ISCC2020部分MISC总结

原创

Qwzf 于 2020-05-29 23:07:57 发布

分类专栏： CTF ISCC 文章标签： CTF

本文链接：https://blog.csdn.net/qq_43625917/article/details/106417191

前言

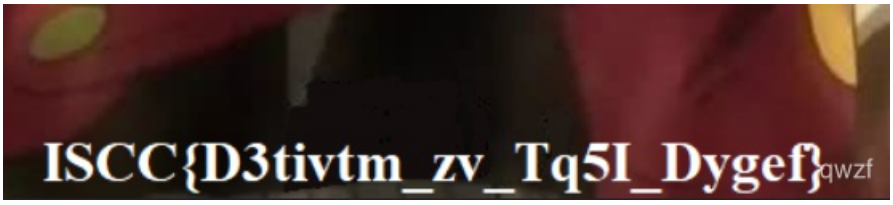
这次ISCC，不得不说杂项题脑洞真大(脑袋坑太少，就做出了几道)，并且没有Crypto题。但Web题目难度还行(不容易的题，主要是涉及脑洞)，由于擂台题的MISC题太难了，所以主要先总结一下练武题的MISC部分题。并且另起一篇总结Web题。

MISC1：ISCC签到

题目难度：简单

考察：图片高度+维吉尼亚密码

在winhex里改图片高度，得到



然后进行维吉尼亚密码解密，并且很容易想到密钥是`high`

MISC2：寻找小明-1

题目难度：普通

考察：Stegsolve+使用cv2库

cv2模块使用

参考：
CV2模块使用（详细教程）
模块cv2的用法
openCV库cv2的使用
首先使用Stegsolve查看不同颜色通道，发现在Red plane 0发现一张二维码

扫描得到一串数字列表

```
[257,1,258,2,259,3,260,4,261,5,262,7,263,8,259,277,438,300,455,319,25,300,456,400,66,366,78,300,421,259,452,23]
```

看到数字列表这种形式，一般考虑对图像进行操作。
然后将数字列表的数字，两个一组作为像素点，发现每个点的R颜色通道值的ASCII码就是flag。脑洞真的大。。。于是写出脚本

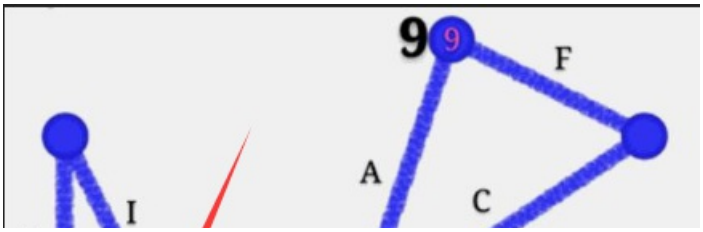
```
import cv2

img=cv2.imread('zd.png')
x=[257,258,259,260,261,262,263,259,438,455,25,456,66,78,421,452]
y=[1,2,3,4,5,7,8,277,300,319,300,400,366,300,259,23]
for i in range(16):
    px = img[y[i], x[i]]
    print(chr(px[2]),end="")
```

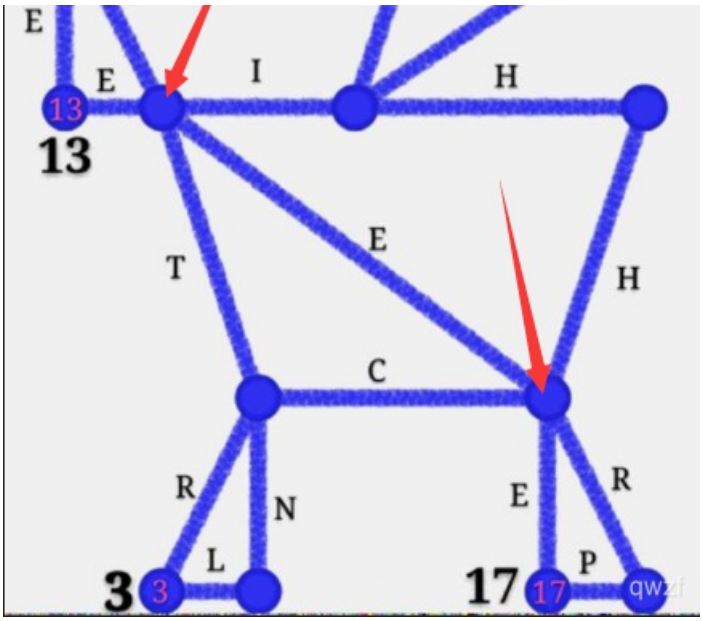
跑脚本，得到flag

```
========== RESTART: C:\Users\
==
flag{I      ny}
>>>
```

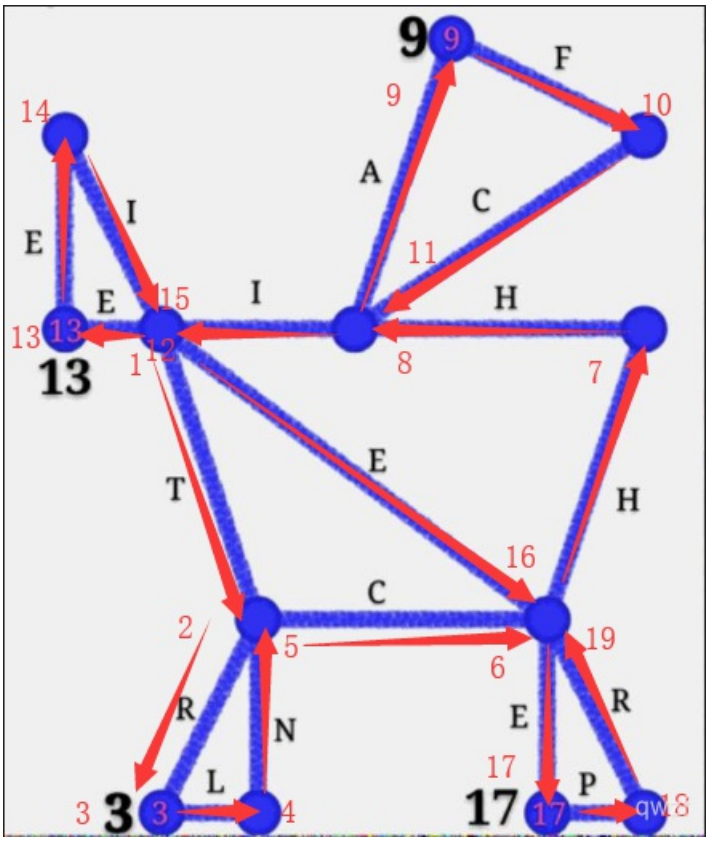
MISC3：ISCC成绩查询-1

题目难度：普通
考察：Stegsolve+图论的哥尼斯堡七桥问题+栅栏密码
使用Stegsolve打开test_ctf.png，查看不同颜色通道，发现一个路径图。看着不太清晰，于是找个脚本把图提取出来(也可以不用提取，直接在Stegsolve的Green plane 1通道看。下边的是我用脚本提取的)：





根据群里师傅们的讨论，涉及图论。然后基本确定是一个哥尼斯堡七桥问题。观察到上图只有箭头所示的点是奇度数节点，所以是存在欧拉通路的半欧拉图。然后从两个奇度数点出发，找到符合图片顺序的欧拉通路。



连接后为：TRLNCHHAFCIEEIEEPR
栅栏密码解密后为：THERAILFENCECIPHER
flag：flag{THERAILFENCECIPHER}

MISC4：神秘组织的邮件-1

题目难度：普通
考察：脑洞
暂时未搞懂咋做，看知乎师傅们的讨论，可以猜到flag，即`flag{加减乘除}`

MISC5：千层套路

题目难度：普通

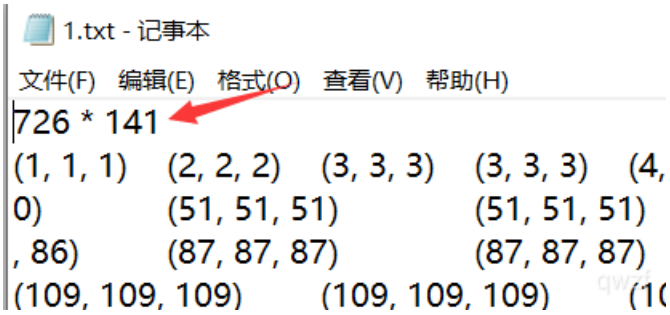
考察：抽帧+拼图+像素点画图+pdf隐藏文字

题目是gif动态图，使用Stegsolve的Frame Browser进行抽帧，提取出来8张图片

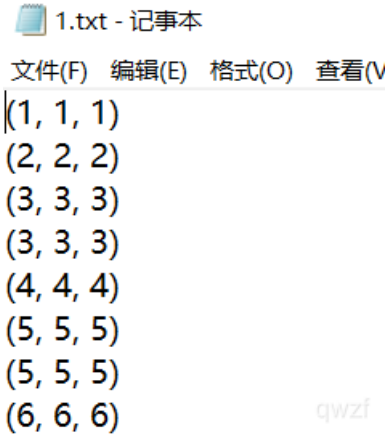
然后进行拼图(在ppt里拼比较快些)，得到

然后使用winrar打开`题目.gif`，发现压缩包(也可以用foremost分离出压缩包)，不过需要密码

而拼图得到的结果就是压缩包密码，输入密码，解压。打开1.txt



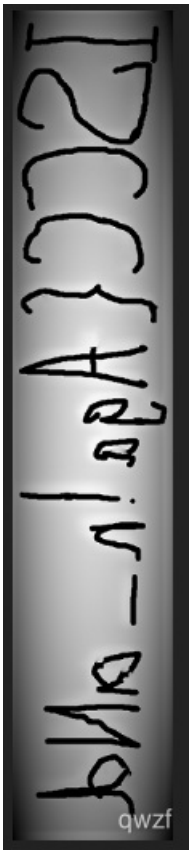
发现箭头所指的应该就是宽高（比较坑的是726是高，141是宽。像素点画图的脚本里会用到）。并且看一下内容，很明显是像素点画图(做MRCTF的千层套路那道题见过)。于是去掉726*141，并改一下格式：



然后将之前的脚本改一下x和y的值，得到如下脚本：

```
from PIL import Image

x = 141    #x坐标  通过对txt里的行数进行整数分
y = 726    #y坐标  x * y = 行数
im = Image.new("RGB", (x, y))
file = open('1.txt')

for i in range(0, x):
    for j in range(0, y):
        line = file.readline()  #获取一行的rgb值
        line = line[:-2]
        line = line[1:]
        #print(line)
        rgb = line.split(", ")  #分离rgb，文本中逗号后面有空格
        im.putpixel((i, j), (int(rgb[0]), int(rgb[1]), int(rgb[2])))
im.save('test2.png')
```

跑脚本，得到



看这个把我眼睛都快看瞎了。希望有更好的方法。。。。。
而这里只有一半flag，接下来看`《道德经》第二十八章.pdf`文件里是否有内容。由文件里“知其白，守其黑”
尝试看看是不是字体为白色。

于是复制出来，和上边一半组合，最终得到flag。

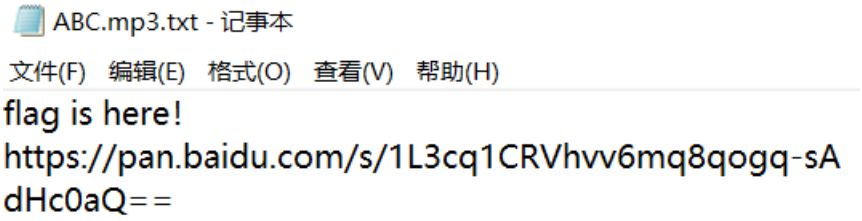
MISC6：耳听为实

题目难度：很难

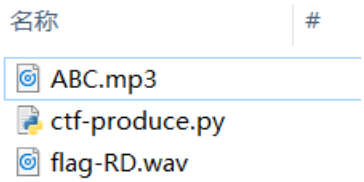
考察：mp3stego工具+python音频信号处理+...

这个题正常做感觉比较难，等有大师傅正常做的wp出来后。再把大师傅的wp链接附上。

首先，题目是`ABC.mp3`，很明显首先想到使用mp3stego工具提取隐藏信息。得到



很明显示是一个网盘链接，且密码是对`dHc0aQ==`的Base64解码。于是打开网盘链接输入分享密码。进行下载，发现依旧是mp3文件，不过使用mp3stego工具毫无作用，其他音频工具也没效果。于是考虑文件分离。文件分离成功，得到



ctf-produce.py

```python
import wave
import numpy as np
import os
# 读取音频信号
f = wave.open(os.path.abspath('./flag.wav'), 'rb') # 二进制只读模式，打开音频文件
params = f.getparams() # 返回音频参数，元组：声道数，量化位数(byte单位)，采样频率，采样点数，压缩类型，压缩类型的描述
nchannels, sampwidth, framerate, nframes=params[:4] # 赋值声道数，量化位数，采样频率，采样点数
str = f.readframes(nframes) # 读取采样点数据，字符串类型
wave_data = np.fromstring(str, dtype=np.short) # 字符串转换为short类型
time = np.arange(0, nframes) * (1.0 / framerate) # 通过采样点数和取样频率计算出每个取样的时间
# 语音信号分帧处理
wlen = 100 # 帧长
inc = 50 # 帧移
signal_length = len(wave_data) # 信号总长度
if signal_length <= wlen: # 若信号长度小于一个帧的长度，则帧数 nf 定义为1，否则，计算帧的总长度
    nf = 1
else:
    nf = int(np.ceil((1.0*signal_length-wlen+inc)/inc))
pad_length = int((nf-1)*inc+wlen) # 所有帧加起来总的铺平后的长度
zeros = np.zeros((pad_length-signal_length), dtype=int) # 不够的长度使用0填补
pad_signal = np.concatenate((wave_data,zeros)) # 填补后的信号记为pad_signal
indices = np.tile(np.arange(0,wlen),(nf,1))+np.tile(np.arange(0,nf*inc,inc), (wlen,1)).T  # 相当于对所有帧的时间点进行抽取，得到nf*wlen长度的矩阵
indices = np.array(indices, dtype=np.int32) # 将indices转化为矩阵
indices = np.random.permutation(indices)
frames = pad_signal[indices] # 得到帧信号
frames = frames.flatten()
w = wave.open(os.path.abspath('./flag-RD.wav'), "wb") # 打开WAV文档
# 配置声道数、量化位数和取样频率
w.setnchannels(nchannels)
w.setsampwidth(sampwidth)
w.setframerate(framerate*2) # 采样频率至少是信号频率最高频率的两倍以上才能重新恢复为原来的模拟信号
w.writeframes(frames.tostring()) # 将wav_data转换为二进制数据写入文件
w.close()
f.close()
```

使用python代码进行音频信号处理得到的flag-RD.wav。审计然后逆着写出flag.wav的恢复脚本，应该可以得到flag.wav(不会写，有点难)。

然后就接不下去了。于是看到一些师傅们在某群里说爆破什么的。于是开个小号，在flag提交框爆破flag，还真的成功了。。

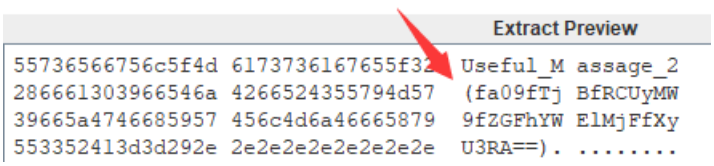
果然flag是个弱口令：`flag{password}`

擂台MISC1：是我DIO哒

题目难度：简单

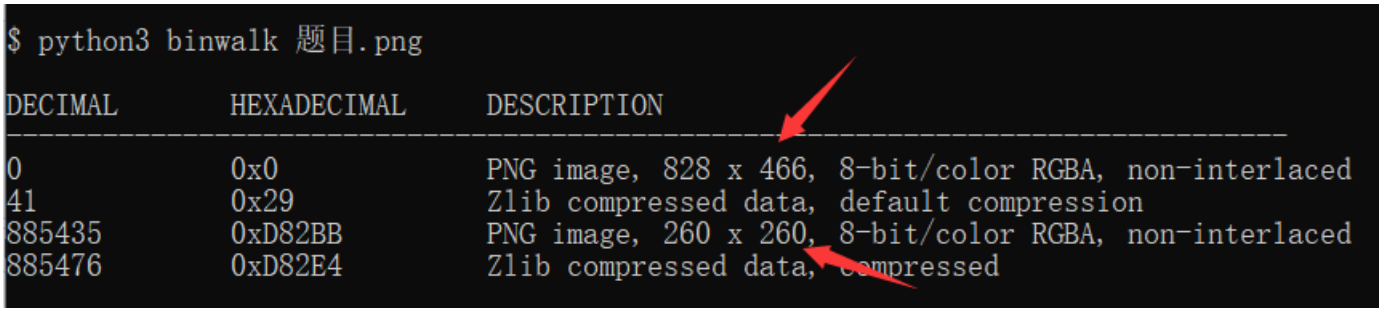
考察：LSB隐写+文件分离+Base64

因为是png图片，首先尝试是否是LSB隐写，发现存在LSB隐写




```
Useful_Massage_2(fa09fTjBfRCUyMW9fZGFhYW ElMjFfXyU3RA==)
```

由于Useful_Massage_2，很明显，上边Useful_Massage_2()里边可能是Base64编码的第二部分。于是继续查找有用信息。
先使用binwalk分析查看图片



发现有两张png图片，然后使用foremost进行分离，得到一张二维码，扫描得

```
Useful_Massage_1(SVNDQyU3QmZsQGdfaXN)
```

两部分括号里的内容按顺序拼接

```
SVNDQyU3QmZsQGdfaXNfa09fTjBfRCUyMW9fZGFhYW ElMjFfXyU3RA==
```

然后Base64解码得：`ISCC%7Bfl@g_is_kO_N0_D%21o_daaaa%21__%7D`
然后进行url解码即得flag。

后记

做了这些杂项题，总共收获了以下新知识：`使用cv2库`、`图论的哥尼斯堡七桥问题`。其他涉及知识点都见过。比如`像素点画图`、`pdf隐藏白色文字`等等。另一篇总结与复现Web题及相关知识点。
ISCC2020练武题Web总结