> Write by NEFUNSI：
>
> ghosin
>
> 0ERROR

签到

signin

下载 `signin.txt` 打开得到一串 `base64` ，解码得到 `flag{we1come_t0_NEFUCTF!}` 。

Re_SignUp

下载 `SignUp` ，使用 `010 Editor` 打开发现是 `Linux` 可执行文件。用 `ida64` 打开可看到 `main` 函数处有字符串 `xr yeI^r}.s{A)qAL/h{lM{AI.(rz?c` ，根据 `Hint` 中的 `xor` 使用在线工具爆破可得到 `fl>g{W@lc0me_7o_R1verSe_W06ld!}` 。

MISC

untitle

下载压缩包，发现加密的 `flag.txt` 和没加密的 `untitle.wav` 。解压 `untitle.wav` ，发现是倒放。使用 `Audacity` 逆转后听歌识曲得到 `Untitled` 。
以 `Untitled` 作为密码，使用 `WinRAR` 解压 `flag.txt` 发现报错和 fakeflag 。

> ! G:\labs\nefuctf\misc\111flag111 (2).rar: 无法设置 C:\Users\ghosi\AppData\Local\Temp\Rar$DRb13916.37568\flag.txt 的安全数据
> 这个安全 ID 不能分配为此对象的所有者。

怀疑是 `NTFS交换数据流隐写` ，使用 `NtfsStreamsEditor2` 扫描得到 `flag{M4ke_m1sc_gr3at_ag@in}` 。

蛇图攻击

下载得到一个 zip 压缩包，解压得到 `看到了吗flag就在那.zip` 和 `蛇图.jpg` ，在 `看到了吗flag就在那.zip` 中看到要找的 `flag.txt` 和已有明文的 `蛇图.jpg` ，使用 `ZIP已知明文攻击` 即可得到 `flag.txt` ，内容为 `flag{congratulation~~}` 。这里注意在压缩 `蛇图.jpg` 的时候要使用 **`Bandizip`** ，使用其他压缩软件可能会使 `AZPR` 类软件出现类似 `No matching files in selected archives` 的错误。

别人家的孩子

下载附件打开听一下，发现左右声道不同。拖进 `Audacity` ，去掉左声道的歌曲，右声道前半部分为莫斯电码，记录下内容为 `..-./.-../.-/---/-/-----/-..../...--` ，可解出内容为 `flagt0b3` ，后半部分为 `dtmf` ，使用 `dtmf2num` 读取到 `12111111711411510110810 2` ，可解出 `yourself` 。至此拼接得到 `flag{t0b3yourself}` 。

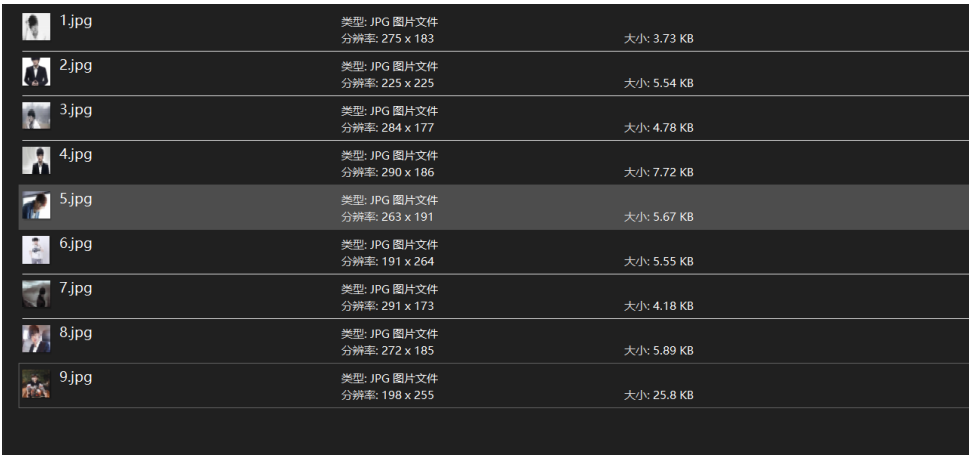
谁还没受过伤呢

根据题意，心愿小姐失恋了，然后附了一个QQ号：2508898249，因此就从QQ号下手找线索。

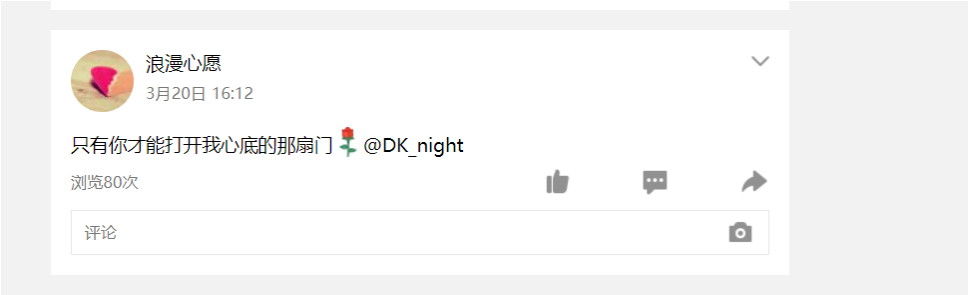
打开资料页可以看出信息量不大，主要就是有生日，其他信息不出意外应该都在空间里面

打开空间，首先看到几条说说，其中包含一条：

百度网盘打开是一个IDOL的文件夹，下载以后是9张许嵩的照片，这里注意了，第9张照片明显体积偏大，而照片又很糊，说明照片里面有东西



继续翻说说，又发现了一个线索说说



不知道有什么用，暂且留在这里不动。

看看其他地方，相册里面有一个有问题的照片，根据问题很明显就是前面心愿小姐所@的人就是问题的答案，填入DK_night后，得到一张图片，扫描图片上的二维码即可得到前一半flag

然后继续看，留言板里有一段话

```
G1v3_me_s0m3_rEd_P4cKe7eS_t0_g3t_7h3_h1nt!
```

不知道有什么用，暂且放着。

空间里能得到的线索就这么多了，再回头看看照片，第九张照片用010Editor打开后发现有隐写的zip格式的压缩包，分离出来以后如图

hint已经很明显了，是生日按照格式输入就是密码，但是填入前面的日期却无法解密，仔细观察后发现上面的生日是农历，将其日期转成公历以后即可得到2001.5.14(这里这个日期有点坑，转成2001年的四月廿二好像日期不太对，换思路转为今年的四月廿二才为正确日期，成功解密)，还有一种方法是知道了密码格式之后直接放进cracker里面按照格式爆破即可得到，速度也很快，这里就不作具体说明了。

打开flag.doc以后发现里面是空的，ctrl+a之后发现有字符，文字换颜色之后发现是个假flag，说明线索不在文档里，再看看文档的详细信息

可以看出备注和管理者里面各有一堆乱序字符，其中管理者里面的字符有明显的Base64加密标志，解密一次后发现变成了乱码，怀疑是加密后的成分残缺导致的错位，考虑到上面备注内的不明所以的乱码，将其和此段合并后解密(N次套娃)后即可得到后半flag

```
even_1f_i_L0Ve_U}
```

CRYPTO

RSA1.0

> 前置知识: https://xz.aliyun.com/t/6459

```python
from Crypto.Util.number import *
import gmpy2

# 由题目可知
c = 45771827077895268121546606393404724664015354394782828337635308358292455516045347632184096192996835878084629358486100781061410
e = 65537

n = 51644741119502351404435614782384203642721946675648874025147110583432277606213746585645021132709993574641883435114709470653840
# yafu 分解 n 可得 p 和 q
p = 71864275630874031237645674350715221527655354858438893865958105583537572498849323077814503088766875267239575220999107380899510
q = 71864275630874031237645674350715221527655354858438893865958105583537572498849323077814503088766875267239575220999107380899510

# 计算私钥 d
d = gmpy2.invert(e, (p-1)*(q-1))

# 解密 m
m = pow(c,d,n)

print(long_to_bytes(m))
# b'flag{now_y0u_know_rsa}'
```

---

are you file?

下载 `6.txt` ，发现是 `Ook!` 编码，使用在线工具解码得到

`呅食食性嘍哮食冬註魚嗥笨塺嚽覺我嘶人雜嗚有怎囑爾發常哮出沒喜唬有意吖人哗哈類更我呆魚蜂圖呆喜食氏告嘶嗒洞現訴訴盜萌現嘶嘿雜非嗒和嘆發笨`

，是熊曰编码，使用在线工具解码得到 `G2aZlXllv1_uvohfim3}deBb=f{yhenaotGmkbFXgfpfoadelf!` ，猜测栅栏密码，使用在线工具，每组字数 `3` 解码得到 `GmG23ma}kZdbleFXBXlbgl=fvfp1{f_youhavedonehalfofit!` 。根据结尾 `_youhavedonehalfofit!` 可知已经解开一半了。去掉这部分，猜测接下来是W型栅栏密码，因为栅栏密码加密后第一位不变，故逆转字符串为 `f{1pfvf=lgblXBXFelbdZk}am32GmG` ，使用在线工具，栏数为 `3` 可得到 `flag{bml1X3BpX2FfeGlvbmdfZGk=}` 。解码其中的 `base64` 可得到 `flag{niu_pi_a_xiong_di}` 。

Web

> 详见：f1oat's blog - 校赛web-wp

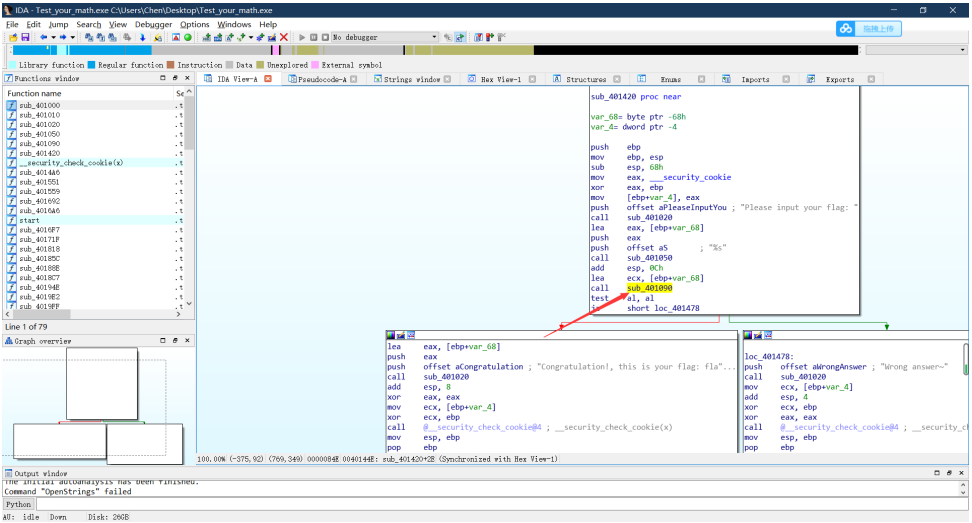
RE

Test_your_math

IDA打开后查看字符串，发现flag关键字

查找之后反向寻找其在程序中的位置

继续在主函数回溯其判断输入数据正确性的位置



得到一个判断输入数据正确性的函数，转为源代码形式

很明显可以看出是一个解方程，只需使用hint中提到的z3解出答案方程答案即可得到flag

maze

拿到题目打开后

可以看出这应该是关键代码的一部分。

这里jnz跳转到了下一行代码，相当于没跳转，

```
.text:0040102C                 jnz     short near ptr loc_40102E+1
.text:0040102E
.text:0040102E loc_40102E:                             ; CODE XREF: .text:0040102C↑j
.text:0040102E                 call    near ptr 0EC85D78Bh
.text:0040102E ; ---------------------------------------------------------------------------
.text:00401033                 db 0
.text:00401034                 dd 0EB000000h, 0EC4D8B09h, 8901C183h, 7D83EC4Dh, 6E7F0DECh
```

而下面的 `call near ptr 0EC85D78Bh` 调用了一个不是地址的地址，可以推断出这段代码添加了花指令，IDA分析失败了。

可以确定这个jnz指令是花指令，还有下面的call指令。

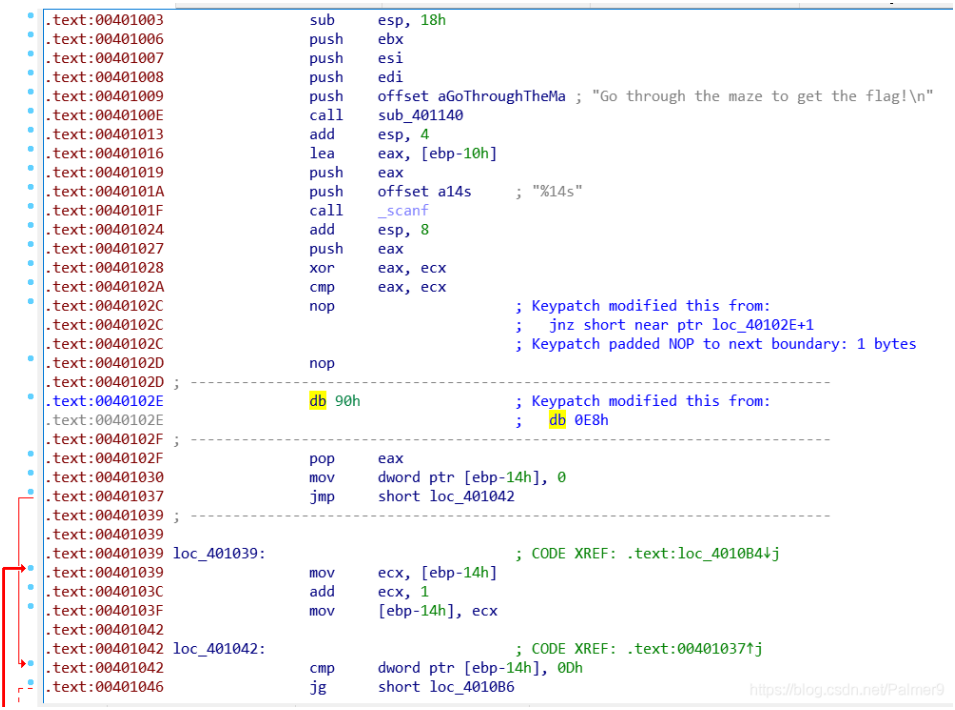
先将jnz指令nop掉。

```
.text:00401000 _main:                                  ; CODE XREF: start+AF↓p
.text:00401000                 push    ebp
.text:00401001                 mov     ebp, esp
.text:00401003                 sub     esp, 18h
.text:00401006                 push    ebx
.text:00401007                 push    esi
.text:00401008                 push    edi
.text:00401009                 push    offset aGoThroughTheMa ; "Go through the maze to get the flag!\n"
.text:0040100E                 call    sub_401140
.text:00401013                 add     esp, 4
.text:00401016                 lea     eax, [ebp-10h]
.text:00401019                 push    eax
.text:0040101A                 push    offset a14s    ; "%14s"
.text:0040101F                 call    _scanf
.text:00401024                 add     esp, 8
.text:00401027                 push    eax
.text:00401028                 xor     eax, ecx
.text:0040102A                 cmp     eax, ecx
.text:0040102C                 nop                    ; Keypatch modified this from:
.text:0040102C                                        ;   jnz short near ptr loc_40102E+1
.text:0040102C                                        ; Keypatch padded NOP to next boundary: 1 bytes
.text:0040102D                 nop
.text:0040102E
.text:0040102E loc_40102E:
.text:0040102E                 call    near ptr 0EC85D78Bh
.text:0040102E ; ---------------------------------------------------------------------------
.text:00401033                 db 0
.text:00401034                 dd 0EB000000h, 0EC4D8B09h, 8901C183h, 7D83EC4Dh, 6E7F0DECh
.text:00401034                 dd 0FEC558Bh, 0F01544BEh, 8BE84589h, 0E983E84Dh, 0E84D8961h
.text:00401034                 dd 16E87D83h, 458B5277h, 8AD233E8h, 40111290h, 9524FF00h
```

之后就是这个call指令，不能全部nop，因为后面那个东西可能是有效代码。摁d将其先转换为字节数据。

```
.text:00401024                 add     esp, 8
.text:00401027                 push    eax
.text:00401028                 xor     eax, ecx
.text:0040102A                 cmp     eax, ecx
.text:0040102C                 nop                    ; Keypatch modified this from:
.text:0040102C                                        ;   jnz short near ptr loc_40102E+1
.text:0040102C                                        ; Keypatch padded NOP to next boundary: 1 bytes
.text:0040102D                 nop
.text:0040102D ; ---------------------------------------------------------------------------
.text:0040102E                 db 0E8h
.text:0040102F                 db  58h ; X
.text:00401030                 db 0C7h
.text:00401031                 db  45h ; E
.text:00401032                 db 0ECh
.text:00401033                 db 0
.text:00401034                 dd 0EB000000h, 0EC4D8B09h, 8901C183h, 7D83EC4Dh, 6E7F0DECh
.text:00401034                 dd 0FEC558Bh, 0F01544BEh, 8BE84589h, 0E983E84Dh, 0E84D8961h
```

经过试验（先将第一个数据nop掉再转成代码，不行再将前俩数据nop……）发现 `db 0E8h` 这条指令是添加的花指令，将其nop掉，nop掉之后IDA自动的将后面的数据转换为代码数据



此时数据地址是红色的，将关键代码全部选中，摁p键将其声明为函数
然后就可以F5伪代码了

IDA View-A | Pseudocode-A | Hex View-1 | Str

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    signed int i; // [esp+10h] [ebp-14h]
4    char v5[16]; // [esp+14h] [ebp-10h]
5
6    sub_401140(aGoThroughTheMa);
7    scanf(a14s, v5);
8    for ( i = 0; i <= 13; ++i )
9    {
10     switch ( v5[i] )
11     {
12       case 'a':
13         --dword_408078;
14         break;
15       case 'd':
16         ++dword_408078;
17         break;
18       case 's':
19         --dword_40807C;
20         break;
21       case 'w':
22         ++dword_40807C;
23         break;
24       default:
25         continue;
26     }
27   }
28   if ( dword_408078 != 5 || dword_40807C != -4 )
29   {
30     sub_401140(aTryAgain);
31   }
32   else
33   {
34     sub_401140(aCongratulation);
35     sub_401140(aHereIsTheFlagF);
36   }
```


根据题目maze可知这是一个迷宫题。
控制的有两个数据，一个是`dword_408078`，另一个是`dword_40807C`
双击进去查看初始数据。
初始状态：

```
dword_408078=7
dword_40807C=0
```

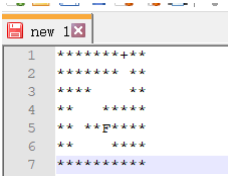
然后经过14次移动需要使

```
dword_408078=5
dword_40807C=-4
```

然后在内存中找到迷宫

```
.data:00408026                 db    0
.data:00408027                 db    0
.data:00408028 unk_408028      db    0                ; DATA XREF: _doexit:loc_4027F4↑o
.data:00408029                 db    0
.data:0040802A                 db    0
.data:0040802B                 db    0
.data:0040802C                 db    0
.data:0040802D                 db    0
.data:0040802E                 db    0
.data:0040802F                 db    0
.data:00408030                 db '********+********* ******    ****   ******* **┌******    *********'
.data:00408030                 db '*****',0
.data:00408077                 db    0
.data:00408078 dword_408078    dd 7                     ; DATA XREF: _main:loc_401096↑r
.data:00408078                                          ; _main+9E↑w ...
.data:0040807C dword_40807C    dd 0                     ; DATA XREF: _main:loc_401074↑r
.data:0040807C                                          ; _main+7D↑w ...
.data:00408080 aGoThroughTheMa db 'Go through the maze to get the flag!',0Ah,0
.data:00408080                                          ; DATA XREF: _main+9↑o
.data:004080A6                 align 4
.data:004080A8 ; char a14s[]
.data:004080A8 a14s            db '%14s',0              ; DATA XREF: _main+1A↑o
.data:004080AD                 align 10h
.data:004080B0 aCongratulation db 'Congratulations!',0Ah,0
.data:004080B0                                          ; DATA XREF: _main+C8↑o
.data:004080C2                 align 4
.data:004080C4 aHereIsTheFlagF db 'Here is the flag:flag{%s}',0Ah,0
.data:004080C4                                          ; DATA XREF: _main+D9↑o
.data:004080DF                 align 10h
```


提取出来排列之后



用wasd控制行走，dword_408078控制左右，dword_40807C控制上下。
s是上，w是下，a是右，d是左。
行走路径为ssaaasaassdddw
则flag为 `flag{ssaaasaassdddw}`