

第一届北京大学信息安全综合能力竞赛解题报告

梁圣通

2021 年 11 月 21 日

1 → 签到 ←

打开 pdf 文件，用福昕 PDF 编辑器¹将页面调大以显示全部文字，选中并复制奇怪文字²，粘贴得到

fa{aeAGetTm@ekaev!lgHv__ra_ieGeGm_1}

注意大括号的位置和最左侧的 fa 容易发现字符串的变换方式，将前半半字符与后半半字符插空重排可以得到 flag

flag{Have_A_Great_Time@GeekGame_v1!}

2 小北问答 Remake

#1 答案：5。常识题³。

#2 答案：407。 <https://news.pku.edu.cn/xwzh/203d197d93c245a1aec23626bb43d464.htm>。

#3 答案：2021-07-11T08:49:53+08:00。 <https://crt.sh/?id=4362000550>⁴。

#4 答案：000{this_is_the_welcome_flag}。 <https://archive.ooo/c/welcome-to-dc2020-quals/358/>。

#5 答案：2933523260166137923998409309647057493882806525577536。利用容斥原理可以使用 python 计算得到答案⁵。

```
1 x=672328094
2 y=386900246
3 allnum=x*y*(x-1)*(y-1)*(x-2)*(y-2)//6
4 threenum=2*((x-y+1)*y*(y-1)*(y-2)+y*(y-1)*(y-2)*(y-3)//2)//6
5 twonum=2*((x-y+1)*y*(y-1)+2*(-2+y)*(-1+y)*y//3)//2*(x-2)*(y-2)
6 twotwonum=4*((x-y-2)*(x-y)*(y-1)*(y-1)//8+(-2+x)*(x-2*y)*(-4+x+4*y)*(x-2*(1+y))//96\
7 +y*(16-20*y-4*y*y+5*y*y*y)//96+(x-2*y)*(-2+2*x*x+6*y-4*y*y-x*(3+2*y))*(y-1)//24\
8 +((y-2)*(y-2)*y//16-(2*x-3*y)*(4*x*x-6*x*(y+2)-9*y*y+18*y+8)//48)*(y-1))
```

¹ <https://editor.foxitsoftware.cn/>。因为我的 Adobe Acrobat 过期了，这也间接导致我三个小时才做出这道题。

² 实际上是 Wingdings Font，然而做出这题并不需要知道这一点。

³ 就算不知道的话枚举也能做出来，当然百度地图也是一种办法。

⁴ 正则表达式中的 3 是一个提示。一开始用 <https://ui.ctsearch.entrust.com/ui/ctsearchui> 搜索，只得到了精确到日期的答案。之后，由于对 ISO 8601 的不熟悉，提交了若干次错误的答案。

⁵ 在计算过程中，有一些复杂的求和式子利用 WolframAlpha 进行化简。附件中的 threequeen.py 中带有一些注释，可以略微体现思路。附件中的 threequeen.cpp 是根据网上的代码修改得到的求解三皇后问题的代码，可以对较小的棋盘得到参考解，以调试 python 脚本。由于一开始使用 / 进行除法，提交了若干次错误的答案。

```

9 twotwonumelse=4*((y//2-1)*y//2//2*x//2+2*(-(-2+y)*y*(2-3*x+y)//48)+(-2+y)*y*y//16\
10 -(x-y)*(-4+x*x+6*y-5*x*y-2*y*y)//48)
11 final=allnum-twonum+twotwonum-twotwonumelse+2*threenum
12 print(final)

```

- x, y 是方形棋盘的边长⁶。上述公式成立的充分条件为

$$x, y \in \mathbb{N}_+, \quad 2 \mid x, \quad 2 \mid y, \quad 3/2 < x/y < 2.$$

- **allnum** 是三个皇后两两不同行列的方法数。

$$\text{allnum} = xy(x-1)(y-1)(x-2)(y-2)/6.$$

- **threenum** 是三个皇后在同一条斜线上的方法数。

$$\text{threenum} = 2 \left((x-y+1)y(y-1)(y-2)/6 + 2 \sum_{i=1}^{y-1} i(i-1)(i-2)/6 \right).$$

- **twonum** 是至少两个皇后在同一条斜线上，且第三个皇后与这两个皇后不同行列的方法数。

$$\text{twonum} = 2 \left((x-y+1)y(y-1)/2 + 2 \sum_{i=1}^{y-1} i(i-1)/2 \right) (x-2)(y-2).$$

- **twotwonum** 是三个皇后在两条斜线上，其中一个皇后被另外两个皇后从不同方向斜向攻击的方法数。

$$\text{twotwonum} = 4 \sum_{i=1}^{x/2} \sum_{j=1}^{y/2} (\min(x-i, y-j) + \min(i-1, j-1)) (\min(x-i, j-1) + \min(i-1, y-j)).$$

- **twotwonumelse** 是三个皇后在两条斜线上，其中一个皇后被另外两个皇后从不同方向斜向攻击，且另外两个皇后在同一行或同一列的方法数。

$$\begin{aligned} \text{twotwonumelse} = 4 \sum_{i=1}^{x/2} \sum_{j=1}^{y/2} & (\min(x-i, y-j, i-1) + \min(x-i, y-j, j-1) \\ & + \min(x-i, i-1, j-1) + \min(y-j, i-1, j-1)). \end{aligned}$$

- **final** 是三个皇后互不攻击的方法数。

$$\text{final} = \text{allnum} - \text{twonum} + \text{twotwonum} - \text{twotwonumelse} + 2\text{threenum}.$$

#6 submits. <https://github.com/PKU-GeekGame/geekgame-0th/blob/main/src/choice/game/db/db.sqlite>⁷。

#7 AS59201. <http://www.bgplookingglass.com/list-of-autonomous-system-numbers-2>⁸。

#8 区域光纤通信网与新型光通信系统国家重点实验室. <https://eecs.pku.edu.cn/info/1060/11528.htm>。

回答正确所有问题后得到两个 flag

```

flag{jiu-Cong-XIan-Zai-Kai-Shi}
flag{Shu1~y3~Zu~Lan~bu~zhuuuu~}

```

⁶在大小为嘉然*神楽七奈 Official 的方形棋盘上放 3 枚（相同的）皇后且它们互不攻击，有几种方法？


⁷一开始提交了文件名 db。

⁸一开始在 <http://www.bgplookingglass.com/list-of-autonomous-system-numbers> 上查找，得到并提交了两个错误的答案。

3 叶子的新歌

3.1 夢は時空を越えて

解压题目文件，在 Windows 系统下直接显示了唱片集

Secret in Album Cover!!				
名称	#	标题	参与创作的艺术家	唱片集
 LeafsNewSong.m...		叶子的新歌	叶子	Secret in Album Cover!!

根据提示，使用 `binwalk` 命令将封面 `png` 图片导出。用 `Stegsolve`⁹ 提取 RGB 三个通道的最低位隐含的信息¹⁰，得到第二个 `png` 图片。这是一个 `Aztex Code`，扫码¹¹得到

Gur frperg va uvfgbtznz.



猜测这是凯撒移位密码，平移 13 位得到

The secret in histogram.

根据提示，使用 `matlab` 作出第二个 `png` 图片的直方图，发现图像对应的向量几乎是二值的¹²。

```
1 a = imread('newmain.png');
2 [counts,binLocations] = imhist(a);
3 imhist(a)
4 w = (counts~=0)*1;
5 bs = '';
6 for i=1:256
7     bs = [bs,num2str(w(i))];
8 end
9 bs
```

⁹<http://www.caesum.com/handbook/Stegsolve.jar>。

¹⁰实际上，我遍历了 <https://ctf-wiki.org/misc/picture/png/> 中的所有方法。

¹¹<https://products.aspose.app/barcode/recognize/aztec>。

¹²最后一个分量与之前的分量均不相同，其余分量是二值的。

将直方图对应的向量二值化得到二进制数

```
1110100011000001111100010001001111100111101001000001101110001011
0011111100110001100011110111000100100000110011111101100110000010
0011101001111000110001001100001111001000011000011001111100010010
0010000100111000010001011100000111001100011100111000011110010111
```

观察到¹³其中连续的 0 和连续的 1 出现的频率较高且长度较大, 尝试将其转换为条形码¹⁴。扫码¹⁵得到

xmcp.ltd/KCwBa



打开链接可以得到由 Ook.、Ook? 和 Ook! 组成的序列。通过 Google 搜索得知这是一种名为 Ook! 的编程语言¹⁶。执行网站中的 Ook! 脚本¹⁷得到 flag

flag{y0u_h4ve_f0rgott3n_7oo_much}

3.2 幻夢界

使用 Audacity¹⁸ 查看 mp3 文件的元数据¹⁹, 发现 TRACKTOTAL 一项为

aHR0cDovL2xhYi5tYXh4c29mdC5uZXQvY3RmL2x1Z2FjeS50Ynoy

通过分析²⁰得知这是 base64 编码, 解码²¹得到

http://lab.maxxsoft.net/ctf/legacy.tbz2

打开链接下载 tbz2 文件, 使用 7-zip 打开该文件, 找到路径为 legacy.tbz2/legacy.tar/To_the_past.img 的 img 文件。该 img 文件可以继续使用 7-zip 打开, 打开路径为 To_the_past.img/NOTE.TXT 的文件得到提示

冥驭令詮怀馭榕喆艺艺冥庚艺怀喆諒令喆諒怀

直接 Google 搜索这段文字可以得到汉字与数字的对应关系, 进而得到

72364209117514983984

¹³实际上, 在此之前我搜索并尝试了许多将二进制数转换为文本的方式, 均告失败。

¹⁴<https://www.dcode.fr/barcode-39>。

¹⁵<https://online-barcode-reader.inliteresearch.com/>。

¹⁶<https://esolangs.org/wiki/Ook!>。

¹⁷<https://www.dcode.fr/ook-language>。

¹⁸<https://sourceforge.net/projects/audacity/>。

¹⁹实际上, 在此之前我搜索并尝试了许多分析音频的方法, 比如频谱分析等, 均告失败。Audacity 就是在尝试时下载的软件之一。

²⁰<https://www.boxentriq.com/code-breaking/cipher-identifier>。

²¹<https://www.boxentriq.com/code-breaking/base64-decoder>。

将其作为密码打开 To_the_past.img/MEMORY.ZIP 得到 MEMORY.ZIP/readme.txt, 部分文本为

你最喜欢在 4399 玩找不同, 而且你还玩的特别棒, 简直就是找不同滴神。

使用 010 Editor²² 打开 MEMORY.ZIP/left.bin 和 MEMORY.ZIP/right.bin, 发现两个文件内容基本相同。结合提示, 使用 python 将两个文件的不同部分按顺序输出为二进制文件 diffnewnew.bin。

```
1 import os
2 i = 0
3 a = [0,0,0,0]
4 b = [0,0,0,0]
5 with open('left.bin','rb') as left:
6     with open('right.bin','rb') as right:
7         with open('diffnewnew.bin','wb') as diff:
8             size = os.path.getsize('left.bin')
9             while i < size:
10                 i = i + 1
11                 ldata = left.read(1)
12                 rdata = right.read(1)
13                 if ldata == rdata:
14                     continue
15                 else:
16                     i = i + 3
17                     a[0] = ldata;
18                     a[1] = left.read(1)
19                     a[2] = left.read(1)
20                     a[3] = left.read(1)
21                     b[0] = rdata;
22                     b[1] = right.read(1);
23                     b[2] = right.read(1);
24                     b[3] = right.read(1);
25                     if a[0:3] == b[1:4]:
26                         diff.write(rdata)
27                         right.read(1)
28                     elif a[1:4] == b[0:3]:
29                         diff.write(ldata)
30                         i = i + 1
31                         left.read(1)
32                     else:
33                         raise Exception('Wrong data!' + str(hex(i)) + \
34                                     '\n' + ldata.hex() + \
35                                     '\n' + rdata.hex())
```

注意到该二进制文件的文件头为 4E4553, 搜索²³得知这是 NES 文件。使用 FCEUX²⁴ 打开该文件, 发现其为修改过的 Super Mario Bros 游戏。使用工具²⁵打开该文件, 得到通关文本

LAB.MAXXSOFT.NET CTF LEAFS
VISIT THIS TO

²²<https://www.sweetscape.com/010editor/>。

²³<https://blog.mythsman.com/post/5d301940976abc05b345469f/>。

²⁴<https://fceux.com/web/home.html>。

²⁵<http://www.romhacking.net/utilities/690/>, 这个工具我搜索了好久, 最后通过 <https://blog.beford.org/2011/09/21/hack-lu-2011-ctf-scottys-last-signal-solution/> 找到了该工具。本来想着播放 tas 录像通关以得到信息, 但未成功。

REVEAL THE TRUTH.

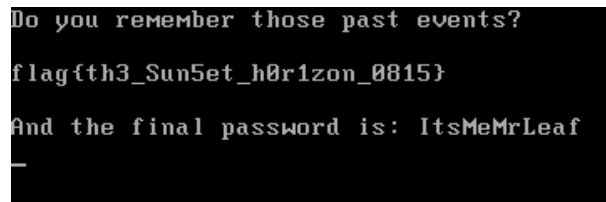
根据提示打开链接

lab.maxxsoft.net/ctf/leafs



被告知需要输入“软盘启动后的密码”²⁶。根据提示，回到之前的 img 文件，使用 VMware Workstation Pro²⁷ 创建 MS-DOS 虚拟机，将 img 文件作为软盘启动，得到 flag

flag{th3_Sun5et_h0r1zon_0815}



3.3 夢と現の境界

将软盘启动后得到的密码

ItsMeMrLeaf

输入到链接 lab.maxxsoft.net/ctf/leafs 的页面可以得到 flag²⁸

flag{W4ke_up_fr0m_7h3_L0NG_dre@m}

4 密码学实践

分析 python 脚本可以得到 MESenc 函数加密的方式。

²⁶一开始我输入了 72364209117514983984 和“冥驭令詮怀馭榕詰艺艺冥庚艺怀詰瞭令詰瞭怀”，均告失败。

²⁷<https://www.vmware.com/>。

²⁸进而我拿下了此题一血（校内），作为第一次接触 CTF 的选手还是比较有成就感的。

```

1 def MESenc(mess:bytes,skey:bytes):
2     assert len(skey)==8*32
3     keys = [bytes_to_long(skey[i*8:(i+1)*8]) for i in range(32)]
4     assert len(mess)%32 == 0
5     cip=b""
6     for it in range(0,len(mess),32):
7         pmess=mess[it:it+32]
8         a = bytes_to_long(pmess[0:8])
9         b = bytes_to_long(pmess[8:16])
10        c = bytes_to_long(pmess[16:24])
11        d = bytes_to_long(pmess[24:32])
12        for key in keys:
13            a, b, c, d = b, c, d, a ^ c ^ key
14        a=long_to_bytes(a,8)
15        b=long_to_bytes(b,8)
16        c=long_to_bytes(c,8)
17        d=long_to_bytes(d,8)
18        cip+=a+b+c+d
19    return cip

```

设 a_1 、 b_1 、 c_1 和 d_1 是执行 12 行的 for 循环之前的 a 、 b 、 c 和 d ，设 a_2 、 b_2 、 c_2 和 d_2 是执行 12 行的 for 循环之后的 a 、 b 、 c 和 d ，则有

$$\begin{aligned}
 a_2 &= c_1^{\text{keys}[2]} \text{keys}[8]^{\dots} \text{keys}[26]^{\text{keys}[4]} \text{keys}[10]^{\dots} \text{keys}[28], \\
 b_2 &= d_1^{\text{keys}[3]} \text{keys}[9]^{\dots} \text{keys}[27]^{\text{keys}[5]} \text{keys}[11]^{\dots} \text{keys}[29], \\
 c_2 &= a_1^{\text{keys}[0]} c_1^{\text{keys}[6]} \text{keys}[30]^{\text{keys}[4]} \text{keys}[10]^{\dots} \text{keys}[28], \\
 d_2 &= b_1^{\text{keys}[1]} d_1^{\text{keys}[7]} \text{keys}[31]^{\text{keys}[5]} \text{keys}[11]^{\dots} \text{keys}[29].
 \end{aligned}$$

定义

$$\begin{aligned}
 \text{key0} &= \text{keys}[2]^{\text{keys}[8]^{\dots} \text{keys}[26]^{\text{keys}[4]} \text{keys}[10]^{\dots} \text{keys}[28]}, \\
 \text{key1} &= \text{keys}[3]^{\text{keys}[9]^{\dots} \text{keys}[27]^{\text{keys}[5]} \text{keys}[11]^{\dots} \text{keys}[29]}, \\
 \text{key2} &= \text{keys}[0]^{\text{keys}[6]^{\dots} \text{keys}[30]^{\text{keys}[4]} \text{keys}[10]^{\dots} \text{keys}[28]}, \\
 \text{key3} &= \text{keys}[1]^{\text{keys}[7]^{\dots} \text{keys}[31]^{\text{keys}[5]} \text{keys}[11]^{\dots} \text{keys}[29]}.
 \end{aligned}$$

可以使用 python 通过已有的原文和密文²⁹得到它们的值。

```

1 from Crypto.Util.number import bytes_to_long, long_to_bytes
2 strings2 = b"\x62\x1b\xf3\x87\xcf\xbf\x7c\x62\x0c\x34\xb9\x93\x34\x7f\x8c\x50\x69\x9a\x3e\xd1\
3 \x43\x42\xb6\xb7\x51\xb6\x08\xfa\x36\x0f\x11\x56\x66\x06\xbf\x82\x9c\xe2\x35\x72\x05\x75\xa7\
4 \x87\x27\x3a\xc9\x54\x4a\xc2\x78\xd4\x1e\x54\xac\xc8\x14\xca\x1d\xee\x24\x0c\x74\x00\x4a\x13\
5 \xf1\xb4\xa0\x8f\x53\x69\x61\x1c\xc8\xf4\x49\x57\xf0\x2f\x7d\x92\x1c\xd1\x2d\x3b\xed\x8a\x60\
6 \x85\x71\xa7\x04\x62\x17\x6d"
7 strings1 = b"\x61\x1d\xbf\x9d\x8a\xaa\x7c\x72\x10\x34\xb9\x93\x34\x2d\xd8\x4e\x71\x96\x6c\xd5\
8 \x10\x57\xb6\xaf\x01\xb9\x04\xed\x70\x18\x78\x48\x72\x1b\xe9\x8e\xcf\xb5\x70\x34\x10\x7b\xb5\
9 \x8e\x61\x3c\x9d\x55\x55\x9a\x26\xc0\x45\x10\xe3\x8e\x01\xc0\x0b\xe7\x23\x1f\x2c\x53\x02\x65\
10 \x88\xfc\xf8\xcf\x02\x03\x7e\x03\xd7\xeb\x56\x48\xef\x30\x35\xe8\x4f\xaf\x78\x7f\x9c\xf2\x61\
11 \xff\x12\xf0\x62\x0f\x2c\x01"

```

²⁹ 通过与 Richard “交谈”可以得到 python 脚本中的 `string2` 和 `string1`，这是密文。查询 `server.py` 可以得到 python 脚本中的 `msg0` 和 `msg`，这是原文。

```

12 a1_2 = bytes_to_long(strings1[0:8])
13 b1_2 = bytes_to_long(strings1[8:16])
14 c1_2 = bytes_to_long(strings1[16:24])
15 d1_2 = bytes_to_long(strings1[24:32])
16 msg0 = "Hello, Alice! I will give you two flags. The first is:"
17 msg = "Sorry, I forget to verify your identity. Please give me your certificate."
18 msgu1 = msg[0:32].encode('utf-8')
19 a1_1 = bytes_to_long(msgu1[0:8])
20 b1_1 = bytes_to_long(msgu1[8:16])
21 c1_1 = bytes_to_long(msgu1[16:24])
22 d1_1 = bytes_to_long(msgu1[24:32])
23 key0 = c1_1 ^ a1_2
24 key1 = d1_1 ^ b1_2
25 key2 = a1_1 ^ c1_2 ^ c1_1
26 key3 = b1_1 ^ d1_2 ^ d1_1
27 for i in range(3):
28     strings0 = strings2[i*32:32+i*32]
29     a2 = bytes_to_long(strings0[0:8])
30     b2 = bytes_to_long(strings0[8:16])
31     c2 = bytes_to_long(strings0[16:24])
32     d2 = bytes_to_long(strings0[24:32])
33     c1 = a2 ^ key0
34     d1 = b2 ^ key1
35     a1 = c2 ^ c1 ^ key2
36     b1 = d2 ^ d1 ^ key3
37     msgu = long_to_bytes(a1,8) + long_to_bytes(b1,8) + long_to_bytes(c1,8) + long_to_bytes(d1,8)
38     print(msgu)

```

解码剩余的密文得到 flag

flag{fe1steL_neTw0rk_ne3d_an_OWf}