**Lab6实验报告**

**57118111 王雯**

1. **实验目的**

SQL注入是一种利用web之间接口漏洞的代码注入技术，应用程序和数据库服务器。当用户的输入没有被正确地检查时，漏洞就会出现在被发送到后端数据库服务器之前，在web应用程序中。

许多web应用程序接受用户的输入，然后使用这些输入构造SQL查询，因此

他们可以从数据库中获取信息。Web应用程序也使用SQL查询来存储信息

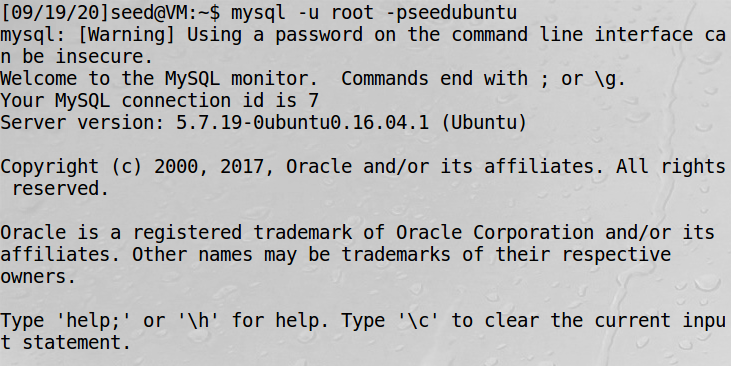
数据库。这些是web应用程序开发中的常见实践。当SQL查询是如果构造不当，可能会出现SQL注入漏洞。SQL注入是最常见的一种对web应用程序的攻击。

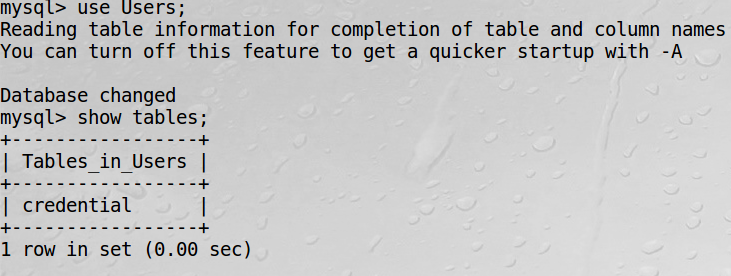
在这个实验室中，我们创建了一个容易受到SQL注入攻击的web应用程序。我们的网站应用程序包括许多web开发人员经常犯的错误。学生的目标是找到方法

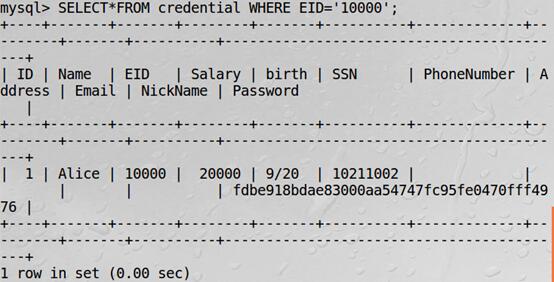
利用SQL注入漏洞，演示攻击可能造成的损害，以及掌握可以帮助防御此类攻击的技术。

**二、实验任务**

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



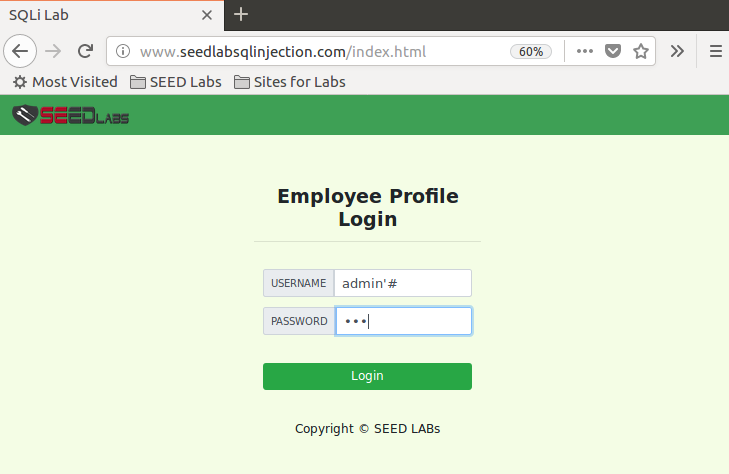


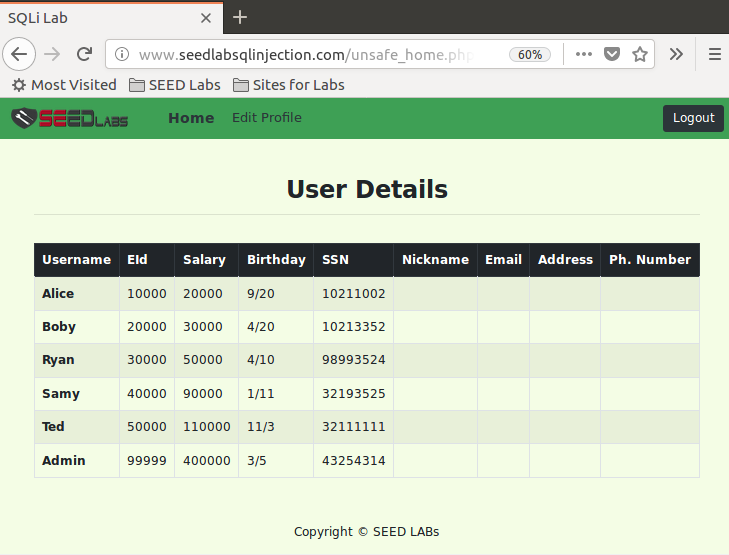


**Task 2: SQL Injection Attack on SELECT Statement**

**Task 2.1: SQL Injection Attack from webpage.**

我们输入用管理员身份登录且不知道管理员密码时，我们可以通过输入用户名admin’#，密码随便输入xyz，这样SQL注入的语句就会从#开始的所有内容被视为注释，这样我们就可以成功登录以管理员身份查看所有信息

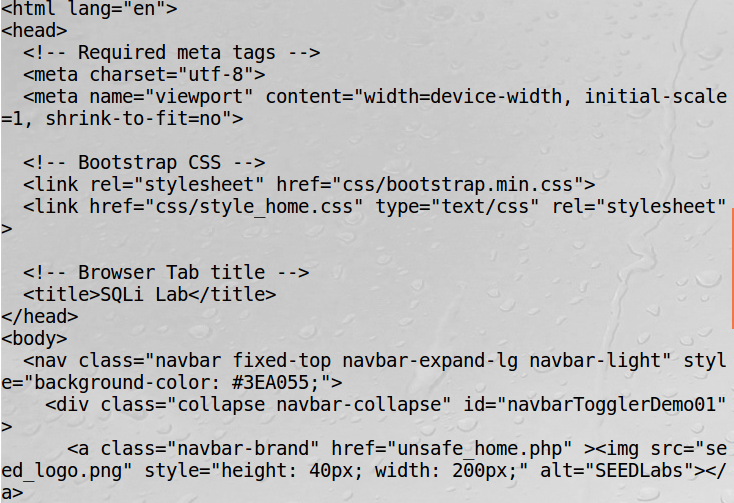




**Task 2.2: SQL Injection Attack from command line.**

<http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23&Password=xyz>



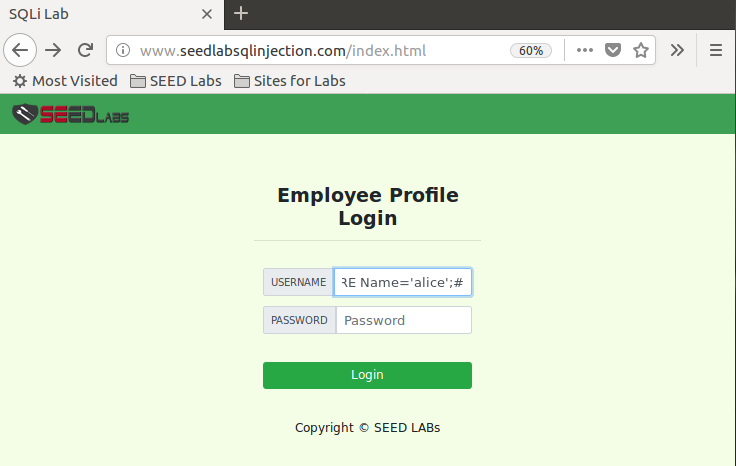


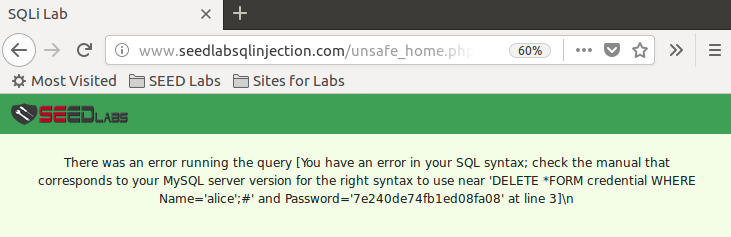


**Task 2.3: Append a new SQL statement.**

在以上两种攻击中，我们只能窃取信息从数据库中;如果我们可以修改数据库使用相同的漏洞在登录页面。方法是使用SQL注入攻击将一条SQL语句转换为两条SQL语句，第二个是update或delete语句。在SQL中，分号(;)用于分隔两个SQL语句。

admin’;DELETE \* FROM credential WHERE Name=’alice’;#





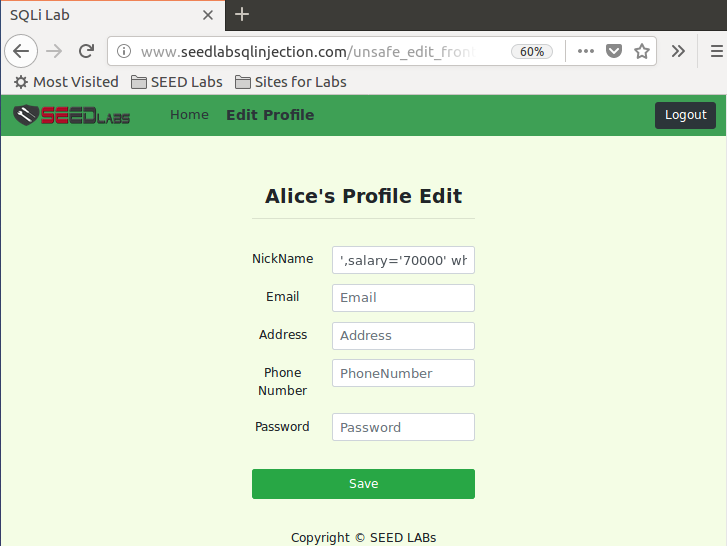
会发现多句的SQL攻击失败，因为上述代码试图通过$mysqli->query()函数执行两条SQL语句，这种攻击对MYSQL无效，因为mysql的query()函数不允许在数据库服务器上运行多条语句。这是一种SQL注入攻击的防护措施。

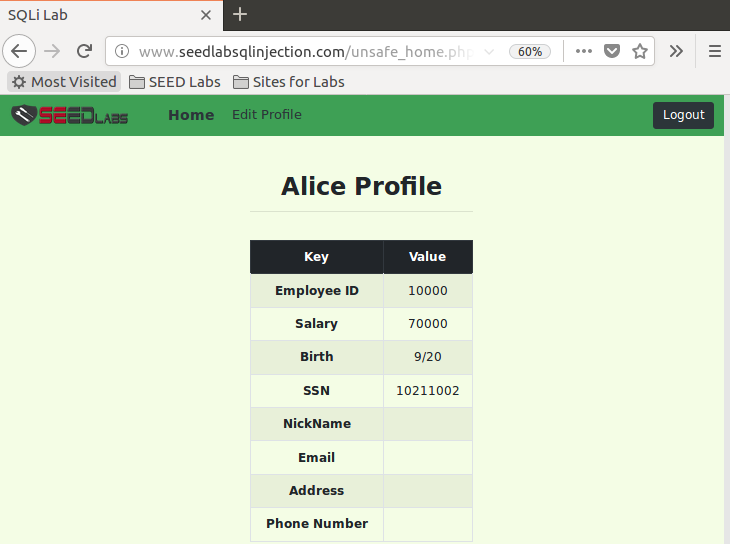
**Task 3: SQL Injection Attack on UPDATE Statement**

**Task 3.1: Modify your own salary**.

如果UPDATE语句出现SQL注入漏洞，则损害会更严重，因为攻击者可以利用该漏洞修改数据库。在我们的员工管理应用程序中，有一个Edit Profile页面(图2)，允许员工更新他们的Profile信息，包括昵称，电子邮件，地址，电话号码和密码。要进入这个页面，员工需要先登录。当员工通过Edit Profile页面更新他们的信息时，将更新以下SQL查询将被执行。PHP文件中实现的PHP代码用于更新员工的个人信息。

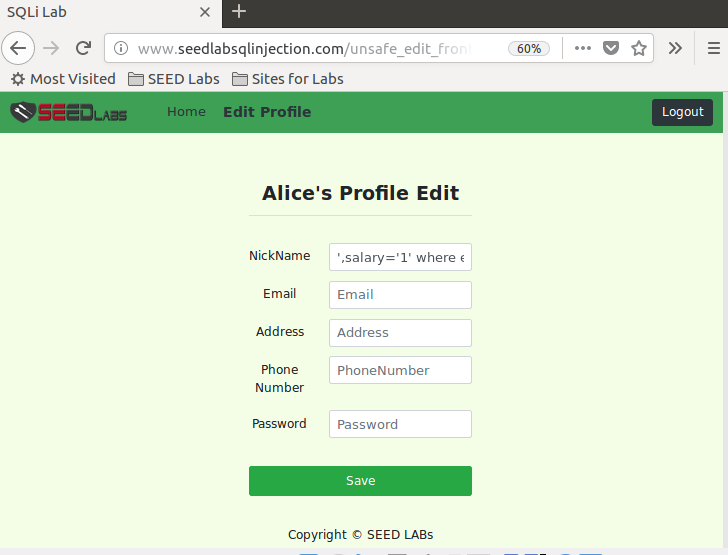
要求Alice修改个人的工资，则我们在edit Profile的界面进行注入攻击，攻击语句为',salary='70000' where eid='10000'#

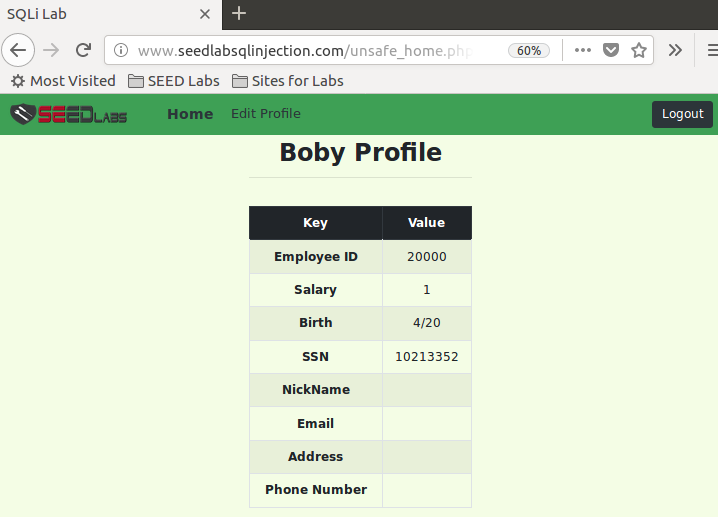




**Task 3.2: Modify other people’ salary.**

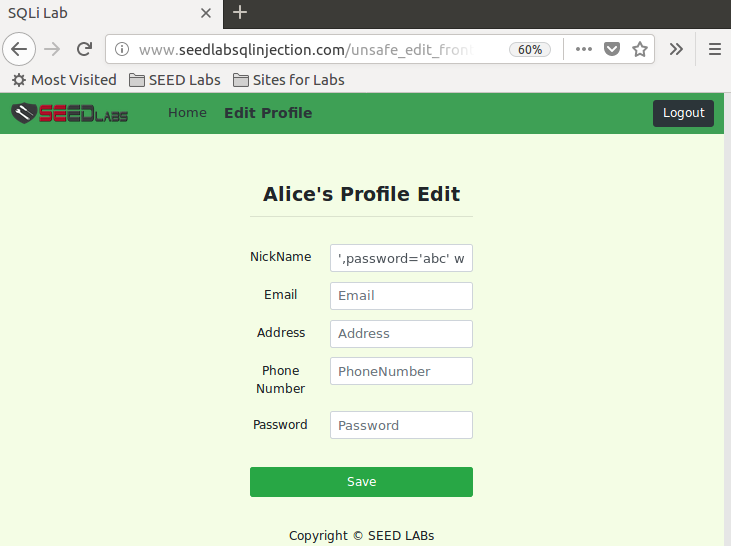
要求修改boby的工资为1，则SQL注入语句为',salary='1' where eid='20000'#，发现攻击成功

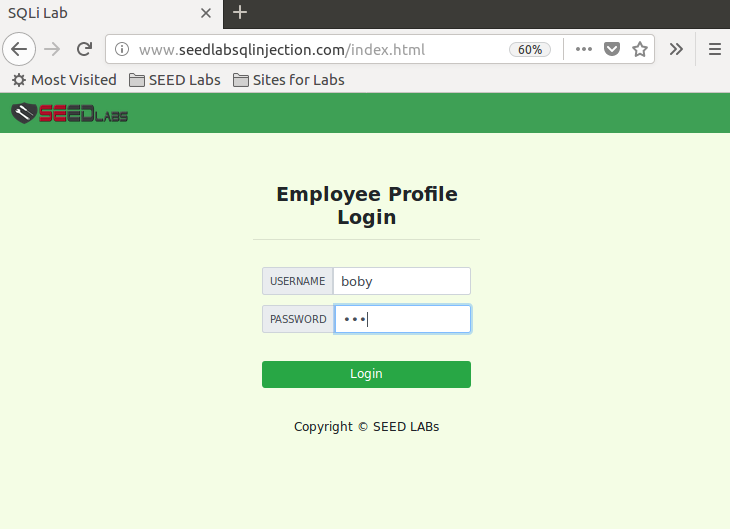




**Task 3.3: Modify other people’ password.**

Alice在edit profile界面对boby的登录密码进行修改，SQL注入语句为',password='abc' where eid='20000';#，我们再次登入boby的网页，使用abc的密码时，可以登录进去，发现boby的密码已经被修改。



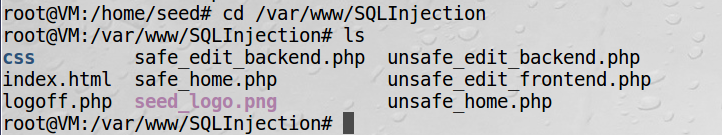


**Task 4: Countermeasure — Prepared Statement**

SQL注入漏洞的根本问题是无法将代码与数据分离。当在构造SQL语句时，程序(例如PHP程序)知道哪一部分是数据，哪一部分是数据是代码。不幸的是，当SQL语句被发送到数据库时，边界消失了;的SQL解释器看到的边界可能不同于由开发人员。要解决这个问题，重要的是确保边界的视图在服务器端代码和数据库中。最安全的方法是使用预备语句。

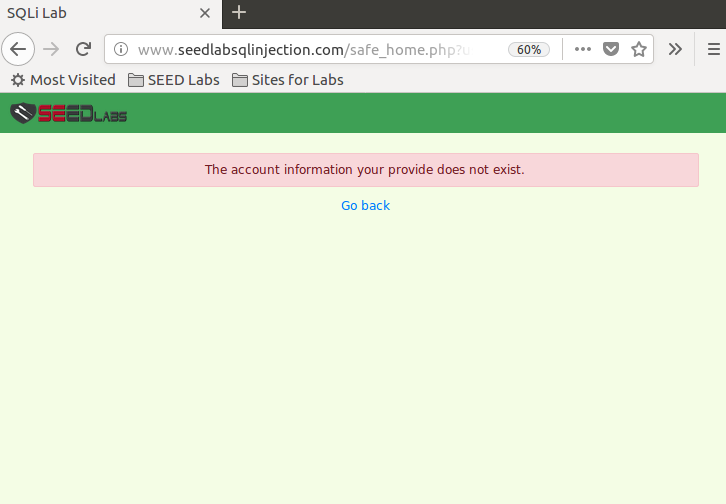
在编译步骤中，查询首先经过解析和规范化阶段，其中查询将根据语法和语义进行检查。下一个阶段是编译阶段，其中的关键字(例如。将SELECT、FROM、UPDATE等转换为机器可以理解的格式。基本上,在这个阶段，查询被解释。在查询优化阶段，考虑不同计划的数量执行查询，从中选择最佳优化的计划。选择的计划存储在缓存中，因此每当下一个查询出现时，它将根据缓存中的内容进行检查;如果它已经存在在缓存中，解析、编译和查询优化阶段将被跳过。编译后的查询然后被传递到实际执行阶段。准备语句出现在编译之后，但在执行步骤之前。一个准备语句将经过编译步骤，并转换为一个预编译的查询占位符的数据。要运行这个预编译的查询，需要提供数据，但这些数据不会被删除

进入终端，进入INJection文件夹，发现有一个已经编译的版本。



使用已经编译预处理过的登录

http://www.seedlabsqlinjection.com/safe\_home.php?username=admin%27#&Password=



**三、实验总结**

通过这次实验，对SQL注入攻击有了更为深刻的了解，以及了解了SQL攻击语句的应用格式，在这次的实验中，在完成实验的同时也遇到了一些问题，在SQL攻击中，在Alice进行工资修改的注入攻击中，第一次没有加入where语句指定特定的用户，而导致所有用户的工资都被修改。因而了解到，要完成指定用户的工资修改，一定不能忘记加where语句。