Write by NEFUNSI:

ghosin

0ERROR



#### 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\_70\_R1verSe\_W061d1} 。

# 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

下載得到一个 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 121111117114115101108102 ,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

1.јрд	类型: JPG 图片文件 分辨率: 275 x 183	大小: 3.73 KB
2.jpg	类型: JPG 图片文件 分辨率: 225 x 225	大小: <b>5.54 KB</b>
3.jpg	类型: JPG 图片文件 分辨率: 284 x 177	大小: 4.78 KB
4.jpg	类型: JPG 图片文件 分辨率: 290 x 186	大小: 7.72 KB
n 5.jpg	类型: JPG 图片文件 分辨率: 263 x 191	大小: 5.67 KB
6.jpg	类型: JPG 图片文件 分辨率: 191 x 264	大小: 5.55 KB
7.jpg	类型: JPG 图片文件 分辨率: 291 x 173	大小: <b>4.18 KB</b>
8.jpg	类型: JPG 图片文件 分辨率: 272 x 185	大小: 5.89 KB
9.jpg	类型: JPG 图片文件 分辨率: 198 x 255	大小: <b>25.8 KB</b>

继续翻说说,又发现了一个线索说说

<b>浪漫心愿</b> 3月20日 16:12			$\sim$
只有你才能打开我心底的那扇门 🖡 @DK_night 浏览80次	ılır	Ţ	*
评论			٥

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

```
from Crypto.Util.number import *
import gmpy2
# 由題目可知
c = 4577182707789526812154660639340472466401535439478282833763530835829245551604534763218409619299683587808462935848610078106141
e = 65537
n = 5164474111950235140443561478238420364272194667564887402514711058343227760621374658564502113270999357464188343511470947065384
# yafu 分解 n 可得 p 和 q
p = 7186427563087403123764567435071522152765535485843889386595810558353757249884932307781450308876687526723957522099910738089951
q = 7186427563087403123764567435071522152765535485843889386595810558353757249884932307781450308876687526723957522099910738089951
g = 7186427563087403123764567435071522152765535485843889386595810558353757249884932307781450308876687526723957522099910738089951
g = 7186427563087403123764567435071522152765535485843889386595810558353757249884932307781450308876687526723957522099910738089951
g = 7186427563087403123764567435071522152765535485843889386595810558353757249884932307781450308876687526723957522099910738089951
g = mpow(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

呋食食性喋唑食冬註魚嚀笨麼噤覺我嘶人離呜有怎嘱爾發常哮出沒喜號有意吖人哞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喂發笨 , 是熊曰编码,使用在线工具解码得到 G2a2lXllv1\_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{lpfvf=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

10A - Test_your_math.exe C:\User	s\Chen\Desktop\Tes	st_your_math.exe				- a x
Eile Edit Jump Search View I	Debugger Options	Windows Help				2518 H-46
👩 🖬 🔶 🕶 🔶 🐴 🆓 🍇 🚳	🌲 🕥 🛋 🖌	1 1 2 - 2 1 2 1	K 🕨 🔲 🖬 No debugger	* 🐮 🚼 🚺 🕈		CO SERVICE
						•
Library function 📕 Regular fu	nction 📕 Instruct	ion 🔳 Data 📕 Unex	plored 📕 External symbol			
Functions window	0 8 × 🔯	IDA View-A 🖾	🖪 Pseudocode-A 🖸 🛛 🖬 Strings	window 🖸 🛛 🖸 Hex View-1 🔹	🖪 Structures 🖸 🖽 Emms 🖾 📆	Imports 🖾 😿 Exports 🖾
Function name	Se ^				sub_401420 proc near	
$ \begin{array}{c} \\ \hline r & r & r & r & r & r & r & r & r & r$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1				var_68= byte ptr -68h var_6x Abord ptr -6 posh edp mov edp, sop mov	; "Please input your flag: "
< Line 1 of 79	>					
A County and an			🖬 🚅 🗵	1		
	100.	.00% (-375,92) (7 <del>8</del> 3	lea push call add xor mov xor call adv mov xor pop 000008#1 004014#E; eu_401	eax, [ebp+var_68] eax offset aCongratulation ; "( sub_40126 esp, 8 eax, eax ecx, [ebp+var_4] ecx, ebp ebp ebp ebp 200428 (Symburonized with Hex Viet	<pre>congratulation1, this is your flag: fla". ;security_check_cookie(x) r=1)</pre>	loc_401478: puth offset abrogdmaser; "birng answer-" wow ecc.[ebywar_4] add esp, 4 sor ecx, ebp sor esx, ecb sor esx, ecc addcocily_check_cookie#4;security_check_cooki
Output window						0 # ×
The inicial aucoanalysis has t	oeen Tinisneu.					^
Command "OpenStrings" failed						×
Python						
AU: idle Down Disk: 266B						

an 🔤 : 🕶 🗛 🛷

✤ 拖拽上传

# 得到一个判断输入数据正确性的函数,转为源代码形式 NGA-Tet your mathcare (-Ukert/Gen/Detkop/Tet your mathcare File Edit Jump Search Yiew Debugger Options Windows Help

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:00401020	inz	short near ntr loc 40102E+1
text:00/01025	52	Shore near per 100_10102011
toxt:0040102E loc 40102E		· CODE VEEL toxt.0040102CA
.text.0040102E 10C_40102E.	ee11	, CODE AREFLEXT.0040102CTJ
. (ext:0040102E	Call	near pur <mark>vecopu/obn</mark>
.text:0040102E ;		
.text:00401033	db 0	
.text:00401034	dd 0EB6	000000h, 0EC4D8B09h, 8901C183h, /D83EC4Dh, 6E/F0DECh
而下面的 call page ptp 05005070Ph	调田了一个不	
III F HELD CALL HEAT PCT OCCOSO780		
可以确定这个ing也会具龙也会 诉	ちて面的の別地	~
		₹•
无将jnz指令nop捍。		Carec, conse char arev, conse char chyp?
.text:00401000 main:		: CODE XREF: start+AF↓p
• .text:00401000	push	ebp
.text:00401001	mov	ebp. esp
.text:00401003	sub	esp. 18h
text:00401006	nush	ebx
• text:00401007	push	esi
• text:00401008	push	edi
• text:00/01009	nush	offset aGoThroughTheMa : "Go through the maze to get the flagl\n"
• text:00/0100F	call	sub A011A0
• text:00/01013	add	esp /
+ext:00401015	100	eav [ebn_10h]
+avt:00401010	nuch	
+ext:00401015	push	offset alls : "%14s"
. LEXT: 0040101A	push coll	sconf
toxt:0040101	add	_scall
toxt:00401024	auu	esp, o
. LEX1.00401027	push	eax
. LEX1.00401028	XOP	eax, ecx
. LEXT. 0040102A	Cilip	edx, etx
. LEXT. 0040102C	пор	, Keypatch moutried this from.
. LEXT: 0040102C		, JHZ SHOPE HEAR PER TOC_40102E+1
.text:00401020		; Reyparch padded NOP to next boundary: I bytes
.text:0040102D	пор	
.text:0040102E	-	
.text:0040102E 10c_40102	E:	I REALING
.text:0040102E	call	near ptr vectou/aun
.text:0040102E ;		
.text:00401033	db 0	
.text:00401034	dd 0E	B000000h, 0EC4D8B09h, 8901C183h, 7D83EC4Dh, 6E7F0DECh
.text:00401034	dd 0⊦	EC558Bh, 0F01544BEh, 8BE84589h, 0E983E84Dh, 0E84D8961h
.text:00401034	dd 16	E87083n, 45885277n, 8A0233E8h, 40111290h, 9524FF00hps://blog.csdn.net/Palmer
- FOUT HARMATAZA	66 AM	
之后就是这个call指令,不能全部nc	p,因为后面别	I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, x
.text:00401027	push	eax

.text:0040102/	push	eax		
.text:00401028	xor	eax,	ecx	
.text:0040102A	стр	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 0EB00	10000h	n, 0EC4D8B09h	n, 8901C183h, 7D83EC4Dh, 6E7F0DECh
.text:00401034	dd 0FEC5	58Bh,	, 0F01544BEh,	8BE84589h, 0E983E84Dh, 0E84D8961hsdn.net/Palmer9
LL.00401034	11 4/F07		4F00F077L	ONDODDERE NOTITIONE OF AFFORE

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 @E8h 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

•	.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 al4s ; "%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:0040102D ;		
•	.text:0040102E	<mark>db</mark> 90h	; Keypatch modified this from:
	.text:0040102E		; <mark>db</mark> 0E8h
	.text:0040102F ;		
•	.text:0040102F	рор	eax
•	.text:00401030	mov	dword ptr [ebp-14h], 0
-	.text:00401037	jmp	short loc_401042
	.text:00401039 ;		
	.text:00401039		
	.text:00401039 loc_401039:		; CODE XREF: .text:loc_4010B4↓j
→°	.text:00401039	mov	ecx, [ebp-14h]
•	.text:0040103C	add	ecx, 1
•	.text:0040103F	mov	[ebp-14h], ecx
	.text:00401042		
	.text:00401042 loc_401042:		; CODE XREF: .text:00401037↑j
	.text:00401042	cmp	dword ptr [ebp-14h], 0Dh
۲ <b>-</b>	.text:00401046	jg	short loc_4010B6 https://blog.csdn.net/Palmer9

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
   1int cdecl main(int argc, const char **argv, const char **envp)
  2 {
     signed int i; // [esp+10h] [ebp-14h]
     char v5[16]; // [esp+14h] [ebp-10h]
  4
  6
    sub_401140(aGoThroughTheMa);
     scanf(a14s, v5);
-
  8
     for (i = 0; i \le 13; ++i)
  9
     {
 10
       switch ( v5[i] )
       {
         case 'a':
13
           --dword 408078;
          break;
• 14
 15
        case 'd':
• 16
           ++dword 408078;
          break;
• 17
         case 's':
 18
 19
           --dword 40807C:
 20
          break;
 21
         case 'w':
 22
           ++dword 40807C;
23
           break;
 24
         default:
 25
           continue;
 26
       }
  27
     3
28
     if ( dword_408078 != 5 || dword_40807C != -4 )
  29
     {
 30
       sub 401140(aTryAgain);
     }
     else
     {
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:lo	c_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	'****** <sub>+</sub> *****	*** ***	****	****	******	**F*****	*******
.data:00408030		db	'****',0						
.data:00408077		db	0						
.data:00408078	dword_408078	dd	7		DATA	XREF:	_main:loc_	4010961r	
.data:00408078					mai	n+9E1w			
.data:0040807C	dword_40807C	dd	0		, DATA	XREF:	_main:loc_	4010741r	
.data:0040807C					_mai	n+7D1w			
.data:00408080	aGoThroughTheMa	db	'Go through the	maze †	to get	the f	lag!',0Ah,0		
.data:00408080					, DATA	XREF:	_main+9↑o		
.data:004080A6		ali	gn 4						
.data:004080A8	; char a14s[]								
.data:004080A8	a14s	db	'%14s',0		, DATA	XREF:	_main+1A↑o		
.data:004080AD		ali	gn 10h						
.data:004080B0	aCongratulation	db	'Congratulation	s! <b>',</b> 0Al	1,0				
.data:004080B0					, DATA	XREF:	_main+C8↑o		
.data:004080C2		ali	gn 4						
.data:004080C4	aHereIsTheFlagF	db	'Here is the fl	ag:flag	g{%s}'	,0Ah,0			
.data:004080C4					, DATA	XREF:	_main+D9↑o		
.data:004080DF		ali	gn 10h						

提取出来排列之后

#### 님 new 1🗵 \*\*\*\*\* 2 \*\*\*\*\*\* \*\* \*\*\*\* \*\* \*\* \*\*\*\* 4 5 \*\* \*\*F\*\*\*\* \*\* \*\*\*\* 6 7 \*\*\*\*\*\*\*

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}