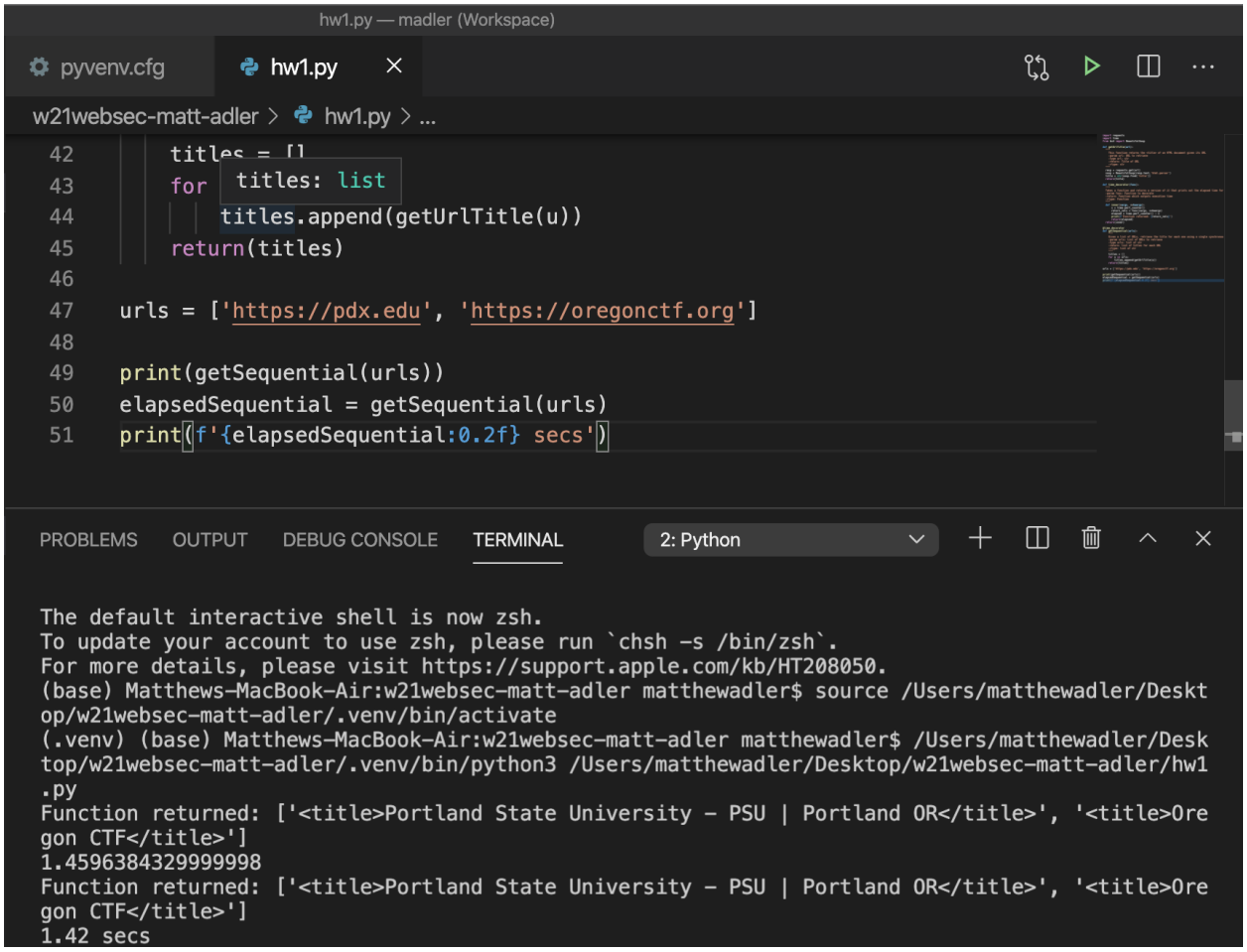
1.2 Web Programming

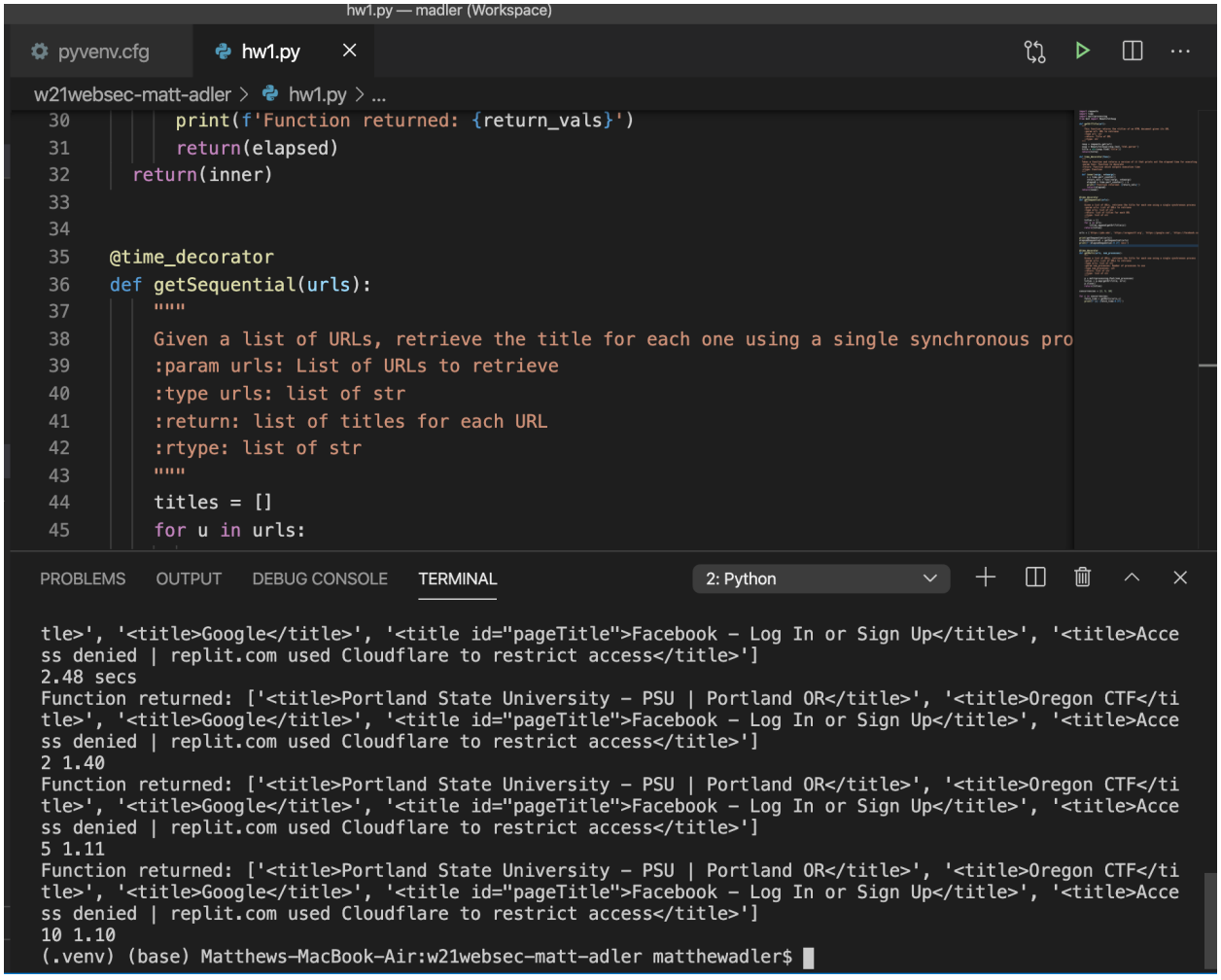
5. Run sequential program

- Take a screenshot of the output of your program's execution that includes your OdinId for your lab notebook.

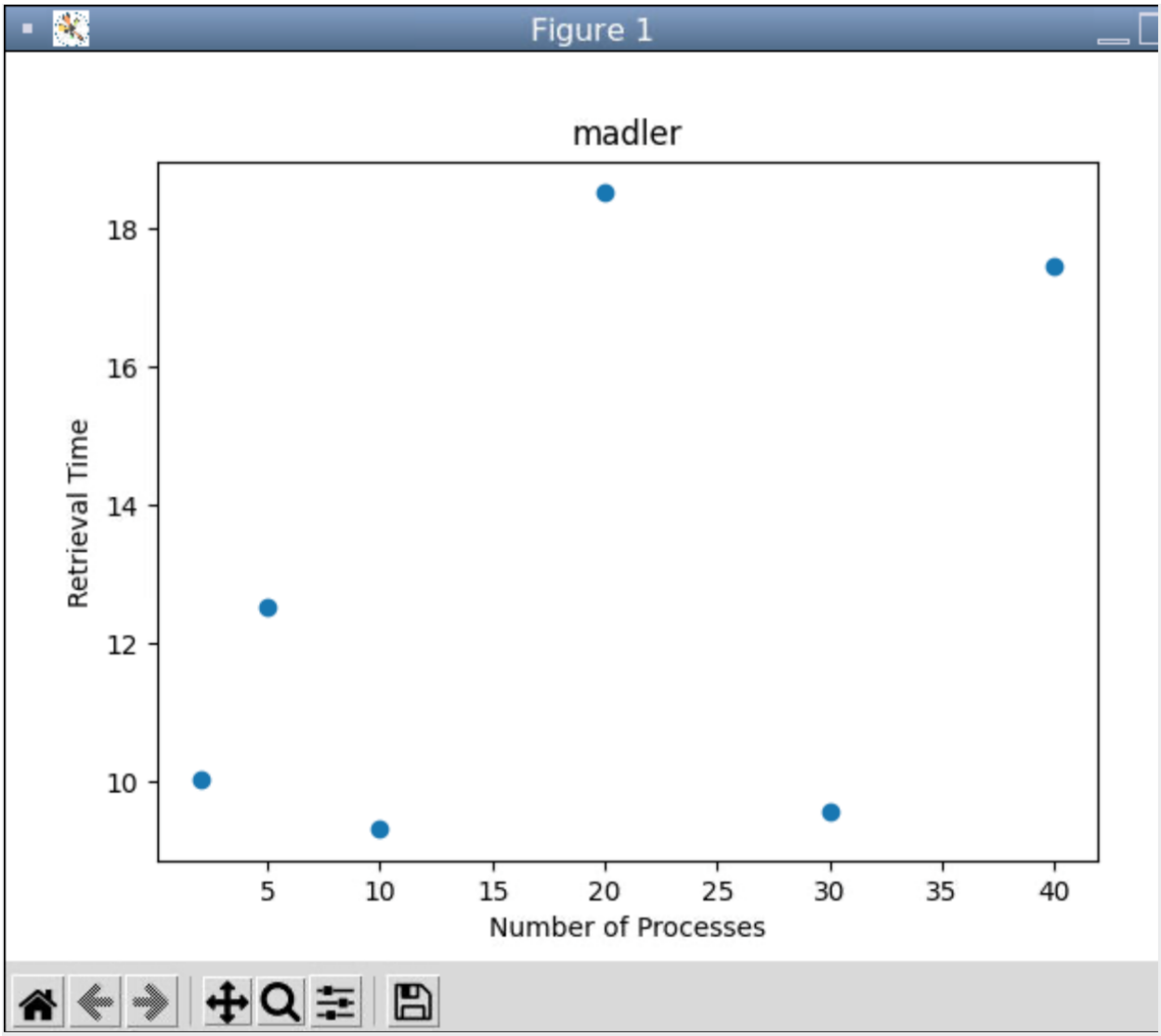


7. Run multiprocessing program

- Take a screenshot of the output of your program's execution that includes your OdinId for your lab notebook.



9. Run matplotlib program on repl.it

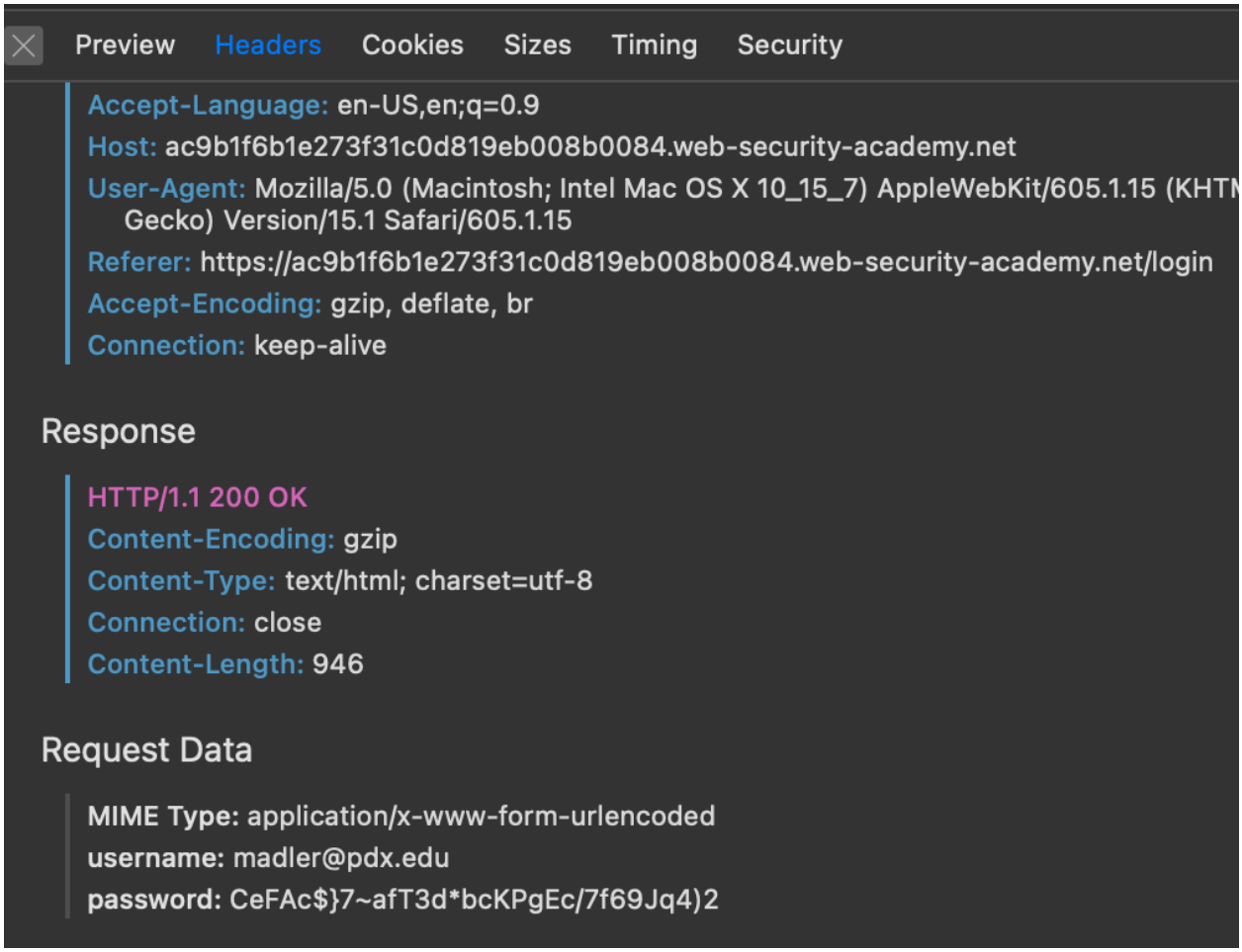


12. Run asyncio program on repl.it

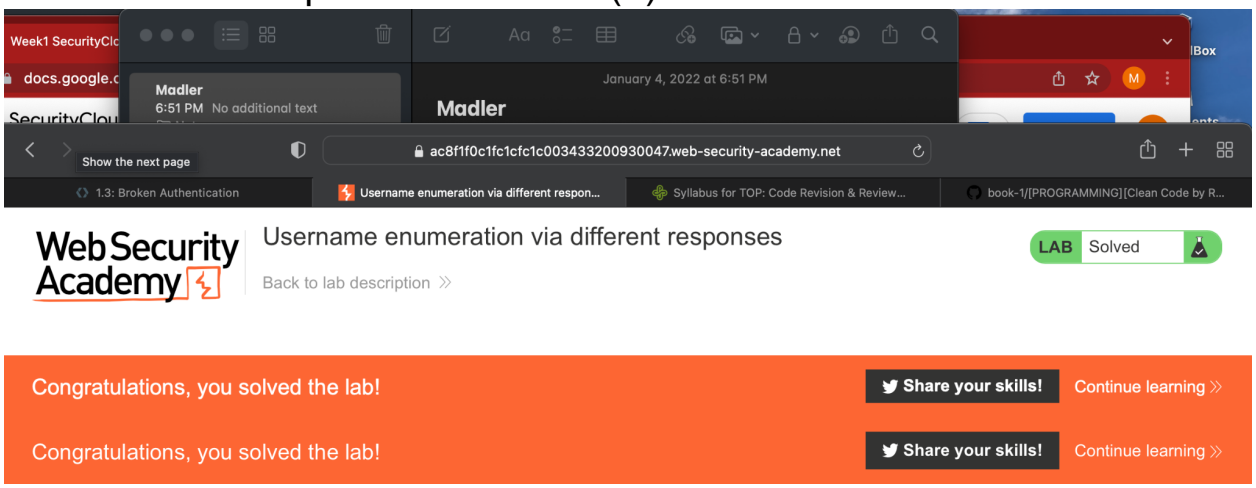

madler (Workspace)
AnxiousForsakenFolder - Replit
Invite
Console
Shell
tle>', '<title>Sign in - Google Accounts</title>', '<title>Google S
/title>', '<title>Google</title>', '<title>Privacy & Terms - Google<
/title>', '<title>Google News</title>', '<title>Gmail</title>', '<title>
Google</title>', '<title>Google</title>', '<title>Google</title>', '<tit
le>Google</title>', '<title>Google Developers</title>', '<title>Google M
arketing Platform - Unified Advertising and Analytics</title>', '<title>
Google</title>', '<title>Google</title>', '<title>Google Books</title>',
'<title>Google</title>', '<title>Google</title>', '<title>Google Transl
ate</title>', '<title>Google</title>', '<title>\n Google Account\n
</title>', '<title>Google Chrome - Download the Fast, Secure Browser
from Google</title>', '<title>Sign in - Google Accounts</title>', '<titl
e>Google</title>', '<title>Google Photos</title>', '<title>Google</title
>', '<title>Google</title>', '<title>Google</title>', '<title>Moving on
from Picasa</title>', '<title>Google</title>']
Async version: 11.73

1.3 Broken Authentication

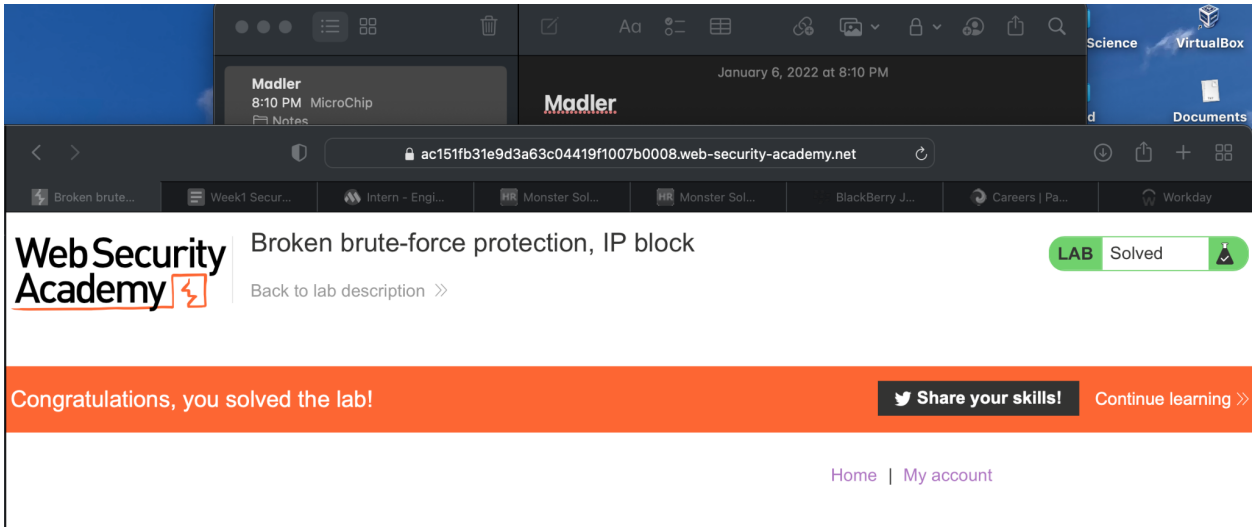
2. Login form



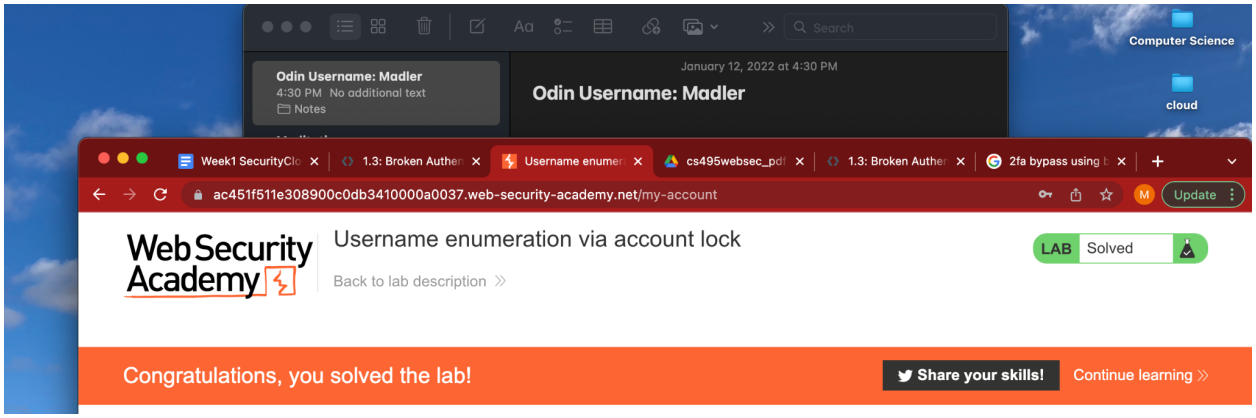
3. authentication/password-based (1)



4. authentication/password-based (2)



5. authentication/password-based (3)



6. oauth

- What is the DNS name of the identity provider? **oauthWebSecurity.net**

- What is the client_id that is sent to the identity provider as a URL parameter?

```
client_id: a72barbz7xqq8oo3e8n8k
```

- What is the value of the redirect_uri (e.g. the client application's callback URL) that the identity provider will send the user back to after authentication and authorization is performed?

```
redirect_uri: https://ac801f4c1e42579ac0ea036b00db0099.web-security-academy.net/oauth-callback
```

- What scopes are being requested by the client application for the user to authorize?

```
scope: openid profile email
```

- What kind of response_type is being requested from the identity provider?

```
response_type: token
```

View the response headers of the request.

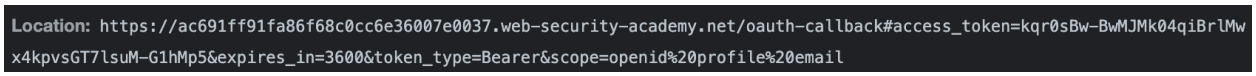
- What is the Location the user is sent to that implements the authentication login form on the identity provider's site?

7. -

- What is URL is the form data sent to when the user logs in as specified in the action attribute of the form?
- 

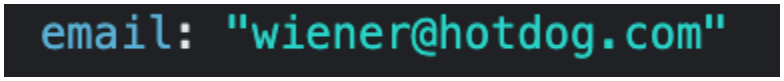
- What is the URL the form is sent to?

- What is the access token the user will relay to the client application via its OAuth callback URL? **Begins with 'access_token='**



8. -

- What is the e-mail address associated with the wiener account?



10. -

- What is the function of the first two const lines? To get an access token

```
const urlSearchParams = new URLSearchParams(window.location.hash.substr(1));
const token = urlSearchParams.get('access_token');
```

- What content is being retrieved from the identity provider in the first fetch?

```
fetch('https://oauth-acd41faf1e45d32bc0f63dbe02fb0021.web-security-academy.net/me', {
    method: 'GET',
    headers: {
        'Authorization': 'Bearer ' + token,
        'Content-Type': 'application/json'
    }
```

- What 3 values are being sent to the client application in the second fetch?

```
    email: j.email,
    username: j.sub,
    token: token
)
```

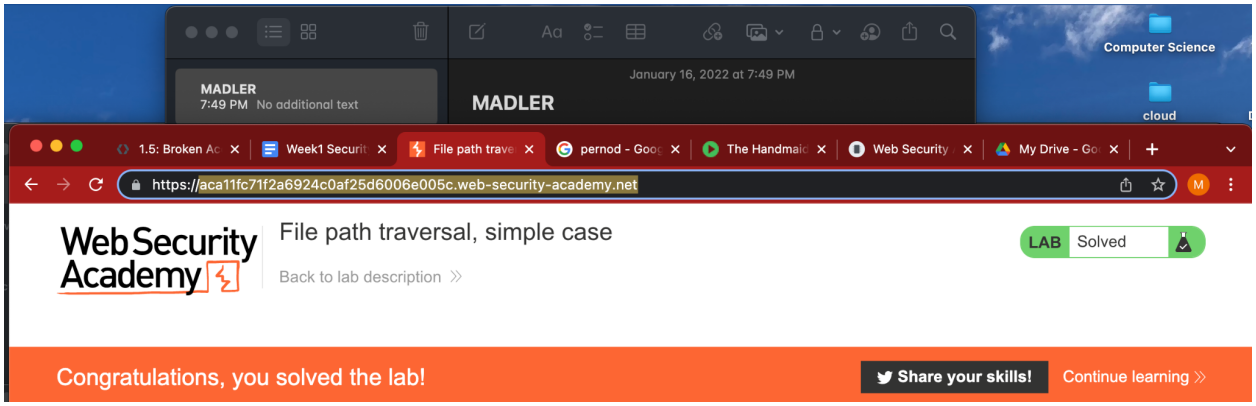
- What location is the user redirected to at the end of the implicit flow?

```
.then(r => document.location = '/'))
```

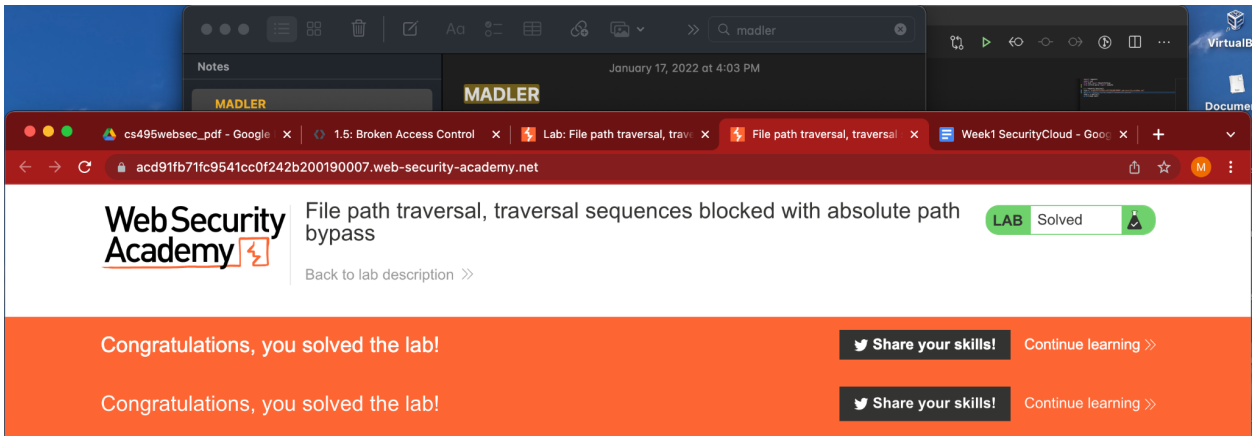
1.5 Broken Access Control

1. file-path-traversal (1)

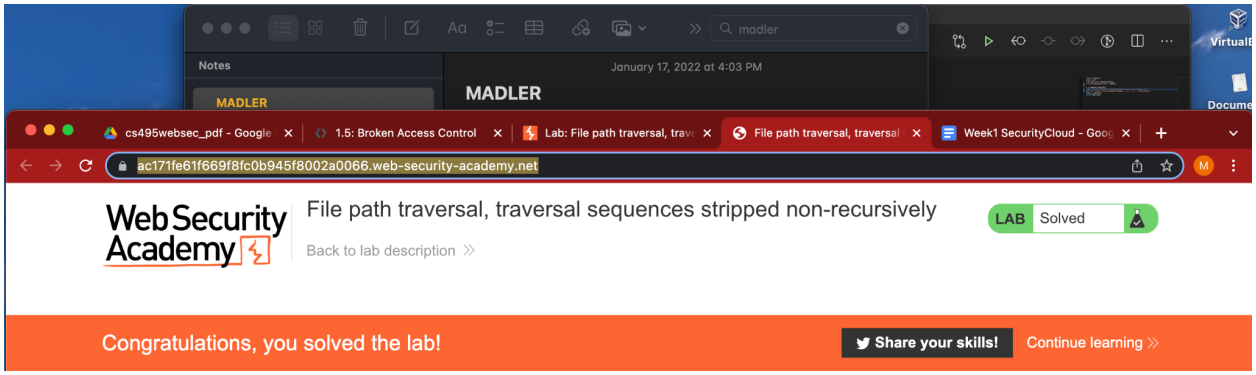
[simple]



absolute-path-bypass

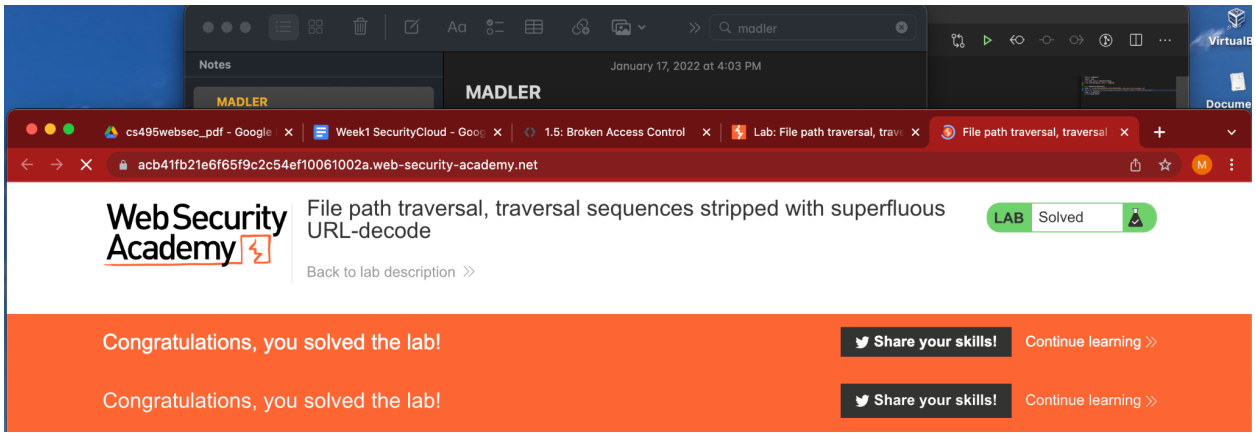


sequences-stripped-non-recursively

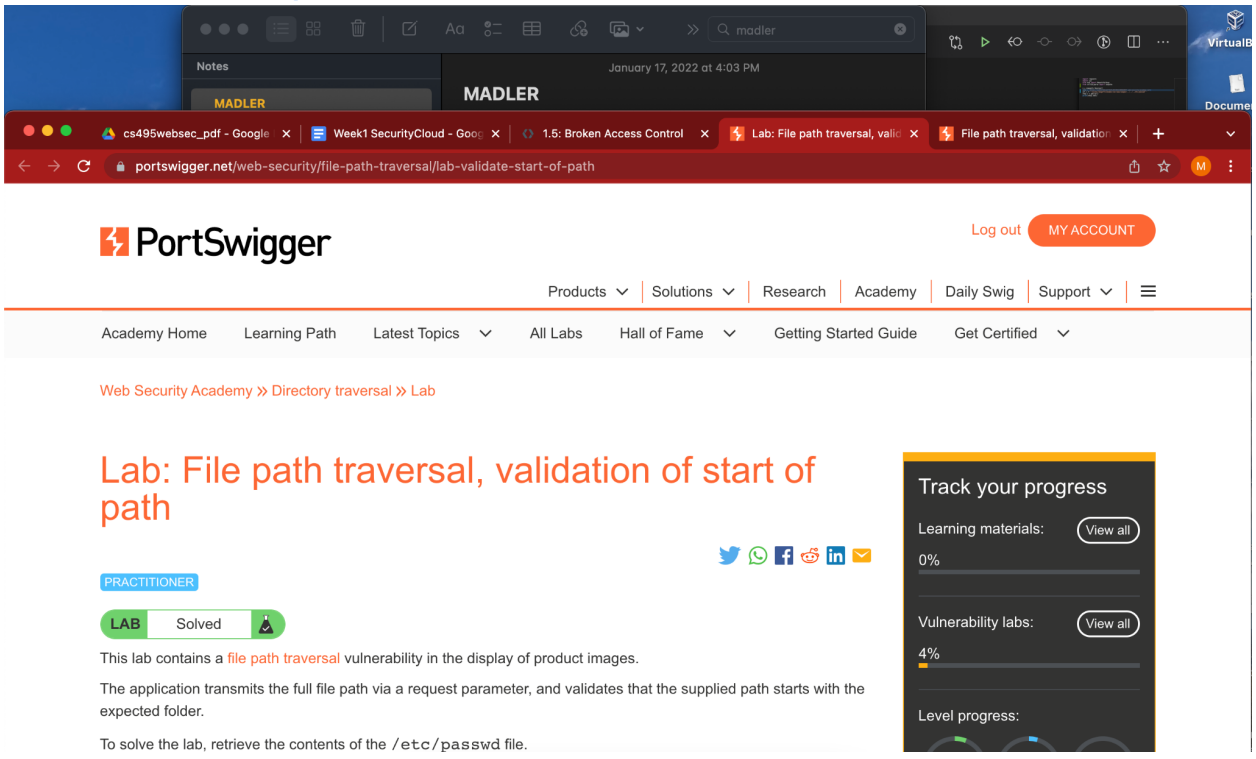


2. file-path-traversal (2)

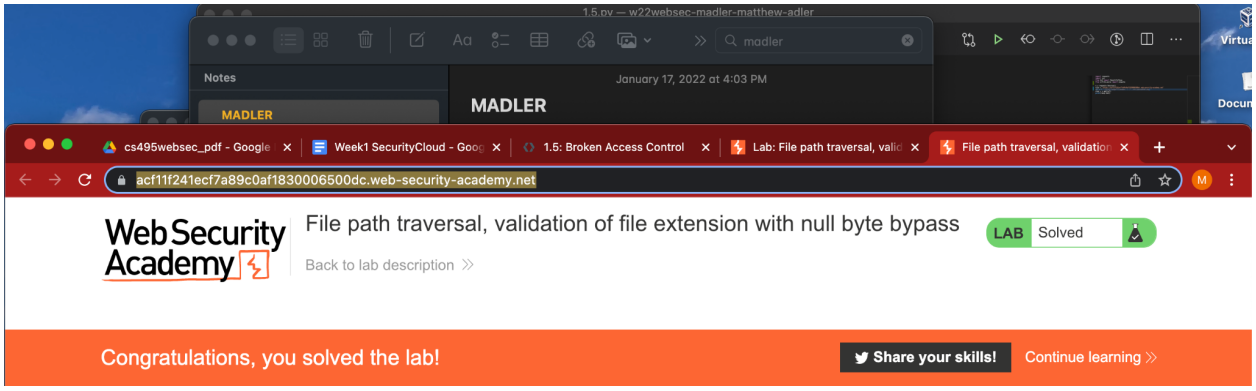
superfluous-url-decode



validate-start-of-path

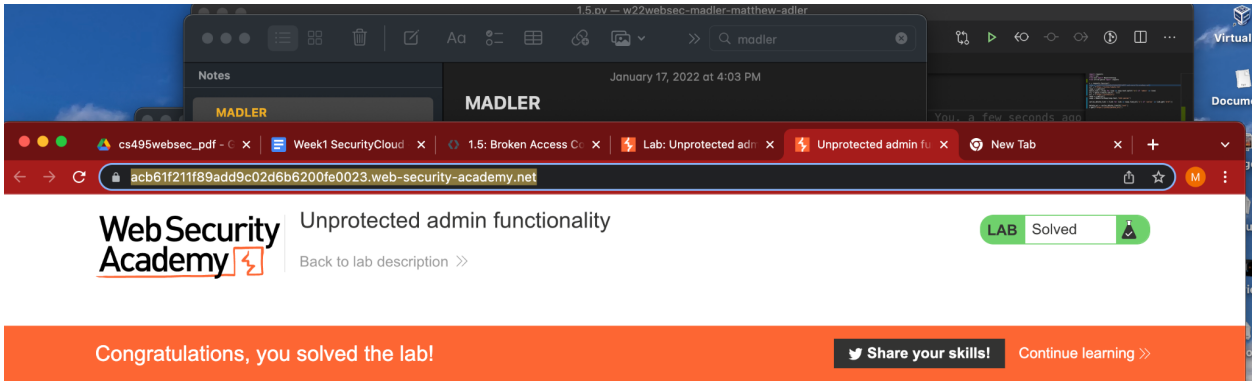


validate-file-extension-null-byte-bypass

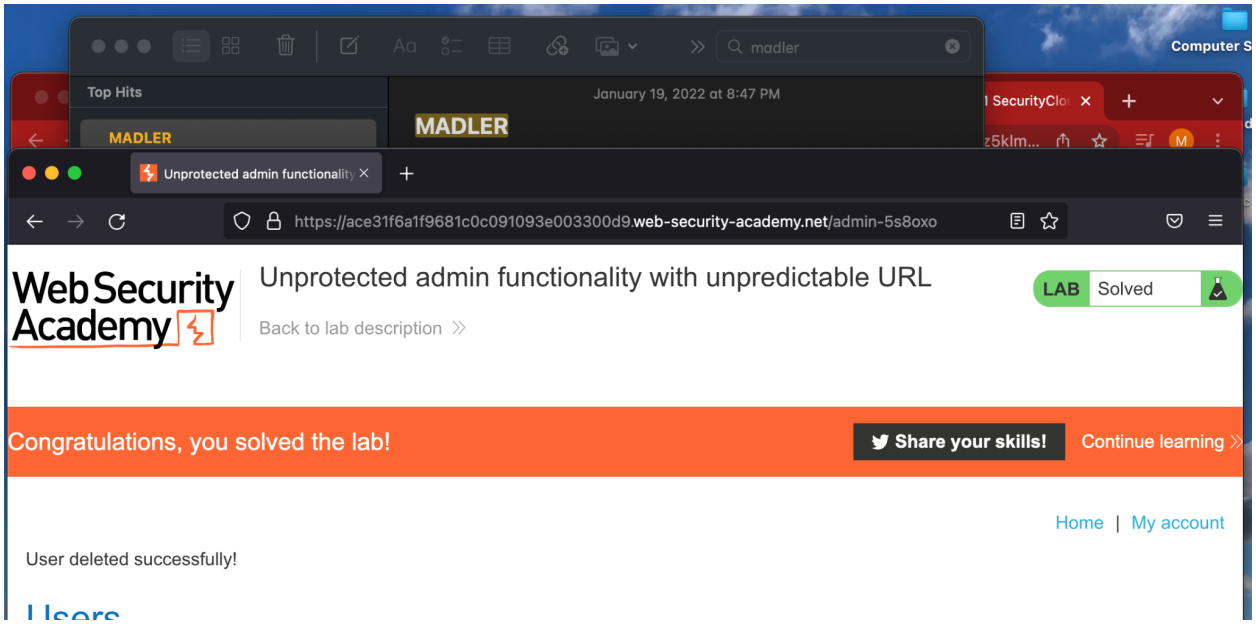


3. access-control (1)

[unprotected-admin-functionality]

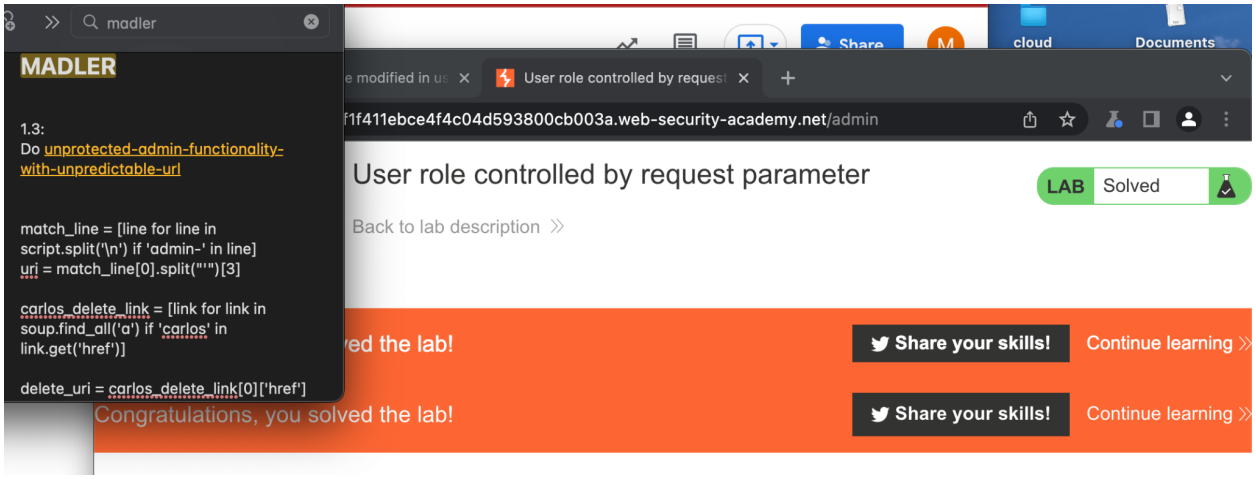


[unprotected-admin-functionality-with-unpredictable-url]



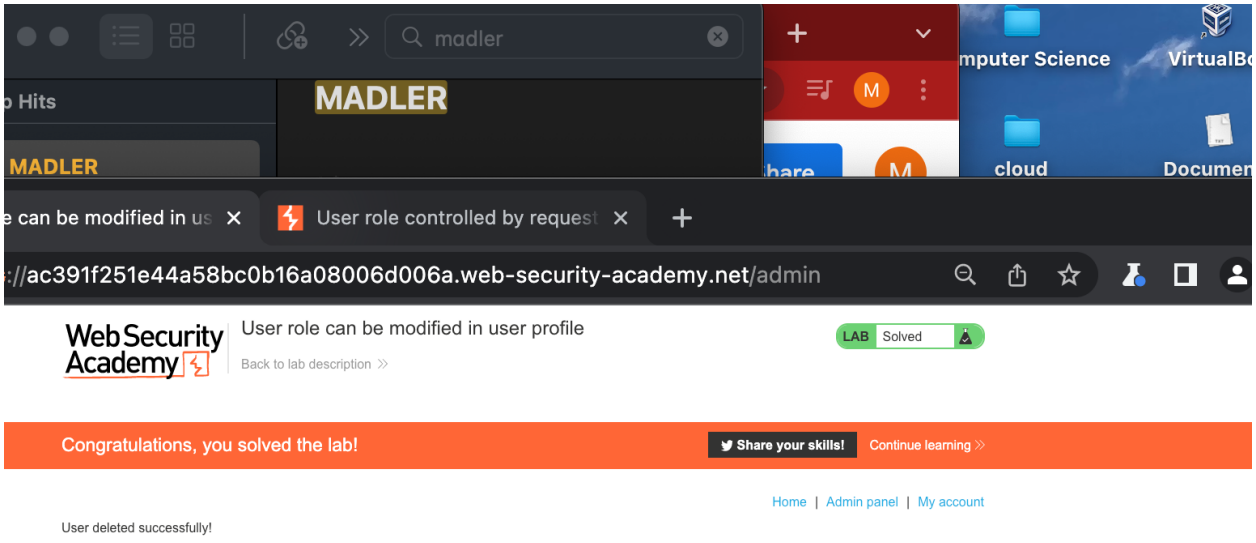
4. access-control (2)

[User-role-controlled-by-request-parameter]



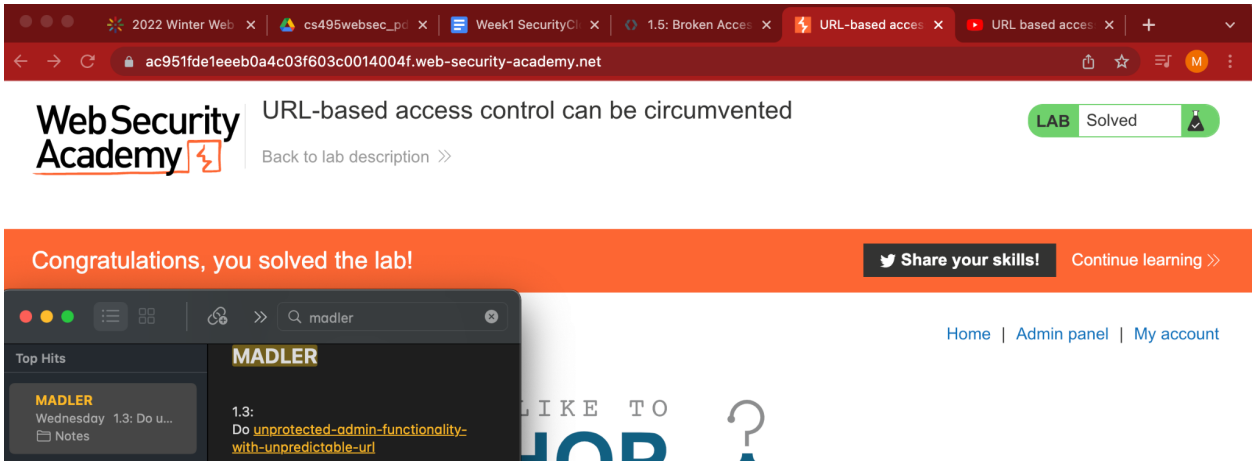
5. access-control (3)

[user-role-can-be-modified-in-user-profile]



6. access-control (4)

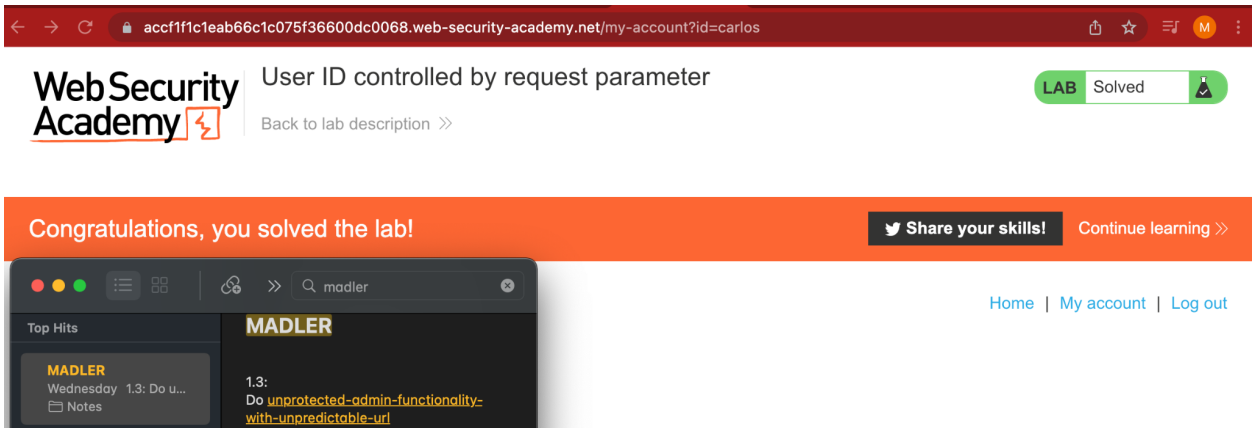
[url-based-access-control-can-be-circumvented]



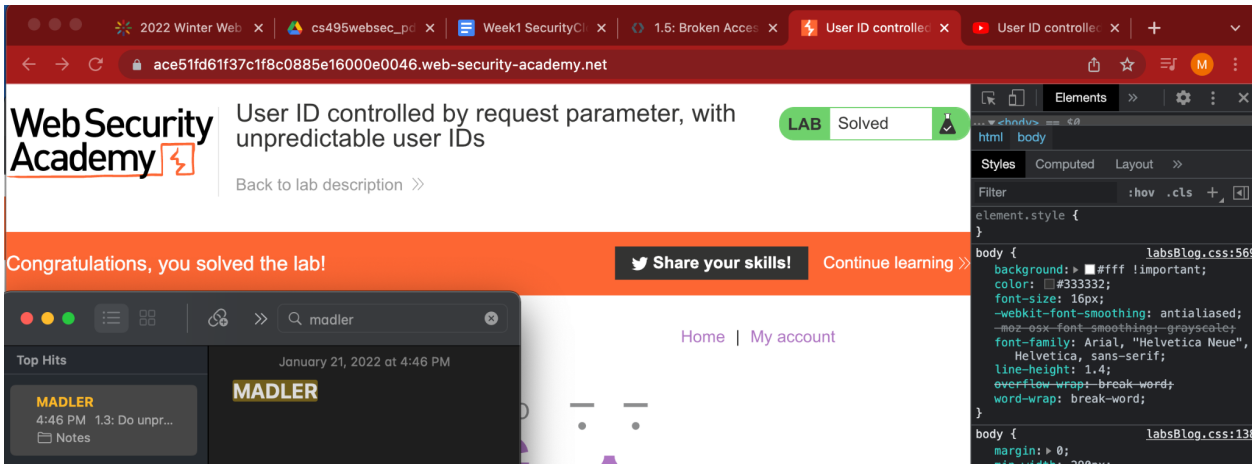
[method-based-access-control-can-be-circumvented]

7. access-control (5)

[user-id-controlled-by-request-parameter]

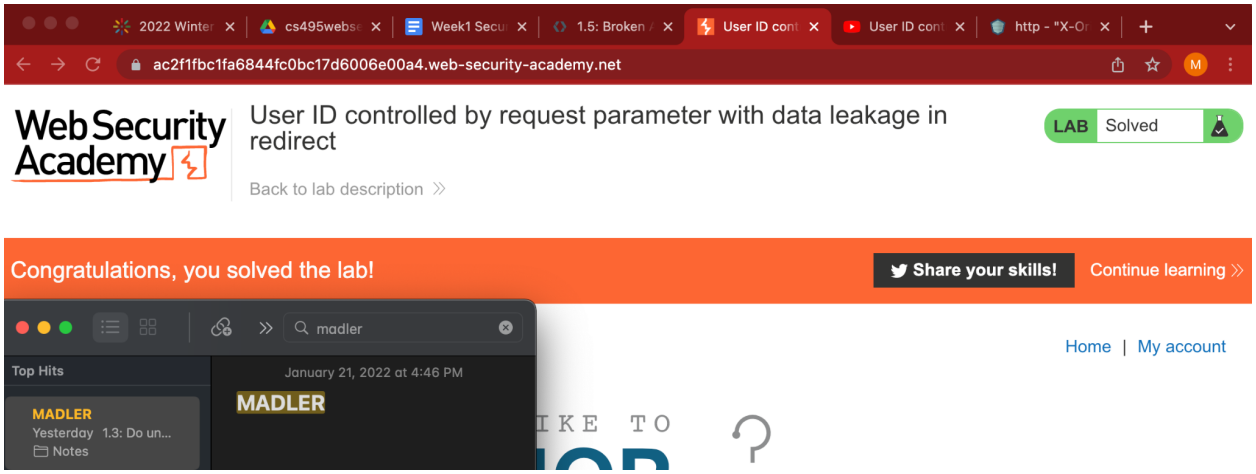


[user-id-controlled-by-request-parameter-with-unpredictable-user-ids]

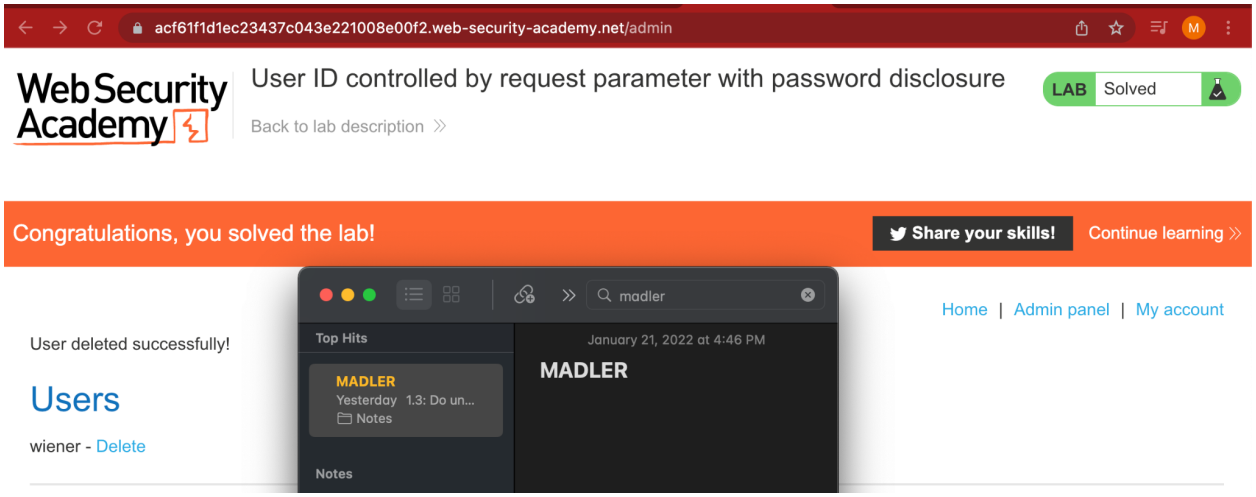


8. access-control (6)

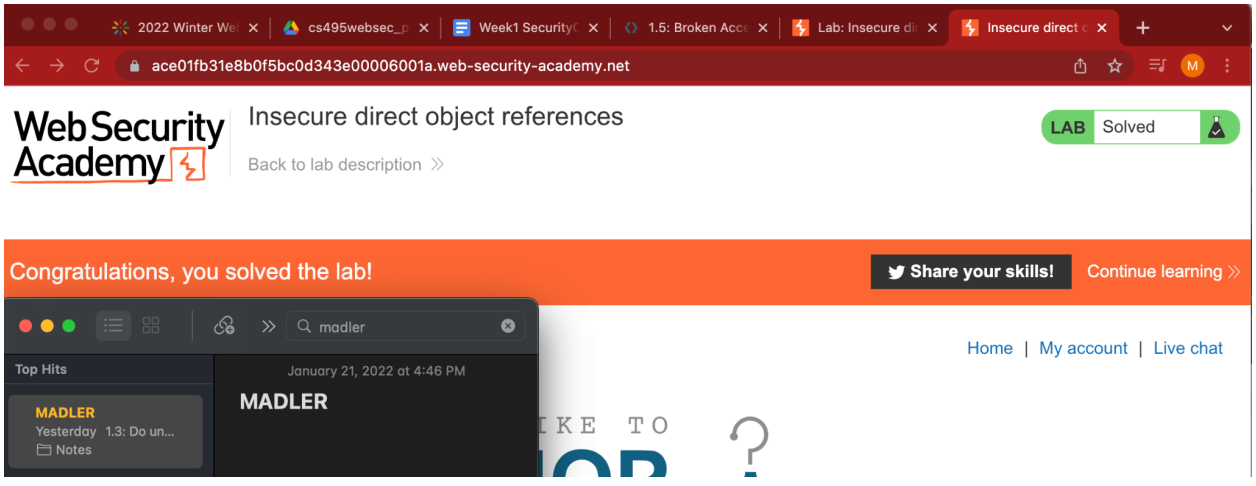
[user-id-controlled-by-request-parameter-with-data-leakage-in-redirect]



user-id-controlled-by-request-parameter-with-password-disclosure

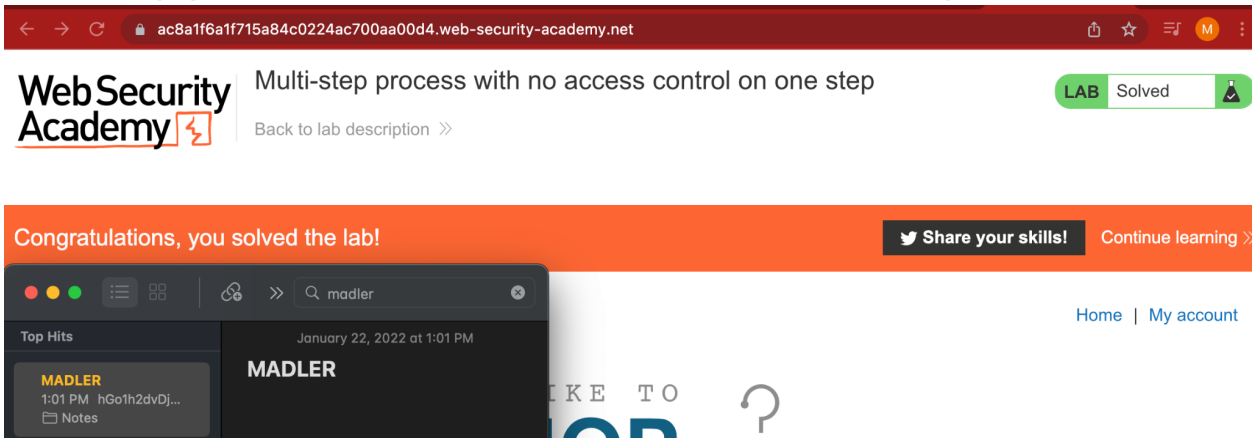


insecure-direct-object-references

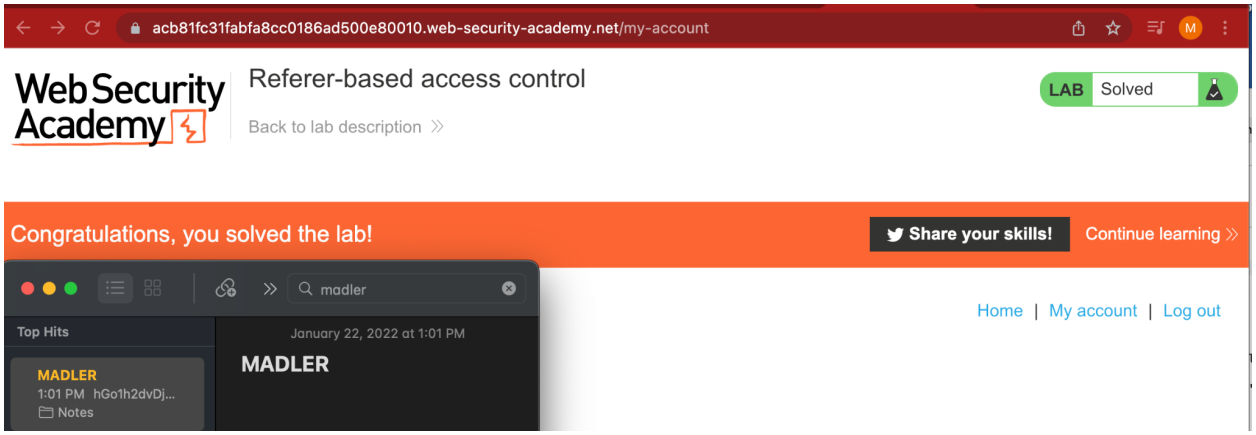


9. access-control (7)

[multi-step-process-with-no-access-control-on-one-step]

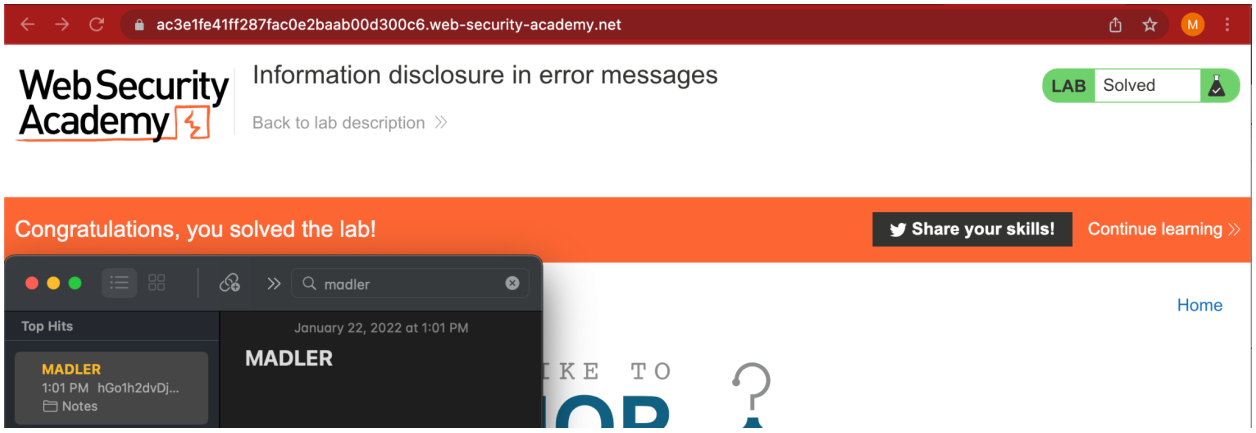


[Referer-based-access-control]

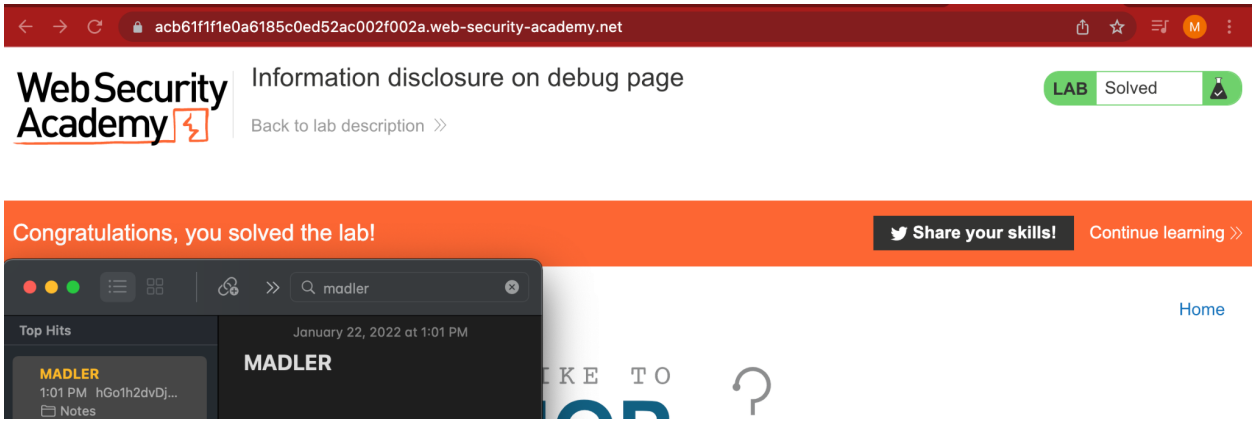


10. Information-disclosure

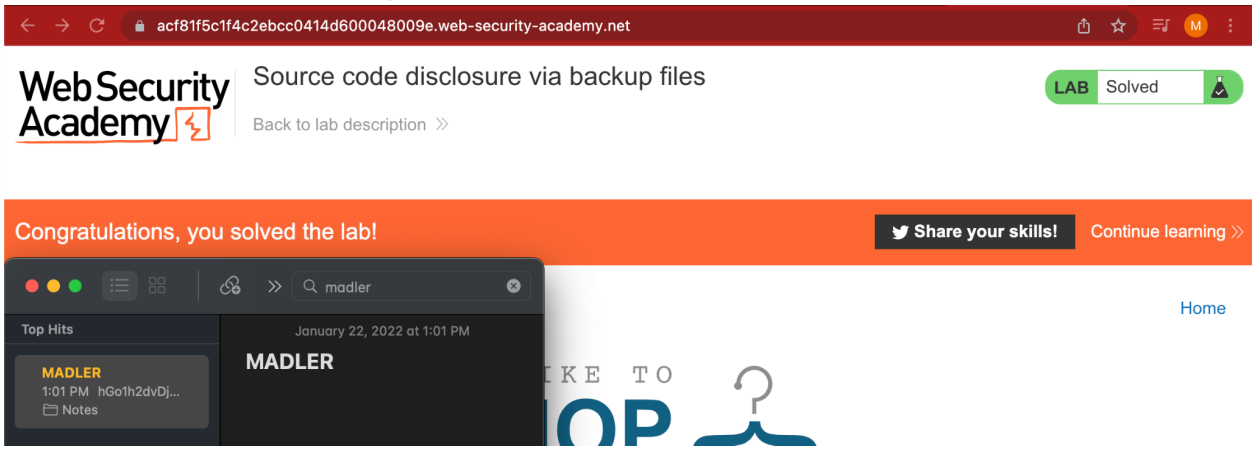
[lab-infoleak-in-error-messages]



lab-infoleak-on-debug-page



lab-infoleak-via-backup-files



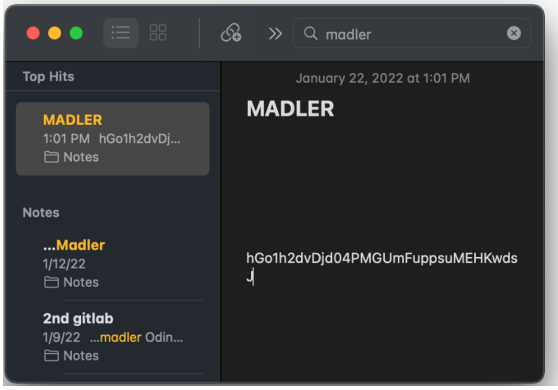
11. WFP1: File upload

Example #1

```
1.php
1.txt
hacker.jpg
```

Last line of the output: hacker.jpg

Return value: 0

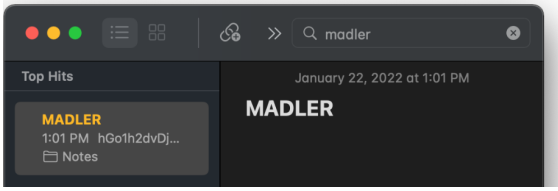


Example #2

```
/var/www/upload/images
```

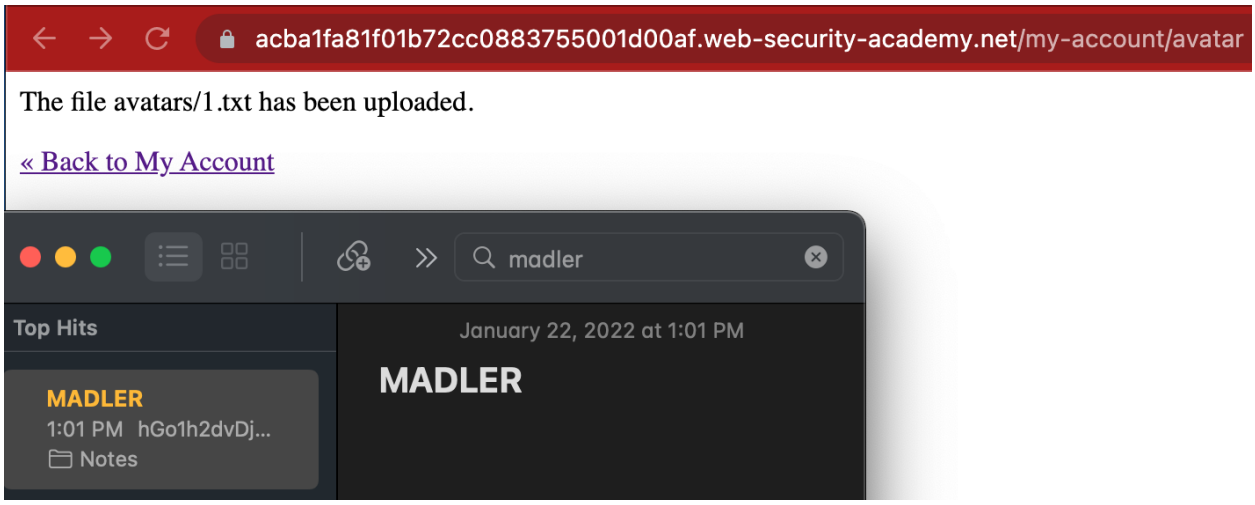
Last line of the output: /var/www/upload/images

Return value: 0



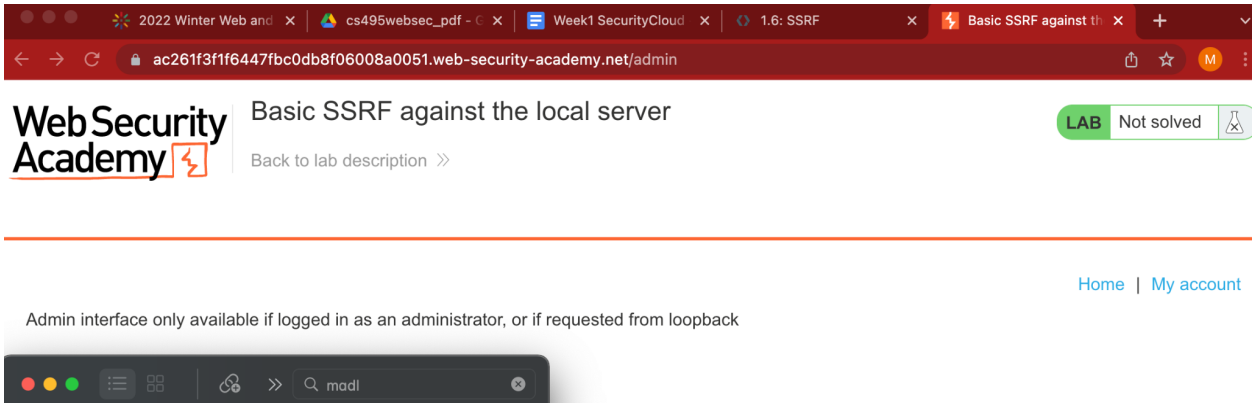
12. file-upload (1)

- What are the names of the form fields that are of type hidden? CSRF, user.
- Take a screenshot of the message and the URL the avatar is stored at on the server
-

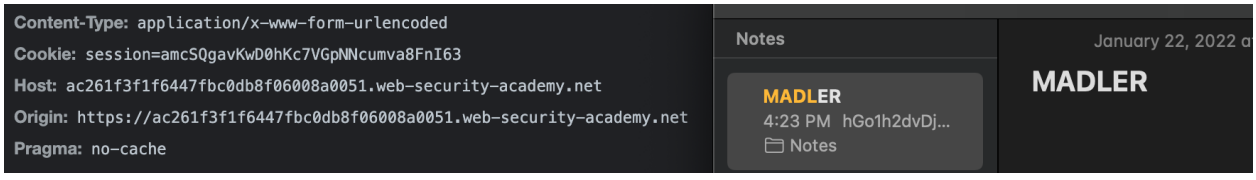
- 

13. file-upload (2)

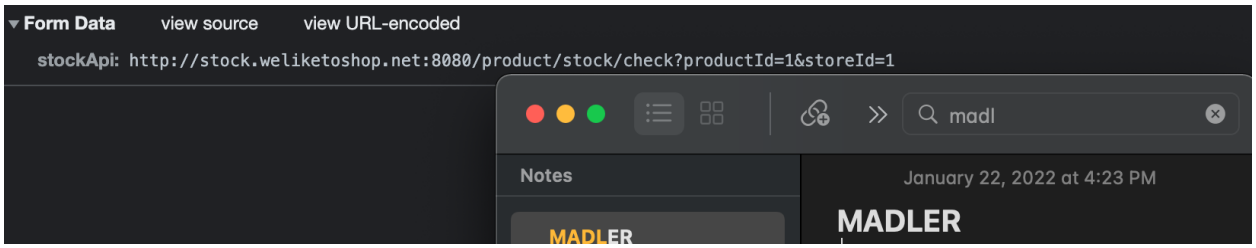
- Take a screenshot of the message that is returned when you attempt to visit the /admin path of your site.



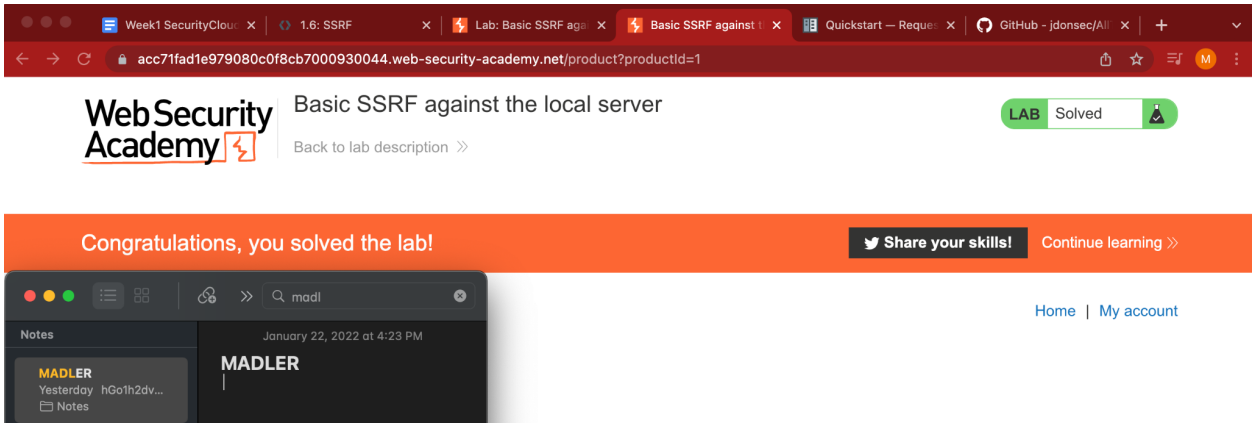
- Take a screenshot that shows the Content-Type: request header that specifies the format of the form submission



- Then, take a screenshot of the form submission payload.

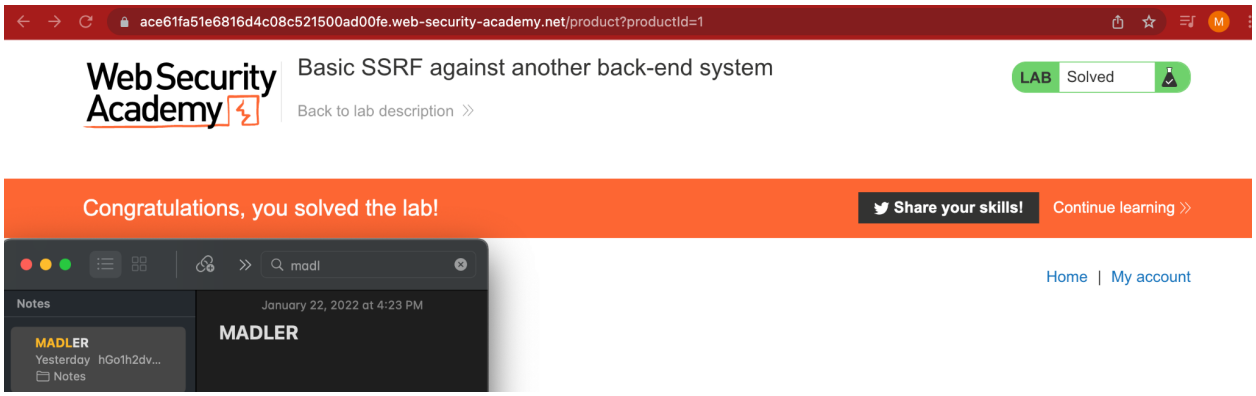


- Take a screenshot showing completion of the level that includes your OdinId

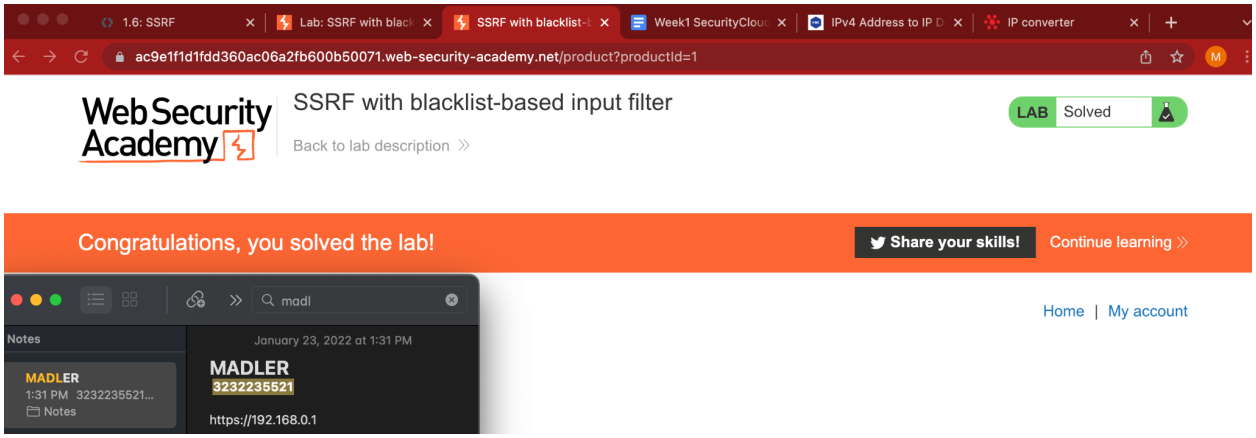


1.6: SSRF

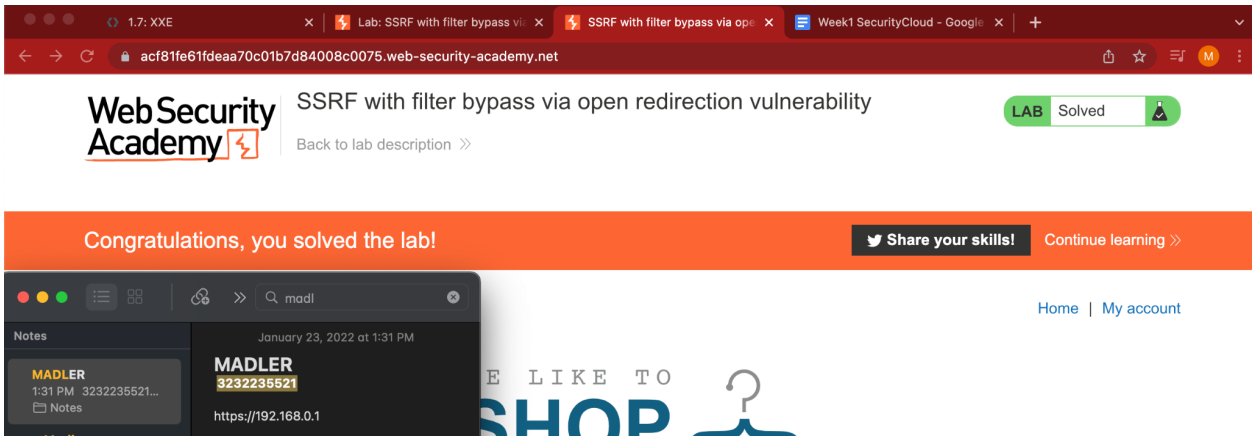
2. ssrf (2)



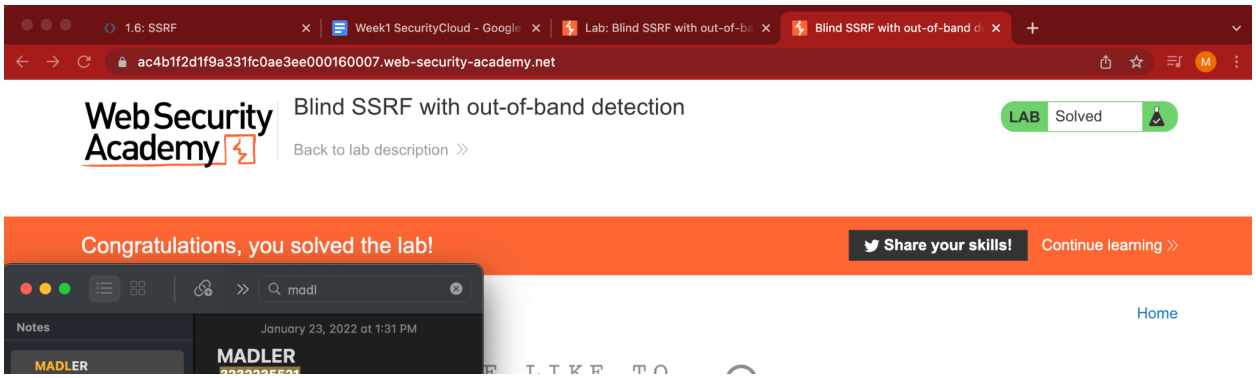
3. ssrf (3)



4. ssrf (4)

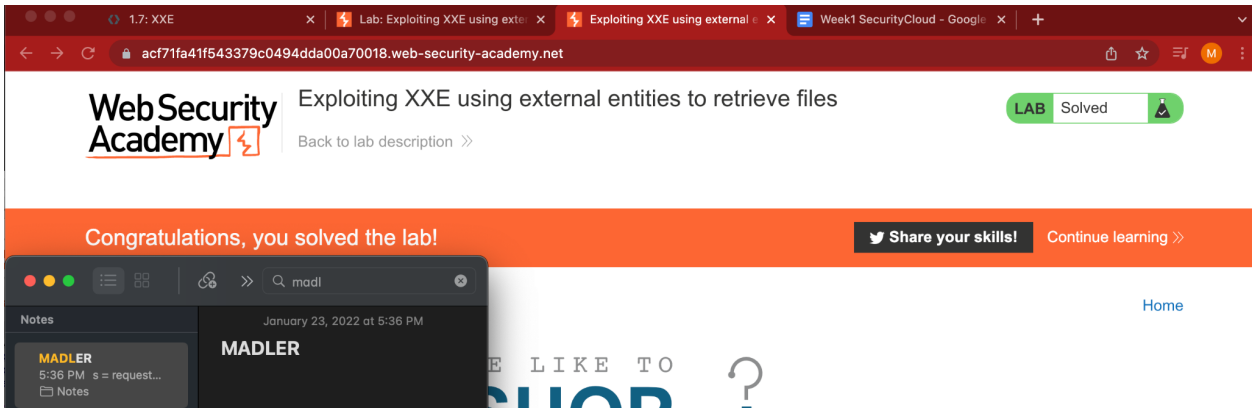


5. ssrf/blind

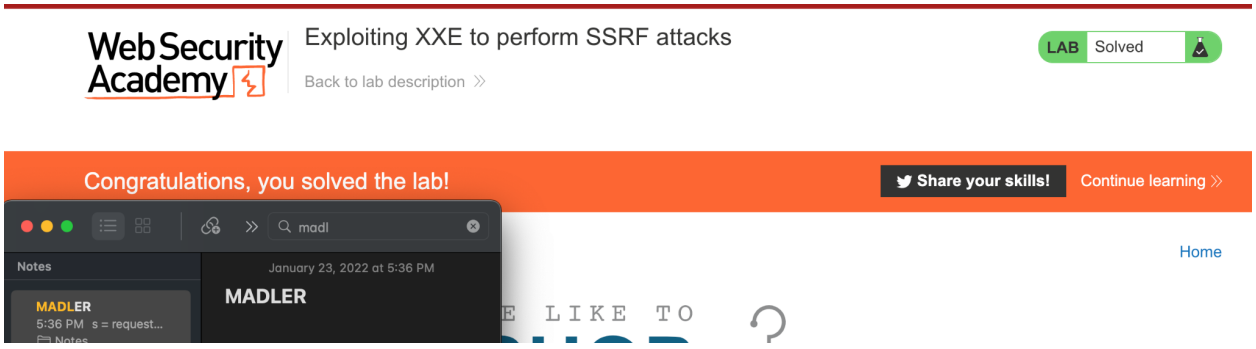


1.7: XXE

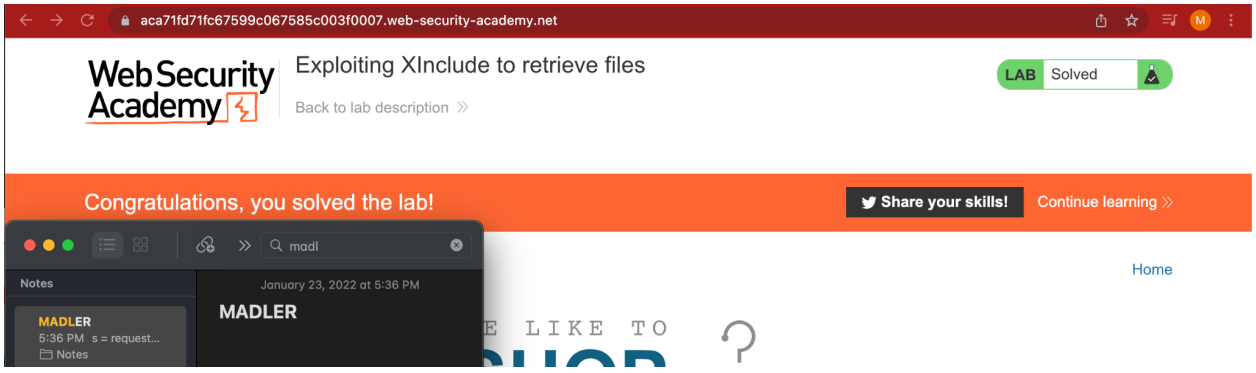
1. xxe (1)



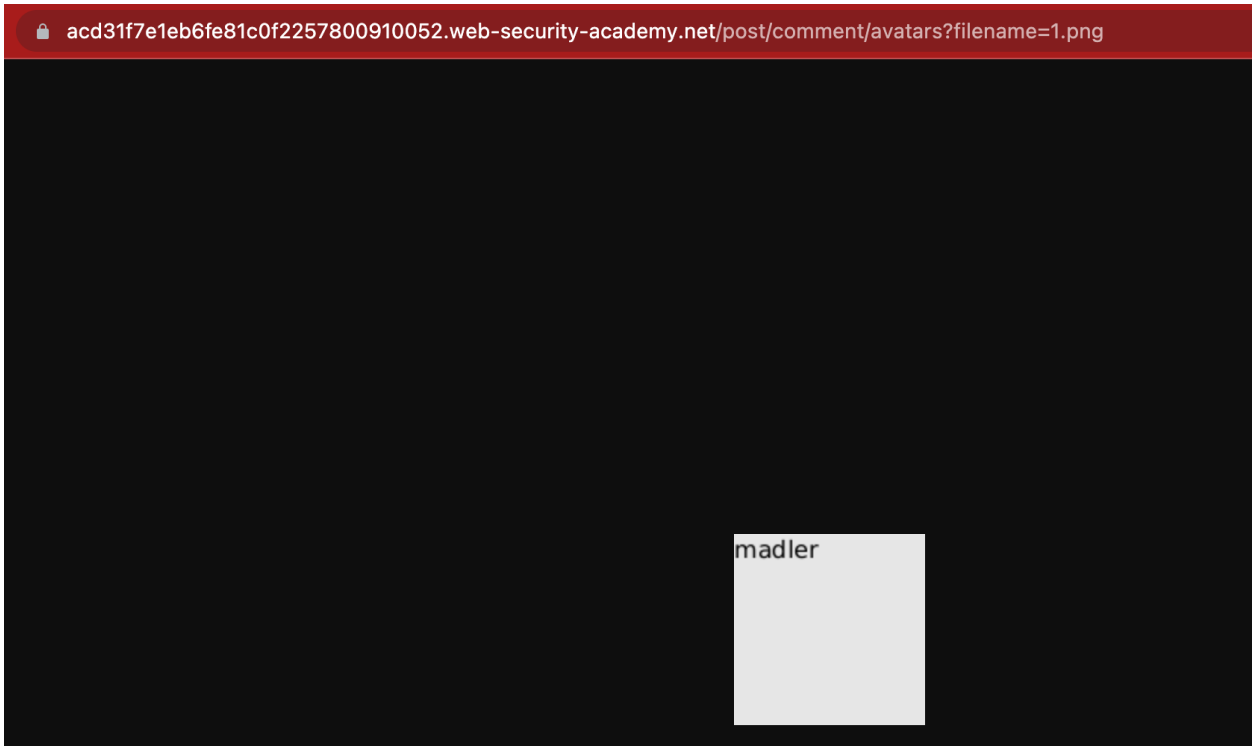
2. xxe (2)



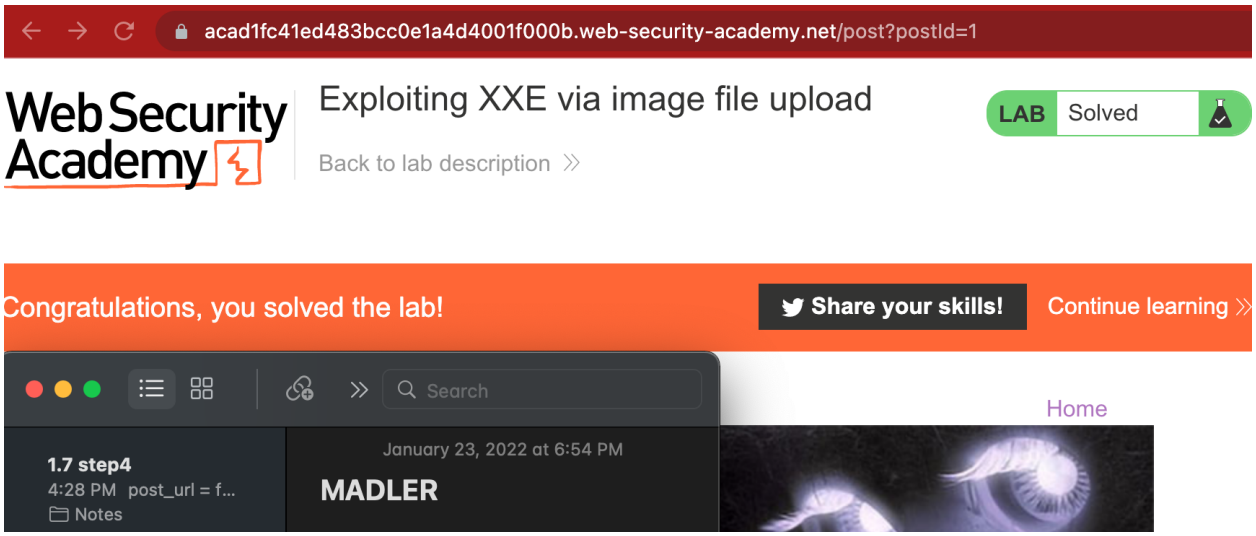
3. xxe (3)



4. xxe (4)


acd31f7e1eb6fe81c0f2257800910052.web-security-academy.net/post/comment/avatars?filename=1.png
madler

5. -


acad1fc41ed483bcc0e1a4d4001f000b.web-security-academy.net/post?postId=1
Web Security Academy
Exploiting XXE via image file upload
Back to lab description
LAB
Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Search
1.7 step4
4:28 PM post_url = f...
Notes
January 23, 2022 at 6:54 PM
MADLER
Home