asong

asong
程序分析
    main
        copy
        j_sort
            sort
    set_cipher
        replace
        or
逆向
    re_or
    re_replace
    index
    sort
完整脚本：

程序分析

首先看到是压缩包里面是一个64-elf， 两个文件， 其中一个out还是乱码，

然后查看下程序。配合动调基本可以确定流程，

然后这里是标注好了的：

main

main函数是这样子， 一开始申请两块空间，一块在后面用于放数据，另一块保存flag：

```
data = malloc(0xBCuLL);
flag = malloc(0x50uLL);
j_setbuf();
reada(flag);
copy(flag);
j_sort("that_girl", data);
set_cipher(flag, data);
return 0LL;
```

其中的函数reada， 就是读取flag，

copy

然后copy中，是判断flag格式并且保留中间的部分， 复制到申请的flag内存块中，

```
dest = malloc(0x50uLL);
if ( memcmp(flag, "QCTF{", 5uLL) )
  exit(-1);
memcpy(dest, (flag + 5), 0x4BuLL);
v1 = strlen(dest);
if ( *(dest + v1 - 1) == '}' )
  *(dest + v1 - 1) = 0;
memcpy(flag, dest, 0x50uLL);
free(dest);
```

j_sort

然后j_sort函数， 读取那个'that_girl' 的文件，然后在函数sort中处理，最后根据处理结果得到data中的对应位置自增一个， 这个是类似与计数的一个效果：

```
v7 = __readfsqword(0x28u);
fd = open(txt, 0, a2, txt);
while ( read(fd, &buf, 1uLL) == 1 )
{
  v2 = sort(buf);
  ++*(4LL * v2 + data);
}
return close(fd);
```

sort

然后其中的sort函数， 一个逆向中也比较常见的转换的流程：

```
case '?':
  result = (a1 - 27);                   // 14
  break;
case '_':
  result = (a1 - 49);                   // 46
  break;
default:
  if ( a1 <= 47 || a1 > 48 )
  {
    if ( a1 <= 64 || a1 > 90 )          // 10--35
    {
      if ( a1 > 96 && a1 <= 122 )       // 10 -- 35
        result = (a1 - 87);
    }
    else
    {
      result = (a1 - 55);
    }
  }
  else
  {
    result = (a1 - '0');                // result = 0
  }
  break;
```

set_cipher

然后后面就是set_cipher函数， 是由我们处理好了的data和flag生成一个cipher：

```
v6 = __readfsqword(0x28u);
len = strlen(flag);
for ( i = 0; i < len; ++i )
  cipher[i] = *(4LL * sort(flag[i]) + data);
replace(cipher);
or(cipher, len);
j_write(cipher, "out", len);
return __readfsqword(0x28u) ^ v6;
```

流程主要是， 先吧flag和那个txt文档一样在sort中处理，然后取出来对应的数值，形成一个cipher，

其实这是吧这个flag中字符转化为， 这个字符在txt文件中出现的次数，是大致这个意思，

然后后面经过replace和or两个处理，最后写入到out文件中，所以我们的out文件中的数据应该是最后的cipher，

replace

然后是转换函数， 就是个互相换位置，这个过程中生成的呼唤的数据还是有点意思，

```
v2[4] = 0;
*v2 = *cipher;
while ( rep_data[*&v2[1]] )
{
  cipher[*&v2[1]] = cipher[rep_data[*&v2[1]]];
  *&v2[1] = rep_data[*&v2[1]];
}
result = v2[0];
cipher[*&v2[1]] = v2[0];
return result;
```

or

这个也是比较常见的一个位运算配合按位或的操作，

```
v3 = *cipher >> 5;
for ( i = 0; len - 1 > i; ++i )
  cipher[i] = 8 * cipher[i] | (cipher[i + 1] >> 5);
result = &cipher[i];
*result = 8 * *result | v3;
return result;
```

逆向

加密流程了解了， 我们逆向去解密，

最开始先从out文件读入最后的cipher，

```python
s = open('out',mode='r').read()
```

re_or

还是一些简单题目会常用的位操作， 这个位置用ipython多调几次还是比较好理解， 主要有前后两位之间的按位或操作：

```python
temp = arr[len(arr)-1] & 0x7
for i in range(len(arr)):
    brr.append((temp << 5) | (arr[i] >> 3))
    temp = arr[i] & 0x7
```

re_replace

首先我们ida-python导出这个rep_data：

```python
addr = 0x6020a0
arr = []
for i in range(38):
    arr.append(Dword(addr+4*i))
print(arr)
```

然后我们去得到中间实际上取到的值：

```python
arr = [....]
v = 0
while arr[v]:
    print(v,end=',')
    v = arr[v]
print(v)
```

就可以得到其中真实参与这个互换的数组：

```python
rep_data =
[0,22,20,19,14,17,4,30,29,28,27,36,34,33,32,31,37,35,26,25,
5,24,15,23,16,13,12,8,21,11,10,18,3,2,6,9,7,1]

```

然后我们加密时就是这个数组， 后一个覆盖前一个，最后的首项再去覆盖最后一项，形成一个互换，

逆向互换：

```python
tmp = brr[1]
for i in range(len(rep_data)-1, -1, -1):
    brr[rep_data[i]] = brr[rep_data[i-1]]
brr[0] = tmp
```

index

由于cipher是flag在data中寻址，我们动调，导出data的数据， 然后使用index方法：

```python
arr = []
flag = ''
# reverse for
for i in range(len(brr)):
    arr.append(data.index(brr[i]))
```

sort

最后得到sort函数处理以后的flag， 去审查sort函数， 但是发现其中大小写字母转换出来是一个值， 即这个sort函数不会分辨大小写，那么我们大小写都求一下，

其中的流程还是比较简单，也是很多简单题目常见的套路：

```python
s1 = ''
s2 = ''
for i in range(len(arr)):
    tmp = arr[i]
    if tmp == 46:
        s1 += '_'
        s2 += '_'
    if tmp == 48:
        s1 += '0'
        s2 += '0'
    if 10 <= tmp <= 36:
        s1 += chr(tmp + 55)
    if 10 <= tmp <= 36:
        s2 += chr(tmp + 87)
```

```
print(s1)
print(s2)
```

完整脚本：

```python
s = open('out',mode='r').read()
print(s)
rep_data =
[0,22,20,19,14,17,4,30,29,28,27,36,34,33,32,31,37,35,26,25
,5,24,15,23,16,13,12,8,21,11,10,18,3,2,6,9,7,1]

data = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 104, 30, 15, 29,
169, 19, 38, 67, 60, 0, 20, 39, 28, 118, 165, 26, 0, 61,
51, 133, 45, 7, 34, 0, 62, 0, 0, 0, 0, 0, 0, 40, 71, 0, 0,
66, 245, 0, 0, 0, 97, 0]


arr = []
brr = []

len_flag = len(s)

# string to array
for i in range(len_flag):
    arr.append(ord(s[i]))

# reverse or
temp = arr[len(arr)-1] & 0x7
for i in range(len(arr)):
    brr.append((temp << 5) | (arr[i] >> 3))
    temp = arr[i] & 0x7


# reverse replace
tmp = brr[1]
for i in range(len(rep_data)-1, -1, -1):
    brr[rep_data[i]] = brr[rep_data[i-1]]
brr[0] = tmp


arr = []
flag = ''
```

```
# reverse for
for i in range(len(brr)):
    arr.append(data.index(brr[i]))

s1 = ''
s2 = ''
for i in range(len(arr)):
    tmp = arr[i]
    if tmp == 46:
        s1 += '_'
        s2 += '_'
    if tmp == 48:
        s1 += '0'
        s2 += '0'
    if 10 <= tmp <= 36:
        s1 += chr(tmp + 55)
    if 10 <= tmp <= 36:
        s2 += chr(tmp + 87)
print(s1)
print(s2)

```