小紫

@2022/11/07

~~八云紫老anghhkjl;l'klhjhgfsdfhjkl;'lkhjfcxsdffhjkhl;lkjhgfsdhjkl;~~

初见

> 得到一张图片，检查发现宽高有问题，将其改回

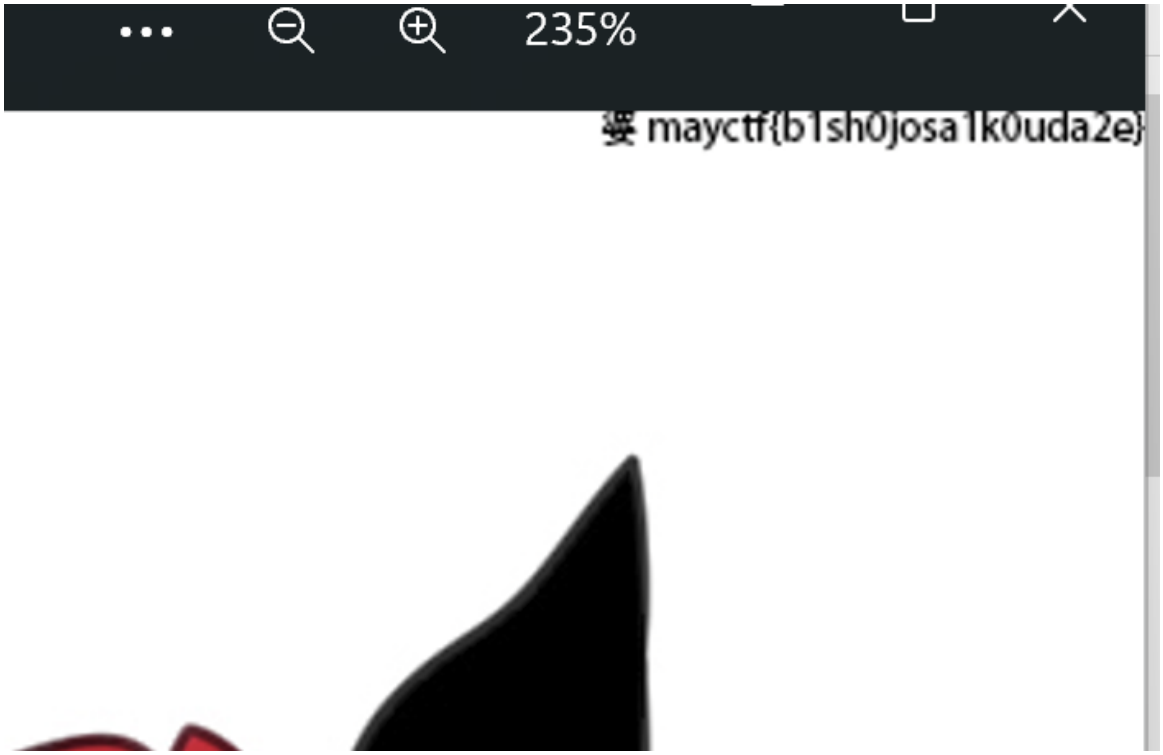
```
PS D:\Data\Coding\Python\Test1> python 2.py -f shojo.png
宽高被改了，是否CRC爆破宽高？(Y/n):Y

CRC32: 0x259e5c86
宽度: 1140, hex: 0x474
高度: 1860, hex: 0x744
PS D:\Data\Coding\Python\Test1>
```

障碍

> 得到一个新bmp图片，由于没有crc校验码，得知原尺寸比较麻烦，所以直接上手改，改到报错为止

解决



得到flag ~~真能藏啊~~

附

在网上找到了一个脚本

爆破

```python
import zlib
import struct
import argparse
import itertools


parser = argparse.ArgumentParser()
parser.add_argument("-f", type=str, default=None, required=True,
                    help="输入同级目录下图片的名称")
args  = parser.parse_args()


bin_data = open(args.f, 'rb').read()
crc32key = zlib.crc32(bin_data[12:29]) # 计算crc
original_crc32 = int(bin_data[29:33].hex(), 16) # 原始crc

```

```
if crc32key == original_crc32: # 计算crc对比原始crc
    print('宽高没有问题!')
else:
    input_ = input("宽高被改了, 是否CRC爆破宽高? (Y/n):")
    if input_ not in ["Y", "y", ""]:
        exit()
    else:
        for i, j in itertools.product(range(4095), range(4095)): # 理论上0x FF FF FF FF, 但考虑到屏幕实际/cpu, 0x 0F FF就差不多了, 也就是4095宽度和高度
            data = bin_data[12:16] + struct.pack('>i', i) + struct.pack('>i', j) + bin_data[24:29]
            crc32 = zlib.crc32(data)
            if(crc32 == original_crc32): # 计算当图片大小为i:j时的CRC校验值, 与图片中的CRC比较, 当相同, 则图片大小已经确定
                print(f"\nCRC32: {hex(original_crc32)}")
                print(f"宽度: {i}, hex: {hex(i)}")
                print(f"高度: {j}, hex: {hex(j)}")
                exit(0)

```

```
bin_data = open(args.f, 'rb').read()
crc32key = zlib.crc32(bin_data[12:29]) # 计算crc
original_crc32 = int(bin_data[29:33].hex(), 16) # 原始c


if crc32key == original_crc32: # 计算crc对比原始crc
    print('宽高没有问题!')
else:
    input_ = input("宽高被改了, 是否CRC爆破宽高? (Y/n):")
    if input_ not in ["Y", "y", ""]:
        exit()
```