WeirdSound的解题思路

题目信息

题目名	类型	难度
WeirdSound	MISC	困难

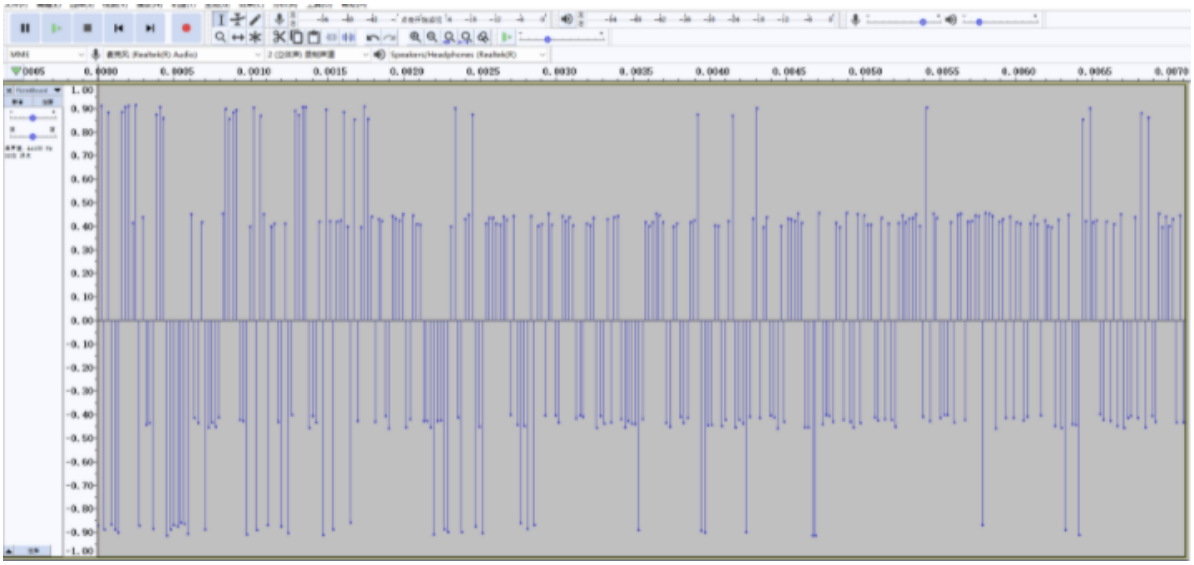
知识点

- Wav振幅高低振幅转换二进制数据

解题步骤

首先`WiredSound`很明显是`wav`文件，添加后缀`.wav`

原声听起来非常嘈杂，使用`Audacity`打开分析



整体看下来，可以发现一个比较明显的特征就是`波形的振幅有高有低`

且高低振幅差距都比较明显，即可猜测振幅高即代表`1`，振幅低代表`0`，从而转换成二进制数据

转换之前先取一些数据做个分析

wav每一帧大小为`两字节`，且存储方式为`小端存储`，所以转换的时候注意高字节位和低字节位换一下

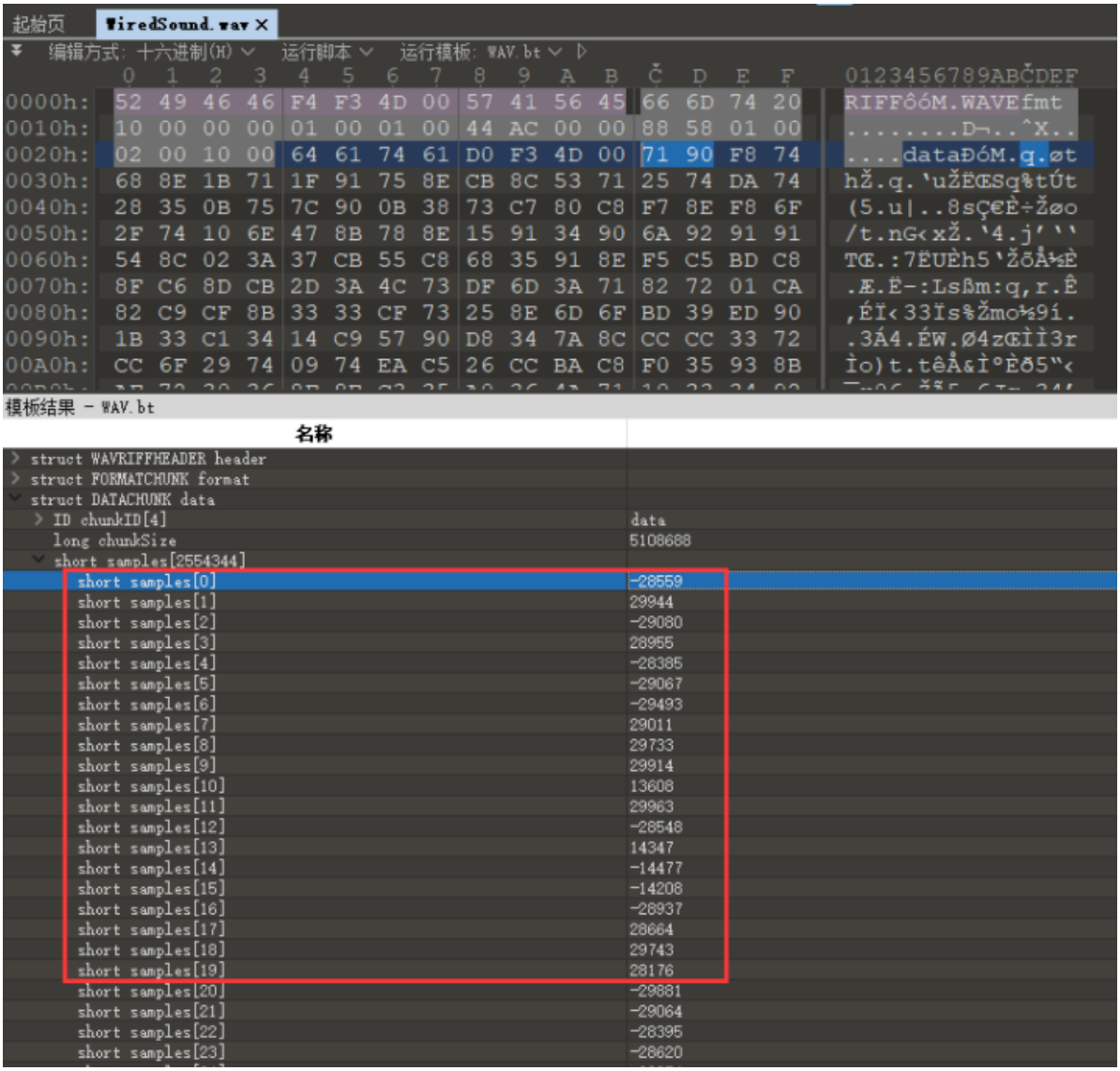
```
import wave

wav = wave.open('WiredSound.wav', 'r')
frames = wav.getnframes()
print("All Frames: {}".format(frames))
framehexdata = wav.readframes(20).hex() # 取前20帧数据做分析
for i in range(0, len(framehexdata), 4):
    data = framehexdata[i:i+4]
    real_data = int(data[2:] + data[:2], 16)
    data1 = data[2:] + data[:2]
    print("第{:<2}帧: {} => {}  数值: {}".format(int((i+4)/4), data, data1,
real_data))
```

运行之后

```
WiredSound\附件1> python .\analyse.py
All Frames: 2554344
第1 帧: 7190 => 9071  数值: 36977
第2 帧: f874 => 74f8  数值: 29944
第3 帧: 688e => 8e68  数值: 36456
第4 帧: 1b71 => 711b  数值: 28955
第5 帧: 1f91 => 911f  数值: 37151
第6 帧: 758e => 8e75  数值: 36469
第7 帧: cb8c => 8ccb  数值: 36043
第8 帧: 5371 => 7153  数值: 29011
第9 帧: 2574 => 7425  数值: 29733
第10帧: da74 => 74da  数值: 29914
第11帧: 2835 => 3528  数值: 13608
第12帧: 0b75 => 750b  数值: 29963
第13帧: 7c90 => 907c  数值: 36988
第14帧: 0b38 => 380b  数值: 14347
第15帧: 73c7 => c773  数值: 51059
第16帧: 80c8 => c880  数值: 51328
第17帧: f78e => 8ef7  数值: 36599
第18帧: f86f => 6ff8  数值: 28664
第19帧: 2f74 => 742f  数值: 29743
第20帧: 106e => 6e10  数值: 28176
```

或者使用`010 Editor`加载wav结构分析插件，也可以看到正确的振幅值



从正振幅上看，对应的数值看起来都是正确的，主要是负振幅的值，每一帧的大小是2字节，而这里负振幅的数据很明显都超过了2字节的最大正数值`2^(15)-1=327687`，这里输出并不能表示负数，所以发生了溢出，转换一下就好了

```
>>> from math import *
>>> pow(2,15)-(36977-pow(2,15))
28559.0 # 第一帧
>>> pow(2,15)-(36456-pow(2,15))
29080.0 # 第三帧
>>> pow(2,15)-(37151-pow(2,15))
28385.0 # 第五帧
```

然后就是解决高低振幅范围的问题，这个其实很简单，多读几帧看看就能确定高振幅范围为：`28000-30000` or `(-30000)-(-28000)`，低振幅范围为：`13000-15000` or `(-15000)-(-13000)`

或者利用`010 Editor`还可以直接通过分析结构中的值直观看出来

确定读取正确数值方法以及判定范围，即可编写Python脚本将所有帧数据转换成二进制数据，然后再转换成文件

```
import wave, math, struct
from binascii import *

obj = wave.open('WiredSound.wav', 'r')
```

```
frames = obj.getnframes()
frames_data = obj.readframes(frames).hex()

bin_data = ''
for idx in range(0, len(frames_data), 4):
    data = frames_data[idx:idx+4]
    data = data[2:] + data[:2]
    if int(data, 16) <= 15000:
        bin_data += '0'
    elif int(data, 16) > 15000 and int(data, 16) <= 30000:
        bin_data += '1'
    elif int(data, 16) > math.pow(2, 15):
        overflow_data = math.pow(2, 15) - (int(data, 16) - math.pow(2, 15))
        if overflow_data > 15000 and overflow_data <= 30000:
            bin_data += '1'
        elif overflow_data <= 15000:
            bin_data += '0'

hex_data = ''
for idx in range(0, len(bin_data), 8):
    hex_data += '{:02x}'.format(int(bin_data[idx:idx+8], 2))

with open('flag.jpg', 'wb') as f1:
    f1.write(unhexlify(hex_data))
```

得到的图片并没有直观看出什么，使用`stegsolve`打开，发现隐藏的flag