HSCCTF2024 Writeup

Misc

SIGN_IN

拿到一张图片，foremost出另一张图片，上面写着HAPPY_NEW_YEAR

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# foremost happy.jpg
Processing: happy.jpg
|*|
```

试了一下不是flag，那么多半是某种隐写的密码，试了一圈，是outguess

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# outguess -r happy.jpg flag.txt -k HAPPY_NEW_YEAR
Reading happy.jpg....
Extracting usable bits:   10517 bits
Steg retrieve: seed: 211, len: 26
```

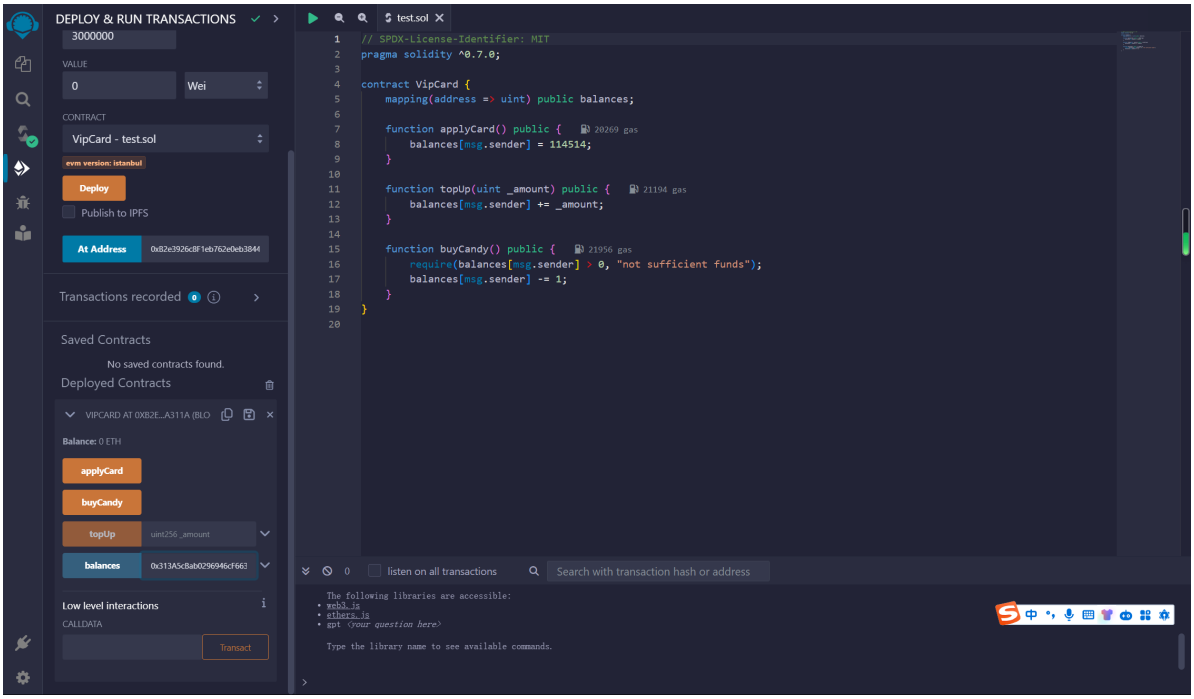
BLOCKCHAIN

BONUS

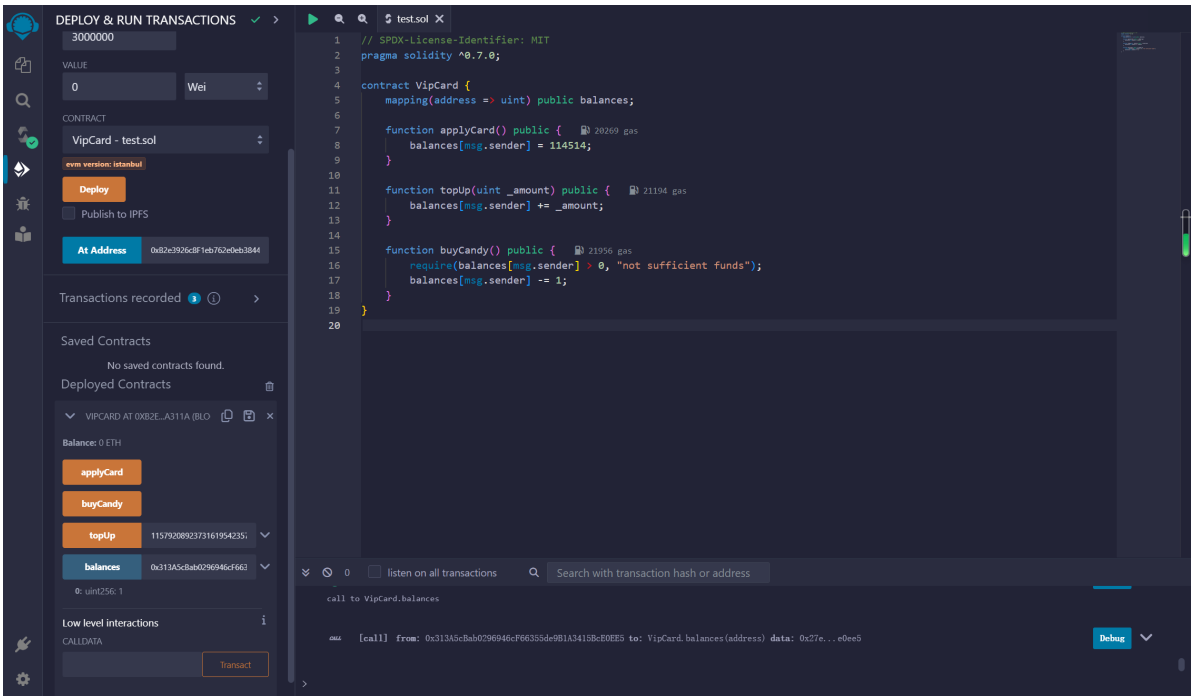
怎么连接到Polygon Mumbai网络看这篇文章https://revoke.cash/zh/learn/wallets/add-network/polygon-mumbai

记得去官方的水龙头接水https://faucet.polygon.technology/

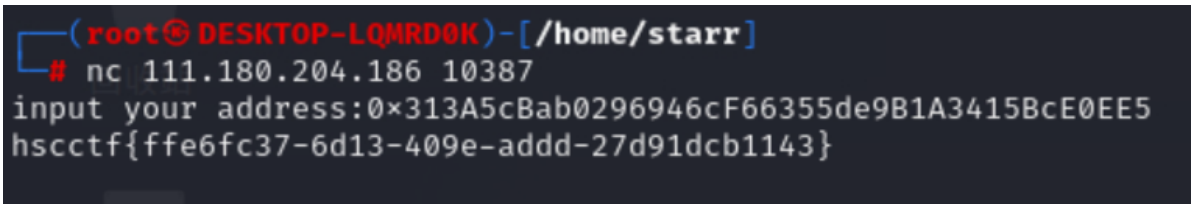
如何在remixide上面将合约部署上链不再赘述



先applyCard生成114514的balance余额。这里每点一次buyCandy，balance就-1，正常来说需要点114513次才能达到要求的1.然而我们注意到，这里的数据类型是uint且没有对topUp的amount进行限制，因此存在数据溢出的情况。uint上限2^256-1，因此，只要topUp 2^256-114513即可溢出至1



nc得到flag



Crypto

FUNNY

直接运行一下就行

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


p =
162785872174359675249783164062835732345924389084436379273640778975157742731925770850768611445663293818377512958572889284520529086475057879835485993399620852145012913117335249011808593996938591130671812978503510805690913593132275590860265656722399885845013076862558755993506475360371865798953379337312492289107784937644758919823912629112726323478436355677778046148938770669905367131562196880706536090766781533660987885679838705789197473046040464722930810544262903806128406027664899343468111453995508682845108558203328886095972975585647806509332754421781399066532682015634046132653827190960571582248495019603380989223212133062302785597047588820258351223421334946268045240072990340049451943159043186471722343880709366983661535685507237057159588980771216665906119003145795407114664541127770223084738926498939171576323245360540409501940218633163320295369305764147530061948286326593651339434106677661933645292395143971474906734850 7
q =
165059760196977338034268528857897083190717585647808271832795575354736221674892279730906525407442580833042731114617534865267230347093979480666213073939602212663043286794259410177026619955285626139270678815060566416554620185859901611571955490060241822054715533030464833509114500388615748465579240918292431860839
c =
209347444484196477254668267578954904181293909211985372842733666165076889646025992007884927283300662833989678431462547742385291078285045060028956748455410812196305788814280968487973559115983082461315009316488143477261307107939530267533197821947240438534411830018967506368459544025037634267899970815198992392565948727381322854560823769432552779881938316107779842525092269591932257206582810678457433166038809740255591841951300818106018061065936083887559841543250701094558114335651396741896281230175124133873529005816838785880680073159461666783649744640398010841282229366323100807220116460976549981263966530038289223576133865992894885483308732021144420507955669760623032114718333853671281281221009397028372653797668774820725717817362383394165014402977767667433542733963298092312930702496796235196487999167666133884266829605503363510544384236303968829214389757297563873395131798724306227675722494797069994655456021649231626359765579028950908065367760981736933167557841586896807116262630848742514005606289980676609588756946993271763119859088002784499062149533167332388600036550273743940503131248513795389688869470432261503038708892320267815658450917855334244502333938120781251564676408212996623419957317150267768978807968208353379334556945

d = inverse(65537, q - 1)
m = pow(c, d, q)
print(long_to_bytes(m))
```

```
b'HSCCTF{xin_nian_hao_ya}'
```

real_sign_in

> 前阵子才刷到过这样的题，之前写的脚本直接拿过来用了(xD

题目

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

flag = 'HSCCTF{************************}'
m = bytes_to_long(flag.encode())
e = 65537

for i in range(1, 11):
    p = getPrime(1024)
    q = getPrime(1024)
    n = p * q
    c = powmod(m, e, n)

    print("n", i, "=", n)
    print("c", i, "=", c)

'''
n1 =
13266094979118433615107077438835111371457503704538616496563510289250600197984029
76795095307172344238127990474772426970655580142588311704097283203539726834760361
20257412194523531302667551509650194410923800302262560327206518586001729963445983
55816134478110939124633509594465443169697188695226358357129879780253706604232278
67846568035801826208419810853307751904950781175842349851256552161593169949449159
50787230981010842176112044425851821032610060861443446123988965760669999509059669
20722348056083426326740287336766359383868804991393096926457607860171709146742941
74086478061022646385201820464709629273791576043029125081 3
c1 =
10454892625278317641789872724686322059272523357918030709377842011690595796341085
03308033729970364983743915474860354063759051514315619953792921367621385574749886
77789114718390798158209224302630161401413640941314968851558824867983024042385239
86615316359385759537428069831379320707742928167131988641527992674882901567191207
76354743229953271094554995958067011234683091950069401497244480390424858617888747
04658679916504800249043274327179113138677131205252026691015247788264455033835497
33326775312427637839488477432162027468606568090607495541492335140760946449358303
99921827726985923088829278023427549527478799600106995997 1
n2 =
13561524319281645924409429988746244285924512610515302181526524036825687984904507
67789461135334473574111053328054037273417896321687694506623292779136920785276214
68268839060938049587232033090308408048979992050744563250627891104529404840774085
80876939074453733126571662068654441714639301799127584606087971553812926212292460
76191598578646180764954628633083471974662438785083385925071779049267188473439305
54540747195859379042325166595827362119332800115916050696867910496283967626900383
33975652490908544846890333889140489210242676577675968427325105786641017561929481
33559768169381843589214085695967375421360028244476104590 9
```

```
c2 =
38099373966389737932811111508377746077688365612564254900710618145433152171826895
79329457784159377040740840790603579270672790822179042079055887400637582022659400
59484442589166649375946952222969226710351629592358966615194894753300297550943050
57780812642686982926062095380472408297569197649585445183119936847901181221638595
71923899116815269989298822142104789156965964388346807167061332155392766290531602
74773191601077431840053353641392587655432455078013619145877662302923180299131949
83462434272464844393919406766406936165197889502501946204553315574355677428740726
3700517175455002061936115015076080739130737749383670307 6
n3 =
20819967432768567938753694422911303382713641435728476601586838051751312782014589
68787410068344701654989385382039924790965351995099516680507238065665209357960480
85694433457001317569018139681857268398634928592315698967479875313410385275770302
74701158504736611486411122416871876820360598851918276248520018793494992493852386
39993591909818529926892068846544625628333391899339216394755947088707448811751096
11722783300260508822498975269269512469153788419617432614510530949127308507542429
34166148766977209435597121171360066632973035097557482087715745940551760350028517
91240279535392967877038348446233783976240438448758206645 3
c3 =
11563829913521314742585136805475993461403465700108053759771484386843765447294825
81259574784456487480182154738594293901771104379259042726624013302783146197492755
86526650672138786835776058923199497040191937522504098138689064217532866905586466
31157080887991363101311595921894128110588310204725909211948500421818595333014728
11492261287023679244518075791803458021038334820517467749278403486370144709721948
01330950059540434906313393612804938100833097039406143715013696474466720521597149
63350601039281652141446041256158792981347520865671115091097366786582678000769990
98948587975656297182518158385205219449296689025878694431 5
n4 =
19517605474769458566764053169103203900538883167125789442948194023333599294853531
27308738656316161374874381834394817076552580204131336909452570803424049203851749
10376974210196986729400280614893770785690272049405257417516077587068377892765718
89269840463242449627036923444536643477208091664006240106337894541529098279162689
79363365915513118332367588323321467409885755348739084401548332671176609082636463
97910723151480154504915094117055434197116328942821056984106294631410480898670161
93055384530738113369352506445669521342292860468297676826301170714618519525841753
64271326708300754189415652281953693512913548828919869488 9
c4 =
12720867554077215609167247949706745145667570273372296302246080668969511222297653
72246245623308826215356230104488540953881693128141468757229690797000444118587122
97863064195868390509613630655879251886233094376878783623497120108614194094910684
81019724226070296781806932012179356550421791040143093709543578156689726256993615
53367923861978408761596741961637673687082489769586373226059560890589882107425028
59039896683032908340030986919029902655266957778994899114268559578445619617290636
44070823142386451495647995297413399808476505524581260464059786283824394788336780
56770295005174336129413164828234332958770907710303531672 0
n5 =
15315765971812428868126611109300419975777573811870059860288938012581131143314666
28708947240872919030674035131692242624473258506737721750211644014545087566568357
54733081027420952443498042933110066643516469521812106868127489124181757360138356
69191549110546127830831456068811493086787520032272004231879188572130659936937680
17797705897569271611696165109303398607329704067870753430096572286826382544419867
92430705074132160872052939970068056363385675858157642333749177458091355688555249
93022583535608349280044851332779641619210310837108819273508771161881712796235518
24357254975627611805373627012278789917369572908991100017 7
```

```
c5 =
19965763259897683243178870126805603705289491946263936646501827547915786977863699
58145137743555674312911707989139883926276952465377328602829900403114611234863417
18370767839804703004685545007426065443227815690338864194869671619373471483995762
43595334868440231111152977183983191207182170437280645952176517381808061949125819
64346594873253580010279706758241793778842399330631789750327865945625117967957133
86363717640430978251688051174777179371220390318551228532710308013356766039192103
86813935519456071527260636926417422864067729261033240785741811766323212877590668
02783643421564667257668981655199672490410258387124896839
n6 =
22501623321194869030526666823514739767749207843970334295553523279818676682652976
95980772938164679267613498810106578313675990445218928881304086002149742942293557
10839931467941862078531464836810943753607747684066295883113813752063313689658367
25531623084019451921330442379292718421616088275750226791314378297932358311378533
56962067142374059803320463963403581468978173610419732134815499975921053265145133
61923765319758372847519786091970685732785512368967137626161060518825057522877732
03197087939983952212761603430314087275042009367400077728025285967811883773835952
54702541185339530732595990421425510011576337752776186324 3
c6 =
16972901653690784382193404778525843883847230468707217844241233207197480301434902
62901808874082288994897210244566803348918132467553153765078576658230711208372141
05068342793634878147590218150058399066567310695217586052934289283227128955104909
68448490116732590950772037788323801838566829526879837783860631971103738986788475
65218708093594442866236357309722532263057470597761614331549727275405060811744993
81617724099249924536403106218992236918513039213235377790683465759976210246609077
50345496836548887894110692437965970524341315015139872598871247336564148953981001
70814525372878979948375393103989494332445817220740851405 5
n7 =
20222972180177072345801227937253710431874005900913732040381986459681923475683145
37299037529635464742066748386645556397649249586330661364519935172532306925403391
46604448846818017015271285789148156472282654575801925914053586466050693620258870
82090772778349481109768859094222573588734354390770166651349176555216207359309348
01834250546424980460787311898468819064839260655083571782816031706260459707910152
95865928098466081535480906891144315698445193179690768521560234298099186181371539
01636552629572816865533796914796916912143735510069871682428043277595493567381686
08644048039857906380149054108720097522437204744766685 3577
c7 =
19248329020030453774384696880312801243022181399316695013193573740699955812434234
45344573316012840888931209077573404778287610014598818332788491721567938238288106
54198707641867219706222487137614948508630827273736732642519512508139331541614226
45331713437130381414540777017708015790836797014483890216068812718266206449620134
39284337003635231024738462406436664658093793833058777370011686937963058428503017 0
82057951745360474953560815331211198096212006775548028507413270268577664668018470
93397207492834507197621929666294565502206253758771807936447505283097006026148803
10376179979591784806195940765508401216691641801931224978 4
n8 =
28799545621025859784610107508239196961299975467825740831913751504915874851330462
43915863557540535681811341033368318008233216569712042711295510314874443942544518
38852004193239006492473909873736437325453514283416721326069906519832171364277797
27640093677134233750683573339125630816150512829880758197582740742903567508671599
67063844169006462497088484915507992426131937841246831520593795963807477901081169
51328636073141751996598214509838251737381227490430638215397066480797314595965931
71608003970504410762959266052601005965264083020025264976022719772719760327913949
45176554959665411277657903291556611200766816777280002 2899
```

c8 =
155791245370947696673707699156106991613202656362395221432589945068477961139600551152519442500237742408059354887171709334429815472241440266257934874981436237250255359687608792440551234613964213004496615091068462935503350591801340806543248968935953365382453182974919650335330342650645751987072957842151770478808383787123968446850664679103142911144902439611114592590027446273135286565743973465097184774897466555552505415511716343556722891604685949816750212973976859057217032619136254245934640514243676448023993012332317521948850118097382020003036538697926564209076122713466985311902796437751606445020367316999674110820 1
n9 =
107247226453042597157952998190656747540429840137862105845062313006195606761084755363952720465699737436412517577460611978285197803156184583771035630622311834834178130261649327428281602459362637176530187460017981066700747138264255387583275380776668184118035567903031187036752145702320323637412446521998778298509413569753496104332706025009496647376148292645707203769826976134979177734078998894724153987383687531503007822017016298402637910998987992239390972377377633057442445209138311513611559264620862523891965485832152752306526272057093629219892354872374378305194793088536390241849407679040194684050245672792691946604571 c9 =
705463612113234358385869930109575071411123581792011545946870788472501819962118526856546209953758372723112393566588030444115510698765583426436362250395222872052524217950807901663108956447269419692829296759200709052790183244098047860554527589329477590573220979646359710511667636350560764438429321090560904489885482334941228236466031666385339991490559120640881321780940909164095426401350314009757434019464208207922992311335823038396957805698822367795673327054284423260967631762122524511318733561339982590040303505605155884427567432712042870632816967182612225212150848091424300840160529323960243459920269872536666352976 n10 =
163048179013974834174566093415161245232324267800233972282729130447233653895713277971345917369775026920577274163949243108288243780248127775293673427078964948159602556829476850369022359978998862916336090416858971972726759113049415010677947952859495403139902264096512921201264927149988922900441115410983656927311857873606091989216817367771037944306536966981411959547190308839351519808473399392130403732528191456550672869863006416689947003909214586825716194094882332892117385210460853112531195266859128934581315347180508026841802851388092172742738773763470240210563175663121861339167445334638797799977945403617603932194891 c10 =
603912266080365615770005473906456433843954948469520621401600612950878236222984296760246407773587454010716145904183534099010881713967213787064293561353131358396393014487145873302639622367480155762103694931211663921744711489972685898428119411650824440398556878576527383857247089892552114702202151686265896758413545447435538107302606642645061478838100205798059044233876686910495757733854668775914122894214937448578589353933528853672932821817227837076318360342262979984661173771086568157225780344896065669250677308652738961617869315288526079530300811193527752956157691511416567856499792363490637705949198939946773806964 1
n11 =
266255915701089179918476240243258190203209892554836082306448794765179152119955428963414165156055769283923177176546957670067847447519622053551070128594261344740746161349864142480547817121308920764622887616726169932874742152710466024432154492100263039782497528253060350978414295677842656568908015894183154499275436601319673652766376625631123284278131833985179442101208518414950012935439717658643820601006001791088148528294738844818672727169070647299911877983287331966762986813283992503560281554118659487584663377630258866854366506434285105936971764167355682506156736428666684165266823417219984815024828779726490185014691

```
c11 =
20247005397625616228661928392695469647749434533141783786437968083066137357689767
28075353260589641917333259752970880194690636378059287611110485030278403313633545
66833611286036624547282343495632389488017129079031859935610767541645528455107952
93521029798357876214260903355702688269527465118808832941386890598851762900876512
38355765394084623288462755795765263858102214082537302317348247179172014879230124
15819197655102200830793653112970816339793988559669910690205844626139810948887369
16444009463104235004451765533162593406749138640537670707679905027165914805299755
93475817274822714649510700260354633859745296475986043108 1
'''
```

给了足足11组n和c，里面多半有相同的p或者q，直接gcd找出来就行了

exp

```
from Crypto.Util.number import long_to_bytes
import gmpy2
import sympy
def gcd(a,b):
     while b!=0:
         a,b=b,a%b
     return a
n1 =
13266094979118433615107077438835111371457503704538616496563510289250600197984029
76795095307172344238127990474772426970655580142588311704097283203539726834760361
20257412194523531302667551509650194410923800302262560327206518586001729963445983
55816134478110939124633509594465443169697188695226358357129879780253706604232278
67846568035801826208419810853307751904950781175842349851256552161593169949449159
50787230981010842176112044425851821032610060861443446123988965760669999509059669
20722348056083426326740287336766359383868804991393096926457607860171709146742941
74086478061022646385201820464709629273791576043029125081 3
c1 =
10454892625278317641789872724686322059272523357918030709377842011690595796341085
03308033729970364983743915474860354063759051514315619953792921367621385574749886
77789114718390798158209224302630161401413640941314968851558824867983024042385239
86615316359385759537428069831379320707742928167131988641527992674882901567191207
76354743229953271094554995958067011234683091950069401497244480390424858617888747
04658679916504800249043274327179113138677131205252026691015247788264455033835497
33326775312427637839488477432162027468606568090607495541492335140760946449358303
99921827726985923088829278023427549527478799600106995997 1
n2 =
13561524319281645924409429988746244285924512610515302181526524036825687984904507
67789461135334473574111053328054037273417896321687694506623292779136920785276214
68268839060938049587232033090308408048979992050744563250627891104529404840774085
80876939074453733126571662068654441714639301799127584606087971553812926212292460
76191598578646180764954628633083471974662438785083385925071779049267188473439305
54540747195859379042325166595827362119332800115916050696867910496283967626900383
33975652490908544846890333889140489210242676577675968427325105786641017561929481
33559768169381843589214085695967375421360028244476104590 9
```

c2 =
380993739663897379328111115083777460776883656125642549007106181454331521718268957932945778415937704074084079060357927067279082217904207905588740063758202265940059484442589166649375946952222969226710351629592358966615194894753300297550943050577808126426869829260620953804724082975691976495854451831199368479011812216385957192389911681526998929882214210478915696596438834680716706133215539276629053160274773191601077431840053335364139258765543245507801361914587766230292318029913194983462434272464844393919406766406936165197889502501946204553315574355677428740726370051717545500206193611501507608073913073774938367030763
n3 =
208199674327685679387536944229113033827136414357284766015868380517513127820145896878741006834470165498938538203992479096535199509951668050723806566520935796048085694433457001317569018139681857268398634928592315698967479875313410385275770302747011585047366114864111224168718768203605988519182762485200187934949924938523863999359190981852992689206884654462562833339189933921639475594708870744881175109611722783300260508822498975269269512469153788419617432614510530949127308507542429341661487669772094355971211713600666329730350975574820877157459405517603500285179124027953539296787703834844623378397624043844875820664533
c3 =
115638299135213147425851368054759934614034657001080537597714843868437654472948258125957478445648748018215473859429390177110437925904272662401330278314619749275586526650672138786835776058923199497040191937522504098138689064217532866905586466311570808879913631013115959218941281105883102047259092119485004218185953330147281149226128702367924451807579180345802103833482051746774927840348637014470972194801330950059540434906313393612804938100833097039406143715013696474466720521597149633506010392816521414460412561587929813475208656711150910973667865826780007699909894858797565629718251815838520521944929668902587869443153
n4 =
195176054747694585667640531691032039005388831671257894429481940233335992948535312730873865631616137487438183439481707655258020413133690945257080342404920385174910376974210196986729400280614893770785690272049405257417516077587068377892765718892698404632424496270369234445366434772080916640062401063378945415290982791626897936336591551311833236758832332146740988575534873908440154833267117660908263646397910723151480154504915094117055434197116328942821056984106294631410480898670161930553845307381133693525064456695213422928604682976768263011707146185195258417536427132670830075418941565228195369351291354882891986948894
c4 =
127208675540772156091672479497067451456675702733722963022460806689695112222976537224624562330882621535623010448854095388169312814146875722969079700044411858712297863064195868390509613630655879251886233094376878783623497120108614194094910684810197242260702967818069320121793565504217910401430937095435781566897262569936155336792386197840876159674196163767368708248976958637322605956089058988210742502859039896683032908340030986919029902655266957778994899114268559578445619617290636440708231423864514956479952974133998084765055245812604640597862838243947883367805677029500517433612941316482823433295877090771030353167204
n5 =
153157659718124288681266111093004199757775738118700598602889380125811311433146662870894724087291903067403513169224262447325850673772175021164401454508756656835754733081027420952443498042933110066643516469521812106868127489124181757360138356691915491105461278308314560688114930867875200322720042318791885721306599369376801779770589756927161169616510930339860732970406787075343009657228682638254441986792430705074132160872052939970068056363385675858157642333749177458091355688555249930225835356083492800448513327796416192103108371088192735087711618817127962355182435725497562761180537362701227878991736957290899110001 77

c5 =
19965763259897683243178870126805603705289491946263936646501827547915786977863699
58145137743555674312911707989139883926276952465377328602829900403114611234863417
18370767839804703004685545007426065443227815690338864194869671619373471483995762
43595334868440231111152977183983191207182170437280645952176517381808061949125819
64346594873253580010279706758241793778842399330631789750327865945625117967957133
86363717640430978251688051174777179371220390318551228532710308013356766039192103
86813935519456071527260636926417422864067729261033240785741811766323212877590668
02783643421564667257668981655199672490410258387124896839
n6 =
22501623321194869030526666823514739767749207843970334295553523279818676682652976
95980772938164679267613498810106578313675990445218928881304086002149742942293557
10839931467941862078531464836810943753607747684066295883113813752063313689658367
25531623084019451921330442379292718421616088275750226791314378297932358311378533
56962067142374059803320463963403581468978173610419732134815499975921053265145133
61923765319758372847519786091970685732785512368967137626161060518825057522877732
03197087939983952212761603430314087275042009367400077728025285967811883773835952
54702541185339530732595990421425510011576337752776186324 3
c6 =
16972901653690784382193404778525843883847230468707217844241233207197480301434902
62901808874082288994897210244566803348918132467553153765078576658230711208372141
05068342793634878147590218150058399066567310695217586052934289283227128955104909
68448490116732590950772037788323801838566829526879837783860631971103738986788475
65218708093594442866236357309722532263057470597761614331549727275405060811744993
81617724099249924536403106218992236918513039213235377790683465759976210246609077
50345496836548887894110692437965970524341315015139872598871247336564148953981001
70814525372878979948375393103989494332445817220740851405 5
n7 =
20222972180177072345801227937253710431874005900913732040381986459681923475683145
37299037529635464742066748386645556397649249586330661364519935172532306925403391
46604448846818017015271285789148156472282654575801925914053586466050693620258870
82090772778349481109768859094222573588734354390770166651349176555216207359309348
01834250546424980460787311898468819064839260655083571782816031706260459707910152
95865928098466081535480906891144315698445193179690768521560234298099186181371539
01636552629572816865533796914796916912143735510069871682428043277595493567381686
08644048039857906380149054108720097522437204744766685 3577
c7 =
19248329020030453774384696880312801243022181399316695013193573740699955812434234
45344573316012840888931209077573404778287610014598818332788491721567938238288106
54198707641867219706222487137614948508630827273736732642519512508139331541614226
45331713437130381414540777017708015790836797014483890216068812718266206449620134
39284337003635231024738462406436664658093793833058777370011686937963058428503017 0
82057951745360474953560815331211198096212006775548028507413270268577664668018470
93397207492834507197621929666294565502206253758771807936447505283097006026148803
10376179979591784806195940765508401216691641801931224978 4
n8 =
28799545621025859784610107508239196961299975467825740831913751504915874851330462
43915863557540535681811341033368318008233216569712042711295510314874443942544518
38852004193239006492473909873736437325453514283416721326069906519832171364277797
27640093677134233750683573339125630816150512829880758197582740742903567508671599
67063844169006462497088484915507992426131937841246831520593795963807477901081169
51328636073141751996598214509838251737381227490430638215397066480797314595965931
71608003970504410762959266052601005965264083020025264976022719772719760327913949
45176554959665411277657903291556611200766816777280002 2899

c8 =
155791245370947696673707699156106991613202656362395221432589945068477961139600551152519442500237742408059354887171709334429815472241440266257934874981436237250255359687608792440551234613964213004496615091068462935503350591801340806543248968935953365382453182974919650335330342650645751987072957842151770478808383787123968446850664679103142911144902439611114592590027446273135286565743973465097184774897466555552505415511716343556722891604685949816750212973976859057217032619136254245934640514243676448023993012332317521948850118097382020003036538697926564209076122713466985311902796437751606445020367316999674110820 1
n9 =
107247226453042597157952998190656747540429840137862105845062313006195606761084755363952720465699737436412517577460611978285197803156184583771035630622311834834178130261649327428281602459362637176530187460017981066700747138264255387583275380776668184118035567903031187036752145702320323637412446521998778298509413569753496104332706025009496647376148292645707203769826976134979177340789988947241539873836875315030078220170162984026379109989879922393909723773776330574424452091383115136115592646208625238919654858321527523065262720570936292198923548723743783051947930885363902418494076790401946840502456727926919466045 7
c9 =
705463612113234358385869930109575071411123581792011545946870788472501819962118526856546209953758372723112393566588030444115510698765583426436362250395222872052524217950807901663108956447269419692829296759200709052790183244098047860554527589329477590573220979646359710511667636350560764438429321090560904489885482334941228236466031666385339991490559120640881321780940909164095426401350314009757434019464208207922992311335823038396957805698822367795673327054284423260967631762122524511318733561339982590040303505605155884427567432712042870632816967182612225212150848091424300840160529323960243459920269872536666352976 1
n10 =
163048179013974834174566093415161245232324267800233972282729130447233653895713277971345917369775026920577274163949243108288243780248127775293673427078964948159602556829476850369022359978998862916336090416858971972726759113049415010677947952859495403139902264096512921201264927149988922900441115410983656927311857873606091989216817367771037944306536966981411959547190308839351519808473399392130403732528191456550672869863006416689947003909214586825716194094882332892117385210460853112531195266859128934581315347180508026841802851388092172742738773763470240210563175663121861339167445334638797799977945403617603932194891 1
c10 =
603912266080365615770005473906456433843954948469520621401600612950878236222984296760246407773587454010716145904183534099010881713967213787064293561353131358396393014487145873302639622367480155762103694931211663921744711489972685898428119411650824440398556878576527383857247089892552114702202151686265896758413545447435538107302606642645061478838100205798059044233876686910495757733854668775914122894214937448578589353933528853672932821817227837076318360342262979984661173771086568157225780344896065669250677308652738961617869315288526079530300811193527752956157691511416567856499792363490637705949198939946773806964 1
n11 =
266255915701089179918476240243258190203209892554836082306448794765179152119955428963414165156055769283923177176546957670067847447519622053551070128594261344740746161349864142480547817121308920764622887616726169932874742152710466024432154492100263039782497528253060350978414295677842656568908015894183154499275436601319673652766376625631123284278131833985179442101208518414950012935439717658643820601006001791088148528294738844818672727169070647299911877983287331966762986813283992503560281554118659487584663377630258866854366506434285105936971764167355682506156736428666684165266823417219984815024828779726490185014691 1

```
c11 =
20247005397625616228661928392695469647749434533141783786437968083066137357689767
28075353260589641917333259752970880194690636378059287611110485030278403313633545
66833611286036624547282343495632389488017129079031859935610767541645528455107952
93521029798357876214260903355702688269527465118808832941386890598851762900876512
38355765394084623288462755795765263858102214082537302317348247179172014879230124
15819197655102200830793653112970816339793988559669910690205844626139810948887369
16444009463104235004451765533162593406749138640537670707679905027165914805299755
93475817274822714649510700260354633859745296475986043108 1
e=65537
n=[n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11]
c=[c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11]
p=q=cc=0
for i in range(11):
    for j in range(i+1,11):
        if (gcd(n[i],n[j])!=1):
            p=gcd(n[i],n[j])
            q=n[i]//p
            cc=c[i]
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(cc,d,p*q)
print(long_to_bytes(m))
```

```
b'HSCCTF{this_1s_yi_zhi_xiao_jiu_li}'
```

SIGN_IN

> 这题也是网上几乎有一模一样的题。。。

题目

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

flag = 'HSCCTF{66666666666666666666666666666666}'
m = bytes_to_long(flag.encode())


p, q = getPrime(1024), getPrime(1024)
n = p * q
g, r1, r2, k1, k2 = [randint(1, n) for _ in range(5)]
g1 = powmod(g, r1 * (p - 1), n)
g2 = powmod(g, r2 * (q - 1), n)
c1 = m * powmod(g1, k1, n) % n
c2 = m * powmod(g2, k2, n) % n
print('enc=', [c1, c2])
print('n, g1, g1=', [n, g1, g2])
'''
```

```
enc=
[152588138011827679579488094114455301147435800056678974275343655898511244019539527008880323300822787365786516741052045536301159298293275209503169025044330793599162573558969375629165560834696029851152542726253619025129669053665057051936031150634425672595058069268855979093468601165111165935628967397346630976318700268144959994055323296627137071409916936255969724044843742965885053694661383537144617878536439476462681479935681117752002977884430856893237038464854126702137801751635624603112563881859486137145511692783283971811768210214946255329730656066785220880110550849565790736654745536888387051056923708608158369687 44,
681225297307204007182776421228706848522450666496292327999461416437484009563046068185940658583886525953515044541457726338383928660266887214663056858309320323735023053189611345373056353655978072505473793070405226383688270496352843789360794037314567484873127419334877045073835616214155171934699469814911282738301104263562599194649154569846451331570525344137235627024853702539151905222833274734009007372664807192105309761252194904058882668488757790003060796445754354901533854468438122055295299572470157942736338206906567954519682694790033260525085869457442349709620735053294951766341687795448491366416810480751005299434 3]
n, g1, g1=
[208070183444864746393074291772799317667307663380681072399155074893000980488954357817565142650698200292463828189707332682715414190585695678956186669221802705840643136715613107040417554226248794181840081004735683961905230339293202457163400194447642332611912789823549314928808775665501894108543954869956042870094600908619521566867724862285838192359306738066326187708755938106177841764092475104655422530436451641642255244917047533844938175632039566650227246745355168707267474059743923525004328056020852613751422165016563425336597310988914816157913545043439874594364890403776621274797937695233593435343078190132896998071011366995491911187355476268572292688507711852899941440216329123793960509524982736126658540434881993689955750075130732808885013346210018990864970535422517136274856733282873379241087736539347236876764222361348394486442122536698401305315964328484282369286597552101340016064385812364551302064736703302253917706370251161827201801934568632432286546158303080552850896155986122880789845834476390970362472308231694666053920826281673499977723668690448716754559641286614619276985942243597961744135596807798133849146667029131049162399694388055707172307329826731061701793399493848160090583382973059991118309611640948760885871708376 4,
160047552728969730883052421348353635659501016670411917670913385280757107306904842412566464021243317341653327172218733000408442571202507177984194290033424748829692921833316317734860563637901195004095113384748851486247447193348295191389417477219409787426684293311693271218485615773573391949704833924180623071336426606259624265709870803481079998492207695418321387193584078338462529083057665465615320036504727703705735524249329085442993314891472209565741848224339455713186884671823652319552765731764335597553138225666878808667913019044615591704167536691098333358242108533222267627055370510093878471222210143802283831274 27]
'''
```

通过费马小定理知

$$g^{r_1(p-1)} \equiv 1(mod\ p)$$

又因为

$$g^{r1(p-1)} \equiv g1(mod\ n)$$

$$n = p * q$$

所以

$$g1 \equiv 1(mod\ q)$$

即g1-1是q的倍数，所以只要求g1-1和n的最大公因数即为q，易得p

exp

```
import gmpy2
from Crypto.Util.number import long_to_bytes
def decrypt(c1, c2):
    xp = c1 % p
    xq = c2 % q
    # Chinese Remainder Theorem
    m = (xp*gmpy2.invert(q, p)*q + xq*gmpy2.invert(p, q)*p) % n
    return m
n=208070183444864746393074291772799317667307663380681072399155074893000980488954
357817565142650698200292463828189707332682715414190585695678956186669221802705840
643136715613107040417554226248794181840081004735683961905230339293202457163400194
944476423326119127898235493149288087756655018941085439548699560428700946009086195
215166867724862285838192359306738066326187708755938106177841764092475104655422530
436451641642255244917047533844938175632039566650227246745355168707267474059743923
525250043280560208526137514221650165634253365973109889148161579135450434398745943
648904037766212747979376952335934353430781901328969980710 1
g1=136699549191118735547626857229268850771185289994144021632912379396050952498273
612665854043488199368995575007513073280888501334621001899086497053542251713627485
673328287337924108773653934723687676422236134839448644212253669840130531596432848
428236928659755210134001606438581236455130206473670330225391770637025116182720180
193456863243228654615830308055285089615598612288078984583447639097036247230823169
466605392082628167349997772366869044871675455964128661461927698594224359796174413
559680779813384914666702913104916239969438805570717230732982673106170179339949384
816009058338297305999111830961164094876088587170837641600475527289697308830524213
483536356595010166704119176709133852807571073069048424125664640212433173416533271
722187330004084425712025071779841942900334247488296929218333163177348605636379011
950040951133847488514862474471933482951913894174772194097874266842933116932712184
856157735733919497048339241806230713364266062596242657098708034810799984922076954
183213871935840783384625290830576654656153200365047277037057355242493290854429933
148914722095657418482243394557131868846718236523195527657317643355975531382256668
788086679130190446155917041675366910983333582421085332222676270553705100938784712
222101438022838312742 7
c1=152588138011827679579488094114455301147435800056678974275343655898511244019539
527008880323300822787365786516741052045536301159298293275209503169025044330793599
162573558969375629165560834696029851152542726253619025129669053665057051936031150
634425672595058069268855979093468601165111165935628967397346630976318700268144959
994055323296627137071409916936255969724044843742965885053694661383537144617878536
439476462681479935681117752002977884430856893237038464854126702137801751635624
603112563881859486137145511692783283971811768210214946255329730656066785220880110
550849565790736654745536888387051056923708608158369687 4
c2=681225297307204007182776421228706848522450666496292327999461416437484009563046
068185940658583886525953515044541457726338383928660266887214663056858309320323735
023053189611345373056353655978072505473793070405226383688270496352843789360794037
314567484873127419334877045073835616214155171934699469814911282738301104263562599
194649154569846451331570525344137235627024853702539151905222833274734009007372664
807192105309761252194904058882668488757790003060796445754354901533854468438122055
295299572470157942736338206906567954519682694790033260525085869457442349709620735
053294951766341687795448491366416810480751005299434 3
p=gmpy2.gcd(g1-1,n)
q=gmpy2.gcd(g2-1,n)
m=decrypt(c1,c2)
print(long_to_bytes(m))
```

```
b'HSCCTF{Any_restr1ct1ons_all_begins_with_his_own_heart}'
```

~~看这个flag预期解不会就是要搜索吧。。。~~

Mixed

> 也是网上有类似的题。。。

题目

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

m = libnum.s2n(flag)
e = 65537
q = getPrime(1024)
q1 = getPrime(1024)
p = getPrime(1024)
p1 = getPrime(1024)
n = p * q
n1 = q * p1
n2 = p * q1
c = pow(m, e, n)
h0 = pow(2023 * p + 2024, q1, n2)
h1 = pow(2024 * p1 + 2023 * q, 114, n1)
h2 = pow(2023 * p1 + 2024 * q, 514, n1)

print(f'n1 = {n1}')
print(f'n2 = {n2}')
print(f'c = {c}')
print(f"h0 = {h0}")
print(f"h1 = {h1}")
print(f"h2 = {h2}")

# n1 =
14689930167621480307737218855980844793204825453111553640005733004311399091452749
41919526172316056266971298107720551448238742854385906040833055545787361238897084
00755552193187242099193796703757544369863227250182297011758255102539876150175422
05904982728558285520666278391637251512741084978804344612886660251741043298476839
94244119778872269124894900801829910151715306790469231016045983932796521312645677
60424703279236030981330196409567028209082998595346763810949741487226681719320914
55546601087513513432342088579253210930555185169910878657816252698605590059713584
04428741199844527910496522911228719577235591401469437 2863
# n2 =
17914624231045471060767877283214815212078805279600237578901916178851704643026979
52227065863521296728513165454620728687572219804056409435169006602855605327411882
64267264369221520568843337908502204887937228367589897702316973920153617457172342
25528175411653330848965062023338707342188196332661335758350340958332400847642883
21965066140889519136307901775014477658759805835092039349811449860304680759728860
38827824733226306319154674701310902003695274367603132893448137440040039579326061
66823948354564464154141854938585629799995141656546295799610158824771809513458197
81293659564932647512340084517812217131754384380235723873 7
```

```
# c =
131836457888285849027349757718338426341347729711970663742787161983319823628455973385419875798275761291269623005550233192768960801716622619833236705164578166080690765767043798715598815851003537379899151431902778561456777337470735309694594796088544838668863814067902210769006240836054186704504395230621906334877649823993349674315248312643379498143679051852826005118297941701396236255452889819920931086195561803526201715502849234528644026241957907818994151385488276840427907743823135209613352932252959225632451485988074983206529734092399219622923952697391516924430691866951374587742273307487754696219089628785487845426560# h0 =
569043585863280105300156827656920081107232478639435128233073643777764389167571376026757350087501803676981920122870749467603353473351643688043400038115229052993412758404768118479773080632111707957843821310659556610291023183164199487077084481079774593533696280165836054175521407510114914248696446166983304300717417393469321452655237591064457472352616402166049747476051445935127069905507638076318802126573500733927224342628647000604024692618430552086006809513400366249683764487587910631823360047800016045033872705492001106353638753276366193497296535732647101267855018838294558412813613183308500623907593175390070585248 1# h1 =
127717041422313613569449313282039928248094729232880350567411453104463231353787398064303576006887808625321803949676074297088117675905411260198054231710235609732504435084276143628545832404024519447305123349087792251747634981220051980844875629125244774762419780665540265696094946418300348172981858877909876274445443916803882132055142841167323253501872045197553998413721441498409146515218424028177012982248032894513785812262040847570794576129843589257673784589946085621740487298063092300610895520451385145427584572436261506738881666208411340637131512402977745514893015042229331274588843748666115982798315624207201697906 88# h2 =
107682691344999059163889290495955098957510981123723236187985421595691740515787017105496090548089957741689313751862135500749003730239708824508226557867877404526812788379071910913385755615702112800958945766740896669816834053858477738192385817250938234297766845616674626269415182480245177079920354380826362598388225628431694415720489564848317306150473633839114546627897367615729128932342128750975369784237582751780507086165053113525119238106548733808082013956564169039003266345696721732282657305157273278325883265815924857132259273192259429651102854772662003804526377460376724113886960069330805091646157866160586644968 51
```

首先对h0分析，

$$h_0 = (2023p + 2024)^{q_1} \, (mod \ n_2)$$

二项式定理展开得到

$$h_0 = 2024^{q_1} \, (mod \ p)$$

根据费马小定理

$$2024^{q_1} \equiv 2024^{p*q_1} \equiv 2024^{n_2} \, (mod \ p)$$

联立

$$\begin{cases} h_0 = 2024^{q_1} + k_1 p \\ 2024^{n_2} = 2024^{q_1} + k_2 p \end{cases}$$

得到

$$h_0 - 2024^{n_2} = k_3 p$$

所以可以与n2=p*q1求最大公因数，得到p

接下来我们求q。根据h1和h2,

$$\begin{cases} h_1 = (2024 * p1 + 2023 * q)^{114} (mod\ n_1) \\ h_2 = (2023 * p1 + 2024 * q)^{514} (mod\ n_1) \end{cases}$$

二项式展开

$$\begin{cases} h_1 = (2024 * p1)^{114} + (2023 * q)^{114} (mod\ n_1)\ (1) \\ h_2 = (2023 * p1)^{514} + (2024 * q)^{514} (mod\ n_1)\ (2) \end{cases}$$

(1)式两边同乘2023^114，(2)式两边同乘2024^514

$$\begin{cases} 2023^{114} h_1 = (2023 * 2024 * p1)^{114} + (2023^2 * q)^{114} (mod\ n_1)\ (1) \\ 2024^{514} h_2 = (2023 * 2024 * p1)^{514} + (2024^2 * q)^{514} (mod\ n_1)\ (2) \end{cases}$$

(1)式两边进行514次方并二项式定理展开，(2)两边进行114次方并二项式定理展开

$$\begin{cases} 2023^{114*514} h_1 = (2023 * 2024 * p1)^{114*514} + (2023^2 * q)^{114*514} (mod\ n_1)\ (1) \\ 2024^{114*514} h_2 = (2023 * 2024 * p1)^{114*514} + (2024^2 * q)^{114*514} (mod\ n_1)\ (2) \end{cases}$$

(2)式-(1)式

$$2024^{114*514} h_2 - 2023^{114*514} h_1 = k_4 q (mod\ n_1)$$

$$2024^{114*514} h_2 - 2023^{114*514} h_1 = k_4 q + k_5 n_1$$

又因为n=p1q，所以

$$2024^{114*514} h_2 - 2023^{114*514} h_1 = k_6 q$$

与n1=p1*q求最大公因数，得到q

exp

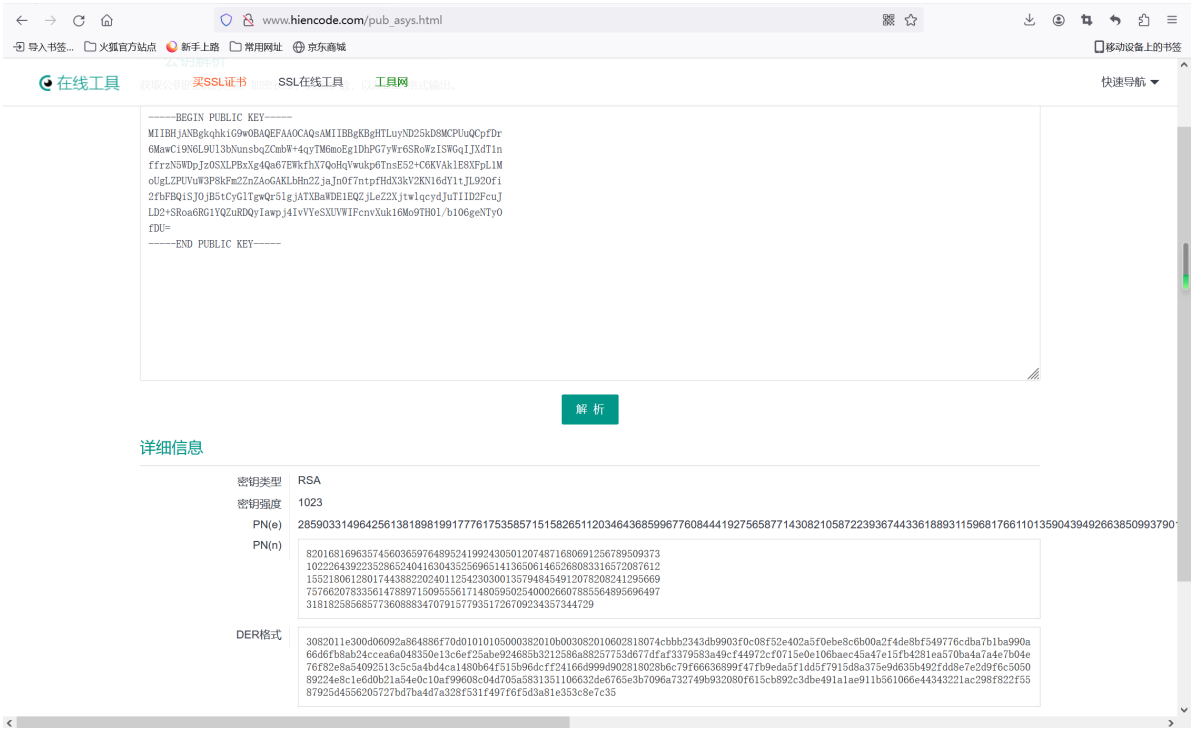
```
import gmpy2
import libnum
from Crypto.Util.number import long_to_bytes
h0 =
569043585863280105300156827656920081107232478639435128233073643777764389167571376026757350087501803676981920122870749467603353473351643688043400038115229052993412758404768118479773080632111707957843821310659556610291023183164199487077084481079774593533696280165836054175521407510114914248696446166983304300717417393469321452655237591064457472352616402166049747476051445935127069905507638076318802126573500733927224342628647000604024692618430552086006809513400366249683764487587910631823360047800016045033872705492001106353638753276366193497296535732647101267855018838294558412813613183308500623907593175390070585248 1
h1 =
127717041422313613569449313282039928248094729232880350567411453104463231353787398064303576006887808625321803949676074297088117675905411260198054231710235609732504435084276143628545832404024519447305123349087792251747634981220051980844875629125244774762419780665540265696094946418300348172981858877909876274445443916803882132055142841167323253501872045197553998413721441498409146515218424028177012982248032894513785812262040847570794576129843589257673784589946085621740487298063092300610895520451385145427584572436261506738881666208411340637131512402977745514893015042229331274588843748666115982798315624207201697906881
```

```
h2 =
10768269134499905916388929049595509895751098112372323618798542159569174051578701
71054960905480899577416893137518621355007490037302397088245082265578678774045268
12788379071910913385755615702112800958945766740896669816834053858477738192385817
25093823429776684561667462626941518248024517707992035438082636259838822562843169
44157204895648483173061504736338391145466278973676157291289323421287509753697842
37582751780507086165053113525119238106548733808082013956564169039003266345696721
73228265730515727327832588326581592485713225927319225942965110285477266200380452
63774603767241138869600693308050916461578661605866449685 1
n1 =
14689930167621480307737218855980844793204825453111553640005733004311399091452749
41919526172316056266971298107720551448238742854385906040833055545787361238897084
00755552193187242099193796703757544369863227250182297011758255102539876150175422
05904982728558285520666278391637251512741084978804344612886660251741043298476839
94244119778872269124894900801829910151715306790469231016045983932796521312645677
60424703279236030981330196409567028209082998595346763810949741487226681719320914
55546601087513513432342088579253210930555185169910878657816252698605590059713584
04428741199844527910496522911228719577235591401469437286 3
n2 =
17914624231045471060767877283214815212078805279600237578901916178851704643026979
52227065863521296728513165454620728687572219804056409435169006602855605327411882
64267264369221520568843337908502204887937228367589897702316973920153617457172342
25528175411653330848965062023338707342188196332661335758350340958332400847642883
21965066140889519136307901775014477658759805835092039349811449860304680759728860
38827824733226306319154674701310902003695274367603132893448137440040039579326061
66823948354564464154141854938585629799995141656546295799610158824771809513458197
81293659564932647512340084517812217131754384380235723873 7
e=65537
c =
13183645788828584902734975771833842634134772971197066374278716198331982362845597
33854198757982757612912696230055502331927689608017166226198332367051645781660806
90765767043798715598815851003537379899151431902778561456777337470735309694594796
08854483866886381406790221076900624083605418670450439523062190633487764982399334
96743152483126433794981436790518528260051182979417013962362554528898199209310861
95561803526201715502849234528644026241957907818994151385488276840427907743823135
20961335293225295922563245148598807498320652973409239921962292395269739151692443
06918669513745877422733074877546962190896287854878454265 6
kp=h0-pow(2024,n2,n2)
p=gmpy2.gcd(n2,kp)
kq=pow(h2*pow(2024,514,n1),114,n1)-pow(h1*pow(2023,114,n1),514,n1)
q=gmpy2.gcd(n1,kq)
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,p*q)
print(libnum.n2s(int(m)))
```

RSATEST

> 这题没告诉格式，被这个卡了很久。。。最后没想到flag头就是flag。。。

先进行一波公钥解析，得到n和e



这里看到e很大，尝试使用维纳攻击

exp

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


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

```
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return d
e =
285903314964256138189819917776175358571515826511203464368599677608444192756587714308210587223936744336188931159681766110135904394926638509937901773810590770451970861339401184020885540880731215914020953239661077224856476434528018065115833273616641147279351714022566118776270719966438412233563623677475555753013
n =
820168169635745603659764895241992430501207487168069125678950937310222643922352865240416304352569651413650614652680833165720876121552180612801744388220240112542303001357948454912078208241295669757662078335614788971509555617148059502540002660788556489569649731818258568577360888347079157793517267092343573447291
d=wienerAttack(e, n)
f=open("output.txt",'rb')
s=f.read()
c=bytes_to_long(s)
m=pow(c, d, n)
flag=long_to_bytes(m)
print(flag)
```

```
b"\x02\xfe\\`\x1dEd\x14\xb4\x8a\xb6;\xc6Qr*\xd9\xc0J\xda\x98Q\x8et\xa8\x9a37K<\xe19\xdb/C\x17o\x10L\xbe\x98\x9f\x8b\x9c\x94\xbd\xb5S\xb5\xd8Fo\x04\xc3'\x9f\xa9N\xe8\x95?KnH\xfb;$\xc3\xe8t3\xfc\xd4\xfe\xd1\xdd\xc6\xb1V\xec\xac\x8b\xb8\xd4\x9bxG\xfa\x90\xb1\xf7\n\xc3\x0f\x0009C96E1A3ACC46E3C9F7603EB592FE4A"
```

前面进行了PKCS7填充，取最后32位即可

flag{09C96E1A3ACC46E3C9F7603EB592FE4A}

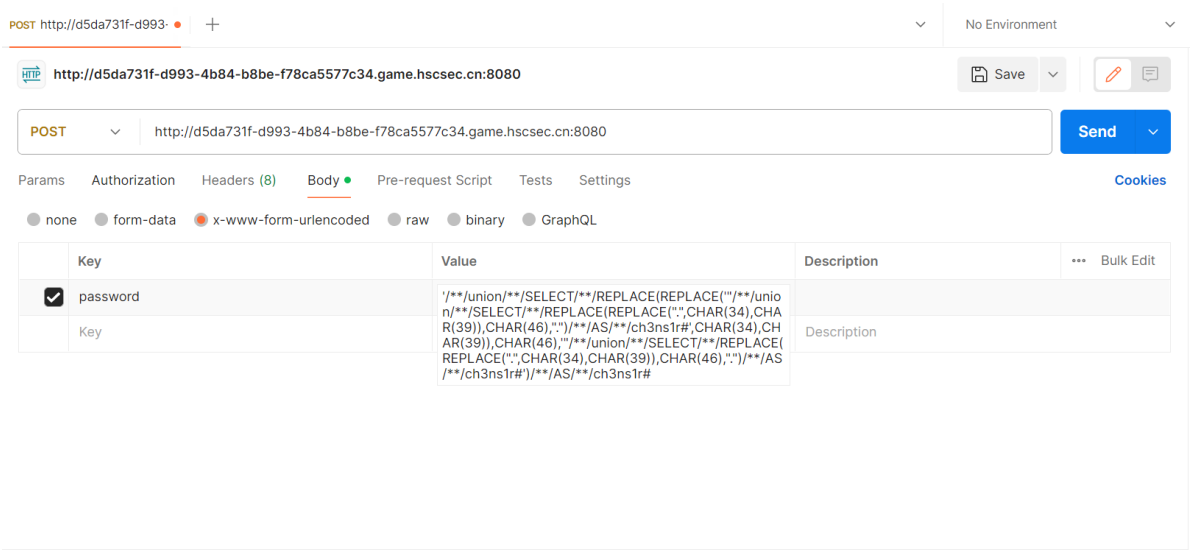
Web

PWD

根据这篇文章来就行https://www.anquanke.com/post/id/253570

Quine trick

payload:



Reverse

NO_PY

> 做着做着记起隔壁BeginCTF有几乎一样的题，最后发现连flag都没改。。。

pyinstaller打包成的exe程序，用pyinstxtractor反编译一下

```
C:\Users\jyzho\Desktop\h4ck3r t0015\pyinstxtractor-master>python pyinstxtractor.py ezpython.exe
[+] Processing ezpython.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.8
[+] Length of package: 5409213 bytes
[+] Found 58 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: ezpython.pyc
[+] Found 78 files in PYZ archive
[+] Successfully extracted pyinstaller archive: ezpython.exe

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

用uncompyle6把反编译出的ezpython.pyc再反编译一下，得到python源码

```
C:\Users\jyzho\Desktop\h4ck3r t0015\pyinstxtractor-master\ezpython.exe_extracted>uncompyle6 ezpython.pyc
# uncompyle6 version 3.9.0
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)]
# Embedded file name: ezpython.py
from gmssl import sm4
from secret import key, enc
import base64

def pad_pkcs7(data):
    """PKCS#7填充"""
    padding_len = 16 - len(data) % 16
    padding = bytes([padding_len] * padding_len)
    return data + padding


def unpad_pkcs7(padded_data):
    """PKCS#7去填充"""
    padding_len = padded_data[-1]
    return padded_data[:-padding_len]


class SM4:

    def __init__(self):
        self.gmsm4 = sm4.CryptSM4()

    def encryptSM4(self, encrypt_key, value):
        gmsm4 = self.gmsm4
        gmsm4.set_key(encrypt_key.encode(), sm4.SM4_ENCRYPT)
        padded_value = pad_pkcs7(value.encode())
        encrypt_value = gmsm4.crypt_ecb(padded_value)
        return base64.b64encode(encrypt_value)


if __name__ == '__main__':
    print('请输入你的flag:')
    flag = input()
    sm4_instance = SM4()
    flag_1 = sm4_instance.encryptSM4(key, flag)
    if flag_1 != enc:
        print('flag错误!!')
    else:
        print('恭喜你获得flag😊😃')
# okay decompiling ezpython.pyc
```

可以看出是个sm4加密，去找一下secret.pyc反编译一下，得到key和密文

```
C:\Users\jyzho\Desktop\h4ck3r t0015\pyinstxtractor-master\ezpython.exe_extracted\PYZ-00.pyz_extracted>uncompyle6 secret.pyc
# uncompyle6 version 3.9.0
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)]
# Embedded file name: secret.py
key = 'Please_Do_not_py'
enc = b'YJ+70VWYioYm3EhF6qdScVXBpCdPm+hCS/HP+Gj421RyxkZbwzni7o2zGG/Mis4Wr6sbXYr4ufnkwQk7vrG8yA=='
# okay decompiling secret.pyc
```

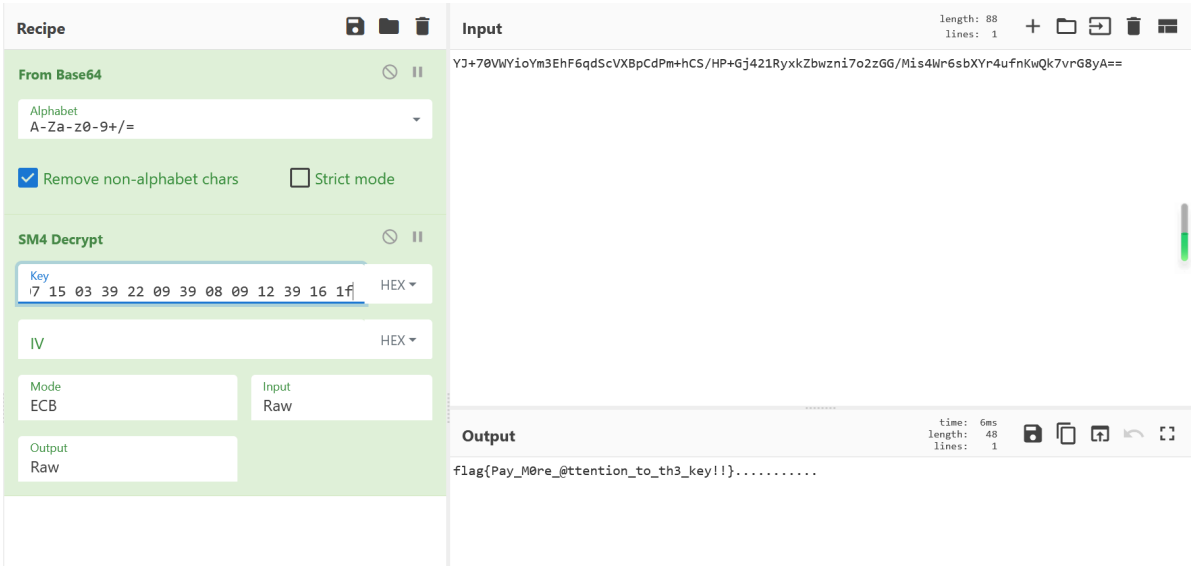
在线网站上直接解密解不出，去找一下sm4.pyc反编译一下，发现在key那边魔改过，对整个key异或了102

```
        rk = bb ^ rotl(bb, 13) ^ rotl(bb, 23)
        return rk

    @classmethod
    def _f(cls, x0, x1, x2, x3, rk):

        def _sm4_l_t(ka):
            b = [
             0, 0, 0, 0]
            a = put_uint32_be(ka)
            b[0] = SM4_BOXES_TABLE[a[0]]
            b[1] = SM4_BOXES_TABLE[a[1]]
            b[2] = SM4_BOXES_TABLE[a[2]]
            b[3] = SM4_BOXES_TABLE[a[3]]
            bb = get_uint32_be(b[0:4])
            c = bb ^ rotl(bb, 2) ^ rotl(bb, 10) ^ rotl(bb, 18) ^ rotl(bb, 24)
            return c

        return x0 ^ _sm4_l_t(x1 ^ x2 ^ x3 ^ rk)

    def set_key(self, key, mode):
        key = bytes_to_list(key)
        key = [i ^ 102 for i in key]
        MK = [0, 0, 0, 0]
        k = [0] * 36
        MK[0] = get_uint32_be(key[0:4])
        MK[1] = get_uint32_be(key[4:8])
        MK[2] = get_uint32_be(key[8:12])
        MK[3] = get_uint32_be(key[12:16])
        k[0:4] = xor(MK[0:4], SM4_FK[0:4])
        for i in range(32):
            k[i + 4] = k[i] ^ self._round_key(k[i + 1] ^ k[i + 2] ^ k[i + 3] ^ SM4_CK[i])
            self.sk[i] = k[i + 4]
        else:
            self.mode = mode
            if mode == SM4_DECRYPT:
                for idx in range(16):
                    t = self.sk[idx]
                    self.sk[idx] = self.sk[31 - idx]
                    self.sk[31 - idx] = t

    def one_round(self, sk, in_put):
        out_put = []
        ulbuf = [
         0] * 36
        ulbuf[0] = get_uint32_be(in_put[0:4])
        ulbuf[1] = get_uint32_be(in_put[4:8])
        ulbuf[2] = get_uint32_be(in_put[8:12])
        ulbuf[3] = get_uint32_be(in_put[12:16])
```

用python转成异或成十六进制形式

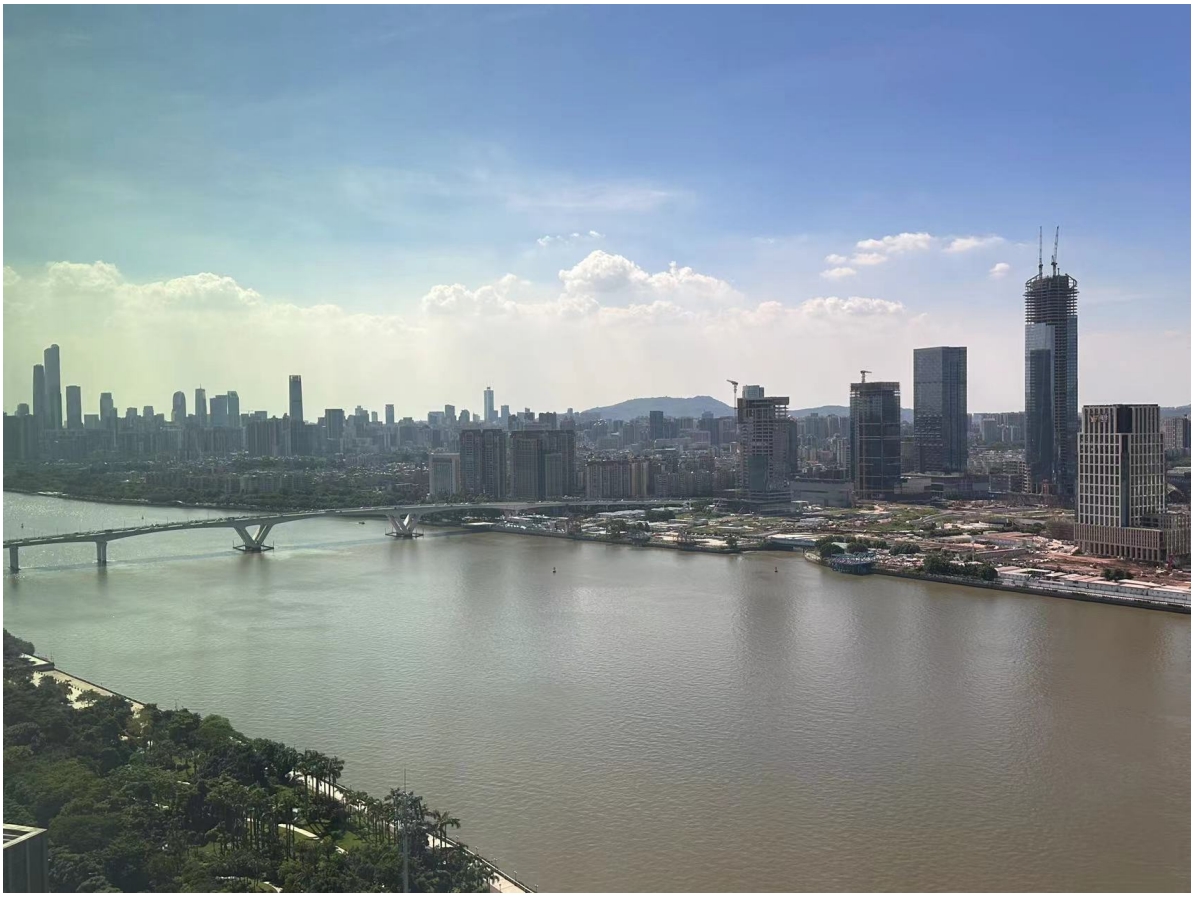
```python
s="Please_Do_not_py"
ls=[]
for i in s:
    ls.append(hex(ord(i)^102)[2:])
print(ls)
```

cyberchef转一下



OSINT

NICE_RIVER

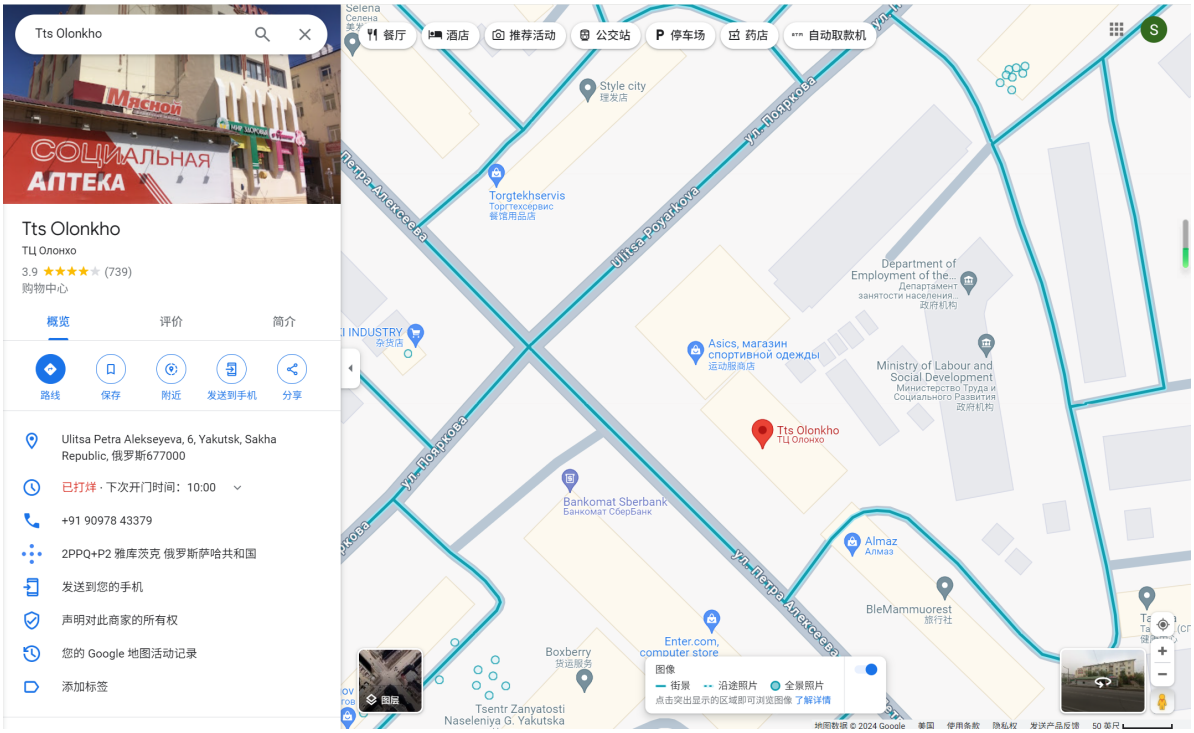


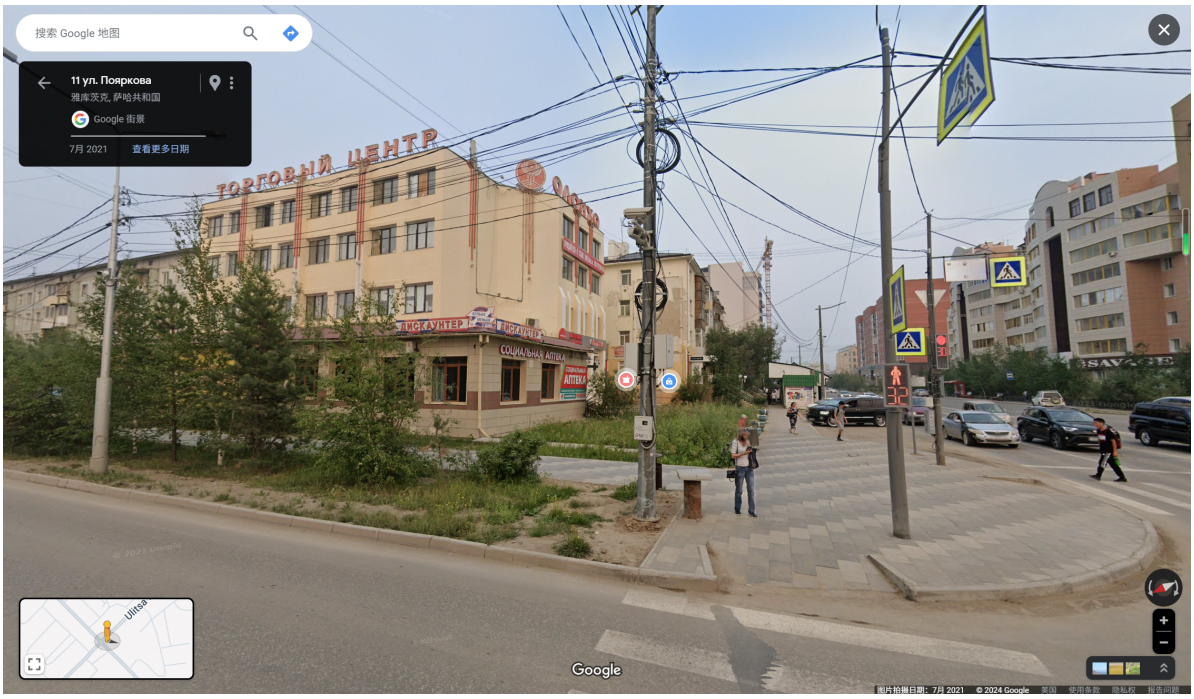
百度识图一下，发现应该是琶洲大桥，按照这个角度，应该是在旁边的保利洲际酒店

HSCCTF{23.10413,113.38464}

город

这个还是比较好找的，放进google lens里面一找就找到了





HSCCTF{62.03725,129.73639}

JAPAN

根据路牌上面的英文很容易锁定地区，在周围找找也能很快找到

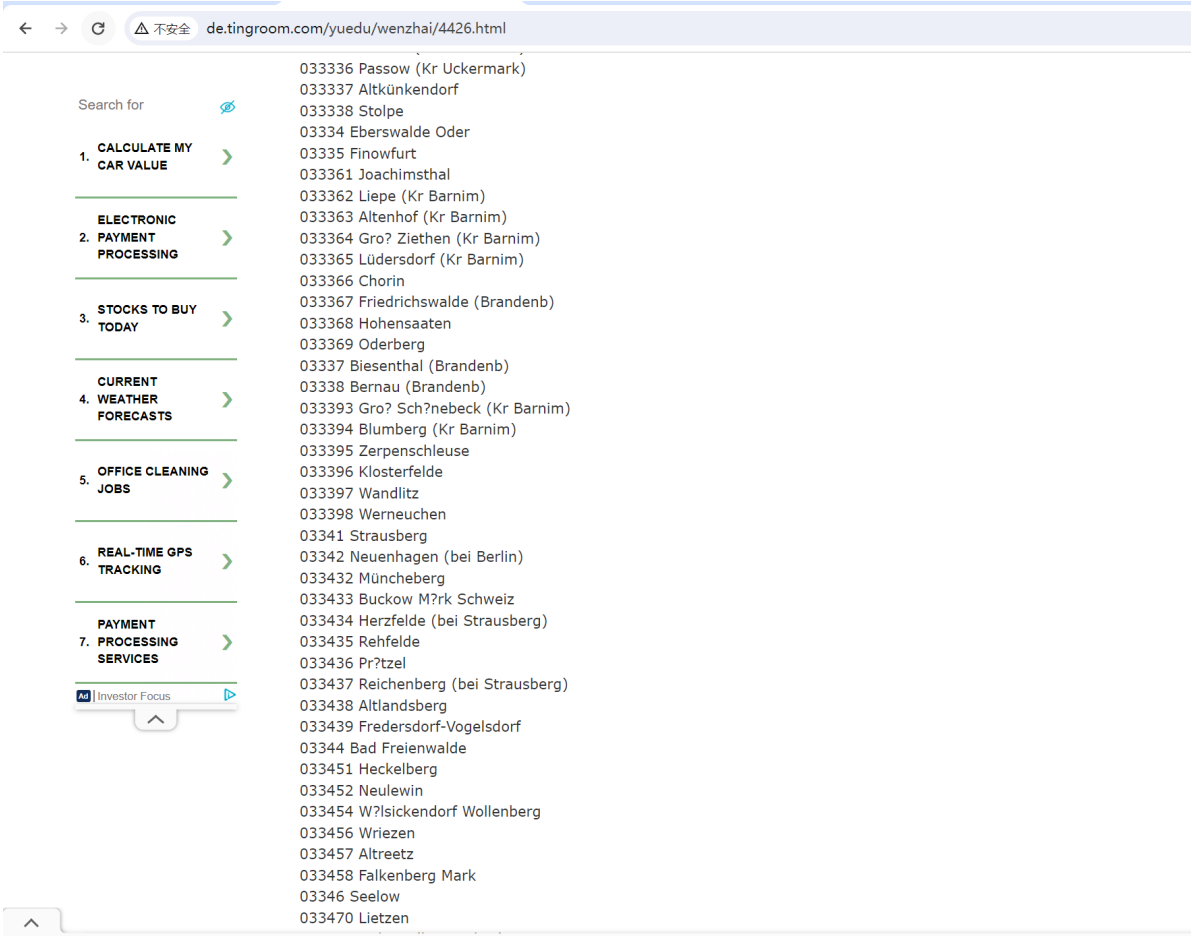
HSCCTF{26.33445,126.77204}

WELCOME_TO_HSCCTF

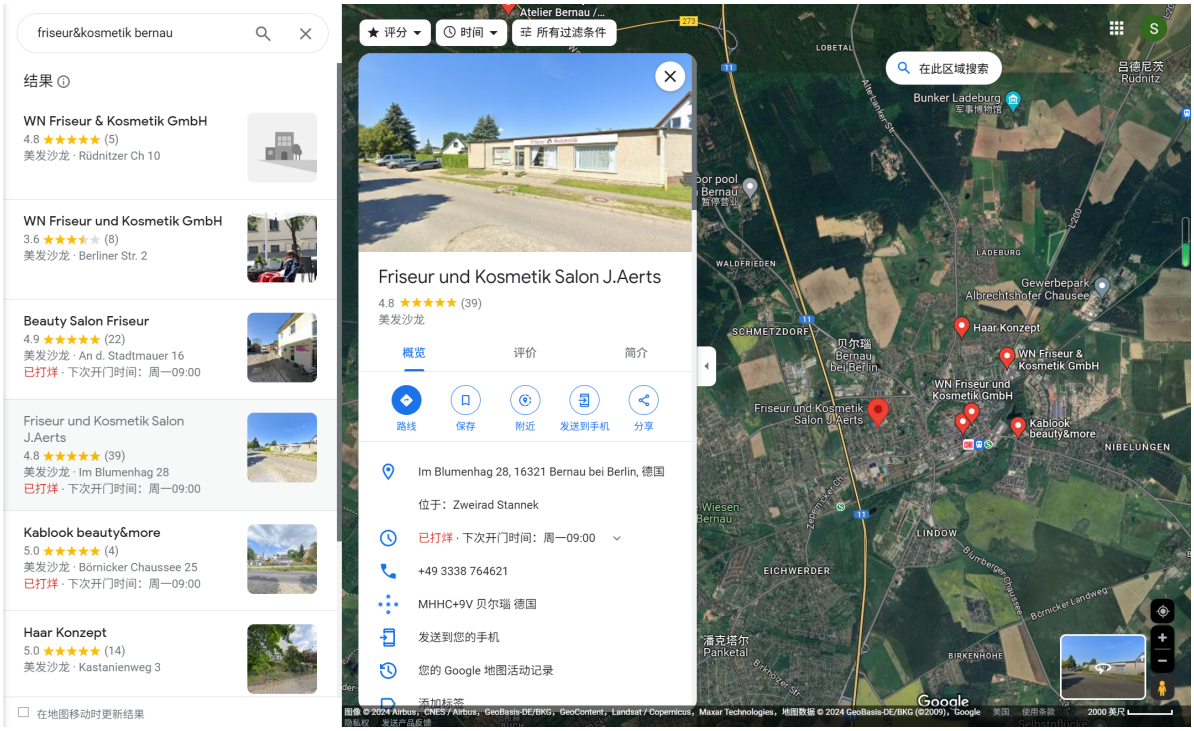
这题犹点意思。

首先看到的肯定是这个店名Friseur&Kosmetik，还有窗户上的montag（周一）什么的，幸好我会点德语，这里一眼德语，确定在德国

然后根据窗户上的tel和倒影在窗帘上的影子可以看出这个电话的区号是03338，网上查一查，发现是德国一个叫作Bernau的地方



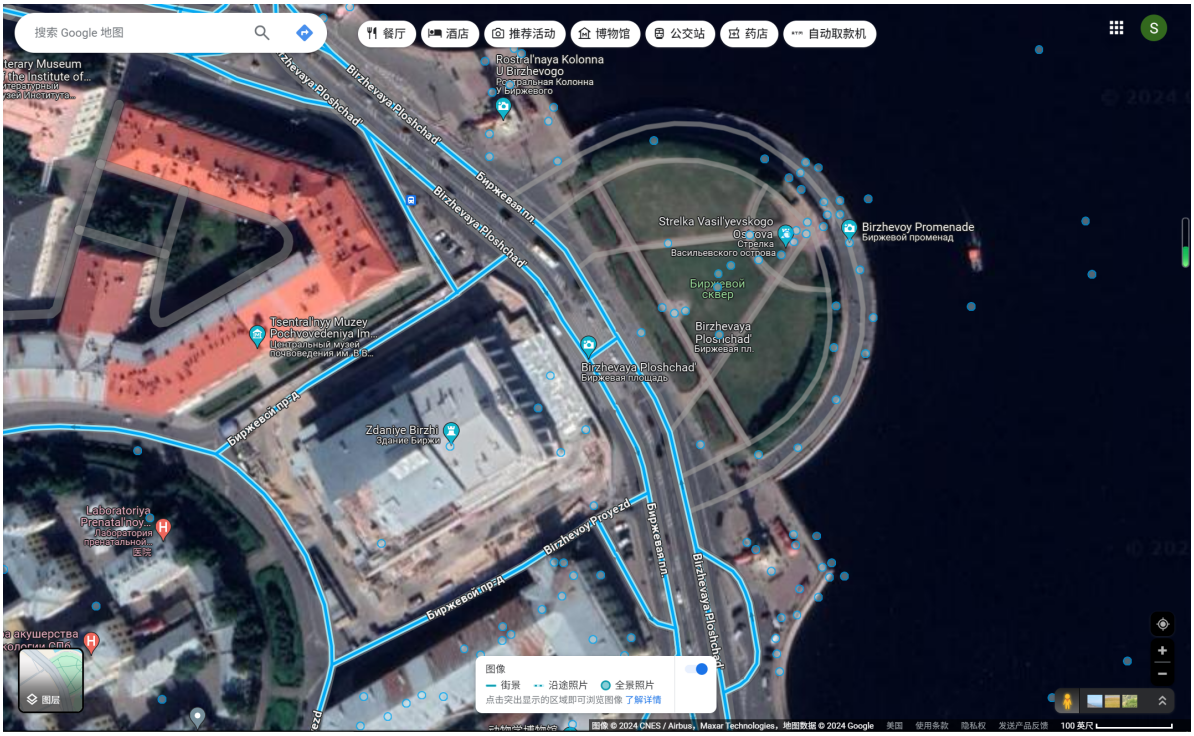
因此很快能够锁定这家店



HSCCTF{52.67842,13.57238}

FLAG

幸好我还会点俄语，这里根据旗帜上写的东西可以确定是俄罗斯的圣彼得堡（美国的应该不会写俄语吧。。。）丢进google lens一查，知道是个叫瓦西里岛的地方，地图上看下，确实在圣彼得堡，因此确定范围，很快能找到这个地方



HSCCTF{59.94392,30.30573}

CHECKIN

CHECKIN

关注两个公众号得到

hscctf{hscctf2O24_w3lc0me}