



后来队友跟我说，可以把每个TTL值二进制码的高两位（高两位比特的数在数据传输中不容易受影响）拿下来，每4个TTL值凑出一个字节的二进制数来。具体为啥想到这个，他跟我说是之前正好看过一篇讲这个的文章。等我找到文章在贴链接。

之后就是写脚本把想法实现了，这里用到了python。

```
with open('ttl.txt') as f:
    lines = f.readlines()
n_num = []
#分析出所有的数
for i in lines:
    if i!='\n':
        n_num.append(int(i.replace('TTL=', '')))
#拿到每个TTL值的高位
r1t = ''
for i in range(0, len(lines)):
    tmp = bin(n_num[i])[2:]
    tmp = '0'*(8-len(tmp)) + tmp
    r1t += tmp[0:2]
#得到最终的结果并保存到文件中
r1t2 = ''
for i in range(0, len(r1t), 8):
    r1t2 += chr(int(r1t[i:i+8], 2))
with open('fi.txt', 'w') as f:
    f.write(r1t2.rstrip())
```

得到的文件中全部都是可打印字符，以FFD8开头，FFD9结束。所以，是jpeg格式的图片无疑了。

```
ffd8ffe1001845786966000049492a00080000000000000000000000000000ffec00114475636b
652e636f6d2f7861702f312e302f003c3f787061636b657420626567696e3d22efbbbf22
223f3e203c783a786d706d65746120786d6c6e733a783d2261646f62653a6e733a6d6574
2e332d633031312036362e3134353636312c20323031322f30322f30362d31343a35363a
64663d22687474703a2f2f7777772e77332e6f72672f313939392f30322f32322d726466
6e207264663a61626f75743d222220786d6c6e733a786d703d22687474703a2f2f6e732e
4d4d3d22687474703a2f2f6e732e61646f62652e636f6d2f7861702f312e302f6d6d2f22
2e636f6d2f7861702f312e302f73547970652f5265736f75726365526566232220786d70
20435336202857696e646f7773292220786d704d4d3a496e7374616e636549443d22786d
4545414139353446322220786d704d4d3a446f63756d656e7449443d22786d702e646964
35344632223e203c786d704d4d3a4465726976656446726f6d2073745265663a696e7374
```

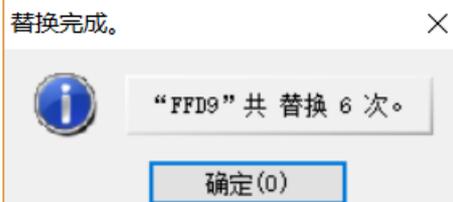
```
56b03cd394659f26fb338199ab0d1552a28c446ebf1afbeacee58129ec31f8dbeebfa6580f618fc6df75fd32c07b0c7e36fbaf99603d863f1b7dd7f4cb035762
f62fd15f3bfdb7ce79ce57fbae570b2b8dfd72f5558ff92ebac0c0d5905e706959981c7cb73762e58e669c4c3cca264aba2f2554d77dd505fdffb04d5ec31f8dbee
bfa6580f618fc6df75fd32c0c0d94e9bfd196aa753de7173ccb1518e5b2796a7116495aebc75afb06eba9eeded81d5602c0580b01602c0580b01602c0580b07f
ffd9
```

之后把这些16进制字符粘贴到winhex里并保存为jpeg格式的图片，得到了一张30KB的残缺二维码。30KB，还残缺!!!一定是夹杂私货了，于是就搜了下FFD8和FFD9的数量，正好6对。



(用搜索的话只能定位，可以用替换搜索，这样可以计数)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00001680	28	D8	E3	9E	73	DD	25	CE	87	B7	C3	80	DB	D7	D3	7F	(?錶s?%?囁胖圩?瓦
00001690	B4	DB	A9	2D	7B	98	B4	8C	B2	12	D0	E5	B2	79	98	A5	篡??{?礲?. 绣晒榭?
000016A0	9C	69	B1	C7	3C	97	3E	73	9D	0F	67	87	00	C0	DD	2B	洪鼻<?>s..g?.??译
000016B0	17	9E	EC	4A	9D	A9	9E	D3	B5	19	1F	E3	25	4F	82	42	.?冀.?苒???.?%0佟
000016C0	82	CF	F9	38	CA	8A	29	50	F9	4C	F0	BF	01	CA	D9	4D	傻??舜)P鹁鹁.?費?
000016D0	A8	F6	65	6A	AA	A0	EA	9A	BE	65	71	8B	B3	32	32	D0	ej猶隱縷q??弟2?
000016E0	CC	24	A2	A0	82	61	CF	74	97	BE	2E	DF	06	00	FF	00	?Z 俛蝶控?.?..
000016F0	01	B0	1B	01	.?..??.?..?..?..												
00001700	B0	1F	FF	D9	FF	D8	FF	E1	00	18	45	78	69	66	00	00	?b ..Exif..?
00001710	49	49	2A	00	08	00	00	00	00	00	00	00	00	00	00	00	II*.....
00001720	FF	EC	00	11	44	75	62	6B	70	00	01	00	04	00	00	00	..Ducky.....
00001730	3C	00	00	FF	E1	03											<.. ??+http://ns?
00001740	2E	61	64	6F	62	65											1 .adobe.com/xap/1
00001750	2E	30	2F	00	3C	3F											5 .0/.<?xpacket be
00001760	67	69	6E	3D	22	EF											5 gin="?蒙" id="W5.
00001770	4D	30	4D	70	43	65											4 M0MpCehiHzreSzNT
00001780	63	7A	6B	63	39	64											0 czkc9d"?> <x:xmp?
00001790	6D	65	74	61	20	78	6D	6C	6E	73	3A	78	3D	22	61	64	meta xmlns:x="ad
000017A0	6F	62	65	3A	6E	73	3A	6D	65	74	61	2F	22	20	78	3A	obe:ns:meta/" x:.
000017B0	78	6D	70	74	6B	3D	22	41	64	6F	62	65	20	58	4D	50	xmptk="Adobe XMP
000017C0	20	43	6F	72	65	20	35	2E	33	2D	63	30	31	31	20	36	Core 5.3-c011 6.
000017D0	36	2E	31	34	35	36	36	31	2C	20	32	30	31	32	2F	30	6.145661, 2012/0.
000017E0	32	2F	30	36	2D	31	34	3A	35	36	3A	32	37	20	20	20	2/06-14:56:27
000017F0	20	20	20	20	20	22	3E	20	3C	72	64	66	3A	52	44	46	"> <rdf:RDF
00001800	20	78	6D	6C	6E	73	3A	72	64	66	3D	22	68	74	74	70	xmlns:rdf="http.
00001810	3A	2F	2F	77	77	77	2E	77	33	2E	6F	72	67	2F	31	39	://www.w3.org/19.
00001820	39	39	2F	30	32	2F	32	32	2D	72	64	66	2D	73	79	6E	99/02/22-rdf-syn
00001830	74	61	78	2D	6E	73	23	22	3E	20	3C	72	64	66	3A	44	tax-ns#"> <rdf:D.
00001840	65	73	63	72	69	70	74	69	6F	6E	20	72	64	66	3A	61	escription rdf:a
00001850	62	6F	75	74	3D	22	22	20	78	6D	6C	6E	73	3A	78	6D	bout="" xmlns:xm
00001860	70	3D	22	68	74	74	70	3A	2F	2F	6E	73	2E	61	64	6F	p="http://ns.ado
00001870	62	65	2E	63	6F	6D	2F	78	61	70	2F	31	2E	30	2F	22	be.com/xap/1.0/"
00001880	20	78	6D	6C	6E	73	3A	78	6D	70	4D	4D	3D	22	68	74	xmlns:xmpMM="ht
00001890	74	70	3A	2F	2F	6E	73	2E	61	64	6F	62	65	2E	63	6F	tp://ns.adobe.co
000018A0	6D	2F	78	61	70	2F	31	2E	30	2F	6D	6D	2F	22	20	78	m/xap/1.0/mm/" x
000018B0	6D	6C	6E	73	3A	73	74	52	65	66	3D	22	68	74	74	70	mlns:stRef="http.



接着就是把图片拆出来了，使用binwalk，会分出6张二维码图片，用PPT的插图和对齐的功能拼接一下，得到一

个二维码。



(PPT拼图真好用)

本以为扫完码flag就出来了，没想到还有一层加密，带密钥的，这里我们使用的autokey解密。

之前说的维吉尼亚，不好意思，最开始我们用的维吉尼亚是个错误的被flag包裹的字符串，后来用的和他相似的autokey通过的。感谢那位发现错误我的博主。

扫码后的结果。

key:AutomaticKey cipher:fftu{2028mb39927wn1f96o6e12z03j58002p}

使用autokey解开密文：（用了个在线的，可能会慢一点）

<http://www.practicalcryptography.com/ciphers/classical-era/autokey/>

得到这么个东西：flagabdfdeabee，数字和括号没处理，自己手动加上就过了。

flag{2028ab39927df1d96e6a12b03e58002e}

参加完这个比赛，感觉除了隐写能写点，其他的，真的是欠缺！尤其那个PWN和reverse，得好好努力了。或许我需要一个大师傅来带带我，求带啊！

转载于：<https://www.cnblogs.com/kevinbruce656/p/10667333.html>