Misc Lab_2

必做部分

1 Challenge 1 : songmingti

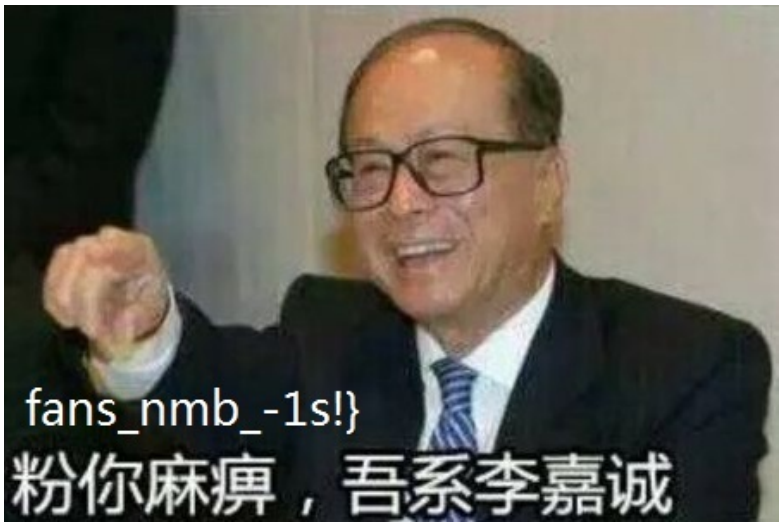
发现第一部分的 `flag` 已经给出了，继续使用图片隐写尝试步骤一把梭：
用 `binwalk` 命令分析这个文件：

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lab2$ binwalk 1.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
26657         0x6821          JPEG image data, JFIF standard 1.01
```

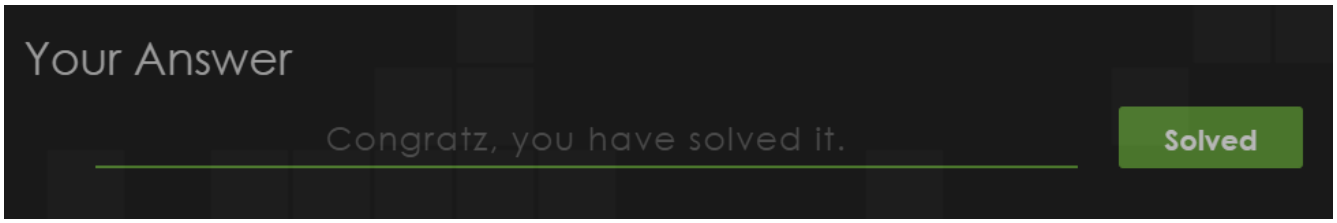
发现在第 `26657` 的位置，还藏了一张图片，我使用 `dd` 命令提取该位置的图片：

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lab2$ dd
if=1.jpg of=12.jpg bs=1 skip=26657
39003+0 records in
39003+0 records out
39003 bytes (39 kB, 38 KiB) copied, 10.0551 s, 3.9 kB/s
```

得到了第二部分的 `flag`



拼接一下，得到最后的`flag:AAA{the_true_fans_fans_nmb_-1s!}`



2 Challenge 2：miaomiaomiao

打开`http://cdn-zjusec-com-s.webvpn.zju.edu.cn:8001/Nov2/miaomiaomiao_2290CB13158C1F7B821EF107B56999C9.html`,发现一直在弹出`miao~`,但这题是图片隐写……所以用第一节课用过的`view-source`找一下图片：

```
<html><script src=/sf-webproxy/api/vpn-config></script><script src=/sf-webproxy/resource/web_p
<body>
    <script>
        for (;;) {
            alert('Miao~');
        }
    </script>
    <img src="img/miao~870F6C667A6CDC0D1F533859E72C48E0.jpg" />
</body>
</html>
```

发现在网页实际上藏了一张`jpg`，下载图片：

单看图片没什么思路，但在查看这张图片的详细信息的时候发现了作者栏非常奇怪：

单纯把key包裹在 `AAA{}` 里交上去，显示 `flag` 有错，这时联想到 `key` 酷似某个密钥，可能用了已有的工具进行隐写：

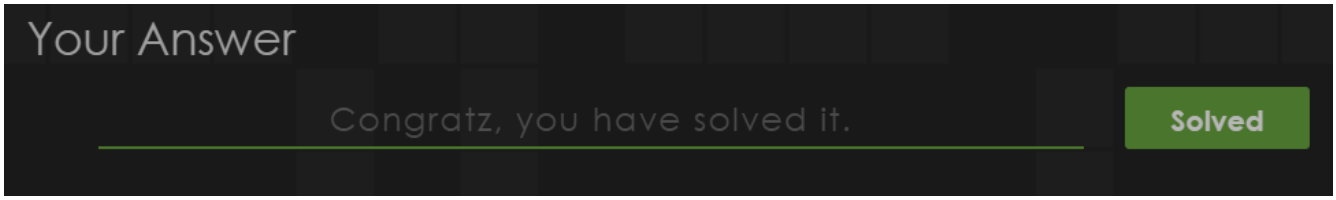
于是尝试 `steghide`：

```
PS D:\MyRepository\slowist-notebook\docs\Coding\CTF\misc-lab2> steghide extract -sf
miao.jpg
Enter passphrase:
wrote extracted data to "secret_file.txt".
```

打开 `<secret_file.txt>`,发现是一串二进制数:

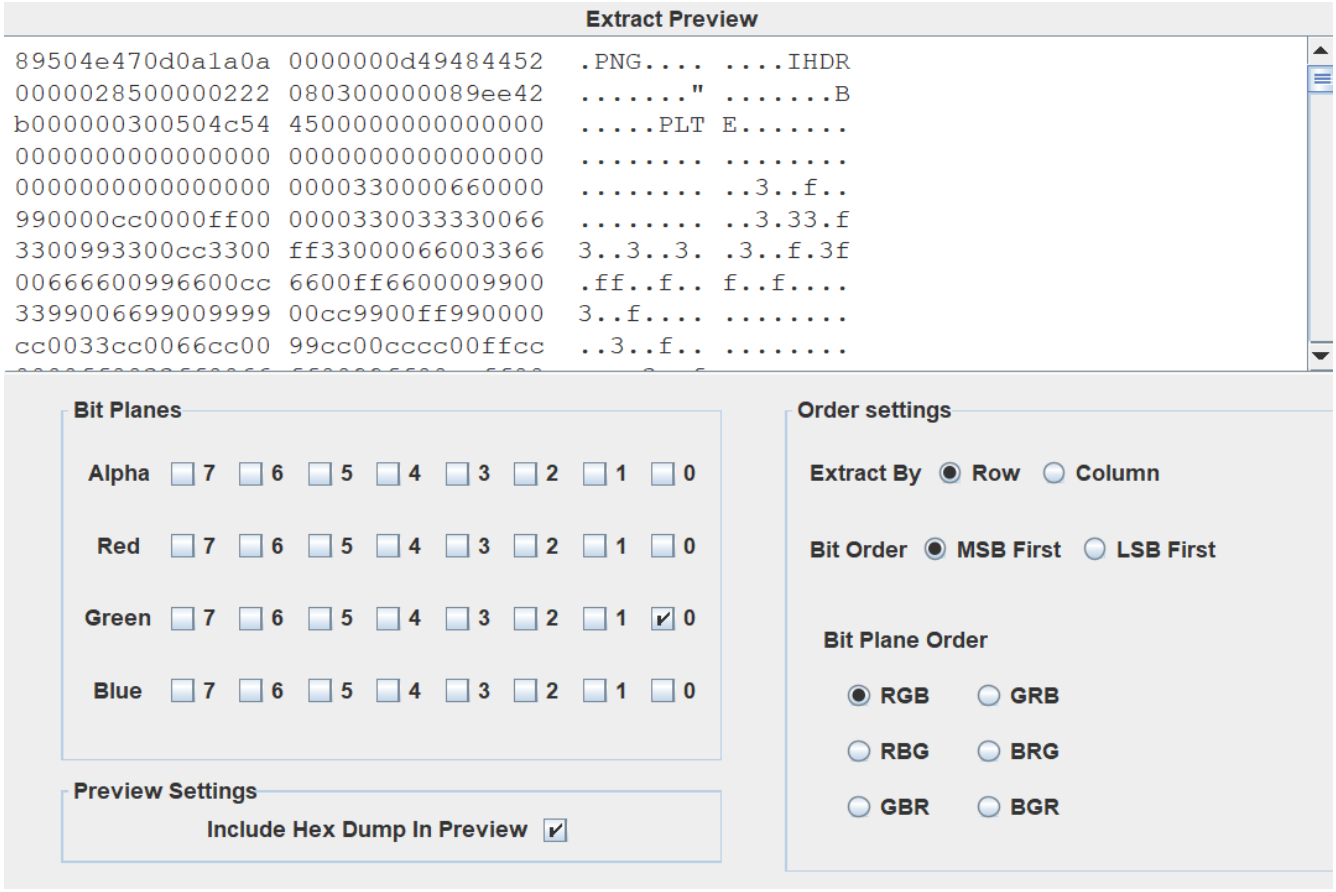
```
01000001010000010100000101111011010001000011000001011111010110010011000001110101010111110100110000110001011010110110010101011111010100110111010001100101001110010100100000110001011001000110010101011111010011010011000101100001001100000111110
```

利用 `cyberchef` 转换一下，就得到了 `flag`：`AAA{D0_Y0u_L1ke_Ste9H1de_M1a0}`



3 Challenge 3：Easy LSB

根据提示，由于和上次的lab0的lsb隐写方式不太一样……由于上次的是图像里的隐写，所以这次猜测是数据的，使用工具 `stegsolve` 打开图片，选择 `Data Extract`，然后尝试:



发现了明显的 `PNG` 文件头和 `IHDR` 数据块，因此猜测里面隐写了一张 `png`，使用 `Save Bin` 保存，用 `png` 格式打开:

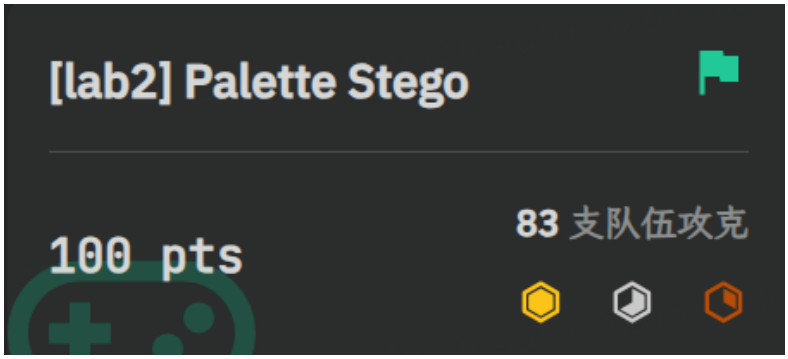
因此猜测这张图被截小了x 于是用修改文件头的还原图片的大小，增加宽度，并且重新计算 `crc32` 校验值：

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  已解码的文本
00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  .PNG........IHDR
00000010  00 00 02 85 00 00 02 72 08 03 00 00 00 85 60 4A  .......r......`J
00000020  47 00 00 03 00 50 4C 54 45 00 00 00 00 00 00 00  G....PLTE.......
```

```
In [6]: import binascii

In [7]: h='49484452 00000285 00000272 08030000 00'

In [8]: b=bytes.fromhex(h)

In [9]: binascii.crc32(b)
Out[9]: 2237680199

In [10]: hex(_)
Out[10]: '0x85604a47'
```

最后得到了完整的图片，也就拿到了flag：

`flag: AAA{w4Tch_G1RL5_8AND_CRY_its_k4mi_an1m3}`



选做部分

4 Challenge A: Palette Stego

主要查到了《EzStego的嵌入、提取与检测的C++实现_苏亚娟》：

> 嵌入消息的提取过程，与上述嵌入过程正好相反。简单 的讲，分为下面几步：①从图像数据流中解码后的得到像素的 在 初始调色板的 索引 Index0；② 根据 在初 始调 色板的索引 Index0 得到像素值 P；③在新的调色板中找到像素 P 对应的索 引 Index1；④如果索引Index1 为奇数，则嵌入位为 1，否则嵌入 位是 0，每 8 位组装成一个字节，就提取出消息。

- 首先，第一步，利用`im.getpalette()`函数，得到了调色板

```
    for i in range(256): # palette only has 8-bit color, so 2^8=256
        r=palette[3*i]
        g=palette[3*i+1]
        b=palette[3*i+2]
        Y = 0.299 * r + 0.587 * g + 0.114 * b
        plst.append(Y) # original palette
```

此时`plst`储存了现在图片的调色板。

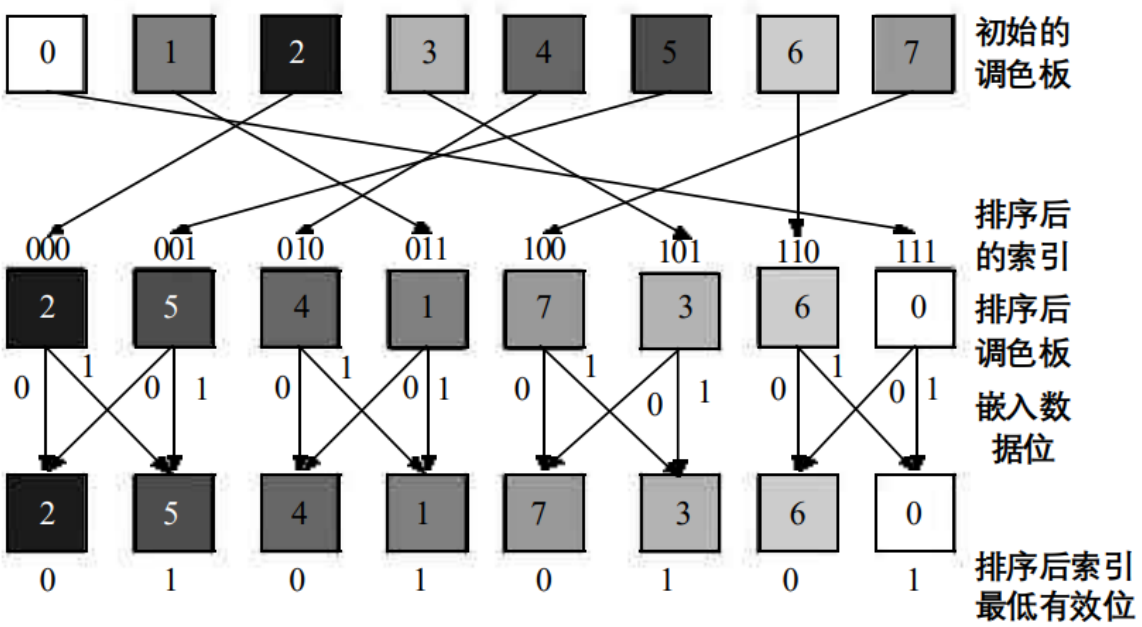


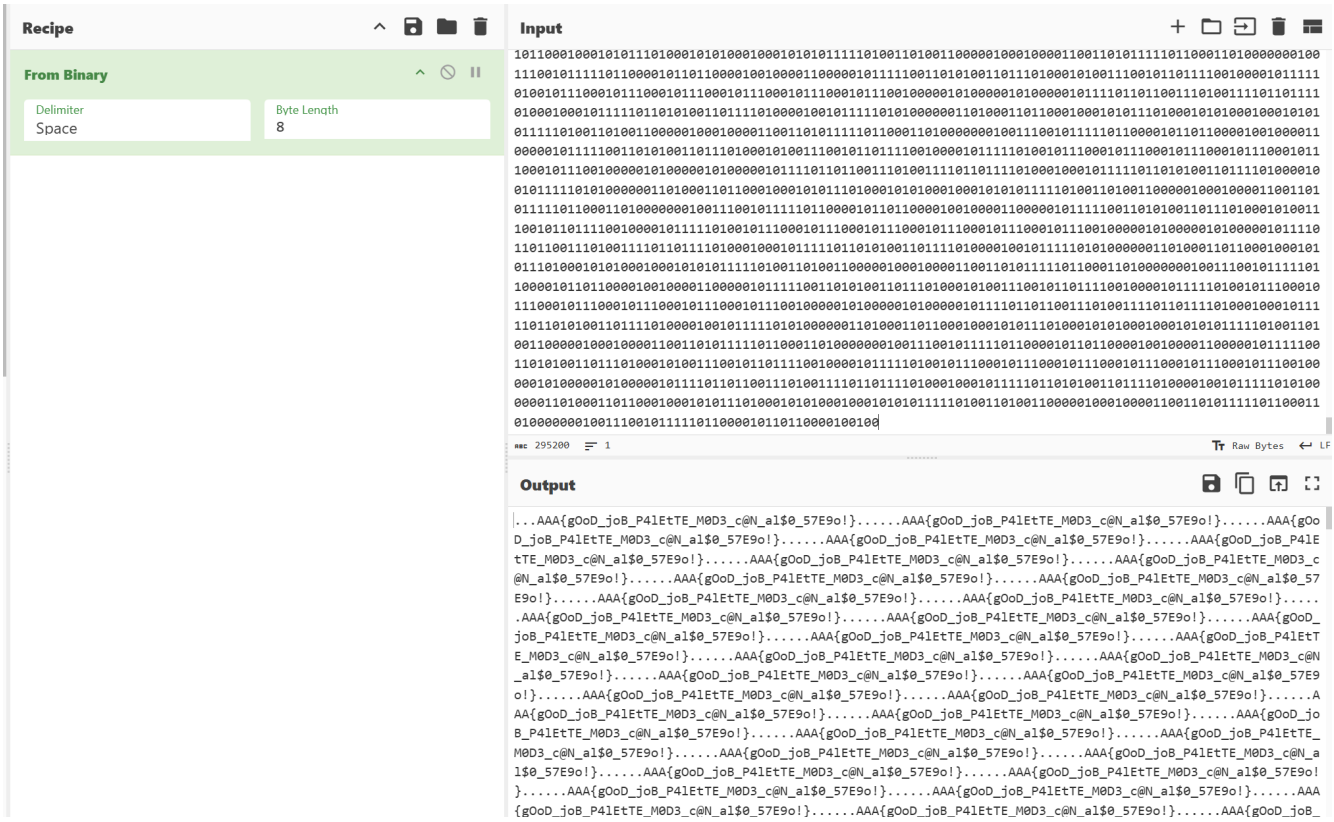
图 1　EzStego 的嵌入

下面根据一般调色板的规则，给调色板的颜色排序，这样就能知道初始的调色板`plst_sorted`：

```
plst_sorted=sorted(plst)
```

在调色板模式中，每一个像素都保存了一个颜色的索引，我通过`pixel_indices = list(im.getdata())`,得到了每个像素相应的在调色板的索引值，找到对应的颜色，再在新调色板里找到对应的索引值。如果索引值是奇数，则`mod 2`为1，否则是0.

最后发现数据有点大，所以放到文件`palette_flag.txt`中，再复制出来放到`CyberChef`里解码：



看到了flag：`AAA{gOoD_joB_P4lEtTE_M0D3_c@N_al$0_57E9o!}`

5 Challenge B: Spectrogram

两首歌名：
《Never Gonna Give you Up》
《一剪梅》

首先先是提取`gif`里的数据，用`ImageSequence`这个库去提取每一帧`frame`,然后不知道为什么不专门设置`RGB`后面用`frame`的时候就会出错，所以就`convert('RGB')`了一下

```
gif_data =
np.array([np.array(frame.copy().convert('RGB').getdata(),dtype=np.uint8).reshape(fra
me.size[1],frame.size[0],3) for frame in ImageSequence.Iterator(file)])
```

接着遍历每一帧`frame`，看生成代码的部分把它逆过来

```
    gif_data = [
        numpy.kron(
            numpy.array([
                [
                    color_pixel if freq % 2 and i < round(
                        frame[freq // 2]) else white_pixel
                    for i in reversed(
                        range(min_db, max_db + 1, quantize))
                ]
                for freq in range(num_freqs * 2 + 1)
            ]),
            numpy.ones([quantize, quantize, 1])
```

```
        )
        for frame in spectrogram.transpose()
    ]
    write_gif(gif_data, dst, fps=sample_rate/frame_step_size)
```

会发现使用了 `kron` 扩大了矩阵，也就是说事实上间距是 `quantize=2`，同时还有 `quantize` 的留白，所以用 `s = i * (quantize * 2) + quantize` 定到矩阵的位置，储存到 `spectrogram` 里。使用 `id` 来确定每一帧的 `index` 来进行还原：

```
for frame in gif_data:
    for i in range(num_freqs):
        s = i * (quantize * 2) + quantize
        data=frame[::quantize,s,0]
        b = (data == 0).sum()
        value=b*quantize+min_db
        spectrogram[id, i]=value
    id+=1
```

- 得到了 `spectrogram` 之后，查 `librosa` 的官方文档，找到了 `power_to_db` 的反面 `db_to_power`，还原到的频谱 `librosa.db_to_power(spectrogram.transpose())`
- 接着变成 `audio`，继续查官方文档，发现两个函数 `istft` 和 `librosa.feature.inverse.mel_to_audio`,在我尝试的时候一直莫名其妙参数报错，然后发现像 `fft_window_size` 这样的参数是默认值，把它删掉，就得到了对应的音频

最后说实话第二首歌我都一下子叫不出歌名了……还得是我妈

19:42 58%

雪花飘飘北风萧萧是什么歌

综合 笔记 视频 图片 AI助手

一剪梅

放到桌面

原唱 费玉清

就在最冷枝头绽放
看见春天走向你我
雪花飘飘北风萧萧
天地一片苍茫
一剪寒梅傲立雪中

查看全部歌词

收藏 评论 999+ 我的音乐

音乐聚合

添加到主屏幕

6 Challenge C RuRu

没做出来qwq 有一些进度qwq 之后实在卡住了qwq

先使用 `binwalk` 命令，发现了藏在里面的压缩包，使用 `binwalk -e ruru.png` 命令进行分离：

```
root@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lab2# binwalk -e ruru.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 613 x 524, 8-bit colormap, non-interlaced
241           0xF1            Zlib compressed data, best compression
77685         0x12F75         Zip archive data, at least v2.0 to extract, compressed size: 42999, uncompressed size: 257
115, name: 999_0061c023.zip
120828        0x1D7FC         End of Zip archive, footer length: 22
```

这个时候打开 `999_0061c023.zip`，发现其中的 `998_00bb4210.zip` 是加密的，结合lab中的提示，可以知道是伪加密，只需要修改其中的加密位把 `01` → `00` 就可以解开。

利用十六进制编辑器打开压缩包，会发现里面不只是 `998_00bb4210.zip` 这个压缩包，其中层层嵌套，应该是套了1000层。这意味着手动改肯定是不行的……

```
50 4B 03 04 0A 00 00 00 00 00 2F 90 29 57 58 35  P K . . . . . . . . / . ) W X 5
B5 BD 5A EB 03 00 5A EB 03 00 10 00 6E 00 39 39  . . Z . . . Z . . . . . n . 9 9
38 5F 30 30 62 62 34 32 61 30 2E 7A 69 70 53 44  8 _ 0 0 b b 4 2 a 0 . z i p S D
59 00 A4 00 00 00 00 08 00 CE 58 E6 91 63 64 60  Y . . . . . . . . . X . . c d `
69 11 61 60 60 30 60 80 00 1F 20 66 64 05 33 59  i . a ` ` 0 ` . . .   f d . 3 Y
45 81 44 ED F7 09 FD CF 8A 66 18 9B 2E A8 B0 78  E . D . . . . . f . . . . . x
C9 8C 5F 8E 89 21 82 81 19 24 2B 20 C2 F0 9F 51  . . _ . . ! . . . $ + . . . Q
9E 81 91 11 A2 56 08 2C 26 01 11 63 82 88 29 00  . . . . . V . , & . . c . . ) .
09 05 10 5B 40 05 22 8E C7 5C 00 55 54 0D 00 07  . . . [ @ . " . . \ . U T . . .
79 42 FC 64 79 42 FC 64 79 42 FC 64 50 4B 03 04  y B . d y B . d y B . d P K . .
0A 00 01 00 00 00 2F 90 29 57 26 39 31 18 59 EA  . . . . . / . ) W & 9 1 . Y .
03 00 59 EA 03 00 10 00 6E 00 39 39 37 5F 37 65  . . Y . . . . . n . 9 9 7 _ 7 e
31 62 32 62 62 35 2E 7A 69 70 53 44 59 00 A4 00  1 b 2 b b 5 . z i p S D Y . . .
00 00 00 08 00 CE 58 E6 91 63 64 60 69 11 61 60  . . . . . X . . c d ` i . a `
60 30 60 80 00 1F 20 66 64 05 33 59 45 81 44 ED  ` 0 ` . . .   f d . 3 Y E . D .
F7 09 FD CF 8A 66 18 9B 2E A8 B0 78 C9 8C 5F 8E  . . . . . f . . . . . x . . _ .
89 21 82 81 19 24 2B 20 C2 F0 9F 51 9E 81 91 11  . ! . . . $ + . . . Q . . . .
A2 56 08 2C 26 01 11 63 82 88 29 00 09 05 10 5B  . V . , & . . c . . ) . . . . [
40 05 22 8E C7 5C 00 55 54 0D 00 07 79 42 FC 64  @ . " . . \ . U T . . . y B . d
79 42 FC 64 79 42 FC 64 50 4B 03 04 0A 00 01 00  y B . d y B . d P K . . . . . .
00 00 2F 90 29 57 30 33 EB 6D 58 E9 03 00 58 E9  . . / . ) W 0 3 . m X . . . X .
03 00 10 00 6E 00 39 39 36 5F 39 62 30 33 39 38  . . . . n . 9 9 6 _ 9 b 0 3 9 8
66 32 2E 7A 69 70 53 44 59 00 A4 00 00 00 00 08  f 2 . z i p S D Y . . . . . .
00 CE 58 E6 91 63 64 60 69 11 61 60 60 30 60 80  . . X . . c d ` i . a ` ` 0 ` .
00 1F 20 66 64 05 33 59 45 81 44 ED F7 09 FD CF  . .   f d . 3 Y E . D . . . .
8A 66 18 9B 2E A8 B0 78 C9 8C 5F 8E 89 21 82 81  . f . . . . x . . _ . . ! . .
19 24 2B 20 C2 F0 9F 51 9E 81 91 11 A2 56 08 2C  . $ + . . . Q . . . . . V . ,
26 01 11 63 82 88 29 00 09 05 10 5B 40 05 22 8E  & . . c . . ) . . . . [ @ . " .
C7 5C 00 55 54 0D 00 07 79 42 FC 64 79 42 FC 64  . \ . U T . . . y B . d y B . d
79 42 FC 64 50 4B 03 04 0A 00 01 00 00 00 2F 90  y B . d P K . . . . . . . . / .
29 57 5B 02 A9 3D 57 E8 03 00 57 E8 03 00 10 00  ) W [ . . = W . . . W . . . . .
6E 00 39 39 35 5F 30 39 33 30 62 34 36 62 2E 7A  n . 9 9 5 _ 0 9 3 0 b 4 6 b . z
69 70 53 44 59 00 A4 00 00 00 00 08 00 CE 58 E6  i p S D Y . . . . . X .
91 63 64 60 69 11 61 60 60 30 60 80 00 1F 20 66  . c d ` i . a ` ` 0 ` . . .   f
64 05 33 59 45 81 44 ED F7 09 FD CF 8A 66 18 9B  d . 3 Y E . D . . . . . f . .
2E A8 B0 78 C9 8C 5F 8E 89 21 82 81 19 24 2B 20  . . . x . . _ . . ! . . . $ +
C2 F0 9F 51 9E 81 91 11 A2 56 08 2C 26 01 11 63  . . . Q . . . . . V . , & . . c
82 88 29 00 09 05 10 5B 40 05 22 8E C7 5C 00 55  . . ) . . . . [ @ . " . . \ . U
54 0D 00 07 79 42 FC 64 79 42 FC 64 79 42 FC 64  T . . . y B . d y B . d y B . d
50 4B 03 04 0A 00 01 00 00 00 2F 90 29 57 FD 49  P K . . . . . . . . / . ) W . I
99 EA 56 E7 03 00 56 E7 03 00 10 00 6E 00 39 39  . . V . . . V . . . . . n . 9 9
34 5F 63 34 65 62 62 37 66 30 2E 7A 69 70 53 44  4 _ c 4 e b b 7 f 0 . z i p S D
```

因此尝试写了一段脚本，对其中的十六进制进行修改。

压缩源文件数据区：
50 4B 03 04：这是头文件标记（0x04034b50）
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密）
08 00：压缩方式
50 4B 01 02：目录中文件文件头标记(0x02014b50)
3F 00：压缩使用的 pkware 版本
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密，这个更改这里进行伪加密，改为09 00打开就会提示有密码了）
08 00：压缩方式
压缩源文件目录结束标志：
50 4B 05 06：目录结束标记
00 00：当前磁盘编号
00 00：目录区开始磁盘编号

由于了解了压缩包对应的头文件之后，回头看这个文件，`50 4B 03 04` 后的第六位是`01 00`，以及`50 4B 01 02`后的第八位是`01`造成了伪加密，所以编写了脚本`modify.py`，对里面相应的位置进行修改。

```
00 01 00 4F 00 00 00 08 FC 01 00 00 00 50 4B 01   . . . O . . . . . . . . P K .
02 1E 00 0A 00 01 00 00 00 28 90 29 57 AC 60 8D   . . . . . . . . ( . ) W . ` .
C1 6D FC 01 00 6D FC 01 00 10 00 11 00 00 00 00   . m . . . m . . . . . . . . .
```

```python
def modifyfile(file_path):
    try:
        # 读取文件内容
        with open(file_path, 'rb') as f:
            content = f.read()
        hex_content=list(content)
        for i in range(len(hex_content)):
            hex_content[i]=hex(hex_content[i])[2:]
        print(hex_content)
        i=0
        while i<len(hex_content)-3:
            if hex_content[i]=='50' and hex_content[i+1]=='4b' and
hex_content[i+2]=='3' and hex_content[i+3]=='4':
                # print(i,hex_content[i:i+7],hex_content[i+6])
                hex_content[i+6]='0'
            elif hex_content[i]=='50' and hex_content[i+1]=='4b' and
hex_content[i+2]=='1' and hex_content[i+3]=='2':
                # print(i,hex_content[i:i+7],hex_content[i+6])
                hex_content[i+8]='0'
            i+=1
        for i in range(len(hex_content)):
            hex_content[i]=int(hex_content[i],16)
        with open('D:/MyRepository/slowist-notebook/docs/Coding/new.zip','wb') as f:
            f.write(bytes(hex_content))
    except Exception as e:
        print(f"An error occurred: {e}")

# 调用函数修改文件
```

```
file_path = 'D:/MyRepository/slowist-notebook/docs/Coding/999_0061c023.zip'
modifyfile(file_path)
```

之后让他自己解压缩，发现里面有一个zip，叫做`The_Password_is_an_8_Character_Hex_String.zip`，解压会发现提示输入密码，这好像就不是伪加密了（

在解开这个密码的部分卡住了qwq

有一点点思路是：发现赛题网站有一个提示`Its original name is 000????????`(:з」∠)_ 然后发现其实这个压缩包都是类似这个结构，比如`998_00bb42a0/997_7e1b2bb5/996_9b0398f2...`，可能和这个会有点关系，但实在是没想到是什么关系qwq