hgame 2022 逆向 reverse 部分题目 Writeup

原创

拾光、 已于 2022-02-24 16:33:08 修改 347 收藏 1
分类专栏： ctf 文章标签： ctf hgame hgame2022 逆向 writeup
于 2022-02-18 17:21:49 首次发布

文章目录

由于csdn字数限制，有些代码贴不出来。。。

Level - Week1

easyasm

ida分析:

分析汇编得到逻辑如下:

```
for  si  in range(28):
  a=si < 4
  b = si >4
  c = a+b
  c ^ 0x17
```

exp:

```
r=[0x91,0x61,0x01,0xC1,0x41,0xA0,0x60,0x41,0xD1,0x21,0x14,0xC1,0x41,0xE2,0x50,0xE1,0xE2,0x54,0x20,0xC1,0xE2,0x60
,0x14,0x30,0xD1,0x51,0xC0,0x17]
for i in range(28):
 a=r[i] ^ 0x17
 r[i] = (((a&0xf)<<4)|(a>>4))
print(r)
print( bytes(r))
#b'hgame{welc0me_to_4sm_w0rld}\x00'
```

creakme

魔改的tea算法，对照着tea算法修改一下即可。

exp：

```
from ctypes import *

def encrypt(v, k):
    v0 = c_uint32(v[0])
    v1 = c_uint32(v[1])
    summ = c_uint32(0)
    delta = 0x12345678
    k0, k1, k2, k3 = c_uint32(k[2]), c_uint32(k[3]), c_uint32(k[0]), c_uint32(k[1])
    w = [0,0]
    for i in range(32):
        summ.value += delta
        v0.value += summ.value^ ((v1.value << 4) + k0.value) ^ (v1.value + summ.value) ^ ((v1.value >> 5) + k1.v
alue)
        v1.value += summ.value^ ((v0.value << 4) + k2.value) ^ (v0.value + summ.value) ^ ((v0.value >> 5) + k3.v
alue)
    w[0], w[1] = v0, v1
    print("summ.value",summ.value)
    return w
def decrypt(v, k):
    v0 = c_uint32(v[0])
    v1 = c_uint32(v[1])
    summ = c_uint32(1183502080)
    delta = 0x12345678
    k0, k1, k2, k3 = c_uint32(k[2]), c_uint32(k[3]), c_uint32(k[0]), c_uint32(k[1])
    w = [0,0]
    for i in range(32):
        v1.value -= summ.value^ ((v0.value << 4) + k2.value) ^ (v0.value + summ.value) ^ ((v0.value >> 5) + k3.v
alue)
        v0.value -= summ.value^ ((v1.value << 4) + k0.value) ^ (v1.value + summ.value) ^ ((v1.value >> 5) + k1.v
alue)
        summ.value -= delta
```

```
    w[0] = v0.value
    w[1] = v1.value
    return w

def u32(bytes_n):
    #b'11223344' -> 0x11223344
    n = 0
    n += bytes_n[0]
    n += bytes_n[1]<<8
    n += bytes_n[2]<<16
    n += bytes_n[3]<<24
    return n

def p32(int_n):
    #0x11223344 -> b'11223344'
    n = int_n.to_bytes(4,'little')
    return n

def pack_teadata(data):# 将bytes数据转换为 整型数组
    pack=[]
    for i in range(len(data)//4):
        pack.append(u32(data[i*4:i*4+4]))
    return pack

def upack_teadata(data):#将整型数组转换为bytes
    bytes_data = b''
    for i in range(len(data)):
        bytes_data+=(p32(data[i]&0xffffffff))
    return bytes_data

key = [ 1145258561, 1212630597, 1280002633, 1347374669 ]
cipher_int=[1222194312, 51123276, 1391163586, 3986482669, 2921328102, 3126465133, 3482485930, 1709241059]
############ 解密
plain_int=[]
for i in range(len(cipher_int)//2):
    tmp = decrypt(cipher_int[i*2:i*2 +2],key)
    plain_int.append(tmp[0])
    plain_int.append(tmp[1])

plain_dec = upack_teadata(plain_int)
print(plain_dec)
#b'hgame{H4ppy_v4c4ti0n!}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

Flag Checker

jadx分析：

```
public void onClick(View view) {
                byte[] bArr = new byte[0];
                try {
                    bArr = MainActivity.encrypt(((EditText) MainActivity.this.findViewById(R.id.editTextTextPers
onName)).getText().toString(), "carol");
                } catch (Exception e) {
                    e.printStackTrace();
                }
                if (Base64.encodeToString(bArr, 0).replace("\n", "").equals("mg6CITV6GEaFDTYnObFmENOAVjKcQmGncF9
0WhqvCFyhhsyqq1s=")) {
                    Toast.makeText(MainActivity.this, "Congratulations!!!", 1).show();
                } else {
                    Toast.makeText(MainActivity.this, "Fail,try again.", 1).show();
                }
            }
```

```
 public static byte[] encrypt(String str, String str2) throws Exception {
        SecretKeySpec secretKeySpec = new SecretKeySpec(str2.getBytes(), 0, str2.length(), "RC4");
        Cipher instance = Cipher.getInstance("RC4");
        instance.init(1, secretKeySpec);
        return instance.doFinal(str.getBytes());
 }
```

得知为rc4算法。

密钥为：

密文：mg6CITV6GEaFDTYnObFmENOAVjKcQmGncF90WhqvCFyhhsyqq1s=

使用工具解密：hgame{weLC0ME_To-tHE_WORLD_oF-AnDr0|D}

猫头鹰是不是猫

ida分析得知算法过程为

```
cat=[16, 16, 16, 16, 15, 15, 。。。。。]
owl=[15, 14, 14, 13, 13, 14, 。。。。。]
a=[104, 。。。。。。]
r=[]
for k in range(64):
 v3=0
 for m in range(64):
  v3 += a[m]*cat[64*m + k]
 r.append(v3)

rr=[]
for k in range(64):
 v3=0
 for m in range(64):
  v3 += r[m]*owl[64*m + k]
 rr.append(v3)
print(rr)
```

分两步 每一步都需要解64元方程：

```
enc[0] = r[0]*owl[0] + r[1]*owl[64] + r[2]*owl[128] + r[3]*owl[192] +...+ + r[63]*owl[4033]
enc[1] = r[0]*owl[1] + r[1]*owl[65] + r[2]*owl[129] + r[3]*owl[193] +...+ + r[63]*owl[4034]
。。。。。。
```

可以使用z3来解，分两步：

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

14, 14, 14, 9, 8, 7, 10, 14, 12, 9, 11, 11, 13, 12, 8, 6, 7, 8, 12, 10, 9, 11, 14, 12, 8, 6, 10, 11, 11, 17, 19,
14, 14, 16, 18, 18, 18, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 15, 15,
15, 15, 15, 15, 14, 17, 14, 15, 11, 7, 6, 11, 10, 7, 4, 7, 10, 14, 14, 12, 8, 6, 3, 4, 3, 2, 6, 8, 5, 5, 2, 6,
6, 7, 13, 17, 14, 10, 13, 15, 15, 17, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16
, 16, 15, 14, 14, 14, 15, 15, 15, 16, 11, 14, 13, 10, 8, 5, 7, 5, 3, 4, 8, 14, 16, 9, 5, 7, 8, 8, 5, 2, 6, 8, 6,
5, 7, 10, 5, 5, 12, 12, 8, 8, 8, 13, 16, 17, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16
, 16, 16, 16, 15, 14, 14, 14, 14, 14, 15, 10, 8, 12, 11, 16, 7, 5, 3, 4, 2, 2, 5, 9, 13, 10, 3, 4, 10, 13, 10, 5
, 6, 9, 6, 5, 12, 8, 5, 6, 12, 9, 6, 6, 5, 10, 15, 16, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 1
6, 16, 16, 16, 16, 16, 15, 14, 14, 14, 14, 14, 12, 8, 6, 8, 13, 15, 4, 4, 3, 2, 0, 1, 2, 4, 8, 8, 2, 3, 7, 13, 1
1, 10, 7, 9, 5, 9, 11, 6, 4, 10, 10, 8, 4, 7, 4, 9, 13, 16, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 16, 16, 16, 16, 14, 14, 14, 14, 15, 12, 9, 6, 5, 7, 13, 12, 2, 2, 4, 1, 0, 0, 1, 4, 8, 7, 2, 3, 5, 7
, 12, 15, 10, 9, 6, 10, 8, 3, 3, 12, 8, 4, 5, 8, 4, 7, 13, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 1
5, 15, 15, 15, 15, 16, 16, 16, 14, 14, 14, 14, 13, 10, 6, 5, 6, 7, 15, 8, 1, 2, 4, 2, 2, 1, 1, 4, 5, 5, 5, 8, 5,
5, 12, 16, 11, 12, 11, 10, 7, 5, 5, 11, 5, 5, 5, 9, 8, 11, 11, 15, 15, 14, 15, 16, 21, 16, 15, 15, 15, 15, 15,
15, 15, 20, 15, 15, 15, 15, 15, 16, 14, 14, 14, 15, 11, 8, 5, 6, 6, 9, 15, 4, 2, 5, 4, 1, 5, 3, 1, 5, 4, 4, 7, 8
, 6, 7, 11, 16, 13, 15, 12, 9, 7, 8, 11, 8, 7, 7, 5, 9, 12, 12, 10, 18, 15, 14, 15, 21, 15, 15, 15, 15, 15, 15,
15, 15, 15, 21, 15, 15, 15, 15, 15, 16, 14, 14, 14, 13, 8, 7, 5, 5, 8, 14, 11, 5, 11, 9, 2, 4, 7, 3, 2, 4, 2, 3,
6, 8, 7, 8, 11, 12, 16, 16, 13, 11, 8, 9, 12, 10, 7, 12, 4, 9, 16, 10, 8, 18, 15, 14, 18, 18, 15, 15, 15, 15, 1
5, 15, 15, 15, 15, 21, 16, 15, 15, 15, 15, 16, 14, 14, 14, 9, 7, 5, 4, 7, 15, 15, 8, 12, 13, 3, 2, 5, 7, 2, 4, 6
, 2, 2, 5, 9, 11, 11, 12, 13, 14, 17, 16, 12, 11, 12, 13, 10, 9, 15, 3, 7, 16, 13, 8, 17, 16, 15, 22, 14, 15, 15
, 15, 15, 15, 15, 15, 15, 15, 21, 16, 15, 15, 15, 15, 16, 14, 14, 11, 9, 4, 3, 9, 14, 11, 8, 6, 14, 8, 2, 2, 6,
4, 2, 6, 7, 3, 4, 4, 9, 13, 13, 13, 12, 12, 16, 17, 11, 13, 14, 12, 11, 9, 15, 3, 4, 12, 17, 9, 15, 17, 17, 19,
14, 15, 15, 15, 19, 15, 15, 15, 15, 15, 21, 16, 15, 15, 15, 15, 16, 14, 14, 10, 11, 4, 2, 12, 7, 14, 4, 6, 14, 1
2, 4, 1, 4, 3, 4, 5, 7, 5, 6, 3, 6, 12, 12, 11, 10, 12, 12, 17, 11, 14, 16, 8, 9, 10, 17, 7, 5, 7, 17, 10, 16, 1
8, 18, 18, 14, 14, 15, 15, 24, 17, 15, 15, 15, 15, 21, 15, 15, 15, 15, 15, 16, 14, 13, 12, 7, 5, 5, 7, 10, 14, 5
, 5, 10, 14, 3, 1, 4, 6, 4, 2, 8, 8, 7, 2, 5, 8, 10, 10, 9, 12, 10, 16, 11, 13, 13, 6, 7, 10, 17, 14, 7, 5, 13,
9, 16, 18, 15, 22, 18, 16, 17, 19, 21, 20, 15, 15, 15, 15, 22, 15, 15, 15, 15, 15, 16, 14, 13, 15, 5, 1, 7, 5, 1
0, 7, 6, 6, 13, 10, 1, 1, 6, 8, 3, 2, 8, 9, 8, 3, 3, 7, 9, 8, 8, 10, 9, 12, 12, 10, 10, 5, 5, 8, 15, 18, 5, 8, 1
1, 11, 17, 16, 14, 15, 18, 19, 19, 18, 15, 20, 21, 19, 20, 22, 18, 15, 15, 15, 15, 15, 16, 14, 13, 12, 4, 9, 2,
5, 9, 9, 6, 8, 10, 8, 2, 5, 7, 6, 1, 2, 6, 7, 9, 5, 1, 5, 9, 5, 4, 8, 9, 15, 10, 7, 5, 3, 4, 6, 13, 16, 9, 11, 1
5, 15, 19, 16, 14, 14, 14, 14, 14, 15, 15, 15, 16, 17, 17, 15, 15, 15, 15, 15, 15, 15, 15, 15, 14, 13, 10, 5, 9, 3,
5, 8, 13, 9, 11, 11, 10, 6, 5, 6, 4, 2, 3, 4, 4, 10, 8, 1, 3, 10, 7, 3, 8, 10, 10, 10, 3, 3, 4, 3, 7, 11, 13, 10
, 14, 15, 19, 18, 16, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 13, 13, 14, 12
, 12, 5, 6, 13, 14, 11, 11, 10, 8, 4, 5, 4, 3, 2, 6, 3, 1, 8, 11, 3, 2, 7, 9, 6, 8, 12, 8, 9, 2, 5, 5, 3, 9, 9,
11, 12, 16, 15, 18, 16, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 13, 13,
16, 17, 11, 7, 10, 15, 13, 11, 8, 7, 9, 6, 4, 6, 3, 2, 7, 7, 1, 4, 11, 6, 1, 3, 8, 8, 8, 10, 8, 3, 2, 4, 7, 6, 1
0, 5, 10, 16, 21, 20, 17, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 13
, 13, 17, 18, 16, 11, 18, 18, 16, 13, 9, 7, 7, 2, 5, 7, 4, 2, 5, 10, 7, 1, 10, 9, 2, 1, 5, 8, 8, 10, 7, 2, 2, 4,
9, 6, 9, 6, 9, 18, 23, 21, 14, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 13, 13, 17, 18, 16, 14, 22, 19, 16, 10, 8, 8, 4, 2, 5, 7, 7, 4, 3, 9, 13, 10, 6, 11, 5, 0, 3, 8, 10, 10, 4,
1, 1, 6, 5, 6, 9, 11, 10, 17, 22, 17, 13, 13, 13, 13, 14, 14, 14, 14, 20, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15
, 15, 15, 15, 13, 13, 18, 19, 17, 14, 19, 12, 12, 10, 10, 6, 4, 4, 7, 8, 8, 5, 2, 8, 11, 11, 9, 12, 9, 3, 3, 8,
10, 6, 3, 1, 3, 5, 2, 8, 11, 10, 13, 14, 18, 15, 13, 13, 13, 13, 13, 14, 14, 14, 20, 14, 14, 14, 14, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 13, 13, 16, 19, 18, 16, 17, 9, 11, 11, 9, 6, 7, 7, 8, 8, 6, 5, 1, 5, 10, 10, 8, 6, 7, 5
, 4, 8, 7, 2, 3, 3, 3, 3, 4, 7, 10, 10, 13, 13, 14, 13, 13, 13, 13, 13, 13, 13, 13, 14, 17, 18, 14, 14, 14, 14, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 13, 13, 15, 20, 19, 15, 18, 11, 12, 10, 9, 4, 7, 6, 7, 7, 6, 4, 2, 3, 7, 10, 10,
7, 7, 4, 7, 7, 3, 3, 3, 3, 3, 5, 6, 9, 9, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 19, 15, 14, 14, 14, 1
4, 15, 14, 15, 15, 15, 15, 15, 15, 15, 13, 13, 13, 19, 20, 14, 17, 14, 13, 11, 10, 4, 7, 4, 6, 6, 7, 5, 3, 4, 5,
8, 10, 10, 5, 1, 5, 2, 1, 3, 4, 2, 4, 7, 9, 8, 13, 14, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 13, 21, 14, 14,
14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 13, 13, 13, 14, 15, 13, 17, 17, 13, 13, 10, 3, 6, 5, 7, 6, 5, 4,
4, 3, 5, 5, 7, 10, 9, 5, 4, 3, 4, 3, 1, 3, 8, 13, 9, 8, 15, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 15, 21,
14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 14, 14, 14, 15, 13, 13, 13, 13, 13, 13, 14, 18, 12, 8, 9, 3, 6, 6, 8, 8
, 5, 2, 3, 2, 2, 6, 8, 7, 6, 6, 5, 3, 1, 2, 3, 8, 13, 11, 7, 16, 14, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
17, 18, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 13, 13, 13, 13, 13, 13, 13, 16, 11, 5, 7, 5, 6,
6, 7, 7, 8, 3, 1, 4, 4, 4, 6, 9, 8, 6, 6, 4, 1, 3, 8, 10, 7, 8, 14, 14, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 19, 16, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 18, 13, 13, 12, 13, 13, 13, 13, 13, 12, 8,
7, 9, 6, 3, 4, 6, 8, 7, 2, 2, 7, 7, 7, 7, 7, 6, 5, 3, 0, 4, 6, 3, 6, 14, 14, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 21, 14, 14, 14, 16, 14, 14, 14, 14, 14, 14, 14, 14, 15, 19, 20, 13, 12, 13, 13, 13, 13, 13, 13,
12, 11, 10, 9, 10, 5, 1, 2, 3, 7, 5, 2, 3, 6, 6, 6, 6, 5, 3, 1, 2, 3, 3, 5, 9, 14, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 21, 14, 14, 14, 21, 14, 14, 14, 16, 18, 20, 18, 15, 18, 21, 21, 13, 13, 13, 13, 13,

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

5, 2, 2, 3, 14, 14, 14, 13, 13, 6, 6, 5, 7, 7, 7, 6, 5, 4, 4, 4, 4, 5, 9, 17, 19, 16, 5, 3, 2, 2, 2, 10, 13, 11,
4, 3, 3, 1, 2, 2, 1, 0, 0, 0, 0, 1, 3, 1, 3, 0, 1, 3, 6, 7, 5, 4, 4, 5, 5, 5, 4, 11, 10, 7, 4, 3, 3, 5, 14, 14,
13, 13, 13, 6, 6, 6, 6, 6, 6, 6, 5, 4, 3, 3, 3, 4, 6, 8, 12, 11, 6, 2, 2, 2, 2, 2, 4, 3, 1, 1, 2, 3, 2, 3, 2, 1
, 1, 1, 2, 2, 2, 1, 2, 1, 0, 3, 12, 16, 15, 10, 6, 4, 5, 5, 4, 12, 12, 10, 8, 6, 6, 8, 14, 13, 13, 13, 12, 6, 6,
7, 6, 6, 6, 5, 4, 3, 3, 3, 3, 3, 4, 3, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 3, 3, 3, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2,
2, 4, 1, 2, 6, 11, 17, 17, 11, 8, 5, 5, 4, 11, 13, 11, 11, 9, 8, 9, 14, 13, 13, 13, 12, 5, 4, 7, 6, 6, 5, 4, 2,
2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 2, 3, 3, 3, 3, 2, 1, 1, 2, 3, 3, 2, 3, 3, 3, 4, 4, 4, 4
, 11, 16, 14, 8, 5, 4, 10, 11, 11, 12, 11, 9, 8, 13, 13, 13, 13, 10, 2, 3, 5, 6, 6, 4, 3, 2, 2, 2, 1, 1, 1, 2, 1
, 1, 1, 1, 2, 1, 1, 1, 2, 2, 3, 2, 3, 3, 3, 3, 2, 1, 1, 1, 1, 1, 3, 3, 2, 3, 2, 3, 4, 5, 5, 4, 2, 14, 17, 14, 5,
4, 9, 10, 10, 11, 11, 9, 9, 12, 12, 12, 12, 10, 4, 3, 3, 5, 5, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 3, 2, 2, 3, 5, 3, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 1, 1, 2, 3, 5, 5, 5, 11, 19, 17, 6, 5, 11, 12, 11, 11,
11, 10, 10, 12, 12, 12, 12, 9, 5, 4, 2, 4, 4, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 2, 2, 2, 4, 3, 2,
2, 3, 2, 2, 2, 2, 3, 2, 2, 3, 1, 1, 2, 3, 3, 2, 3, 4, 5, 6, 8, 17, 19, 7, 5, 12, 12, 12, 11, 11, 11, 11, 11, 11,
11, 11, 9, 3, 3, 4, 3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 2
, 2, 2, 1, 0, 1, 2, 3, 4, 3, 3, 3, 4, 5, 5, 7, 12, 6, 5, 11, 11, 11, 11, 11, 11, 11, 12, 12, 13, 12, 9, 2, 2, 3,
3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 3, 3, 2, 2, 1, 2, 2, 1, 1, 1,
2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 5, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 12, 3, 2, 2, 2, 1, 1, 1, 1, 1, 1,
0, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 4, 5, 8, 10, 11, 12, 8, 5, 3, 2, 2, 3, 2, 2,
3, 4, 4, 3, 2, 2, 2, 2, 4, 11, 12, 12, 12, 13, 14, 14, 14, 14, 14, 13, 3, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 0, 1,
1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 3, 10, 13, 14, 14, 15, 15, 17, 16, 14, 12, 11, 8, 3, 2, 2, 2, 3, 3, 3
, 2, 2, 2, 3, 10, 12, 12, 13, 13, 14, 14, 15, 15, 14, 13, 4, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2,
2, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 9, 12, 17, 18, 17, 16, 15, 17, 17, 16, 16, 15, 16, 15, 4, 2, 2, 2, 3, 3, 3, 2,
2, 3, 10, 13, 13, 13, 14, 15, 15, 15, 15, 15, 13, 4, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1,
1, 0, 1, 1, 1, 1, 1, 1, 2, 8, 12, 18, 19, 17, 15, 17, 18, 18, 18, 18, 17, 15, 4, 2, 2, 2, 2, 3, 2, 2, 2, 3, 10,
14, 14, 14, 14, 16, 15, 15, 15, 14, 11, 2, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 2, 9, 12, 14, 16, 14, 17, 16, 16, 17, 15, 13, 10, 1, 2, 1, 2, 2, 2, 3, 2, 1, 3, 11, 13, 13, 1
4, 15, 16, 15, 15, 15, 14, 7, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1
, 0, 0, 1, 3, 9, 11, 12, 13, 13, 12, 12, 12, 12, 10, 3, 2, 1, 1, 1, 1, 1, 2, 2, 2, 4, 13, 15, 14, 14, 15, 16, 15
, 15, 15, 12, 4, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 2,
2, 6, 9, 11, 10, 11, 11, 10, 9, 4, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 4, 12, 14, 15, 15, 16, 17, 16, 15, 16, 16, 3,
1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 2, 1, 2, 4, 6, 6
, 5, 3, 1, 0, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 4, 13, 14, 13, 15, 16, 16, 18, 19, 19, 18, 6, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 2, 0, 0, 0, 0, 0, 0, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 5, 14, 15, 15, 15, 15, 19, 19, 19, 19, 19, 12, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 0, 1, 2, 1, 0, 0, 0, 0, 1, 1, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
6, 12, 16, 16, 16, 16, 19, 19, 19, 19, 19, 16, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 1, 0, 1
, 0, 0, 0, 1, 0, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 8, 17, 15, 16, 18, 1
8, 19, 20, 20, 20, 20, 16, 3, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1
, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 12, 18, 20, 18, 17, 20, 19, 20, 20, 20, 1
9, 17, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 12, 20, 20, 20, 20, 18, 19, 19, 20, 19, 19, 18, 10, 1, 1, 1,
1, 1, 0, 0, 1, 0, 0, 2, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 0, 0, 1, 1, 1, 1, 1, 1, 2, 5, 16, 17, 20, 20, 21, 21, 18, 18, 18, 19, 19, 18, 12, 2, 1, 1, 1, 1, 1, 1, 0, 0, 1
, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, 1, 9, 18, 19, 18, 21, 20, 21, 13, 14, 15, 15, 17, 18, 15, 4, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 2, 14, 20, 1
9, 20, 19, 21, 20, 9, 12, 12, 9, 11, 16, 14, 6, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1
, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 3, 16, 21, 21, 19, 20, 18, 21, 1
0, 10, 10, 10, 12, 13, 13, 7, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0
, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 10, 17, 20, 21, 20, 19, 20, 19, 11, 12, 14, 14, 16
, 17, 13, 8, 2, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 15, 20, 19, 20, 21, 20, 20, 20, 17, 18, 17, 19, 20, 19, 18, 12, 2,
1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 6, 19, 20, 19, 20, 20, 21, 20, 21, 19, 18, 20, 20, 20, 20, 19, 13, 3, 1, 2, 1, 1, 1, 0
, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 2, 15, 20, 21, 20, 19, 21, 21, 22, 20, 19, 19, 20, 20, 20, 19, 18, 14, 3, 2, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0
, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 5, 20, 2
1, 21, 21, 19, 21, 22, 22, 21, 20, 20, 20, 20, 19, 19, 18, 13, 3, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, 21, 21, 21, 21, 21,
19, 22, 22, 22, 20, 20, 20, 20, 19, 19, 17, 9, 2, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 14, 22, 22, 22, 22, 21, 20, 22, 22, 22,
 19, 20, 20, 20, 20, 18, 13, 4, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 18, 22, 22, 22, 21, 21, 21, 20, 22, 22, 15, 16, 17, 17
, 17, 14, 7, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 12, 22, 24, 23, 22, 23, 23, 21, 20, 22, 22, 13, 12, 12, 11, 11, 8, 3, 1,
1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 1, 1, 1, 1, 1, 2, 21, 25, 25, 25, 25, 25, 25, 21, 20, 20, 21, 13, 12, 12, 10, 7, 3, 1, 1, 1, 1, 1, 1, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
1, 5, 25, 25, 25, 25, 25, 25, 25, 21, 20, 17, 18, 12, 12, 11, 8, 5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11, 25, 25, 25,
 25, 25, 25, 24, 19, 20, 13, 12, 12, 11, 9, 5, 2, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 19, 25, 25, 25, 25, 25, 25, 23, 2
5, 25, 24, 15, 11, 9, 6, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 10,
6, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 7, 3, 2, 1, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 1, 22, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 5, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10,
 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 3, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 21, 25, 25, 25, 25, 2
5, 25, 25, 25, 25, 25, 25, 25, 25, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
 25, 25, 25, 25, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 13, 25, 25, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25]
enc=[39654868, 38564788, 37998449, 36705100, 34811134, 29241290, 21154342, 14031189, 9709480, 10192500, 9488434,
 8917703, 8436312, 7963105, 7514984, 7654219, 7487295, 8176216, 9031004, 8970914, 9341190, 9144480, 9366669, 938
5331, 9139311, 9514114, 9272562, 8593561, 8340901, 6839039, 5285074, 5402594, 5804603, 5414593, 5336346, 5993555
, 7350502, 9029370, 10868398, 13859348, 12111786, 11582283, 11430308, 10556747, 10341351, 10607648, 11143139, 11
689137, 11730328, 11611945, 14952608, 18555268, 23156514, 27591411, 29403805, 30676639, 32476022, 36955579, 4126
4880, 44681781, 44601420, 44143855, 43151161, 42898115]

#z3分步求解
from z3 import *
s=Solver()
r = [ Int(f'a[{i}]') for i in range(64)]
rr=[]
for k in range(64):
 v3=0
 for m in range(64):
  v3 += r[m]*owl[64*m + k]
 s.add(v3==enc[k])

print(s.check())
m=s.model()
print(m)

#得到一组解:
enc[33] = 38575
enc[36] = 46137
enc[9] = 42574
enc[35] = 40920
enc[19] = 32015
enc[12] = 38813
enc[27] = 32739
enc[29] = 34208
enc[13] = 33034
enc[34] = 37661
enc[16] = 29057
```

```
enc[24] = 33438
enc[28] = 34235
enc[17] = 26765
enc[20] = 34769
enc[11] = 40936
enc[2] = 50090
enc[10] = 43020
enc[1] = 49684
enc[7] = 46797
enc[41] = 53609
enc[30] = 32414
enc[0] = 50409
enc[6] = 47074
enc[15] = 32345
enc[23] = 32113
enc[21] = 34933
enc[31] = 32762
enc[22] = 32060
enc[8] = 45297
enc[43] = 52995
enc[4] = 47646
enc[3] = 49395
enc[37] = 48875
enc[14] = 32512
enc[58] = 58319
enc[5] = 44689
enc[25] = 34385
enc[18] = 27745
enc[32] = 34717
enc[26] = 34094
enc[40] = 51468
enc[63] = 60254
enc[51] = 55296
enc[50] = 54881
enc[44] = 52056
enc[52] = 54632
enc[45] = 51479
enc[38] = 45934
enc[39] = 47834
enc[61] = 58052
enc[62] = 58623
enc[56] = 55535
enc[59] = 58074
enc[47] = 52533
enc[55] = 56164
enc[42] = 50819
enc[60] = 57949
enc[57] = 59558
enc[54] = 54943
enc[53] = 53955
enc[48] = 55281
enc[49] = 55774
enc[46] = 52784

#继续z3求解
s=Solver()
r = [ Int(f'a[{i}]') for i in range(64)]
rr=[]
for k in range(64):
 v3=0
```

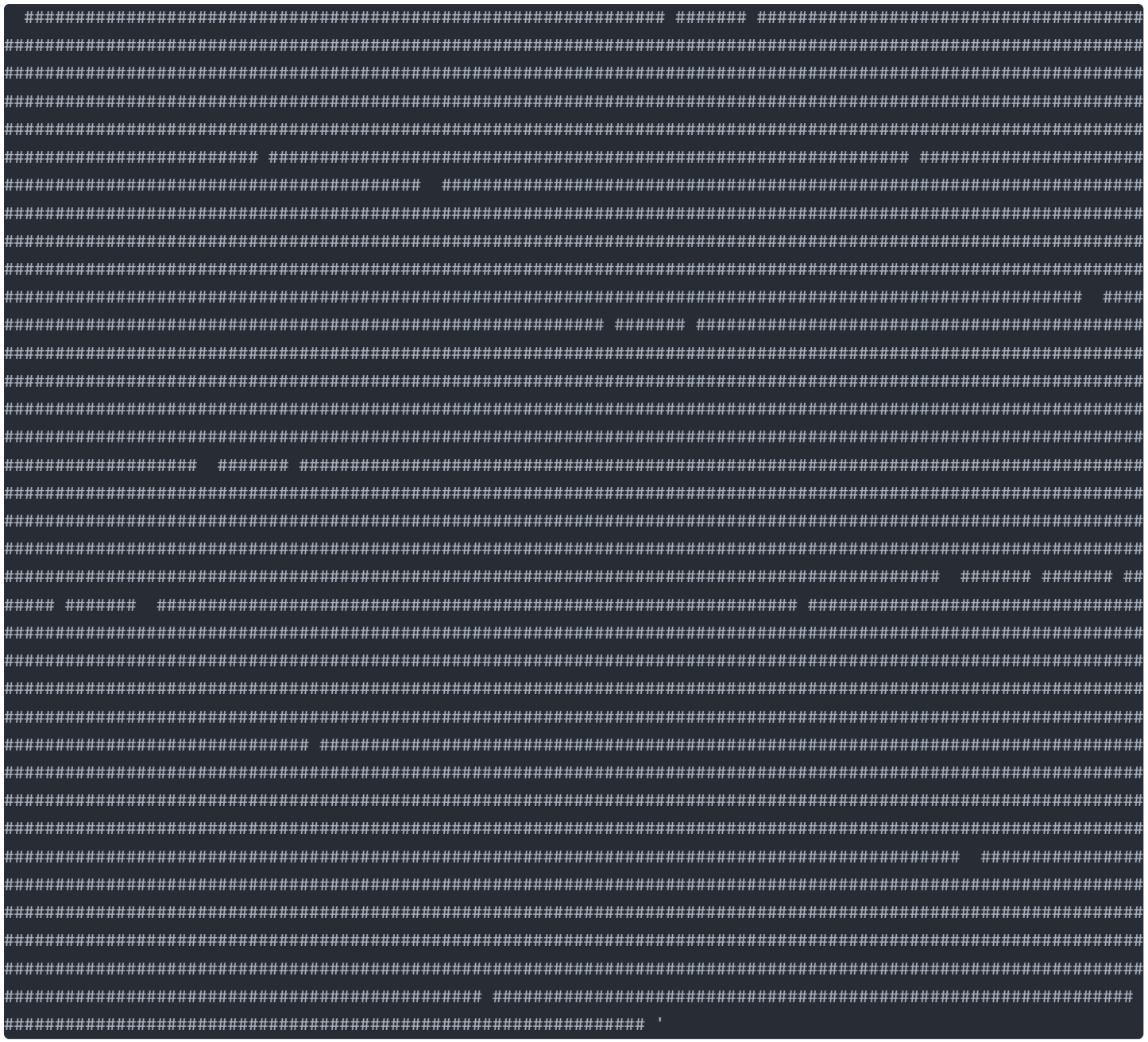
```
 for m in range(64):
  v3 += r[m]*cat[64*m + k]
 s.add(v3==enc[k])

print(s.check())
m=s.model()
print(m)

#得到flag的数组：

a=[ 0 for i in range(64)]
a[33] = 49
a[36] = 48
a[44] = 48
a[62] = 49
a[51] = 48
a[27] = 48
a[40] = 49
a[29] = 48
a[54] = 48
a[50] = 48
a[34] = 49
a[16] = 48
a[39] = 48
a[28] = 48
a[20] = 48
a[49] = 48
a[61] = 49
a[26] = 48
a[11] = 49
a[2] = 97
a[57] = 48
a[41] = 48
a[60] = 49
a[0] = 104
a[6] = 49
a[23] = 48
a[31] = 48
a[59] = 49
a[48] = 49
a[22] = 48
a[55] = 49
a[3] = 109
a[37] = 48
a[56] = 49
a[5] = 123
a[63] = 125
a[58] = 48
a[53] = 48
a[32] = 48
a[9] = 48
a[35] = 48
a[38] = 49
a[45] = 48
a[19] = 49
a[12] = 49
a[13] = 48
a[24] = 49
a[42] = 49
```

```
a[10] = 49
a[1] = 103
a[7] = 48
a[46] = 48
a[30] = 48
a[15] = 48
a[21] = 48
a[43] = 49
a[8] = 48
a[4] = 101
a[14] = 48
a[25] = 48
a[18] = 49
a[17] = 48
a[47] = 48
a[52] = 49
print(bytes(a))
#hgame{10001110000011000010000000011000101011000010001001100111}
```

Level - Week2

xD MAZE

一维迷宫,根据空格的步长，还原即可。



exp:

```
m=[32, ] #csdn字数限制，这里就不写全了
m=bytes(m)
print(bytes(m))
last=0
flag=""
m=m[1:]
for i in range(0,28):
 a=m.find(b' ')
 print(a)
 if a==0:
  flag +='3'
 if a==7:
  flag +='2'
 if a==63:
  flag +='1'
 if a==511:
  flag +='0'
 m=m[a+1:]

print(flag)

#hgame{31201130312032032222231003011}
```

upx magic 0

没有upx壳。。。算法爆破：

```
r=[36200, 40265, 10770, 43802, 52188, 47403, 11826, 40793, 56781, 40265, 43274, 3696, 62927, 2640, 23285, 65439,
 40793, 48395, 22757, 14371, 48923, 30887, 43802, 18628, 43274, 11298, 40793, 23749, 24277, 30887, 9842, 22165]

def calcc(data):
 data = ord(data)
 data = data<<8
 for j in range(8):
   if  (data & 0x8000) != 0 :
     data = (2 * data) ^ 0x1021;
   else:
     data *= 2;
 data = data &0xffff
 return data
import string
flag=""
for i in r:
 for t in  string.printable:
  if calcc(t) == i:
   flag+=t
   break
print(flag)
#noW_YOu~koNw-UPx~mAG|C_@Nd~crC16
```

fake shell

ida分析，需要执行 sudo cat flag.txt

但是在sudo中有校验密码

校验逻辑为：

1、rc4加密，密钥为w0wy0ugot1t（有反调试，需要运行后attach上去）。

2、rc4密钥与内存数据对比。

解密过程为：

1、计算rc4密文：

```
v7=[1,1,1,1]
v7[0] = 0xE0B25F3D8FFA94B6
v7[1] = 0xE79D6C9866D20FEA
v7[2] = 0x6D6FBEC57140081B
v7[3] = 0xF6F3BDA88D097B7C

r=b''
for i in v7:
    a=hex(i)[2:]
    a=bytes.fromhex(a)[::-1]
    r+=a
print(r.hex())
#b694fa8f3d5fb2e0ea0fd266986c9de71b084071c5be6f6d7c7b098da8bdf3f6
```

2、rc4解密：

密钥：w0wy0ugot1t

密文：b694fa8f3d5fb2e0ea0fd266986c9de71b084071c5be6f6d7c7b098da8bdf3f6

得到：hgame{s0meth1ng_run_bef0r_m4in?}

creakme2

更加复杂的魔改TEA。

```
#encoding=utf-8
import sys
from ctypes import *

def u32(bytes_n):
    n = 0
    n += bytes_n[0]
    n += bytes_n[1]<<8
    n += bytes_n[2]<<16
    n += bytes_n[3]<<24
    return n

def p32(int_n):
    #0x11223344 -> b'11223344'
    n = int_n.to_bytes(4,'little')
    return n

def pack_teadata(data):# 将bytes数据转换为 整型数组
    pack=[]
    for i in range(len(data)//4):
        pack.append(u32(data[i*4:i*4+4]))
    return pack

def upack_teadata(data):#将整型数组转换为bytes
    bytes_data = b''
    for i in range(len(data)):
        bytes_data+=(p32(data[i]&0xffffffff))
    return bytes_data
```

```
def tea_enc(v, k):
    y = c_uint32(v[0])
    z = c_uint32(v[1])
    sum = c_uint32(0)
    delta = 0x9E3779B1
    n = 32
    w = [0,0]

    while(n>0):
        y.value += (k[sum.value & 3] + sum.value) ^ (z.value + ((z.value >> 5) ^ (z.value << 4)))
        sum.value += delta
        if sum.value & 0x80000000 == 0:
            sum.value ^= 0x1234567
        z.value += (k[(sum.value >> 11) & 3] + sum.value) ^ (y.value + ((y.value >> 5) ^ (16 * y.value)))
        n -= 1
    w[0] = y.value
    w[1] = z.value
    print(hex(sum.value))
    return w

def tea_dec(v, k):
    y = c_uint32(v[0])
    z = c_uint32(v[1])
    sum = c_uint32(0xc78e4d05)
    delta = 0x9E3779B1
    n =32
    w = [0,0]

    while(n>0):
        z.value -= (k[(sum.value >> 11) & 3] + sum.value) ^ (y.value + ((y.value >> 5) ^ (16 * y.value)))
        if sum.value & 0x80000000 == 0:
            sum.value ^= 0x1234567
        sum.value -= delta
        y.value -= (k[sum.value & 3] + sum.value) ^ (z.value + ((z.value >> 5) ^ (z.value << 4)))
        n -= 1
    w[0] = y.value
    w[1] = z.value
    return w
Buf2=[1,1,1,1,1,1,1,1]
Buf2[0] = 0x457E62CF
Buf2[1] = 0x9537896C
Buf2[2] = 0x1F7E7F72
Buf2[3] = 0xF7A073D8
Buf2[4] = 0x8E996868
Buf2[5] = 0x40AFAF99
Buf2[6] = 0xF990E34
Buf2[7] = 0x196F4086

key  = [1,2,3,4,5,6,7,8,9,0]
flag=b''
for i in range(0,8,2):
    p = tea_dec(Buf2[i:],key)
    a=upack_teadata(p)
    flag+=a
print(flag)
#b'hgame{SEH_s0und5_50_1ntere5ting}'
```

upx magic 1

upx手动脱壳，脱壳后算法同upx magic 0

```
v14 = [0 for i in range(37)]
v14[0] = 36200;
v14[1] = 40265;
v14[2] = 10770;
v14[3] = 43802;
v14[4] = 52188;
v14[5] = 47403;
v14[6] = 11826;
v14[7] = 40793;
v14[8] = 56781;
v14[9] = 40265;
v14[10] = 43274;
v14[11] = 3696;
v14[12] = 62927;
v14[13] = 24277;
v14[14] = 15363;
v14[15] = 31879;
v14[16] = 9842;
v14[17] = 43802;
v14[18] = 2640;
v14[19] = 23285;
v14[20] = 65439;
v14[21] = 40793;
v14[22] = 48395;
v14[23] = 22757;
v14[24] = 14371;
v14[25] = 48923;
v14[26] = 30887;
v14[27] = 43802;
v14[28] = 18628;
v14[29] = 43274;
v14[30] = 11298;
v14[31] = 40793;
v14[32] = 23749;
v14[33] = 24277;
v14[34] = 30887;
v14[35] = 9842;
v14[36] = 22165;

def checkval(data):
  v12 = ord(data) << 8
  for j in range(8):
    if (v12 & 0x8000) != 0:
      v12 = (2 * v12)^ 0x1021
    else:
      v12 = 2 * v12
  print(data,hex(v12&0xffff))
  return v12&0xffff

import string

def checkval1(dest):
  for i in string.printable:
    if checkval(i) == dest:
      return i
  return 0
```

```
flag=''
for i in range(37):
  flag+=checkval1(v14[i])
print(flag)

#noW_YOu~koNw-rea1_UPx~mAG|C_@Nd~crC16
```

Level - Week3

Answer's Windows

换表base64：

```
import base64
a = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'    #标准表
b = [33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
 88, 89, 90, 91, 92, 93, 94, 95, 96]
b = bytes(b).decode()

c = ";'>B<76\\=82@-8.@=T\"@-7ZU:8*F=X2J<G>@=W^@-8.@9D2T:49U@1aa"
trantab = c.maketrans(b, a)
print(base64.b64decode(c.translate(trantab)))

#b'hgame{qt_1s_s0_1nteresting_so_1s_b4se64}\x06\x9a'
```

creakme3

IDA只能看到汇编，Ghidra可以看到伪代码。

exp

```
a=[48, 48, 48, 48, 48, 49, 50, 50, 50, 50, 50, 51, 51, 51, 51, 51, 51, 51, 51, 51, 53, 56, 56, 57, 57, 57, 57, 6
6, 95, 95, 95, 95, 97, 100, 100, 100, 100, 100, 101, 101, 102, 102, 102, 102, 102, 102, 102, 103, 103, 104, 104,
 104, 104, 104, 105, 105, 105, 106, 106, 107, 107, 108, 109, 110, 110, 110, 111, 111, 111, 112, 114, 114, 114, 1
15, 115, 115, 115, 115, 115, 115, 116, 116, 116, 117, 117, 119, 119, 123, 125]
b=[20093, 26557, 31304, 33442, 37694, 39960, 23295, 27863, 42698, 48505, 52925, 12874, 12946, 14597, 17041, 2326
2, 28319, 42282, 48693, 52067, 32571, 14612, 45741, 14554, 20048, 27138, 45327, 30949, 32502, 35235, 36541, 3837
1, 29658, 21388, 25403, 40604, 46987, 51302, 12974, 30329, 10983, 19818, 22280, 26128, 41560, 47116, 51333, 2893
8, 31988, 16246, 28715, 41966, 44368, 47815, 16420, 35362, 49237, 11090, 50823, 24320, 50199, 24962, 30171, 1545
7, 18838, 24001, 11638, 32023, 43291, 39661, 17872, 33895, 43869, 20611, 25122, 36243, 37434, 38686, 46266, 5107
7, 13656, 34493, 38712, 14096, 38777, 12095, 17629, 30945, 40770]
c=[20093, 26557, 31304, 33442, 37694, 39960, 23295, 27863, 42698, 48505, 52925, 12874, 12946, 14597, 17041, 2326
2, 28319, 42282, 48693, 52067, 32571, 14612, 45741, 14554, 20048, 27138, 45327, 30949, 32502, 35235, 36541, 3837
1, 29658, 21388, 25403, 40604, 46987, 51302, 12974, 30329, 10983, 19818, 22280, 26128, 41560, 47116, 51333, 2893
8, 31988, 16246, 28715, 41966, 44368, 47815, 16420, 35362, 49237, 11090, 50823, 24320, 50199, 24962, 30171, 1545
7, 18838, 24001, 11638, 32023, 43291, 39661, 17872, 33895, 43869, 20611, 25122, 36243, 37434, 38686, 46266, 5107
7, 13656, 34493, 38712, 14096, 38777, 12095, 17629, 30945, 40770]

c.sort()
print(c)
f=[]
for i in c:
 idx=b.index(i)
 f.append(a[idx])
print(bytes(f))
#b'fjow33etu938nhi3wrnf90sdf32nklsdf0923hgame{B0go_50rt_is_s0_stup1d}fh32orh98sdfh23ikjsdf32'
#hgame{B0go_50rt_is_s0_stup1d}
```

hardened

加了壳的apk，先用FRIDA-DEXDump脱壳。

java源码：

```
    public void sendPwd(View view) {
        Intent intent = new Intent(this, rightpage.class);
        if (bbbbb(aesEncryption(((EditText) findViewById(2131165238)).getText().toString().getBytes())).equals("
mXYxnHYp61u/5qksdDel6TgiKqcvUbBkX3xErlR4lO0aEAdU0acJY8PRSVXJxxsRR8Dq9MTJhkWLSbBvCG5gtm=="
            startActivity(intent);
        } else {
            Toast.makeText(this, "fail >__<", 1).show();
        }
    }
```

bbbbb和aesEncryption函数实现在libenc.so中，bbbbb为换表base64，aesEncryption为AES CBC加密。

分析 aesEncryption 得到key和iv计算发现不正确，调试时发现key iv 还有 base64表数据不同。

当时怀疑时不同的区域异或了一个数据，但是没有深入分析，爆破了一下key 和iv的数据。

```
from Crypto.Cipher import AES
import base64

key=[122, 101, 99, 100, 111, 113, 111, 126, 127, 98, 125, 113, 124, 111, 123, 117, 105, 111, 118, 127, 98, 111,
105, 127, 101, 111, 100, 127, 111, 116, 117, 115]
iv=[6, 16, 10, 32, 25, 22, 17, 27, 32, 18, 26, 94, 94, 94, 94, 94]
msg_enc=[114,92,-80,1,-41,-2,-4,2,115,85,18,-114,43,-19,39,-29,19,-54,-71,-66,101,46,-87,-90,-5,1,-102,-20,92,14
,-9,36]
for i in range(32):
 msg_enc[i] = msg_enc[i] & 0xff
msg_enc = bytes(msg_enc)

msg_p=b'1234567812345678'
for i in range(0xff):
 tkey=[]
 for x in key:
  tkey.append(x ^ i)
 tkey=bytes(tkey)
 for j in range(0xff):
  tiv=[]
  for y in iv:
   tiv.append(y ^ j)
  tiv=bytes(tiv)

  aes = AES.new(tkey,AES.MODE_CBC,tiv)
  msg = aes.encrypt(msg_p)
  if msg[:16] == msg_enc[:16]:
   print(i,j)
   print(tkey,tiv)

key=b'JUST_A_NORMAL_KEY_FOR_YOU_TO_DEC'
iv=b'you_find_me!!!!!'

a = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'     #标准表
b = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz+/'    #新表
c = 'mXYxnHYp61u/5qksdDel6TgiKqcvUbBkX3xErlR4lO0aEAdU0acJY8PRSVXJxxsRR8Dq9MTJhkWLSbBvCG5gtm=='
trantab = c.maketrans(b, a)

msg_enc=base64.b64decode(c.translate(trantab))
aes = AES.new(key,AES.MODE_CBC,iv)
msg = aes.decrypt(msg_enc)
print(msg.decode())

#48 127
#b'JUST_A_NORMAL_KEY_FOR_YOU_TO_DEC' b'you_find_me!!!!!'
#hgame{cONGraTUl4T|0N5!N0w_yoU_C4n_eN?Oy~thE~MUsIc}
```

Level - Week4

(WOW)

程序本身存在解密函数，在数据对比后，又对输入数据的密文进行了解密。只要把flag密文复制 输入密文那即可。

1、在结果比较前下断点，在return时下断点，随便输入。
2、读取32个字节的密文标准数据，patch Buf2的内容。继续执行。
3、在return时，v10中存放flag

server

这个题拿到了一血。

跟踪调试得到加密过程为

```
import gmpy2
from Crypto.Util.number import bytes_to_long,long_to_bytes

p=92582184765240663364795767694262273105045150785272129481762171937885924776597
q=10731052865803998570889663655911240033426200536764917674642953127430085949899 3

n=p*q
e=950501
m=b'hgame{12123333333333333333333333333333333333333}'
m=bytes_to_long(m)
c=pow(m,e,n)
print(hex(c), c)
cc=str(c).encode().ljust(153,b'\0')
cc =b'480856246755930918853646960284513996465042625507755175073557975588365361488153082402793738425677614000307090113926819061232387511326704622862441103917794'
tmp=0x66
dd=[]
for i in range(0,153):
 dd.append(cc[i] ^ tmp)
 tmp=cc[i]
 #print(i,hex(tmp))
#print(hex(tmp), dd)
ee=[]
for i in range(0,153):
 ee.append(dd[i] ^ tmp)
 tmp=dd[i]
```

得到算法的核心为：

```
假设明文：0123456
第一步：0^0x66  1^0 2^1 3^2 4^3 5^4 6^5
第二步：0^0x66^6 1^0^0^0x66  2^1^1^0 3^2^2^1  4^3^3^2  5^4^4^3  6^5^5^4
一二步可以简化为：  0^0x66^6 1^0x66  2^0 3^1  4^2  5^3  6^4
```

1、3、5。。。奇数位置可以直接解：

```
enc[1]=c[1]^0x66
enc[3]=c[3]^c[1]
enc[5]=c[5]^c[3]
...
得到：
c[1]=enc[1]^0x66
c[3]=enc[3]^c[1]
c[5]=enc[5]^c[3]
...
```

但是偶数位置因为 enc[0]=c[0]^{0x66}c[6] 里面 c[0]和c[6]均未知，但是计算出enc[0]后续enc[2、4、6]都可以计算。尝试使用爆破c[6]的方式爆破出c即可。c为纯数字。

```
import gmpy2
from Crypto.Util.number import bytes_to_long,long_to_bytes

p=92582184765240663364795767694262273105045150785272129481762171937885924776597
q=10731052865803998570889663655911240033426200536764917674642953127430085949899 3
n=p*q
e=950501
ccc=[99, 85, 4, 3, 5, 5, 5, 3, 7, 7, 2, 8, 8, 11, 1, 2, 10, 4, 2, 13, 8, 9, 12, 9, 4, 13, 8, 0, 14, 0, 15, 13, 1
4, 10, 2, 2, 1, 7, 3, 5, 6, 4, 6, 7, 6, 2, 2, 5, 3, 3, 9, 6, 0, 11, 13, 11, 0, 2, 3, 8, 3, 11, 7, 1, 11, 5, 14,
5, 0, 10, 14, 15, 13, 7, 13, 7, 14, 1, 15, 1, 11, 5, 6, 2, 12, 6, 10, 4, 1, 7, 4, 2, 6, 3, 6, 12, 5, 12, 3, 12,
6, 0, 4, 15, 2, 14, 7, 0, 14, 14, 12, 4, 3, 4, 2, 0, 0, 2, 6, 2, 3, 6, 4, 4, 4, 7, 1, 2, 3, 9, 2, 12, 8, 1, 12,
3, 12, 2, 0, 3, 14, 3, 14, 12, 9, 1, 7, 15, 5, 7, 2, 2, 4, 102, 94]

tmp=0x66
for i in range(1,153,2):
 ccc[i] ^= tmp
 tmp = ccc[i]
 #print(i, chr(tmp))

print(hex(tmp))
for j in range(0x30,0x3a):
 tmp=j
 tmp ^= 0x66
 cct=[i for i in ccc]
 for i in range(0,153,2):
  cct[i] ^= tmp
  tmp = cct[i]
 a=bytes(cct)
 print(a)

c=13500556210982903419905914947489634156630760022714828952506853229772789740977687325096322567046834086827097997
536747452711551200391594579596759908772002 4
#c=34015463119928024098049048464997351467317701237049299424078433287626887508766972240862235771478241878371969874377575537014502102905844785866589186730125
d=gmpy2.invert(e,(p-1)*(q-1))
m=pow(c,d,n)
print(long_to_bytes(m))

#b'hgame{g0_and_g0_http_5erv3r_nb}'
```

ezvm

vm虚拟机

分析每个指令的功能，一步步跟踪下逻辑：

1、初始化几个寄存器。

2、输入flag以回车结束

3、对flag长度判断是否是0x20。

4、循环处理flag每个字符：

```
flag[i]  *=2
flag[i] ^= salt[i]
```

5、同结果比较。

复现算法：

```
flag=b'hgame{12345678901234567890abcde}'
flag=list(flag)
salt=[94, 70, 97, 67, 14, 83, 73, 31, 81, 94, 54, 55, 41, 65, 99, 59, 100, 59, 21, 24, 91, 62, 34, 80, 70, 94, 5
3, 78, 67, 35, 96, 59]
for i in range(0x20):
 flag[i]  *=2
 flag[i] ^= salt[i]
print(flag)

#enc=[142, 136, 163, 153, 196, 165, 195, 221, 25, 236, 108, 155, 243, 27, 139, 91, 62, 155, 241, 134, 243, 244,
164, 248, 248, 152, 171, 134, 137, 97, 34, 193]
```

exp：

```
salt=[94, 70, 97, 67, 14, 83, 73, 31, 81, 94, 54, 55, 41, 65, 99, 59, 100, 59, 21, 24, 91, 62, 34, 80, 70, 94, 5
3, 78, 67, 35, 96, 59]
enc=[142, 136, 163, 153, 196, 165, 195, 221, 25, 236, 108, 155, 243, 27, 139, 91, 62, 155, 241, 134, 243, 244, 1
64, 248, 248, 152, 171, 134, 137, 97, 34, 193]
for i in range(32):
 enc[i] ^= salt[i]
 enc[i] = enc[i]//2
print(bytes(enc))

#b'hgame{Ea$Y-Vm-t0-PrOTeCT_cOde!!}'
```