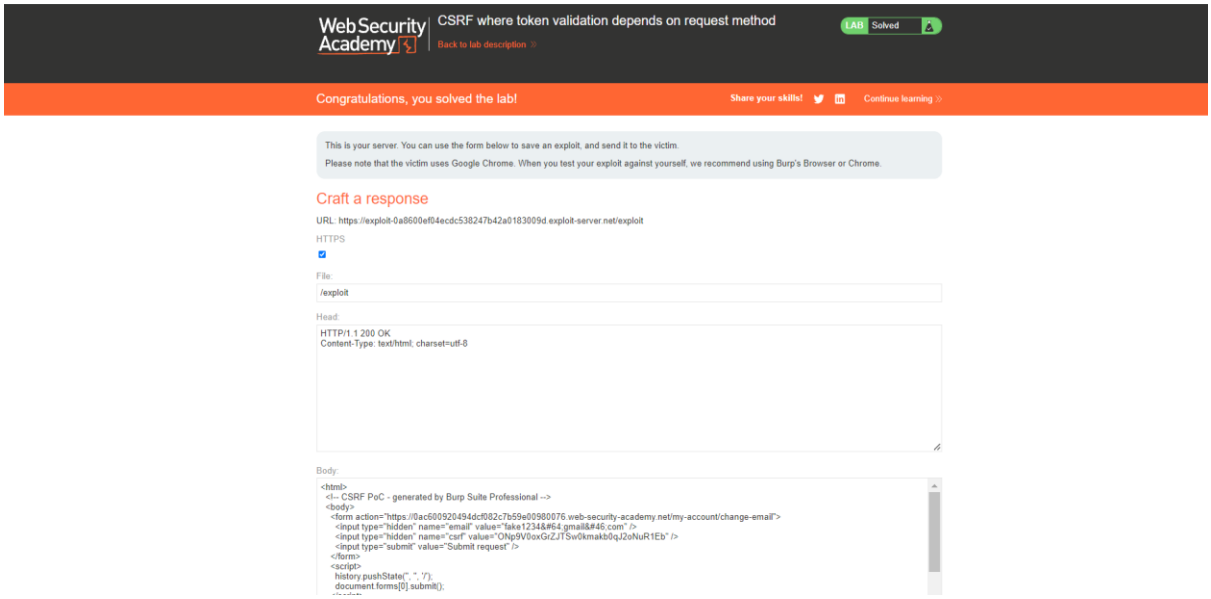
Go to exploit server and paste the copied HTML on the Body.



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

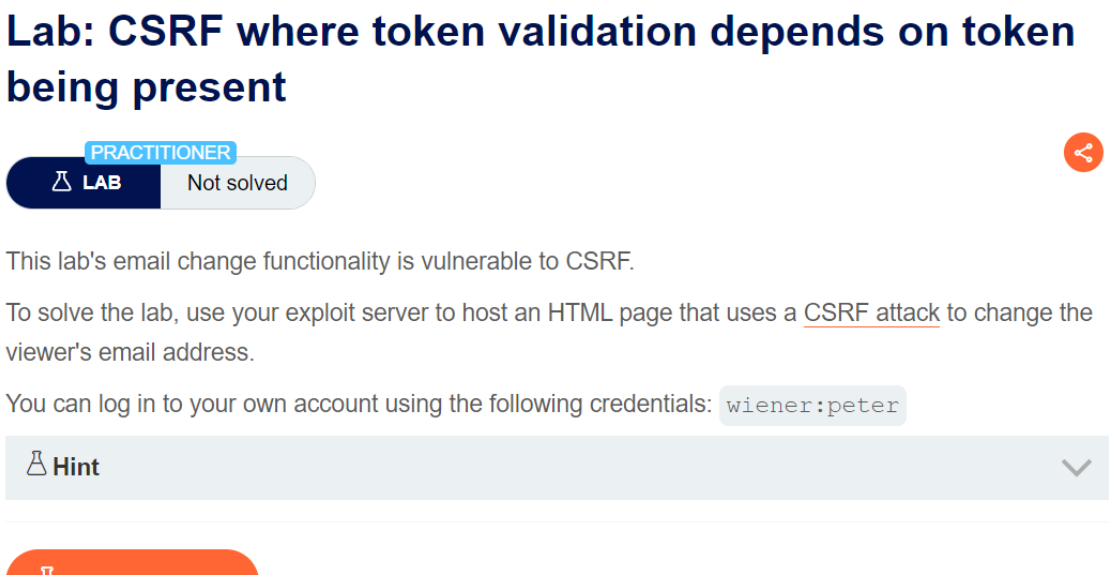
Here the exploit is delivered to the victim, before only we could see the result, now the all server got exploited.

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



Hurray we completed the lab 😊

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

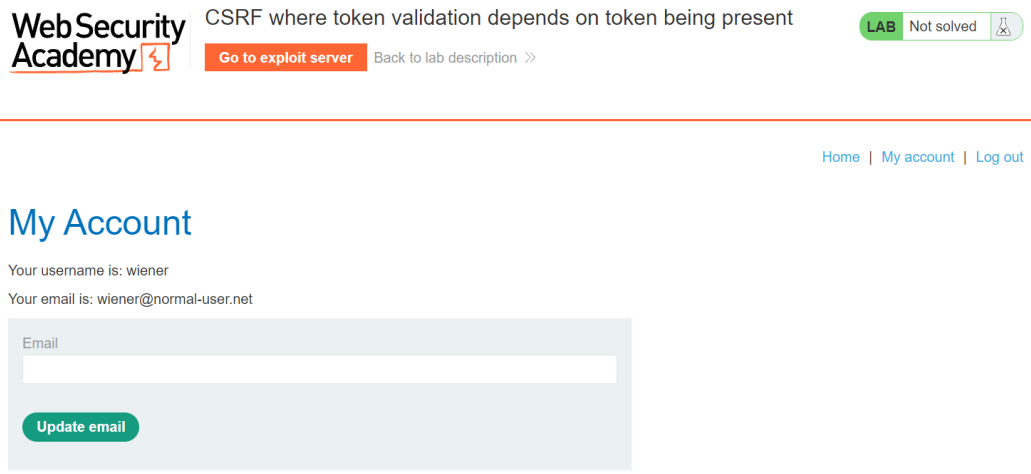


This is a CSRF vulnerable lab available on the portswigger. The description describes there is a vulnerability present in the email changing functionality.

And they also given credential details to login,

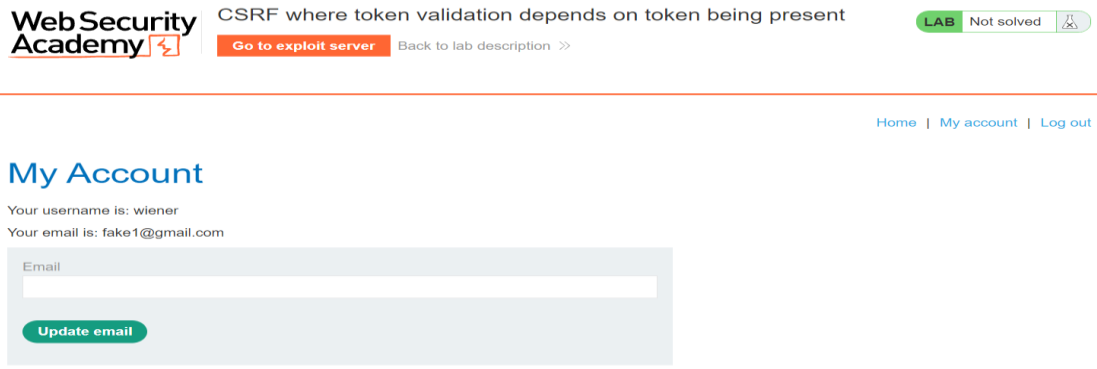
So let’s access the lab.

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



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,



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.**

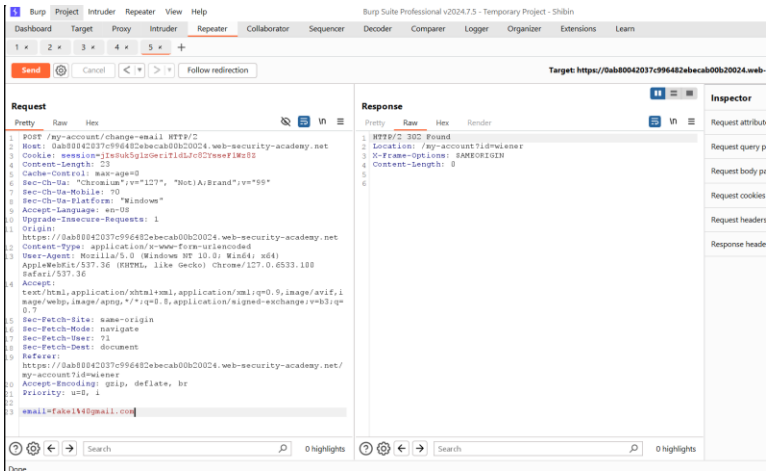
```
POST /my-account/change-email HTTP/2
Host: 0ab80042037c996482ebecab00b20024.web-security-academy.net
Cookie: session=jIsSuk5glzGeriTldLJc82YsseF1Wz8Z
Content-Length: 61
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin:
https://0ab80042037c996482ebecab00b20024.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
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,i
mage/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://0ab80042037c996482ebecab00b20024.web-security-academy.net/
my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

email=fake1%40gmail.com&csrf=AMlOtbalaxawj9lycaugnMHWRQJHfDYk
```

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

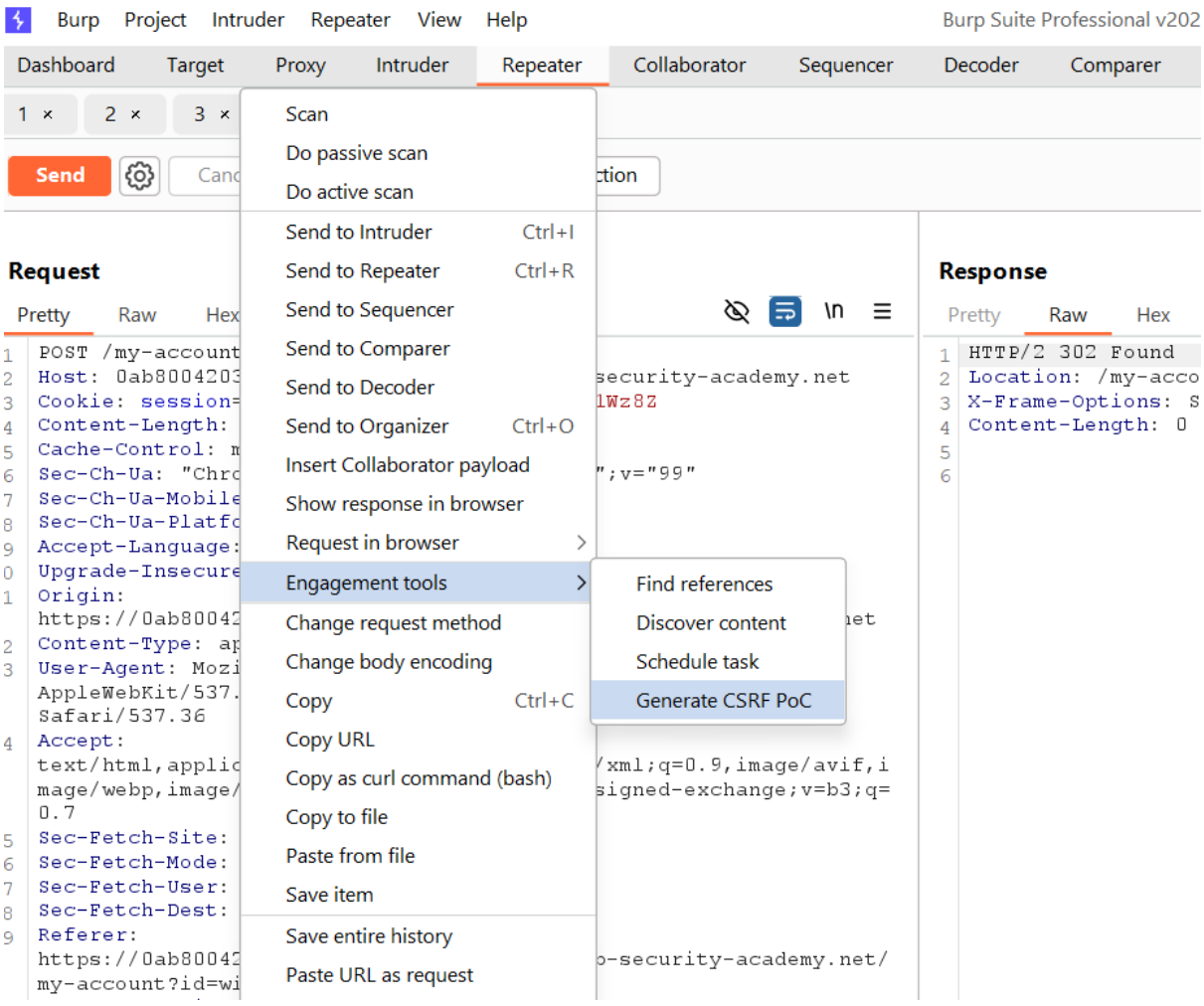
Email was updated and there was a request and a 302 Response on Burp.

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



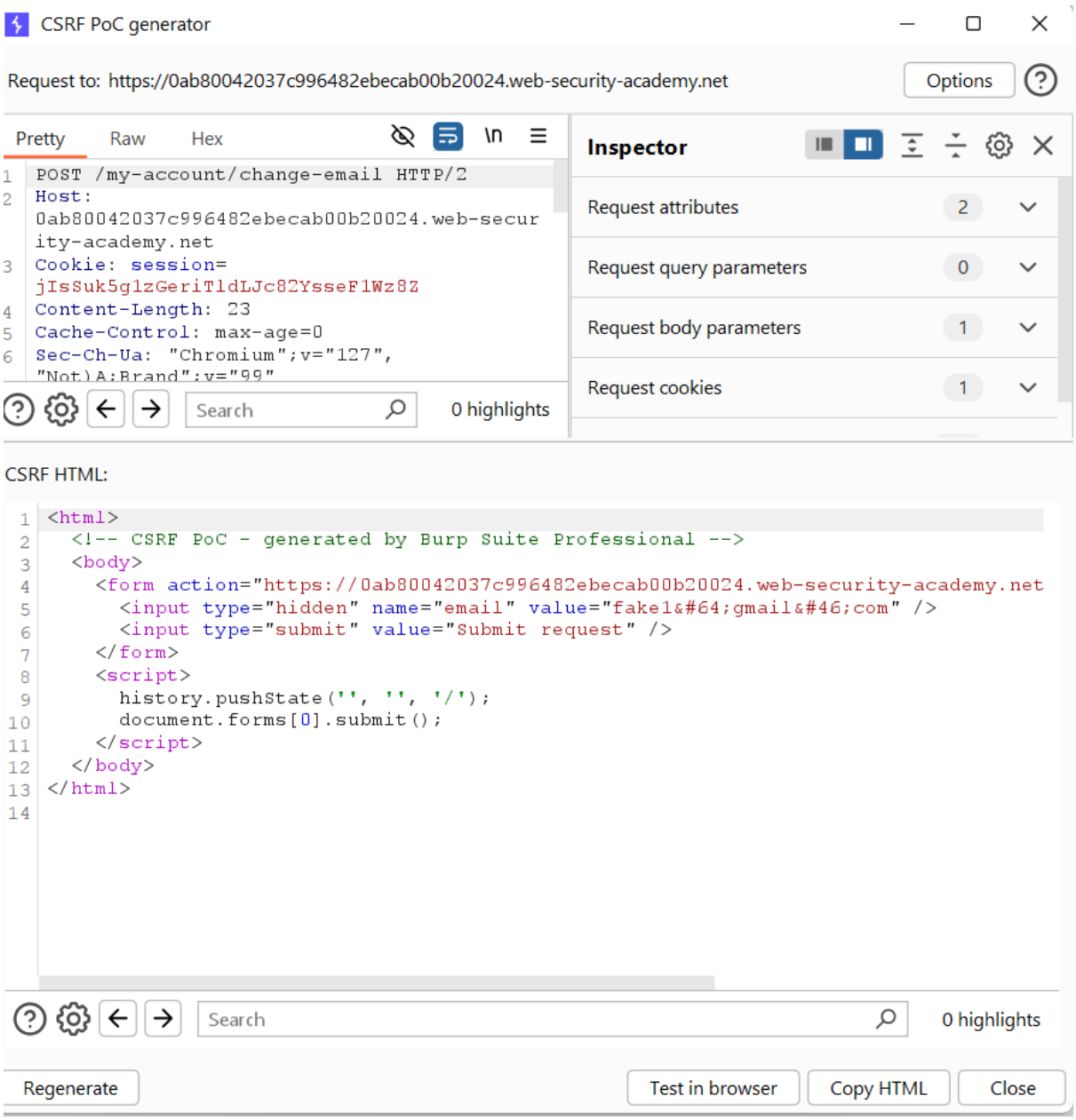
It is responding, Soo in professional version there is a option engagement tool which is available to create **CSRF Poc(proof of concept).** Or it is also called to create the payload.

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



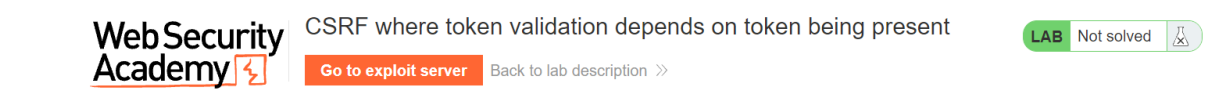
Open options in that window and make sure the “include auto-submit script” option is turned on.

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



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.

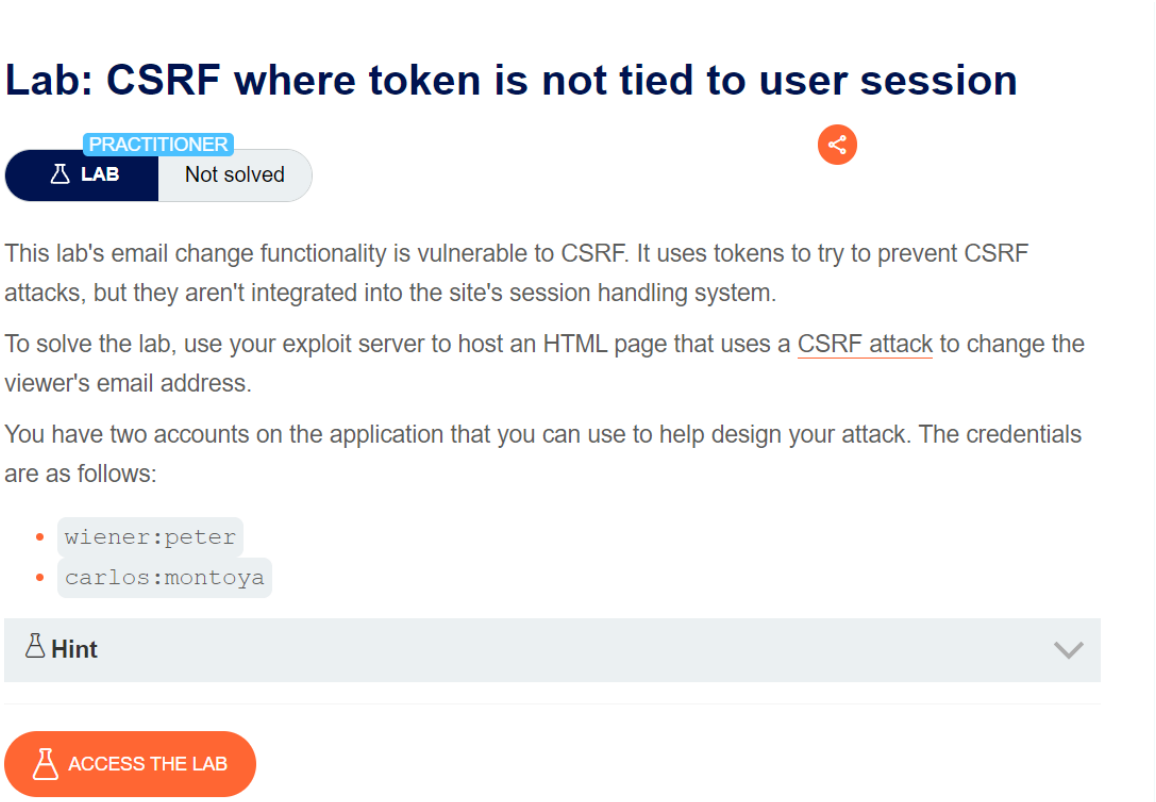
Change the email from the code and Later click “**Store**” and **“Deliver exploit to victim”** .

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

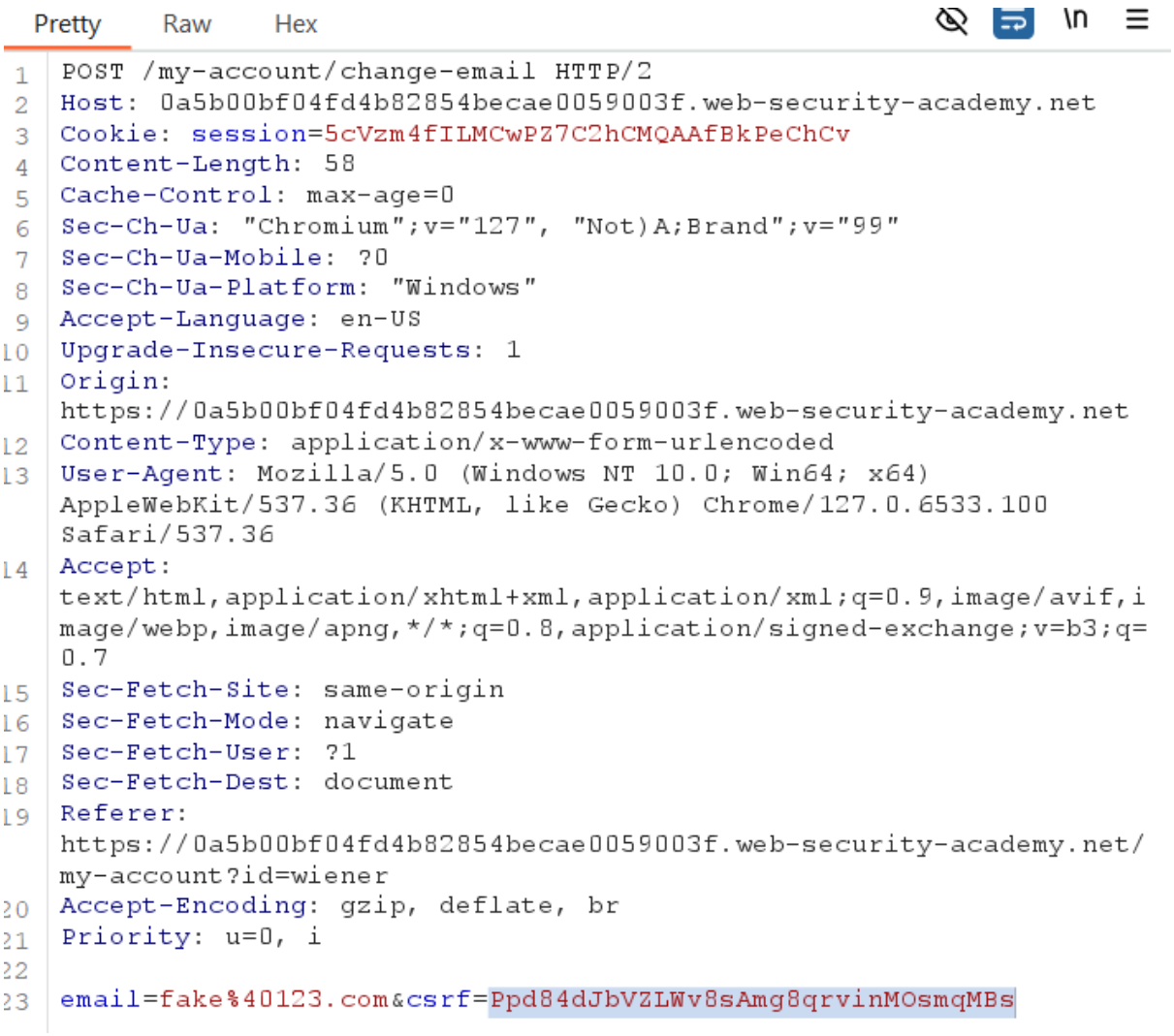


This lab have CSRF vulnerability on the email changing functionality. It is mentioned that it uses token address to prevent attacks. So they had given 2 accounts and we have to login Wiener account in 1 browser and Carlos account in another browser.

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

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

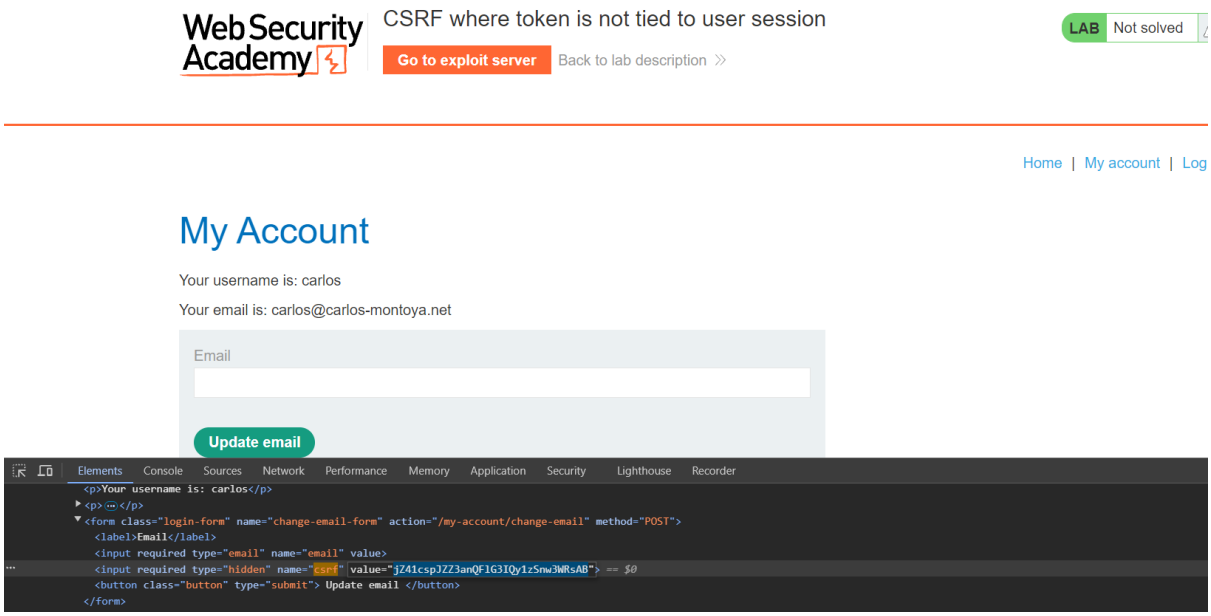
Enter any fake mail id and intercept the request in **burp professional.**



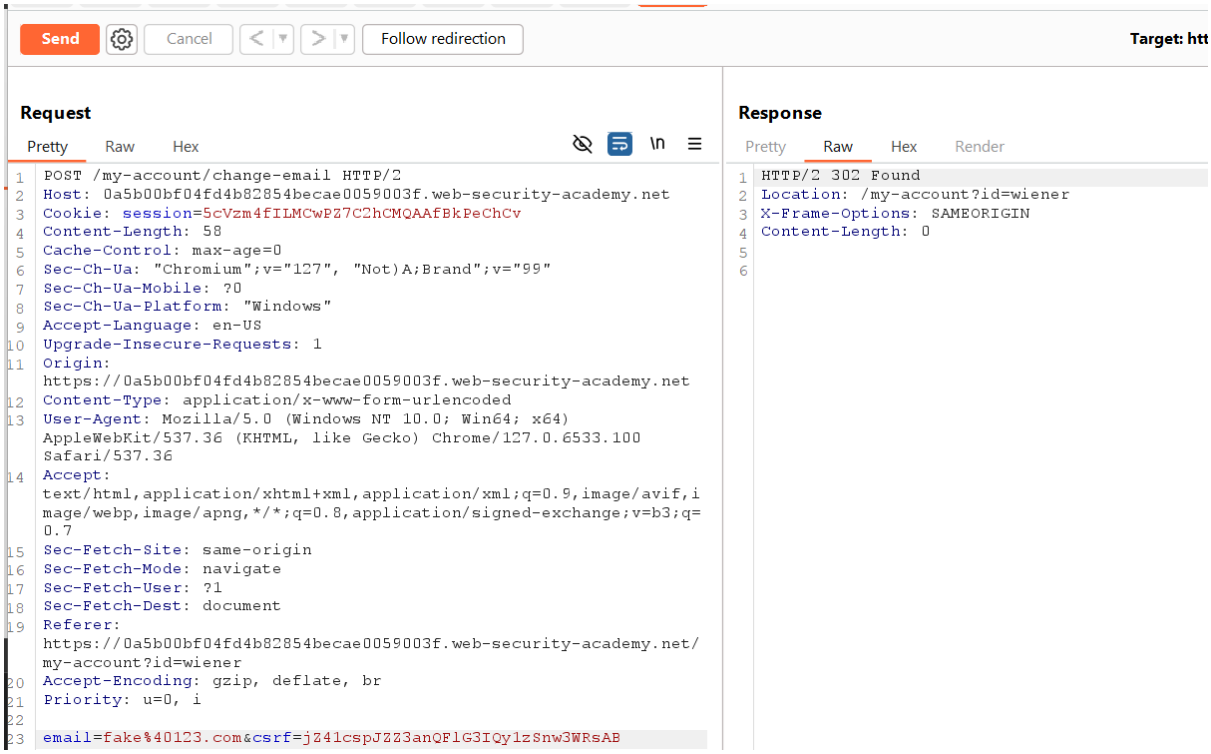
```
Pretty  Raw  Hex
1  POST /my-account/change-email HTTP/2
2  Host: 0a5b00bf04fd4b82854becae0059003f.web-security-academy.net
3  Cookie: session=5cVzm4fILMCwPZ7C2hCMQAAfBkPeChCv
4  Content-Length: 58
5  Cache-Control: max-age=0
6  Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
7  Sec-Ch-Ua-Mobile: ?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,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=Ppd84dJbVZLWv8sAmg8qrvinMOsmqMBs
```

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.

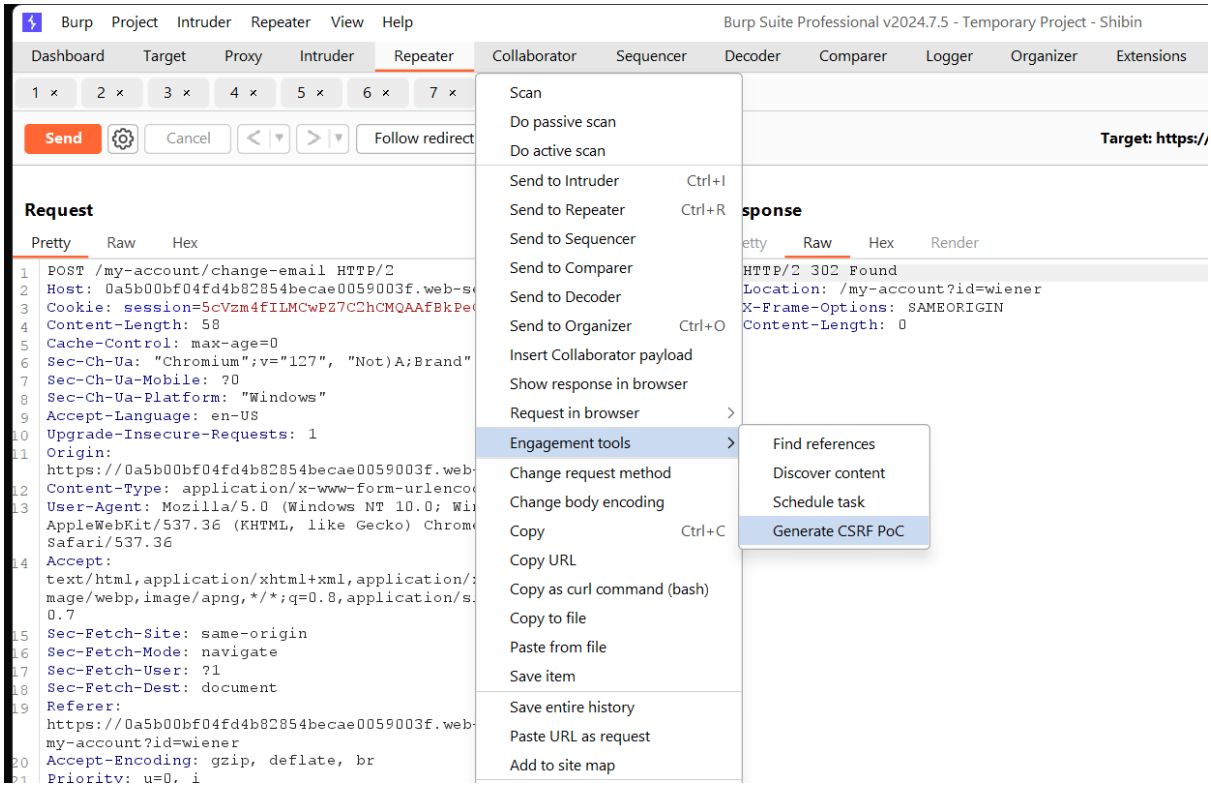
It is already showed on the description that csrf token is protecting, so let’s swap the token with 2 users.



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

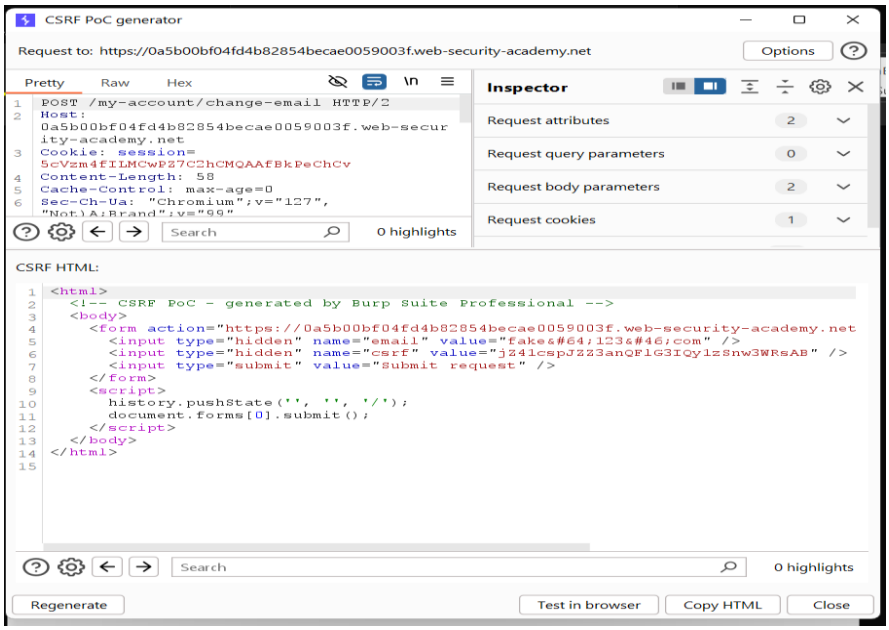


After swapping the response shows it is accepted, so generate CSRF PoC from **Burp suite professional.**



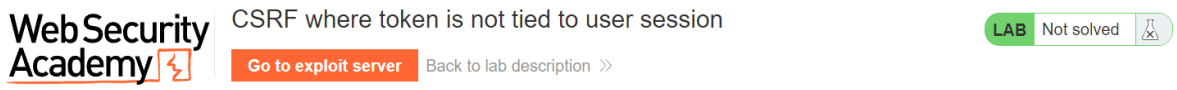
Open options in that window and make sure the “include auto-submit script” option is turned on.

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

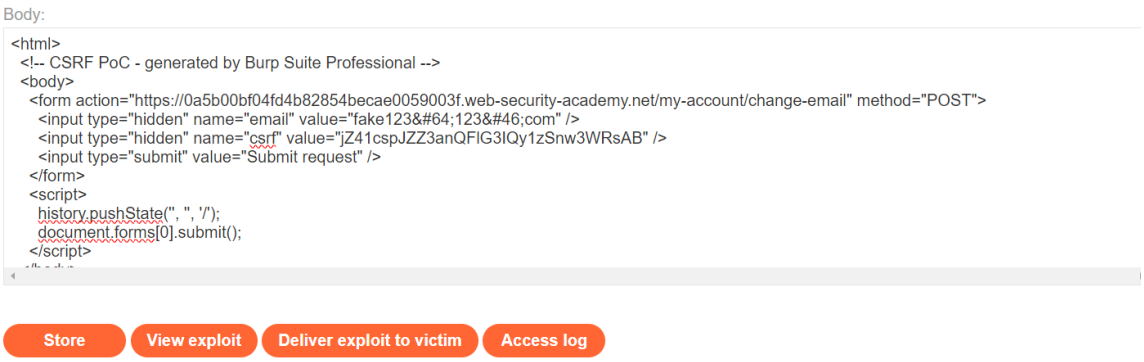


After this open our lab and there we can see a option of **“Go To Exploit Server”**

On the top.



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



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.