|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《网络与系统安全》 实验报告** |
|  |
| 实验三  SQL注入 实验  学 院: 计算机科学与技术学院   |  |  | | --- | --- | | 姓 名: | 王志铭 | | 学 号: | 200110611 | | 专 业: | 计算机科学与技术 | | 日 期: | 2023年4月 | |

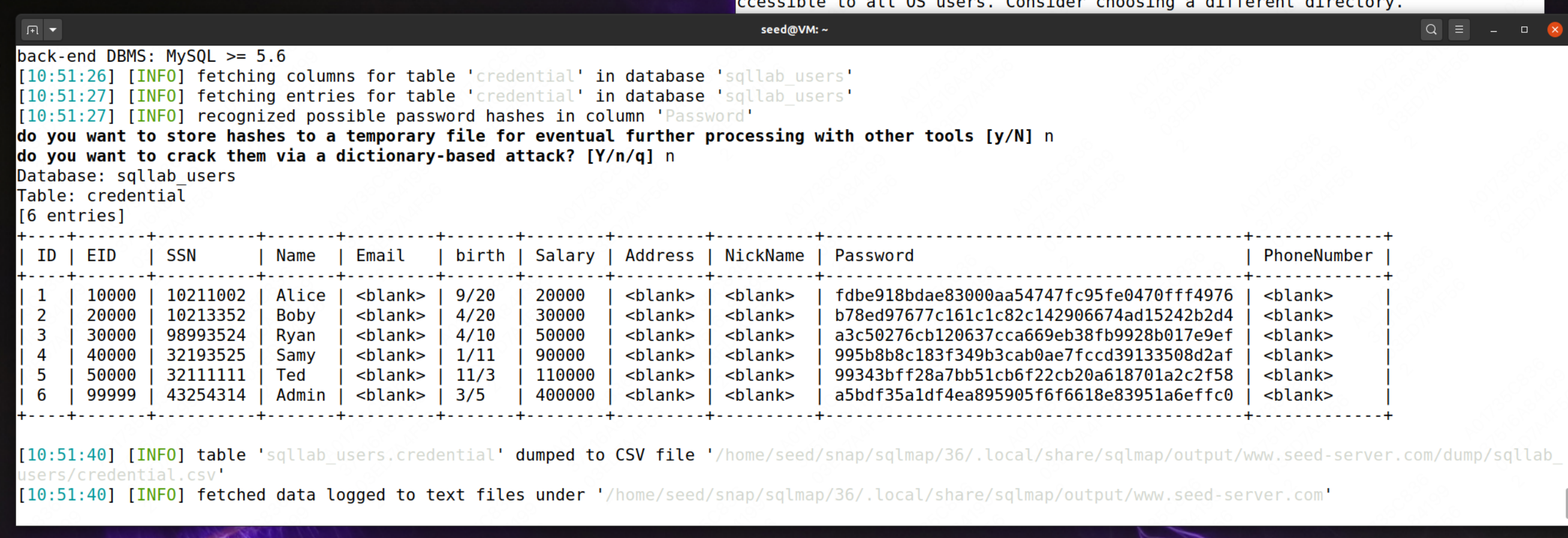
# 一、实验过程

每个实验步骤（共9个小任务（任务1.1、1.2，任务2.1、2.2、2.3，任务3.1、3.2、3.3，任务4））要求有具体截图和分析说明。

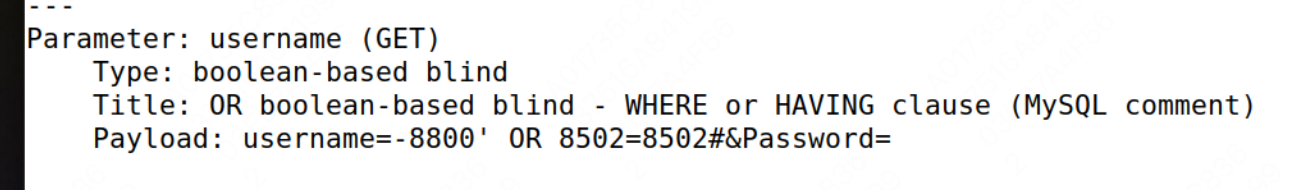
**\*\*任务1.1\*\*** 运行上述命令后，需要使用SQL命令打印员工Alice的所有概要信息。请提供你的结果截图。



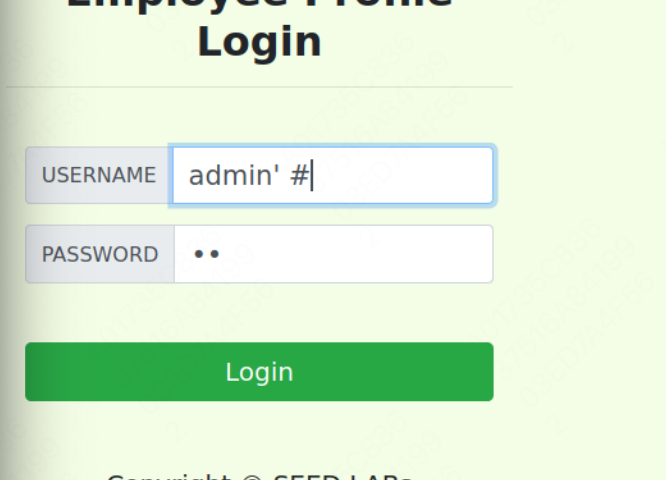
**\*\*任务1.2\*\*** 运行上述命令后，可以将credential中的所有信息dump下来。请提供你的结果截图，并根据第一条命令找出可以注入的信息点。



根据命令1给出的提示，可以采用布尔盲注进行攻击，即伪造 OR 1 = 1 或者用注释#，来通过where子句的条件判断。

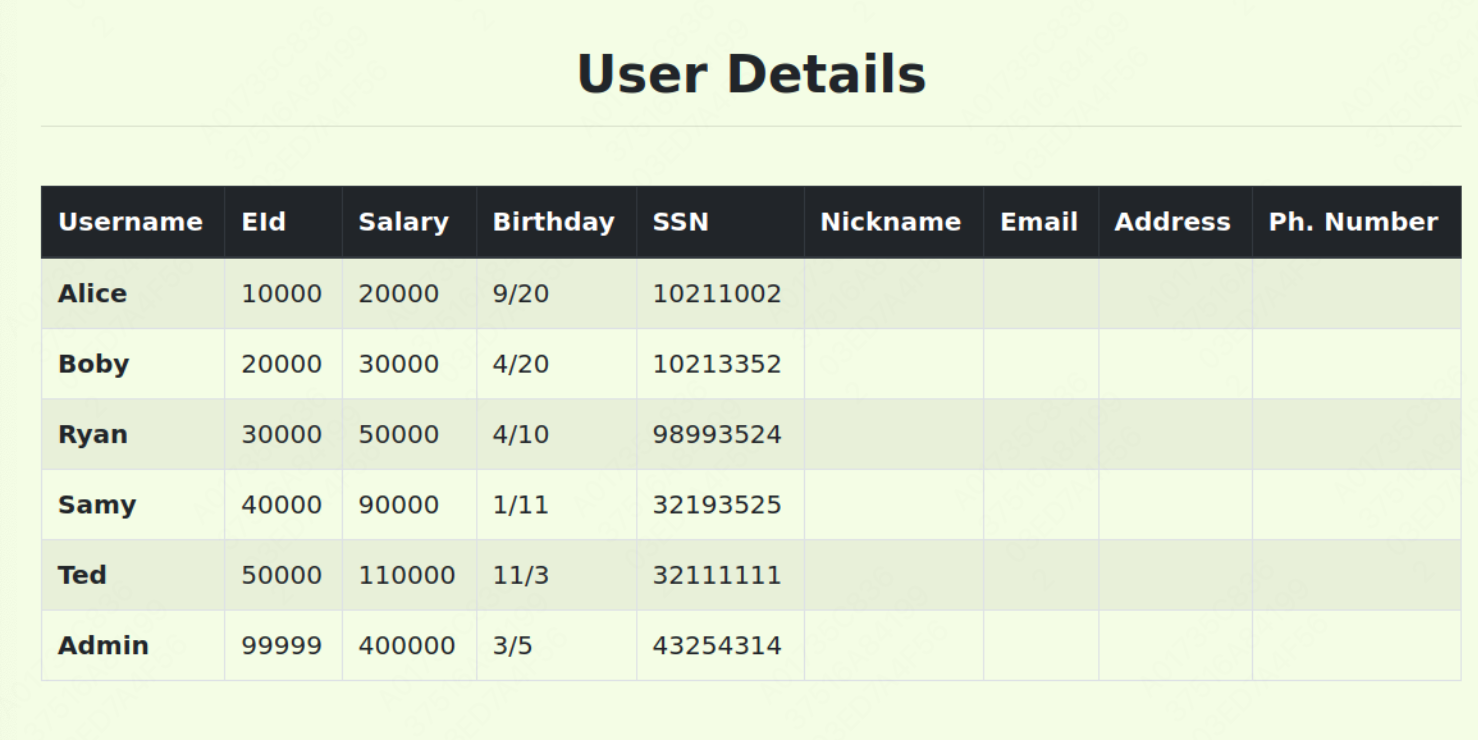


**\*\*任务2.1\*\***:网页SQL注入攻击。利用Admin账户，可以尝试使用其他账户。

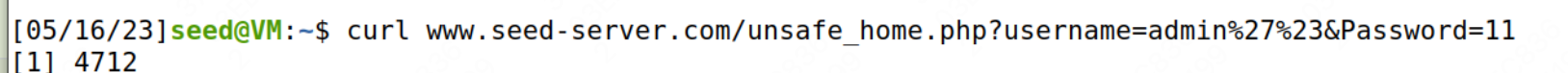


帐号为admin’#，密码任意。

此时原SQL当中的password校验会被注释，实现了攻击。



**\*\*任务2.2\*\***:命令行SQL注入攻击。利用crul命令进行攻击



帐号构造与上面的类似，注意使用转义字符。密码任意。

得到html页面的结果：



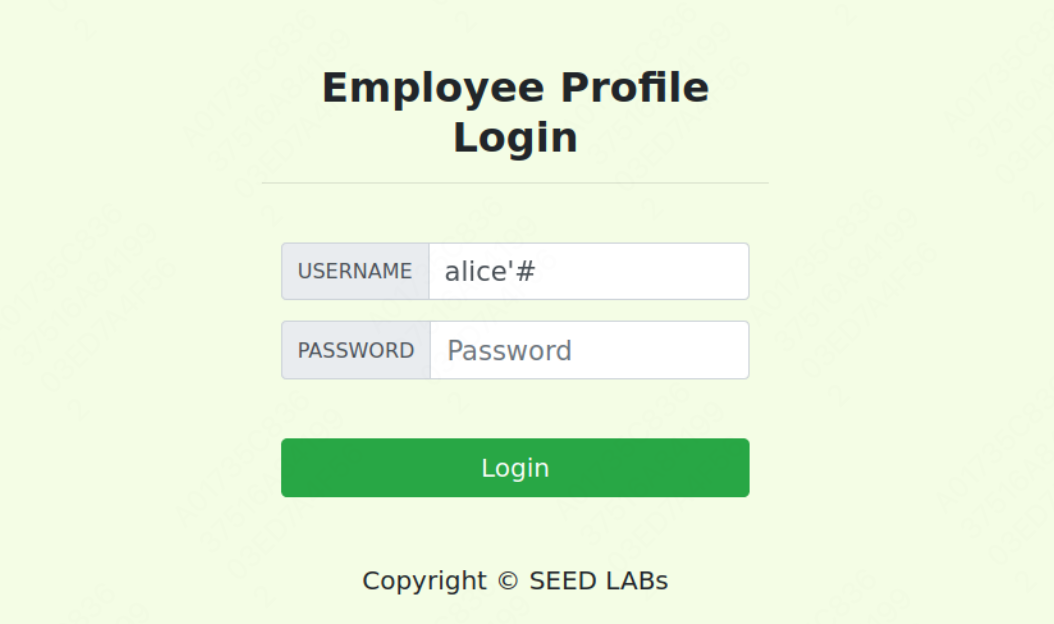
**\*\*任务2.3\*\***:追加一条新的SQL语句。请尝试通过登录页面运行两条SQL语句,并说明是否能够获取到信息。

尝试让帐号为admin'; update credential set Salary = 66666 where Name = 'admin';-- '，从而使原语句转化为两条语句：select … where name = 'admin'; update credential set Salary = 66666 where Name = 'admin';-- ''and password = ‘xxx’;

但是发现攻击失败了。

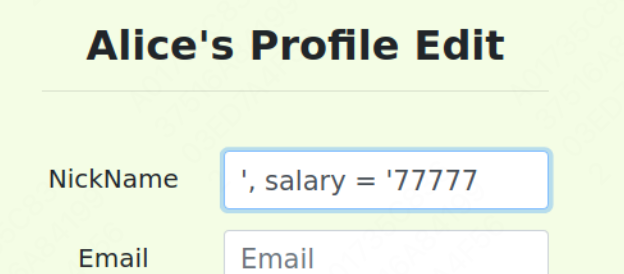


**\*\*任务3.1\*\***:修改自己的工资。

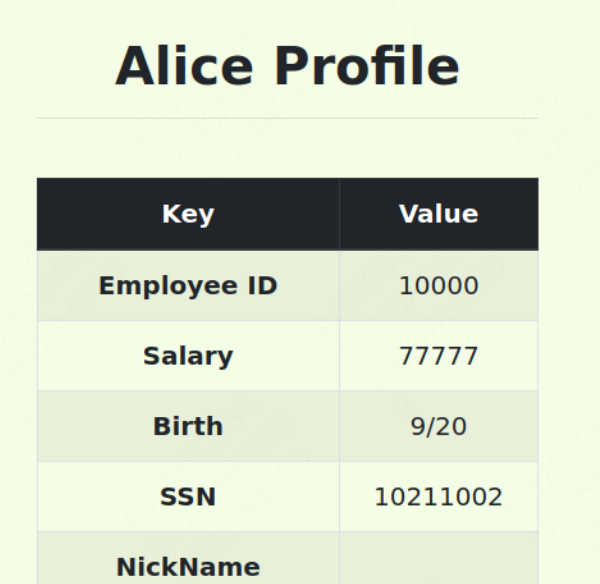


登录进alice的账号，进入编辑界面。

我们的目标是把alice的salary修改为77777 利用SQL注入，在条件里多加一个salary = 77777. 可以从nickname入手，填入如下内容：



可以看到，修改成功了。



**\*\*任务3.2\*\***:修改其他人的工资。

利用update注入，将admin的工资修改为1.

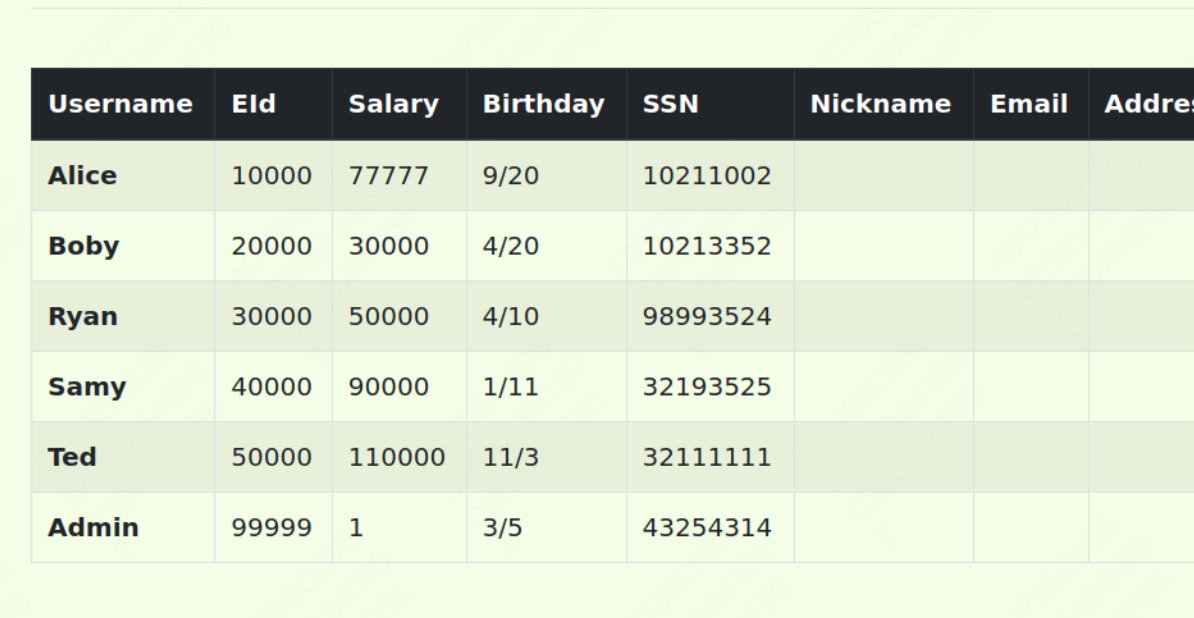
同样从nickname入手，注入salary和where，其余的部分用# 注释掉。

在nickname处填入

', salary = 1 where name = 'admin' #

即原语句变成 … set nickname = ‘’, salary = 1 where name = ‘admin’# …

此时再进admin，可以看到salary变成1了。



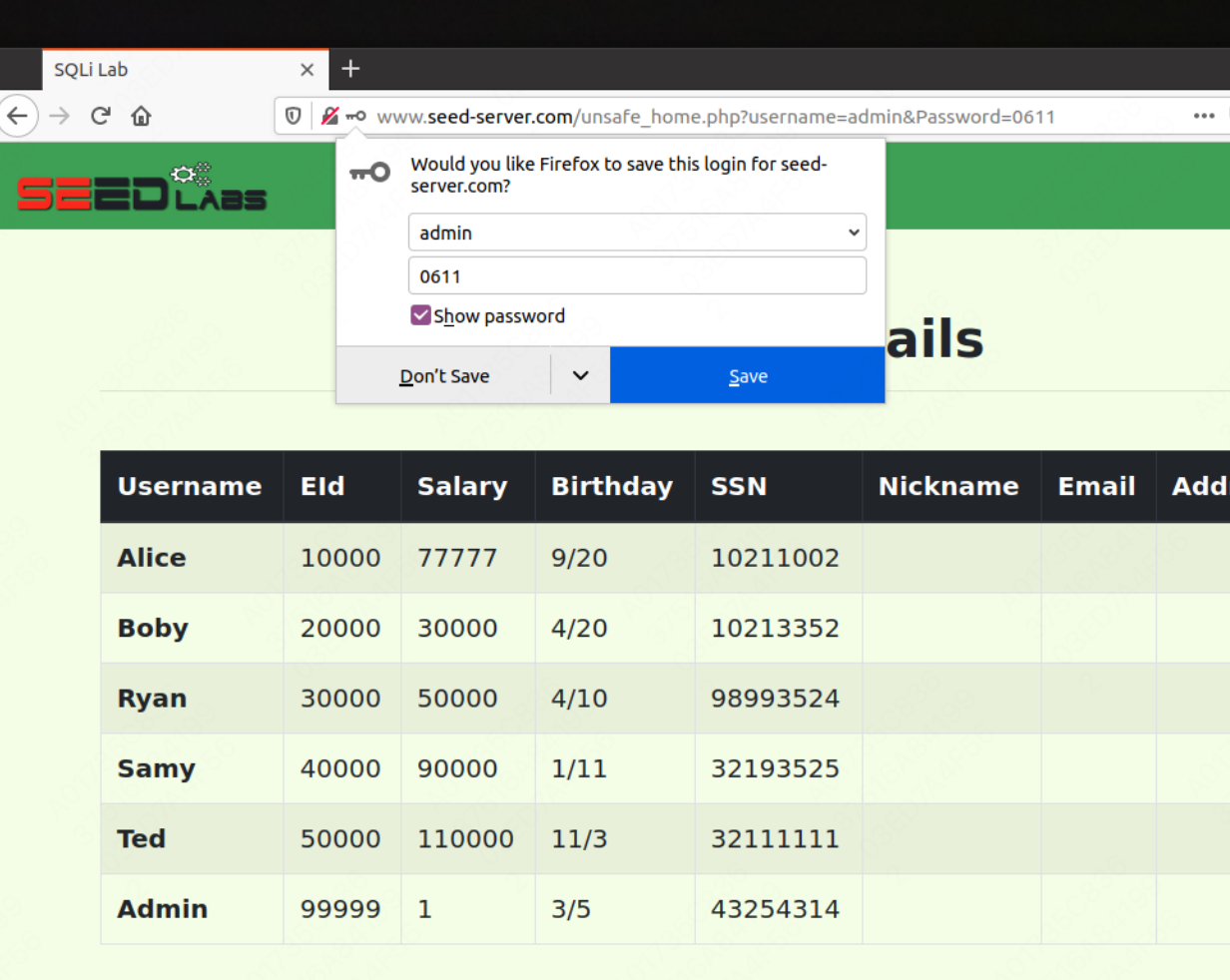
**\*\*任务3.3\*\***:修改他人密码。

修改密码与修改工资是类似的，唯一的不同在于密码还额外采用了SHA1加密。我们想要将密码修改成0611. 对应的SHA为

91108d18fcc80f97e015b3e24f56361f312cff22

我们在nickname处填入

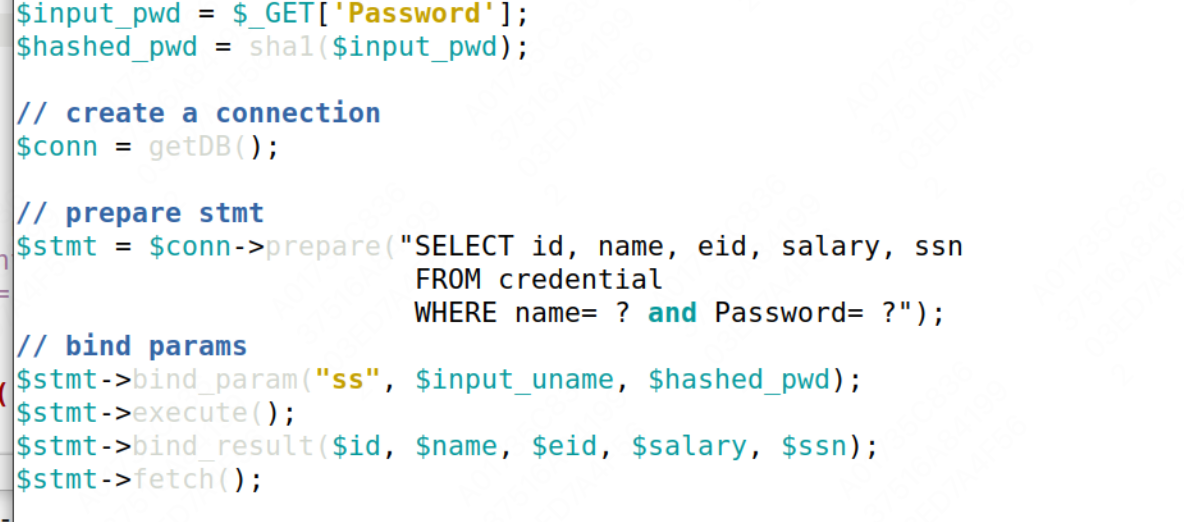
', password = '91108d18fcc80f97e015b3e24f56361f312cff22' where name = 'admin' #



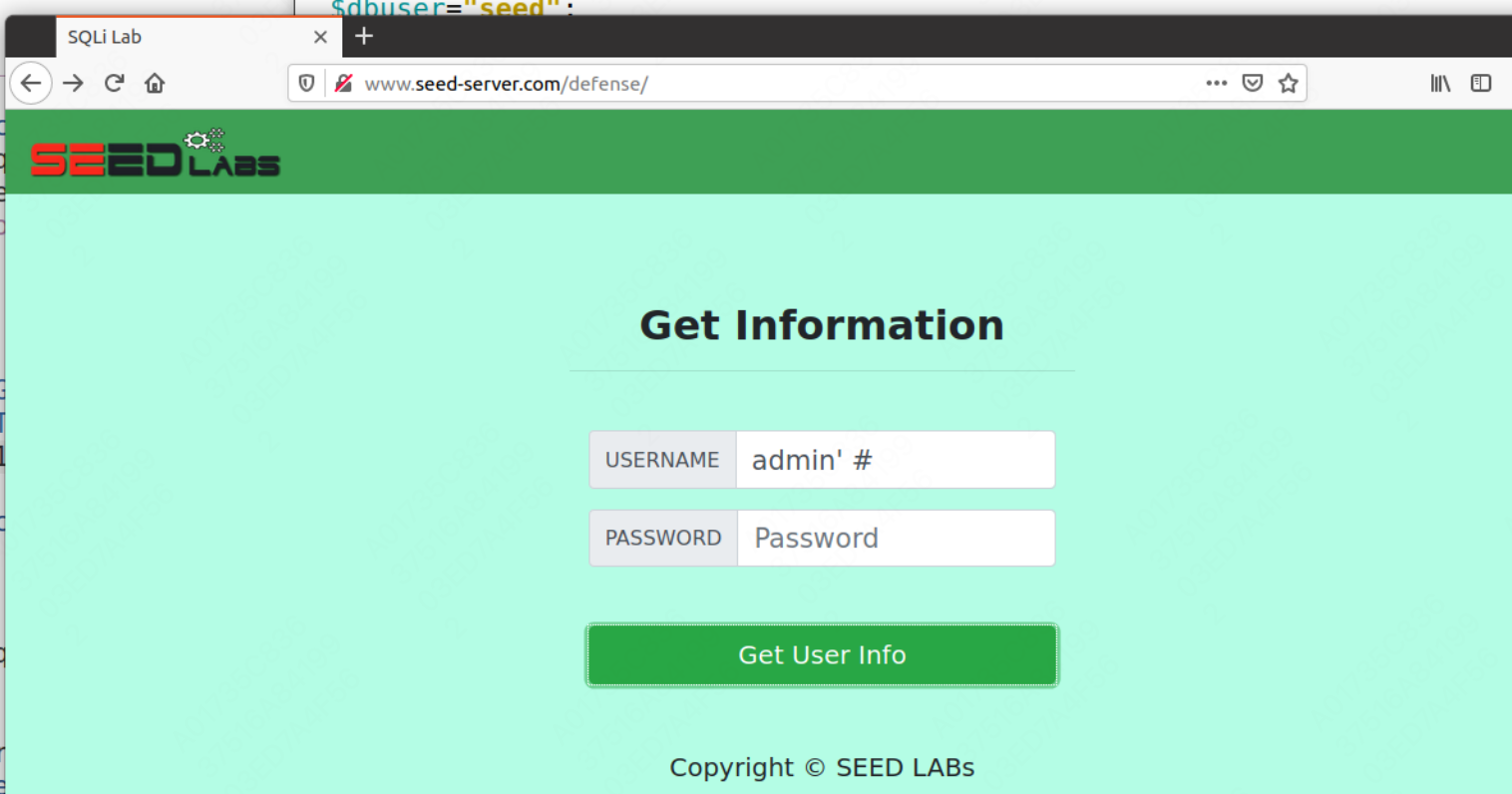
可以看到，我们用0611成功进入了admin。

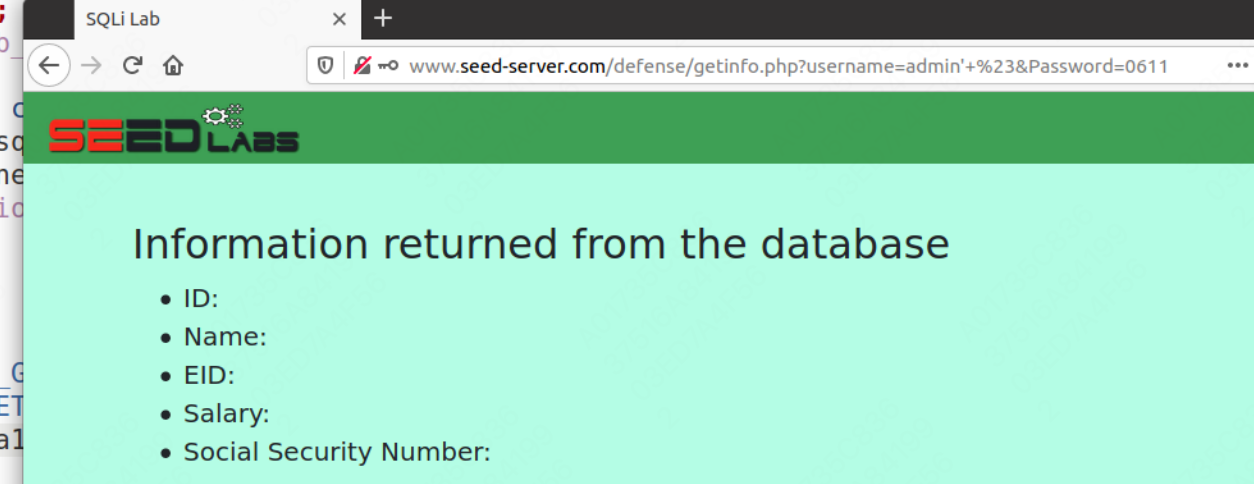
**\*\*任务4\*\***：在这个任务中，我们将使用预处理语句机制来修复SQL注入漏洞。

利用预处理preparedStatement来避免SQL注入，修改代码如下，通过bind来传入参数，以及读出数据。



此时再用 admin’#去攻击，发现已经无法得到结果了。





如果你在任务4中还完善了其他代码，请一并说明。

有余力的同学可以完善下unsafe\_home.php、unsafe\_edit\_frontend.php和unsafe\_edit\_backend.php代码文件，把相关的漏洞修复。

# 二、遇到问题及解决方法

本次实验进行得比较顺利，但是还是遇到了一些小问题。

首先是单引号‘匹配的问题，做的时候偶尔会漏掉，这时候就需要多检查一下自己写的攻击语句，放到整个SQL里面看看。

然后是有时候SQL语句会写出语法问题，因为没装workbench，我用的是线上SQL语法检查器<https://www.coderstool.com/sql-syntax-checker>。