CTF-i春秋网鼎杯第二场misc部分writeup

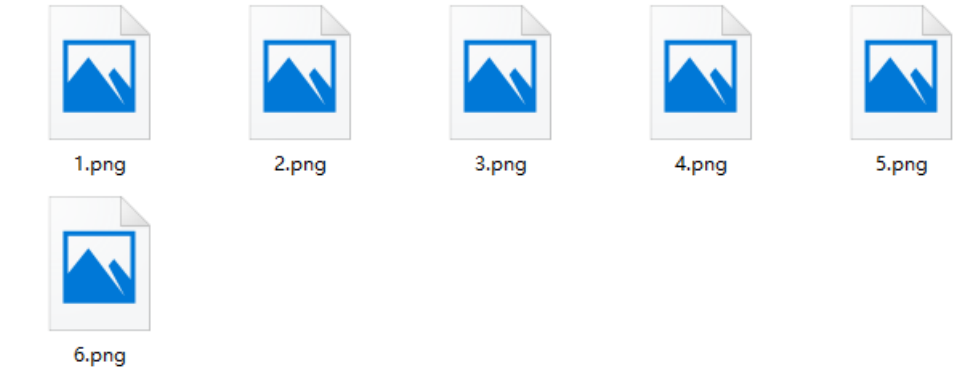
转载

powerx_yc 于 2018-08-23 15:23:00 发布 217 收藏
文章标签： 数据结构与算法
原文链接：http://www.cnblogs.com/pureqh/p/9523988.html

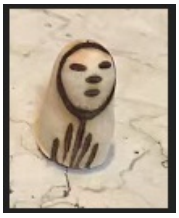
CTF-i春秋网鼎杯第二场misc部分writeup

套娃

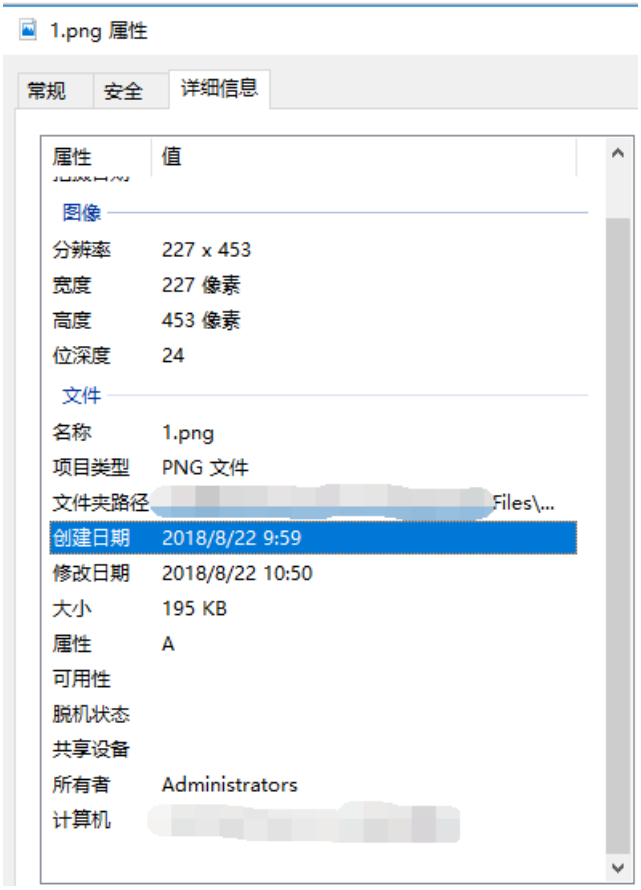
下载下来是六张图片



直接看并没有什么信息

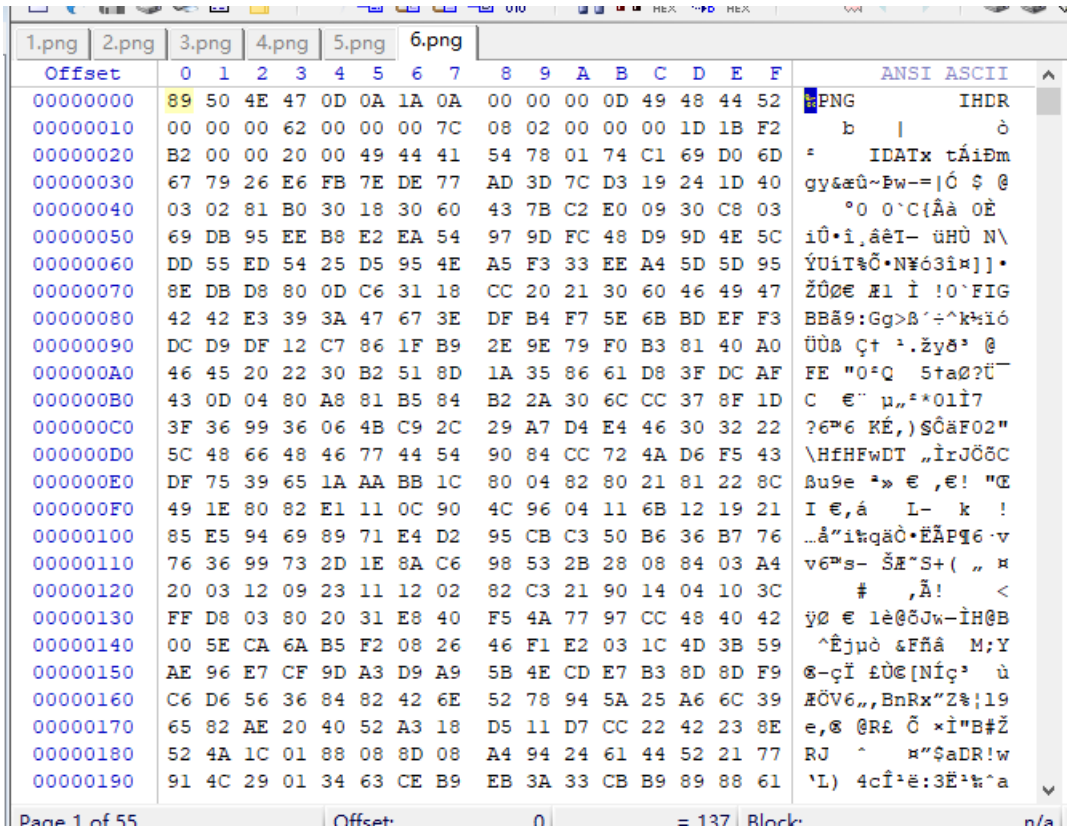


一个一个查看属性



没有找到有用信息

到winhexv里看一下



都是标准的png图片，而且没有flag写入Hex数据

继续扔到kali里用binwalk分析一下

```
root@kali:~/Desktop# binwalk 4.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 145 x 263, 8-bit/color RGB, non-interlaced
62            0x3E            Zlib compressed data, default compression

root@kali:~/Desktop# binwalk 5.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 174 x 246, 8-bit/color RGB, non-interlaced
62            0x3E            Zlib compressed data, default compression

root@kali:~/Desktop# binwalk 6.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 98 x 124, 8-bit/color RGB, non-interlaced
```

图片未嵌入其他文件

直接在kali双击打开png图片，发现可以正常打开(注:被修改过高度的图片无法在kali中直接打开，会显示无法载入图像)

那继续分析IDAT块，IDAT是png图片中储存图像像数数据的块，不清楚的可以去补充一下关于png图片格式知识

我们使用pngcheck分析图片的IDAT块

```
C:\pngcheck-2.3.0-win32>pngcheck.exe -v 1.png
File: 1.png (200329 bytes)
  chunk IHDR at offset 0x0000c, length 13
    227 x 453 image, 24-bit RGB, non-interlaced
  chunk pHYs at offset 0x00025, length 9: 2835x2835 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0003a, length 32768
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x08046, length 32768
  chunk IDAT at offset 0x10052, length 32768
  chunk IDAT at offset 0x1805e, length 32768
  chunk IDAT at offset 0x2006a, length 32768
  chunk IDAT at offset 0x28076, length 32768
  chunk IDAT at offset 0x30082, length 3571
  chunk IEND at offset 0x30e81, length 0
No errors detected in 1.png (10 chunks, 35.1% compression).

C:\pngcheck-2.3.0-win32>pngcheck.exe -v 2.png
File: 2.png (102345 bytes)
  chunk IHDR at offset 0x0000c, length 13
    171 x 315 image, 24-bit RGB, non-interlaced
  chunk pHYs at offset 0x00025, length 9: 2835x2835 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0003a, length 32768
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x08046, length 32768
  chunk IDAT at offset 0x10052, length 32768
  chunk IDAT at offset 0x1805e, length 3927
  chunk IEND at offset 0x18fc1, length 0
No errors detected in 2.png (7 chunks, 36.7% compression).

C:\pngcheck-2.3.0-win32>pngcheck.exe -v 3.png
File: 3.png (140019 bytes)
  chunk IHDR at offset 0x0000c, length 13
    205 x 367 image, 24-bit RGB, non-interlaced
  chunk pHYs at offset 0x00025, length 9: 2835x2835 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0003a, length 32768
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x08046, length 32768
  chunk IDAT at offset 0x10052, length 32768
  chunk IDAT at offset 0x1805e, length 32768
  chunk IDAT at offset 0x2006a, length 8821
  chunk IEND at offset 0x222eb, length 0
No errors detected in 3.png (8 chunks, 38.0% compression).
```

```
C:\pngcheck-2.3.0-win32>pngcheck.exe -v 4.png
File: 4.png (64219 bytes)
  chunk IHDR at offset 0x0000c, length 13
    145 x 263 image, 24-bit RGB, non-interlaced
  chunk pHYs at offset 0x00025, length 9: 2835x2835 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0003a, length 32768
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x08046, length 31361
  chunk IEND at offset 0x0fad3, length 0
No errors detected in 4.png (5 chunks, 43.9% compression).

C:\pngcheck-2.3.0-win32>pngcheck.exe -v 5.png
File: 5.png (66041 bytes)
  chunk IHDR at offset 0x0000c, length 13
    174 x 246 image, 24-bit RGB, non-interlaced
  chunk pHYs at offset 0x00025, length 9: 2835x2835 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0003a, length 32768
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x08046, length 32768
  chunk IDAT at offset 0x10052, length 403
  chunk IEND at offset 0x101f1, length 0
No errors detected in 5.png (6 chunks, 48.6% compression).
```

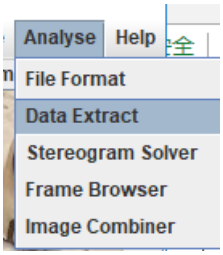
```
C:\pngcheck-2.3.0-win32>pngcheck.exe -v 6.png
File: 6.png (22564 bytes)
  chunk IHDR at offset 0x0000c, length 13
    98 x 124 image, 24-bit RGB, non-interlaced
  chunk IDAT at offset 0x00025, length 8192
    zlib: deflated, 32K window, superfast compression
  chunk IDAT at offset 0x02031, length 8192
  chunk IDAT at offset 0x0403d, length 6099
  chunk IEND at offset 0x0581c, length 0
No errors detected in 6.png (5 chunks, 38.1% compression).
```

1-5图片正常，但是6图片的块没有到32768就满了，猜测6图片可能有问题，划重点

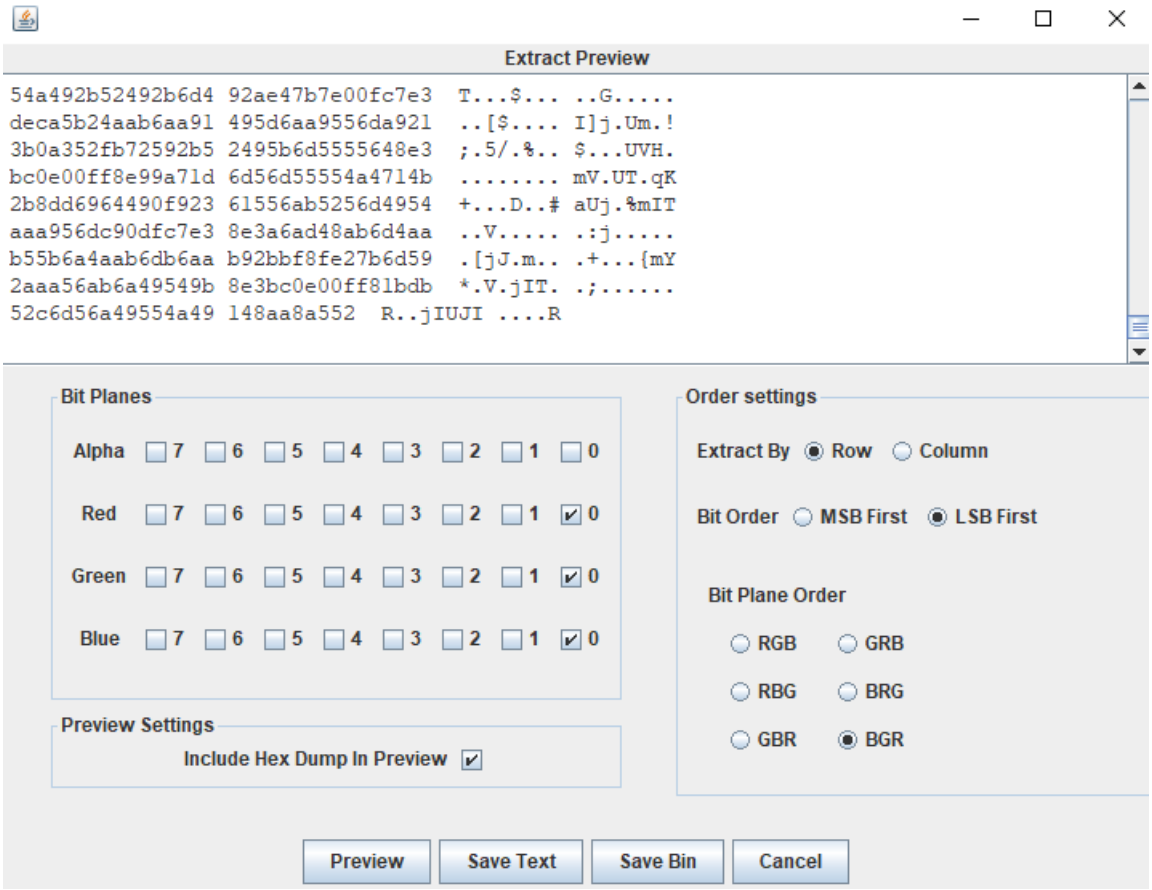
上Stegsolve重点分析6.png

先向后翻几页，看看是不是直接可以得到信息

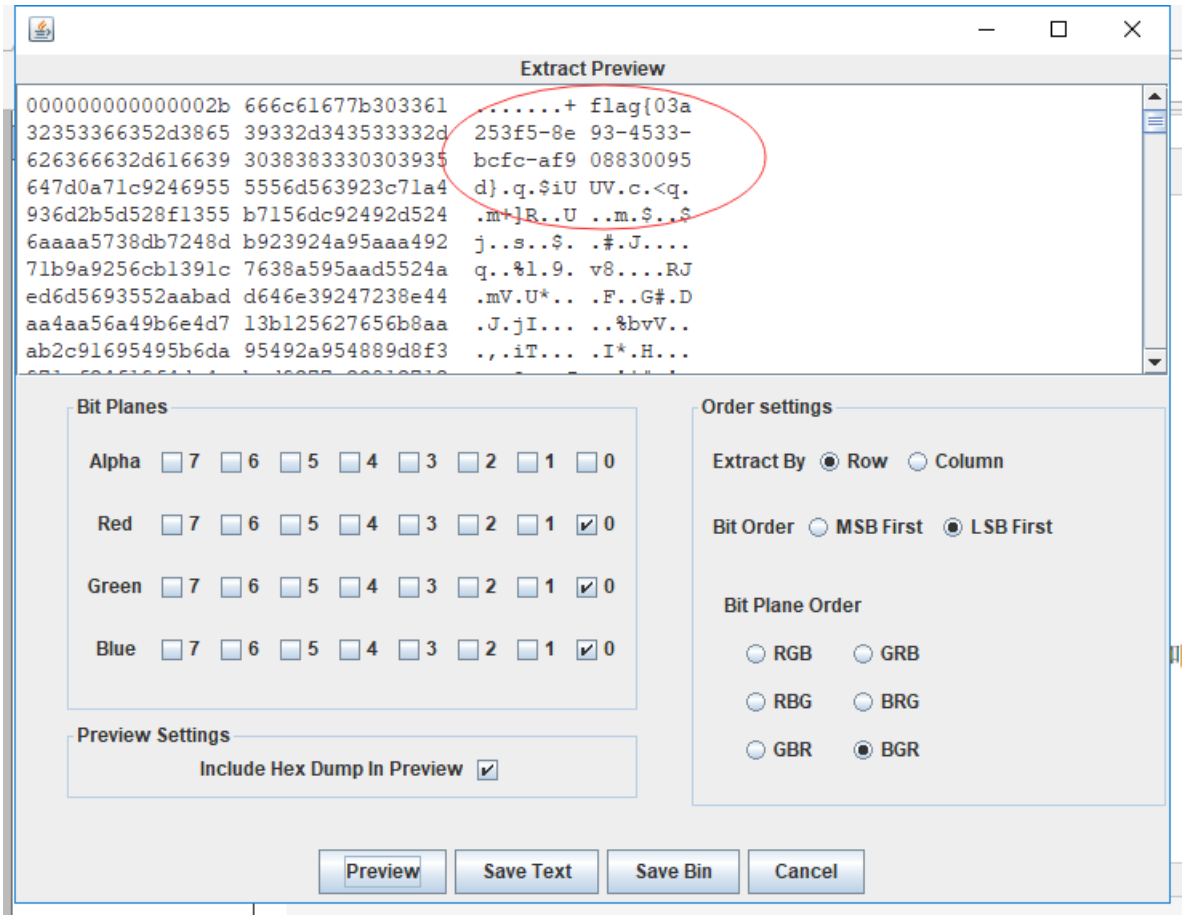
找到头也没找见什么东西，也不存在上一场的0通道lsb隐写，好吧，继续查看通道信息



勾选Red、Green、Bile的0通道，Order settings栏勾选LSB First BGR，点击Preview如下所示



把滑动栏移到顶端，好吧，flag在playload中...(ps:当初确实做到这一步了，但是因为默认是跳到最后的，没往上找，痛苦...所以还是要细心)

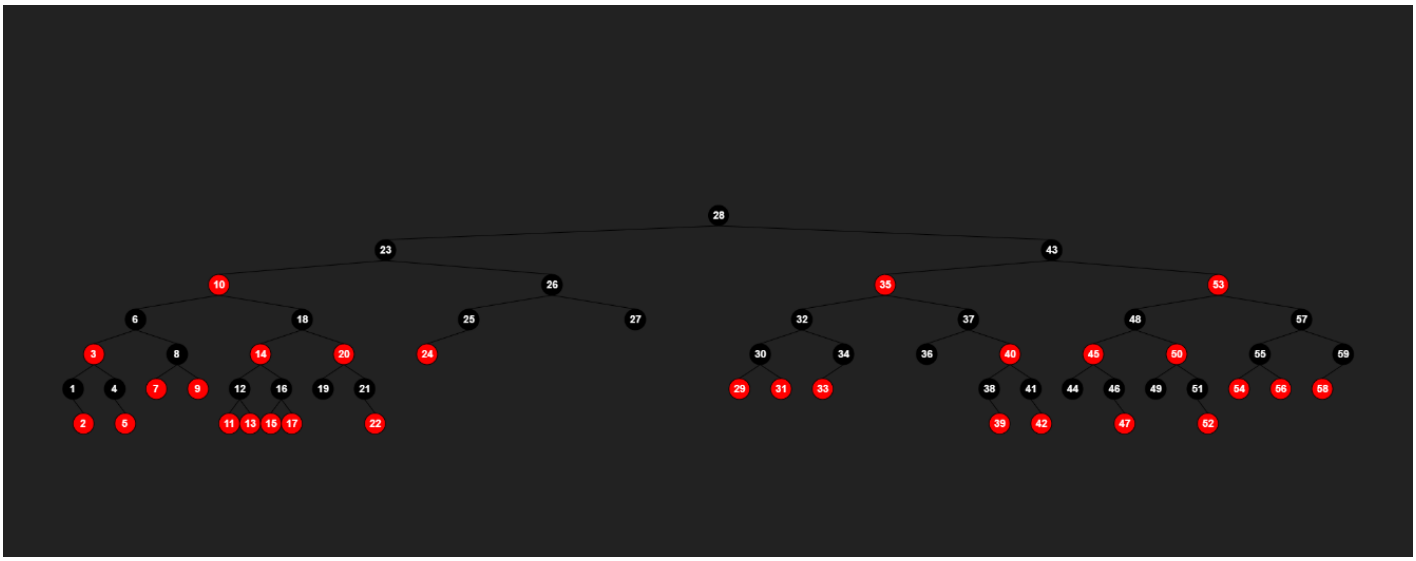


神奇的二叉树

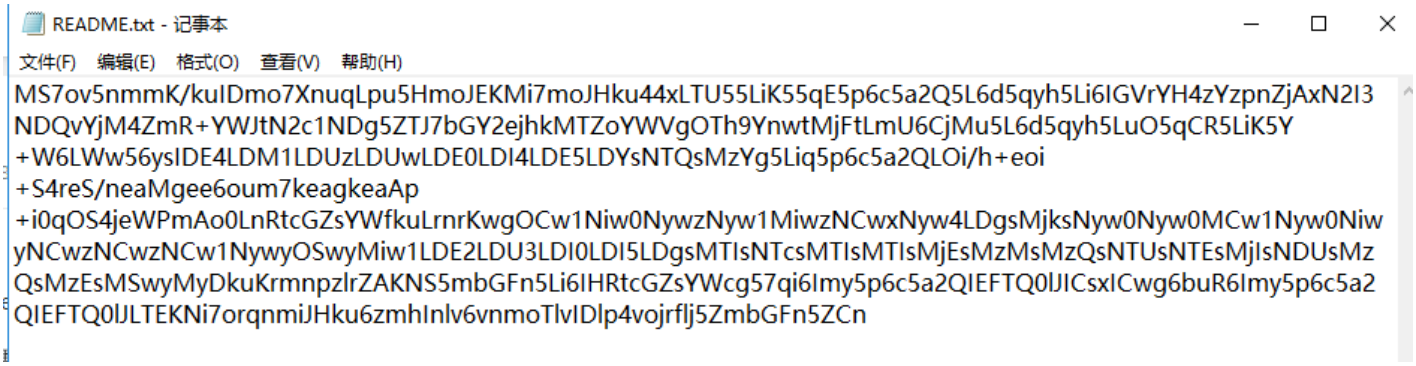
下载到题目后有两个文件，一个是文本文档一个是图片，既然给了文本文档那就看看吧


1.png


README.txt



README.txt内容如下




```
MS7ov5nmmK/kuIDmo7XnuqLpu5HmoJEKMi7moJHku44xLTU55LiK55qE5p6c5a2Q5L6d5qyh5Li6IGVrYH4zYzpnZjAxN2I3NDQvYjM4ZmR+
YWJtN2c1NDg5ZTJ7bGY2ejhkMTZoYWVgOTh9YnwtMjFtLmU6CjMu5L6d5qyh5LuO5qCR5LiK5Y+W6LWw56ysIDE4LDM1LDUzLDUwLDE0LDI4
LDE5LDYsNTQsMzYg5Liq5p6c5a2QLOi/h+eoi+S4reS/neaMgee6oum7keagkeaAp+i0qOS4jeWPmAo0LnRtcGZsYWfkuLrnrKwgOCw1Niw0
NywzNyw1MiwzNCwxNyw4LDgsMjksNyw0Nyw0MCw1Nyw0NiwyNCwzNCwzNCw1NywyOSwyMiw1LDE2LDU3LDI0LDI5LDgsMTIsNTcsMTIsMTIs
MjEsMzMsMzQsNTUsNTEsMjIsNDUsMzQsMzEsMSwyMyDkuKrmnpzlrZAKNS5mbGFn5Li6IHRtcGZsYWcg57qi6Imy5p6c5a2QIEFTQ0lJICsx
ICwg6buR6Imy5p6c5a2QIEFTQ0lJLTEKNi7orqnmiJHku6zmhInlv6vnmoTlvIDlp4vojrflj5ZmbGFn5ZCn
```


看样子应该是base64加密，拿去解密一下

明文:

1.这是一棵红黑树
2.树从1-59上的果子依次为
ek`~3c:gf017b744/b38fd~abm7g5489e2{lf6z8d16hae`98}b|-21m.e:
3.依次从树上取走第 18,35,53,50,14,28,19,6,54,36 个果子,过程中保持红黑树性质不变
4.tmpflag为第
8,56,47,37,52,34,17,8,8,29,7,47,40,57,46,24,34,34,57,29,22,5,16,57,24,29,8,12,57,12,12,21,33,34,55,51,22,45,34,31,1,23 个果子
5.flag为 tmpflag 红色果子 ASCII +1 , 黑色果子 ASCII-1
6.让我们愉快的开始获取flag吧

好的，规则出来了，按照规则有共59个红色和黑色的球，1号到59号各代表一个字母或符号，需要拿掉题目中提到的10个球后依然保持红黑树的性质，然后在保证这个性质的同时比对剩下的球和字符，按照红色ASCll+1和黑色ASCll-1的规则确认最终的flag。

好吧，那么什么是红黑树呢，红黑树（Red Black Tree） 是一种自平衡二叉查找树，是在计算机科学中用到的一种数据结构，典型的用途是实现关联数组。

它的性质如下：

- **性质1. 节点是红色或黑色。**
- **性质2. 根节点是黑色。**
- **性质3 每个叶节点（NIL节点，空节点）是黑色的。**
- **性质4 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)**
- **性质5. 从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点。**

有了性质就好说了，但是并不推荐手动画图，因为红黑树非是唯一解，所以最好利用网站：https://sandbox.runjs.cn/show/2nngvn8w在线生成红黑树。

按层添加节点，生成树后，按顺序删除节点，发现不止一个解，排序头5个字母应为flag{，最后一个应为}，得出8,56,47,37,52,23应为黑，黑，红，红，黑，黑
选择一个符合条件的红黑树，输出结果。

得到flag为

flag{10ff49a7_db11_4e43_b4f6_66ef12ceb19d}