

# SQL 注入攻击实验

57118109 徐一鸣

## Task 1: Get Familiar with SQL Statements

虚拟机中已有一个名为 Users 的数据库，其中包含一个名为 credential 的表，该表存储了每个员工的个人信息(如 eid、密码、工资、ssn 等)，这个任务的目标就是通过使用该数据库熟悉 SQL 命令。

步骤 1: MySQL 是一个开源的关系数据库管理系统，SEEDUbuntu 的 VM 映像中已经设置了 MySQL，用户名是 root，密码是 seedubuntu，使用以下命令登录到 MySQL 控制台。

```
[09/19/20]seed@VM:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can
be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.
```

步骤 2: 登录后可以创建新的数据库或加载现有的数据库，因为已经创建了 Users 数据库，所以只需要使用以下命令加载这个现有的数据库。

```
mysql> use Users
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

步骤 3: 要显示用户数据库中有哪些表，可以使用以下命令打印所选数据库的所有表。

```
mysql> show tables;
+-----+
| Tables_in_Users |
+-----+
| credential      |
+-----+
1 row in set (0.00 sec)
```

步骤 4: 在运行上述命令之后，使用 SQL 命令打印 Alice 雇员的所有配置文件信息。

```
mysql> SELECT*FROM credential WHERE EID='10000';
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	20000	9/20	10211002					fdb918bdae83000aa54747fc95fe0470fff4976

```
1 row in set (0.00 sec)
```

## Task 2: SQL Injection Attack on SELECT Statement

SQL 注入是一种技术，通过它攻击者可以执行他们的恶意 SQL 语句（通常称为恶意负载），以窃取受害者数据库中的信息，甚至可能更改数据库，我们的员工管理 web 应用程序有 SQL 注入漏洞，模仿了开发人员经常犯的错误。

我们将使用 [www.SEEDLabSQLInjection.com](http://www.SEEDLabSQLInjection.com) 的登录页面来完成这个任务，它要求用户提供用户名和密码，web 应用程序根据这两部分数据对用户进行身份验证，因此只有知道自己密码的员工才允许登录。作为攻击者，这次的任务是在不知道任何员工密码的情况下登录到 web 应用程序。

### 2.1 网页 SQL 注入攻击。

作为管理员从登录页面登录到 web 应用程序，这样就可以看到所有员工的信息，我们假设知道管理员的帐户名，即 admin，但不知道密码，您需要决定在用户名和密码字段中输入什么才能成功地进行攻击。

在登录界面的用户名处输入 admin' #, 即可登录，登录后即可查看用户信息。

## Employee Profile Login

---

USERNAME

PASSWORD

Login

## User Details

Username	EId	Salary	Birthday	SSN	N
Alice	10000	20000	9/20	10211002	
Boby	20000	30000	4/20	10213352	
Ryan	30000	50000	4/10	98993524	

### 2.2: 命令行 SQL 注入攻击

重复任务 2.1，但需要在不使用网页的情况下完成。可以使用命令行工具，比如 curl，它可以发送 HTTP 请求。如果想在 HTTP 请求中包含多个参数，需要把 URL 和参数放在一对单引号之间，否则用于分隔参数的特殊字符(如&)将由 shell 程序解释，从而改变命令的含义。

```
[09/19/20]seed@VM:~$ curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=admin&Password=111'
```

```
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

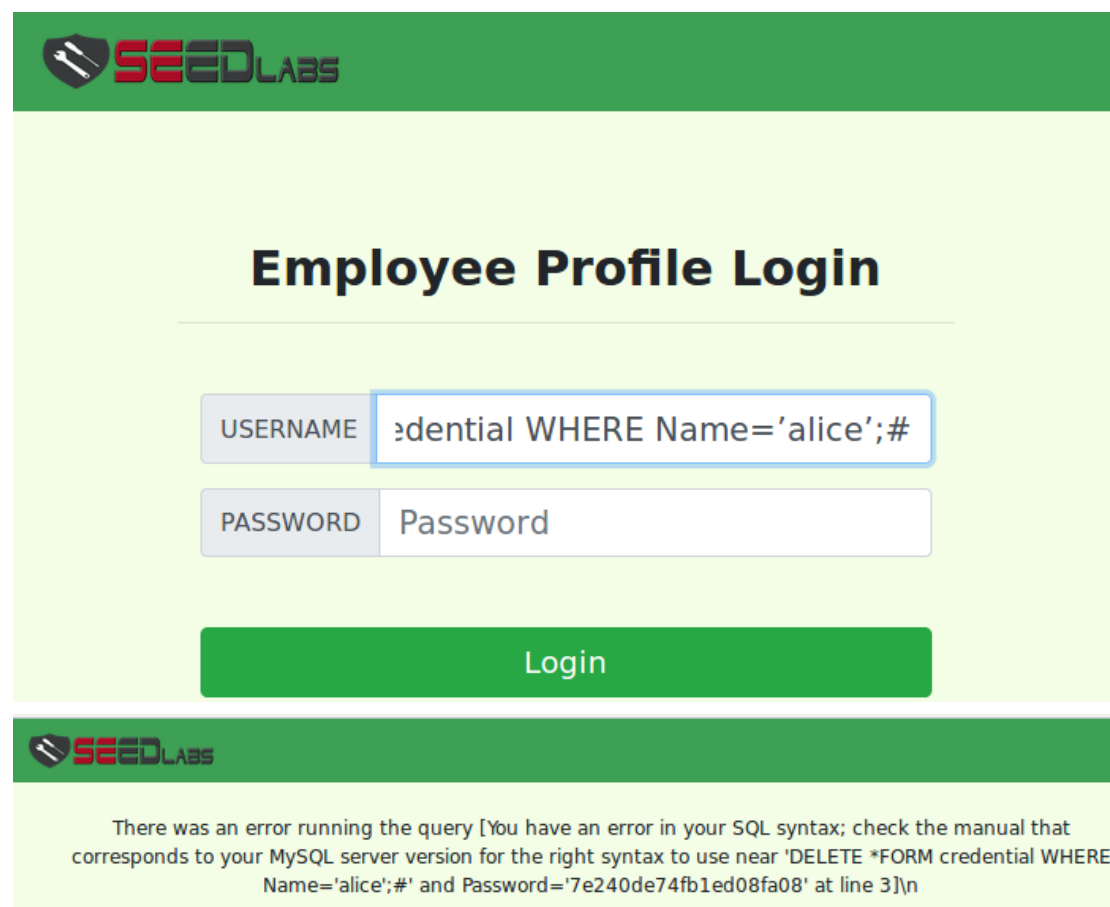
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet"
>

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01"
  >
    <a class="navbar-brand" href="unsafe_home.php" ></a>
```

### 2.3:追加一条新的 SQL 语句

在以上两种攻击中，我们只能从数据库中窃取信息，如果我们能在登录页面上使用相同的漏洞来修改数据库，那就更好了。一种想法是使用 SQL 注入攻击将一条 SQL 语句转换为两条，而第二条是 update 或 delete 语句。在 SQL 中，分号(;)用于分隔两个 SQL 语句，请描述如何使用登录页面让服务器运行两条 SQL 语句，尝试从数据库中删除一条记录，并描述观察结果。

在登录界面的用户名处输入 `admin';DELETE * FROM credential WHERE Name='alice';#`，即删除该记录，删除后发现攻击失败，因其试图通过 `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。



The screenshot shows a web application interface for 'Employee Profile Login'. At the top is a green header with the 'SEEDLABS' logo. The main content area has a light green background. The title 'Employee Profile Login' is centered. Below it are two input fields: 'USERNAME' and 'PASSWORD'. The 'USERNAME' field contains the text `credential WHERE Name='alice';#`. The 'PASSWORD' field contains the text 'Password'. Below the input fields is a green 'Login' button. At the bottom of the page, there is a green footer with the 'SEEDLABS' logo. Below the footer, a message box displays an error: 'There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DELETE \*FORM credential WHERE Name='alice';#' and Password='7e240de74fb1ed08fa08' at line 3]\n

## Task 3: SQL Injection Attack on UPDATE Statement

如果 UPDATE 语句出现 SQL 注入漏洞，则破坏会更严重，因为攻击者可以利用该漏洞修改数据库，在员工管理应用程序中，有一个编辑配置文件页面，允许员工更新他们的配置文件信息，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，将执行 SQL update 查询，PHP 文件中实现的 PHP 代码用于更新员工的配置文件信息，PHP 文件位于 `/var/www/ sqlinject` 目录中。

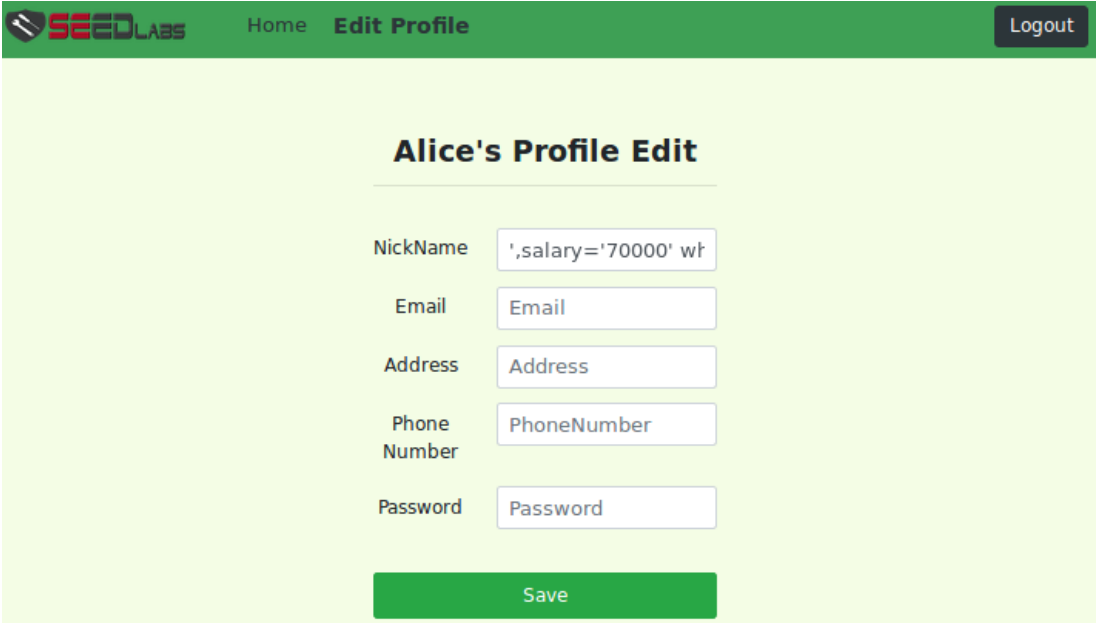
### 3.1:修改自己的工资

如 Edit Profile 页面所示，员工只能更新昵称、电子邮件、地址、电话号码和密码，



他们无权改变工资。假设你 (Alice) 是一名心怀不满的员工，而你的老板 Bobby 今年没有给你加薪，希望利用编辑-配置文件页面中的 SQL 注入漏洞来增加自己的收入，我们假设知道工资存储在一个名为 “salary” 的列中。

在 edit Profile 界面 Nickname 处输入 `',salary='70000' where eid='10000' #`，登录 Alice 账号后发现 Alice 工资被修改至 70000。



**Alice's Profile Edit**

NickName: ',salary='70000' where eid='10000' #'

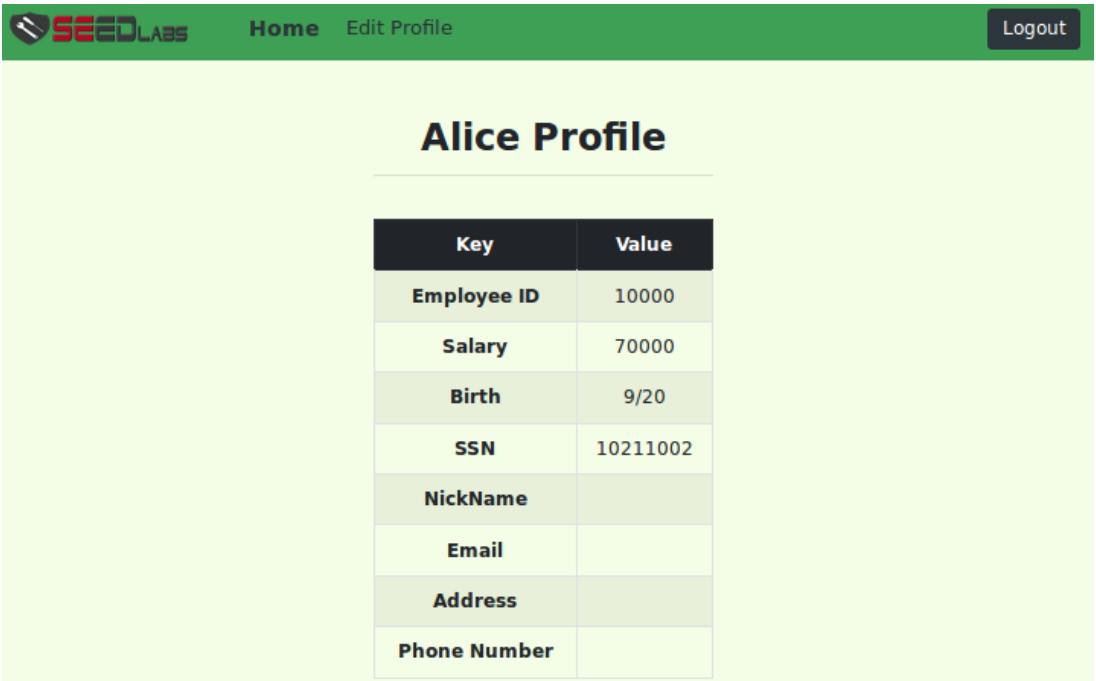
Email:

Address:

Phone Number:

Password:

Save




**Alice Profile**

Key	Value
Employee ID	10000
Salary	70000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

### 3.2: 修改其他人的工资

给自己加薪后，你决定惩罚你的老板，把他的薪水降到 1 美元。

在 edit Profile 界面 Nickname 处输入 `',salary='1' where eid='20000' #`，登录 Bobby 账号后发现 Bobby 工资被修改至 1。

 Home Edit Profile Logout

### Alice's Profile Edit

NickName',salary='1' where €


EmailEmail

AddressAddress

Phone NumberPhoneNumber

PasswordPassword

Save

 Home Edit Profile Logout


### Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

### 3.3:修改他人的密码

在更改了 Boby 的工资后，你仍然不满意，想把 Boby 的密码改成你知道的密码，然后可以登录他的账户，做进一步的破坏。

在 edit Profile 界面 Nickname 处输入',password=' abc' where eid=' 20000' ;#，登入 boby 账号后，使用 abc 即可登录，密码修改成功。

 [Home](#) [Edit Profile](#) [Logout](#)

### Alice's Profile Edit


NickName

Email

Address

Phone Number

Password



### Employee Profile Login

USERNAME

PASSWORD

Login

## 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，在构造 SQL 语句时，程序(例如 PHP 程序)知道哪一部分是数据，哪一部分是代码。但当 SQL 语句被发送到数据库时，边界消失了，SQL 解释器看到的边界可能与开发人员设置的原始边界不同。要解决这个问题，务必确保边界视图在服务器端代码和数据库中是一致的，最安全的方法是使用预备语句。

要理解准备语句如何防止 SQL 注入，我们需要理解当 SQL server 接收到查询时发生了什么，查询首先要经过解析和规范化阶段，在这个阶段将根据语法和语义检查。下一个阶段是编译阶段，其中关键字(例如 SELECT、FROM、UPDATE 等)被转换为机器可以理解的格式。

打开SQLInjection文件夹,发现 safe\_home.php,使用代码和数据分离的 safe\_home.php [http://www.seedlabsqlinjection.com/safe\\_home.php?username=admin%27#&Password=](http://www.seedlabsqlinjection.com/safe_home.php?username=admin%27#&Password=)，发现登录不成功，防守攻击成功。

```
root@VM:/home/seed# cd /var/www/SQLInjection
root@VM:/var/www/SQLInjection# ls
css          safe_edit_backend.php  unsafe_edit_backend.php
index.html   safe_home.php          unsafe_edit_frontend.php
logoff.php   seed_logo.png          unsafe_home.php
```



The account information your provide does not exist.

[Go back](#)

## 实验总结

在本次实验中，我们在先前的基础上，了解了 SQL 攻击的原理和方法，同时了解了防御 SQL 攻击的原理：将代码和数据分离，为今后的账号安全防御打下了基础。