Tristan Gomez

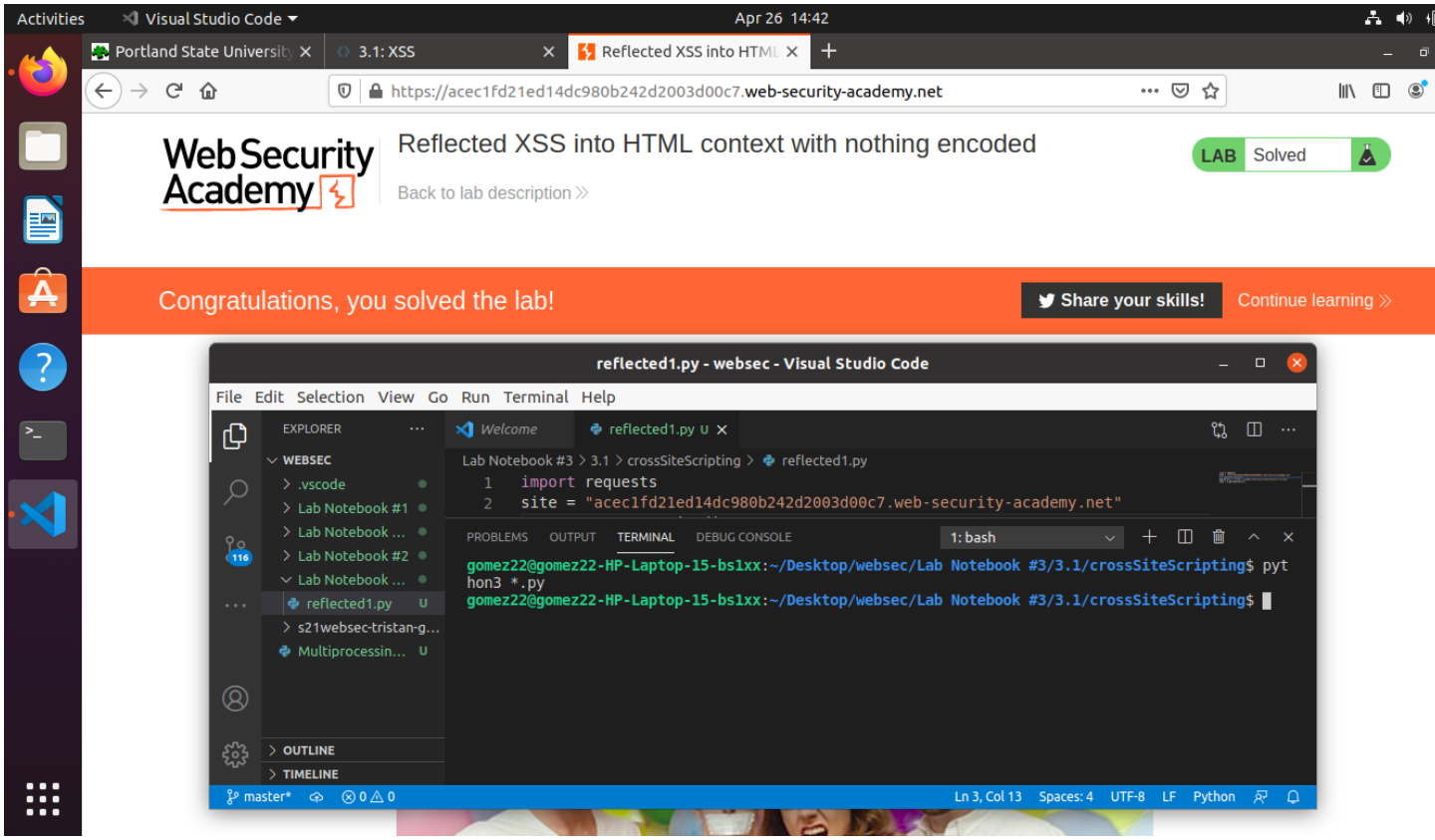
CS 595

Lab Notebook #3

3.1 XSS

1) Cross-site-scripting/reflected(1)

This lab has a vulnerability where I can inject JavaScript into the search functionality of the lab website. The site returns the search term to the HTML content of the results page, so if I insert a JavaScript function, the browser will execute that script when the results page is loaded. To remediate this, the site should filter <script> tags from the search functionality.

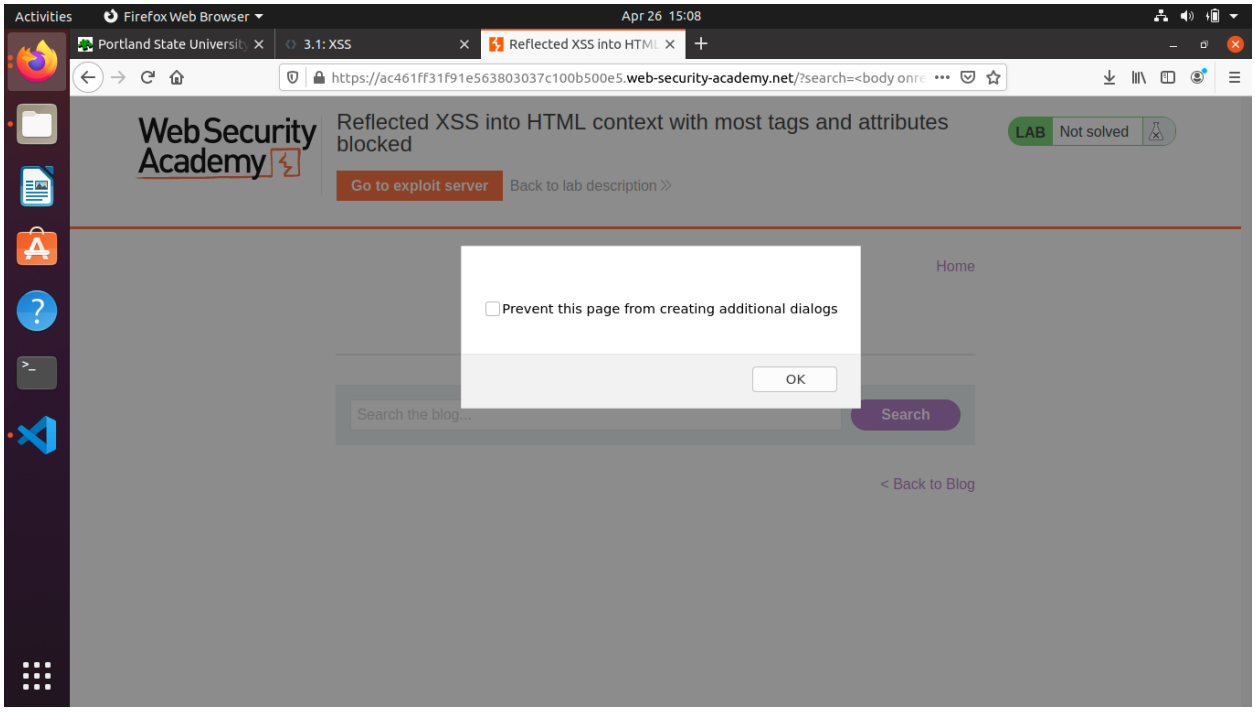


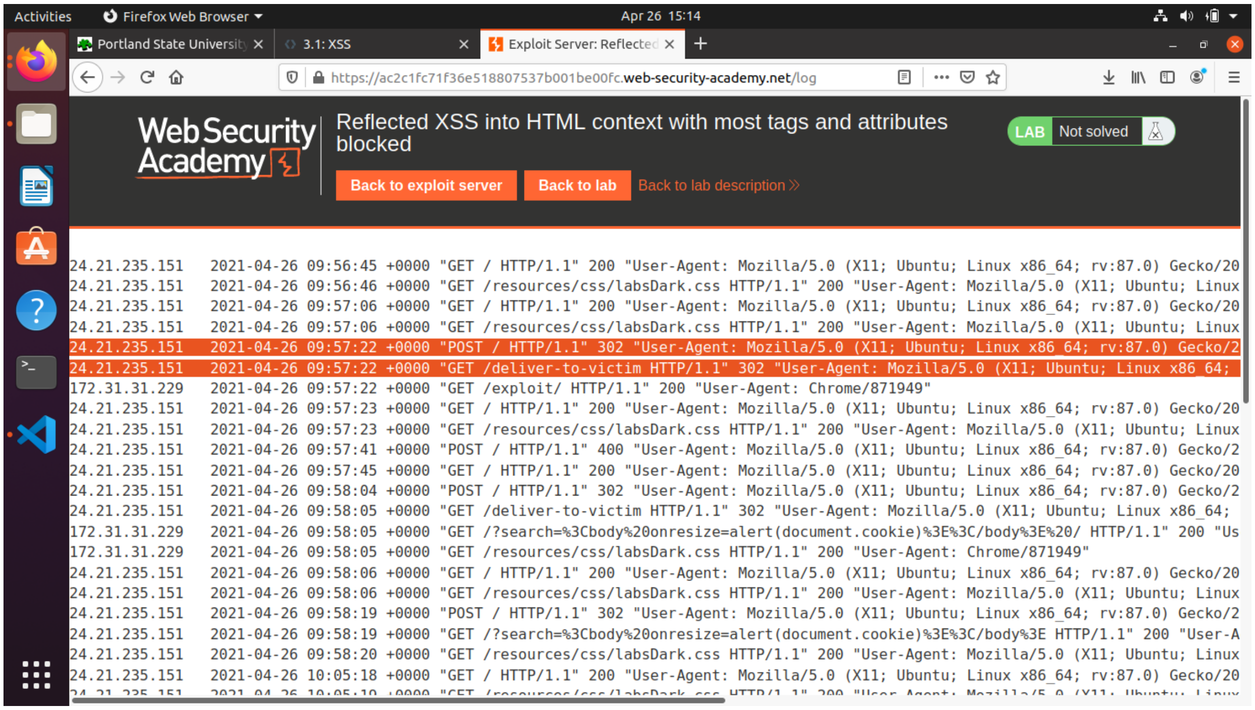
2) Cross-site-scripting/reflected(2)

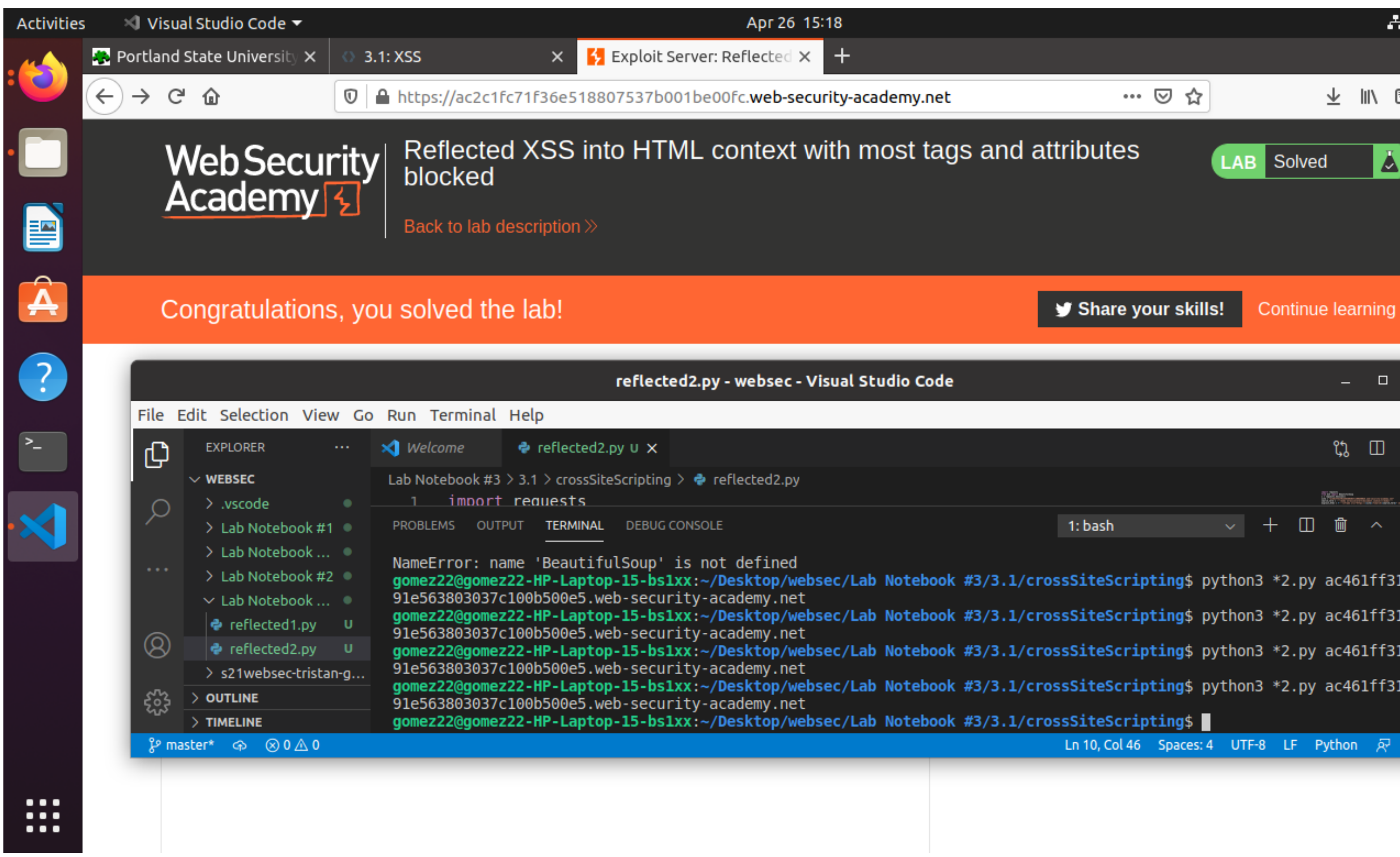
-This lab is vulnerable to XSS, but the <script> tag is filtered out of the search query. As an attacker, I experimented with using <body> tags and was able to get events to run. I then deliver the payload to solve the lab in an <iframe> tag. To remediate this vulnerability, the developers need to be sure to sanitize the input and not allow any event attributes to be injected by the client.

List which window event attributes are allowed for your lab notebook.

-The window events that are allowed are “onresize” and “onstorage”.



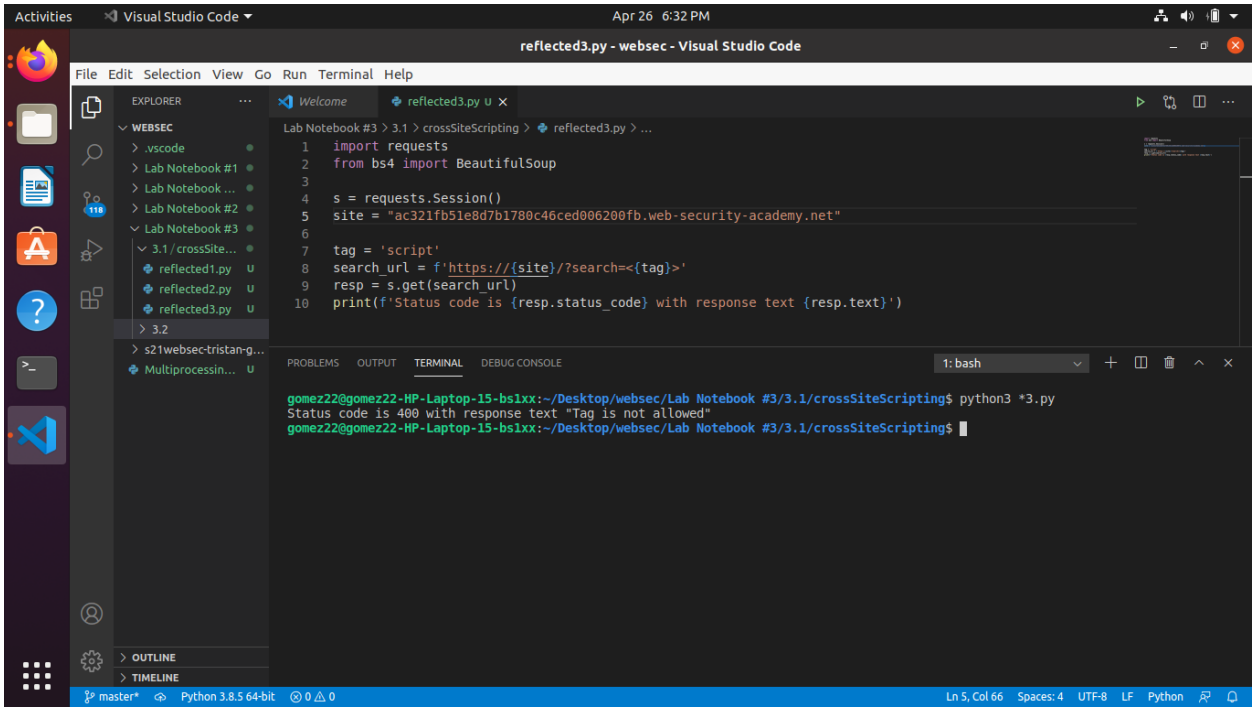

Activities
Firefox Web Browser
Apr 26 15:14
Portland State Universit
3.1: XSS
Exploit Server: Reflected
https://ac2c1fc71f36e518807537b001be00fc.web-security-academy.net/log
Web Security Academy
Reflected XSS into HTML context with most tags and attributes blocked
LAB Not solved
Back to exploit server
Back to lab
Back to lab description »
24.21.235.151 2021-04-26 09:56:45 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20
24.21.235.151 2021-04-26 09:56:46 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
24.21.235.151 2021-04-26 09:57:06 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20
24.21.235.151 2021-04-26 09:57:06 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
24.21.235.151 2021-04-26 09:57:22 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/2
24.21.235.151 2021-04-26 09:57:22 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
172.31.31.229 2021-04-26 09:57:22 +0000 "GET /exploit/ HTTP/1.1" 200 "User-Agent: Chrome/871949"
24.21.235.151 2021-04-26 09:57:23 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20
24.21.235.151 2021-04-26 09:57:23 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
24.21.235.151 2021-04-26 09:57:41 +0000 "POST / HTTP/1.1" 400 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/2
24.21.235.151 2021-04-26 09:57:45 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20
24.21.235.151 2021-04-26 09:58:04 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/2
24.21.235.151 2021-04-26 09:58:05 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
172.31.31.229 2021-04-26 09:58:05 +0000 "GET /?search=%3Cbody%20onresize=alert(document.cookie)%3E%3C/body%3E%20/ HTTP/1.1" 200 "Us
172.31.31.229 2021-04-26 09:58:05 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Chrome/871949"
24.21.235.151 2021-04-26 09:58:06 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20
24.21.235.151 2021-04-26 09:58:06 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
24.21.235.151 2021-04-26 09:58:19 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/2
24.21.235.151 2021-04-26 09:58:19 +0000 "GET /?search=%3Cbody%20onresize=alert(document.cookie)%3E%3C/body%3E HTTP/1.1" 200 "User-A
24.21.235.151 2021-04-26 09:58:20 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
24.21.235.151 2021-04-26 10:05:18 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20


Activities
Visual Studio Code
Apr 26 15:18
Portland State Universit
3.1: XSS
Exploit Server: Reflected
https://ac2c1fc71f36e518807537b001be00fc.web-security-academy.net
Web Security Academy
Reflected XSS into HTML context with most tags and attributes blocked
LAB Solved
Back to lab description »
Congratulations, you solved the lab!
Share your skills!
Continue learning
reflected2.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBSEC
.vscode
Lab Notebook #1
Lab Notebook ...
Lab Notebook #2
Lab Notebook ...
reflected1.py
reflected2.py
s21websec-tristan-g...
OUTLINE
TIMELINE
Welcome
reflected2.py
Lab Notebook #3 > 3.1 > crossSiteScripting > reflected2.py
1 import requests
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: bash
NameError: name 'BeautifulSoup' is not defined
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *2.py ac461ff31
91e563803037c100b500e5.web-security-academy.net
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *2.py ac461ff31
91e563803037c100b500e5.web-security-academy.net
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *2.py ac461ff31
91e563803037c100b500e5.web-security-academy.net
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *2.py ac461ff31
91e563803037c100b500e5.web-security-academy.net
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$
master*
Ln 10, Col 46 Spaces: 4 UTF-8 LF Python

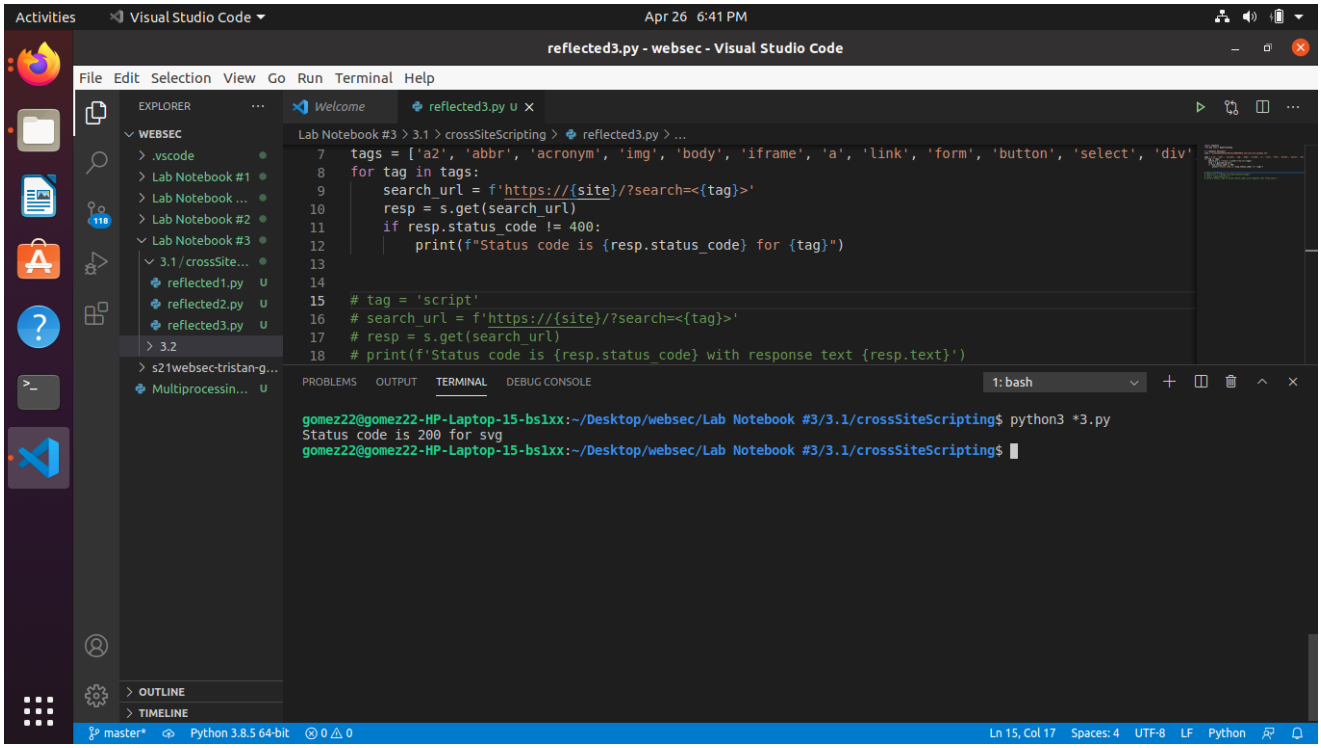
Cross-site-scripting/reflected (3)

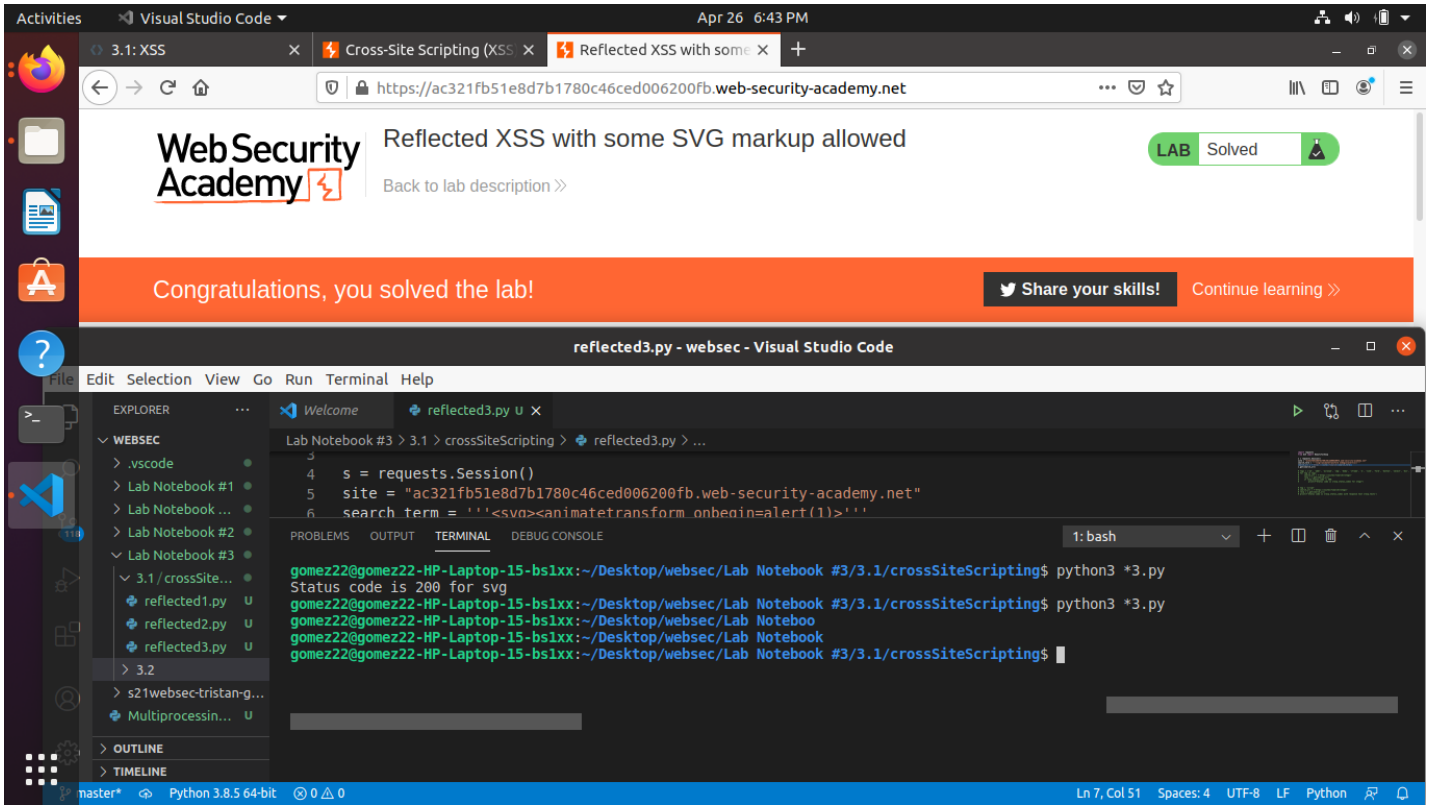
-The site has a vulnerability in its search functionality that allows for cross site scripting. This lab filters out the <script> tag along with many others. However, the svg and animatetransform tags are allowed, which lets me deliver my payload and solve the level. To remediate this, the site developers should sanitize the search input and check for every possible tag.

Show the HTTP status code and response text obtained.



Show the list of tags that are not filtered.


Activities
Visual Studio Code
Apr 26 6:41 PM
reflected3.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBSEC
.vscode
Lab Notebook #1
Lab Notebook ...
Lab Notebook #2
Lab Notebook #3
3.1 / crossSite...
reflected1.py
reflected2.py
reflected3.py
3.2
s21websec-tristan-g...
Multiprocessin...
Welcome
reflected3.py
Lab Notebook #3 > 3.1 > crossSiteScripting > reflected3.py > ...
tags = ['a2', 'abbr', 'acronym', 'img', 'body', 'iframe', 'a', 'link', 'form', 'button', 'select', 'div'
for tag in tags:
search_url = f'https://{site}/?search=<{tag}>'
resp = s.get(search_url)
if resp.status_code != 400:
print(f"Status code is {resp.status_code} for {tag}")
# tag = 'script'
# search_url = f'https://{site}/?search=<{tag}>'
# resp = s.get(search_url)
# print(f'Status code is {resp.status_code} with response text {resp.text}')
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: bash
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *3.py
Status code is 200 for svg
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$
OUTLINE
TIMELINE
master*
Python 3.8.5 64-bit
Ln 15, Col 17
Spaces: 4
UTF-8
LF
Python


Activities
Visual Studio Code
Apr 26 6:43 PM
3.1: XSS
Cross-Site Scripting (XSS
Reflected XSS with some
https://ac321fb51e8d7b1780c46ced006200fb.web-security-academy.net
Web Security Academy
Reflected XSS with some SVG markup allowed
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
reflected3.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBSEC
.vscode
Lab Notebook #1
Lab Notebook ...
Lab Notebook #2
Lab Notebook #3
3.1 / crossSite...
reflected1.py
reflected2.py
reflected3.py
3.2
s21websec-tristan-g...
Multiprocessin...
Welcome
reflected3.py
Lab Notebook #3 > 3.1 > crossSiteScripting > reflected3.py > ...
s = requests.Session()
site = "ac321fb51e8d7b1780c46ced006200fb.web-security-academy.net"
search_term = '''<svg><animatetransform onbegin=alert(1)>'''
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: bash
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *3.py
Status code is 200 for svg
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$ python3 *3.py
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Noteboo
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting$
OUTLINE
TIMELINE
master*
Python 3.8.5 64-bit
Ln 7, Col 51
Spaces: 4
UTF-8
LF
Python

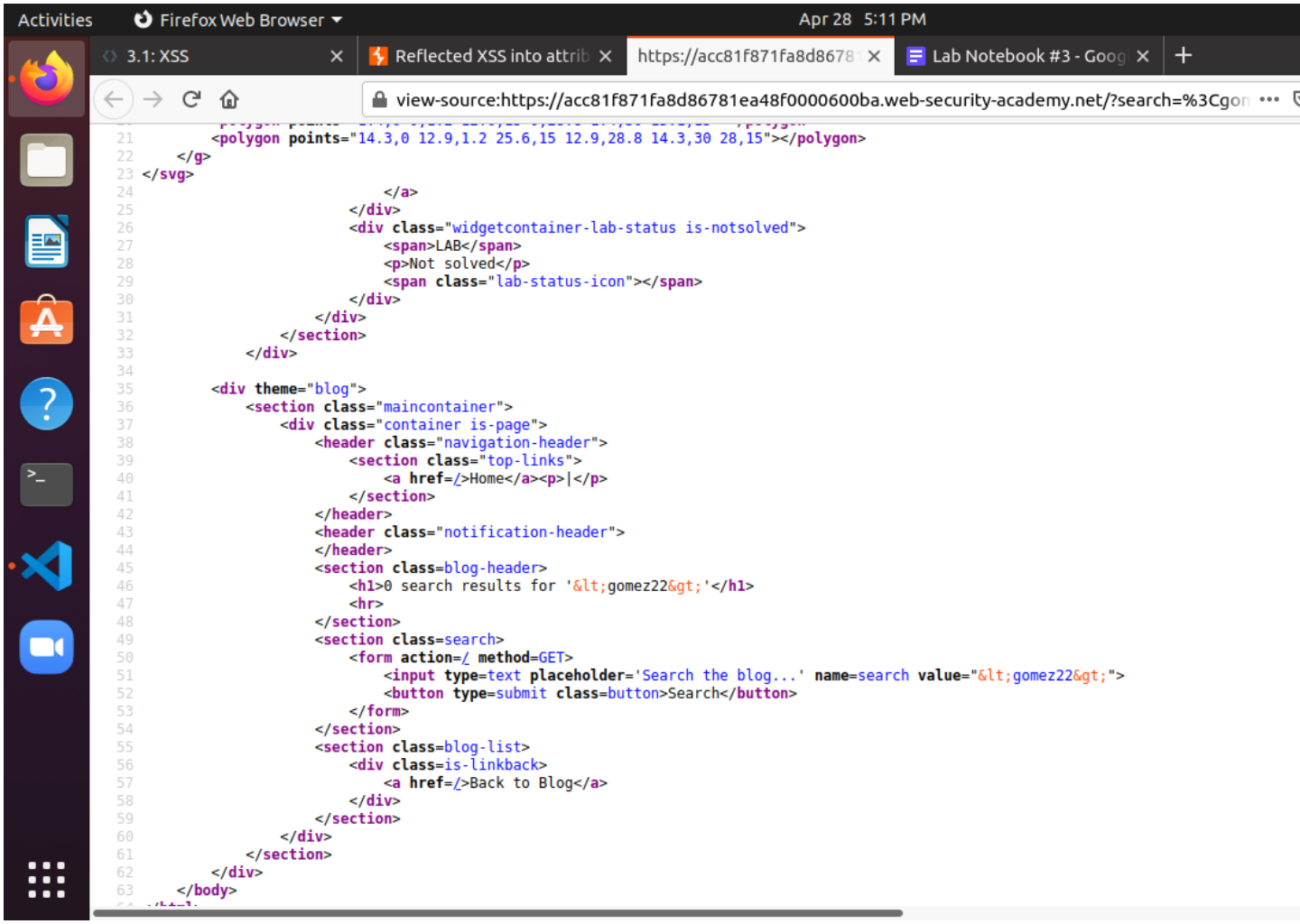
Cross-site-scripting/reflected (4)

-This lab prevents XSS attacks through the search field, but it is vulnerable to injecting code into a html tag on the page. As an attacker, I can use this vulnerability to insert an alert() to solve the level. To remediate this attack, the developers should encode content inside of tags before rendering them on the DOM.

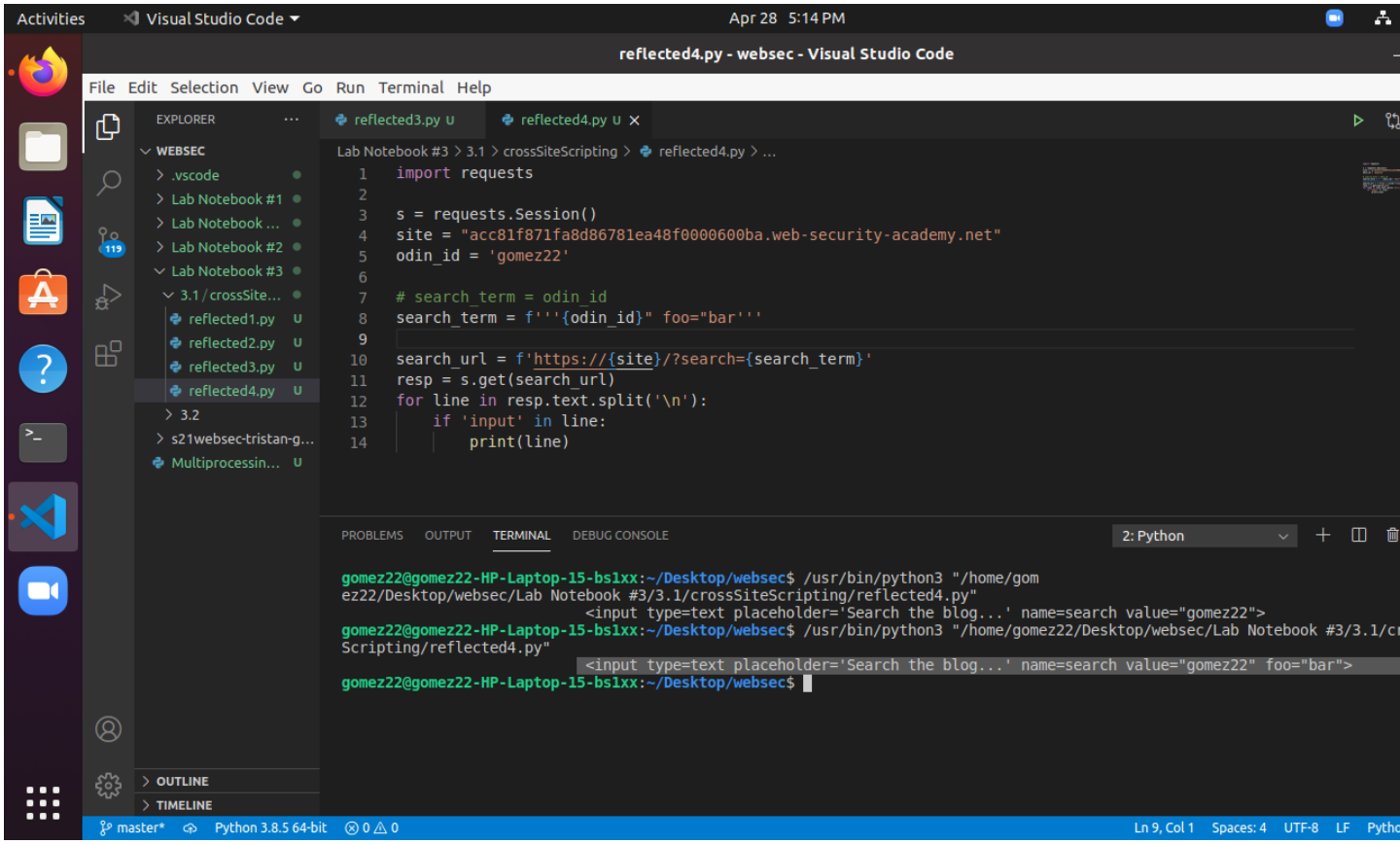
Show the part of the source demonstrating that the content has been HTML-encoded in the HTML context

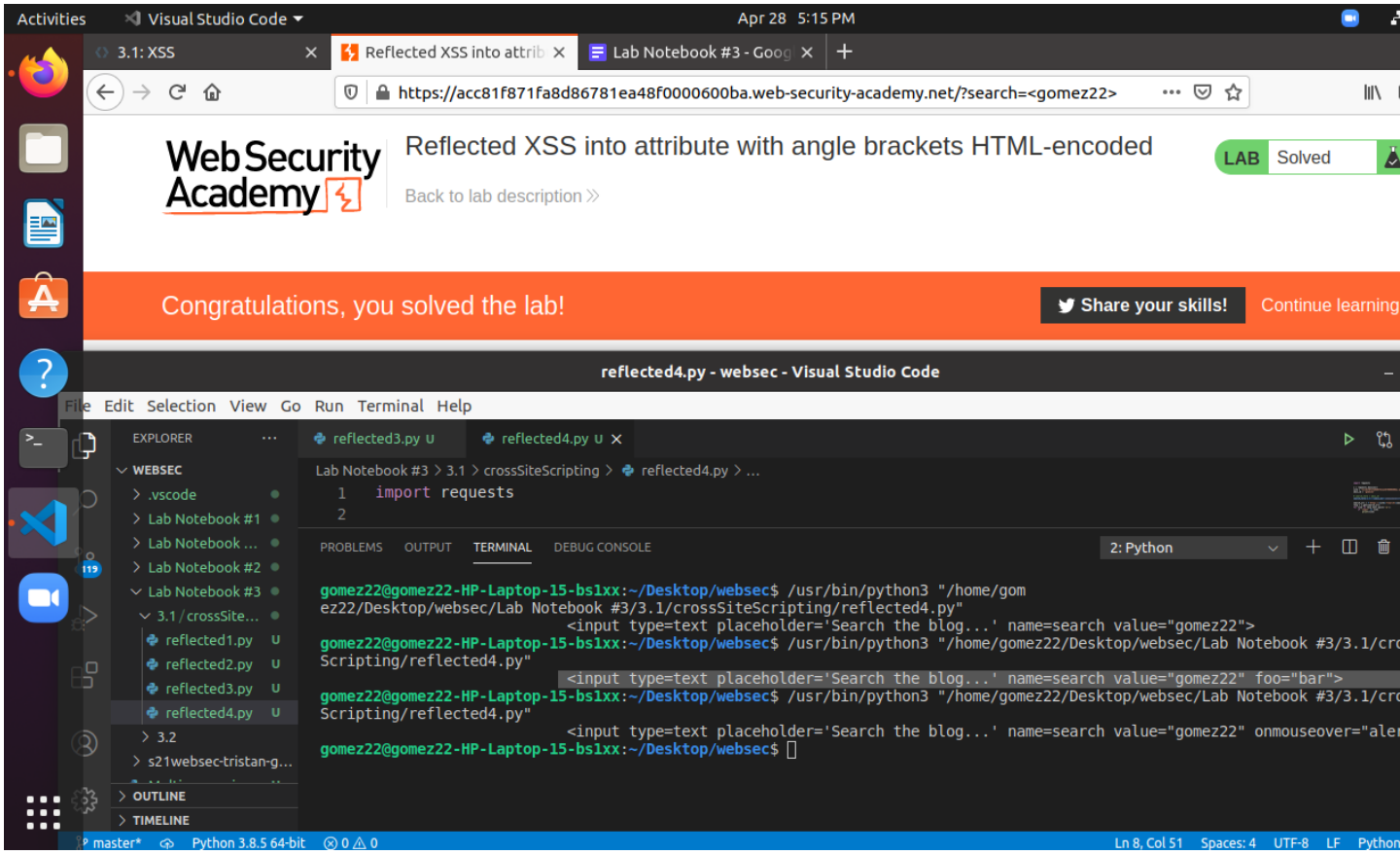
Show the part of the source demonstrating that the reflected content also appears within an HTML tag's context

-The below screenshot contains the answer to both above questions.



Take a screenshot of the output for your lab notebook

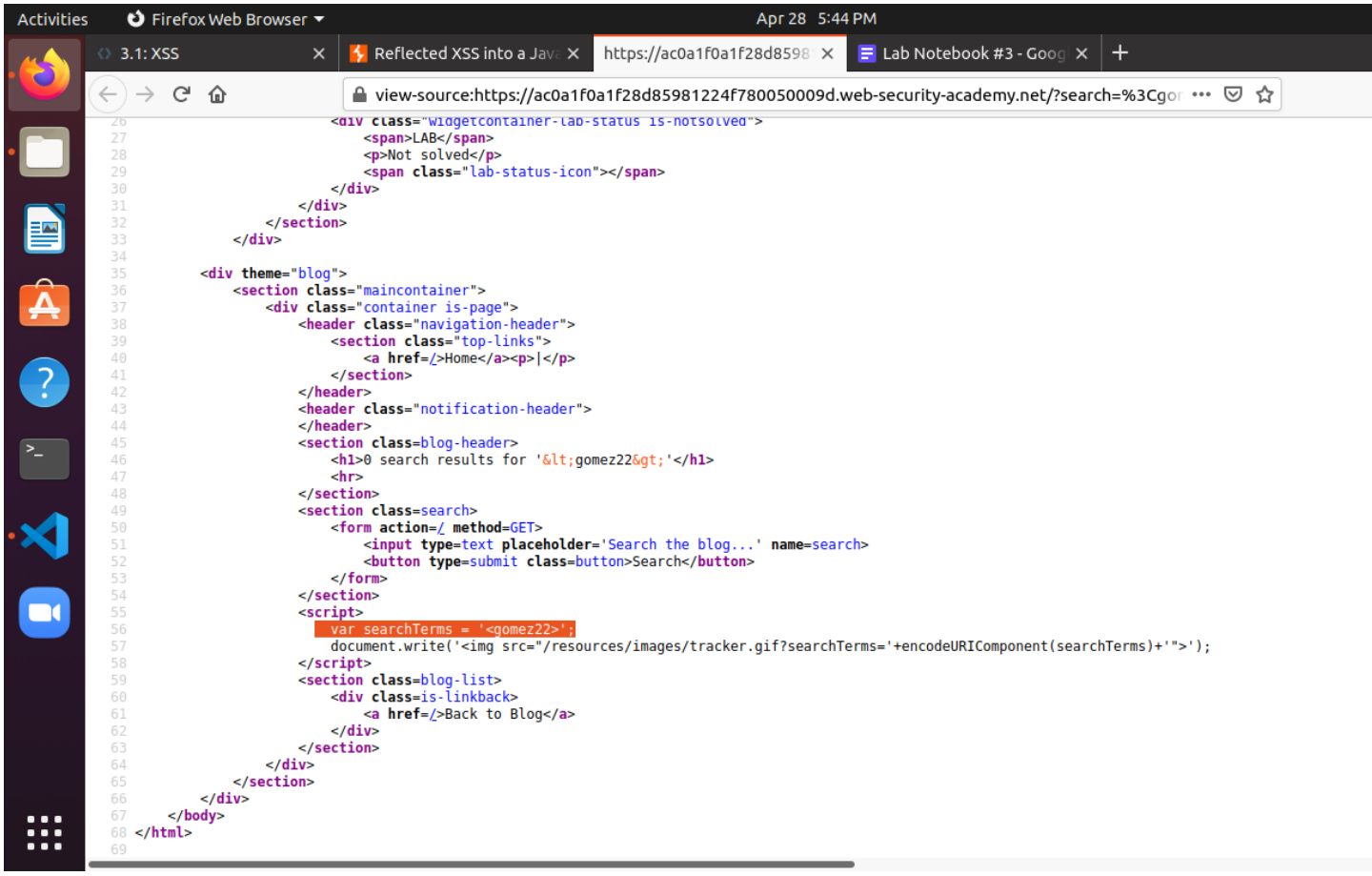

Activities
Visual Studio Code
Apr 28 5:14 PM
reflected4.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBSEC
.vscode
Lab Notebook #1
Lab Notebook ...
Lab Notebook #2
Lab Notebook #3
3.1 / crossSite...
reflected1.py U
reflected2.py U
reflected3.py U
reflected4.py U
3.2
s21websec-tristan-g...
Multiprocessin... U
OUTLINE
TIMELINE
reflected3.py U
reflected4.py U
Lab Notebook #3 > 3.1 > crossSiteScripting > reflected4.py > ...
1 import requests
2
3 s = requests.Session()
4 site = "acc81f871fa8d86781ea48f0000600ba.web-security-academy.net"
5 odin_id = 'gomez22'
6
7 # search_term = odin_id
8 search_term = f'''{odin_id}" foo="bar'''
9
10 search_url = f'https://{site}/?search={search_term}'
11 resp = s.get(search_url)
12 for line in resp.text.split('\n'):
13 if 'input' in line:
14 print(line)
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
2: Python
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/gom
ez22/Desktop/websec/Lab Notebook #3/3.1/crossSiteScripting/reflected4.py"
<input type=text placeholder='Search the blog...' name=search value="gomez22">
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/gomez22/Desktop/websec/Lab Notebook #3/3.1/cr
Scripting/reflected4.py"
<input type=text placeholder='Search the blog...' name=search value="gomez22" foo="bar">
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec$
master* Python 3.8.5 64-bit 0 0 Ln 9, Col 1 Spaces: 4 UTF-8 LF Pytho



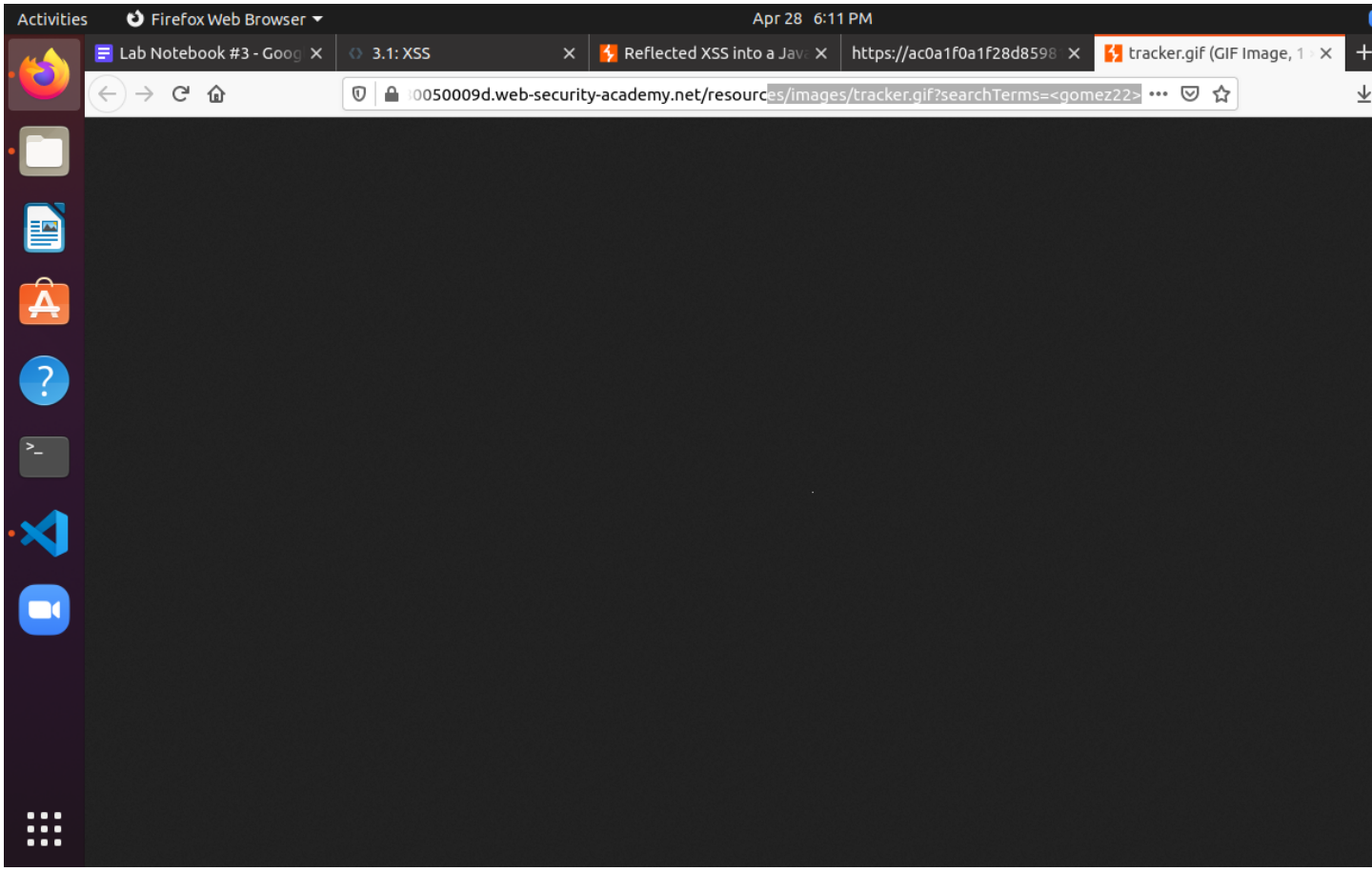
Cross-site-scripting/reflected (5)

-This lab has a vulnerability where it does not encode the “searchTerms” variable in a tracker image. As an attacker, I can use this to inject malicious code which is reflected in the site and lets me get code execution. To remediate this, the site’s developers should encode characters in the “searchTerms” variable.

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

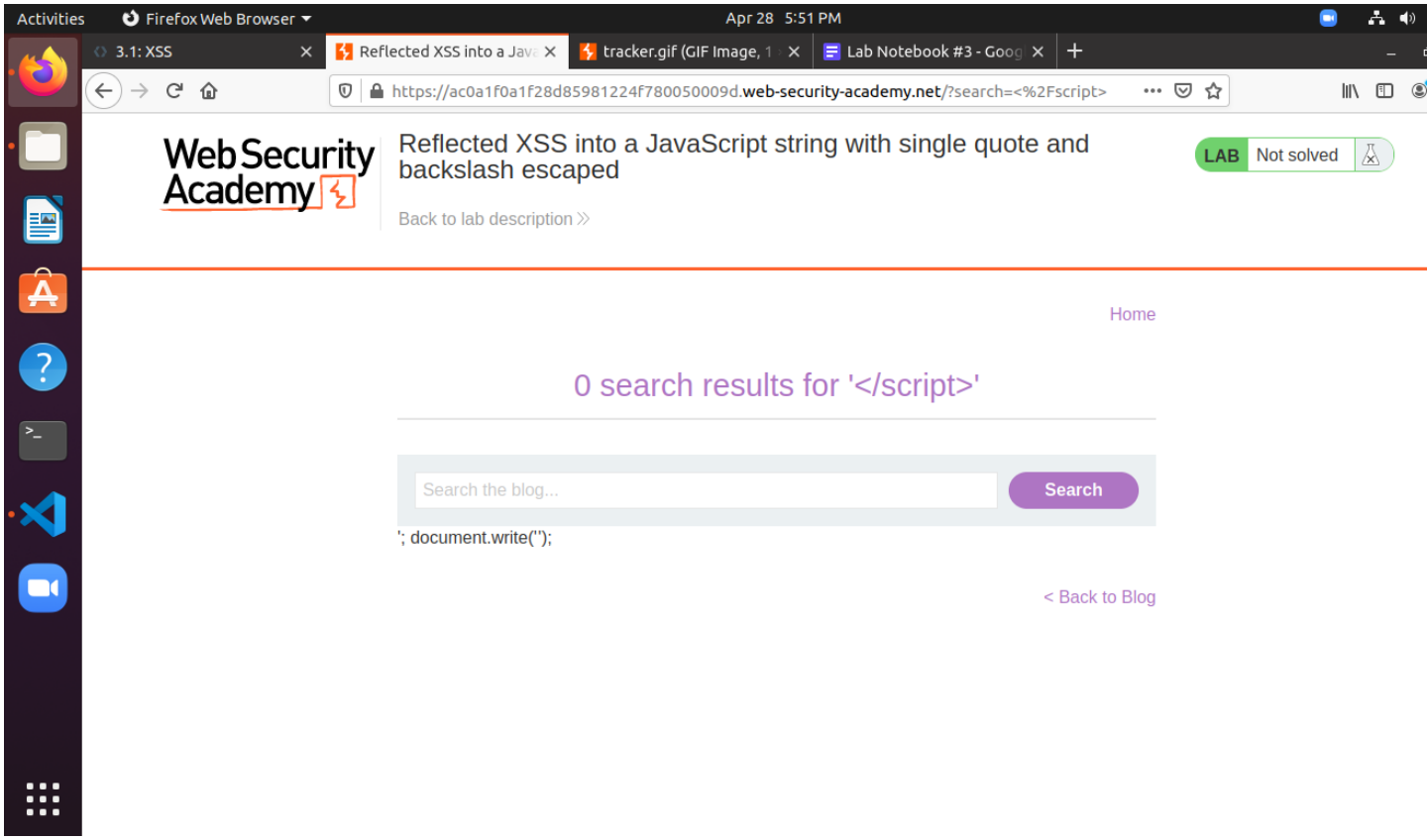


Open the tracker image in a new tab and show its URL that contains the search term



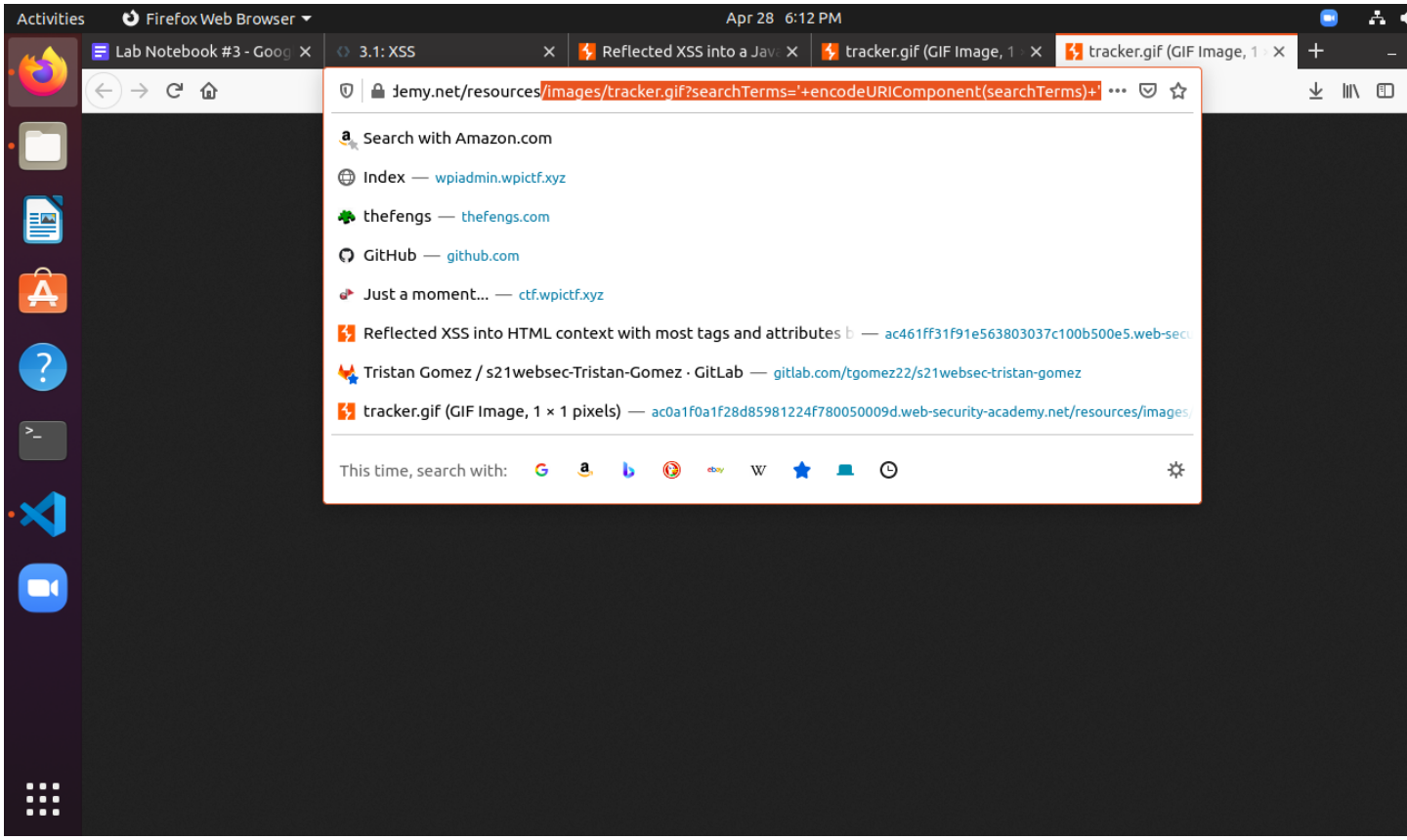
Perform the search for </script> and explain why the string below the search form appears and where it came from

-The string appears because the </script> is not encoded, so it is interpreted as a closing script tag which cuts off the rest of the JS code that was after the “searchTerms” variable. The string that appears below the search form is then misinterpreted by the browser as plain text on the page.

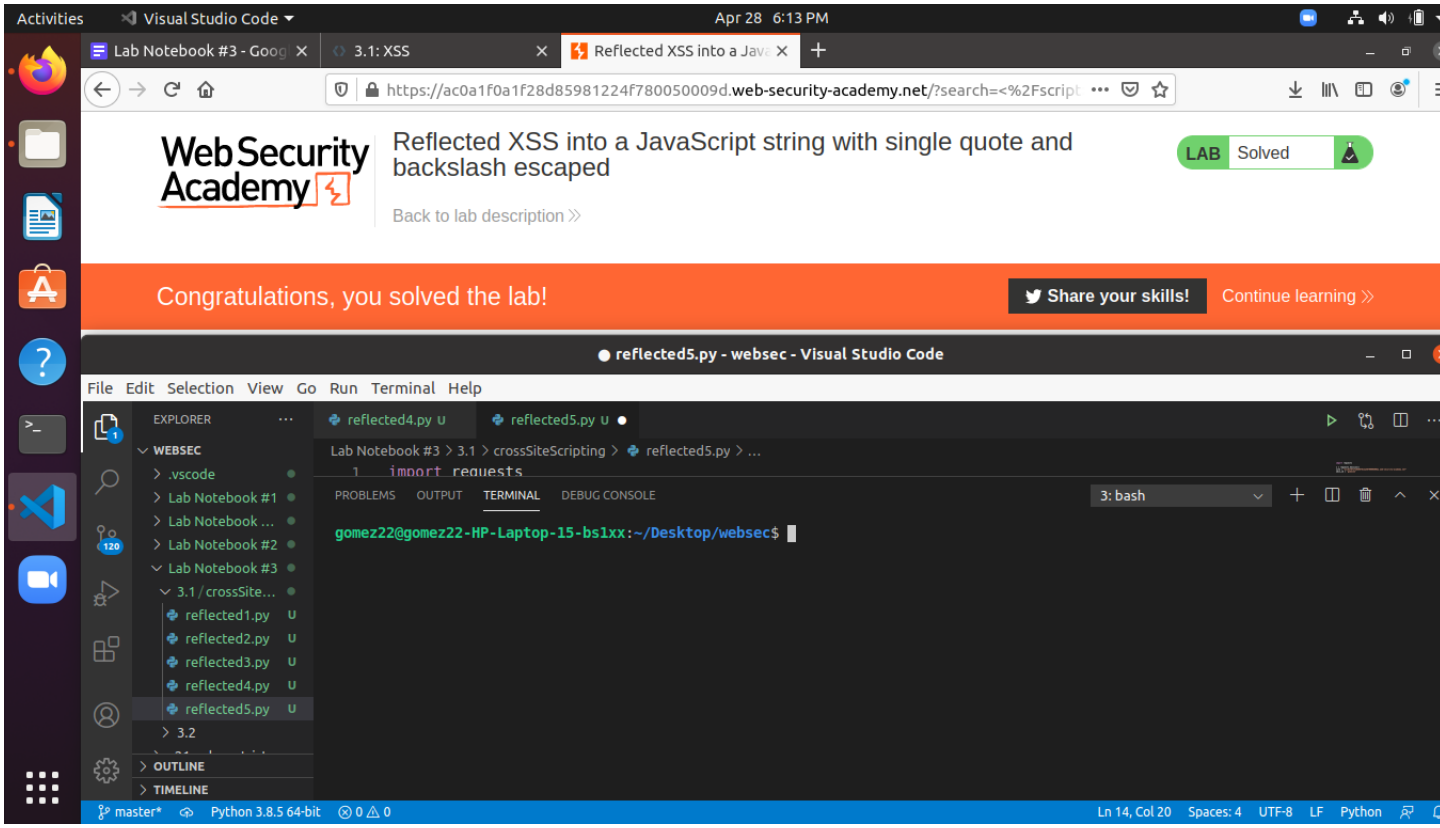


Show its URL and explain why it no longer contains the search term

-The search term is no longer in the URL because the browser interpreted the </script> as a closing script tag since it wasn't encoded upon form submission. This prematurely closes the script and the searchTerm was interpreted to be an empty string.



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

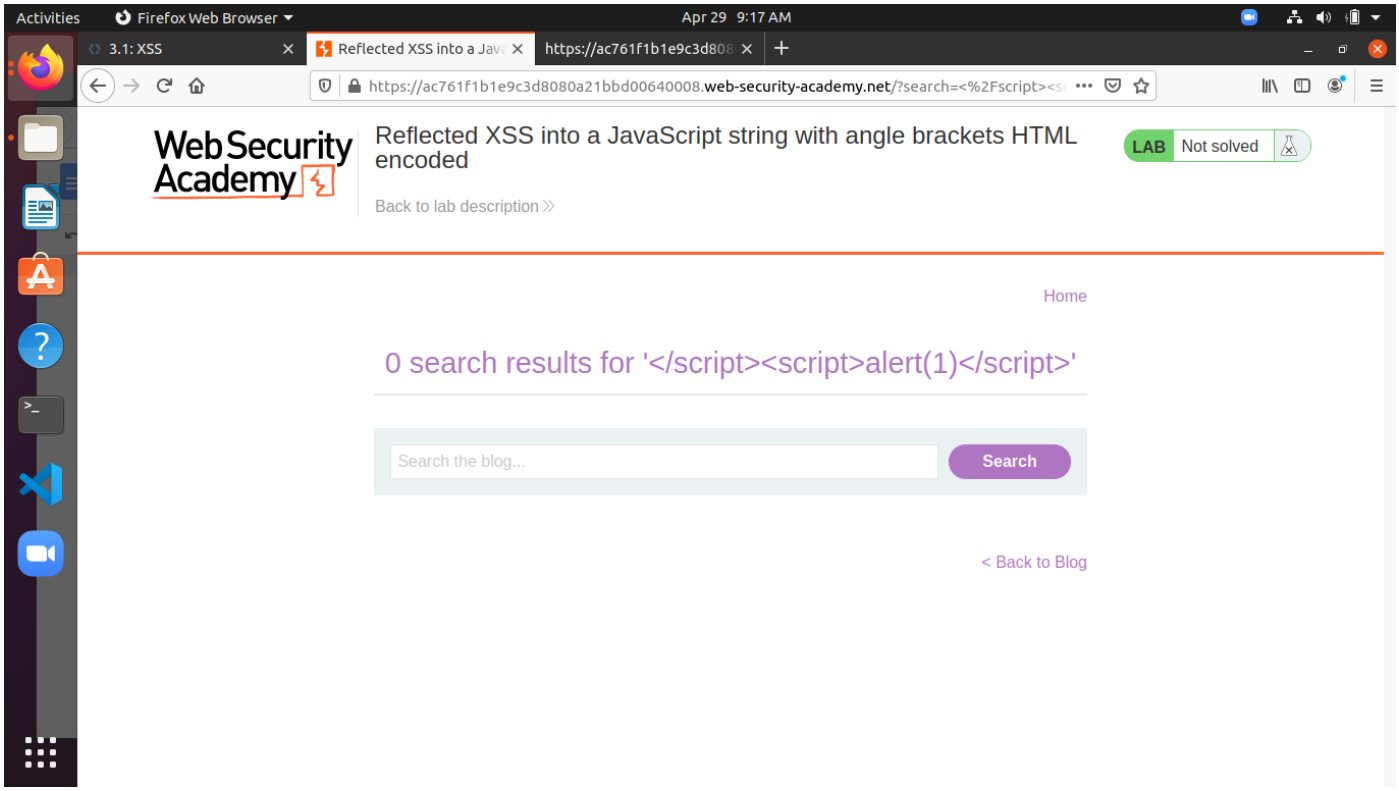


Cross-site-scripting/reflected (6)

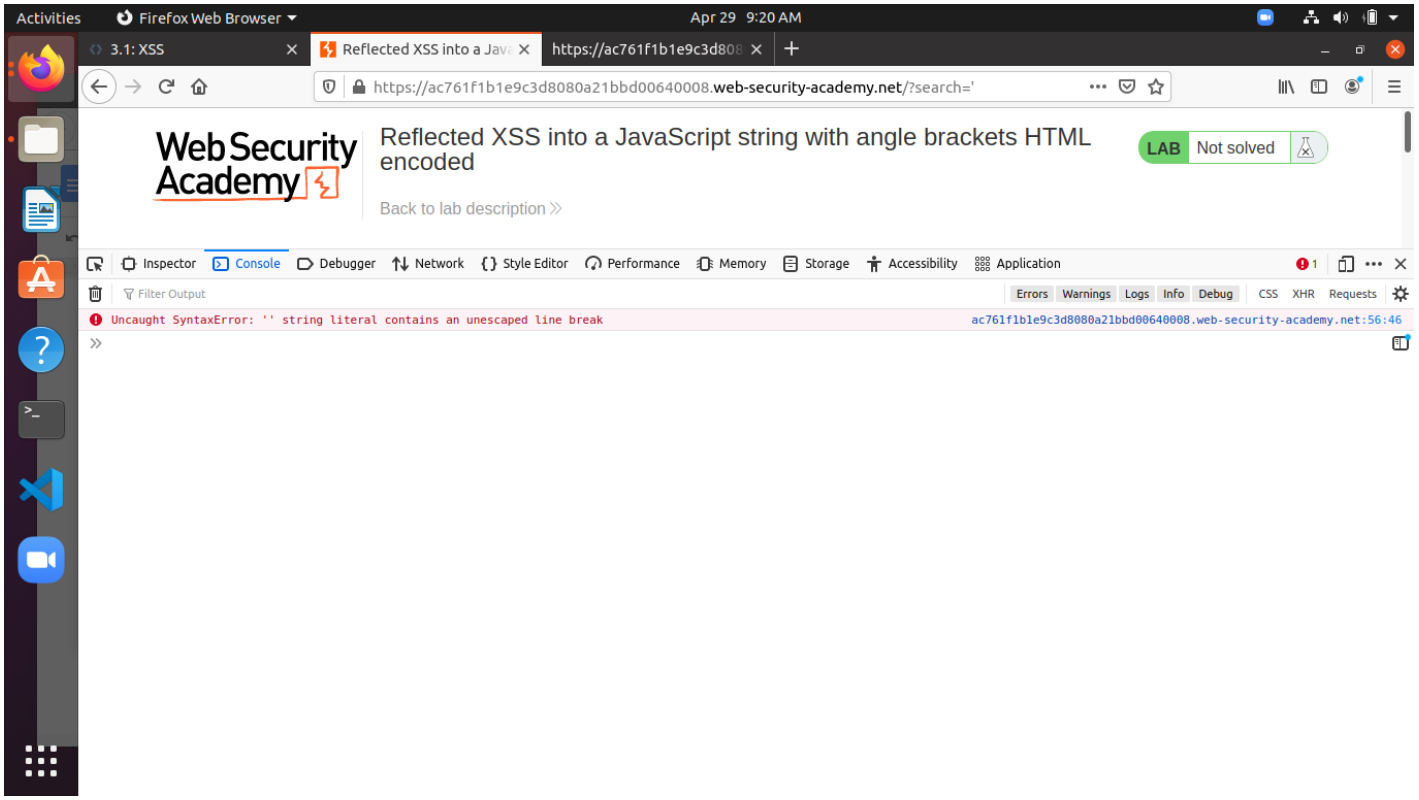
-This lab is vulnerable to an XSS attack.I can use a single quote character to break syntax of a statement and inject my own code. To remediate this, the developers need to account for all potential escape or special characters when reading client input.

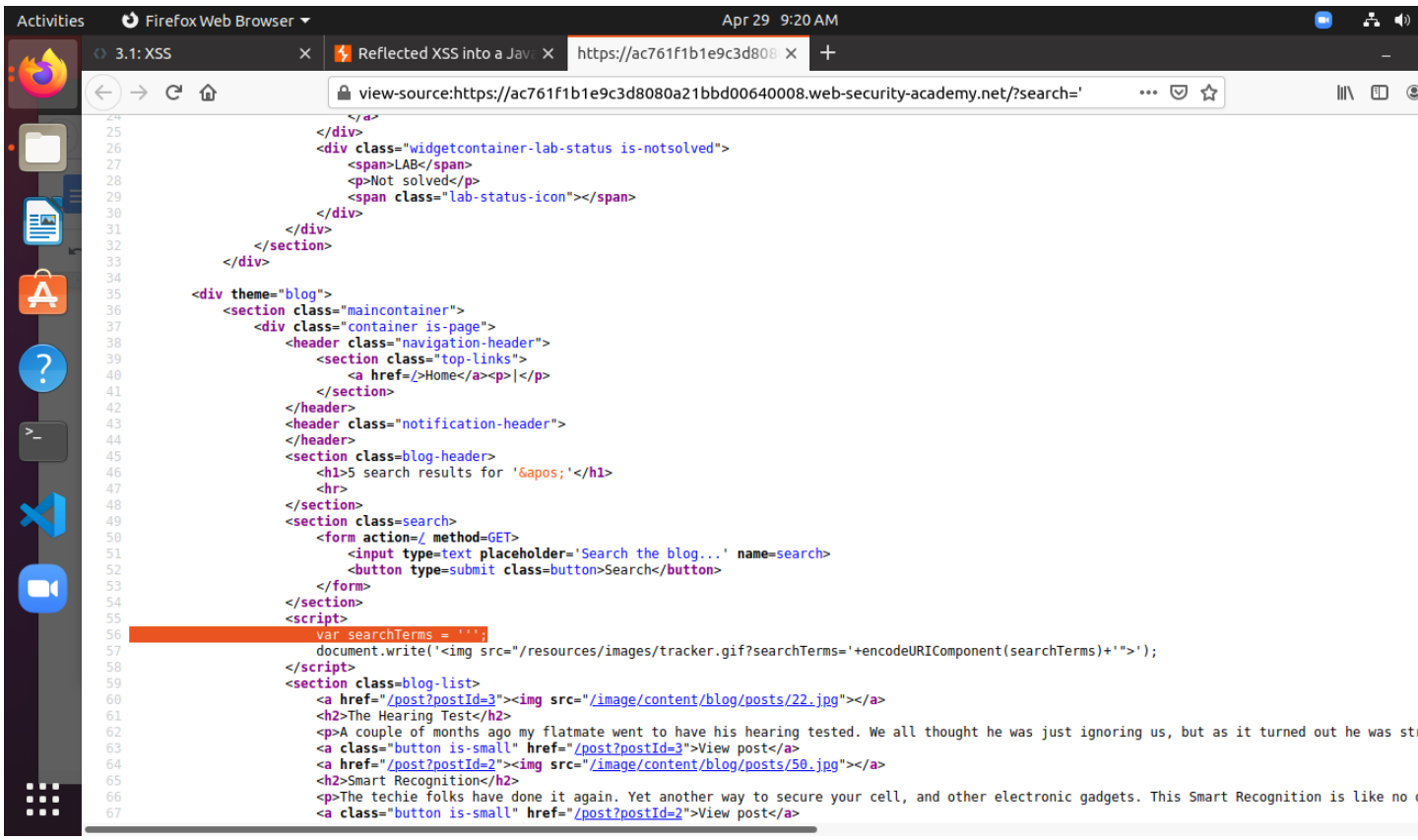
Take a screenshot of the search term as it is reflected back in the Javascript code. What has been done to the search term as it appears in the Javascript code?

-The search term has encoded the special characters to prevent XSS injection attacks.



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


Activities
Firefox Web Browser
Apr 29 9:20 AM
3.1: XSS
Reflected XSS into a Jav
https://ac761f1b1e9c3d808
https://ac761f1b1e9c3d8080a21bbd00640008.web-security-academy.net/?search='
Web Security Academy
Reflected XSS into a JavaScript string with angle brackets HTML encoded
LAB
Not solved
Back to lab description
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
Uncaught SyntaxError: '' string literal contains an unescaped line break
ac761f1b1e9c3d8080a21bbd00640008.web-security-academy.net:56:46


Activities
Firefox Web Browser
Apr 29 9:20 AM
3.1: XSS
Reflected XSS into a Jav
https://ac761f1b1e9c3d808
view-source:https://ac761f1b1e9c3d8080a21bbd00640008.web-security-academy.net/?search='

LAB
Not solved

</section>

<section class="maincontainer">

<header class="navigation-header">
<section class="top-links">
Home |
</section>
</header>
<header class="notification-header">
</header>
<section class=blog-header>
<h1>5 search results for '''</h1>
<hr>
</section>
<section class=search>
<form action=/ method=GET>
<input type=text placeholder='Search the blog...' name=search>
<button type=submit class=button>Search</button>
</form>
</section>
<script>
var searchTerms = '';
document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
</script>
<section class=blog-list>
<img src="/image/content/blog/posts/22.jpg">
<h2>The Hearing Test</h2>
A couple of months ago my flatmate went to have his hearing tested. We all thought he was just ignoring us, but as it turned out he was st
View post
<img src="/image/content/blog/posts/50.jpg">
<h2>Smart Recognition</h2>
The techie folks have done it again. Yet another way to secure your cell, and other electronic gadgets. This Smart Recognition is like no
View post

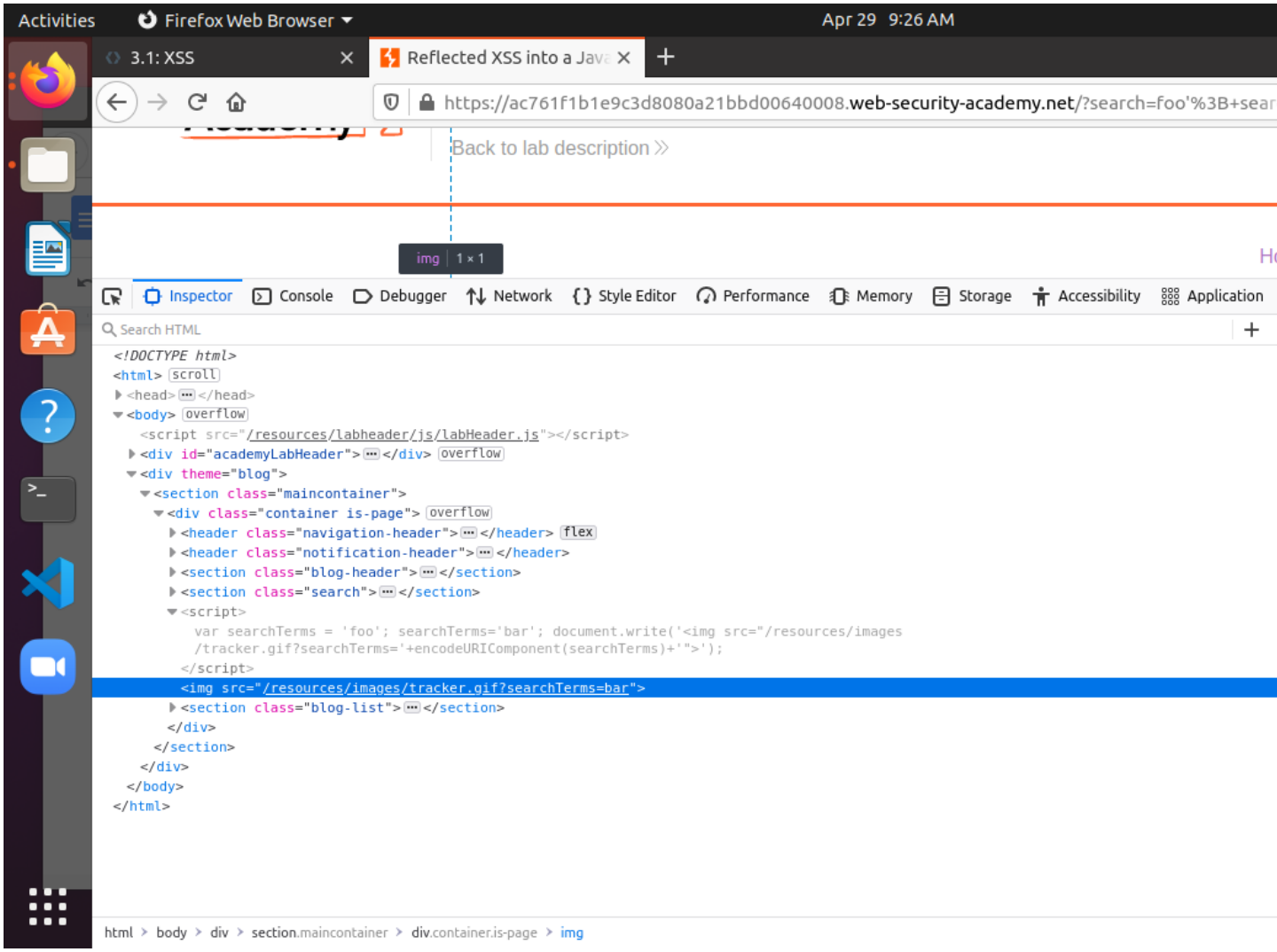
Why does the search term not appear in the <img> tag?

-The search term does not appear because the ‘; characters close the opening single quote character while the semicolon is interpreted as the ending of a JS statement. This results in searchTerms being assigned an empty string.

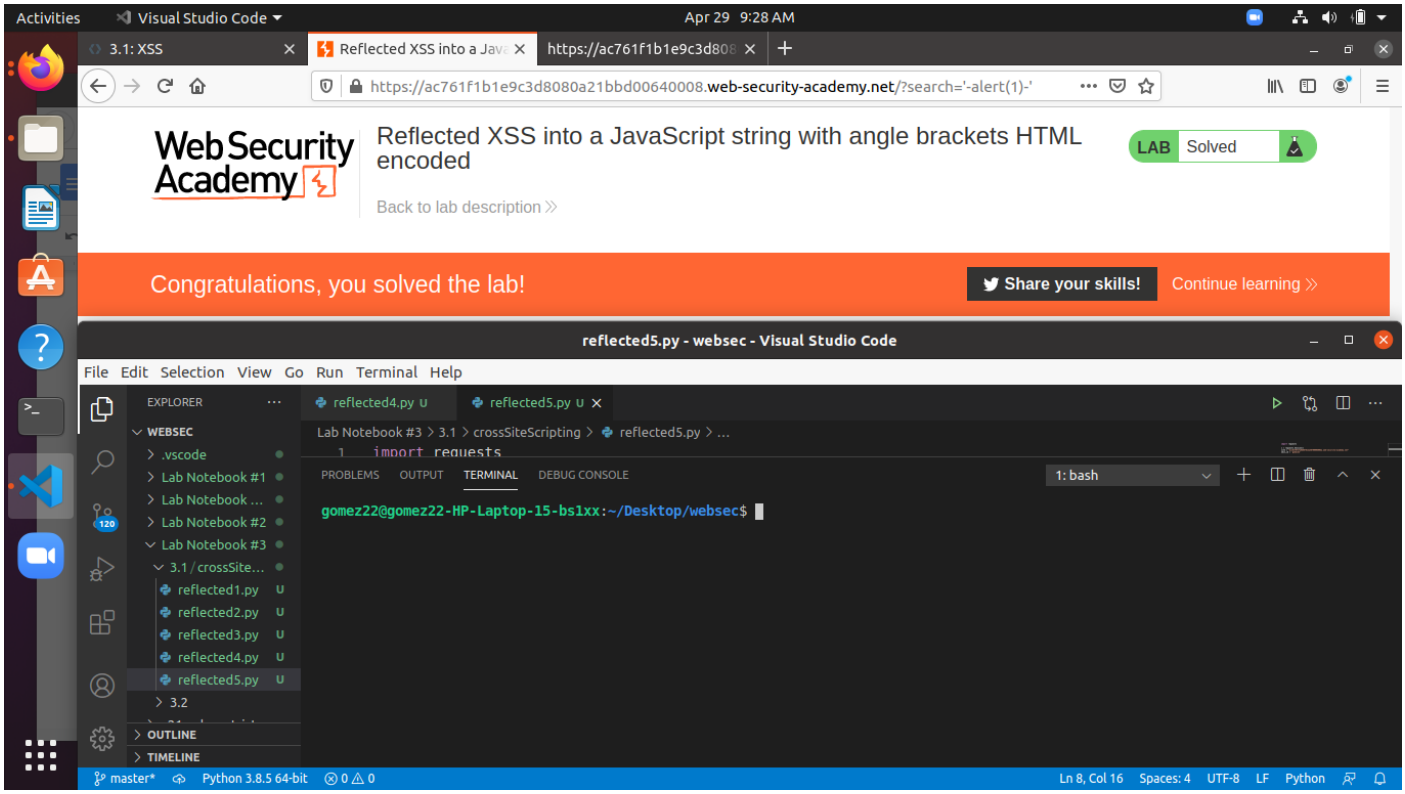
What does // do in the Javascript code?

-The // characters together signifies a comment.

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

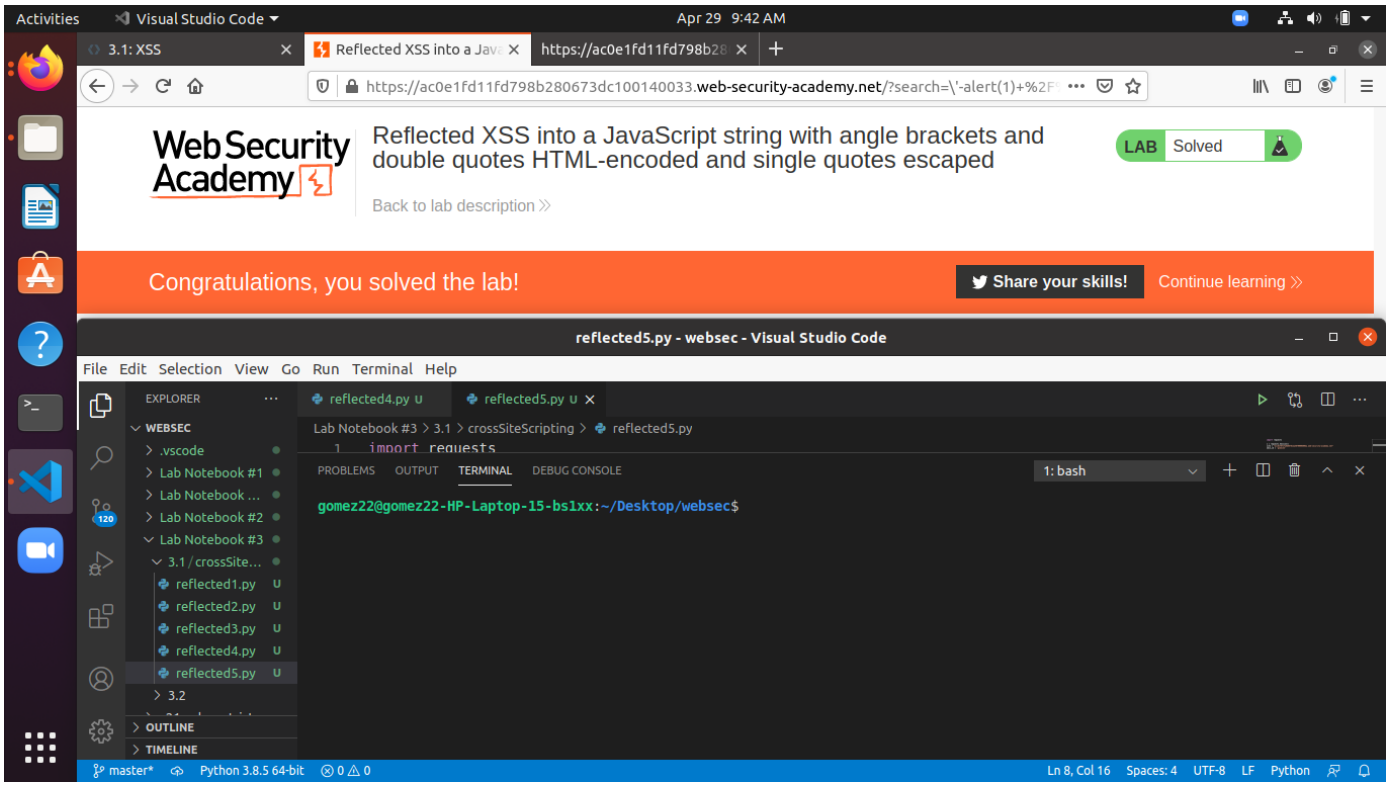


Cross-site-scripting/reflected (7)

Explain why this error has happened and what needs to be done to fix it

-This error has happened because the site inserts an escape character (\) if a single quote is inserted in order to prevent an error from occurring due to mismatched quotes. Inserting an additional \ character escapes the inserted escape character (\) causing the single quote to be interpreted as code instead of as a string, resulting in a mismatched quotes error. To fix this, the developers would need to insert an \ escape character for every \ given by the user as input in order to prevent the error from occurring.

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

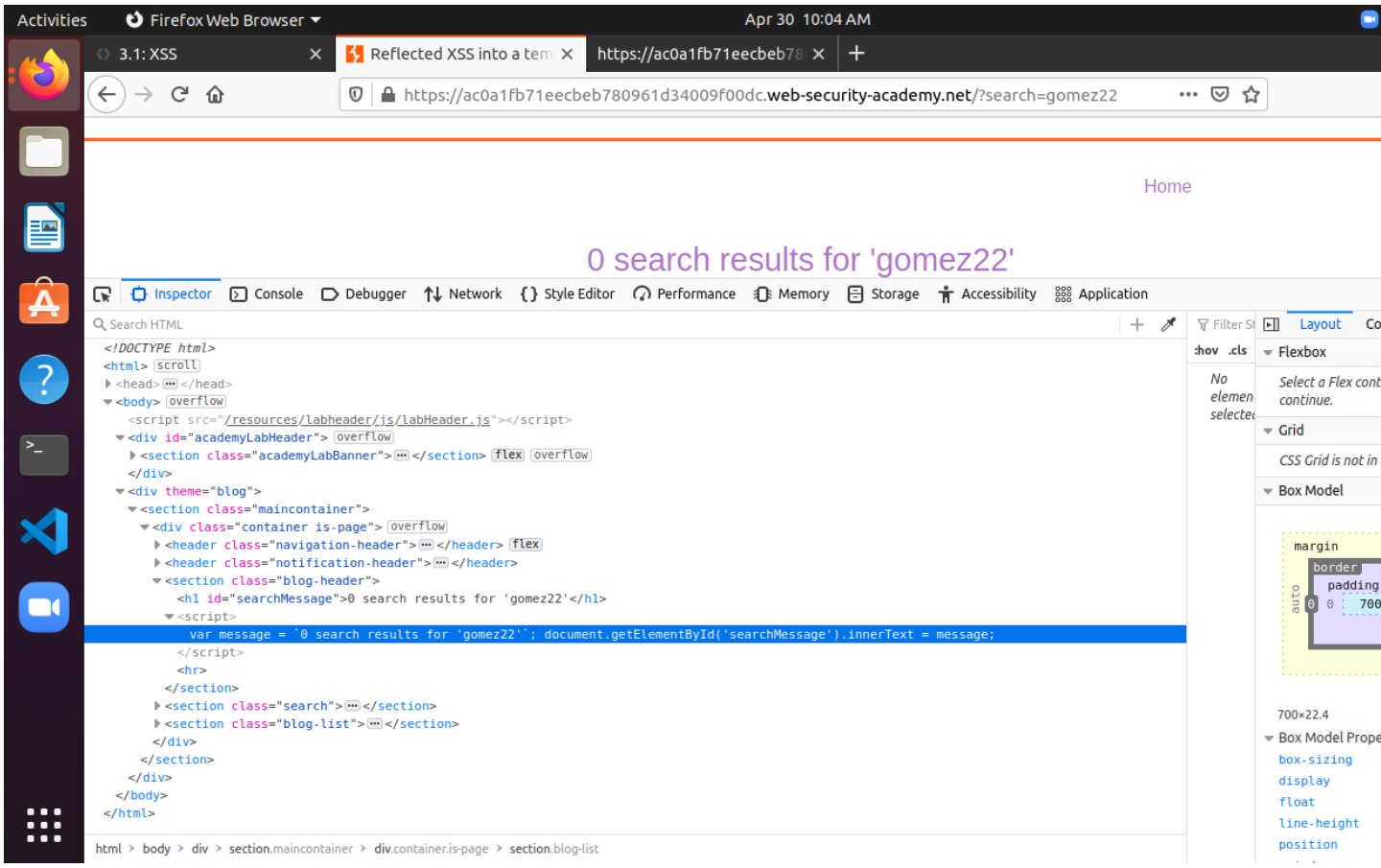


Cross-site-scripting/reflected (8)

-This lab is vulnerable to an XSS attack due to it directly reading in client input into a template literal. As an attacker I can break the syntax of a statement then insert my own malicious code. To remediate this, the developers should change how they process input and not directly read in client input into a template literal.

Show this line of Javascript

-In the picture below, the highlighted line is actually two individual statements in JavaScript. The “document.getElementById …” line is the line that inserts the text into the HTML element that you see on the page.



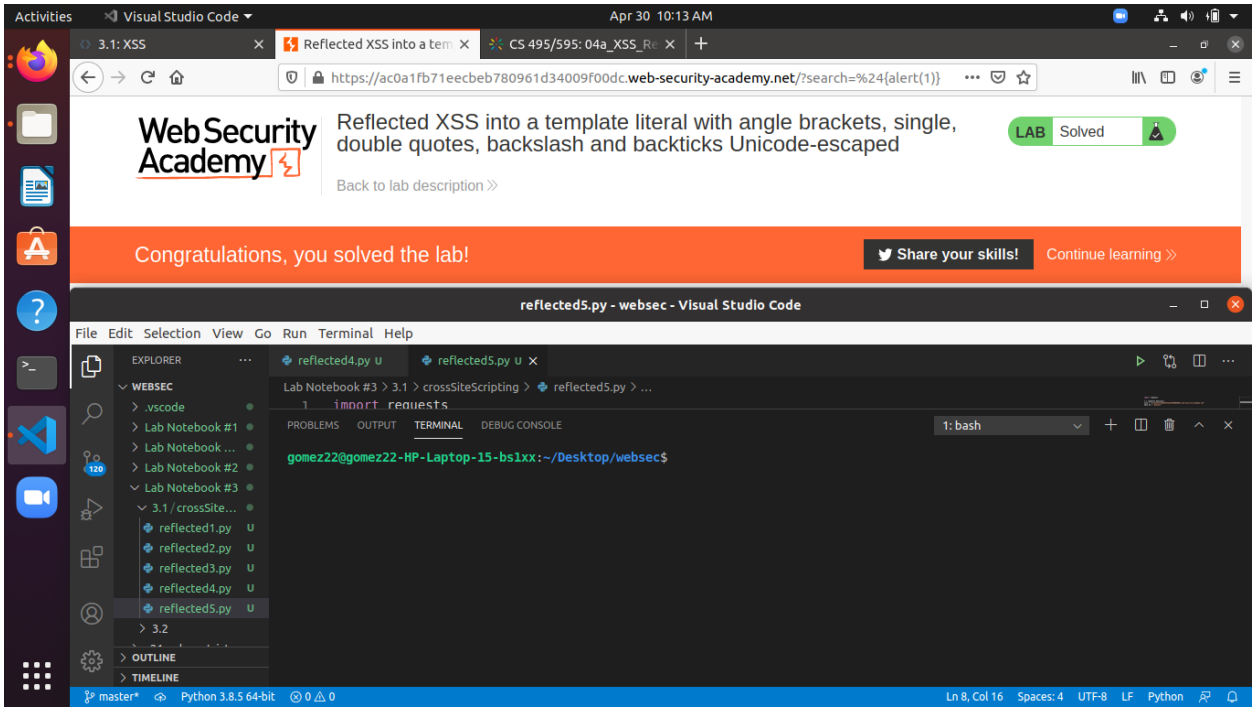
Show the line that defines the template literal.

-The photo above shows the line that defines the template literal in the highlighted line. The leftmost line of code “var message = `0 search results for ‘gomez22’` is the one that defines the template literal. The string being assigned to var message is the template literal which is denoted by the backticks.

Explain the results

-The template literal format allows the ‘cs${490+5}’ to be evaluated in a special way, causing the ${490+5} to be evaluated as code, so 490+5 is evaluated and the final resulting value 495 takes the place of the statement ${490+5}.

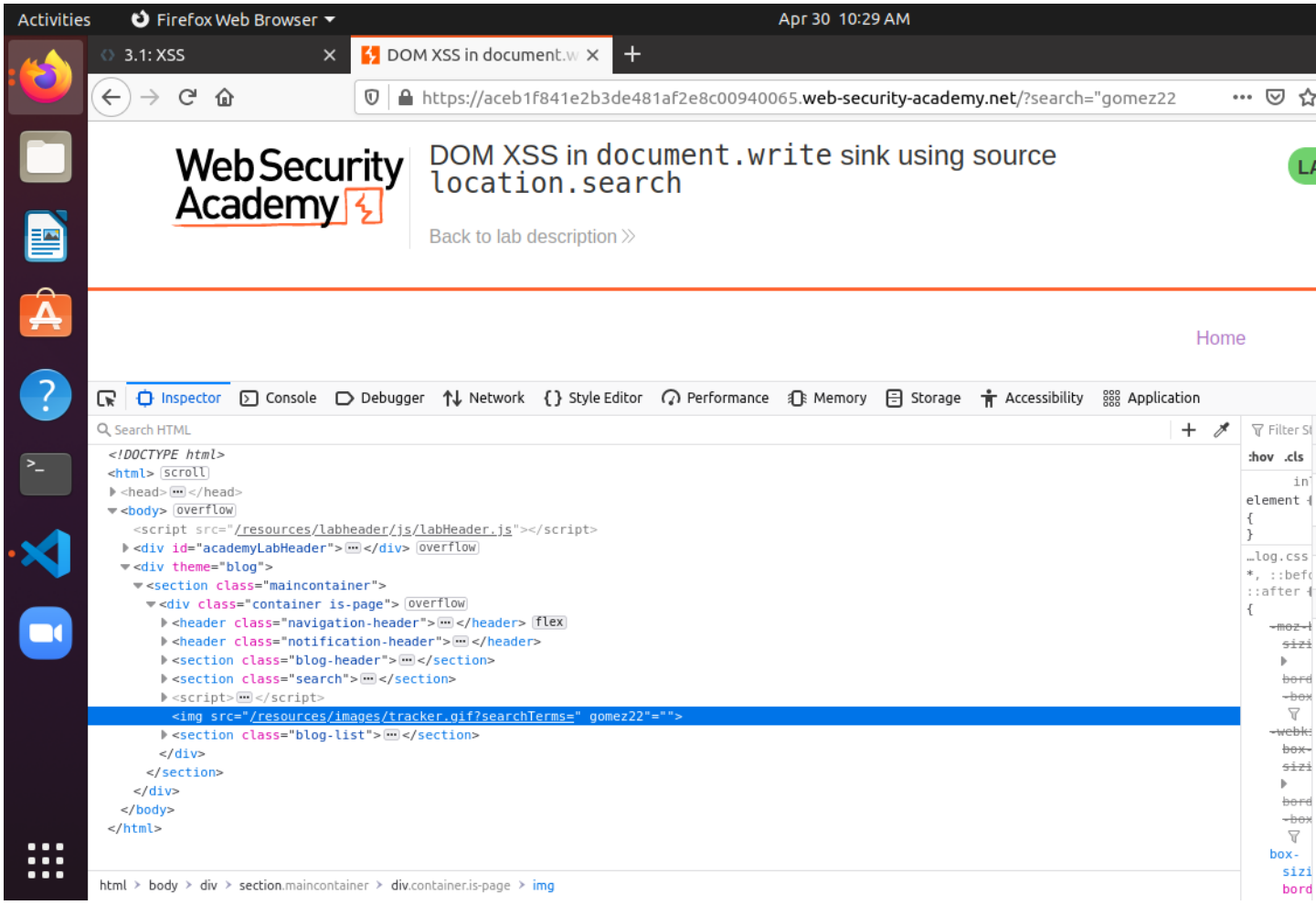
Take a screenshot showing completion of the level that includes your OdinId



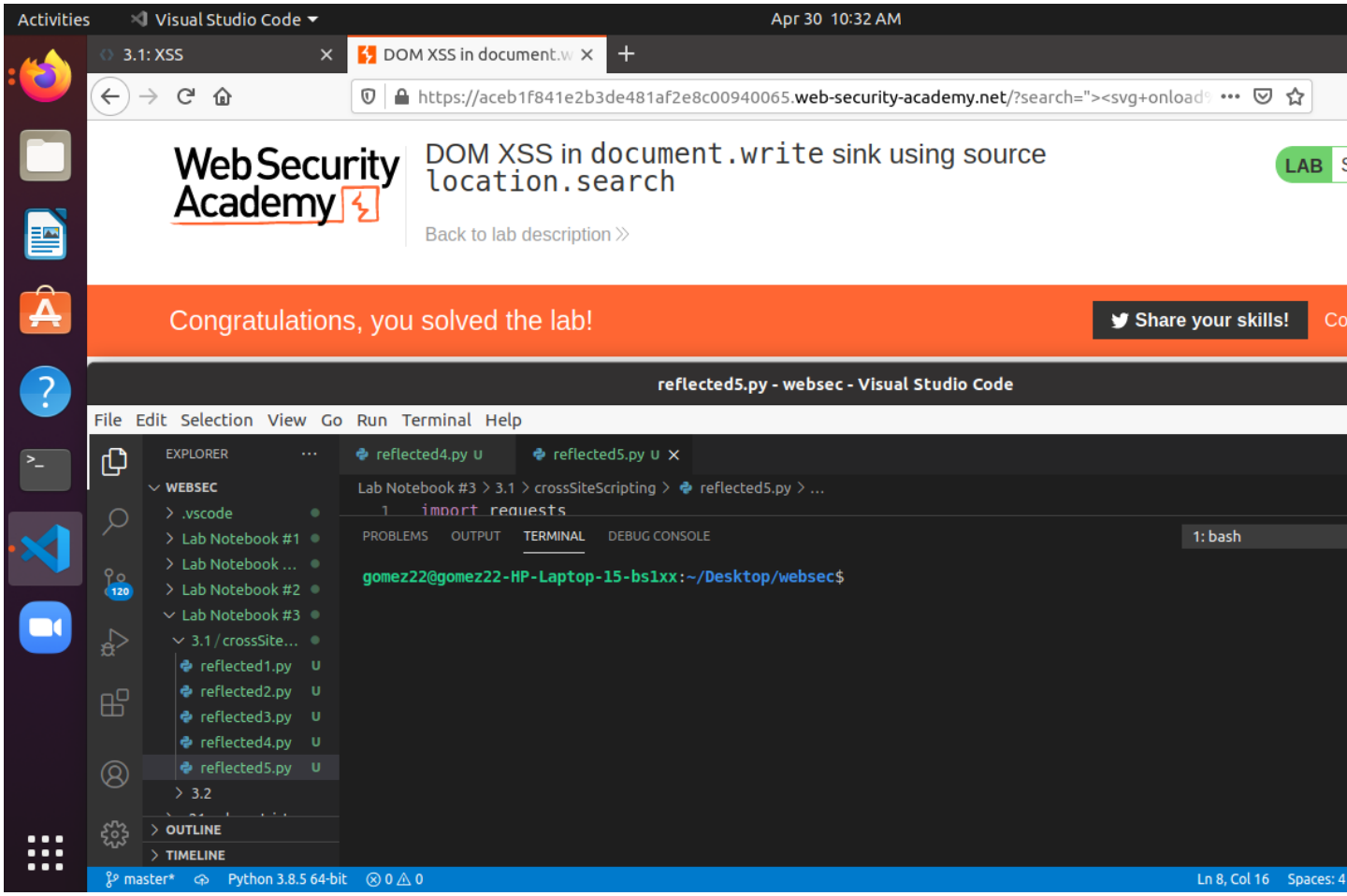
Cross-site-scripting/dom-based (1)

-This lab is vulnerable because it reads in untrusted client input and inserts it into a url inside of an element tag. As an attacker, I can break the tag’s syntax and insert my own malicious code. To remediate this, the developers should properly sanitize client input.

Take a screenshot of the <img> tag in Developer Tools and show that its syntax has been broken.



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



Cross-site-scripting/dom-based (2)

-This level has a vulnerability where a variable is read from the url. As an attacker, I can insert malicious code into the url and hijack the site. To remediate this, the developers should properly sanitize all user input.

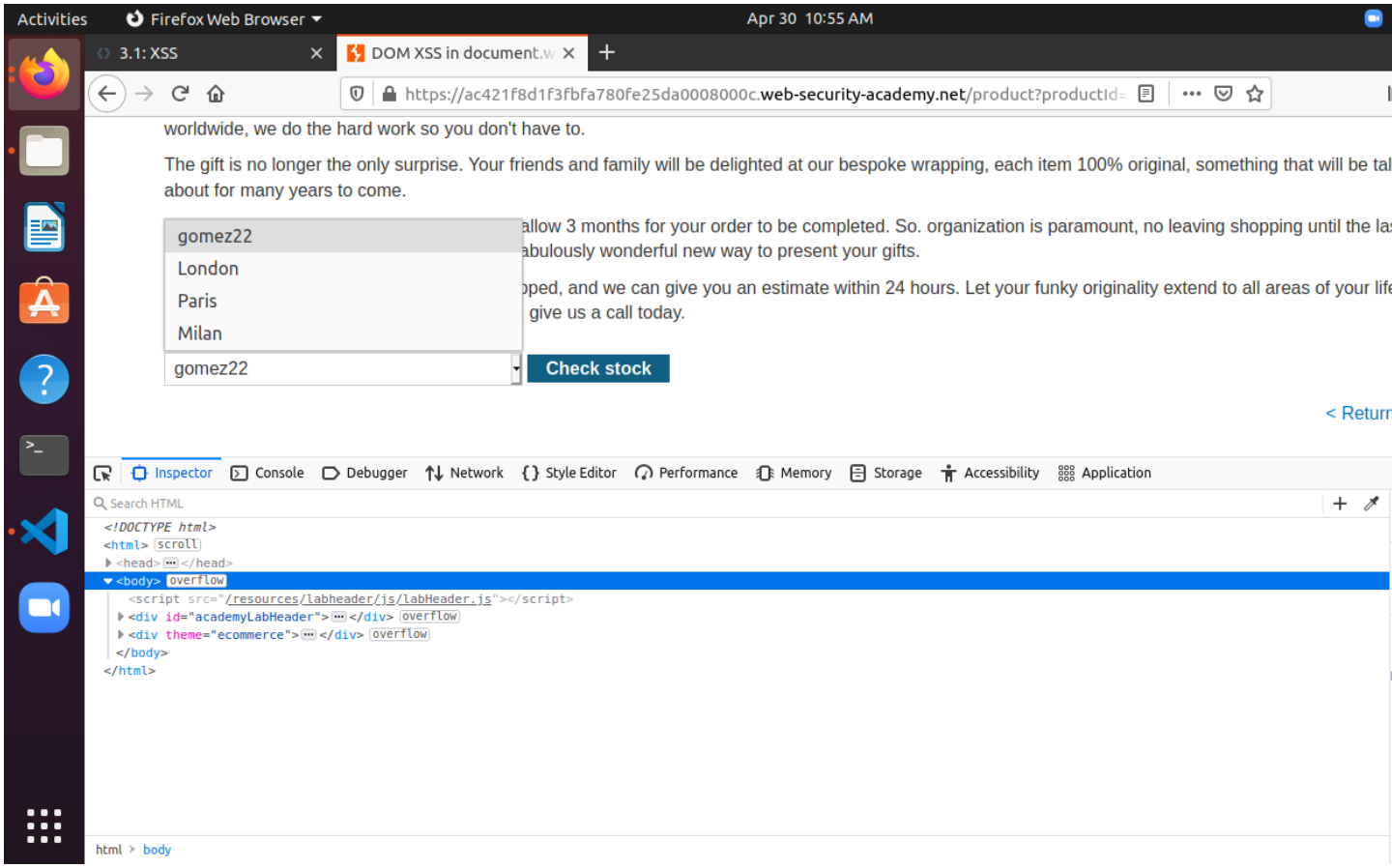
How does the store variable get set? Can one always assume that it is one of the values in the stores list?

-The store variable is set by “.get(‘storeId’)”. It appears to grab the value for this variable from the URL if possible. I don’t think that one can always assume that it is one of the values in the stores list because the store variable doesn’t pull its set its value directly from the stores list. If a malicious option was inserted into the select element then I think the store variable could be hijacked.

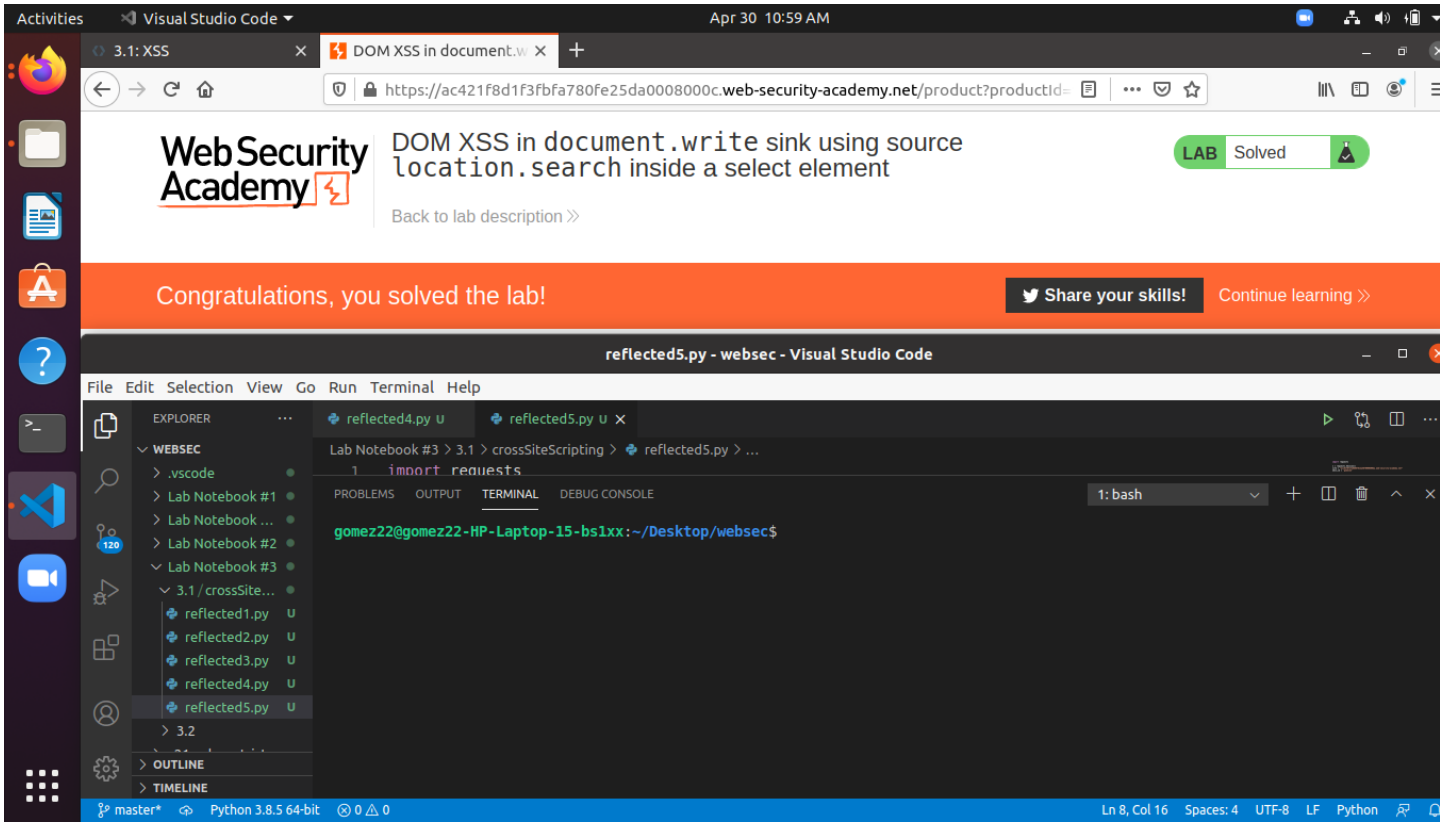
What is the purpose of the first if statement? What is the purpose of the second if statement?

-The first if statement says if store exists in the URL then write <option selected> {value of store here} </option> to the dom. The second if statement lies within a for loop. This if statement says if one of the stores in the array being iterated through is the same as the store variable then don’t write it to the dom again.

Take a screenshot showing that you have successfully injected a bogus store that is not one of the initial ones in the stores list.

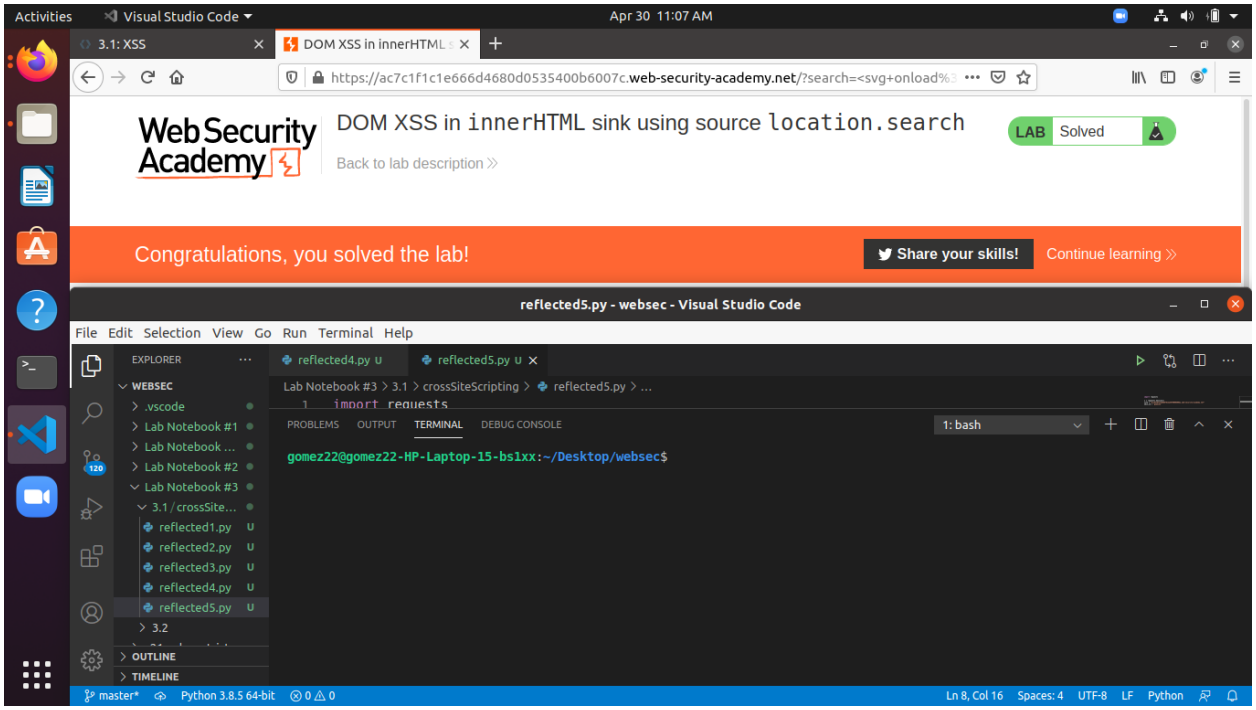


Take a screenshot showing completion of the level that includes your OdinId



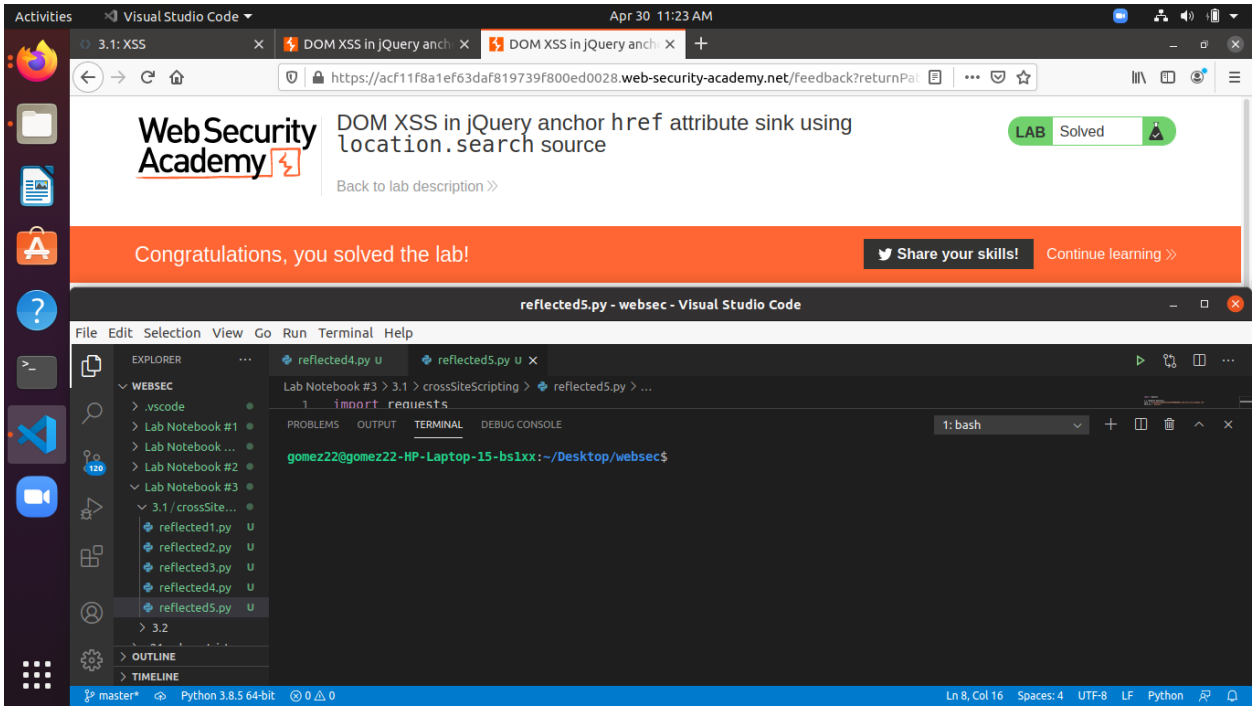
Cross-site-scripting/dom-based (3)

-The vulnerability in this lab is that the search bar uses JavaScript code to insert the search query onto the dom using innerHTML assignment. As an attacker, I can insert a new tag with malicious attributes to get code execution and control of the site. To remediate this, the developers should sanitize all user input.



Cross-site-scripting/dom-based (4)

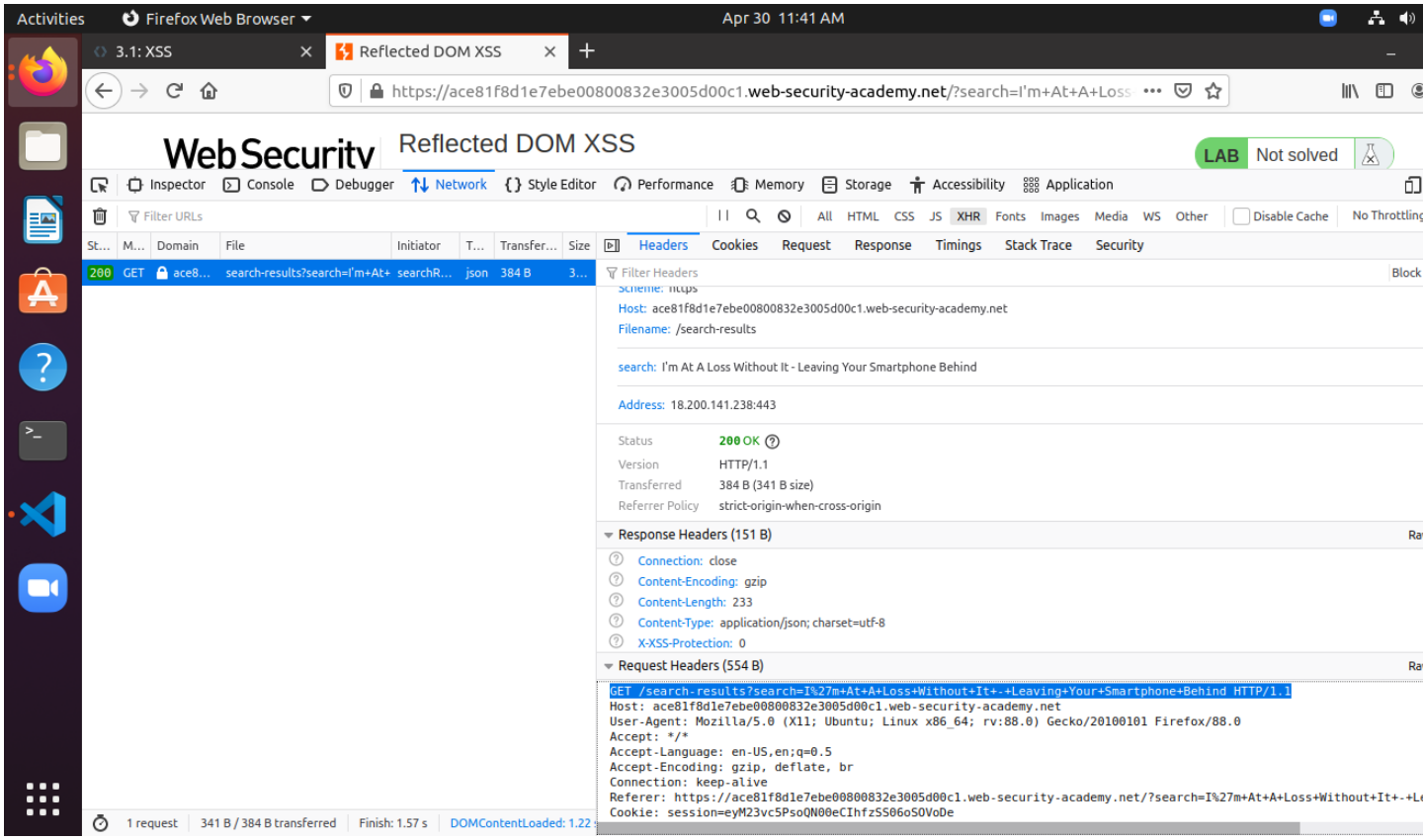
-The vulnerability in this lab is that JavaScript running on the site takes untrusted input from the client and places it into an <a> tag directly. As an attacker, I can use this to insert my malicious code. To remediate this, the site developers need to sanitize all user input.



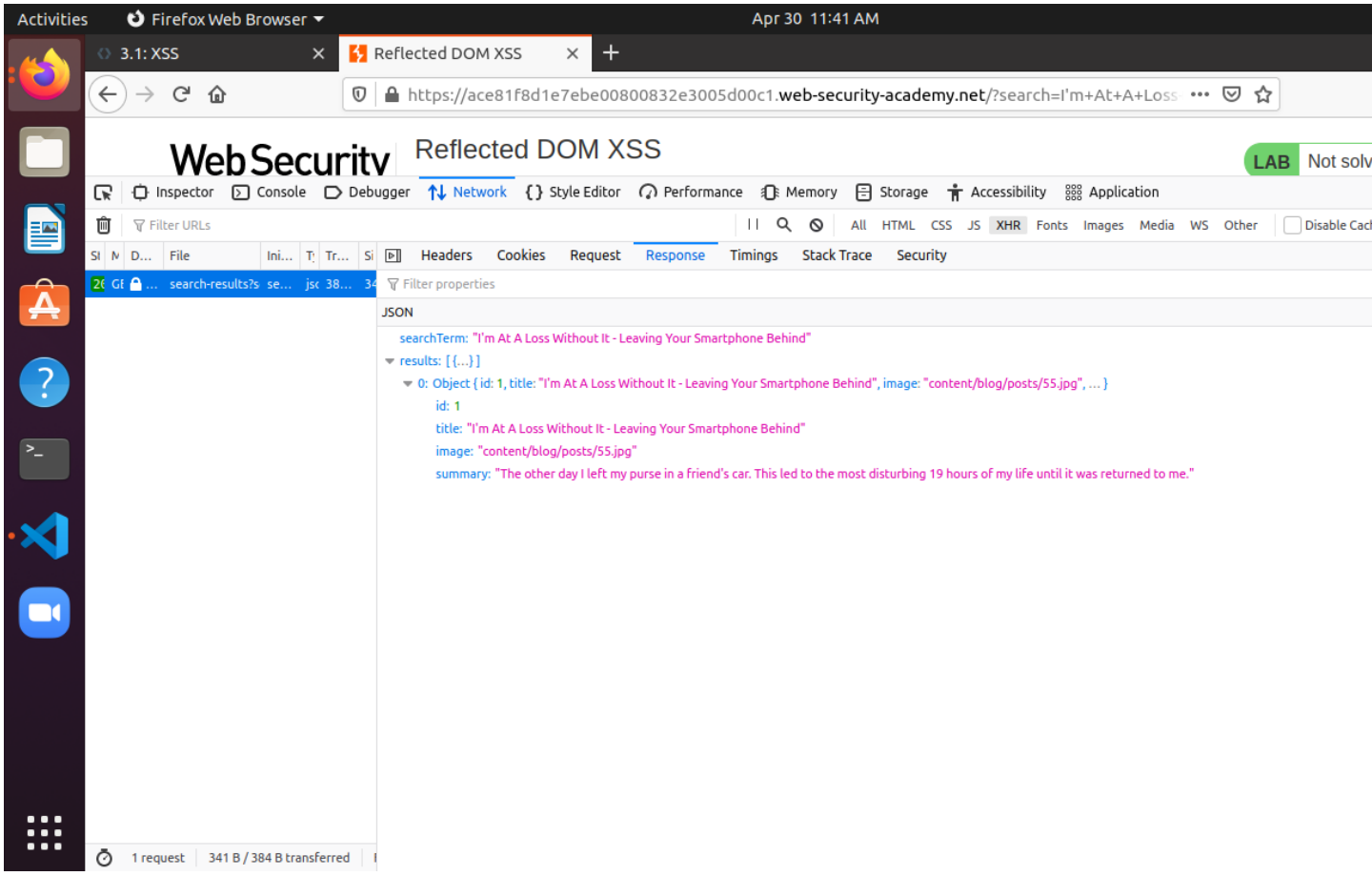
Cross-site-scripting/dom-based (5)

-The vulnerability of this level is that eval() is called on non-sanitized user input queries, which means that as an attacker, I can break the query syntax and insert code that can be executed. To remediate this, the site should sanitize all user input and not call eval() if possible.

Show a screenshot of the URI path of the XHR request

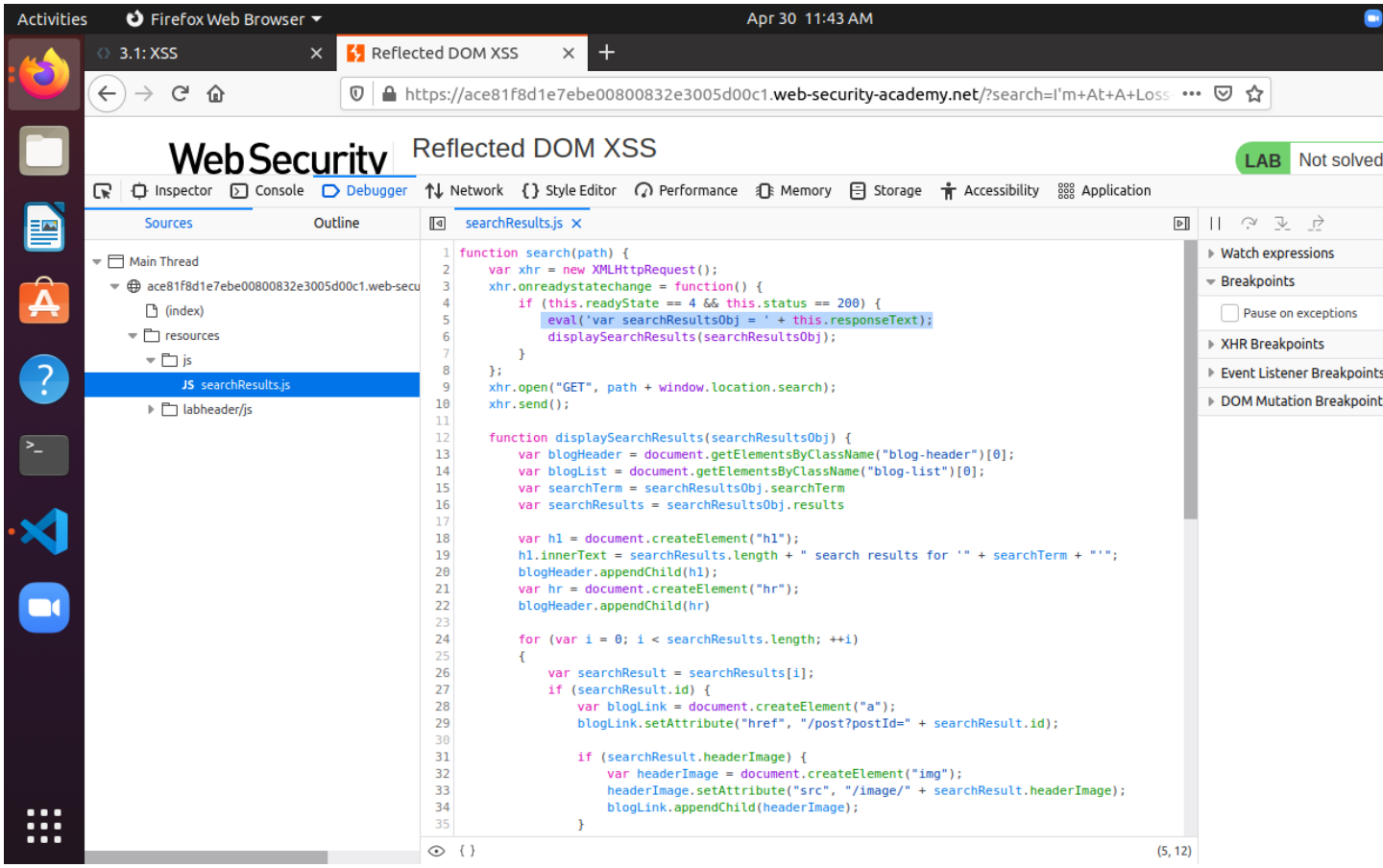


Show a screenshot of the JSON response that echoes the search term



Take a screenshot of the vulnerable line of code

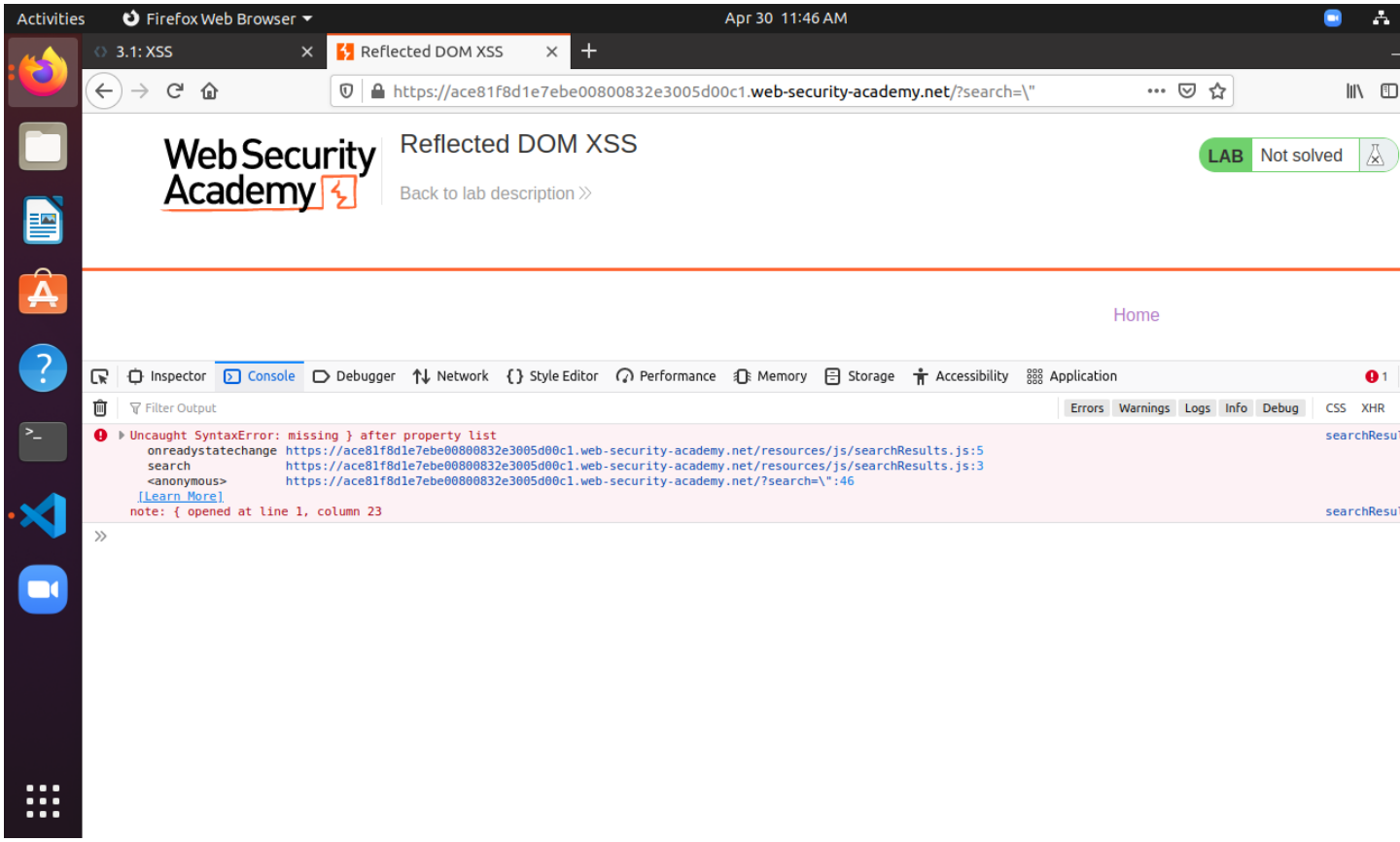
-The vulnerable line of code is highlighted in the image below.



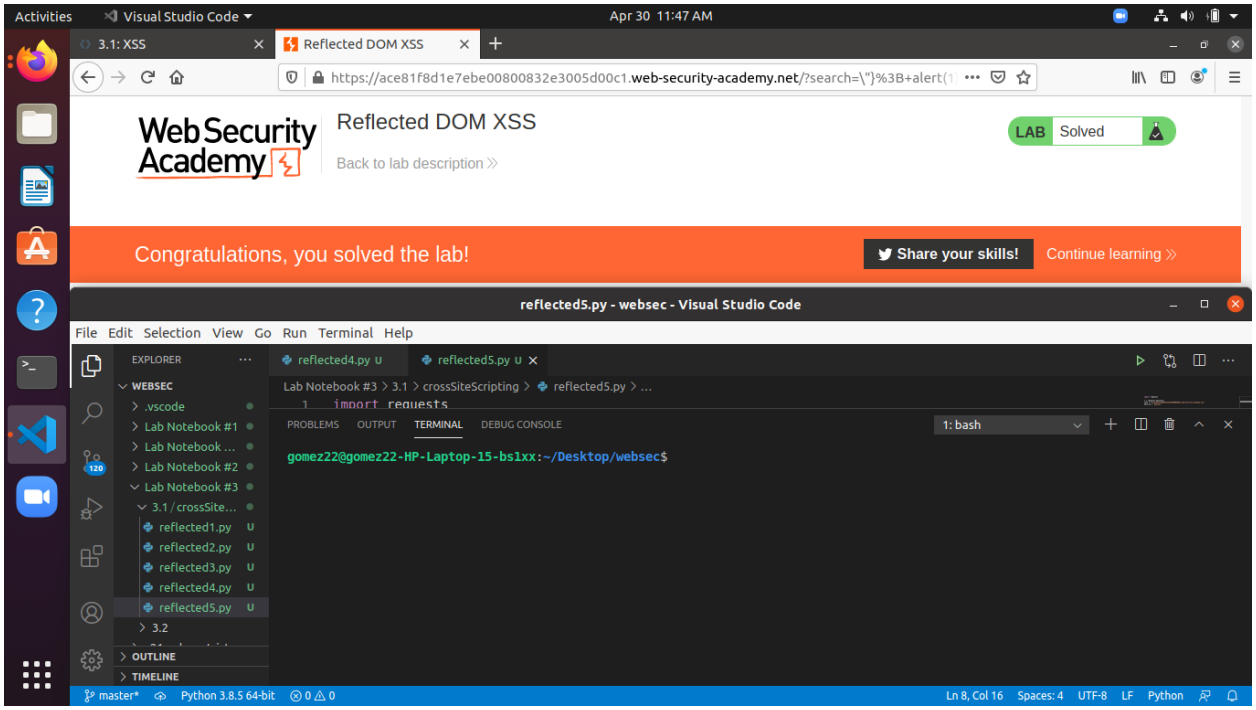
What has been done to the delimiter to prevent syntax from being broken?

-An escape character “\” was inserted to prevent syntax from being broken.

Take a screenshot of the error that has been produced in the console.

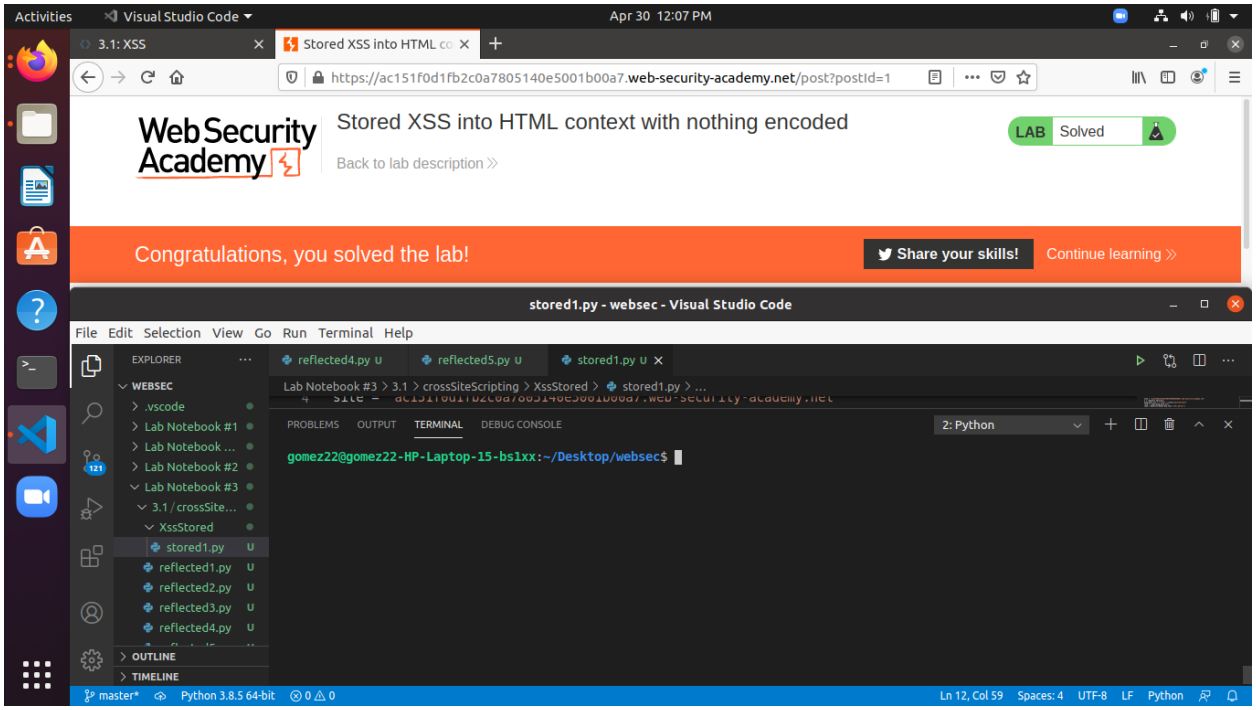


Take a screenshot showing completion of the level that includes your OdinId



Cross-site-scripting/stored (1)

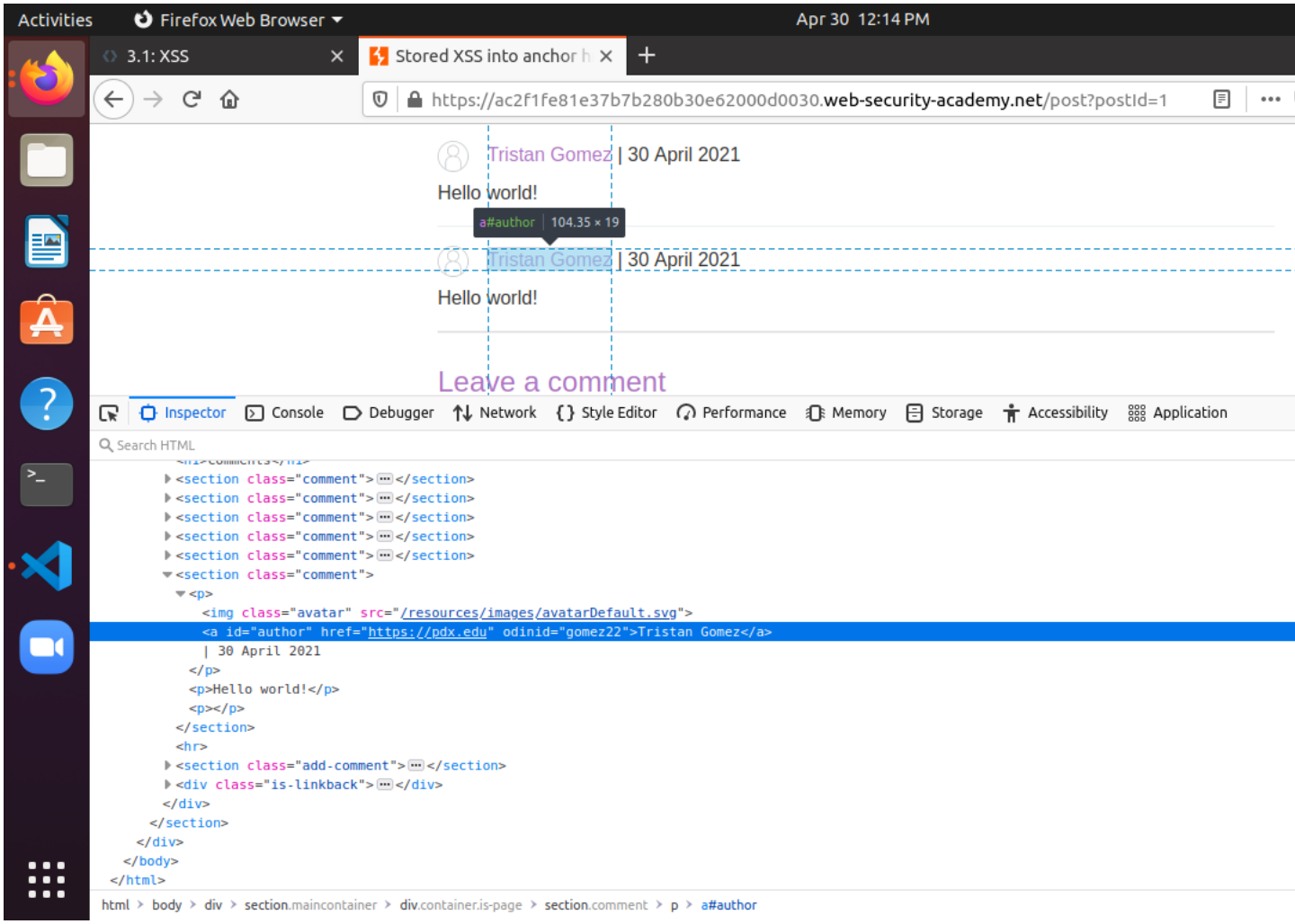
-This level is vulnerable to an XSS attack because it doesn't sanitize user comments which allows me to insert malicious javascript into a comment which will be executed when the dom is rendered. To remediate this, the developers need to sanitize all user input.



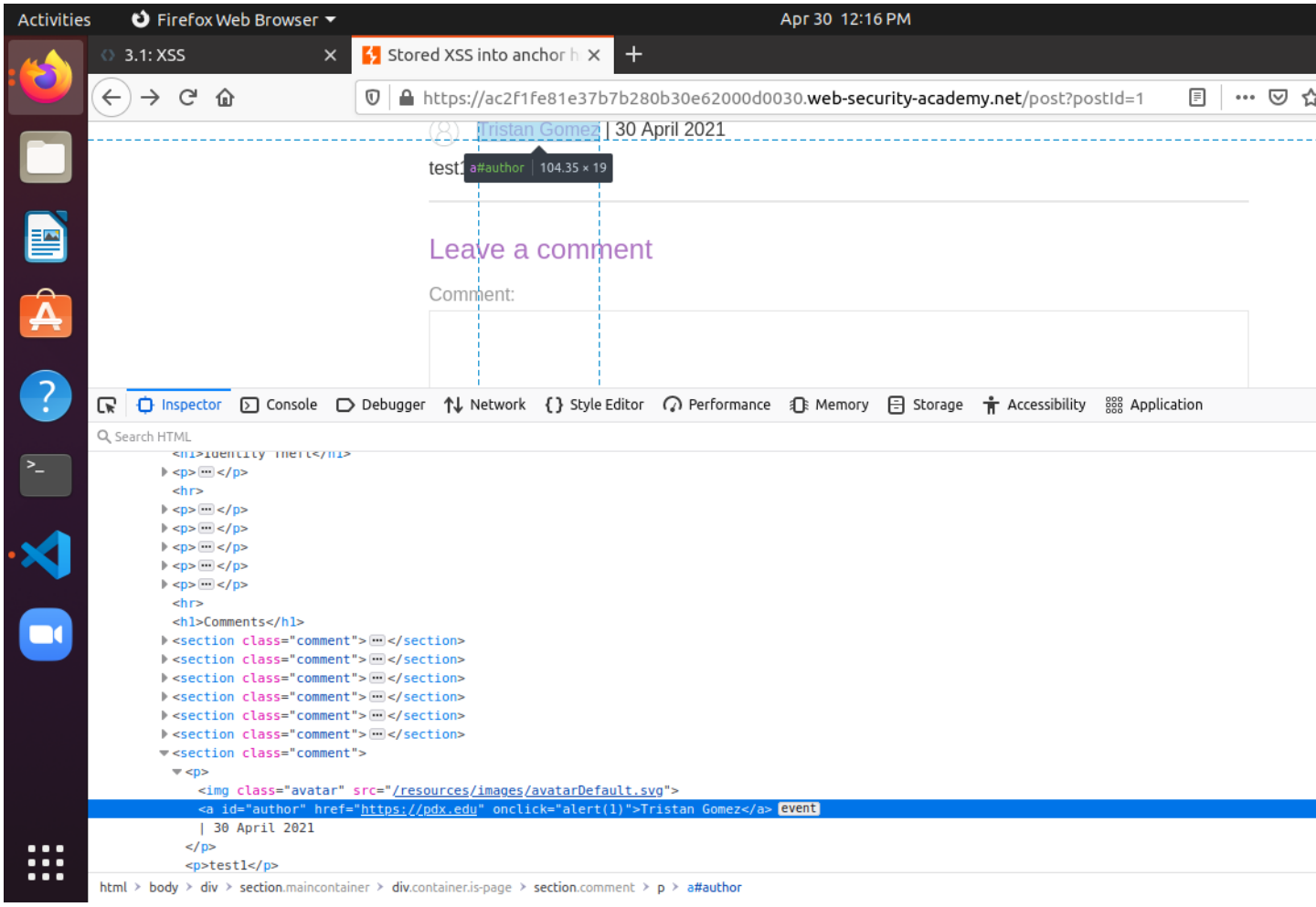
Cross-site-scripting/stored (2)

-The vulnerability in this level is that the site doesn't encode all characters given by the user as input. As an attacker, I can break the syntax of the <a> tag and insert my own malicious JavaScript. To remediate this, the developers should encode special characters given by the user as input and sanitize the user input in general.

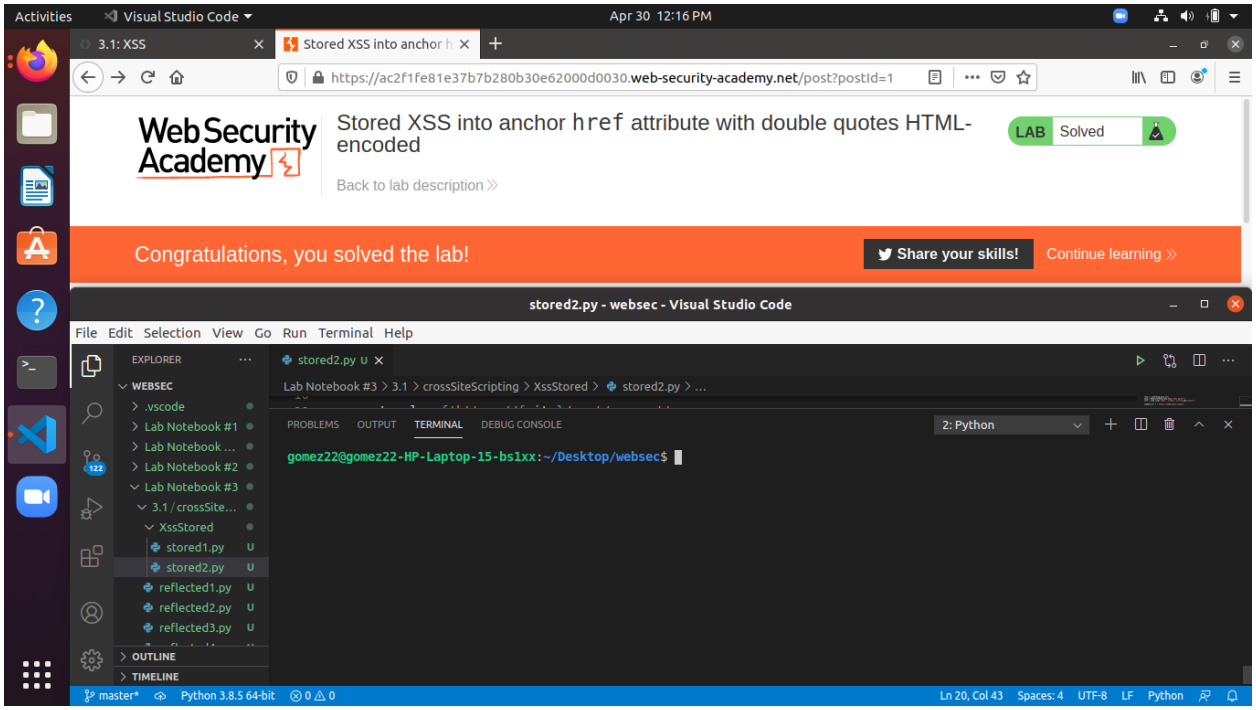
Take a screenshot showing that you have successfully added the OdinId attribute to the author's website link.



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



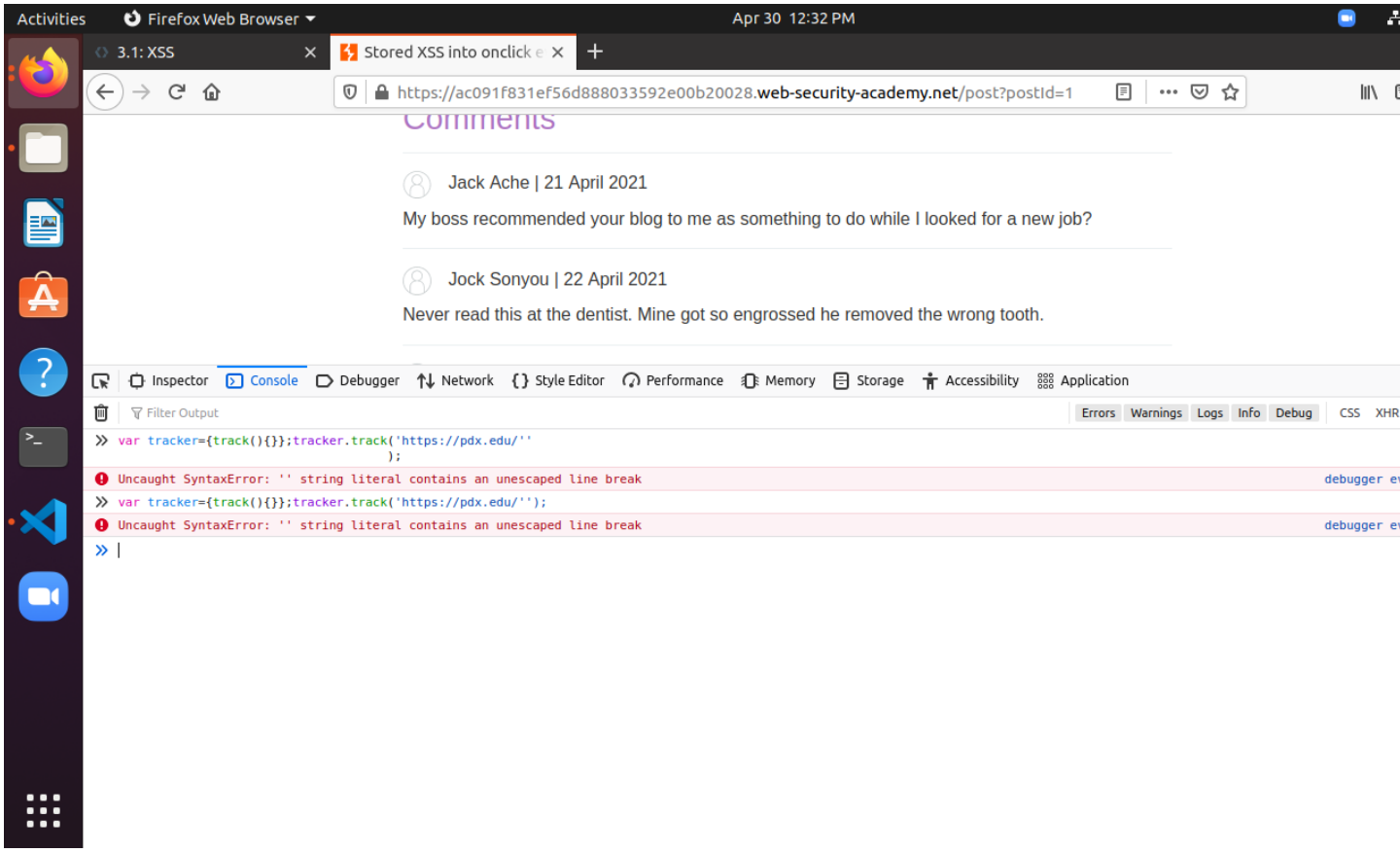
Take a screenshot showing completion of the level that includes your OdinId



Cross-site-scripting/stored (3)

-The vulnerability of this level is an XSS one where I can encode single quote characters in order to bypass the security features of posting a comment. This allows me to insert arbitrary code and hijack the site. To remediate this, the developers need to sanitize user input.

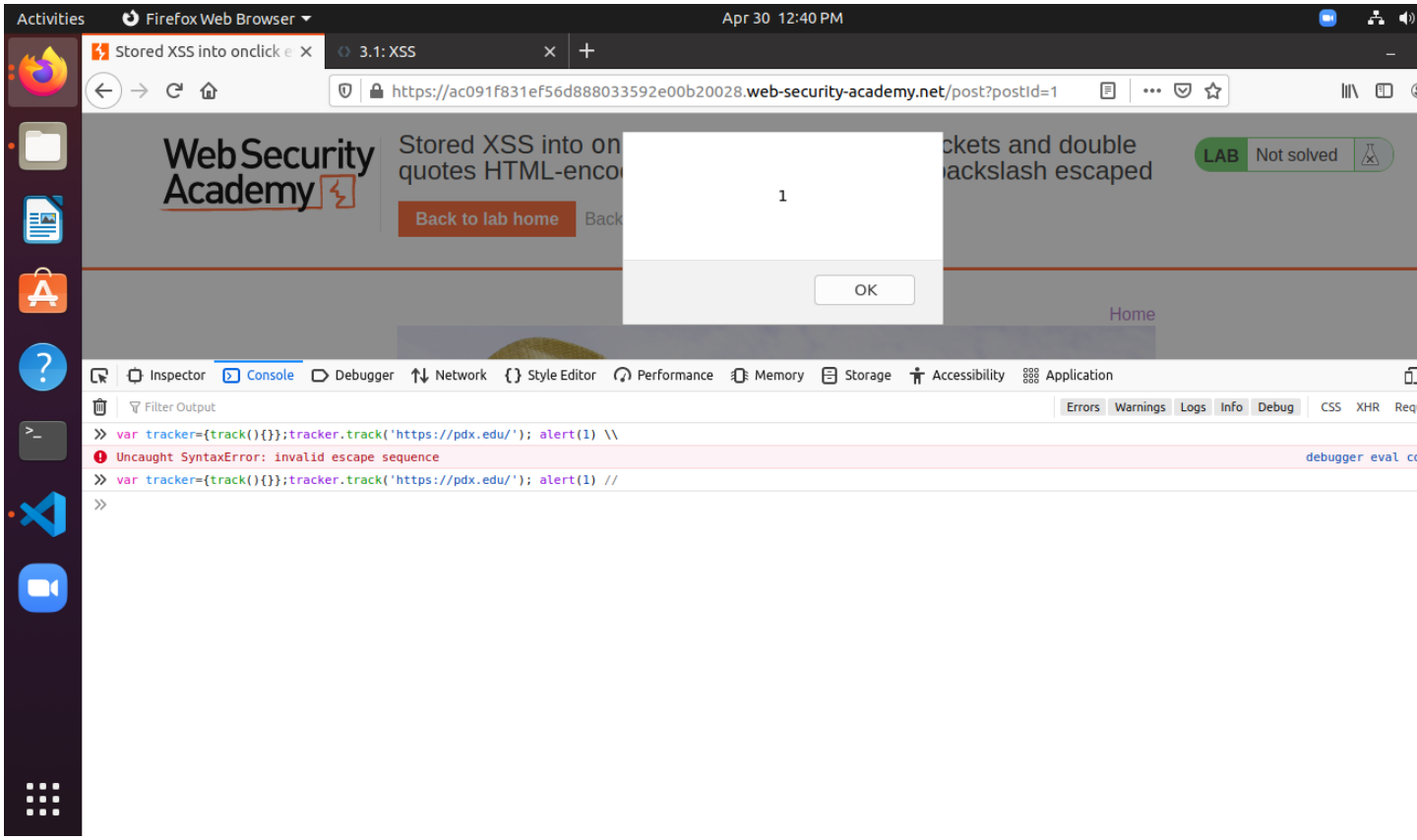
Replace the URL with https://pdx.edu' (e.g. the original URL with a single-quote added to break Javascript syntax). Execute the code and show a screenshot of the error that is returned.



Explain what happens when the URL is replaced with https://pdx.edu');//

-The syntax is still broken if you have the extra ‘ character inserted in it. I get an error from the console window; however, in the page source, an escape character is inserted to prevent syntax breaking.

Finally, using this URL, insert an alert(1); into it and take a screenshot of the results in the console including the pop-up



Explain why they differ

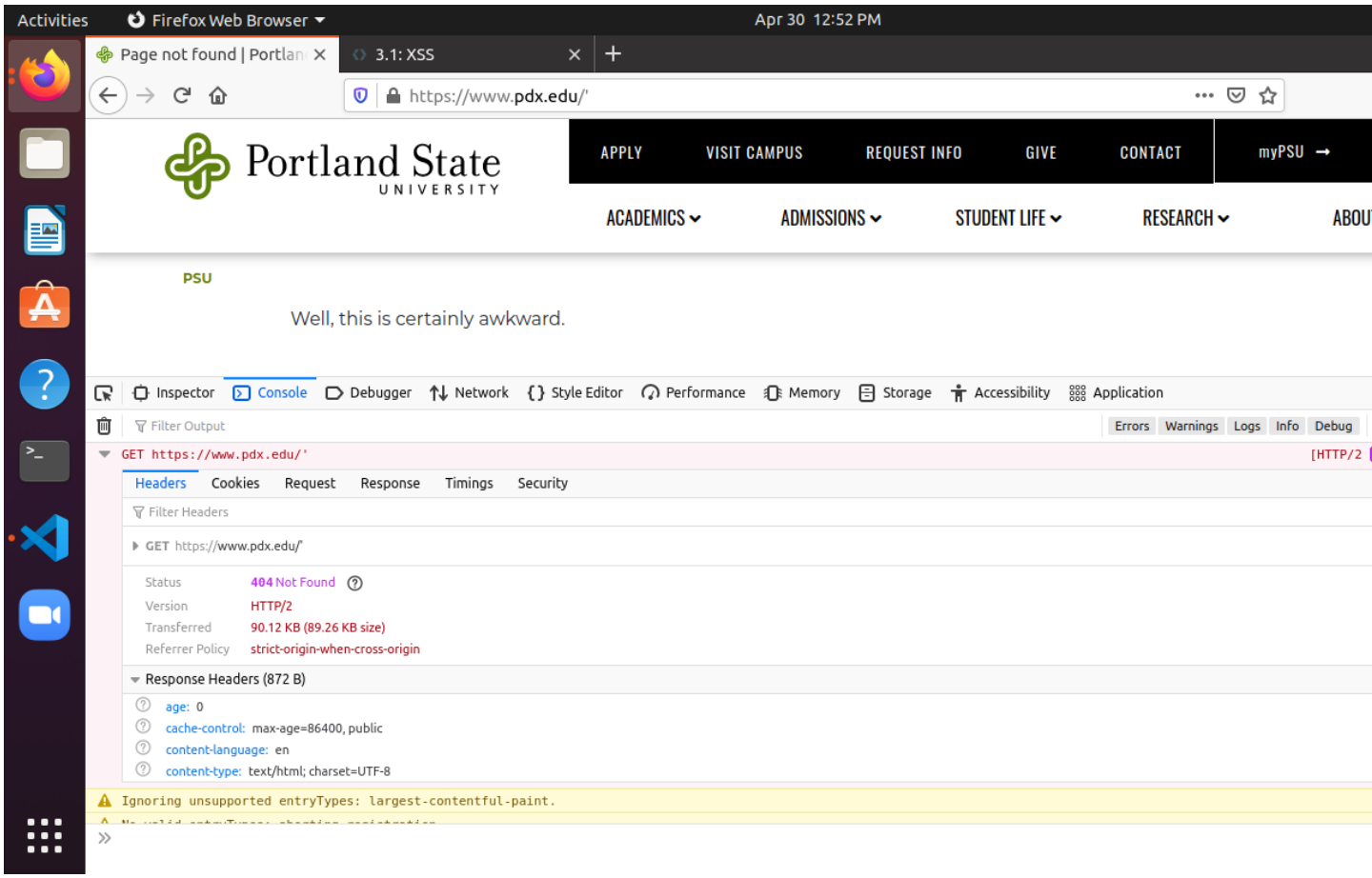
-They differ because the ‘ character is interpreted as ending the string accepted by .track which is contained in single quotes. An additional single quote needs to have an escape character appended to it to prevent the code from breaking. The double quote character “ does not break the syntax of the statement, so it does not need an escape character.

Are the results the same?

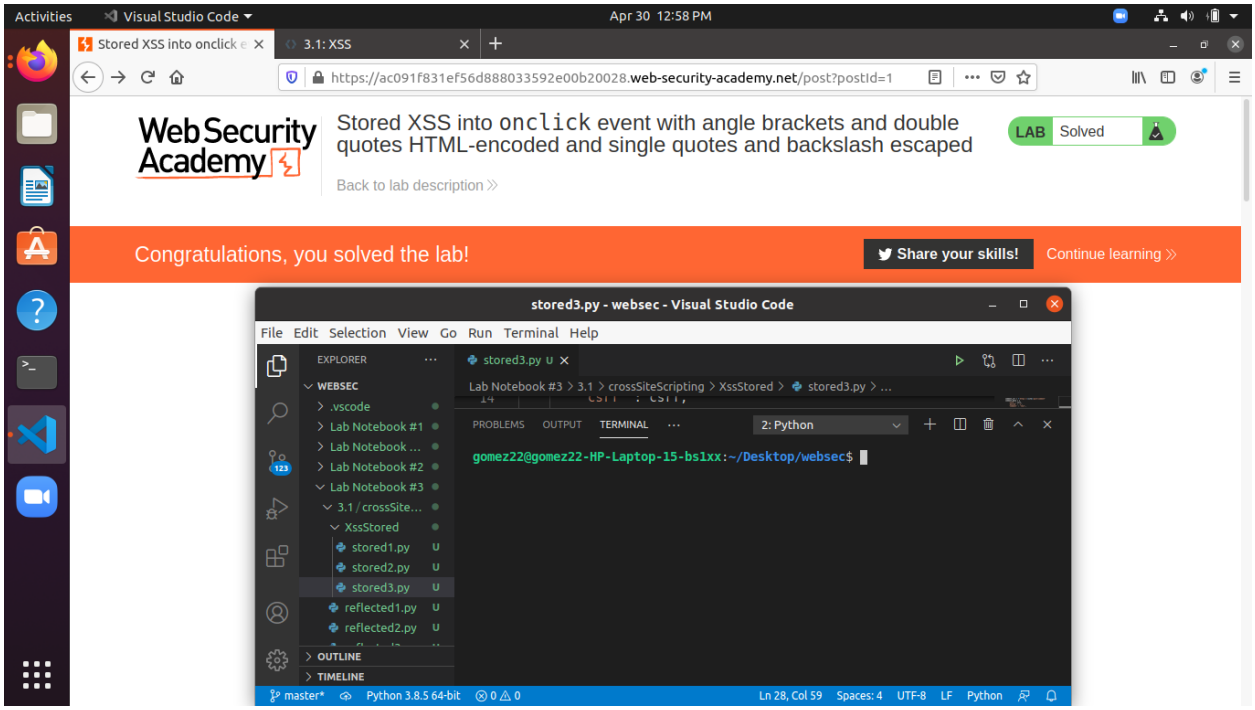
-Yes, the results are the same. In the href and in the .track parameter, the encoded double quote character is displayed as “.

Take a screenshot of the error message in the console. Is it similar to one that you have seen when crafting an exploit?

-The error message is different in that it resulted in a 404 error, but I can see that this is because the single quotes ‘ character was inserted into the url, indicating that I successfully broke syntax and can insert arbitrary code.



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

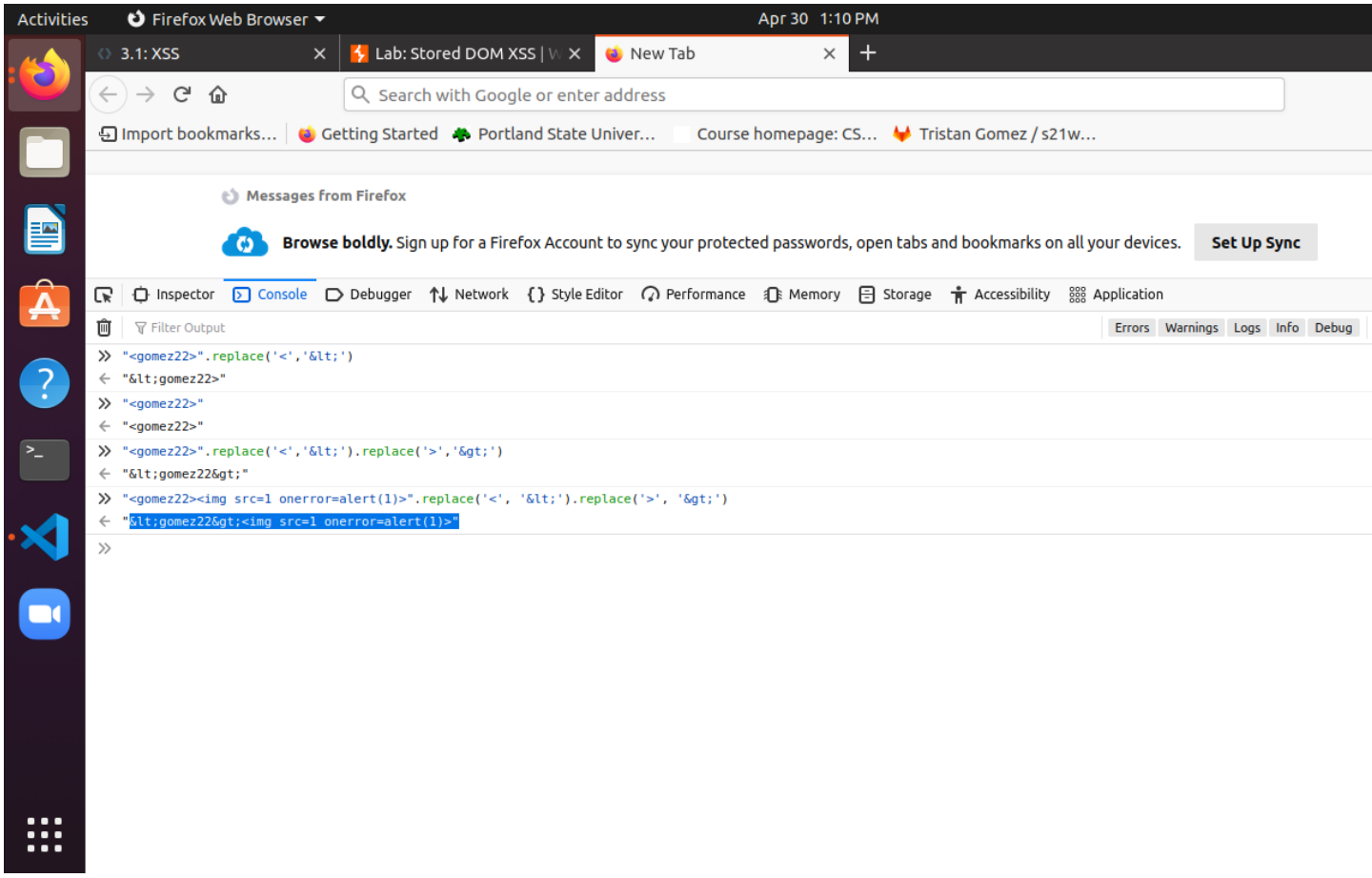


Cross-site-scripting/dom-based (6)

-The vulnerability in this lab is that user input is not fully sanitized which lets me insert html tags to get JavaScript code execution. To remediate this, the developers should properly sanitize all client information.

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?

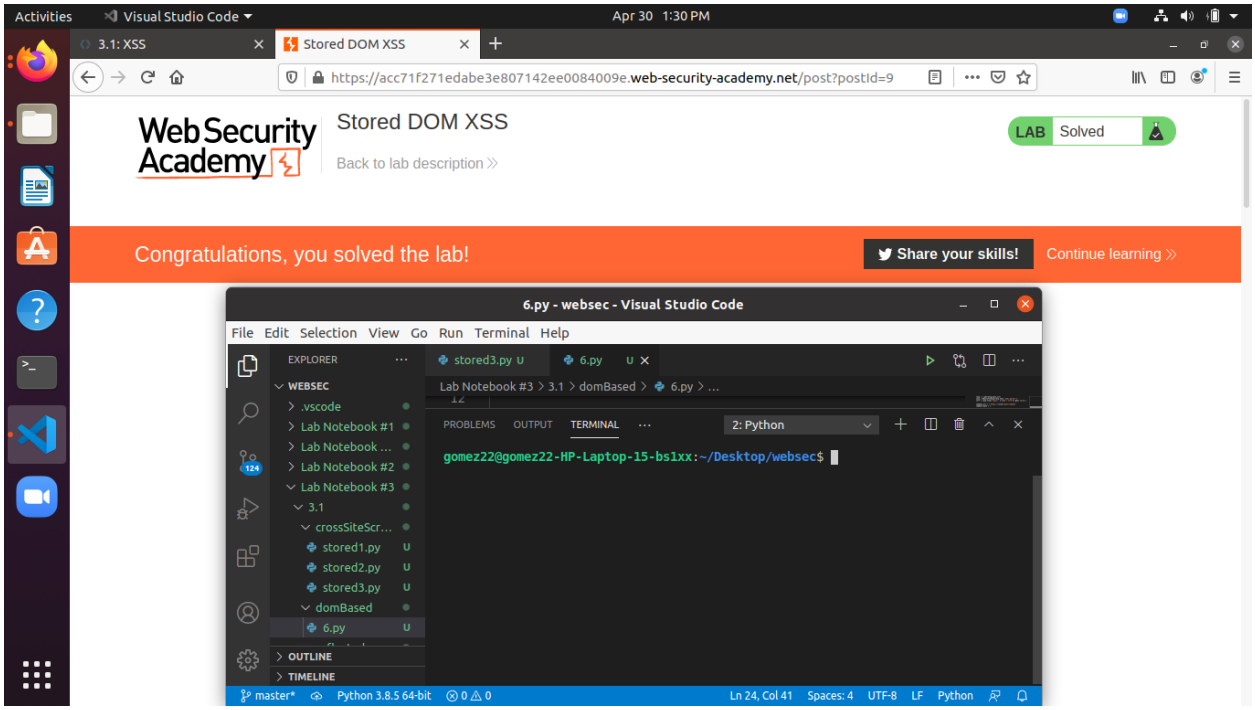
-The issue is that not all of the < or > characters are replaced, only the first instance of each character is replaced. The “replaceAll” method would fix this issue.



Which three fields of JSON are vulnerable to a cross-site scripting attack?

-Avatar, Author, and Body are all vulnerable to an XSS attack.

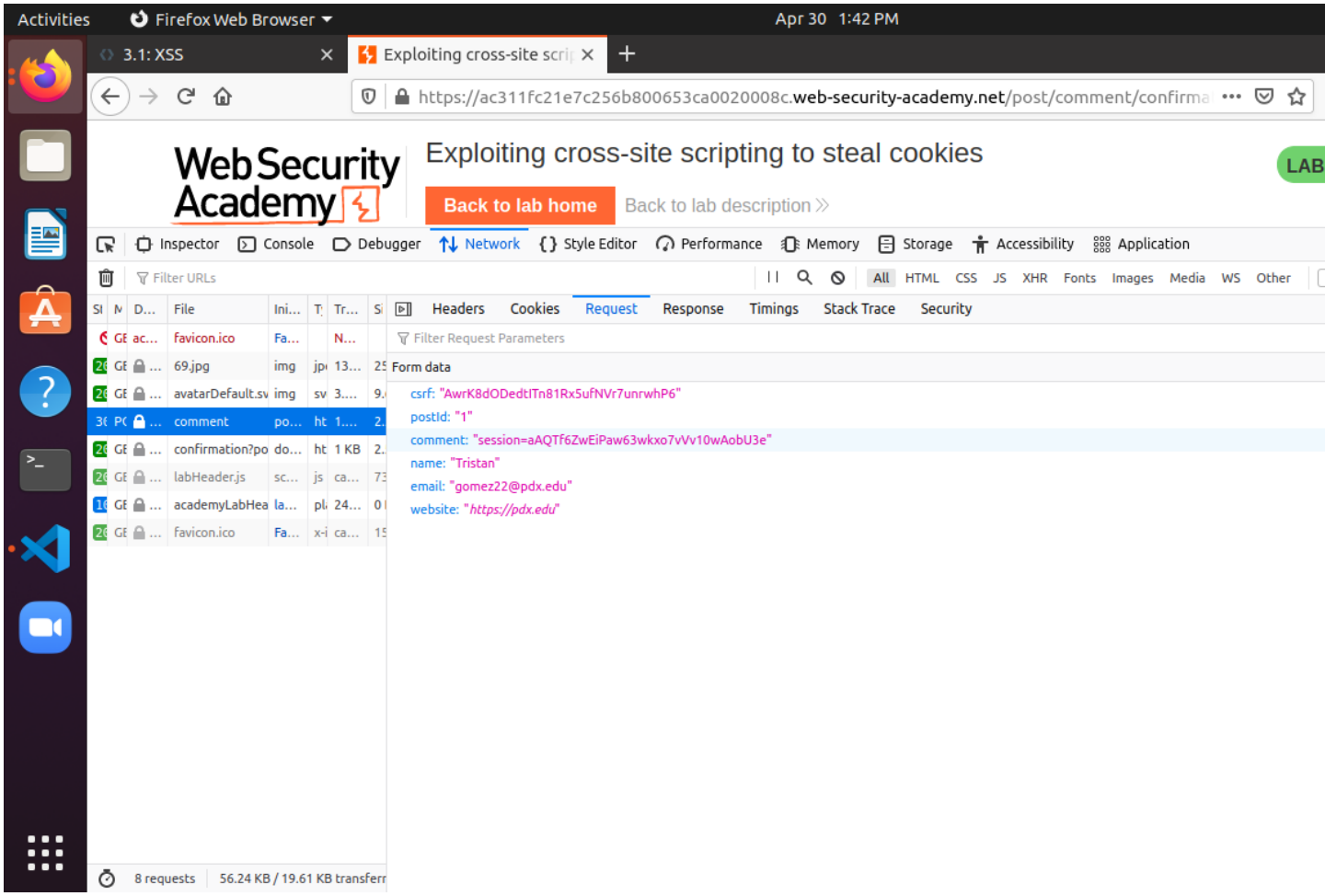
Take a screenshot showing completion of the level that includes your OdinId



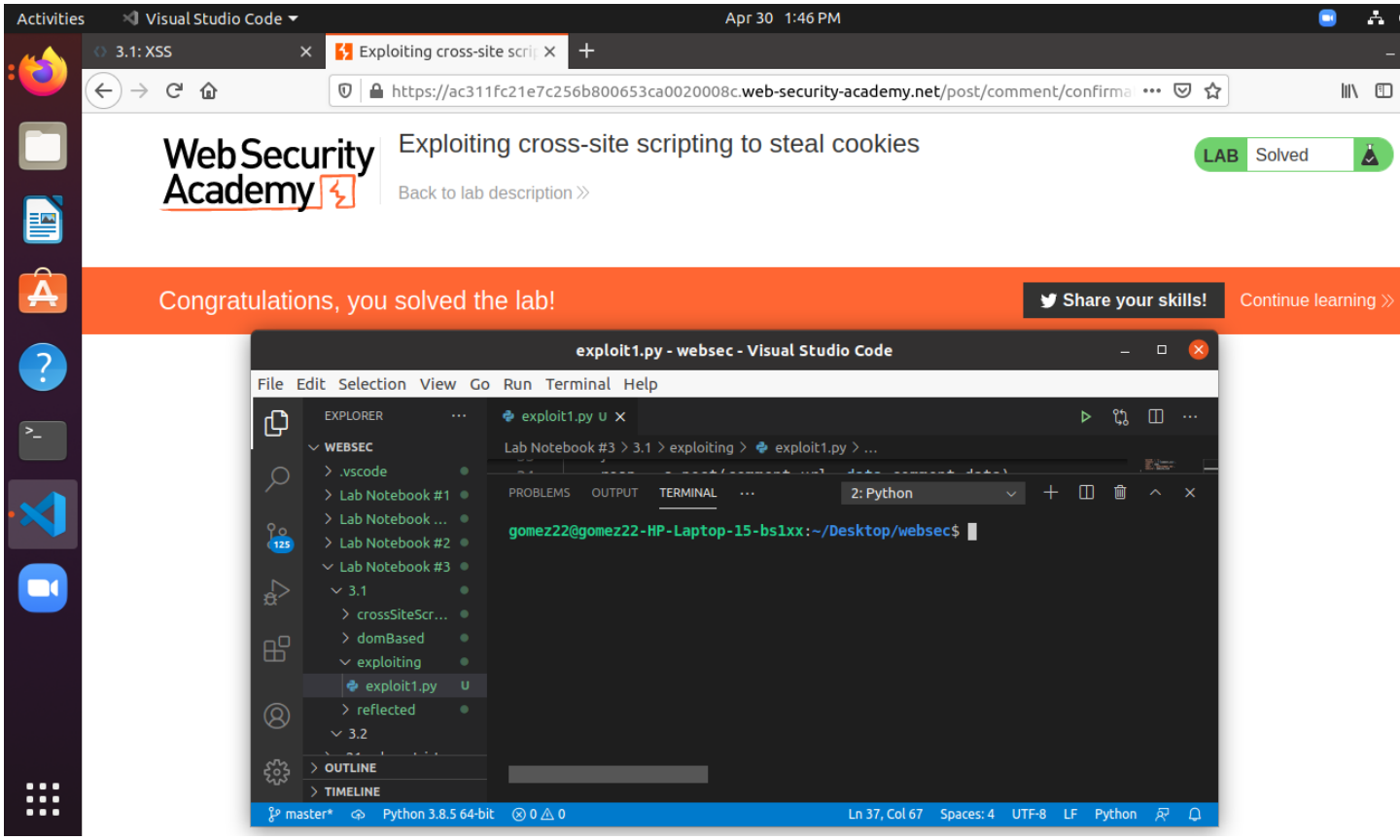
Cross-site-scripting/exploiting (1)

-The vulnerability in this lab is an XSS vulnerability. As an attacker I can insert a script tag and code to steal privileged user information of those who access the page I infected. To remediate this, the developers should sanitize all user input.

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

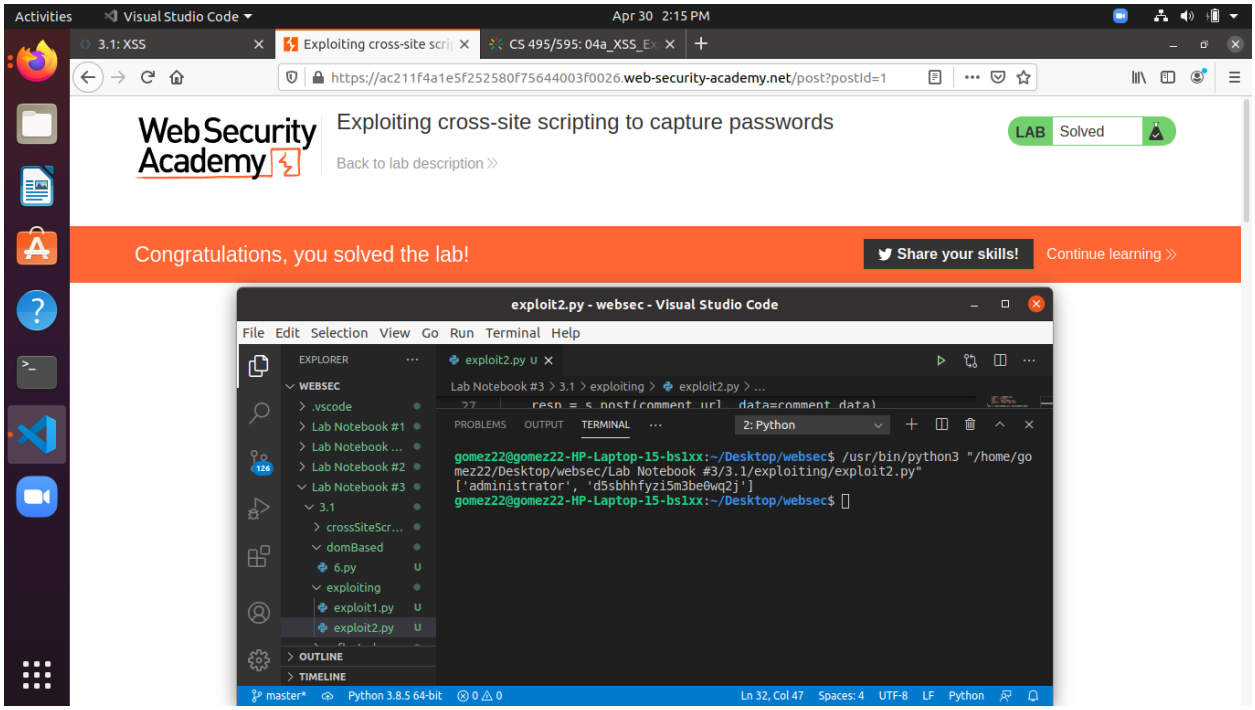


Take a screenshot showing completion of the level that includes your OdinId



Cross-site-scripting/exploit (2)

-This level is vulnerable to an XSS injection attack where I insert code that exfiltrates a person's username and password upon them loading the infected page. To remediate this, the developers should sanitize all user input before rendering it to the dom.



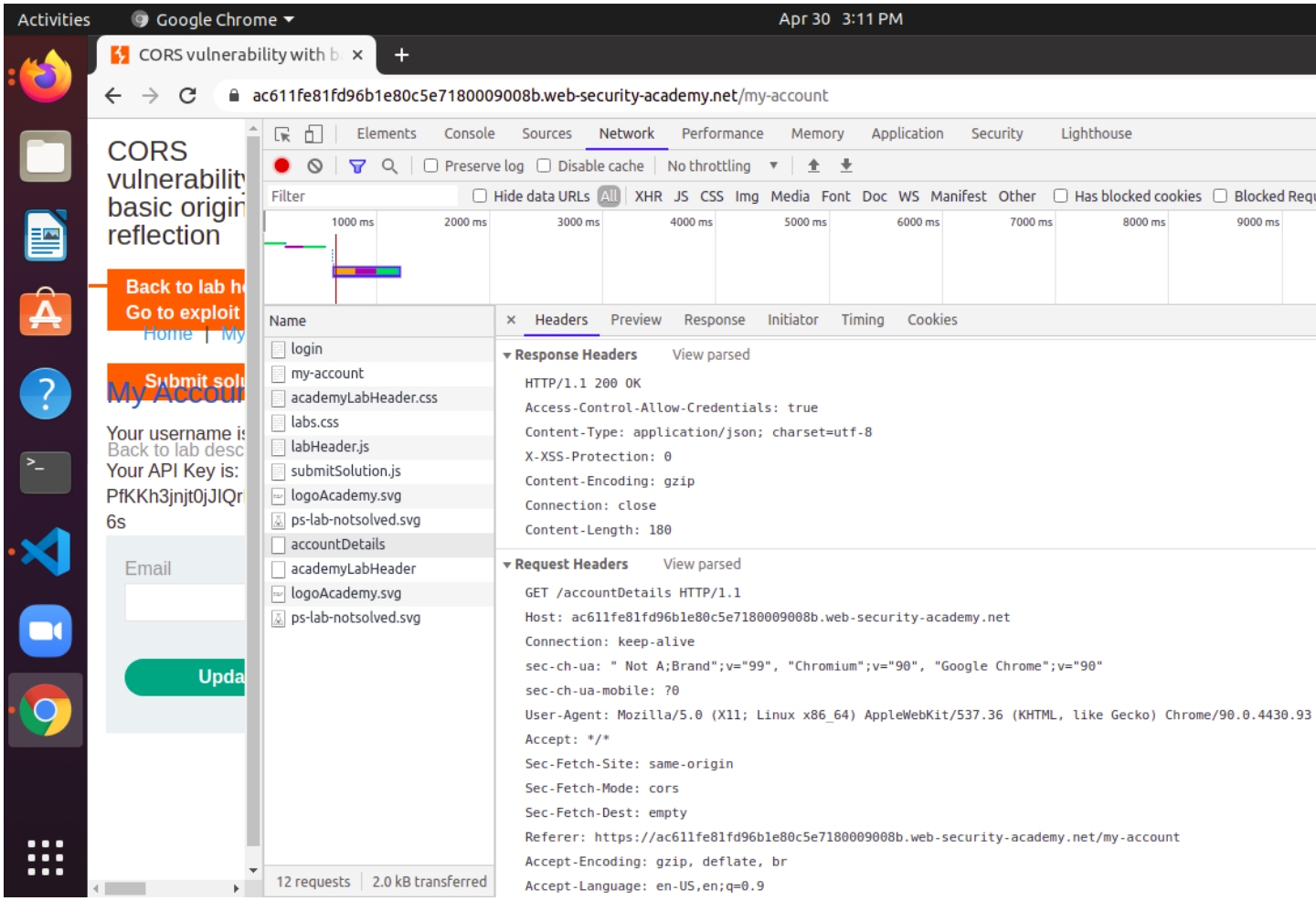
3.2 Cors

Cors (1)

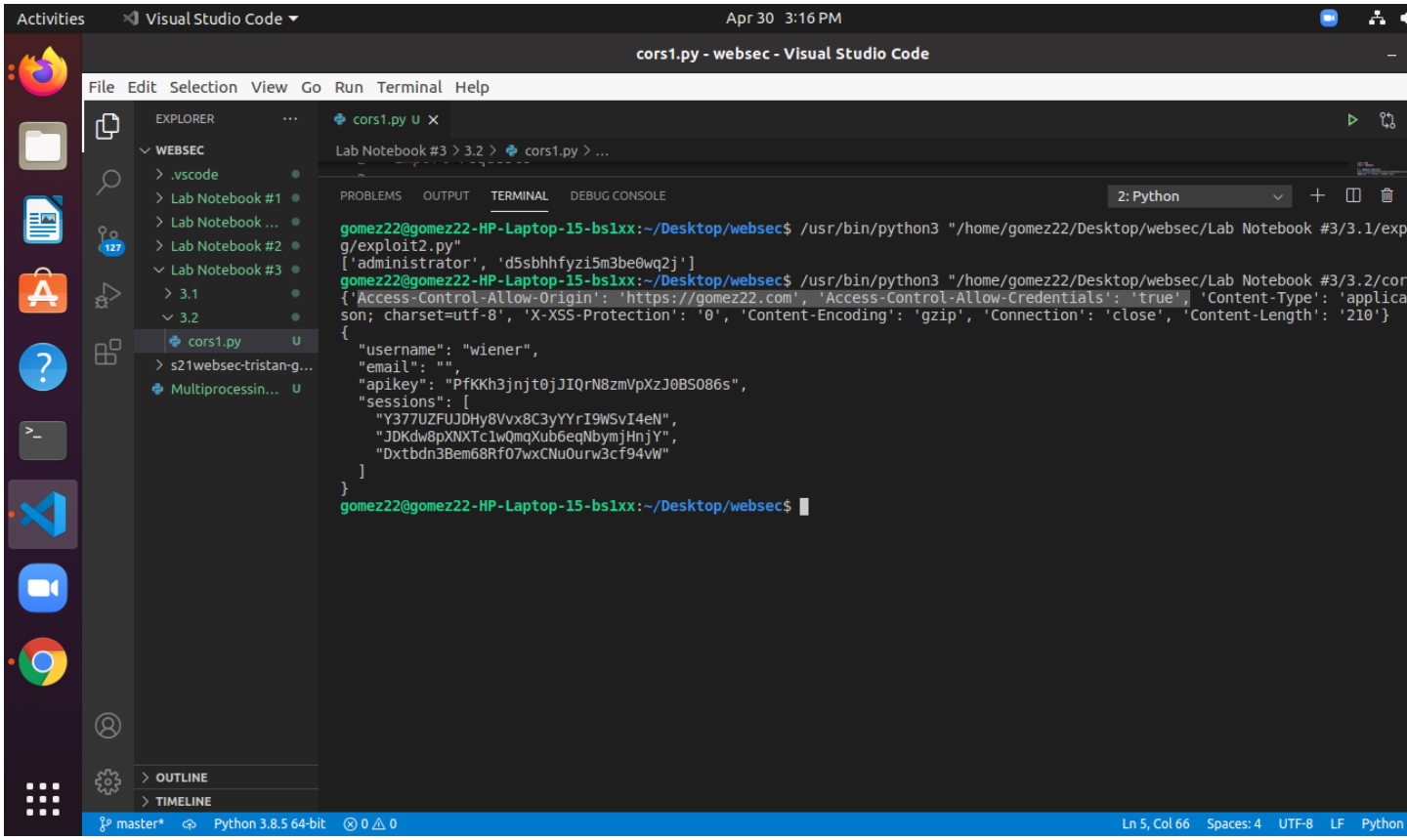
-The vulnerability of this lab is that it allows access to all sites by reflecting the origin header contents into the Access-Control-Allow-Origin header. To remediate this vulnerability the developers should have explicitly listed out only the allowed sites in the Access-Control-Allow-Origin header.

Show a screenshot of the CORS header that enables credentials to be sent to get the key.

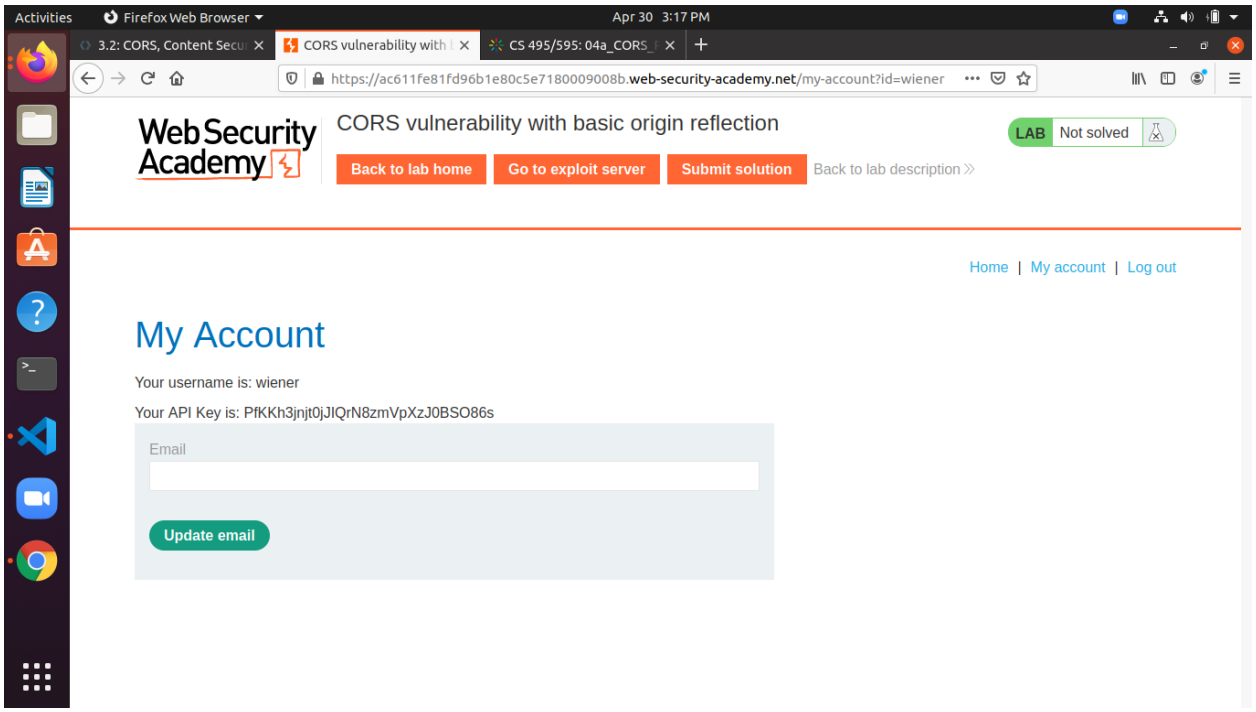
-The header is “Access-Control-Allow-Credentials”



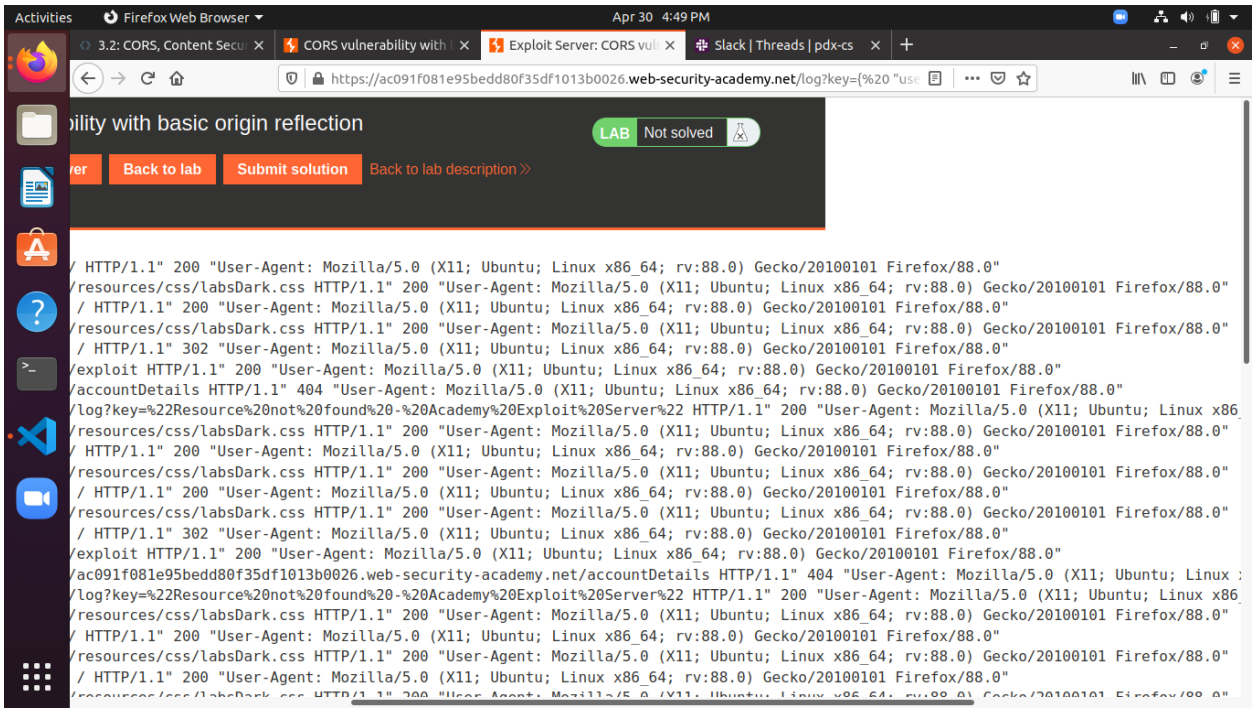
Show a screenshot of the headers in the response to see that the site simply reflects the header in Access-Control-Allow-Origin:

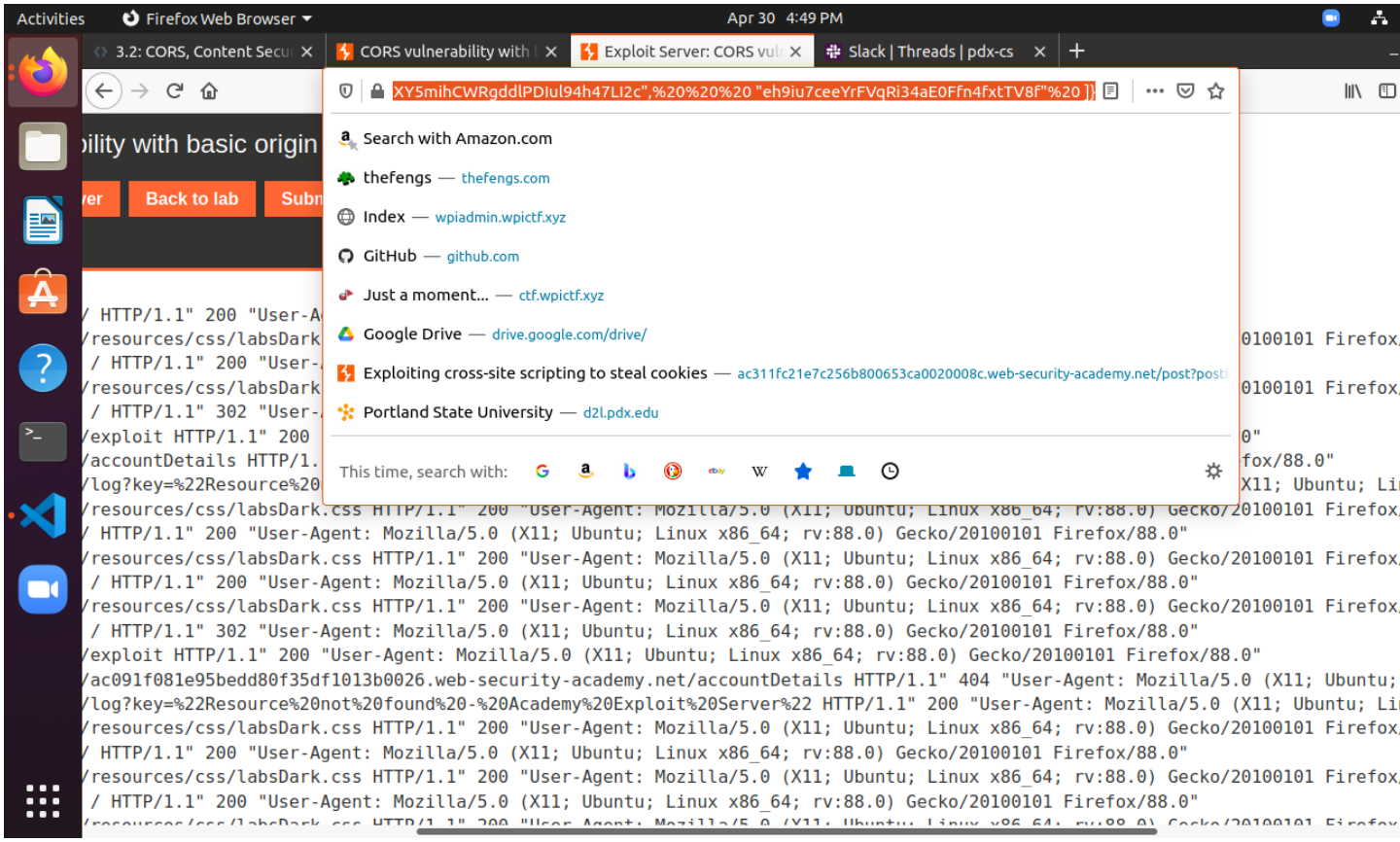


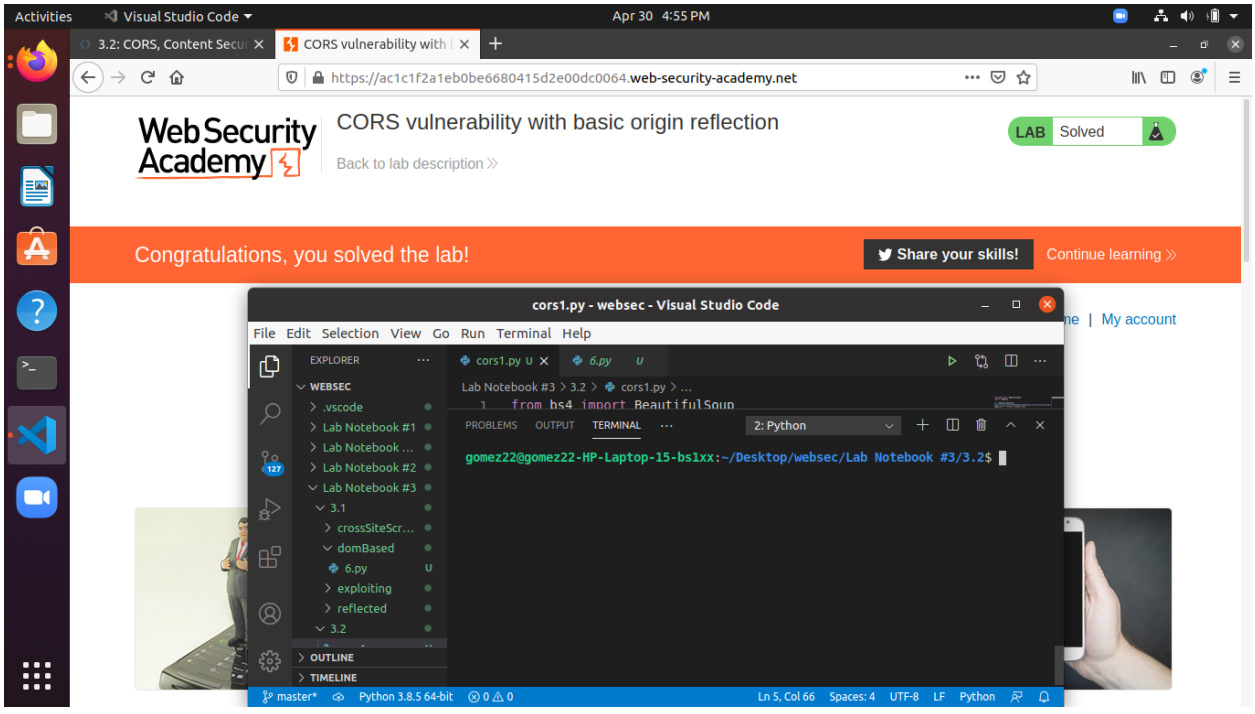
Take a screenshot of the API key and include it in your lab notebook.



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



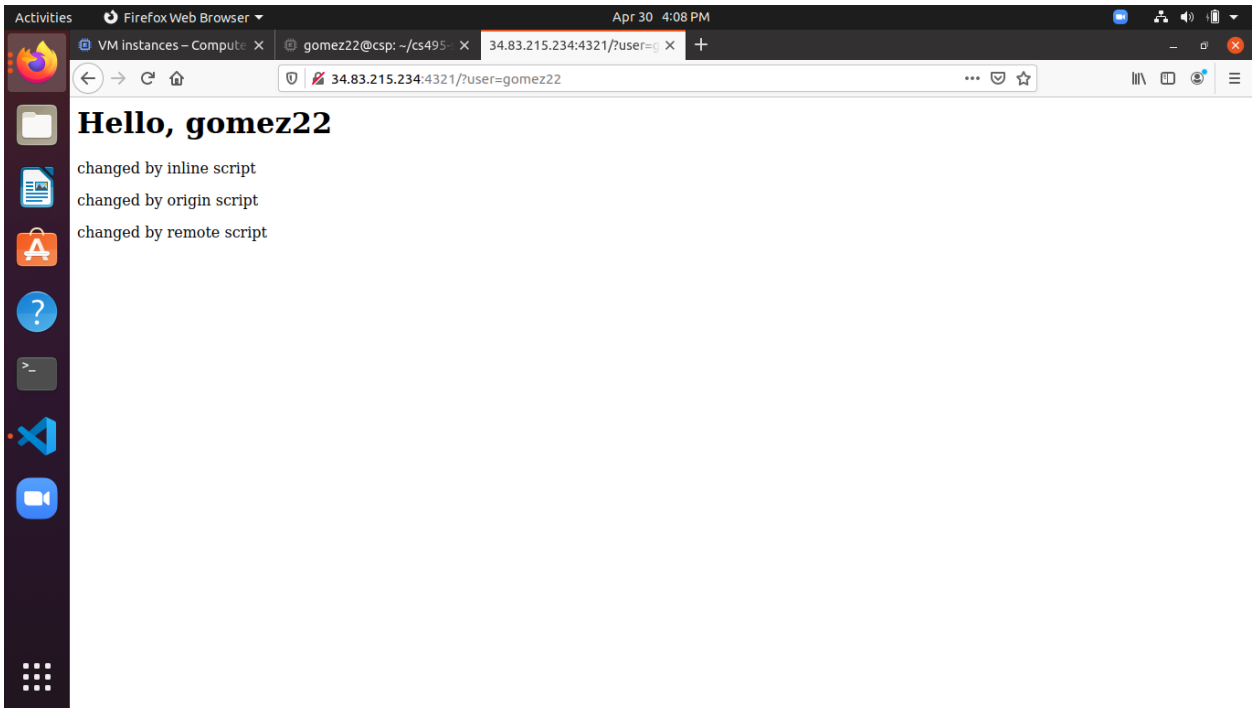

Activities
Firefox Web Browser
Apr 30 4:49 PM
3.2: CORS, Content Secu
CORS vulnerability with
Exploit Server: CORS vul
Slack | Threads | pdx-cs
XY5mihCWRgddlPDIul94h47LI2c",%20%20%20 "eh9iu7ceeYrFVqRi34aE0Ffn4fxtTV8f"%20 }]
Search with Amazon.com
thefengs — thefengs.com
Index — wpiadmin.wpictf.xyz
GitHub — github.com
Just a moment... — ctf.wpictf.xyz
Google Drive — drive.google.com/drive/
Exploiting cross-site scripting to steal cookies — ac311fc21e7c256b800653ca0020008c.web-security-academy.net/post?posti
Portland State University — d2l.pdx.edu
This time, search with:
ility with basic origin
Back to lab
/ HTTP/1.1" 200 "User-A
/resources/css/labsDark
/ HTTP/1.1" 200 "User-A
/resources/css/labsDark
/ HTTP/1.1" 302 "User-A
/exploit HTTP/1.1" 200
/accountDetails HTTP/1.
/log?key=%22Resource%20
/resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/ HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/ HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/ HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/exploit HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/ac091f081e95bedd80f35df1013b0026.web-security-academy.net/accountDetails HTTP/1.1" 404 "User-Agent: Mozilla/5.0 (X11; Ubuntu;
/log?key=%22Resource%20not%20found%20-%20Academy%20Exploit%20Server%22 HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
/resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox
/ HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
/resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox
/ HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"


Activities
Visual Studio Code
Apr 30 4:55 PM
3.2: CORS, Content Secu
CORS vulnerability with
https://ac1c1f2a1eb0be6680415d2e00dc0064.web-security-academy.net
Web Security Academy
CORS vulnerability with basic origin reflection
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
My account
cors1.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBSEC
.vscode
Lab Notebook #1
Lab Notebook ...
Lab Notebook #2
Lab Notebook #3
3.1
crossSiteScr...
domBased
6.py
exploiting
reflected
3.2
OUTLINE
TIMELINE
cors1.py U
6.py U
Lab Notebook #3 > 3.2 > cors1.py > ...
from bs4 import BeautifulSoup
PROBLEMS OUTPUT TERMINAL
2: Python
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #3/3.2$
master*
Python 3.8.5 64-bit
Ln 5, Col 66
Spaces: 4
UTF-8
LF
Python

CORS - CSP

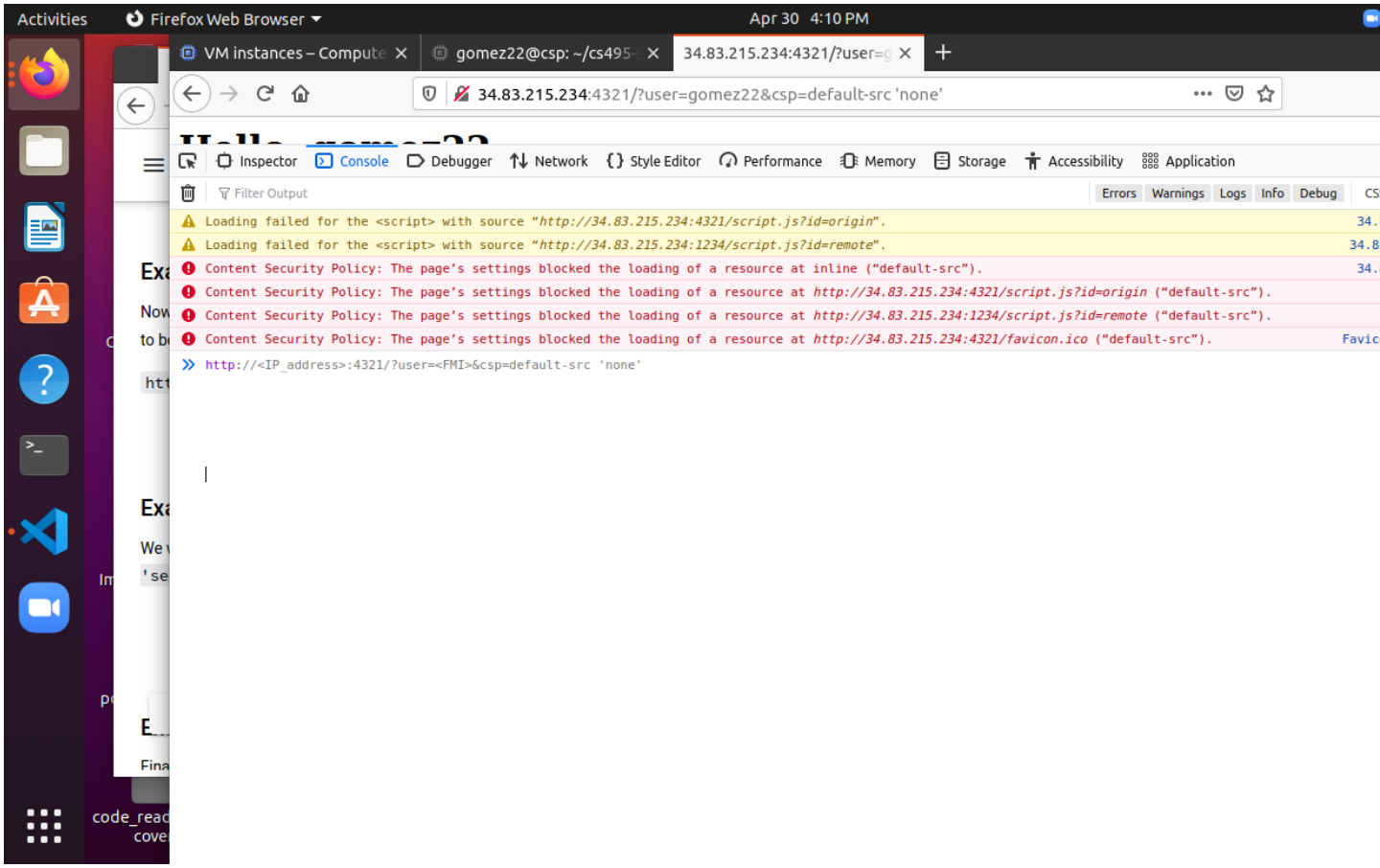
Example #1

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

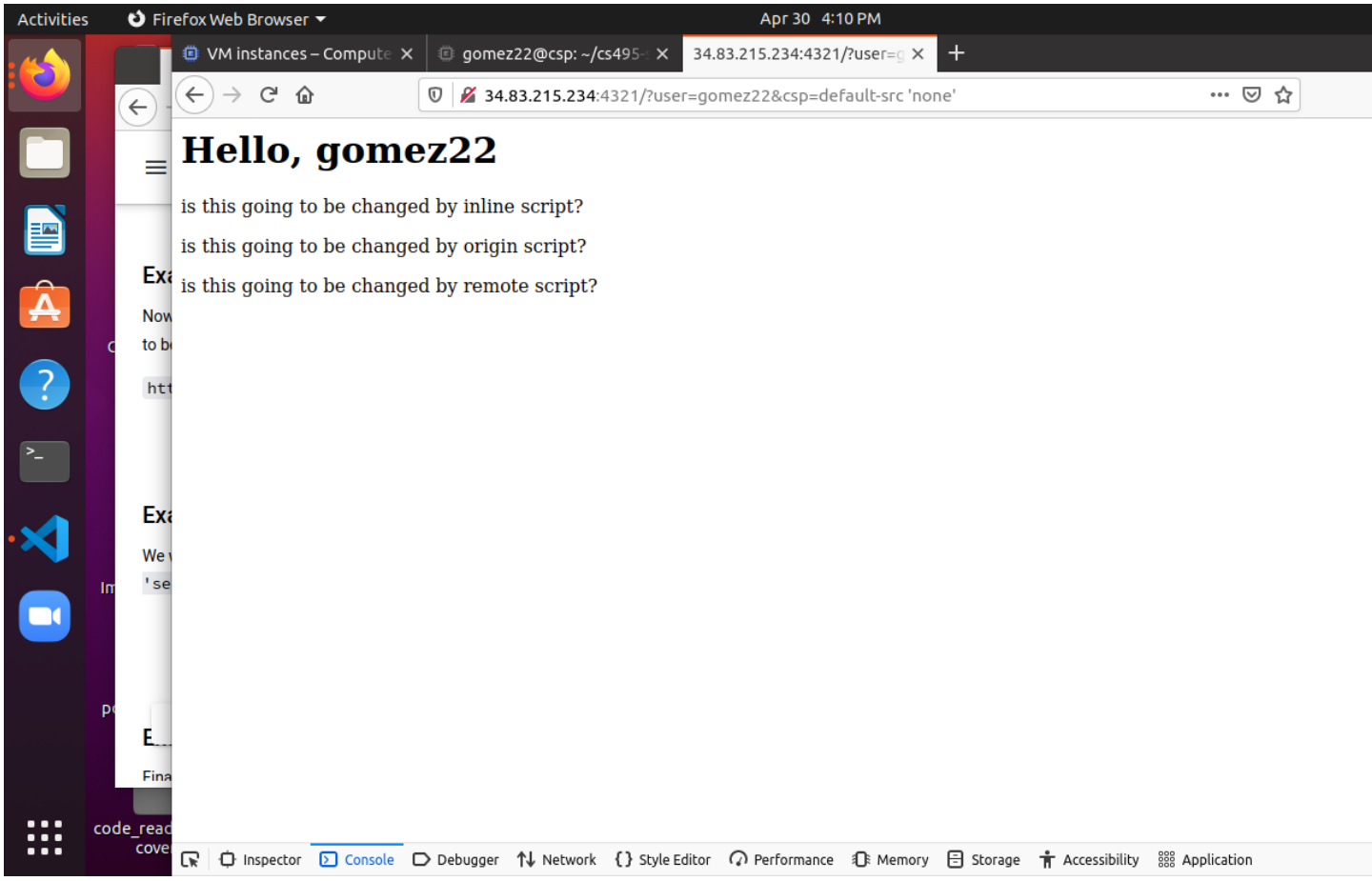


Example #2

Take a screenshot of the console output showing all scripts blocked



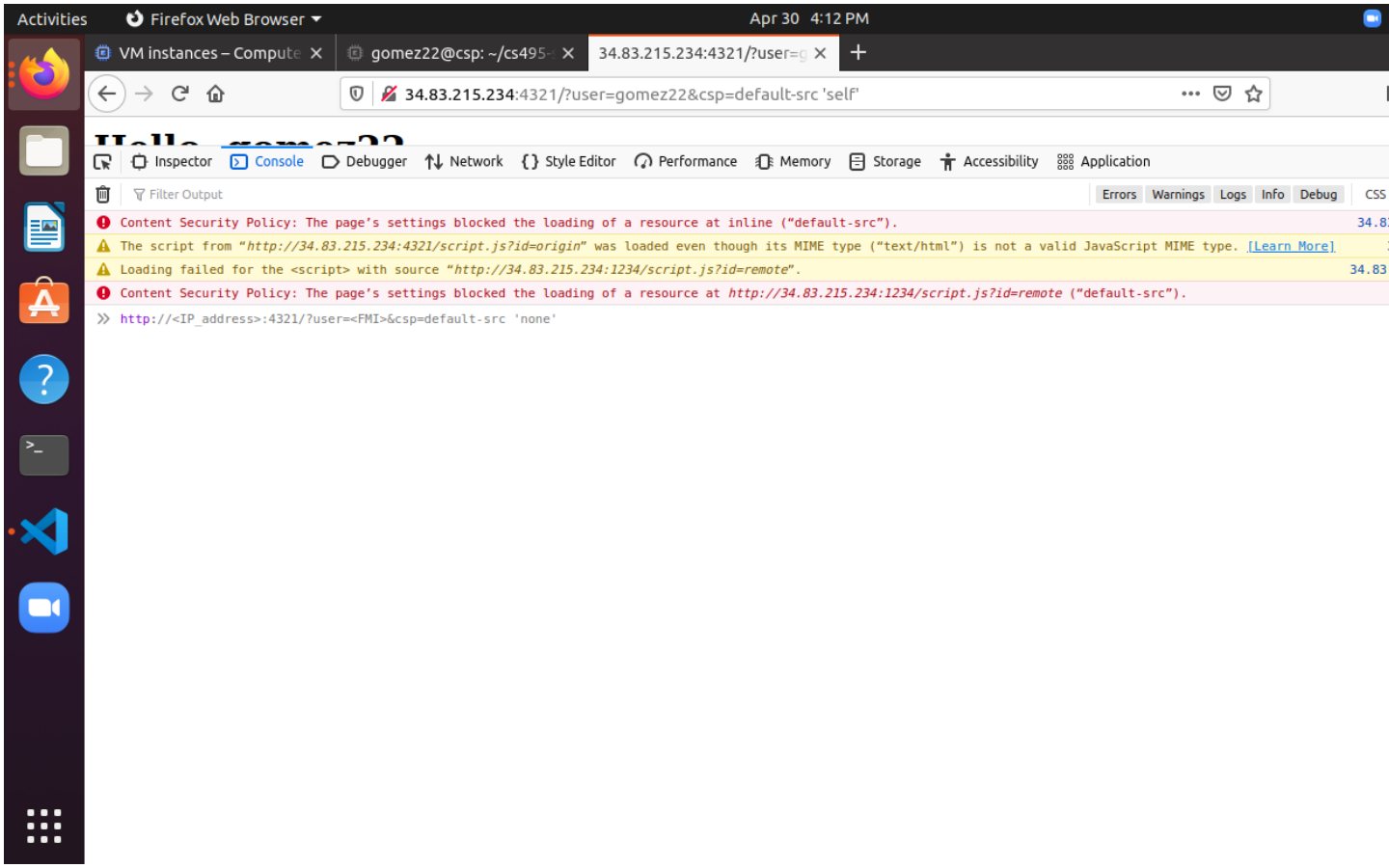
Take a screenshot of the page result that includes the URL in the browser



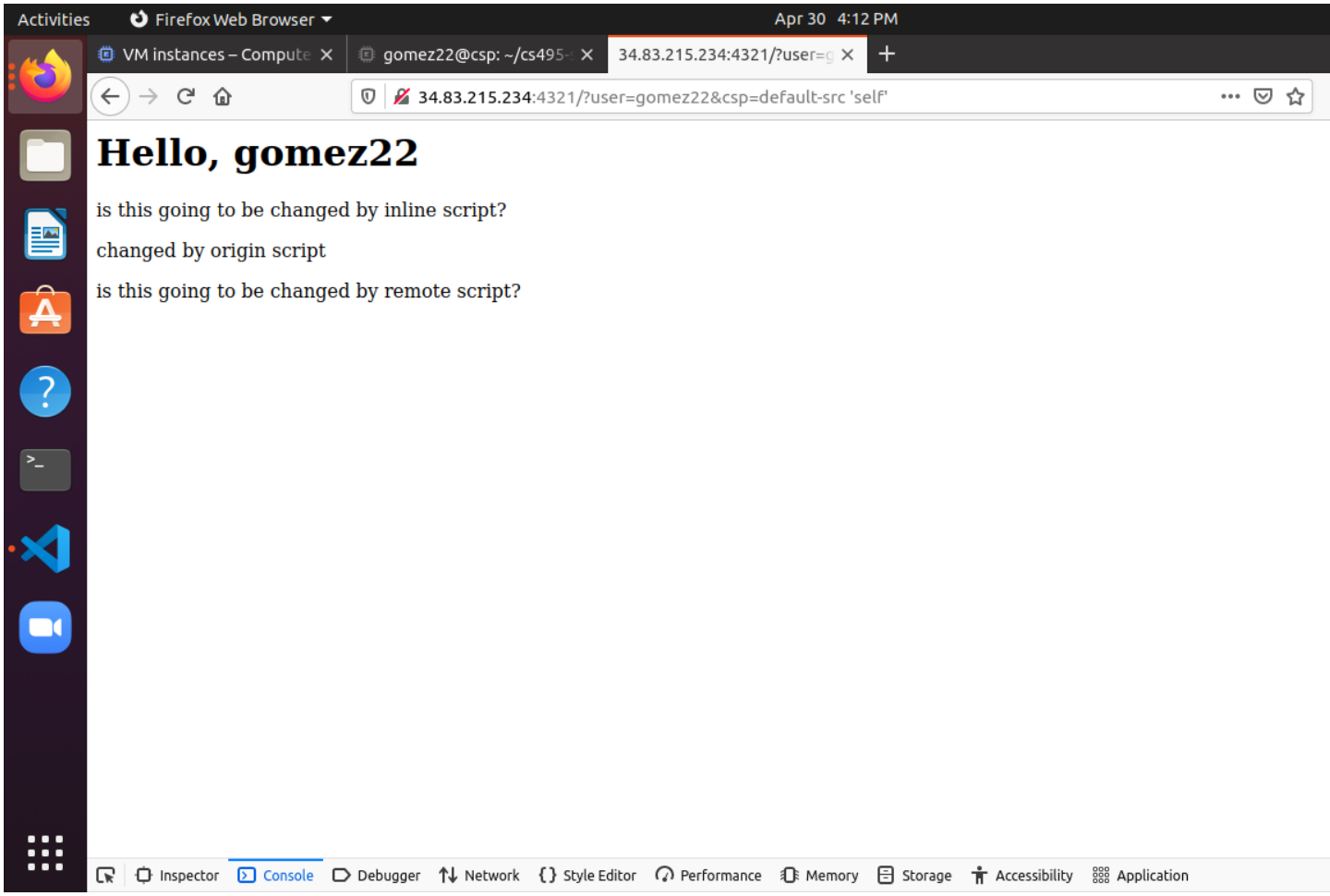
Example #3

Take a screenshot of the console output showing the scripts that have been blocked. Explain why these results differ from the previous example.

-Two of the three scripts were blocked. This is different from the previous example because we set the CSP to allow origin/self requests so the javascript was able to be executed since it is now allowed by the CS policy.



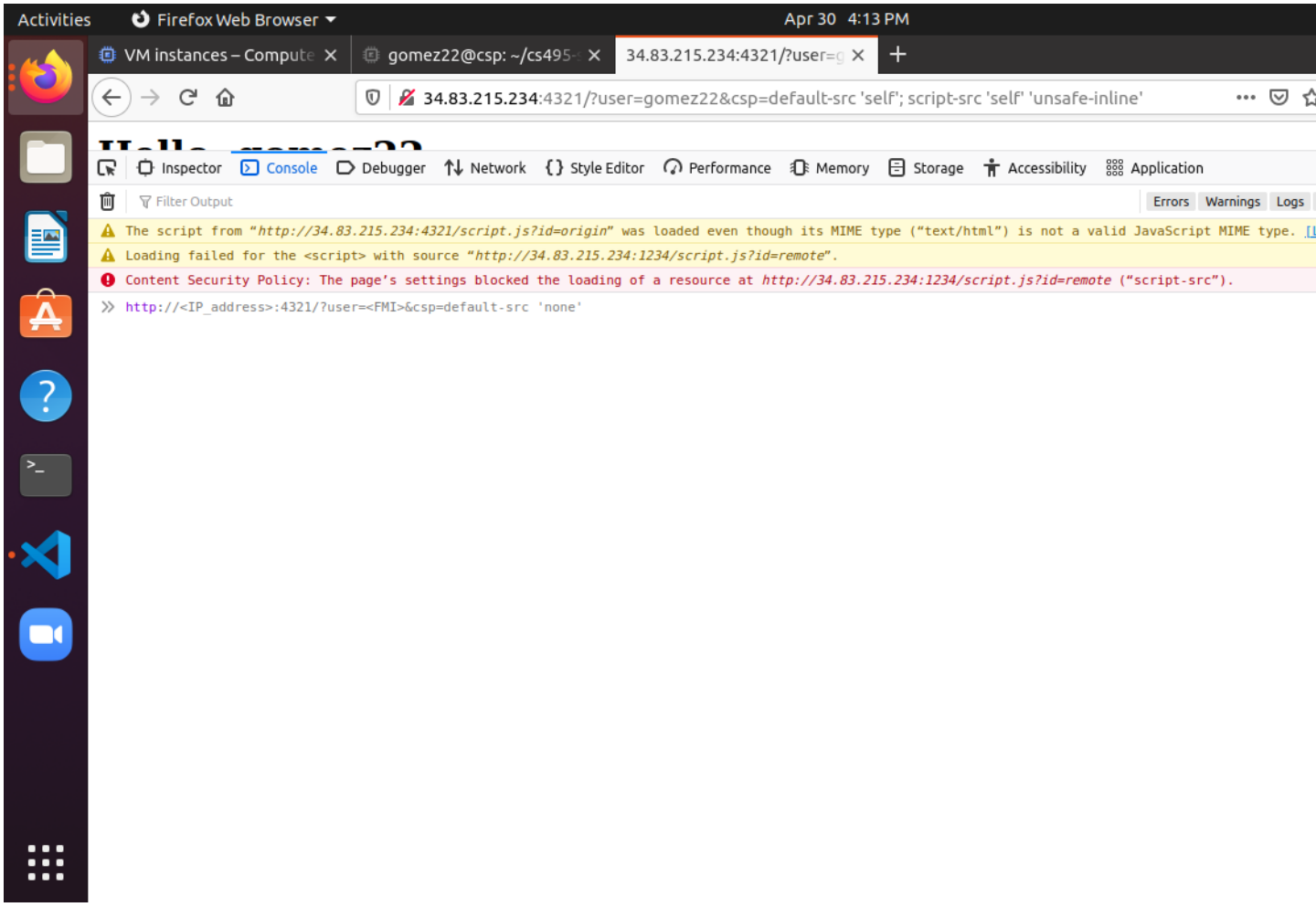
Take a screenshot of the page result that includes the URL in the browser



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.

-These results now allow 2 of the three scripts to load on the page. Again, self is a parameter to CSP which allows the origin/self script to execute. Now, the ‘unsafe-inline’ parameter is passed to the CSP which sets the permissions to let the inline script execute/load on the page.



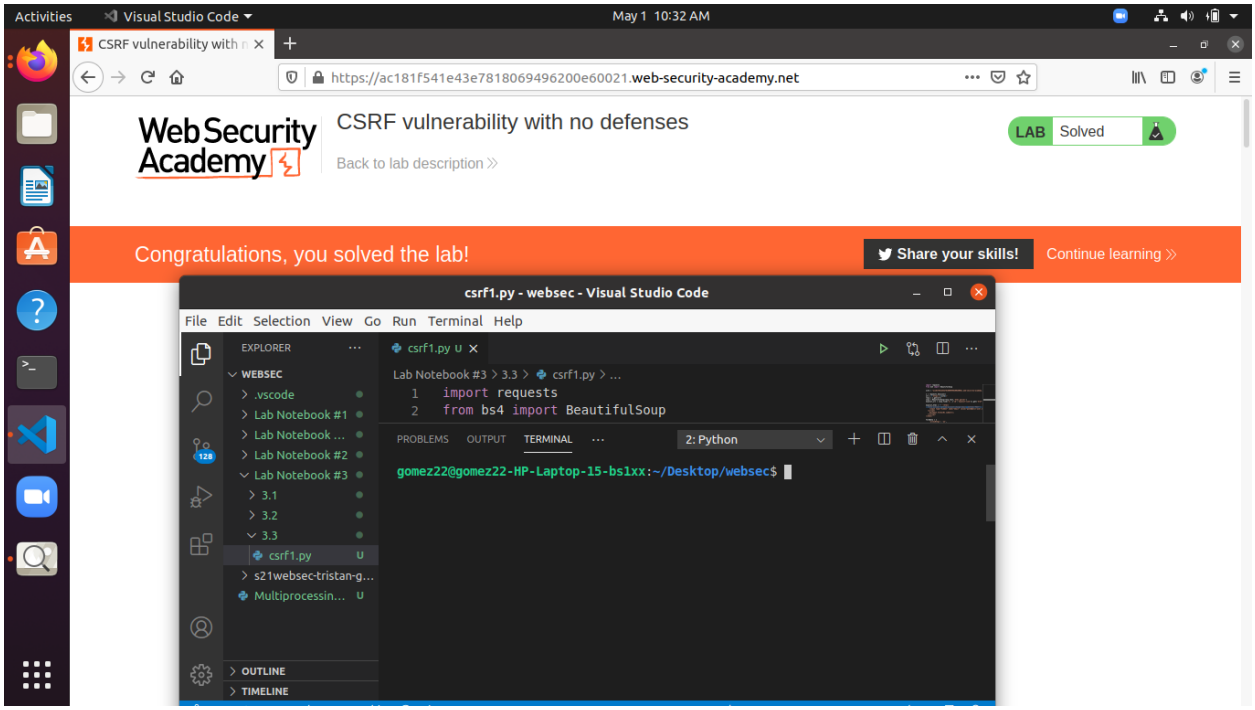
Take a screenshot of the page result that includes the URL in the browser

3.3 CSRF

Csrf (1)

-The vulnerability of this lab is that it does not use a CSRF token that binds the form submission to a previously issued HTML form. As an attacker, I can leverage this vulnerability to steal privileged information. To remediate this, the site developers need to issue CSRF tokens that bind form submission to previously issued HTML forms.

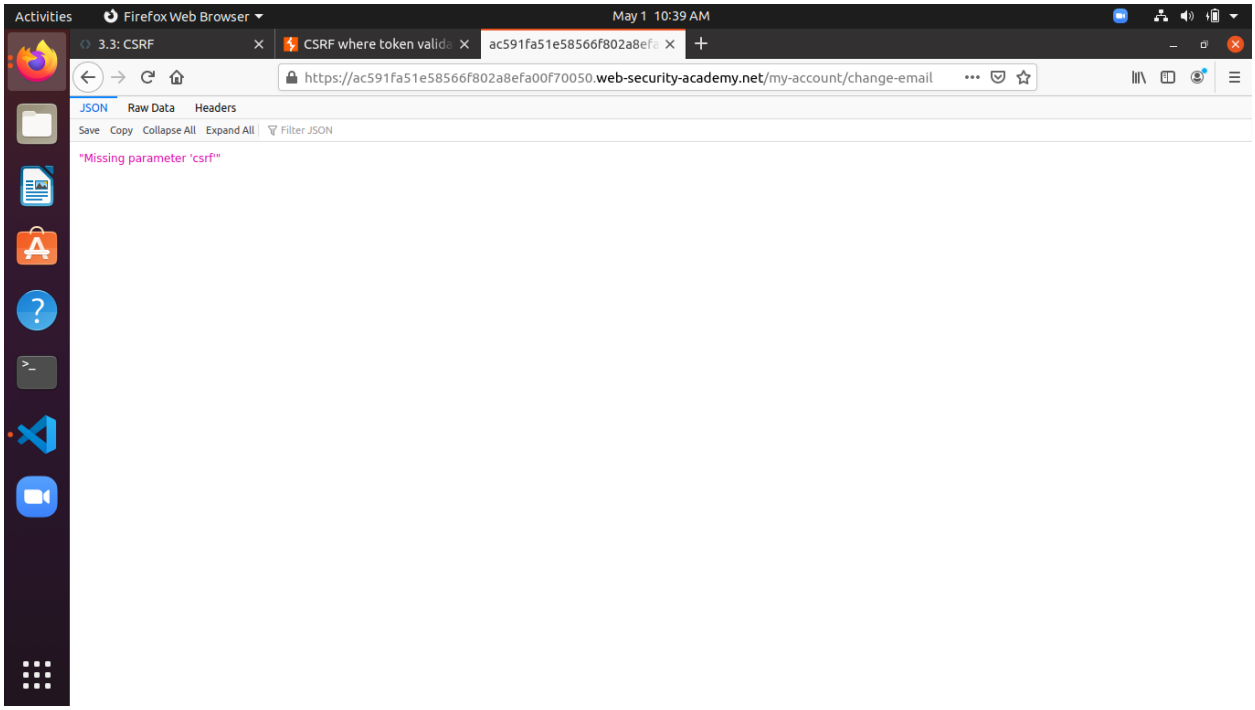
Take a screenshot showing completion of the level that includes your OdinId



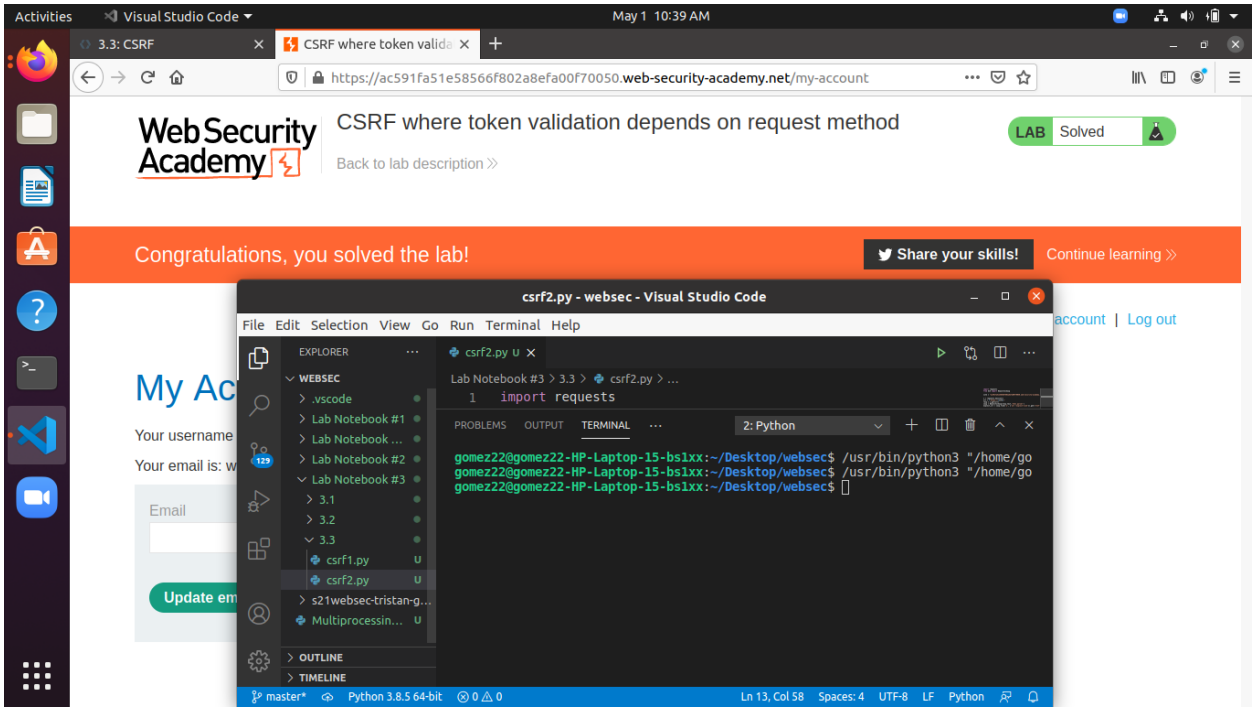
Csrf (2)

-The vulnerability of this lab is that the site developers did not protect all routes and all HTTP methods used on those routes with CSRF tokens. As an attacker, I found an unprotected method “GET” which I was able to exploit. To remediate this, the developers should account for all routes and methods like “GET”.

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

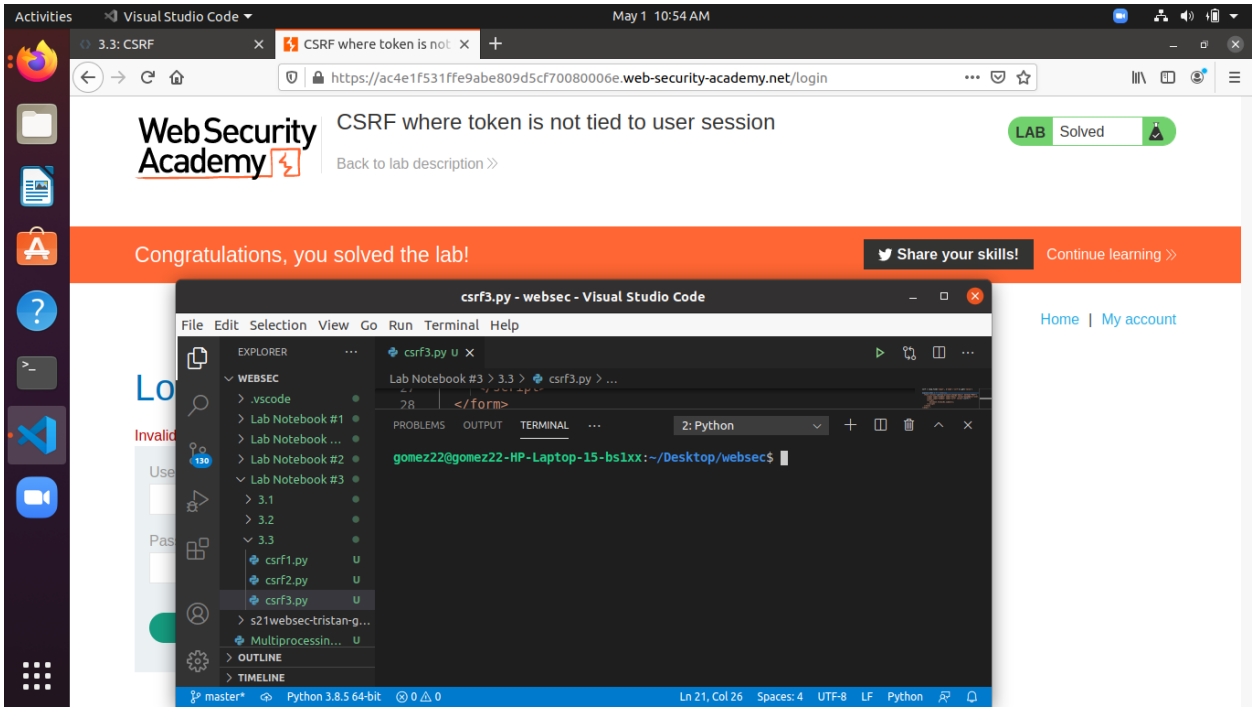


Take a screenshot showing completion of the level that includes your OdinId



Csrf (3)

-The vulnerability in this lab is that the site attempts to protect privileged tasks with a CSRF token; however, the token is not tied to unique user sessions. As an attacker, I can get a CSRF token and use it to get access to another person's protected info. In this case, use it to change a victim's email address. To remediate this, the site developers need to make CSRF tokens tie in to individual user sessions.

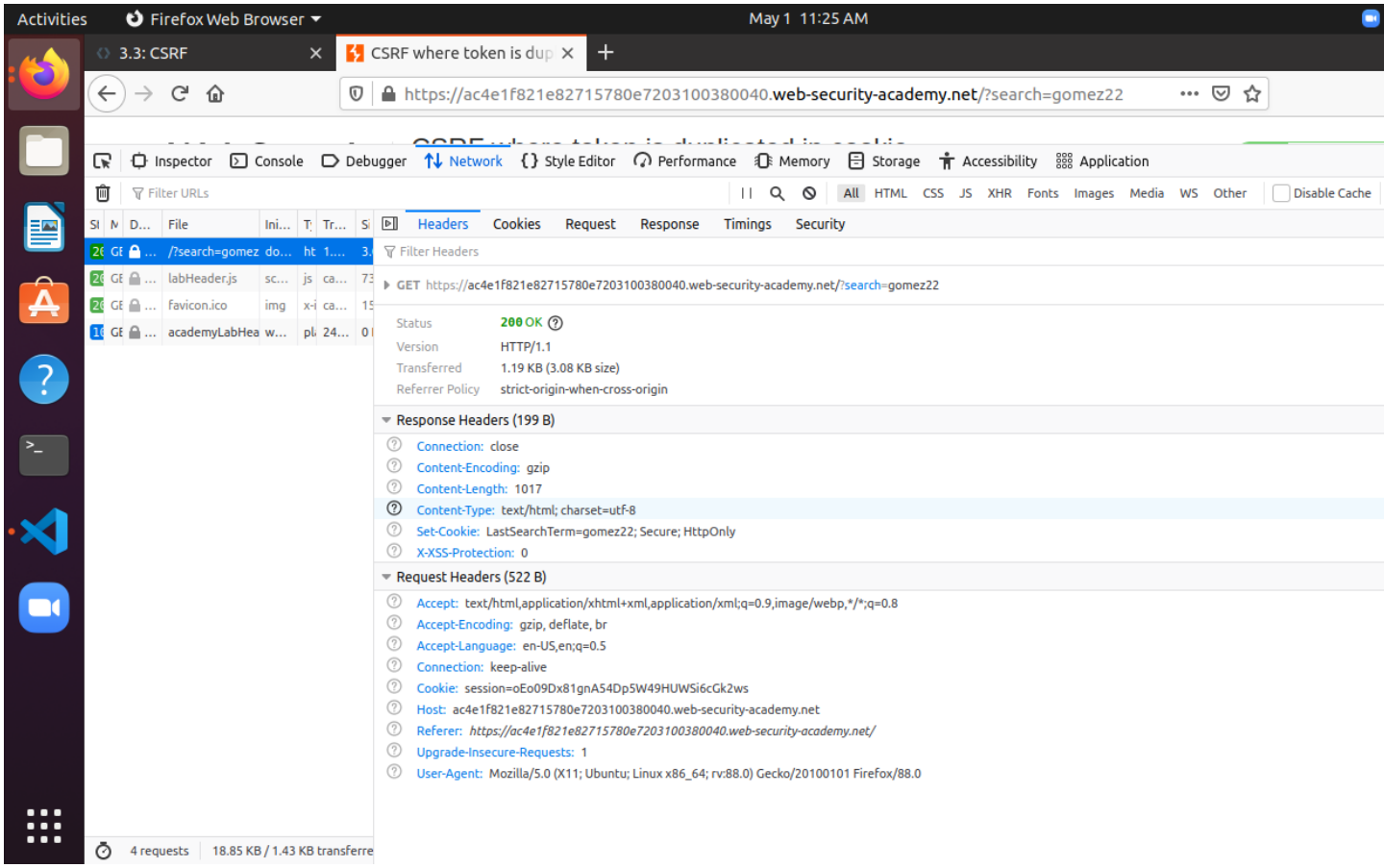


CSRF(4)

-This lab has 2 vulnerabilities. The first vulnerability is a header injection vulnerability where client input are reflected back into the HTTP headers of subsequent responses. The site developers didn't properly implement escaping newline characters which lets me alter headers or even set cookies. The second vulnerability is that the site validates a submission by checking if a cookie's csrf token matches the csrf token in the form being submitted. As an attacker, I can change the value of the form token and

also the cookie token's value to match thereby tricking the server into thinking my request is valid.

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



How many cookies are returned? What are their names?

-It appears as though two cookies were returned, "LastSearchTerm" and "session"

How have foo and bar been interpreted?

-It appears as though they have been interpreted as a header.

How many cookies have been returned?

-It appears as though three cookies have been returned. LastSearchTerm, session, and foo.

Explain the results. Give one reason why a developer would choose to implement CSRF protection in this manner

-I think the reason would be to attempt to tie CSRF tokens to unique users by comparing the CSRF against a user's cookie to see if they match in order to increase security. This would be more secure if the developers made the cookie unable to be tampered with but that is not the case in this lab.

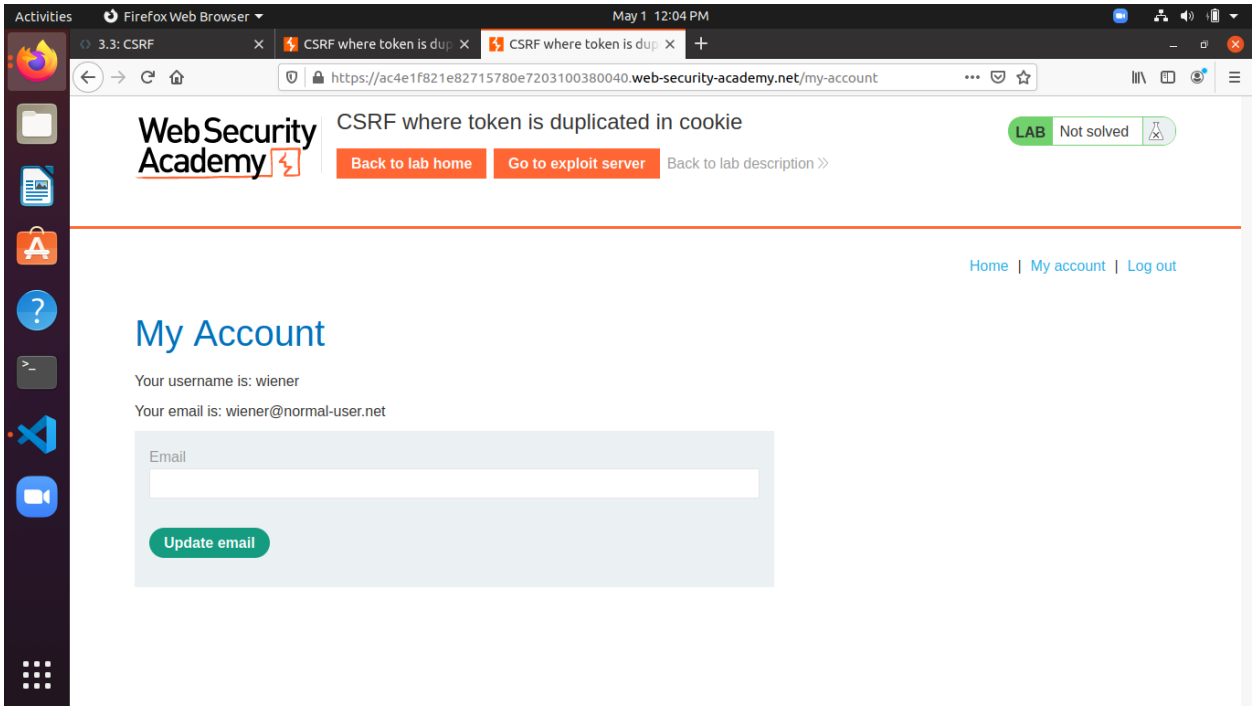
What status code is returned? What is the value of the csrf field in the HTML form that is given back as a response?

-The status code returned was 200 OK, and the value of the csrf field in the response was "gomez22", my Odin Id.

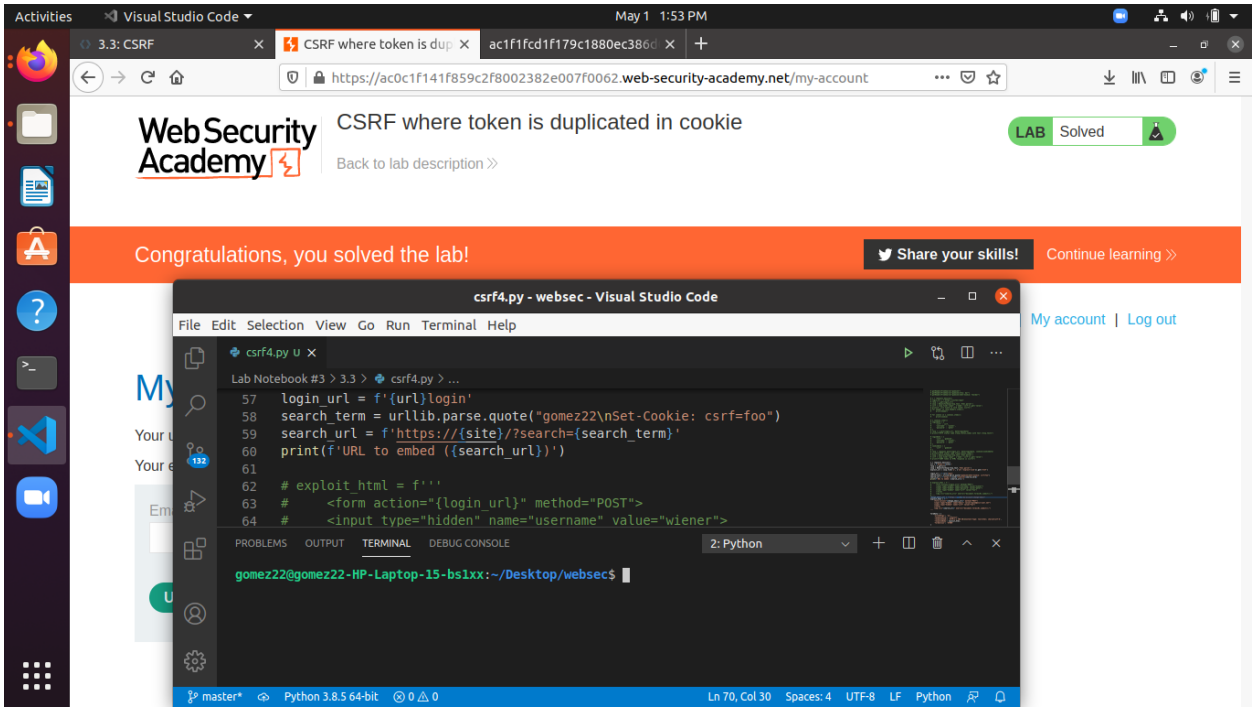
How might a developer use a keyed hash function on the server to prevent this request from succeeding without being forced to store each token?

-Upon a successful login, the server could generate and return a unique session Id for subsequent requests. This could authenticate further requests from the same user. This could be done to store unique session ids without the need to store CSRF tokens.

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



Csrf (5)

-The vulnerability of this lab is that authentication is done with the referer header. As an attacker I can use the fact that the developers do not properly escape new line characters in their headers so I can leverage that to set my own headers. This lets me set the referer header to be valid. To remediate this, the developers should properly escape new line characters as well as use some stronger form of authentication than the referer header.

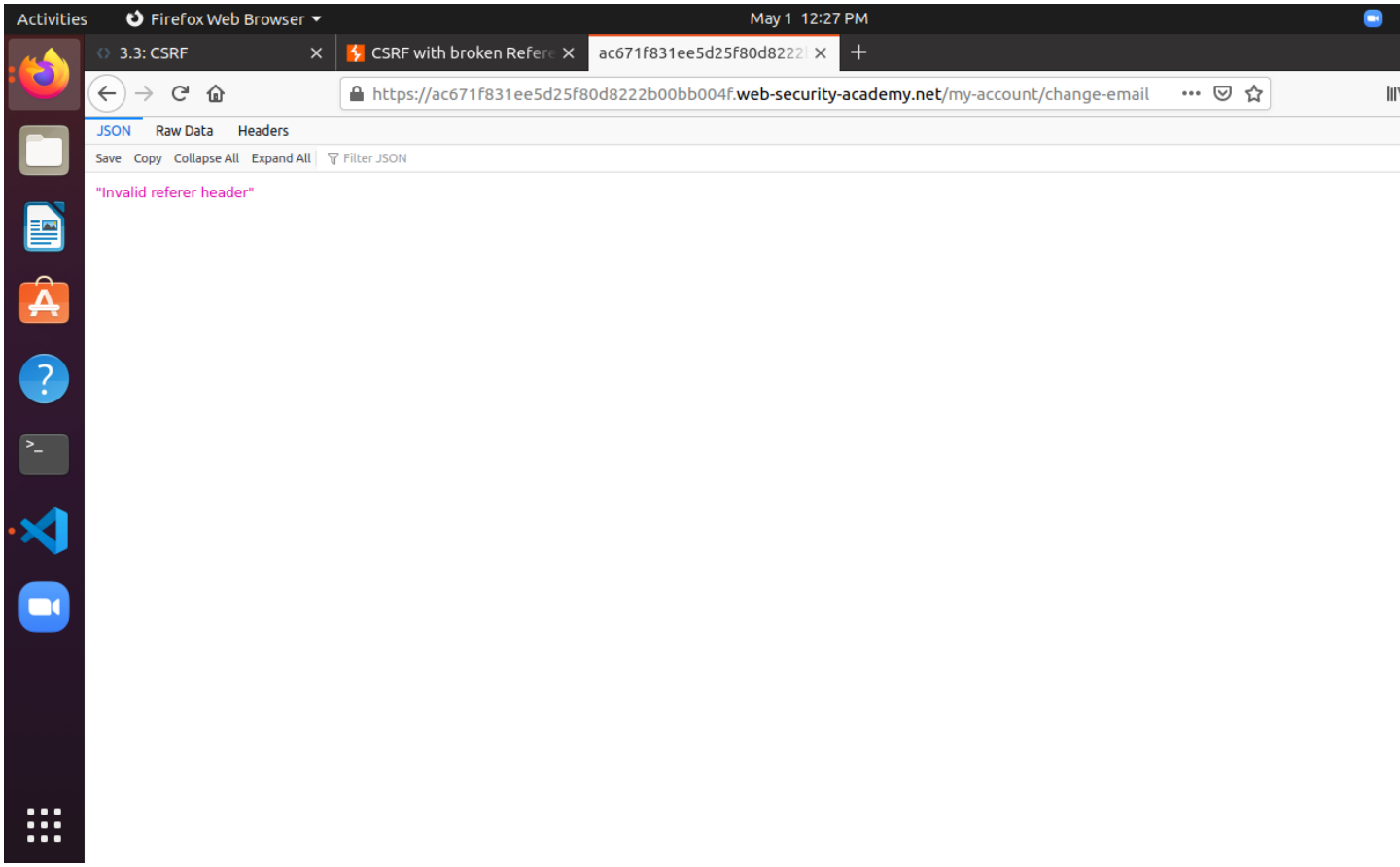
What status code and response text is returned?

-I received a status code 400 with the text “Invalid referer header”.

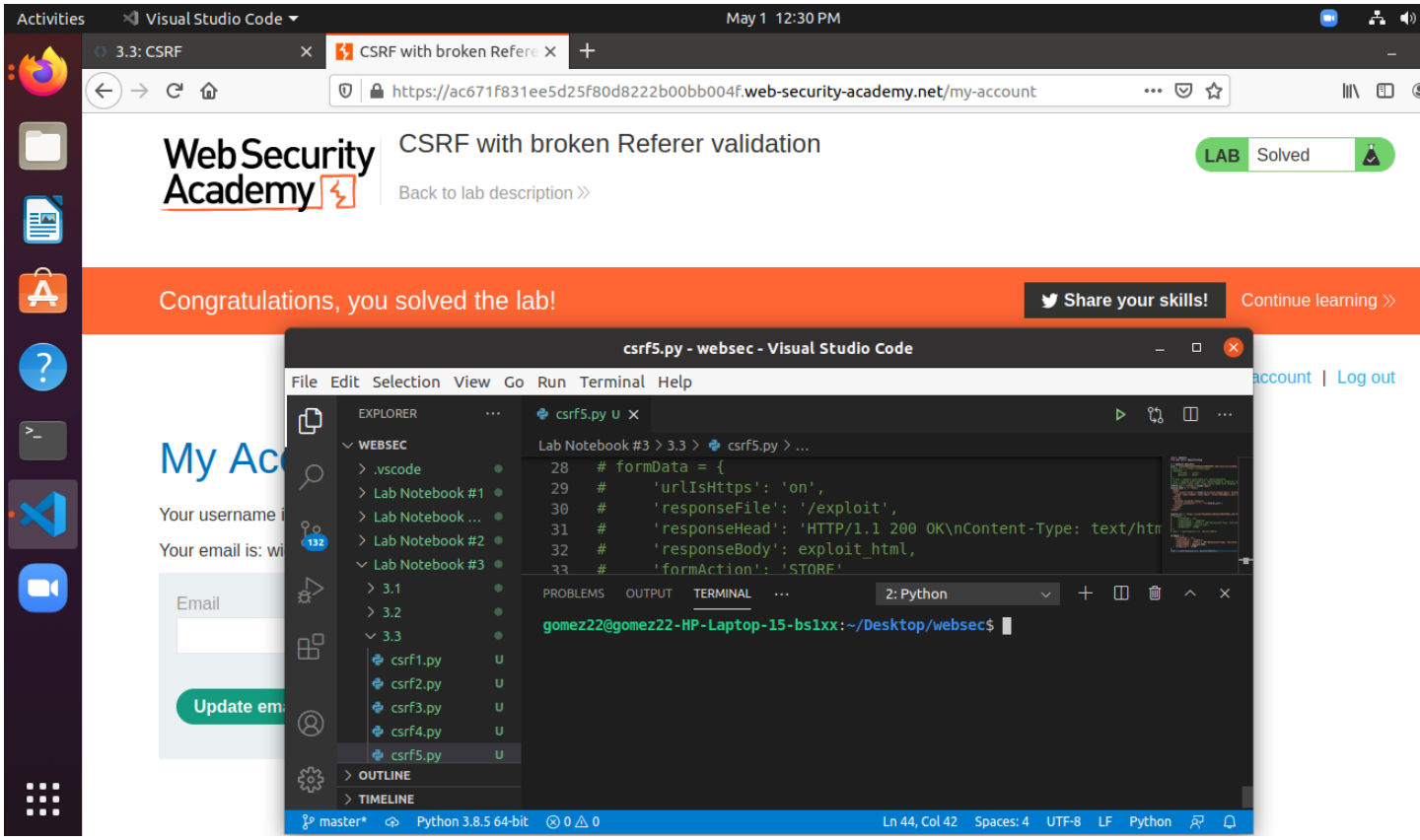
What status code is returned?

-I received a status code 200 OK.

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

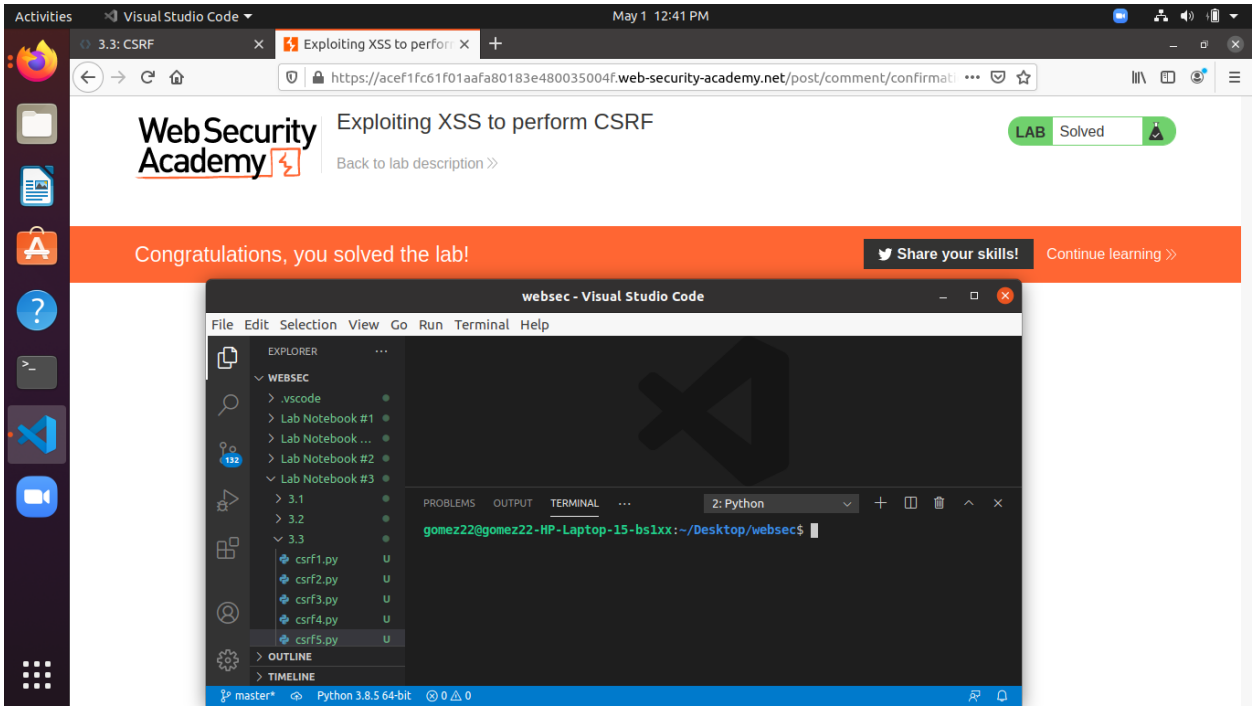


Take a screenshot showing completion of the level that includes your OdinId



Cross-site-scripting/exploiting

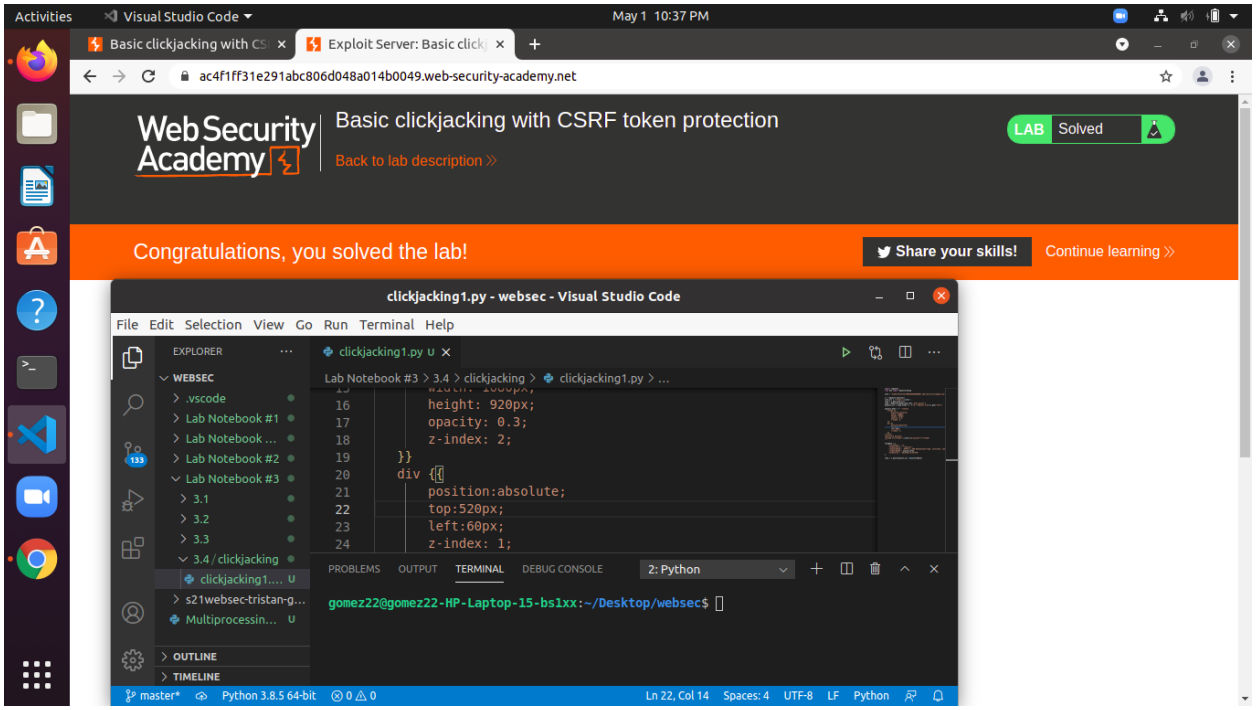
-This lab has a vulnerability where I can insert JavaScript into comments and have that code execute when a victim visits the infected page. The code is then run when the dom renders on the victim's browser and I can do many malicious things like steal their unique tokens. To remediate this, the site developers should properly sanitize inputs from clients.



3.4 Clickjacking

Clickjacking (1)

-This vulnerability has a victim visit a malicious site that has invisible to the eye elements embedded in the page which are covered by content which is meant to trick the victim into taking some action, but actually takes the action of the hidden content embedded in the page. To remediate this vulnerability, the developers should alter their CSP to prevent their site from being embedded/displayed in foreign sites.



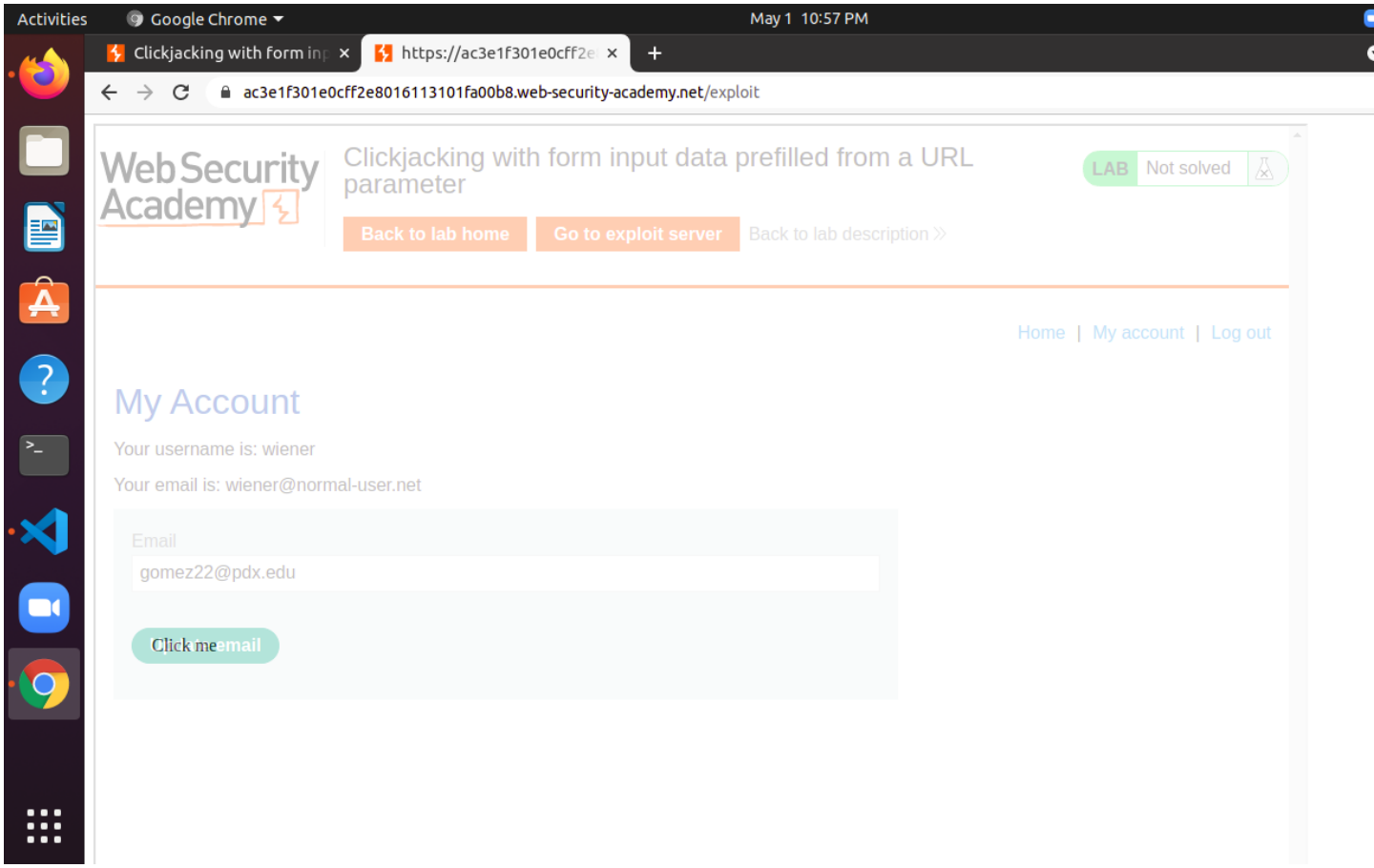
Clickjacking (2)

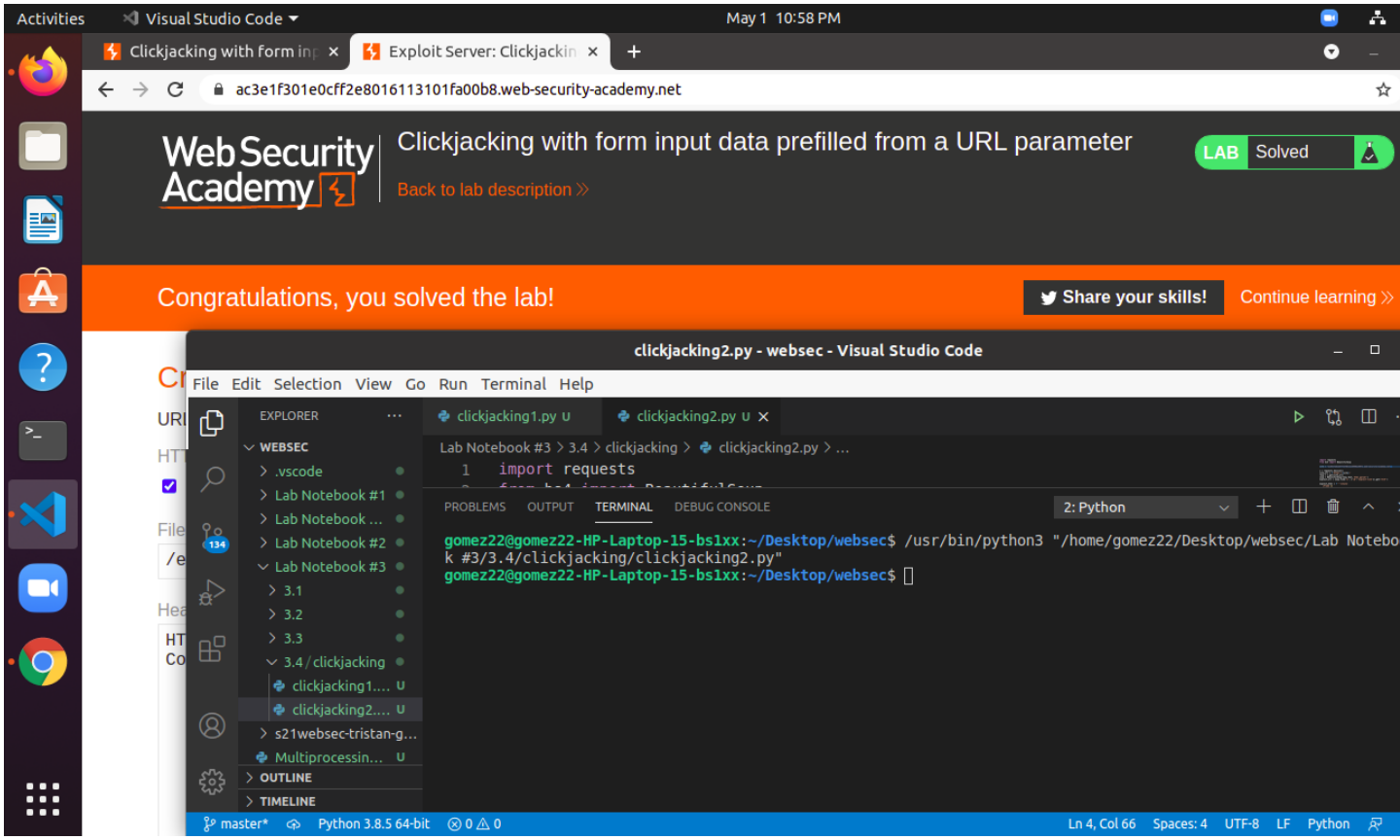
-This vulnerability has a victim visit a malicious site that has invisible to the eye elements embedded in the page which are covered by content which is meant to trick the victim into taking some action, but actually takes the action of the hidden content embedded in the page. To remediate this vulnerability, the developers should alter their CSP to prevent their site from being embedded/displayed in foreign sites.

In the form's HTML, how many fields are present? Which one has been prefilled with a value?

-There are two fields in the form. The “csrf” input has a prefilled value.

Take a screenshot showing that the form in the transparent <iframe> has been prefilled similar to below


Activities
Google Chrome
May 1 10:57 PM
Clickjacking with form inp
https://ac3e1f301e0cff2e
ac3e1f301e0cff2e8016113101fa00b8.web-security-academy.net/exploit
Web Security Academy
Clickjacking with form input data prefilled from a URL parameter
LAB
Not solved
Back to lab home
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
gomez22@pdx.edu
Click me email



Clickjacking (3)

-This vulnerability has a victim visit a malicious site that has invisible to the eye elements embedded in the page which are covered by content which is meant to trick the victim into taking some action, but actually takes the action of the hidden content embedded in the page. To remediate this vulnerability, the developers should alter their CSP to prevent their site from being embedded/displayed in foreign sites.

What is the name of the HTML element that has been updated after the form has been submitted?

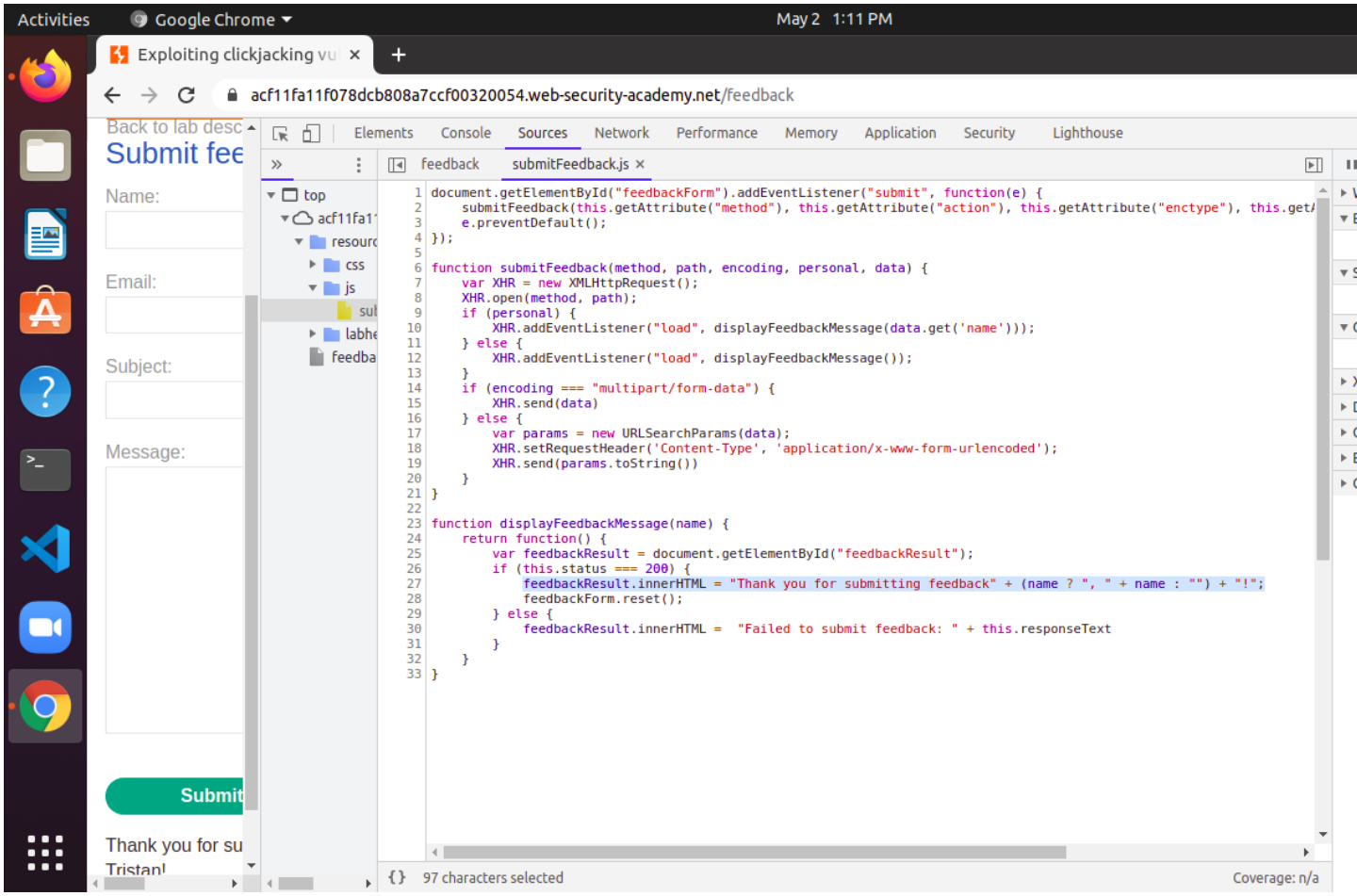
-The Id of the HTML element that was updated upon form submission is “feedbackResult”.

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

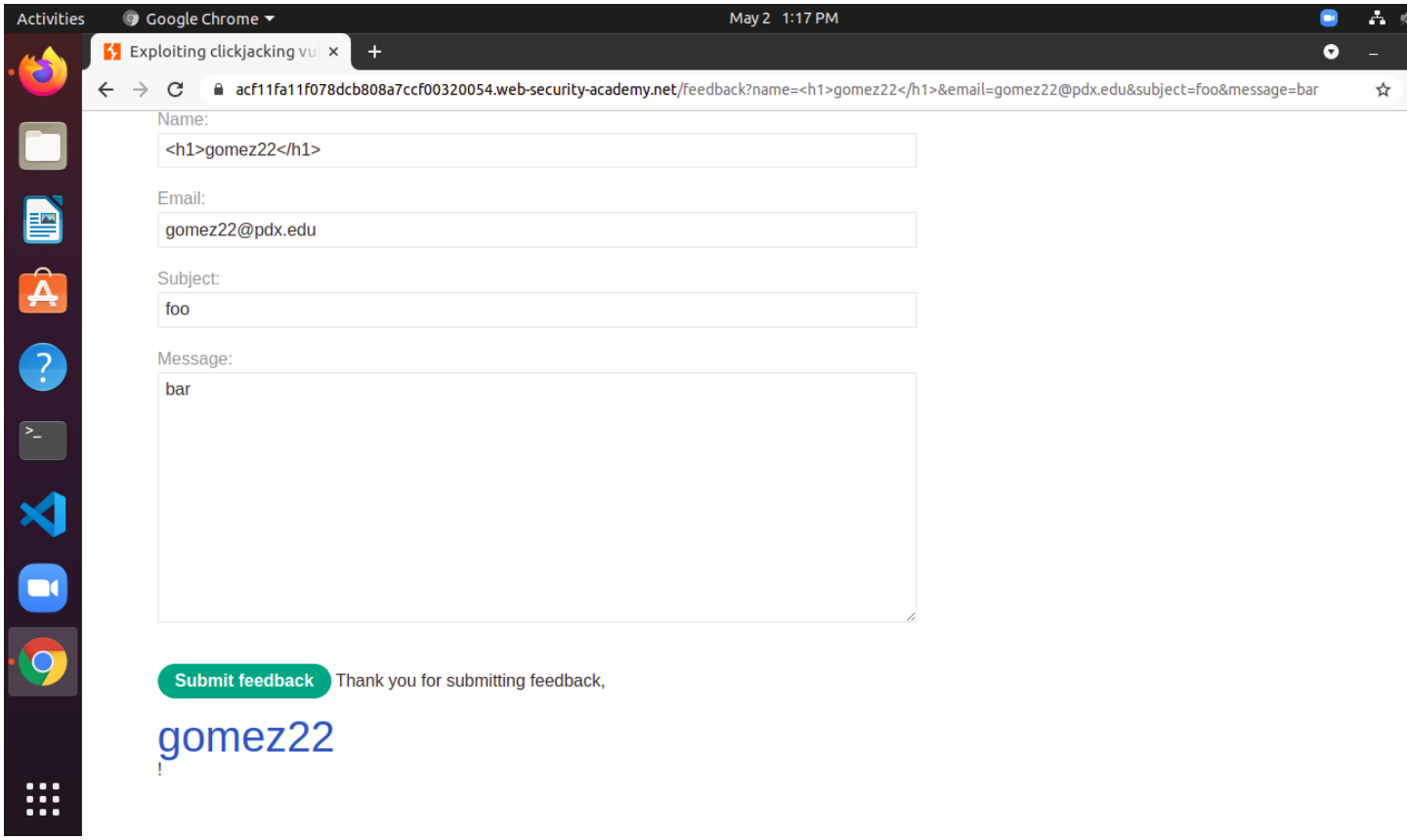
-The event listener function is called “displayFeedbackMessage”.

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

-The vulnerable line of code in the image below is the highlighted line. It is subject to an XSS attack because it grabs whatever is in the “name” field and will echo/reflect it back onto the page without doing any checks or sanitizing the input. This will allow me to insert JavaScript code into the page.



Take a screenshot that includes the URL demonstrating successful injection



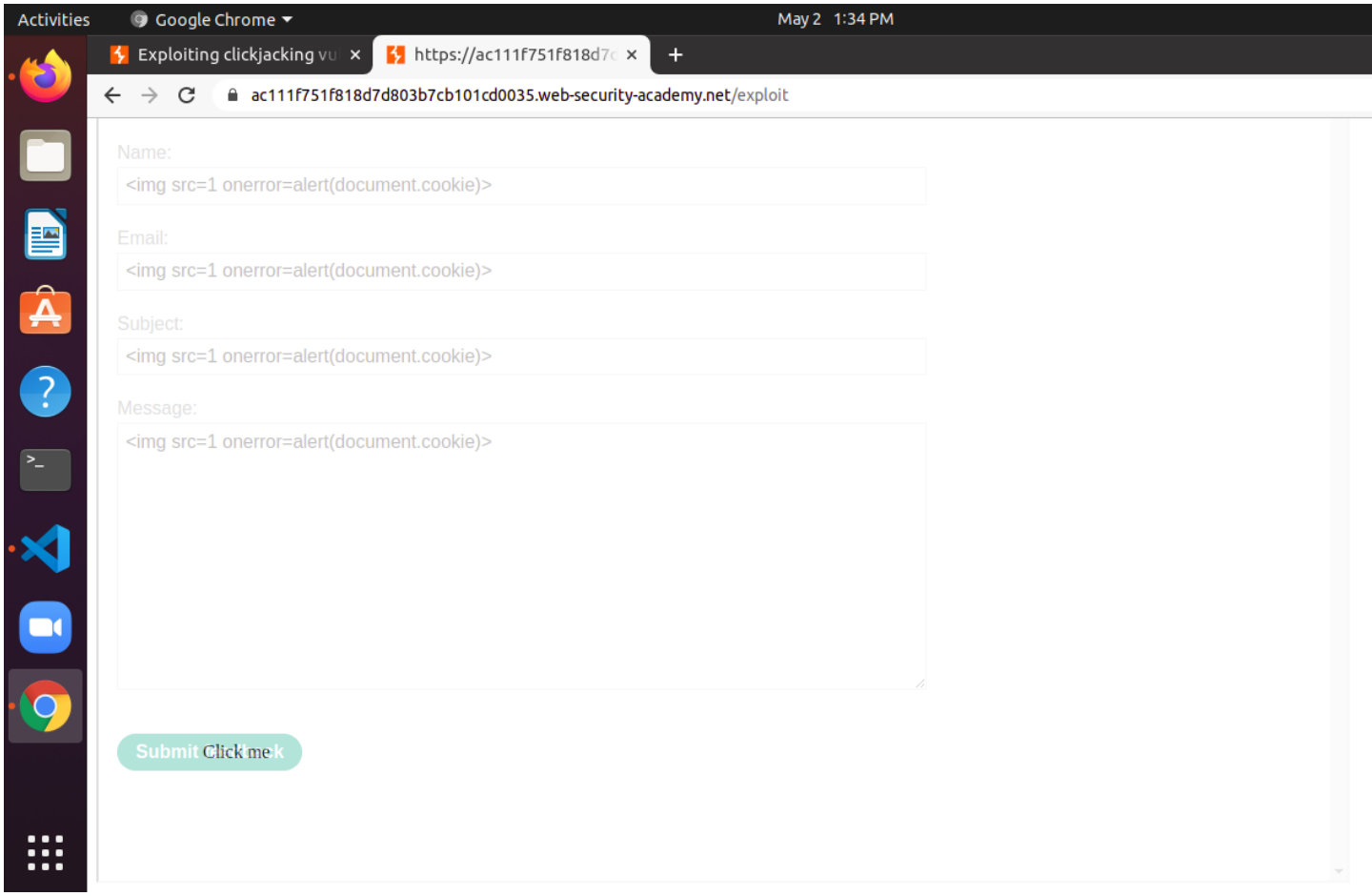
Did the payload get successfully returned into the page?

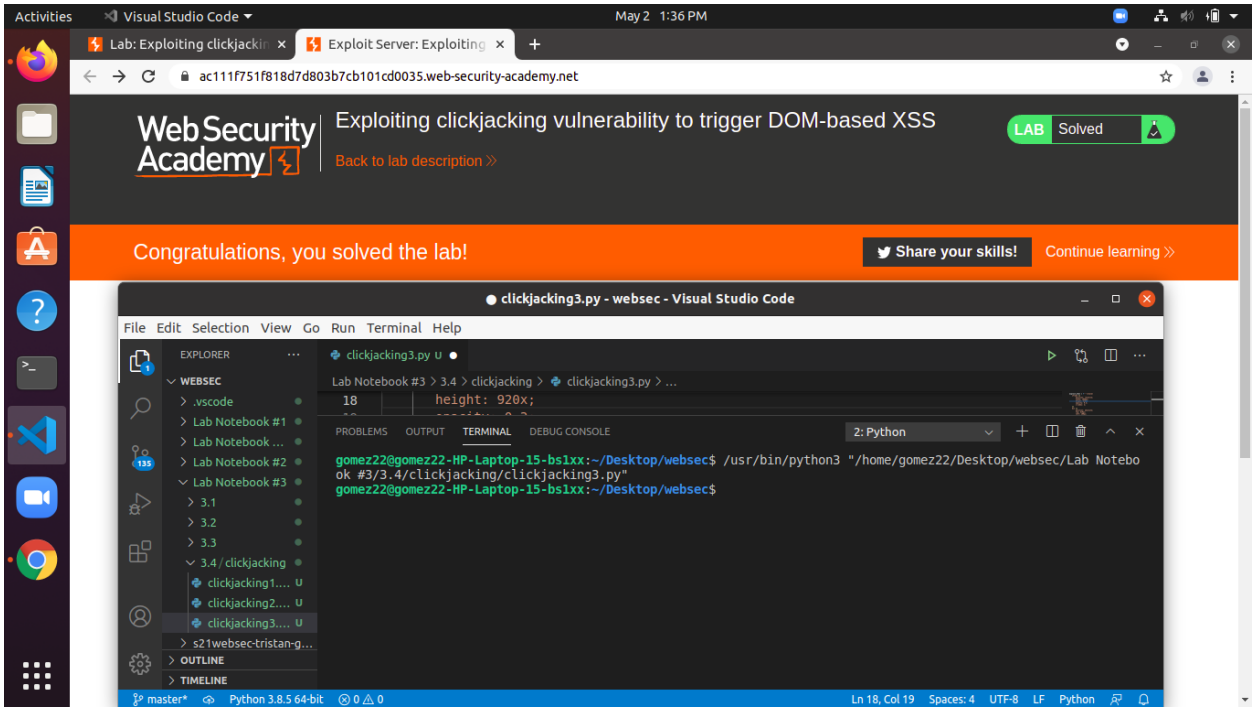
-No it didn’t get echoed back onto the page.

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

-The payload did not execute. The developers may sanitize tag inputs, or possibly the CSP is set to not allow scripts that are inserted into an element.

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

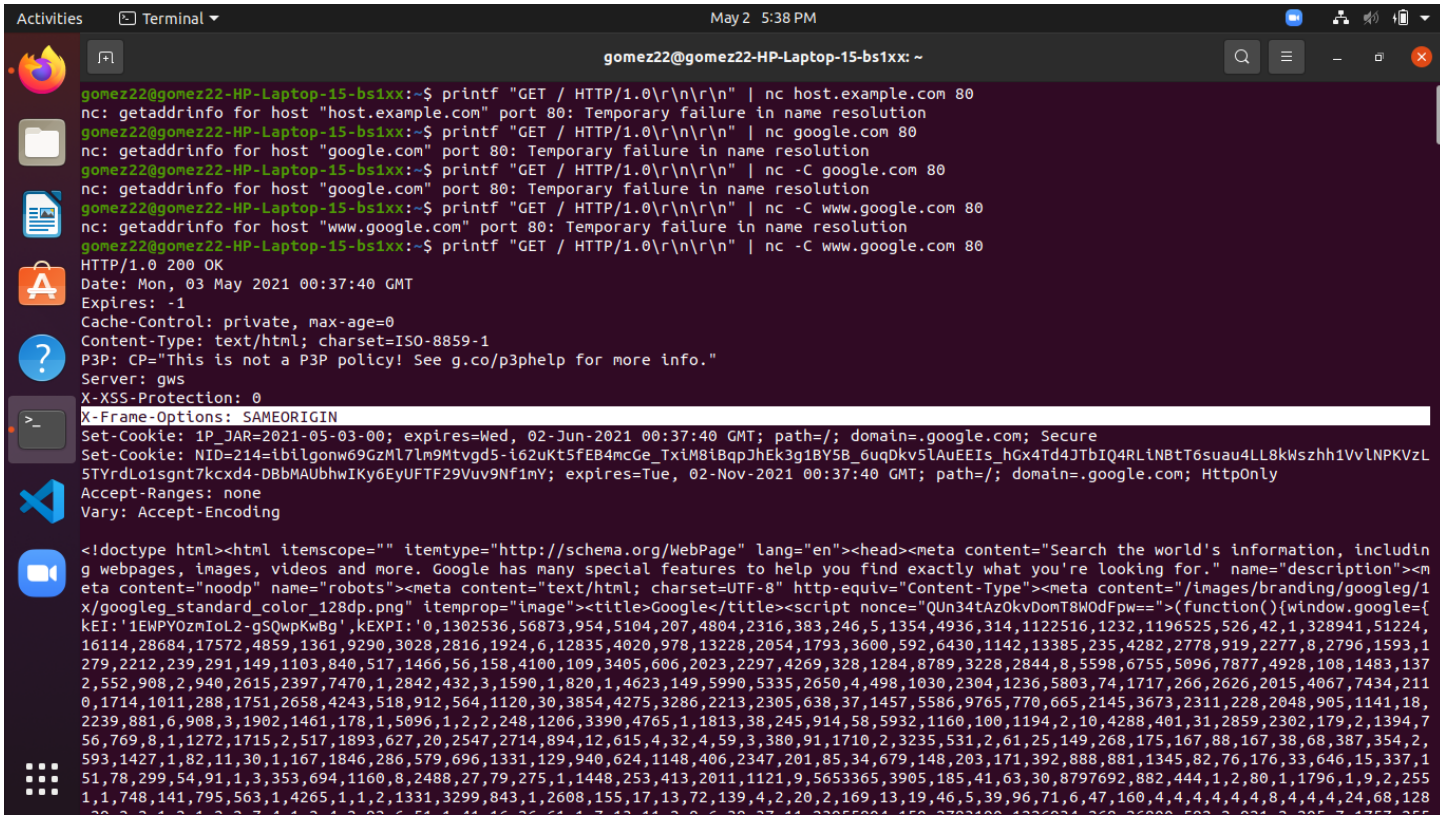

Activities
Google Chrome
May 2 1:34 PM
Exploiting clickjacking vu
https://ac111f751f818d7
ac111f751f818d7d803b7cb101cd0035.web-security-academy.net/exploit
Name:
<img src=1 onerror=alert(document.cookie)>
Email:
<img src=1 onerror=alert(document.cookie)>
Subject:
<img src=1 onerror=alert(document.cookie)>
Message:
<img src=1 onerror=alert(document.cookie)>
Submit Click me:k


Activities
Visual Studio Code
May 2 1:36 PM
Lab: Exploiting clickjackin
Exploit Server: Exploiting
ac111f751f818d7d803b7cb101cd0035.web-security-academy.net
Web Security Academy
Exploiting clickjacking vulnerability to trigger DOM-based XSS
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
clickjacking3.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEBSEC
.vscode
Lab Notebook #1
Lab Notebook ...
Lab Notebook #2
Lab Notebook #3
3.1
3.2
3.3
3.4 / clickjacking
clickjacking1....
clickjacking2....
clickjacking3....
s21websec-tristan-g...
OUTLINE
TIMELINE
clickjacking3.py U
Lab Notebook #3 > 3.4 > clickjacking > clickjacking3.py > ...
18 height: 920x;
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
2: Python
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/gomez22/Desktop/websec/Lab Notebook #3/3.4/clickjacking/clickjacking3.py"
gomez22@gomez22-HP-Laptop-15-bs1xx:~/Desktop/websec$
master*
Python 3.8.5 64-bit
Ln 18, Col 19
Spaces: 4
UTF-8
LF
Python

Prevention (X-frame options)

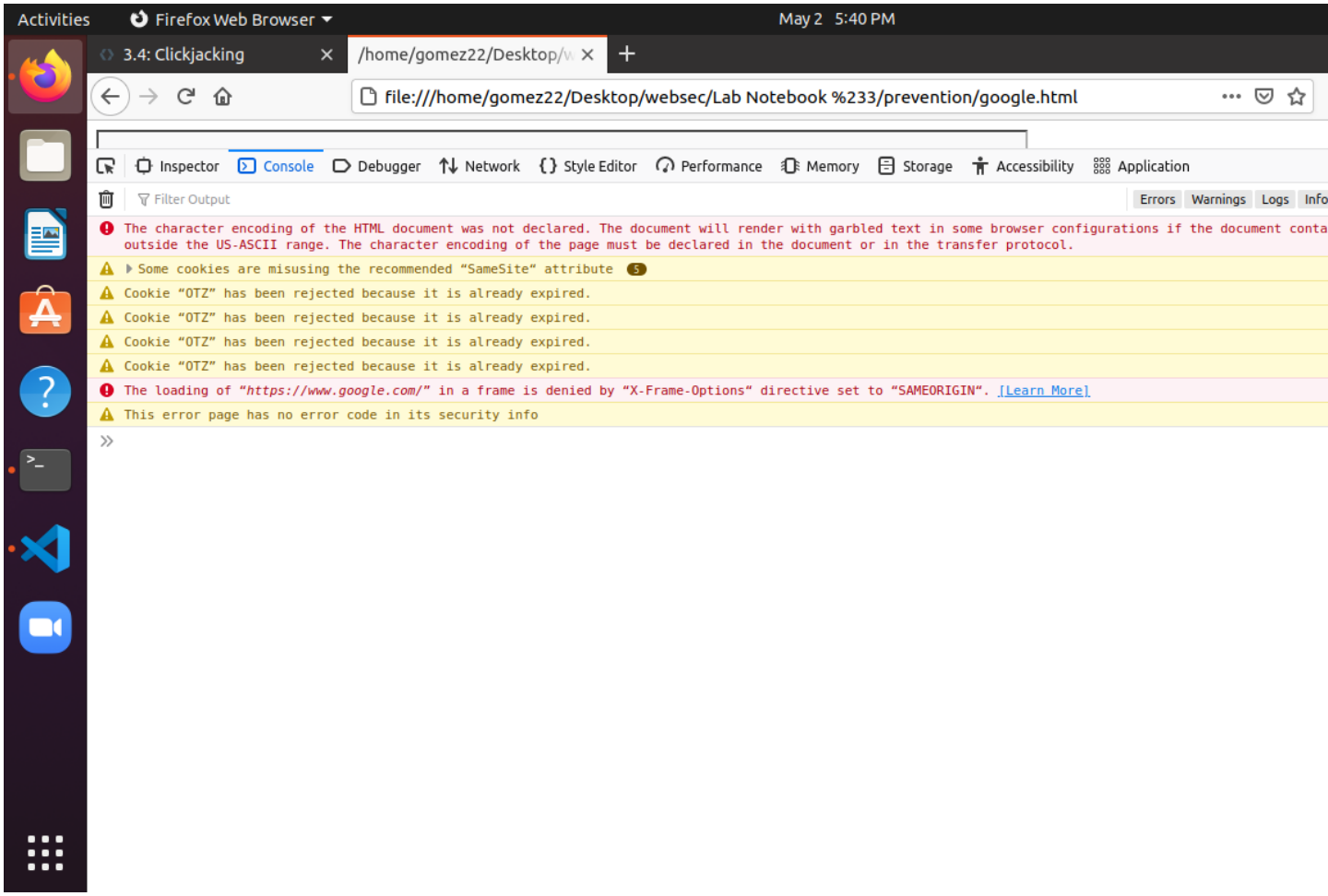
Google

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



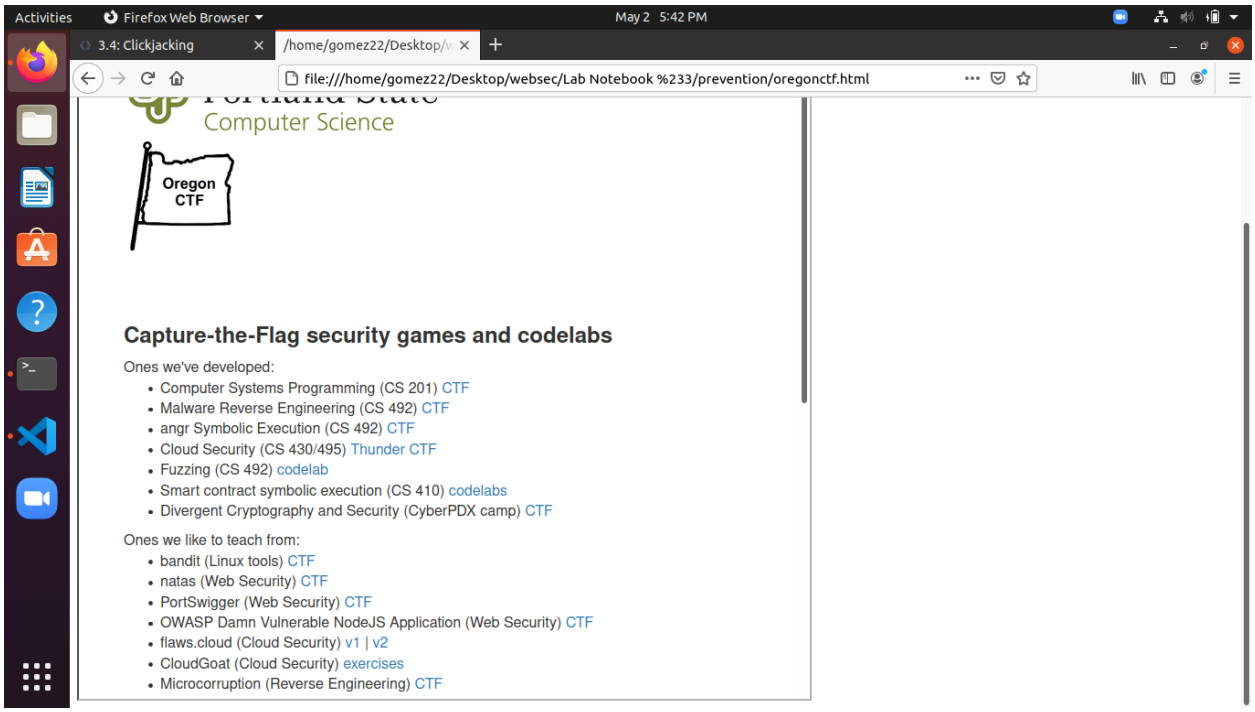
Load the file in a browser and see if the page loads. If not, show the error in the console and explain the results.

-Google set their X-Frame-Options to SAMEORIGIN, meaning that I cannot embed their page into a tag because I don't share the page's origin or subdomain.

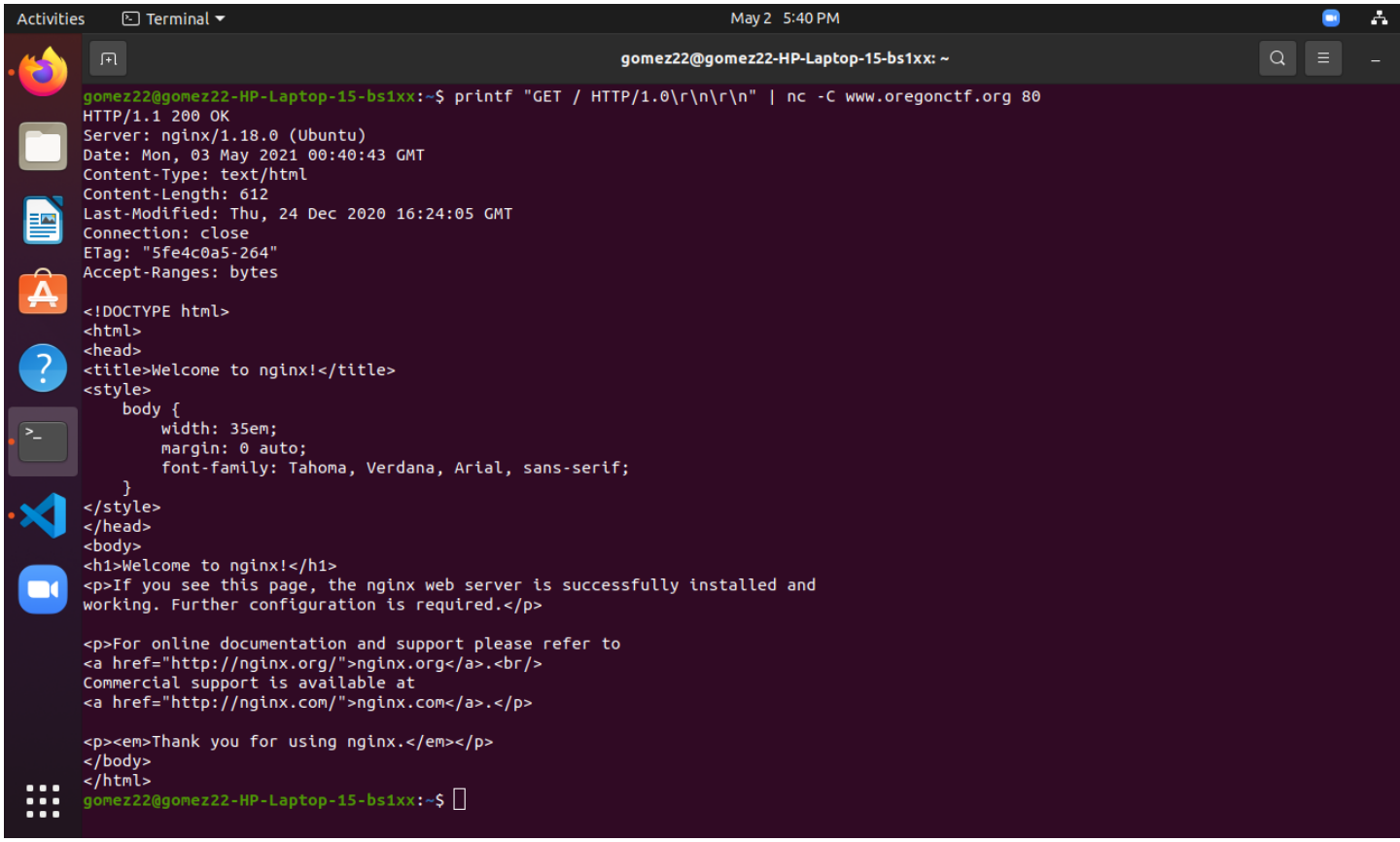

Activities
Firefox Web Browser
May 2 5:40 PM
3.4: Clickjacking
/home/gomez22/Desktop/
file:///home/gomez22/Desktop/websec/Lab Notebook %233/prevention/google.html
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter Output
Errors
Warnings
Logs
Info
The character encoding of the HTML document was not declared. The document will render with garbled text in some browser configurations if the document conta
outside the US-ASCII range. The character encoding of the page must be declared in the document or in the transfer protocol.
Some cookies are misusing the recommended "SameSite" attribute
5
Cookie "OTZ" has been rejected because it is already expired.
Cookie "OTZ" has been rejected because it is already expired.
Cookie "OTZ" has been rejected because it is already expired.
Cookie "OTZ" has been rejected because it is already expired.
The loading of "https://www.google.com/" in a frame is denied by "X-Frame-Options" directive set to "SAMEORIGIN". [Learn More]
This error page has no error code in its security info

Oregon CTF

Show a screenshot of what OregonCTF passes back and whether there is a X-Frame-Options: header.



-There was no X-Frame-Options header.



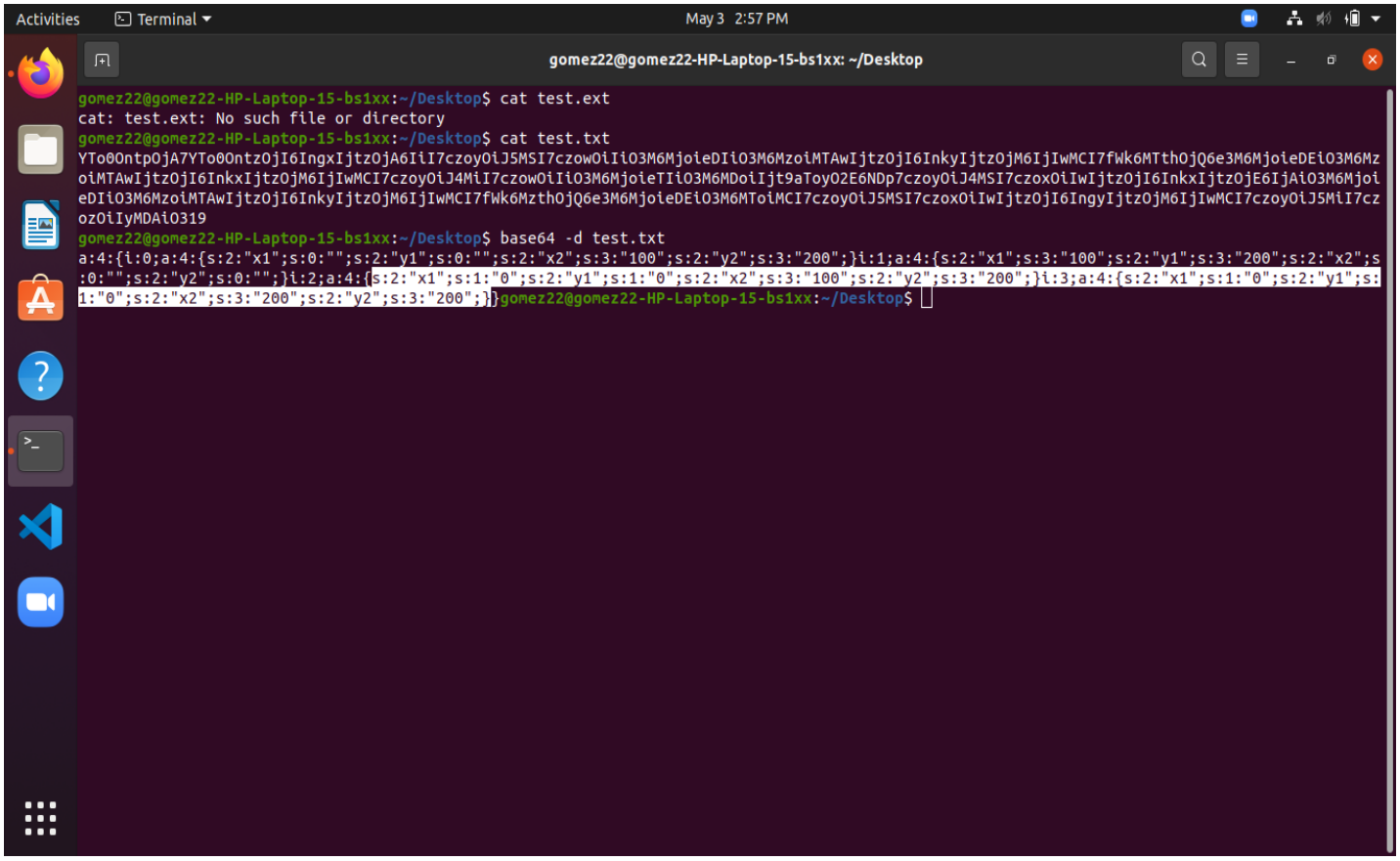
-I am allowed to embed the oregonctf content. There is no error.

3.5 Insecure Deserialization (PHP)

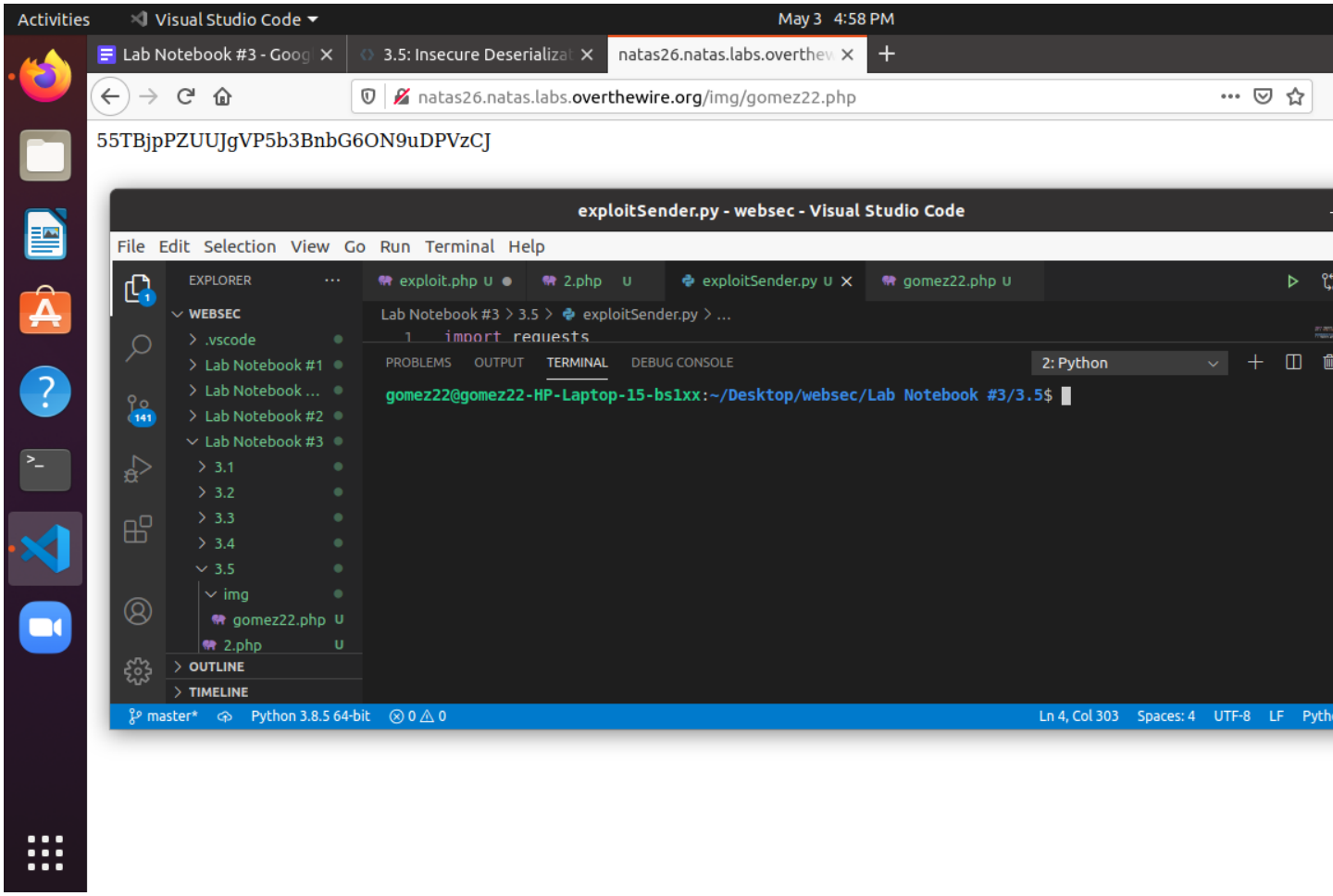
How does the state for the drawing get updated?

-It looks as though the site is using the function “drawFromUserData” to pull the coordinate data from the x1,x2,y1, and y2 input elements on the page. Then it pulls data from the “drawing” cookie, base64 decodes the cookie’s value, then unserializes it to pull “saved”/previously entered coordinates.

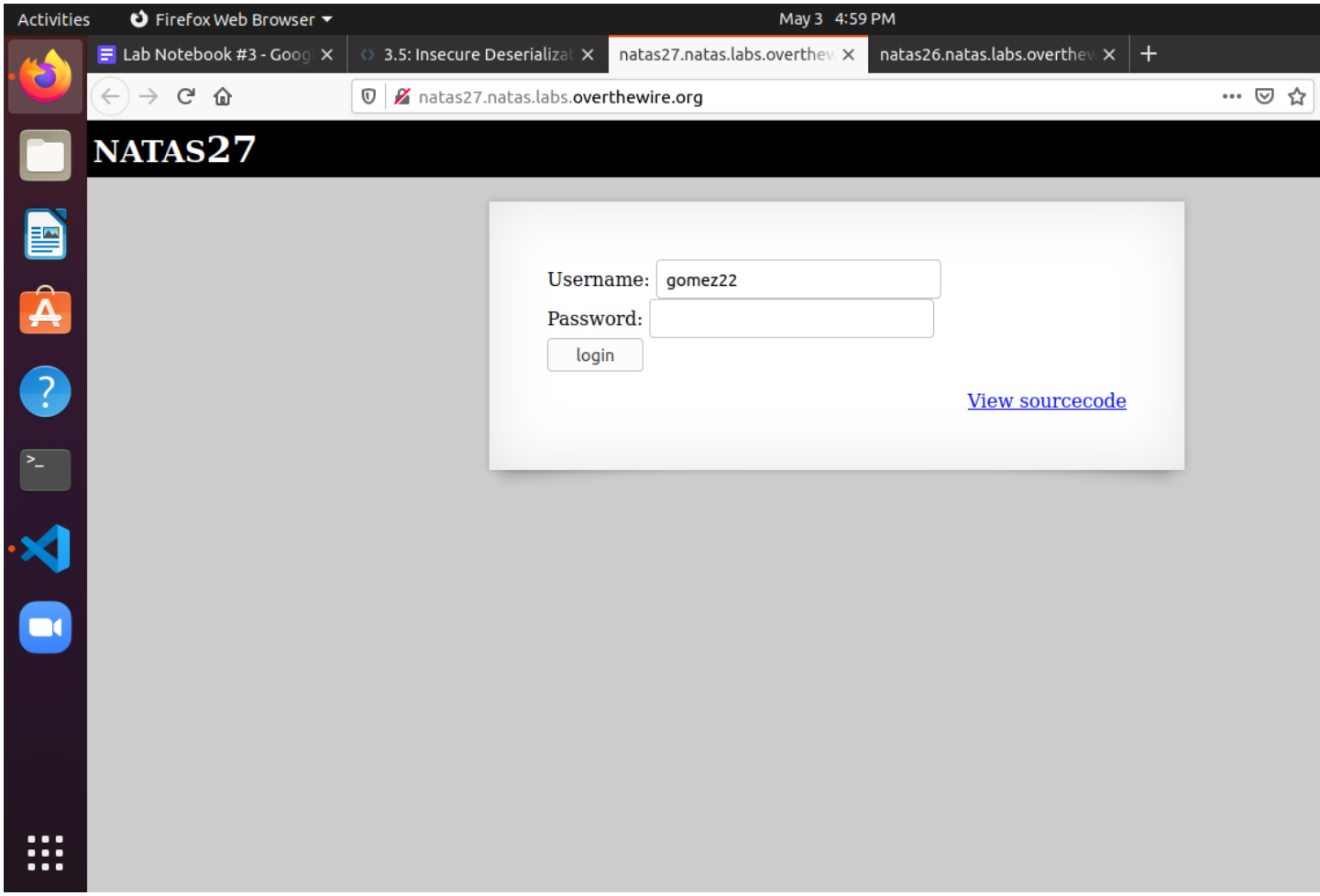
Run a base64 decode operation on the file and show the output (e.g. base64 -d <filename>). Highlight the coordinates you entered previously via the UI in the decoded output

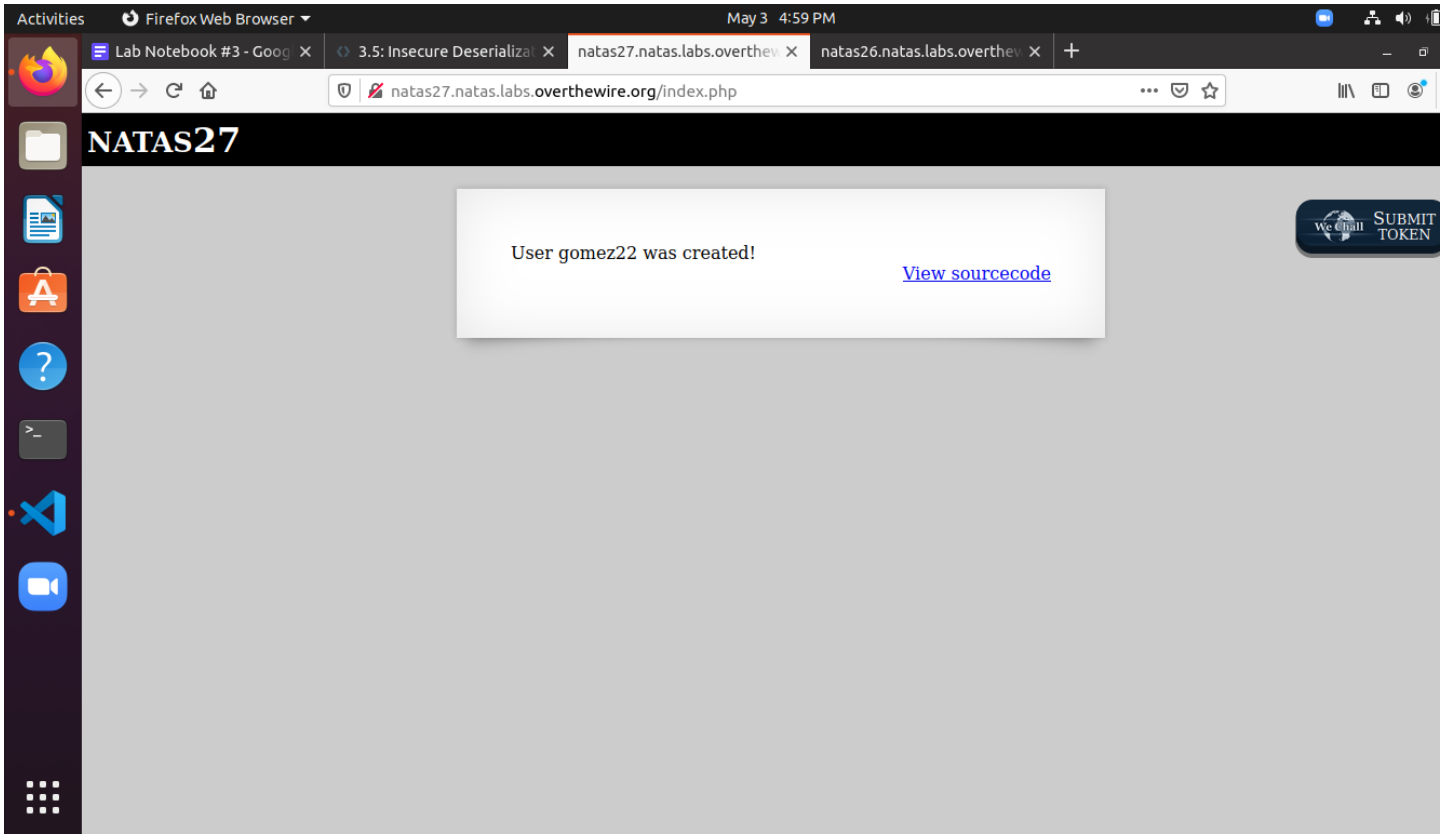


Access the file that you created via the malicious Logger class from your web browser to reveal the password to natas27. Take a screenshot of it with your OdinID to include in your lab notebook.



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.


Activities
Firefox Web Browser
May 3 4:59 PM
Lab Notebook #3 - Goog
3.5: Insecure Deserializat
natas27.natas.labs.overthev
natas26.natas.labs.overthev
natas27.natas.labs.overthewire.org
NATAS27
Username: gomez22
Password:
login
View sourcecode



3.6 Insecure deserialization (JavaScript)

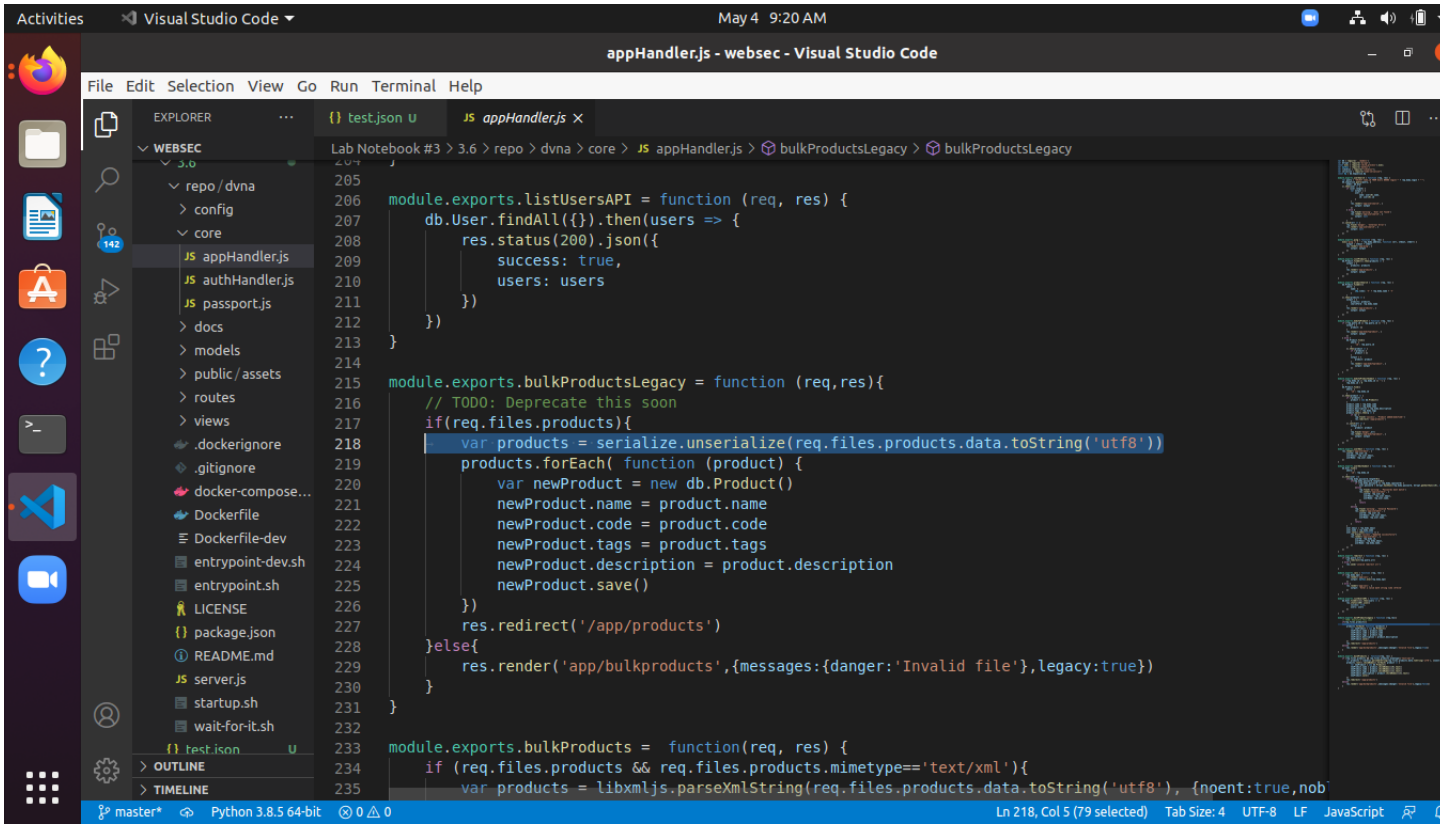
Within this source tree, find the file and the line number that contains the vulnerable deserialization. Address the items below and include them in your lab notebook

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

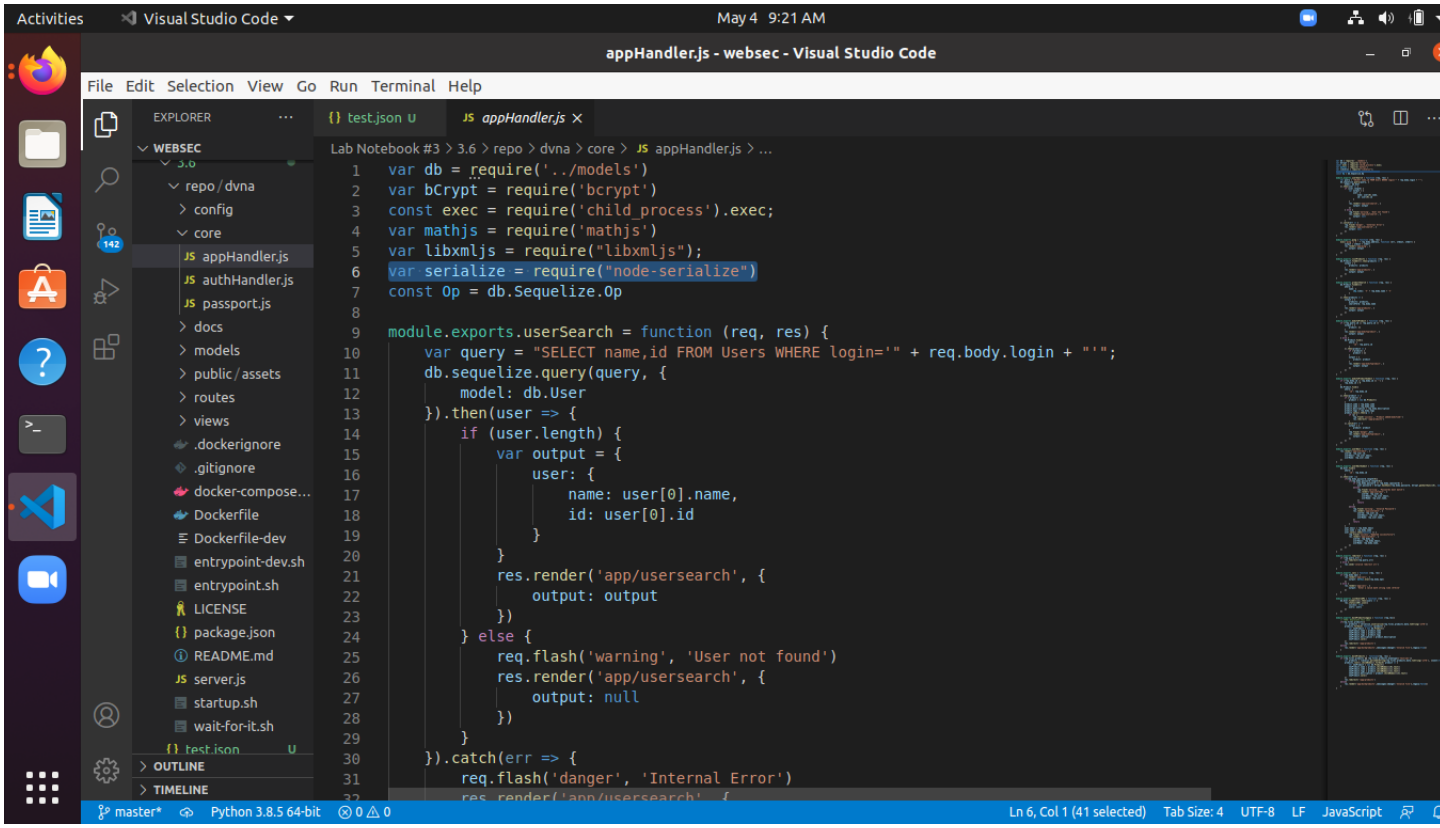
-The file name is “appHandler.js”.

Show the line of source code in the application that performs it.

-The vulnerability is highlighted in the screenshot below on line 218.



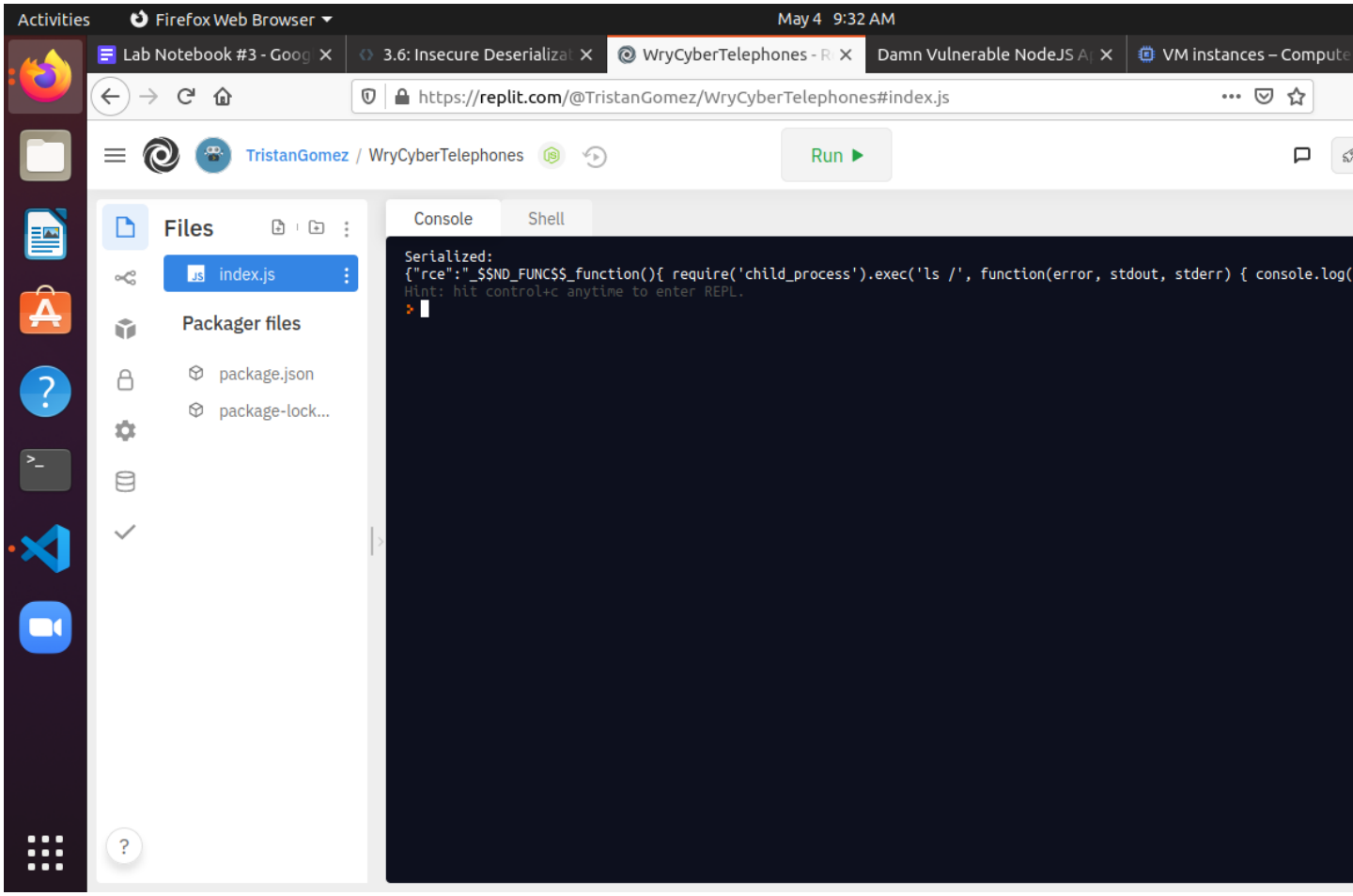
Show the "require" call at the top of the file that includes the name of the package being used to perform the deserialization.



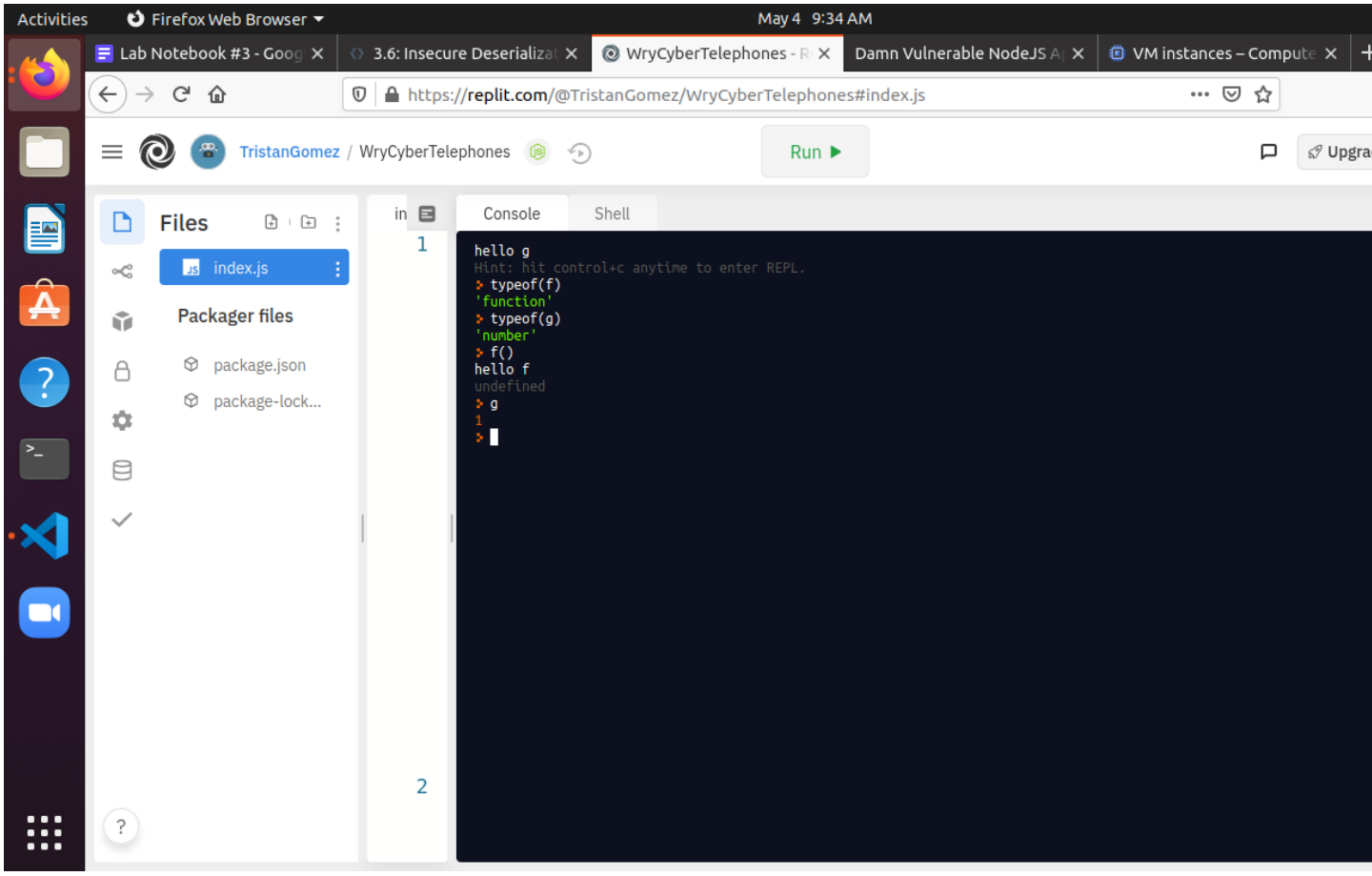
What is the name of the special function that NodeJS uses to execute a function that is included in a serialized object?

-The type is an “Immediately Invoked Function Expression (IIFE)”, but the name is “_$$ND_FUNC$$_function()”.

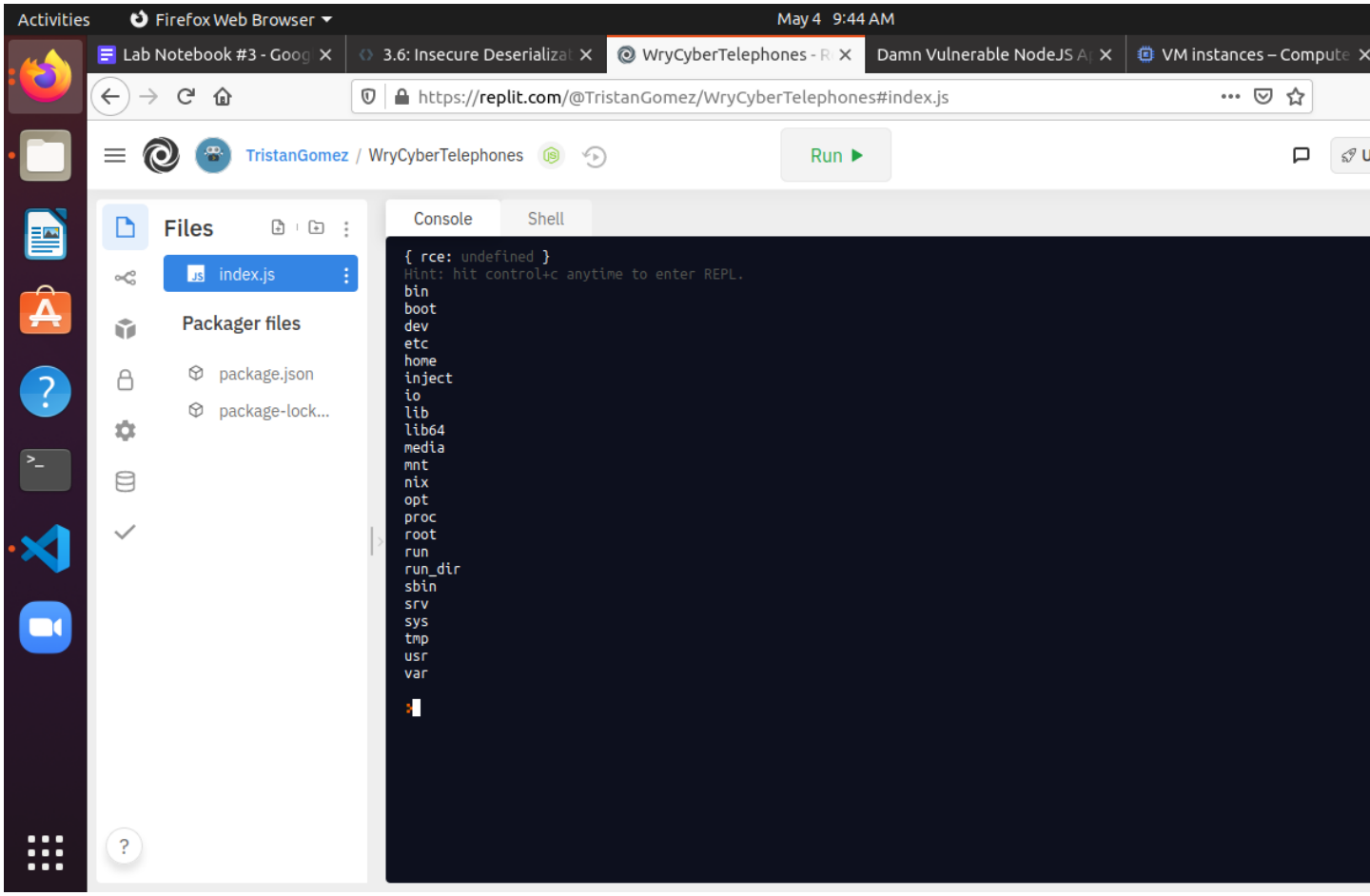
Show the output and note the specially named function that is used to invoke the exec call.



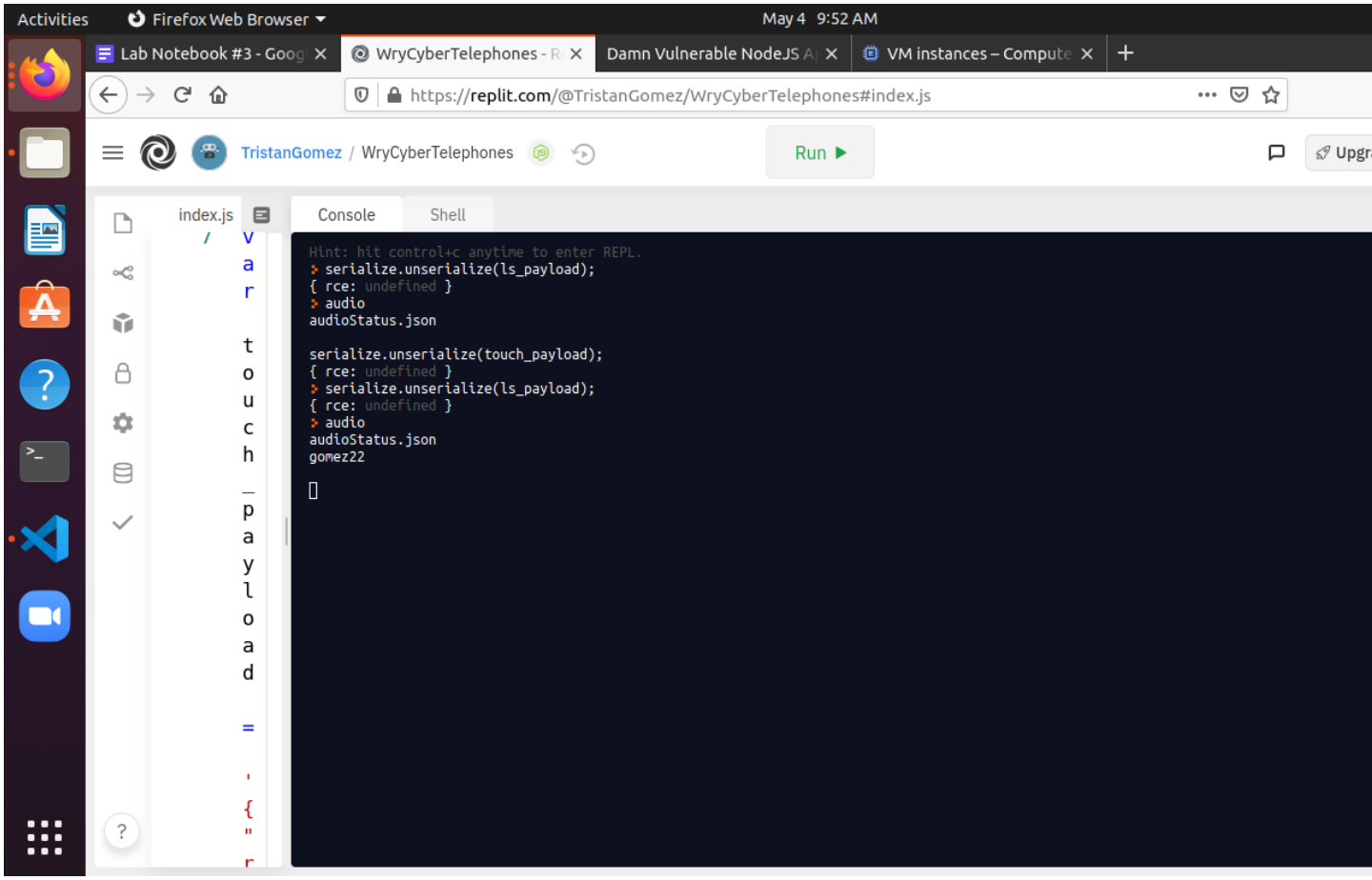
Take a screenshot of the output to include in your lab notebook.



Show the output of its execution in your lab notebook.



Screenshot the output in the console, ensuring it is your OdinID rather than mine.


Activities
Firefox Web Browser
May 4 9:52 AM
Lab Notebook #3 - Goog
WryCyberTelephones - R
Damn Vulnerable NodeJS A
VM instances – Compute
https://replit.com/@TristanGomez/WryCyberTelephones#index.js
TristanGomez / WryCyberTelephones
Run
Upgr
index.js
Console
Shell
var touch_payload = '{ "r
Hint: hit control+c anytime to enter REPL.
> serialize.unserialize(ls_payload);
{ rce: undefined }
> audio
audioStatus.json
serialize.unserialize(touch_payload);
{ rce: undefined }
> serialize.unserialize(ls_payload);
{ rce: undefined }
> audio
audioStatus.json
gomez22

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