

# 考点：流量分析+隐写

## 步骤

1.对1进行流量分析

1为tcp文件传输流量

搜索 flag 寻找到qq文件传输的流量包

The image shows a Wireshark packet capture analysis. The top pane displays a list of network packets. Packet 315 is highlighted, showing an HTTP GET request from 192.168.31.26 to 192.168.31.171. The packet details pane on the left shows the structure of the TCP segment, including the sequence number (99), acknowledgment number (278), and window size (4096). The packet bytes pane on the right shows the raw data of the packet, with the flag 'flag/fla' visible in the payload.

No.	Time	Source	Destination	Protocol	Length	Info
299	6.321478	192.168.31.26	49.4.17.138	TCP	54	57153 → 443 [ACK] Seq=518 Ack=2080 Win=132352 Len=0
300	6.321593	192.168.31.26	49.4.17.138	TLSv1.3	84	Change Cipher Spec, Application Data
301	6.321761	192.168.31.26	49.4.17.138	TLSv1.3	84	Change Cipher Spec, Application Data
302	6.321822	192.168.31.26	49.4.17.138	TCP	54	57151 → 443 [FIN, ACK] Seq=548 Ack=2080 Win=132352 Len=0
303	6.321931	192.168.31.26	49.4.17.138	TCP	54	57153 → 443 [FIN, ACK] Seq=548 Ack=2080 Win=132352 Len=0
304	6.329389	192.168.31.26	120.233.19.16	UDP	241	4022 → 8000 Len=199
305	6.357160	49.4.17.138	192.168.31.26	TCP	54	443 → 57152 [FIN, ACK] Seq=2080 Ack=549 Win=30720 Len=0
306	6.357160	49.4.17.138	192.168.31.26	TCP	54	443 → 57151 [FIN, ACK] Seq=2080 Ack=548 Win=30720 Len=0
307	6.357160	49.4.17.138	192.168.31.26	TCP	54	443 → 57151 [ACK] Seq=2081 Ack=549 Win=30720 Len=0
308	6.357160	49.4.17.138	192.168.31.26	TCP	54	443 → 57153 [FIN, ACK] Seq=2080 Ack=549 Win=30720 Len=0
309	6.357223	192.168.31.26	49.4.17.138	TCP	54	57152 → 443 [ACK] Seq=549 Ack=2081 Win=132352 Len=0
310	6.357291	192.168.31.26	49.4.17.138	TCP	54	57153 → 443 [ACK] Seq=549 Ack=2081 Win=132352 Len=0
311	6.361913	fe2d:fa:11:04:5c	IntelCor_d4:64:f3	ARP	42	192.168.31.171 is at fe2d:fa:11:04:5c
312	6.361937	192.168.31.26	192.168.31.171	TCP	66	57154 → 8083 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_P...
313	6.390572	192.168.31.171	192.168.31.26	TCP	66	8083 → 57154 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS...
314	6.390686	192.168.31.26	192.168.31.171	TCP	54	57154 → 8083 [ACK] Seq=1 Ack=1 Win=131328 Len=0
315	6.391244	192.168.31.26	192.168.31.171	HTTP	331	GET /qqline/q3aTFRq7eFmgtyG4dA+YTLF5jMpad0n7ySsfBa7kR3mB4UG6aMW...
316	6.397367	192.168.31.171	192.168.31.26	TCP	54	8083 → 57154 [ACK] Seq=1 Ack=278 Win=261824 Len=0
317	6.398595	192.168.31.171	192.168.31.26	TCP	152	8083 → 57154 [PSH, ACK] Seq=1 Ack=278 Win=262144 Len=98 [TCP se...
318	6.402607	192.168.31.171	192.168.31.26	TCP	1514	8083 → 57154 [ACK] Seq=99 Ack=278 Win=262144 Len=1460 [TCP segm...
319	6.402607	192.168.31.171	192.168.31.26	TCP	1514	8083 → 57154 [ACK] Seq=1559 Ack=278 Win=262144 Len=1460 [TCP se...

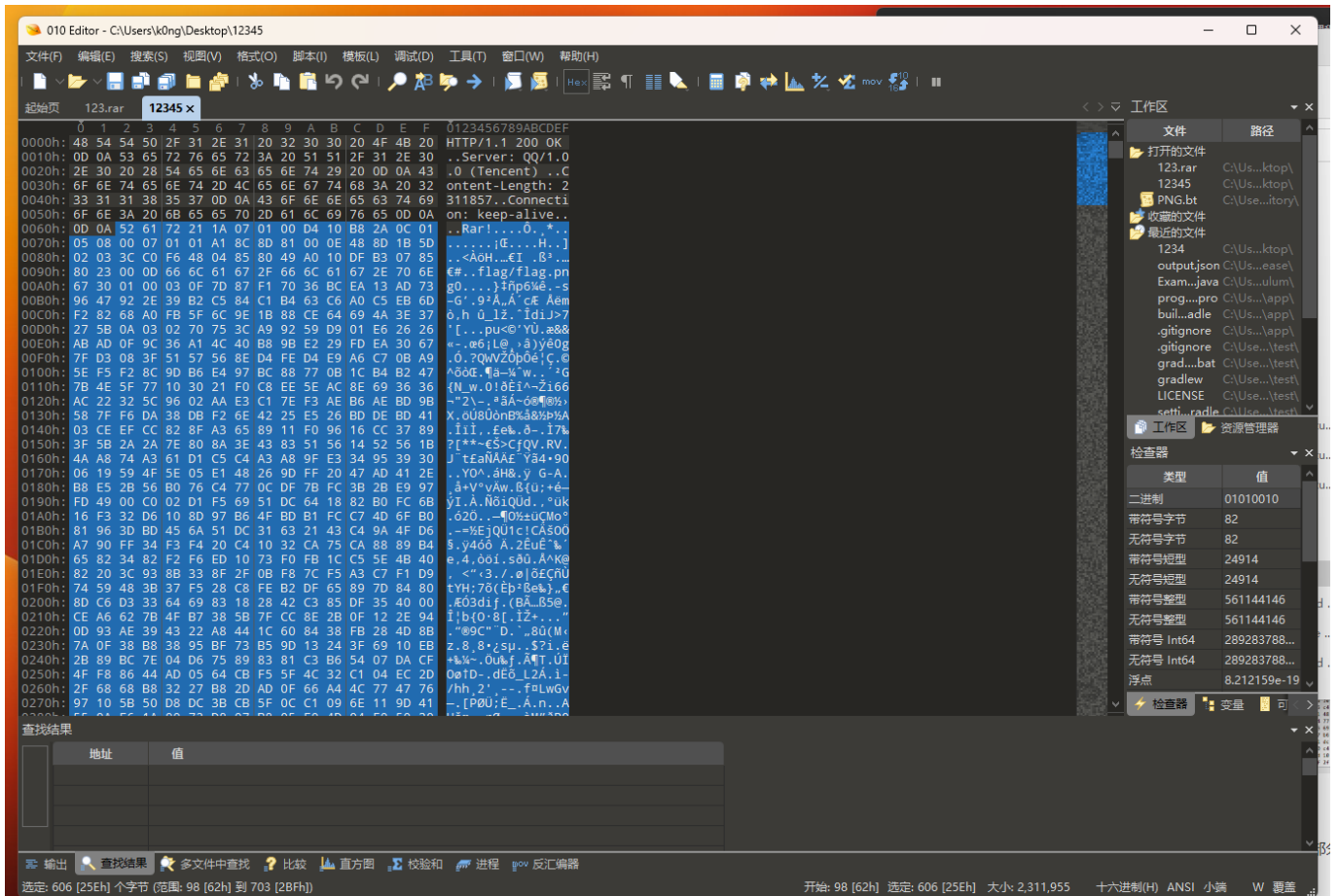
[TCP Segment Len: 1460]  
Sequence Number: 99 (relative sequen...)  
Sequence Number (raw): 4087224562  
[Next Sequence Number: 1559 (relativ...)  
Acknowledgment Number: 278 (relative...)  
Acknowledgment number (raw): 177965884  
0101 .... = Header Length: 20 bytes (5)  
Flags: 0x010 (ACK)  
Window: 4096  
[Calculated window size: 262144]  
[Window size scaling factor: 64]  
Checksum: 0xf2ec [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
[Timestamps]  
[SEQ/ACK analysis]  
TCP payload (1460 bytes)  
[Reassembled PDU in frame: 2042]  
TCP segment data (1460 bytes)

0060 df b3 07 85 80 23 00 0d 66 6c 61 67 2f 66 6c 61 .....#...flag/fla  
0070 67 2e 70 6e 67 30 01 00 03 0f 7d 87 f1 70 36 bc g.png0...}...p6  
0080 ea 13 ad 73 96 47 92 2e 39 b2 c5 84 c1 b4 63 c6 ...s.G..9....c  
0090 a0 c5 eb 6d f2 82 68 a0 fb 5f 6c 9e 1b 88 ce 64 ...m..h..\_l....d  
00a0 69 4a 3e 37 27 5b 0a 03 02 70 75 3c a9 92 59 d9 iJ>7' [...pu<..Y  
00b0 01 e6 26 26 ab ad 0f 9c 36 a1 4c 40 8b 9b e2 29 ...&&....6.L@...  
00c0 fd ea 30 67 7f d3 08 3f 51 57 56 8e d4 fe d4 e9 ..0g...? QWV....  
00d0 a6 c7 0b a9 5e f5 f2 8c 9d b6 e4 97 bc 88 77 0b ....^.....w  
00e0 1c b4 b2 47 7b 4e 5f 77 10 30 21 f0 c8 ee 5e ac ...G{N\_w 0!....^  
00f0 8e 69 36 36 ac 22 32 5c 96 02 aa e3 c1 7e f3 ae ...i66."2\.....~  
0100 b6 ae bd 9b 58 7f f6 da 38 db f2 6e 42 25 e5 26 ....X...8...nB%&  
0110 bd de bd 41 03 ce ef cc 82 8f a3 65 89 11 f0 96 ...A.....e...  
0120 16 cc 37 89 3f 5b 2a 2a 7e 80 8a 3e 43 83 51 56 ...7.?[\*\*...>C.QV  
0130 14 52 56 1b 4a a8 74 a3 61 d1 c5 c4 a3 a8 9f e3 ...RV.J..t..a.....  
0140 34 95 39 30 06 19 59 4f 5e 05 e1 48 26 9d ff 20 4.90..YO ^..H&..  
0150 47 ad 41 2e b8 e5 2b 56 b0 76 c4 77 0c df 7b fc G.A...+V ..v.w...{.  
0160 3b 2b e9 97 fd 49 00 c0 02 d1 f5 69 51 dc 64 18 ;+...I...-iQ..d  
0170 82 b0 fc 6b 16 f3 32 d6 10 8d 97 b6 4f bd b1 fc ...k.-2....0...  
0180 c7 4d 6f b0 81 96 3d bd 45 6a 51 dc 31 63 21 43 ..Mo...= EjQ.1c!C  
0190 c4 9a 4f d6 a7 90 ff 34 f3 f4 20 c4 10 32 ca 75 ..0....4...-2..u  
01a0 ca 88 89 b4 65 82 34 82 f2 f6 ed 10 73 f0 fb 1c ....e..4....s...  
01b0 c5 5e 4b 40 82 20 3c 93 8b 33 8f 2f 0b f8 7c f5 ...^K@..<..3./..|

A data segment used in reassembly of a lower-level protocol (top segment data), 1,460 byte(s) | 分组: 2271 | 已显示: 2271 (100.0%) | 配置: Default

跟踪tcp 以原始数据导出流量包

用010打开导出的流量包，选取rar数据部分另存为rar



## 2.对2进行流量分析

## 2为usb键盘鼠标流量

## 给出脚本

```
#!/usr/bin/env python
# coding:utf-8
import argparse
import os
from tempfile import NamedTemporaryFile

BOOT_KEYBOARD_MAP = {
    0x00: (None, None),
    0x01: ('', ''),
    0x02: ('', ''),
    0x03: ('', ''),
    0x04: ('a', 'A'),
    0x05: ('b', 'B'),
    0x06: ('c', 'C'),
    0x07: ('d', 'D'),
    0x08: ('e', 'E'),
    0x09: ('f', 'F'),
    0x0a: ('g', 'G'),
    0x0b: ('h', 'H'),
    0x0c: ('i', 'I'),
    0x0d: ('j', 'J'),
    # Reserved (no event indicated)
    # ErrorRollOver
    # POSTFail
    # ErrorUndefined
    # a
    # b
    # c
    # d
    # e
    # f
    # g
    # h
    # i
    # j
}
```

0x0e: ('k', 'K'),	# k
0x0f: ('l', 'L'),	# l
0x10: ('m', 'M'),	# m
0x11: ('n', 'N'),	# n
0x12: ('o', 'O'),	# o
0x13: ('p', 'P'),	# p
0x14: ('q', 'Q'),	# q
0x15: ('r', 'R'),	# r
0x16: ('s', 'S'),	# s
0x17: ('t', 'T'),	# t
0x18: ('u', 'U'),	# u
0x19: ('v', 'V'),	# v
0x1a: ('w', 'W'),	# w
0x1b: ('x', 'X'),	# x
0x1c: ('y', 'Y'),	# y
0x1d: ('z', 'Z'),	# z
0x1e: ('1', '!'),	# 1
0x1f: ('2', '@'),	# 2
0x20: ('3', '#'),	# 3
0x21: ('4', '\$'),	# 4
0x22: ('5', '%'),	# 5
0x23: ('6', '^'),	# 6
0x24: ('7', '&'),	# 7
0x25: ('8', '*'),	# 8
0x26: ('9', '('),	# 9
0x27: ('0', ')'),	# 0
0x28: ('\n', '\n'),	# Return (ENTER)
0x29: ('[ESC]', '[ESC]'),	# Escape
0x2a: ('\b', '\b'),	# Backspace
0x2b: ('\t', '\t'),	# Tab
0x2c: (' ', ' '),	# Spacebar
0x2d: ('-', '_'),	# -
0x2e: ('=', '+'),	# =
0x2f: ('[', '{'),	# [
0x30: (']', '}'),	# ]
0x31: ('\\', ' '),	# \
0x32: ('', ''),	# Non-US # and ~
0x33: (';', ':'),	# ;
0x34: ('\\', '"'),	# '
0x35: ('`', '~'),	# `
0x36: ('<', '<'),	# ,
0x37: ('.', '>'),	# .
0x38: ('/', '?'),	# /
0x39: ('[CAPSLOCK]', '[CAPSLOCK]'),	# Caps Lock
0x3a: ('F1', 'F1'),	# F1
0x3b: ('F2', 'F2'),	# F2
0x3c: ('F3', 'F3'),	# F3
0x3d: ('F4', 'F4'),	# F4
0x3e: ('F5', 'F5'),	# F5
0x3f: ('F6', 'F6'),	# F6
0x40: ('F7', 'F7'),	# F7
0x41: ('F8', 'F8'),	# F8
0x42: ('F9', 'F9'),	# F9
0x43: ('F10', 'F10'),	# F10

```

0x44: ('F11', 'F11'), # F11
0x45: ('F12', 'F12'), # F12
0x46: ('[PRINTSCREEN]', '[PRINTSCREEN]'), # Print Screen
0x47: ('[SCROLLLOCK]', '[SCROLLLOCK]'), # Scroll Lock
0x48: ('[PAUSE]', '[PAUSE]'), # Pause
0x49: ('[INSERT]', '[INSERT]'), # Insert
0x4a: ('[HOME]', '[HOME]'), # Home
0x4b: ('[PAGEUP]', '[PAGEUP]'), # Page Up
0x4c: ('[DELETE]', '[DELETE]'), # Delete Forward
0x4d: ('[END]', '[END]'), # End
0x4e: ('[PAGEDOWN]', '[PAGEDOWN]'), # Page Down
0x4f: ('[RIGHTARROW]', '[RIGHTARROW]'), # Right Arrow
0x50: ('[LEFTARROW]', '[LEFTARROW]'), # Left Arrow
0x51: ('[DOWNARROW]', '[DOWNARROW]'), # Down Arrow
0x52: ('[UPARROW]', '[UPARROW]'), # Up Arrow
0x53: ('[NUMLOCK]', '[NUMLOCK]'), # Num Lock
0x54: ('[KEYPADSLASH]', '/'), # Keypad /
0x55: ('[KEYPADASTERISK]', '*'), # Keypad *
}

def parse_boot_keyboard_report(data: bytearray):
    # 数据解析
    modifiers = data[0] # 修改键字节
    keys = data[2:8] # 键码字节

    # 将修改键字节中的位解码为按键修饰符
    ctrl = (modifiers & 0x11) != 0
    shift = (modifiers & 0x22) != 0
    alt = (modifiers & 0x44) != 0
    gui = (modifiers & 0x88) != 0

    # 解析键码字节并将其映射为字符
    characters = []
    for key in keys:
        if key != 0:
            # 键码不为0则查询映射表
            if key in BOOT_KEYBOARD_MAP:
                characters.append(BOOT_KEYBOARD_MAP[key][shift])
            else:
                characters.append(None)
    return (ctrl, shift, alt, gui, characters)

def help_formatter(prog):
    return argparse.HelpFormatter(prog, max_help_position=40)

def main():
    # 解析命令行参数
    parser = argparse.ArgumentParser(
        description='Parse keyboard report data and output as text',
        formatter_class=help_formatter)
    parser.add_argument('pcapng_file', help='path to the pcapng file')
    args = parser.parse_args()

    # 通过tshark解析pcapng文件, 获取键盘数据包

```

```
tmpfile = NamedTemporaryFile(delete=False)
tmpfile.close()

command = "tshark -r %s -T fields -e usbhid.data -e usb.capdata > %s" % (
    args.pcapng_file, tmpfile.name)
os.system(command)

with open(tmpfile.name, 'r') as f:
    lines = f.readlines()

os.unlink(tmpfile.name)

# 解析键盘数据包, 获取输入字符
text = ""
for line in lines:
    capdata = line.strip().replace(':', '')
    if capdata:
        data = bytearray.fromhex(capdata)
        characters = parse_boot_keyboard_report(data)[-1]
        for character in characters:
            if character:
                text += character
    else:
        pass

print(f'Text output:\n{text}')

if __name__ == "__main__":
    main()
```

在tshark目录下进行

```
python test.py 2.pcapng
```

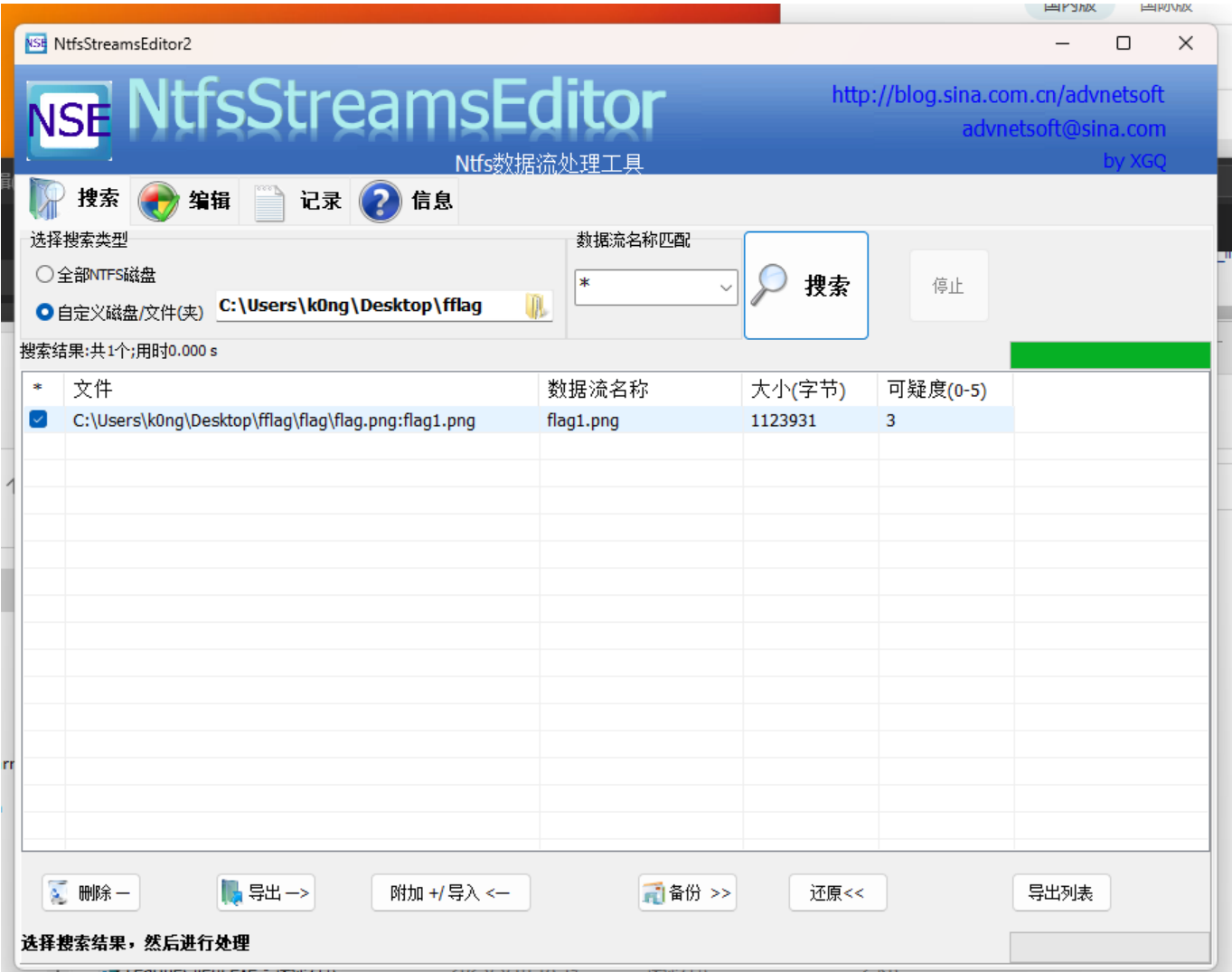
可以在输出中找到rar密码

```
287 5
288 5e[ESC]5[PRINTSCREEN][ESC]5[ESC]5[ESC]5[ESC]55[RIGHTARROW]5 5 5 5\ 5 [UPARROW]5 [L
289 F7q
290 F7
291 [DELETE]F7
292 [DELETE]F7
293 -F7
294 -F7
295 zF7z[ESC]zF7z[ESC]kF7F9[ESC]F6[ESC]F6[ESC]F6[ESC]F6[ESC]F6F6[PAGEUP]F6[PAGEUP]FFFFFFF6e F6
296 F12[INSERT].F12.F12.F12.[PRINTSCREEN].\[PRINTSCREEN].[PRINTSCREEN].[PRINTSCREEN].[PRINTSCREEN].[PR
297 the_Rar_Password_is:zhejiangnormalF4[END];]-[END];][DELETE];][DELETE];][DELETE];][DELETE];][DELETE
298 f'
299 f'
300 f'F5
301 f'l
302 ['0['0['0['0['0['d0[KEYPADASTERISK]'9[KEYPADASTERISK]'9[KEYPADASTERISK]'9[KEYPADASTERISK]'9[KEYPADAS
303 F5
304
305
306
307
```

密码：zhejiangnormal

3.使用rar进行解压后得到flag文件夹

使用工具ntfsstreamseitor.exe检测flag文件夹，发现ntfs流隐藏，导出隐藏的文件为png



4.同时在flag文件夹中也能找到一张相似的图片，显然为盲水印

给出脚本：

```
#!/usr/bin/env python
# -*- coding: utf8 -*-

import sys
import random

cmd = None
debug = False
seed = 20160930
oldseed = False
alpha = 3.0

if __name__ == '__main__':
    if '-h' in sys.argv or '--help' in sys.argv or len(sys.argv) < 2:
        print ('Usage: python bwm.py <cmd> [arg...] [opts...]')
        print ('  cmds:')
        print ('    encode <image> <watermark> <image(encoded)>')
        print ('    image + watermark -> image(encoded)')
        print ('    decode <image> <image(encoded)> <watermark>')
        print ('    image + image(encoded) -> watermark')
        print ('  opts:')
        print ('    --debug,          Show debug')
        print ('    --seed <int>,    Manual setting random seed (default is 20160930)')
        print ('    --oldseed         Use python2 random algorithm.')
        print ('    --alpha <float>, Manual setting alpha (default is 3.0)')
        sys.exit(1)
    cmd = sys.argv[1]
    if cmd != 'encode' and cmd != 'decode':
        print ('Wrong cmd %s' % cmd)
        sys.exit(1)
    if '--debug' in sys.argv:
        debug = True
        del sys.argv[sys.argv.index('--debug')]
    if '--seed' in sys.argv:
        p = sys.argv.index('--seed')
        if len(sys.argv) <= p+1:
            print ('Missing <int> for --seed')
            sys.exit(1)
        seed = int(sys.argv[p+1])
        del sys.argv[p+1]
        del sys.argv[p]
    if '--oldseed' in sys.argv:
        oldseed = True
        del sys.argv[sys.argv.index('--oldseed')]
    if '--alpha' in sys.argv:
        p = sys.argv.index('--alpha')
        if len(sys.argv) <= p+1:
            print ('Missing <float> for --alpha')
```

```
        sys.exit(1)
    alpha = float(sys.argv[p+1])
    del sys.argv[p+1]
    del sys.argv[p]
if len(sys.argv) < 5:
    print ('Missing arg...')
    sys.exit(1)
fn1 = sys.argv[2]
fn2 = sys.argv[3]
fn3 = sys.argv[4]

import cv2
import numpy as np
import matplotlib.pyplot as plt

# OpenCV是以(BGR)的顺序存储图像数据的
# 而Matplotlib是以(RGB)的顺序显示图像的
def bgr_to_rgb(img):
    b, g, r = cv2.split(img)
    return cv2.merge([r, g, b])

if cmd == 'encode':
    print ('image< %s> + watermark< %s> -> image(encoded)< %s>' % (fn1, fn2, fn3))
    img = cv2.imread(fn1)
    wm = cv2.imread(fn2)

    if debug:
        plt.subplot(231), plt.imshow(bgr_to_rgb(img)), plt.title('image')
        plt.xticks([], plt.yticks([]))
        plt.subplot(234), plt.imshow(bgr_to_rgb(wm)), plt.title('watermark')
        plt.xticks([], plt.yticks([]))

    # print img.shape # 高, 宽, 通道
    h, w = img.shape[0], img.shape[1]
    hwm = np.zeros((int(h * 0.5), w, img.shape[2]))
    assert hwm.shape[0] > wm.shape[0]
    assert hwm.shape[1] > wm.shape[1]
    hwm2 = np.copy(hwm)
    for i in range(wm.shape[0]):
        for j in range(wm.shape[1]):
            hwm2[i][j] = wm[i][j]

    if oldseed: random.seed(seed, version=1)
    else: random.seed(seed)
    m, n = list(range(hwm.shape[0])), list(range(hwm.shape[1]))
    if oldseed:
        random.shuffle(m, random=random.random)
        random.shuffle(n, random=random.random)
    else:
        random.shuffle(m)
        random.shuffle(n)

    for i in range(hwm.shape[0]):
        for j in range(hwm.shape[1]):
```



```

        hwm[i][j] = hwm2[m[i]][n[j]]

rwm = np.zeros(img.shape)
for i in range(hwm.shape[0]):
    for j in range(hwm.shape[1]):
        rwm[i][j] = hwm[i][j]
        rwm[rwm.shape[0] - i - 1][rwm.shape[1] - j - 1] = hwm[i][j]

if debug:
    plt.subplot(235), plt.imshow(bgr_to_rgb(rwm)), \
        plt.title('encrypted(watermark)')
    plt.xticks([], plt.yticks([]))

f1 = np.fft.fft2(img)
f2 = f1 + alpha * rwm
_img = np.fft.ifft2(f2)

if debug:
    plt.subplot(232), plt.imshow(bgr_to_rgb(np.real(f1))), \
        plt.title('fft(image)')
    plt.xticks([], plt.yticks([]))

img_wm = np.real(_img)

assert cv2.imwrite(fn3, img_wm, [int(cv2.IMWRITE_JPEG_QUALITY), 100])

# 这里计算下保存前后的(溢出)误差
img_wm2 = cv2.imread(fn3)
sum = 0
for i in range(img_wm.shape[0]):
    for j in range(img_wm.shape[1]):
        for k in range(img_wm.shape[2]):
            sum += np.power(img_wm[i][j][k] - img_wm2[i][j][k], 2)
miss = np.sqrt(sum) / (img_wm.shape[0] * img_wm.shape[1] * img_wm.shape[2]) *
100
print ('Miss %s%% in save' % miss)

if debug:
    plt.subplot(233), plt.imshow(bgr_to_rgb(np.uint8(img_wm))), \
        plt.title('image(encoded)')
    plt.xticks([], plt.yticks([]))

f2 = np.fft.fft2(img_wm)
rwm = (f2 - f1) / alpha
rwm = np.real(rwm)

wm = np.zeros(rwm.shape)
for i in range(int(rwm.shape[0] * 0.5)):
    for j in range(rwm.shape[1]):
        wm[m[i]][n[j]] = np.uint8(rwm[i][j])
for i in range(int(rwm.shape[0] * 0.5)):
    for j in range(rwm.shape[1]):
        wm[rwm.shape[0] - i - 1][rwm.shape[1] - j - 1] = wm[i][j]

```

```

if debug:
    assert cv2.imwrite('_bwm.debug.wm.jpg', wm)
    plt.subplot(236), plt.imshow(bgr_to_rgb(wm)), plt.title(u'watermark')
    plt.xticks([], plt.yticks([]))

if debug:
    plt.show()

elif cmd == 'decode':
    print ('image<%s> + image(encoded)<%s> -> watermark<%s>' % (fn1, fn2, fn3))
    img = cv2.imread(fn1)
    img_wm = cv2.imread(fn2)

    if debug:
        plt.subplot(231), plt.imshow(bgr_to_rgb(img)), plt.title('image')
        plt.xticks([], plt.yticks([]))
        plt.subplot(234), plt.imshow(bgr_to_rgb(img_wm)),
plt.title('image(encoded)')
        plt.xticks([], plt.yticks([]))

    if oldseed: random.seed(seed,version=1)
    else: random.seed(seed)
    m, n = list(range(int(img.shape[0] * 0.5))), list(range(img.shape[1]))
    if oldseed:
        random.shuffle(m,random=random.random)
        random.shuffle(n,random=random.random)
    else:
        random.shuffle(m)
        random.shuffle(n)

    f1 = np.fft.fft2(img)
    f2 = np.fft.fft2(img_wm)

    if debug:
        plt.subplot(232), plt.imshow(bgr_to_rgb(np.real(f1))), \
            plt.title('fft(image)')
        plt.xticks([], plt.yticks([]))
        plt.subplot(235), plt.imshow(bgr_to_rgb(np.real(f1))), \
            plt.title('fft(image(encoded))')
        plt.xticks([], plt.yticks([]))

    rwm = (f2 - f1) / alpha
    rwm = np.real(rwm)

    if debug:
        plt.subplot(233), plt.imshow(bgr_to_rgb(rwm)), \
            plt.title('encrypted(watermark)')
        plt.xticks([], plt.yticks([]))

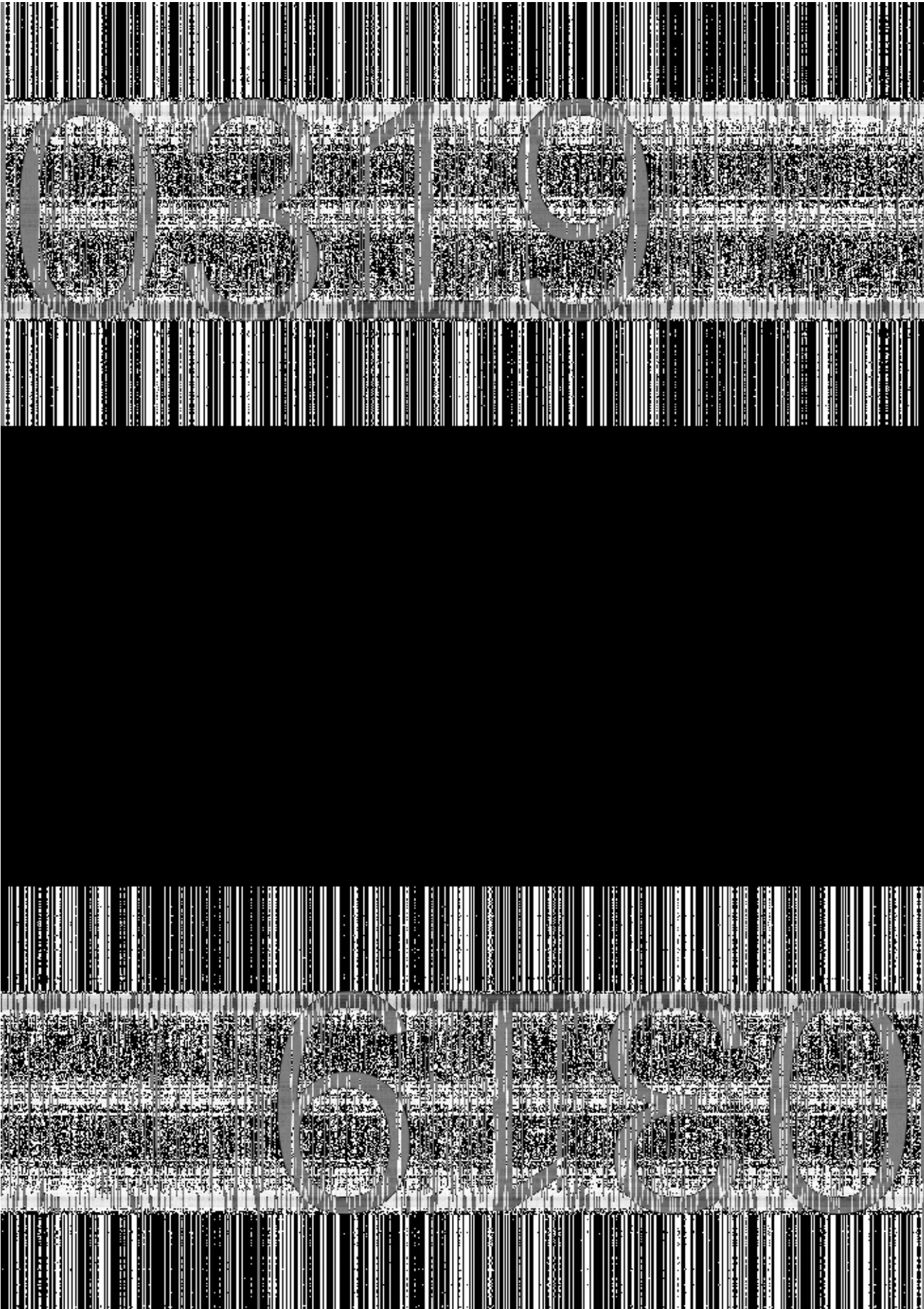
    wm = np.zeros(rwm.shape)
    for i in range(int(rwm.shape[0] * 0.5)):
        for j in range(rwm.shape[1]):
            wm[m[i]][n[j]] = np.uint8(rwm[i][j])
    for i in range(int(rwm.shape[0] * 0.5)):

```

```
        for j in range(rwm.shape[1]):
            wm[rwm.shape[0] - i - 1][rwm.shape[1] - j - 1] = wm[i][j]
    assert cv2.imwrite(fn3, wm)

    if debug:
        plt.subplot(236), plt.imshow(bgr_to_rgb(wm)), plt.title(u'watermark')
        plt.xticks([]), plt.yticks([])

    if debug:
        plt.show()
```



获得密码：0319

5.结合flag.txt中的 Q3EEu5kN84rv2yXPUMRA+TPoYplwmPb3uR/ggiiDcr+T+Ut7l4bXEg== 不难想到rc4加密 使用密码0319解码后得 empudWN0ZntUaDFzX2k4XzdoRV9FbmRfTTFzY30= 再进行base64解码后得到flag zjnuctf{Th1s\_i8\_7hE\_End\_M1sc}