2048*16

先打开开发者工具，发现有无限debugger存在，于是先输入 `for(let i = 1; i < 99999; i++) {window.clearInterval(i);}` 来取消掉所有的定时器。之后再观察堆栈，发现n函数是无限递归的，于是令`n = ()=>{}`无效掉这个函数。之后在界面发生改变的动画位置设置断点，按一个上下键发生暂停。观察堆栈，发现一个二维数组里面存了每一个位置的方块对象。于是改变两个方块的值为2048 * 16, 再令两个方块融合，即得到2048*16

ezASM

用编辑工具打开文档，发现flag的逻辑是对某一块数据的每一个位置对0x22取异或。于是对这块数据取异或。得到flag

ezPYC

先用pyinstxtractor反编译回pyc, 之后再用pycdc将pyc反编译为可读的python字节码。发现获取flag的逻辑是循环对[1, 2, 3, 4]取异或，从而得到flag

ezUPX

先用UPX去壳，再反汇编，发现flag是某一块数据对0x32取异或，从而得到flag

ezIDA

反汇编后直接在数据区发现flag

奇怪的图片

先看python源文件，发现图片是加密原理是一个字符一个字符写入flag并用另一张图片取异或。于是利用a^b^c^b = a^(b^b)^c = a^c 的原理，利用a和c的背景图片相同，只有差异的地方会出现图案，先用一张图片对其他所有图片取异或，发现无序的字符。再找出字符最多的两张图片，这两张分别是第一张和最后一张。经过测试可以发现第一张。再用第一张对每一张图片取异或，按照字符个数排序后，用前一张异或后一张，得到每一个位置的字符，从而得到flag

SignIn

将图片从下面看，即可得到flag