杂项题目练习（五）

公众号Lemonsec

前言：

以下是我在 bugku 练习的解题思路，编号跟我前面分享的基础是对应的，理论基础结合实践。

所有题目目录如下：

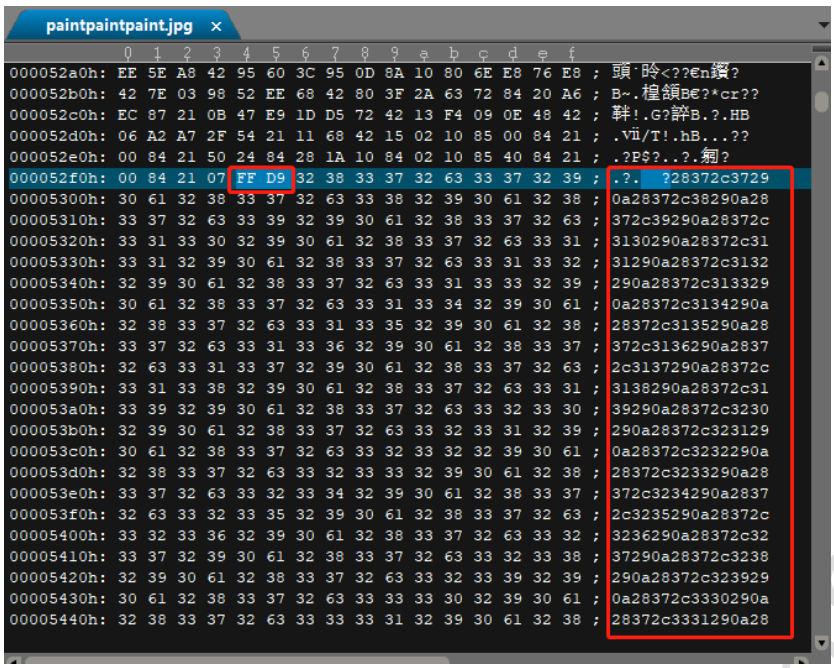
以下是对 34-40 题的介绍

杂项第三十四题：图穷匕见

https://ctf.bugku.com/files/f6697e1f904a0c30b56f72fcf0023434/paintpaintpaint.jpg
打开图片保存到本地
查看属性，有些提示看不懂先保存下来

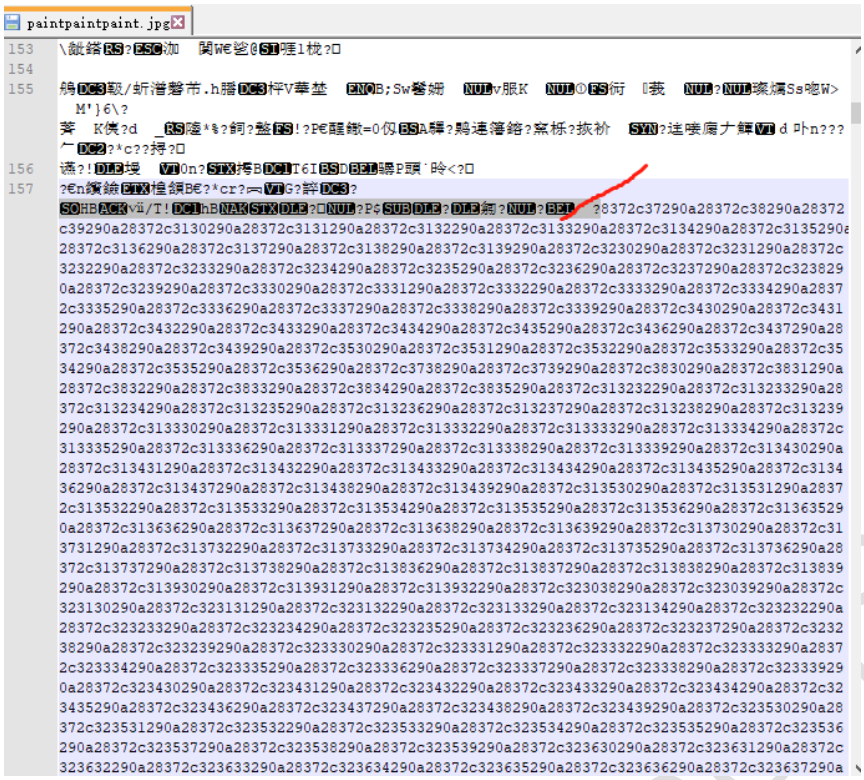
图片的标题图穷 flag 见以及题目图穷匕见都暗示该图片在文件末尾隐藏了信息。
用 16 进制编辑器打开图片，找到 jpg 的文件尾 **FF D9**，发现其后还有大量的数据

LemonSec

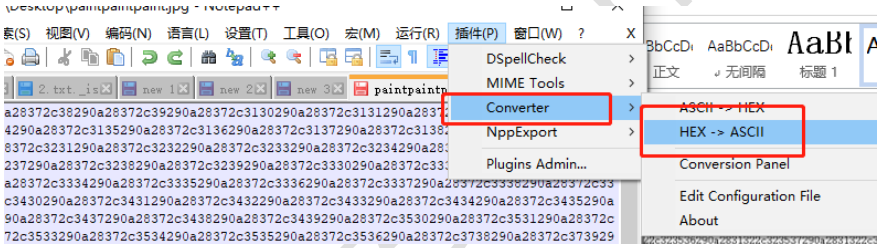


将之后的数据保存到 txt 中

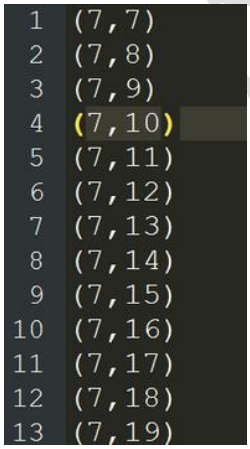
其实用 notepad++或者 vim 打开就可以看到后面大量的数据。



乍一看无从下手，其实只要尝试将数据按 **16 进制->ASCII** 方式解码，思路就很明显了，下图中使用的是 notepad++中的插件 Converter 进行解码



解码结果如下，很明显是坐标的形式



这时候再结合**会画图吗**的提示，将这些坐标做成一张图即可，用 **gnuplot** 这个工具比较方便，因此将坐标转为 gnuplot 能识别的格式 **坐标 1 坐标 2**

```
 1 7 7
 2 7 8
 3 7 9
 4 7 10
 5 7 11
 6 7 12
 7 7 13
 8 7 14
 9 7 15
10 7 16
11 7 17
12 7 18
13 7 19
```

在 Linux 中使用

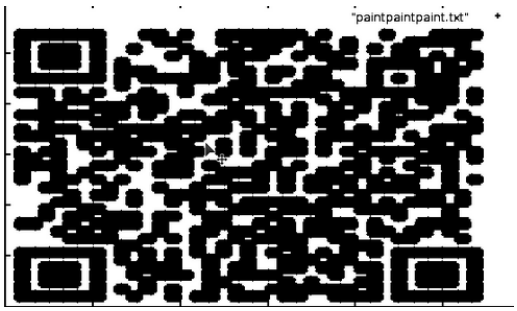
gunplot

plot "文件名"（注意“）

如下图

得到一个二维码

对该二维码在做灰度等的处理后，扫描可得 flag



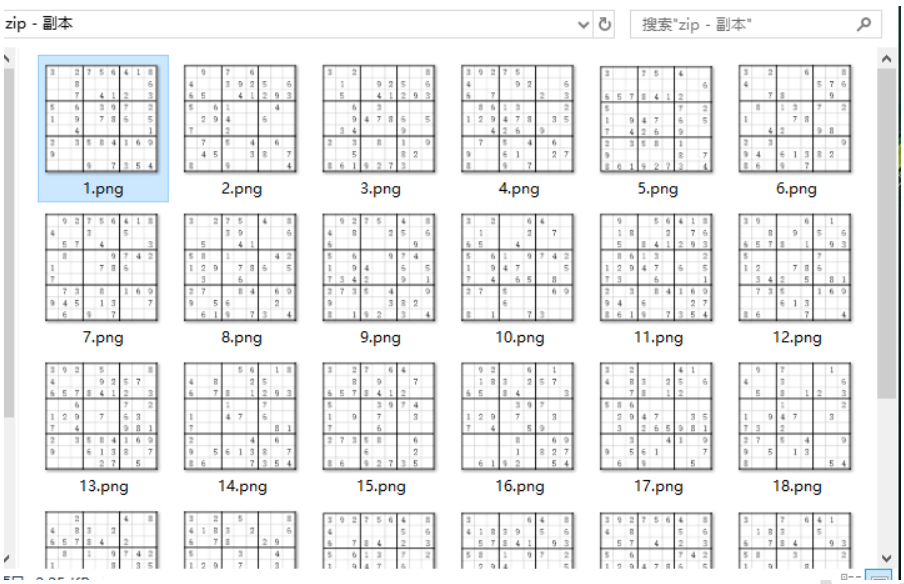
flag 为 **flag{40fc0a979f759c8892f4dc045e28b820}**

上面手工梳理的话工作量 很大，可以代码解决

```
import    matplotlib.pyplot    as plt
i=0
fig=plt.figure()
with open("text.txt") as f:
        for data in f.readlines():
                data=data.strip()
                data=eval(data)
                plt.scatter(data[0],data[1],c="255",marker=".")
                i=i+1
                print("\r\n[+] Has dealed",i,"lines")

plt.show()
```

杂项第三十五题：很普通的数独(ISCCCTF)

下载压缩包
https://ctf.bugku.com/files/2678e3fdabcb61c0ae67719c27732497/zip
下载后改成.zip 后缀
解压后得到许多图

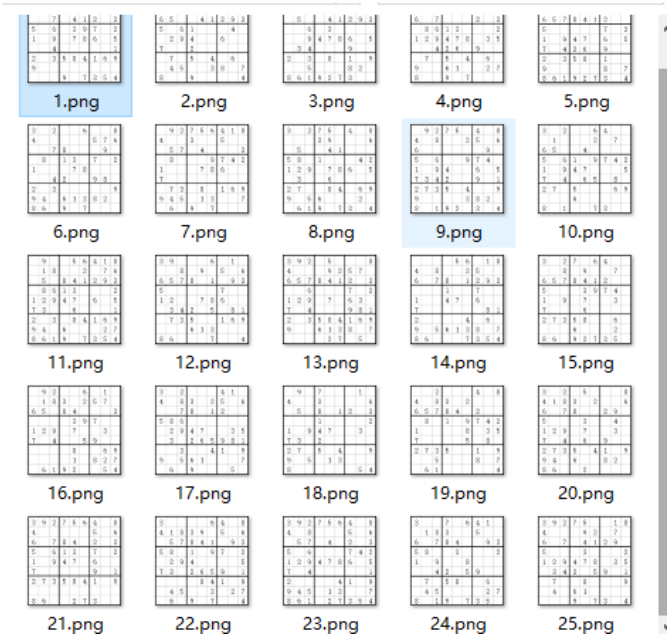


开始以为是让把这个数独完成才能得到下一步线索，于是在把图片拖到 PowerPoint 里，背景色设为透明，找了几个合适的图片叠了起来，得到了完整的数独的答案：

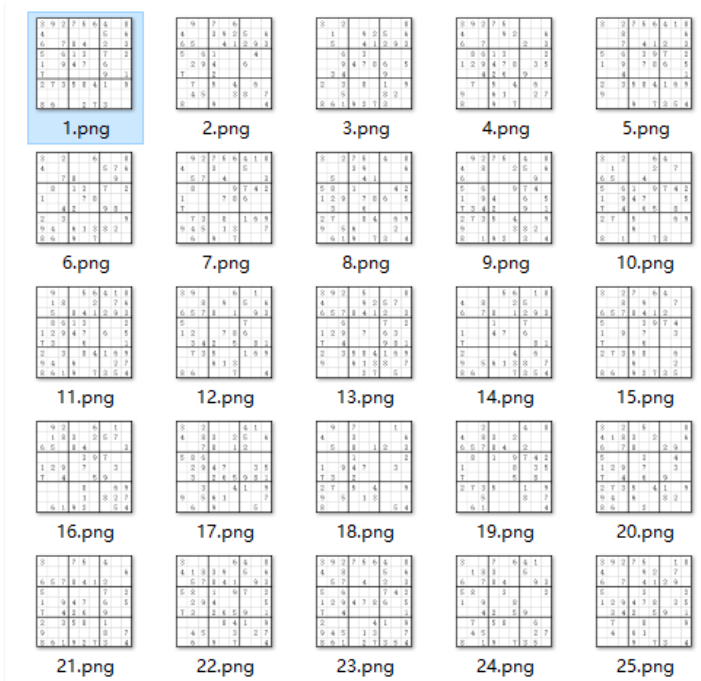
3	9	2	7	5	6	4	1	8
4	1	8	3	9	2	5	7	6
6	5	7	8	4	1	2	9	3
5	8	6	1	3	9	7	4	2
1	2	9	4	7	8	6	3	5
7	3	4	2	6	5	9	8	1
2	7	3	5	8	4	1	6	9
9	4	5	6	1	3	8	2	7
8	6	1	9	2	7	3	5	4

谁知道并没有什么卵用，只有看别人的 wp 了。正确方法是把这 25 个图片按 5×5 排列，然后把有数字的格记为数字 1(代表黑色)，没有的记为 0(代表白色)，再把得到的数字用 python 画出二维码。

公众号 Memorisec



1.png,5.png,21.png 仔细看看就是是二维码的定位形状，三个角上的方形块，但是按排列的画，这三个图的顺序不对，需要将图片 1.png,5.png,21.png 重命名成:5.png,21.png,1.png,



然后把 01 提取出来:

我将有数字的格子写成 0，没有的写成 1，

比如第一行写成 00000001010101011101011111000000111101000000，然后用之前写的 01 数字转换成图像的脚本生成了二维码 hhh

```
11111110101010100010100000111111000001011111111
10000010110011111010100111011000110010010000001
10111010111001111101001111110100010100101011101
10111010110110001000101000001111000110101101101
```

10111010001110010000111110111111101110101110
10000010110010000001100010000111010000100000 1
11111110101010101010101010101010101011101111111
00000000001100110100100011010011001111000000000
11001110010010000111111110010010100000010111 1
10100100101111111110111010101111010110100 1100
10000011110010010000011000110100110101000 1010
00110001001101000101001100010000001011001000 0
01011010101000111111010001110100111010110111 1
10001100010001110011101110110110001011011100 01
00110011010000000001001000011110010110101101 0
10100000101101011111001101111110100111010001 1
11011111011101100110110001010000111000010000 0
11010100001010100001110110110111010110100 1100
01001111111000101111101000100001101110110 1100
01100101100101010110001111010100110000101001 0
01011111111110101111111110110110111111111 1100
01111000110000010000100010100010010010001111 0
11111010111001110011101011010011010010101001 0
11001000101110101110100011110000001110001000 0
10101111101110011110111111110000101011111001 0
11010001100011100010011110110111110100010001 0
11110111111000100100001101011000111111011111 0
01100101010100011001010001000100010110101000 1
01110111010110110110010000110110100011110100 1
11011000100110110001010110111111010010110011 0
00001110011100000000010000101010111100010010
11101001001111001110111001010000101111101001 0
10100110001011111111010000010000101010101010 0
00001001100100110111010100111110010111110110 1
00001011110111000110101100000100010111010011 0
01111001101010001010000001101100000111001000 0
10011010010000110111111110110010111011111001 1
00000000111111010110100010101110010010001101 0
11111110001111101101101010110111001110101111 0
10000010111010110110100011111001000110001000 1
10111010101110000111111110110100100011111101 1
10111010011011110110100000100110110001110110 1
10111010000000111011000011010101100100100010 1
10000010101100101111101100101100001101011000 0
11111110101010100111101111010110111000010110 1

脚本如下：

```
from PIL import Image
```

公众号Lemonsec

```
x = 45
y = 45

im = Image.new('RGB', (x, y))
white = (255, 255, 255)
black = (0, 0, 0)

with open('file.txt') as f:
    for i in range(x):
        ff = f.readline()
        for j in range(y):
            if ff[j] == '1':
                im.putpixel((i, j), black)
            else:
                im.putpixel((i, j), white)
im.save("1.jpg")
```

二维码:

扫出来是:

Vm0xd1NtUXlWa1pPVldoVFlUSlNjRlJVVGtOamJGWnlWMjFHVlUxV1ZqTldNakZlWVcxS1lxTnNhRmhoTVZweVdWUkdXbVZHWkhOWGJGcHBWa1paWExacIpEUmhNVXBYVW14V2FHVnFRVGs9

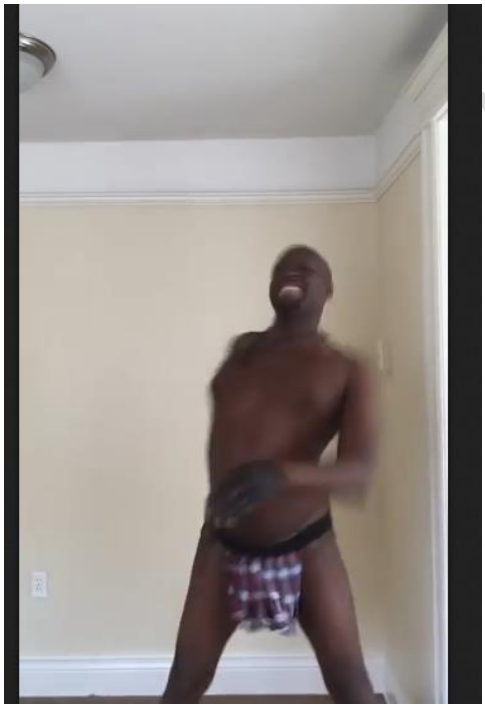
是个多层的 base64，解密拿到 flag。

flag{y0ud1any1s1}

公众号 LemonSec

杂项第三十六题: PEN_AND_APPLE

下载 mp4
https://ctf.bugku.com/files/2192f6a3bb1978887b4941a286727b56/test.mp4
之前做了 mp3,做了音频，这是头一次碰到视频。



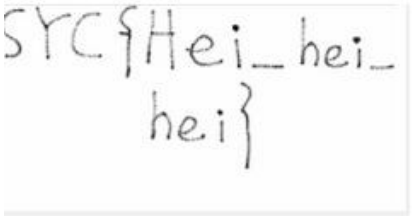
。。。。。
视频另存到桌面
提示是 Windows 下的 type 命令，Windows 下的 type 命令可以显示文件内容。

用 alternatestreamview 可以提取出其中的文件，得到 flag

Lemonsec

(说实话自己做了做不出来哈啊哈哈哈哈哈哈)

这道题有些脑洞，不过也是一种很常见的隐写技术，后来提示与 win 的 type 命令有关，可以找到相关资料，type 命令可以用于 ntfs 文件写入，通过工具“`alternatestreamview“`扫描文件得到如下所示



SYC{Hei_hei_hei}

之后又顺带下载了大佬的一个 NTFS 数据流扫描工具，

利用 NTFS 流文件隐藏

首先用记事本新建两个文本文档,分别名为“1.txt”“2.txt”,其内容为“正常文件、数据流文件”,打开 CMD 命令行窗口，进入两个文件所在文件夹,输入 type 2.txt>1.txt: shujuliu.txt,回车.即可将文件 2.txt 的内容加入 1.txt，内容以数据流方式保存,该数据流名为 shujuliu.txt.在资源管理器中查看宿主文件 1.txt,发现文件的修改日期和文件大小 都无变化,现在删除 2.txt,执行命令:notepad 1.txt:shujuliu.txt ,即可查看数据流文件中的文件内容了.

NTFS 数据流及高级文件隐藏

一、NTFS 数据流是什么:

在介绍 NTFS 数据流之前，我们先简单了解一下 NTFS 文件系统。NTFS 是微软 Windows NT 内核的系列操作系统支持的、一个特别为网络和磁盘配额、文件加密等管理安全特性设计的磁盘格式。NTFS 比 FAT 文件系统更稳定，更安全，功能也更为强大。如果要让 FAT 文件系统转换为 NTFS 文件系统，可以在“命令提示符”中输入“convert 分区盘符: /fs:ntfs”，即可将该分区的文件系统转换为 NTFS。

NTFS 交换数据流（alternate data streams，简称 ADS）是 NTFS 磁盘格式的一个特性，在 NTFS 文件系统下，每个文件都可以存在多个数据流，就是说除了主文件流之外还可以有许多非主文件流寄宿在主文件流中。它使用资源派生来维持与文件相关的信息，虽然我们无法看到数据流文件，但是它却是真实存在于我们的系统中的。创建一个数据交换流文件的方法很简单，命令为“宿主文件：准备与宿主文件关联的数据流文件”。

NTFS 数据流的创建和打开方式:

1、创建一个 NTFS 数据流

e:\type file.exe>>1.jpg:a.exe

以 1.jpg 文件为宿主文件，将 file.exe 文件寄宿在 1.jpg 文件中，创建了一个 ntfs 数据流，名称为 1.jpg:a.exe，

这样就讲 file.exe 文件以 ntfs 数据流的形式寄宿在了 1.jpg 文件中，

并且形成了一个流文件 1.jpg:a.exe，只要执行这个流文件就可以打开隐藏在 1.jpg 中的 file.exe 文件了。

2、打开一个 NTFS 数据流

start e:\1.jpg:a.exe

打开 ntfs 数据流文件 1.jpg:a.exe，这里 start 参数后面必须跟绝对路径，且只能在 cmdshall 中打开，双标双击无效。

然而在微软为了安全性，在 windows-xp 以后的操作系统（从 win7 开始）就不允许直接使用 start 打开 ntfs 数据流文件，而必须使用 mklink 命令来为流文件建立一个链接，然后通过这个链接文件来打开流文件，以上面的为例：直接使用 start e:\1.jpg:a.exe 是无法打开这个流文件的，提示权限不够，所以我们必须这样做:

1、首先为流文件建立一个链接

mklink e:\b.exe e:\1.jpg:a.exe

将流文件 1.jpg:a.exe 链接到快捷方式 b.exe 上

2、使用 start 打开这个流文件的快捷方式

start e:\b.exe

这样就可以打开 1.jpg:a.exe 这个流文件了

二、高级文件隐藏

公众号Lemonsec

将文件藏入另一个文件内，举例将文件藏入 123.jpg 文件

1、windows 下隐藏：

copy /b 123.jpg +123.rar 1234.jpg

将 123.rar 隐藏到 123.jpg 中，生成的新文件为 1234.jpg，修改后缀名还原，注意图片文件和压缩文件的顺序不能错，图片在前，否则图片不能正确显示。

2、linux 下隐藏：

cat 123.zip >> 123.jpg

将 123.zip 文件隐藏到 123.jpg 中，重定向到 123.jpg 中，修改后缀名还原

三、windows 下创建系统隐藏文件夹

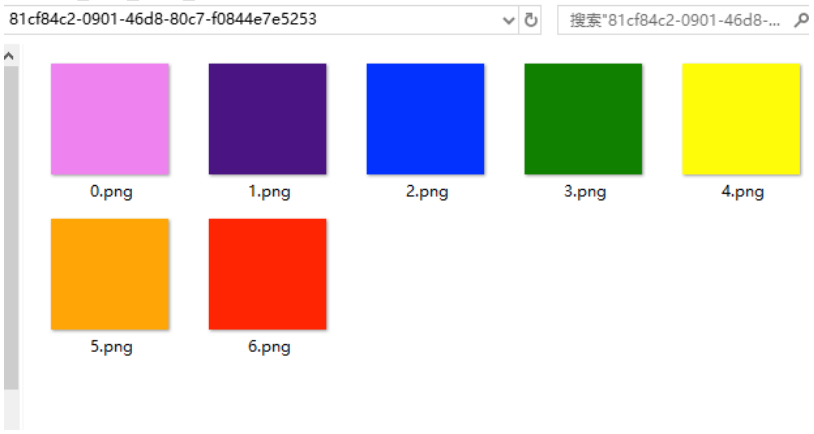
1、windows 下创建隐藏文件夹

attrib +a +s +h 文件夹名称，创建一个系统隐藏的文件夹

杂项第三十七题: color

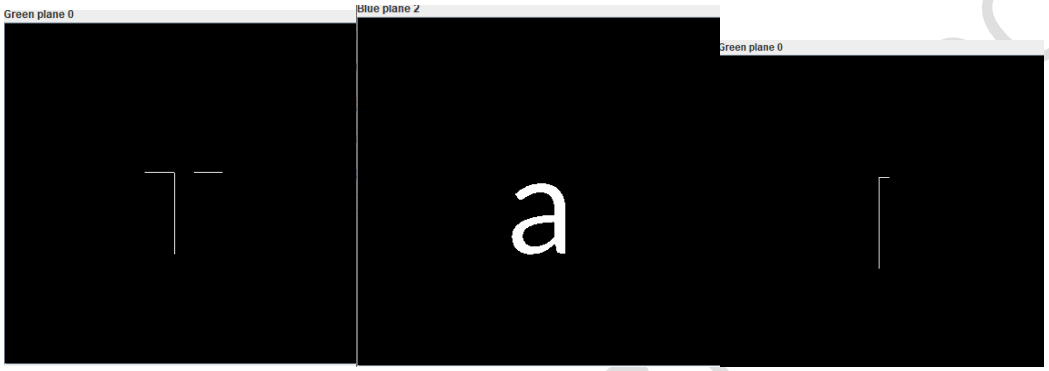
下载压缩包： https://ctf.bugku.com/files/b5f4b306f5cb020e91308895cf5ad767/81cf84c2-0901-46d8-80c7-f0844e7e5253.zip

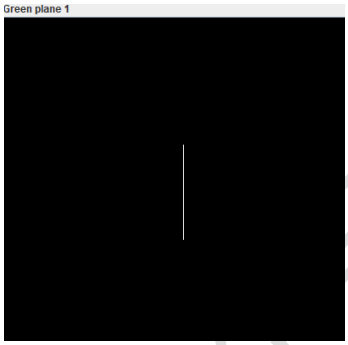
下载解压后 发下是 0-6 一共七张图片



查看属性没什么发现

那么我们在 StegSolve 中依次查看 发现!!!!!!!



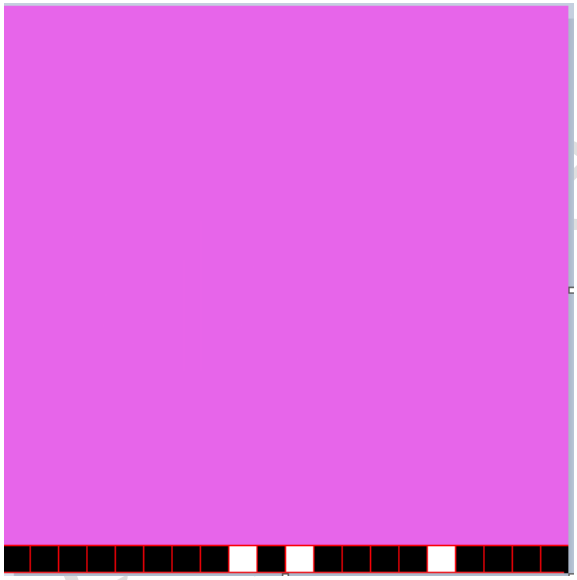


Make me tall
让我变高
发现提交不了　那么　意思是让我变高　那么我们就在 十六进制编译器里面把他们都变高看看
IHDR 后八个字节

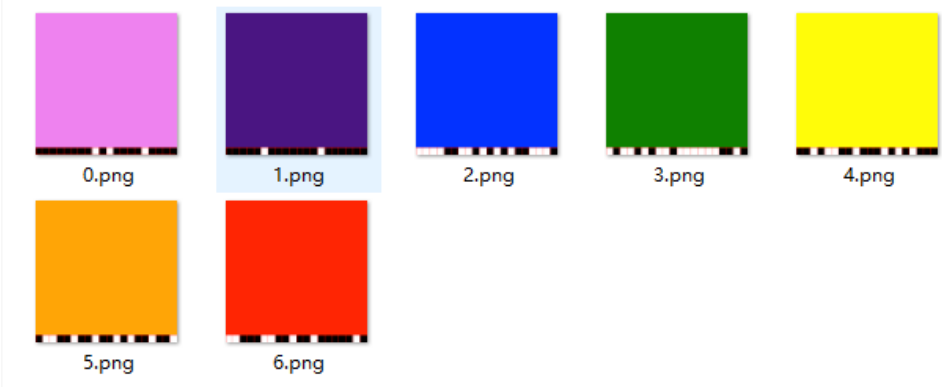
0.png ×

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 ; 塒NG........IHDR
00000010h: 00 00 01 90 00 00 01 7C 08 06 00 00 00 D0 D1 F7 ; ...?..|.....醌?
00000020h: A8 00 00 00 04 67 41 4D 41 00 00 D8 EB F5 1C 14 ; ?...gAMA..仉?.
00000030h: AA 00 00 00 20 63 48 52 4D 00 00 87 0F 00 00 8C ; ?.. cHRM..?..?
00000040h: 0D 00 00 F9 93 00 00 84 E5 00 00 7B 82 00 00 EB ; ...鵠..勫..{?.?
00000050h: 75 00 00 3F B4 00 00 22 58 75 6B 5E 9C 00 00 04 ; u..??."Xuk^?..
00000060h: 18 69 43 43 50 6B 43 47 43 6F 6C 6F 72 53 70 61 ; .iCCPkCGColorSpa
00000070h: 63 65 47 65 6E 65 72 69 63 52 47 42 00 00 48 C7 ; ceGenericRGB..H?
00000080h: 8D 55 5D 68 1C 55 14 3E BB 73 67 23 24 CE 53 6C ; 峉]h.U.>敉g#$鋻l
00000090h: 34 85 74 A8 3F 0D 25 0D 9B 56 34 A1 B4 BA 7F DD ; 4鄱?.%.沄4〈??
000000a0h: DD 36 6E 96 49 36 DA 22 E8 64 F6 EE CE 98 C9 CC ; ?n胱6?鑤鲆蝫商
000000b0h: 38 33 BB FD A1 4F 45 50 7C 31 EA 9B 14 C4 BF B7 ; 83积  EP|1陰.目?
000000c0h: 80 20 28 F5 0F DB 3E B4 2F 95 0A 25 DA D4 20 28 ; € (????%谊 (
000000d0h: 3E B4 F8 83 50 E8 8B A6 EB 99 3B 33 99 69 BA B1 ; >带慟鑾〓?3櫛罕
000000e0h: DE 65 EE 7C F3 9D EF 9E 7B EE B9 67 EF 05 E8 B9 ; 辟顥龚餅{罟g?辎
000000f0h: 2A 5B 96 9E 14 01 16 0C D7 96 8A 59 F1 B9 C3 47 ; *[枯....讖奩宻肓
00000100h: C4 9E 15 48 C2 43 D0 0B 83 D0 2B 2B 8E 95 A9 56 ; 臍.H腋?同++帘￣
00000110h: 27 01 9B A7 85 BB DA AD EF 21 E1 BD AF EC EA 6E ; '.泠吁诃?峤  阯
00000120h: FF CF D6 5B A7 8E 02 90 B8 0F B1 59 77 94 05 C4 ;   现[  .惛.盱w??
00000130h: C7 00 F8 D3 8A 65 BB 18 5F 3F F2 E3 47 5D CB C3 ; ?  奴?_?蜇G]嗣
00000140h: 5E 0C FD 36 06 88 F8 45 0F 37 7D EC 7A 78 CE C7 ; ^.?.堷E.7}靭x吻
00000150h: AF 31 CD 8C 94 43 7C 1A B1 A0 A8 72 1D F1 12 E2 ; ?蜷擱|.睜,.??
00000160h: 91 B9 18 DF 8C 61 3F 06 D6 FA 8B D4 A0 B6 A6 88 ; 懈.邵a?.助嬉釉
00000170h: 5E 2E AA B6 D9 D0 74 1A 0B F7 1E E6 FF D9 16 F4 ; ^.  傩t..????
00000180h: 56 38 DF 36 7C FA 9C F9 E9 43 F8 1E C6 B5 BF 52 ; V8?|鹊  C?频绑
00000190h: 97 F3 1E 1E 45 BC A4 C8 85 69 C4 8F 20 BE D6 D6 ; 槿..E激薄i膹 局?
000001a0h: 66 2B 01 BE 6D B9 59 09 F1 63 00 C9 ED AD F9 5A ; f+.綽笃.蛊.身梳Z
```

改高后，看图



白和黑，我们把其他图也改了



我们

把白色转为 0 黑色转为 1
得到七串二进制

11111111010111101111

11111011111110111111

00001100101010110001

01001010010000001101

11010011011101010111

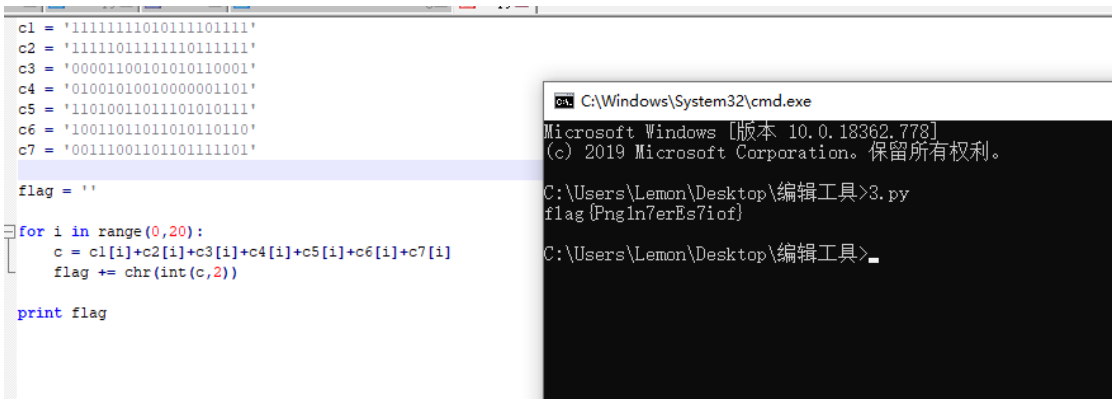
10011011011010110110

00111001101101111101

发现第一列从上到下的二进制码刚好对应 的 ascii 码
1100110 对应 f

```
c1 = '11111111010111101111'
c2 = '11111011111110111111'
c3 = '00001100101010110001'
c4 = '01001010010000001101'
c5 = '11010011011101010111'
c6 = '10011011011010110110'
c7 = '00111001101101111101'

flag = ''

for i in range(0,20):
    c = c1[i]+c2[i]+c3[i]+c4[i]+c5[i]+c6[i]+c7[i]
    flag += chr(int(c,2))

print flag
```

公众号 LemonSec



得到

flag{Png1n7erEs7iof}

杂项第三十八题: 小明的密码

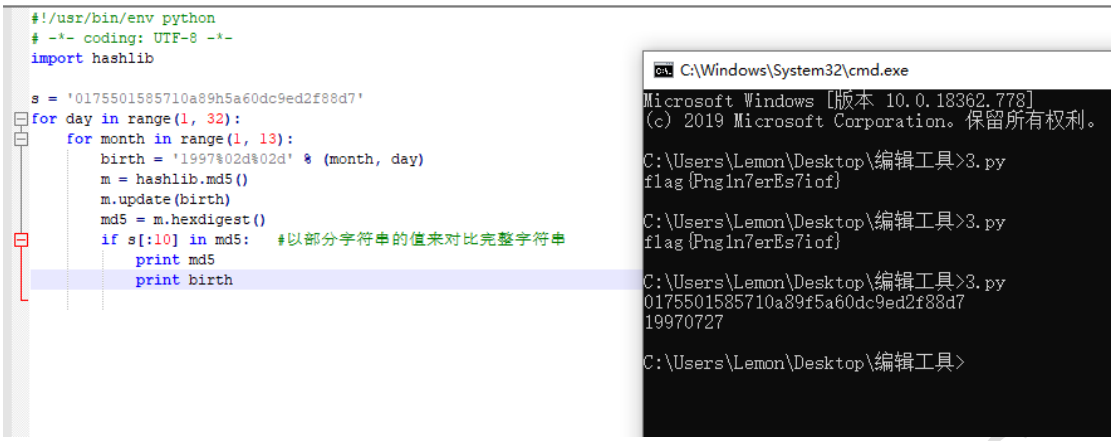
97 年出生的小明用自己的生日作为自己网站的密码，现在，得到一串被篡改过一个字符的字符串，你能解出小明的生日吗?

0175501585710a89h5a60dc9ed2f88d7

md5 加密

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
import hashlib

s = '0175501585710a89h5a60dc9ed2f88d7'
for day in range(1, 32):
    for month in range(1, 13):
        birth = '1997%02d%02d' % (month, day)
        m = hashlib.md5()
        m.update(birth)
        md5 = m.hexdigest()
        if s[:10] in md5:   #以部分字符串的值来对比完整字符串
            print md5
            print birth
```

公众号LemonSec



杂项第三十九题: 仿射加密

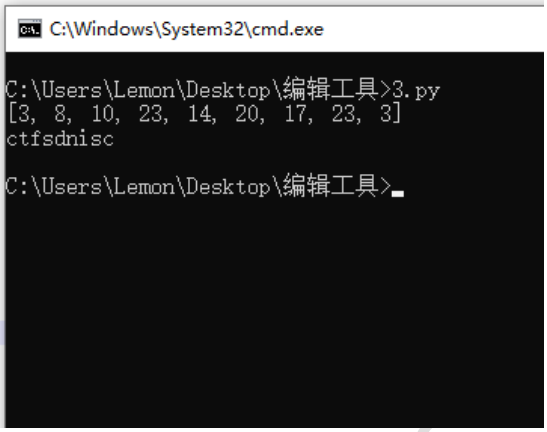
已知仿射加密变换为 c=（11m+7）mod26，试对密文 dikxourxd 解密

一、脚本解密

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
a = 'dikxourxd'
a1=[]
for i in a:
    a2 = ord(i)-97
    a1.append(a2)
print a1
s = ""
for i in a1:
    for j in range(0,26):
        c = (11*j+7)%26
        if(c==i):
            s += chr(j+97)
print s
```

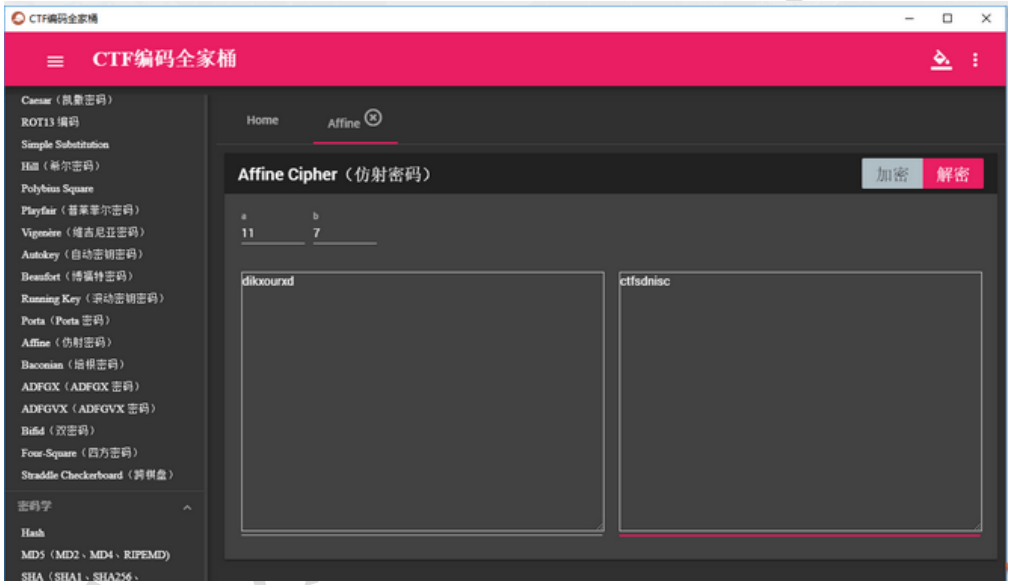
公众号LemonSec

```python
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
a = 'dikxourxd'
al=[]
for i in a:
    a2 = ord(i)-97
    al.append(a2)
print al
s = ""
for i in al:
    for j in range(0,26):
        c = (11*j+7)%26
        if(c==i):
            s += chr(j+97)
print s
```



二、网站解密

工具解密



仿射密码解析与实例

仿射密码
原理

仿射密码的加密函数是 $E(x)=(ax+b)(modm)$

，其中

- x 表示明文按照某种编码得到的数字

- a 和 m 互质
- m 是编码系统中字母的数目。

解密函数是 D(x)=a-1(x-b)(modm)

，其中 a-1 是 a 在 Zm

群的乘法逆元。

下面我们以 E(x)=(5x+8)mod26

函数为例子进行介绍，加密字符串为 AFFINE CIPHER，这里我们直接采用字母表 26 个字母作为编码系统

明文	A	F	F	I	N	E	C	I	P	H	E	R
x	0	5	5	8	13	4	2	8	15	7	4	17
$y = 5x + 8$	8	33	33	48	73	28	18	48	83	43	28	93
$y \mod 26$	8	7	7	22	21	2	18	22	5	17	2	15
密文	I	H	H	W	V	C	S	W	F	R	C	P

https://blog.csdn.net/qq_41725312

其对应的加密结果是 IHHWVCSWFRCP。

对于解密过程，正常解密者具有 a 与 b，可以计算得到 a-1 为 21，

所以其解密函数是 D(x)=21(x-8)(mod26)

，解密如下

密文	I	H	H	W	V	C	S	W	F	R	C	P
y	8	7	7	22	21	2	18	22	5	17	2	15
$x = 21(y - 8)$	0	-21	-21	294	273	-126	210	294	-63	189	-126	147
$x \mod 26$	0	5	5	8	13	4	2	8	15	7	4	17
明文	A	F	F	I	N	E	C	I	P	H	E	R

https://blog.csdn.net/qq_41725312

可以看出其特点在于只有 26 个英文字母。

破解

首先，我们可以看到的是，仿射密码对于任意两个不同的字母，其最后得到的密文必然不一样，所以其也具有最通用的特点。当密文长度足够长时，我们可以使用频率分析的方法来解决。

其次，我们可以考虑如何攻击该密码。可以看出当 a=1

时，仿射加密是凯撒加密。而一般来说，我们利用仿射密码时，其字符集都用的是字母表，一般只有 26 个字母，而不大于 26 的与 26 互素的个数一共有

$\phi(26)=\phi(2)\times\phi(13)=12$

算上 b 的偏移可能，一共有可能的密钥空间大小也就是

$12\times26=312$

一般来说，对于该种密码，我们至少得是在已知部分明文的情况下才可以攻击。下面进行简单的分析。

这种密码由两种参数来控制，如果我们知道其中任意一个参数，那我们便可以很容易地快速枚举另外一个参数得到答案。

但是，假设我们已经知道采用的字母集，这里假设为 26 个字母，我们还有另外一种解密方式，我们只需要知道两个加密后的字母 y1,y2

即可进行解密。那么我们还可以知道

$y1=(ax1+b)(mod26)\ y2=(ax2+b)(mod26)$

两式相减，可得

$y1-y2=a(x1-x2)(mod26)$

这里 y1,y2

已知，如果我们知道密文对应的两个不一样的字符 x1 与 x2 ，那么我们就可以很容易得到 a ，进而就可以得到 b 了。

例子

这里我们以 TWCTF 2016 的 super_express 为例进行介绍。简单看一下给的源码

```
import sys
key = '****CENSORED***************'
flag = 'TWCTF{*******CENSORED********}'
```

公众号Lemonsec

```
if len(key) % 2 == 1:
    print("Key Length Error")
    sys.exit(1)

n = len(key) / 2
encrypted = ''
for c in flag:
    c = ord(c)
    for a, b in zip(key[0:n], key[n:2*n]):
        c = (ord(a) * c + ord(b)) % 251
    encrypted += '%02x' % c

print encrypted
```

```
import sys
key = '****CENSORED***************'
flag = 'TWCTF{*******CENSORED********}'

if len(key) % 2 == 1:
    print("Key Length Error")
    sys.exit(1)

n = len(key) / 2
encrypted = ''
for c in flag:
    c = ord(c)
    for a, b in zip(key[0:n], key[n:2*n]):
        c = (ord(a) * c + ord(b)) % 251
    encrypted += '%02x' % c

print encrypted
```

可以发现，虽然对于 flag 中的每个字母都加密了 n 次，如果我们仔细分析的话，我们可以发现

c1=a1c+b1c2=a2c1+b2=a1a2c+a2b1c+b2=kc+d

根据第二行的推导，我们可以得到其实 cn

也是这样的形式，可以看成 cn=xc+y

，并且，我们可以知道的是，key 是始终不变化的，所以说，其实这个就是仿射密码。

此外，题目中还给出了密文以及部分部分密文对应的明文，那么我们就很容易利用已知明文攻击的方法来攻击了，利用代码如

```
import gmpy
```

```
key = '****CENSORED****************'
flag = 'TWCTF{*******CENSORED********}'

f = open('encrypted', 'r')
data = f.read().strip('\n')
encrypted = [int(data[i:i + 2], 16) for i in range(0, len(data), 2)]
plaindelta = ord(flag[1]) - ord(flag[0])
cipherdalte = encrypted[1] - encrypted[0]
a = gmpy.invert(plaindelta, 251) * cipherdalte % 251
b = (encrypted[0] - a * ord(flag[0])) % 251
a_inv = gmpy.invert(a, 251)
result = ""
for c in encrypted:
    result += chr((c - b) * a_inv % 251)
print result
```

```
import gmpy

key = '****CENSORED****************'
flag = 'TWCTF{*******CENSORED********}'

f = open('encrypted', 'r')
data = f.read().strip('\n')
encrypted = [int(data[i:i + 2], 16) for i in range(0, len(data), 2)]
plaindelta = ord(flag[1]) - ord(flag[0])
cipherdalte = encrypted[1] - encrypted[0]
a = gmpy.invert(plaindelta, 251) * cipherdalte % 251
b = (encrypted[0] - a * ord(flag[0])) % 251
a_inv = gmpy.invert(a, 251)
result = ""
for c in encrypted:
    result += chr((c - b) * a_inv % 251)
print result
```

结果如下:

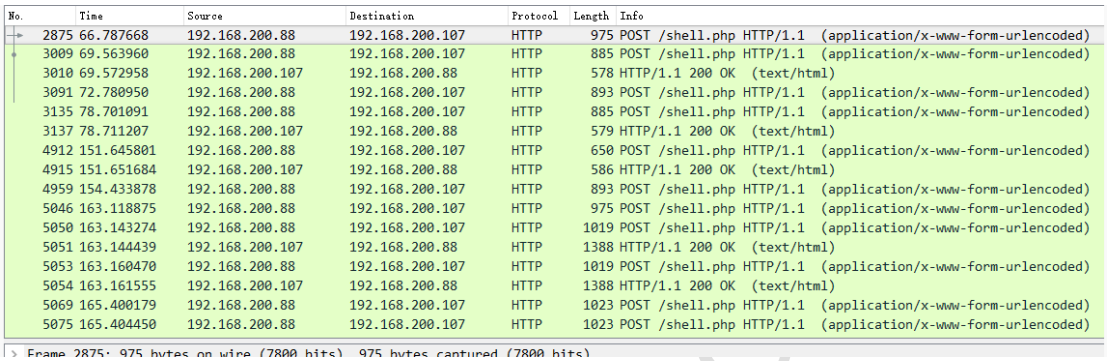
```
➜  TWCTF2016-super_express git:(master) ✗ python exploit.py

TWCTF{Faster_Than_Shinkansen!}
```

杂项第四十题：黑客的机密信息

黑客通过 webshell 往 Web 服务器写入了一串机密信息，你能找出机密信息吗？

首先分析流量包，过滤 http 请求，在数据包末端发现 shell.php，利用 Wireshark 的 filter 过滤出带有 shell.php 的流量，http contains "shell.php"

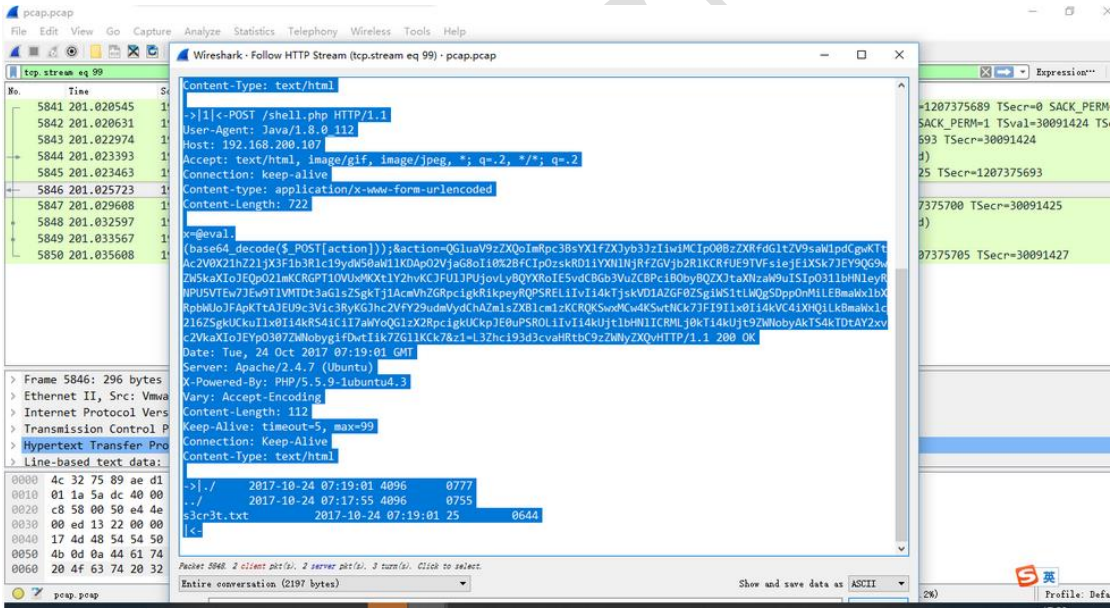
利用一句话执行的命令都经过 base64 编码，不能直接搜索到 flag 关键词。

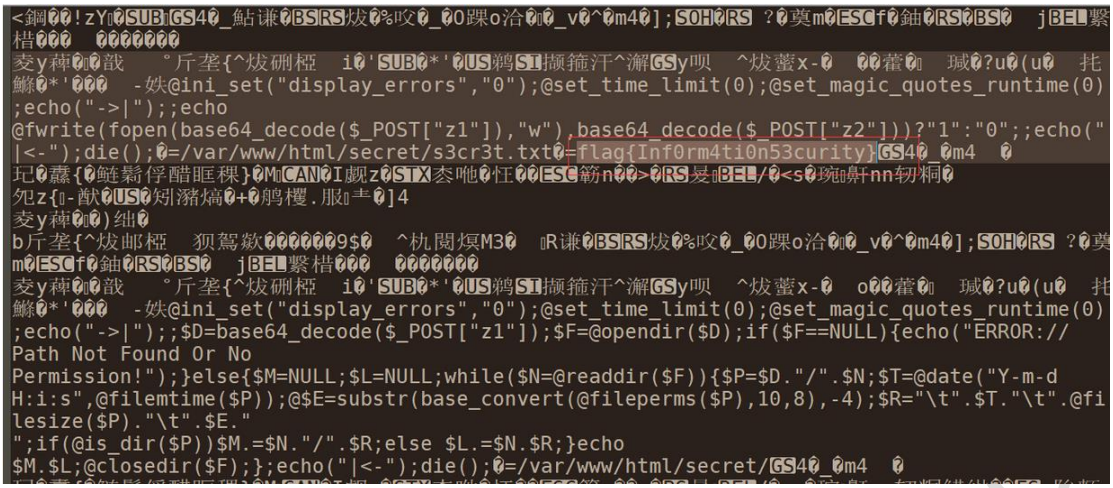


No.	Time	Source	Destination	Protocol	Length	Info
2875	66.787668	192.168.200.88	192.168.200.107	HTTP	975	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
3009	69.563960	192.168.200.88	192.168.200.107	HTTP	885	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
3010	69.572958	192.168.200.107	192.168.200.88	HTTP	578	HTTP/1.1 200 OK (text/html)
3091	72.780950	192.168.200.88	192.168.200.107	HTTP	893	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
3135	78.701091	192.168.200.88	192.168.200.107	HTTP	885	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
3137	78.711207	192.168.200.107	192.168.200.88	HTTP	579	HTTP/1.1 200 OK (text/html)
4912	151.645801	192.168.200.88	192.168.200.107	HTTP	650	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
4915	151.651684	192.168.200.107	192.168.200.88	HTTP	586	HTTP/1.1 200 OK (text/html)
4959	154.433878	192.168.200.88	192.168.200.107	HTTP	893	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
5046	163.118875	192.168.200.88	192.168.200.107	HTTP	975	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
5050	163.143274	192.168.200.88	192.168.200.107	HTTP	1019	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
5051	163.144439	192.168.200.107	192.168.200.88	HTTP	1388	HTTP/1.1 200 OK (text/html)
5053	163.160470	192.168.200.88	192.168.200.107	HTTP	1019	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
5054	163.161555	192.168.200.107	192.168.200.88	HTTP	1388	HTTP/1.1 200 OK (text/html)
5069	165.400179	192.168.200.88	192.168.200.107	HTTP	1023	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)
5075	165.404450	192.168.200.88	192.168.200.107	HTTP	1023	POST /shell.php HTTP/1.1 (application/x-www-form-urlencoded)

> Frame 2875: 975 bytes on wire (7800 bits), 975 bytes captured (7800 bits)

通过跟踪流来查找 flag





公众号LemonSec