

xctf攻防世界 MISC高手进阶区 奇怪的TTL字段

原创

[i8947943](#) 已于 2022-01-27 21:03:14 修改 3002 收藏 1

分类专栏: [攻防世界misc之路](#) 文章标签: [misc](#)

于 2022-01-27 20:52:27 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/i8947943/article/details/122723363>

版权



[攻防世界misc之路](#) 专栏收录该内容

68 篇文章 2 订阅

订阅专栏

1. 进入环境, 下载附件

题目给的压缩包, 并提示TTL有东西, 我们打开TTL观察一下:

```
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=63
TTL=63
TTL=255
TTL=191
TTL=63
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
```

CSDN @i8947943

没有其他的信息了, 乱码文件不知所措。

2. 问题分析

1. 理解TTL含义

统计了一下, 题目中的TTL有四种, 分别是63,127,191,255, 以往的猜测, 这种数字必然与二进制和ASCII码有关。

```
63 = 00111111
127=01111111
191=10111111
255=11111111
```

可观察到，除了前两位不同，后六位都一样，那么起到重要作用的是前两位。

2. 转换成图片

这一步没思路，参考网上的wp，发现猜想对了一部分，其实目的就是前两位组成有效图片。为什么能想到这里，大佬的脑回路我也理解不了，上代码！

```
import binascii

with open('xctf/pic/ttl.txt', 'r') as file:
    lines = file.readlines()

    ttl_data = ''
    for line in lines:
        prefix = "{0:b}".format(int(line[4:])).zfill(8)
        ttl_data += prefix[0:2]

flag = ''
for i in range(0, len(ttl_data), 8):
    flag += chr(int(ttl_data[i:i + 8], 2))

# 打印出来的flag前一段为ffd8ffe1001845786966000049492a00080000000000000000000000
# ffd8ff明显是图片的文件头，因此需要存储成图片
print(flag)

flag = binascii.unhexlify(flag)
with open('./res.jpg', 'wb') as file:
    file.write(flag)
```

参考格式化代码：[Python-如何将int转换为二进制字符串？](#)

得到结果如图：



这么一小块二维码肯定不行，猜想是否有隐藏信息。

继续分离

丢入kali，binwalk一下，如图：

```
zhangfa@kali: ~/下载
文件 动作 编辑 查看 帮助
(zhangfa@kali)-[~/下载]
└─$ binwalk res.jpg
DECIMAL          HEXADECIMAL     DESCRIPTION
-----          -
0                0x0             JPEG image data, EXIF standard
```

```
0          0x0          JPEG image data, EXIF standard
12         0xC          TIFF image data, little-endian offset of first imag
e directory: 8
5892      0x1704       JPEG image data, EXIF standard
5904      0x1710       TIFF image data, little-endian offset of first imag
e directory: 8
11883     0x2E6B       JPEG image data, EXIF standard
11895     0x2E77       TIFF image data, little-endian offset of first imag
e directory: 8
18711     0x4917       JPEG image data, EXIF standard
18723     0x4923       TIFF image data, little-endian offset of first imag
e directory: 8
25132     0x622C       JPEG image data, EXIF standard
25144     0x6238       TIFF image data, little-endian offset of first imag
e directory: 8
31090     0x7972       JPEG image data, EXIF standard
31102     0x797E       TIFF image data, little-endian offset of first imag
e directory: 8

(zhangfa@kali) - [~/下载]
```

CSDN @I8947943

发现还真有东西！那么继续foremost分离，得到结果如图：



CSDN @I8947943

这么多隐写图片，牛逼！

合成二维码

丢入Photoshop中，我们拼凑一下：



CSDN @I8947943

手机扫码后得到信息。

key:AutomaticKey cipher:fftu{2028mb39927wn1f96o6e12z03j58002p}

5. 在线解密

<https://www.wishingstarmoye.com/ctf/autokey>, 如图:

Key AutomaticKey

flag{2028ab39927df1d96e6a12b03e58002e}

fftu{2028mb39927wn1f96o6e12z03j58002p}

Encode Copy 小写 大写 Clear Decode Copy 小写 大写 Clear

CSDN @18947943

最终的答案为: `flag{2028ab39927df1d96e6a12b03e58002e}`