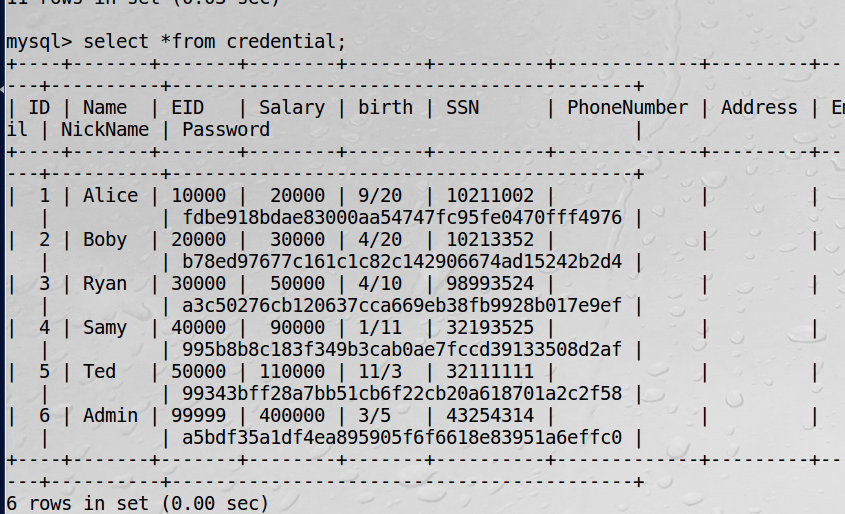
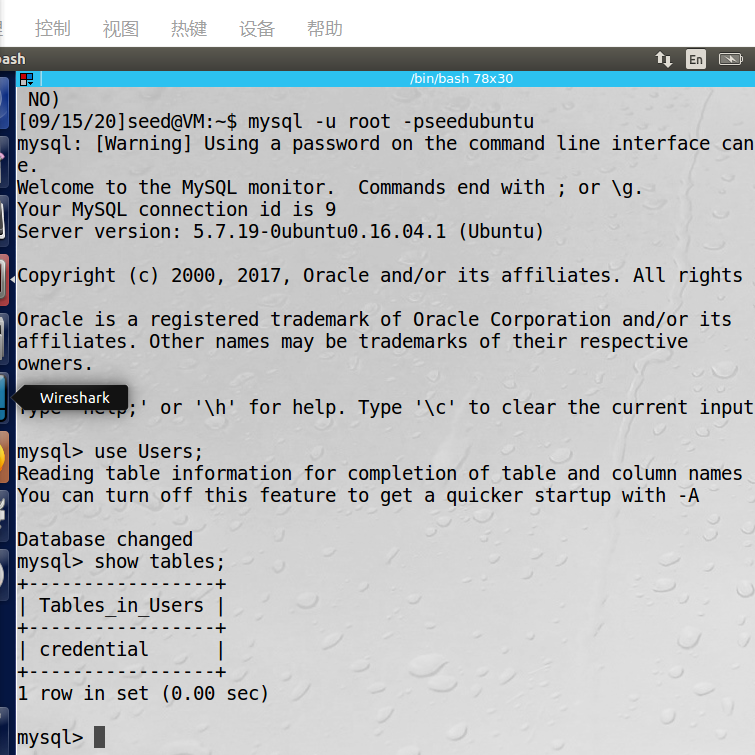
SQL Injection Attack Lab

57118138 李嘉怡

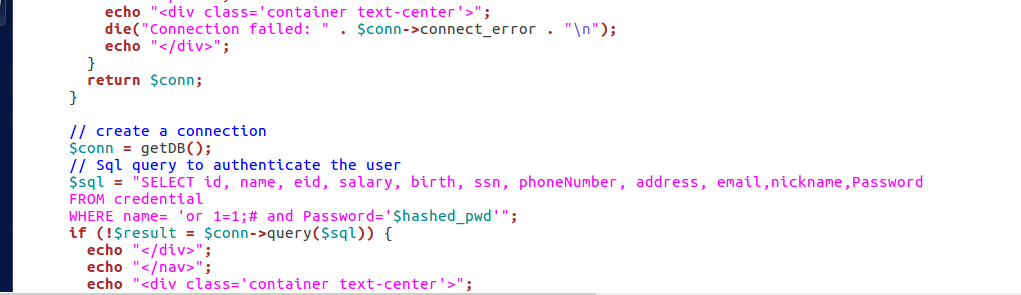
Task 1: Get Familiar with SQL Statements

输入指令，能够显示数据库内容。

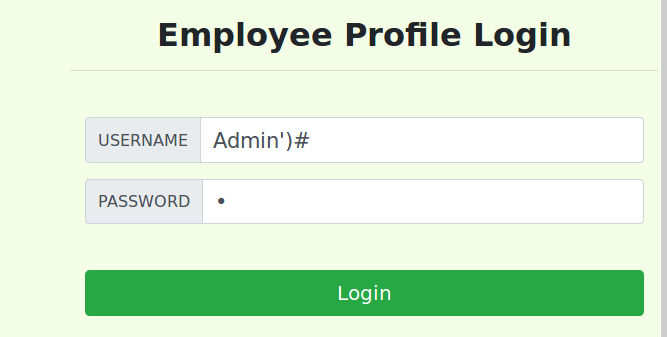


Task 2: SQL Injection Attack on SELECT Statement

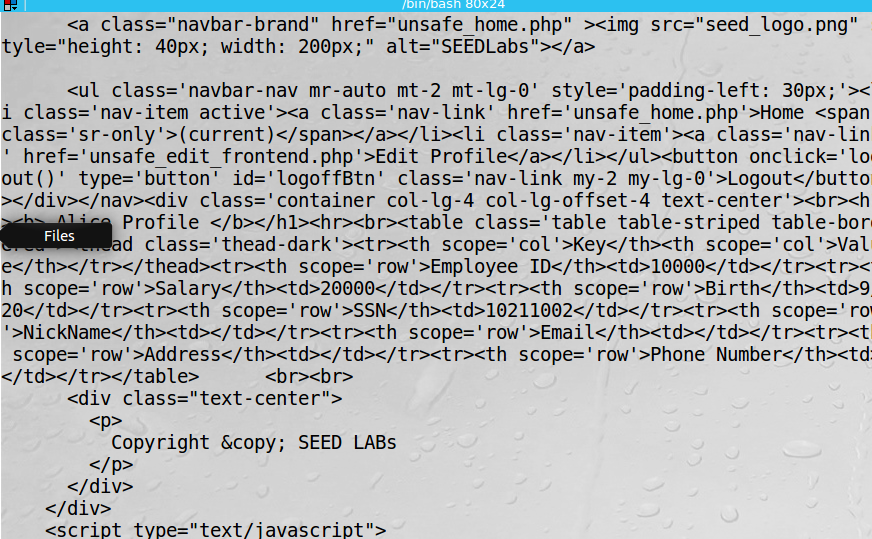
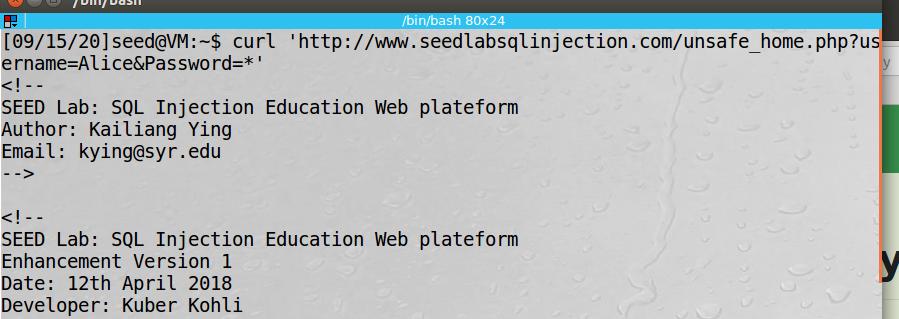
1.查看登陆验证文件：/var/www/SQLInjection/unsafe\_home.php，修改验证密码的语句（下图框中）如下图所示，在$user后面加上)#,之后重启apache，这样就会只验证用户名。因为#后面的内容会被#注释，相当于少了一个WHERE条件。



输入用户名+”)#”，密码随便输入，就可以不做密码的验证直接登录。

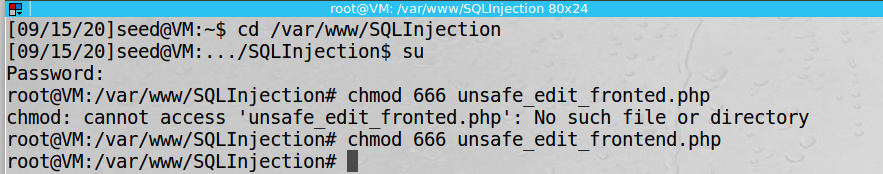


2.使用命令行用curl发送http请求，在不使用网页的情况下完成。发现随意输入一个密码之后，网页能在终端显示出来。



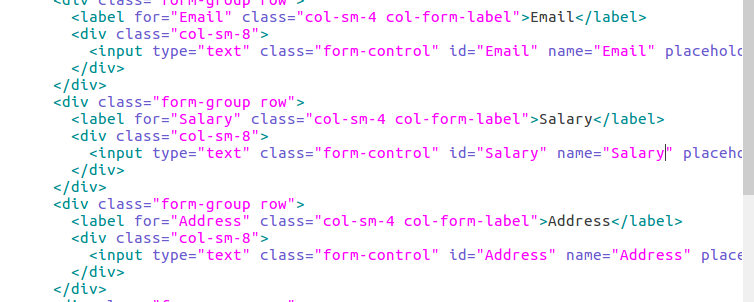
Task 3: SQL Injection Attack on UPDATE Statement

1. 打开/var/www/SQLInjection/unsafe\_edit\_frontend.php文件，这个文件是用来控制编辑信息的。事先应修改其权限使这个文件可以被修改。

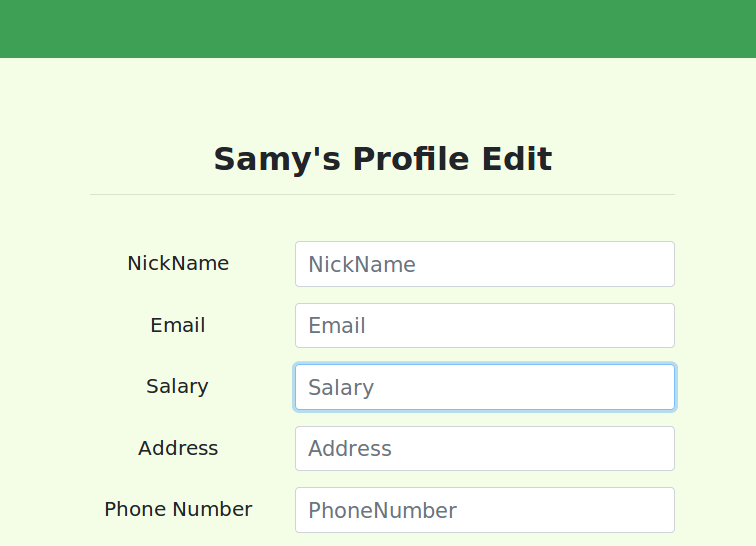


在其中添加salary这一项，保存。

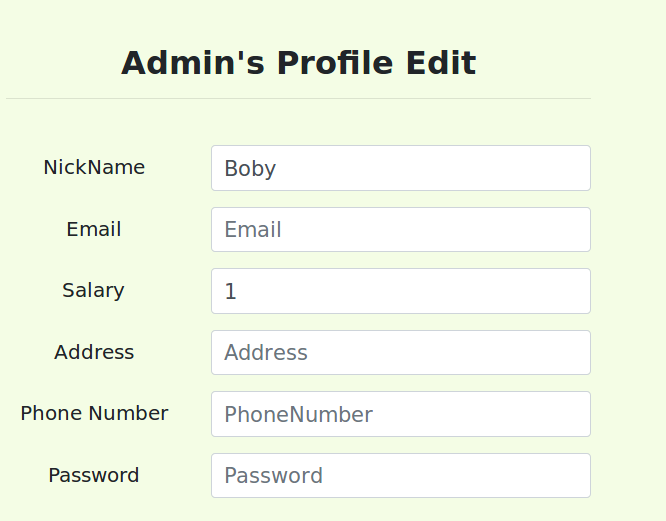


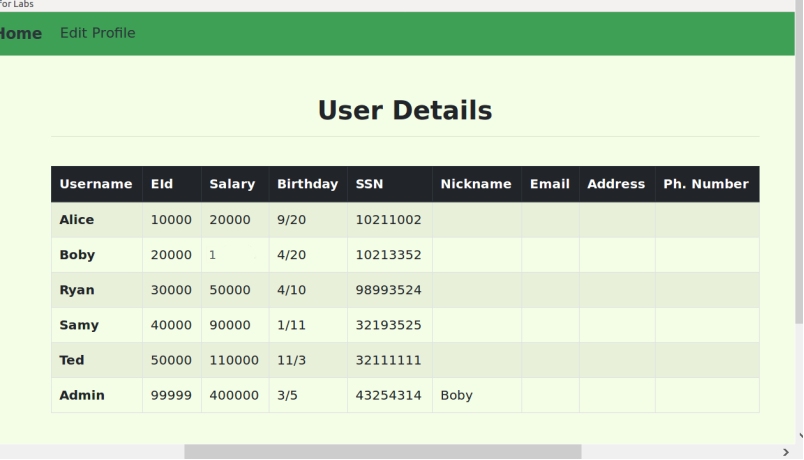


再次打开网页时，salary就可以被编辑了。

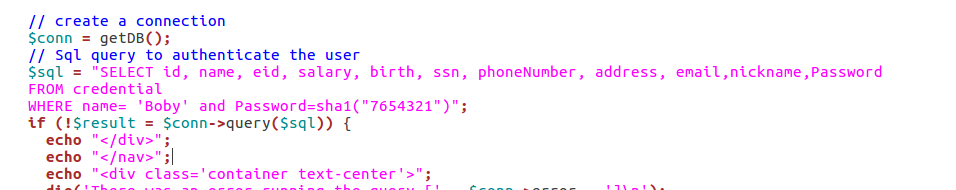


1. 运用task1的方法登录admin，以admin身份edit profile，将Boby的salary改为1。



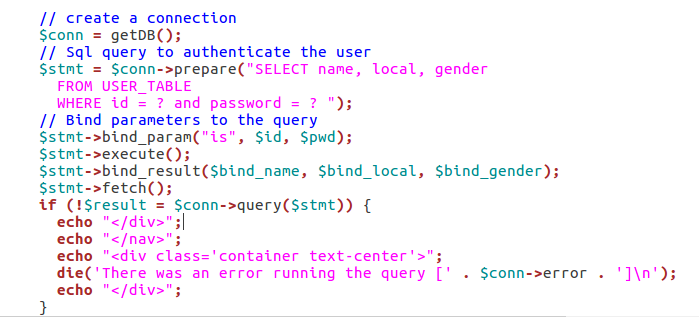


3.与task1类似，SELECT WHERE处存在sql注入漏洞，在”name”处填Boby，”Password”填7654321的sha1值，这样就把Boby的密码改成了7654321。



Task 4: Countermeasure — Prepared Statement

将unsafe\_edit\_frontend.php文件中select代码使用预备语句重写（如下图所示），并保存。使用预备语句机制，将向数据库发送SQL语句的过程分为两个步骤：第一步是只发送代码部分，即不发送实际数据的SQL语句，实际数据被问号(?)所替代；在这一步之后，使用bind param()将数据发送到数据库，数据库将只将此步骤中发送的所有内容视为数据，而不再是代码，它将数据绑定到预备语句的相应问号。



再次执行修改自己salary的攻击：发现修改没能成功。

