攻防世界misc进阶

原创

舞动的獾 于 2019-05-22 21:12:34 发布 7025 收藏 6

分类专栏： 网络安全misc 文章标签： 攻防世界misc 进阶 wp

本文链接：https://blog.csdn.net/Yu_csdnstory/article/details/90452291

004眼见非实

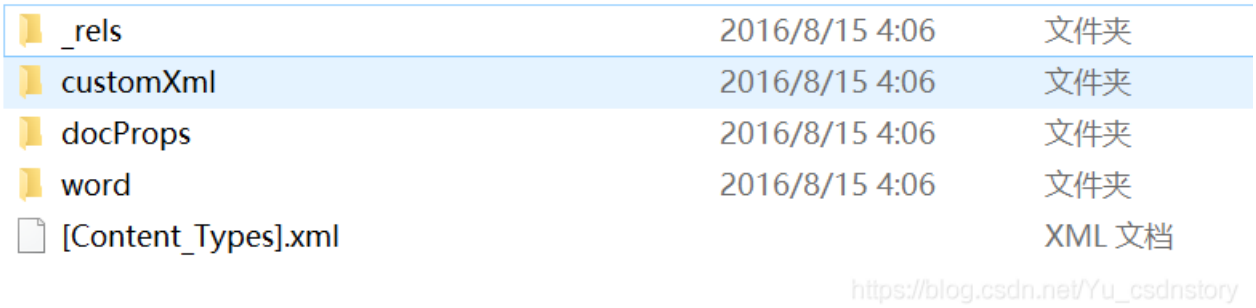
下载附件

打开后解压，发现了.docx文件

我尝试了改后缀，转格式，编码都失败了，最后在winhex打开，查看文件头

50 4B 03 04

发现这是一个zip文件，改后缀为.zip 直接解压，拿到了一堆.xml文件 **猜测在最外面的文件里**



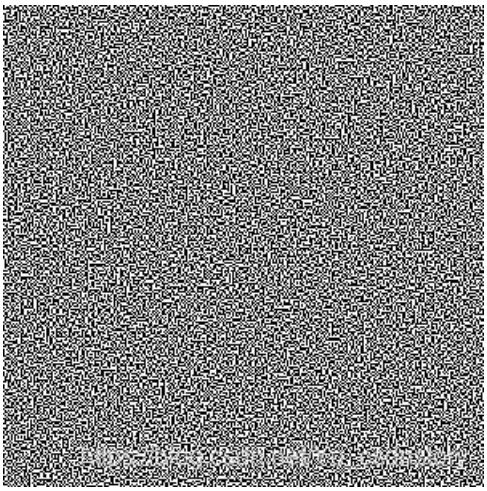
但是并没有flag

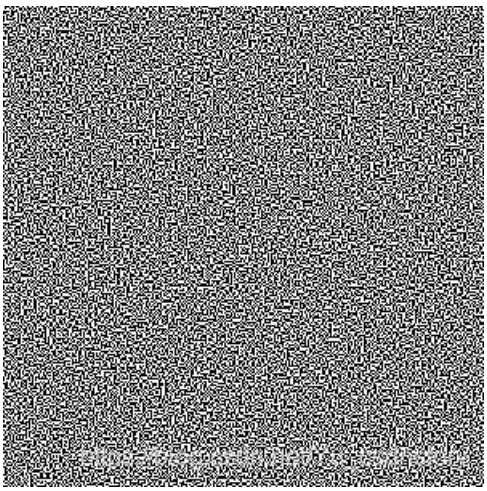
还有word文件夹，首先打开，找到document.xml

在sublime下打开，find flag

What-is-this

下载附件，发现是个压缩包文件，改名为.zip压缩打开
发现还是压缩文件，一路解压，最终发现两张图片





对两张图片进行对比，用Stegsolve.jar

得到了一张图片

图像过于清晰，导致大写的i，看成了小写的l,提交了n次，这里给出flag，留出时间做其他

```
AZADI TOWER
```

** 千万不要加上任何前缀**

Training-Stegano-1

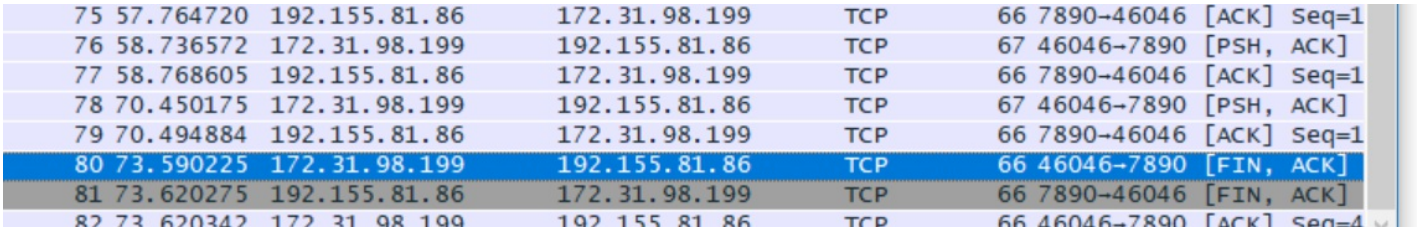
图片用winhex打开，直接得到flag，不要想太多

```
steganoI
```

easycap

用winshark打开文件

ctrl+F查找flag，发现条目



75	57.764720	192.155.81.86	172.31.98.199	TCP	66 7890→46046 [ACK] Seq=1
76	58.736572	172.31.98.199	192.155.81.86	TCP	67 46046→7890 [PSH, ACK]
77	58.768605	192.155.81.86	172.31.98.199	TCP	66 7890→46046 [ACK] Seq=1
78	70.450175	172.31.98.199	192.155.81.86	TCP	67 46046→7890 [PSH, ACK]
79	70.494884	192.155.81.86	172.31.98.199	TCP	66 7890→46046 [ACK] Seq=1
80	73.590225	172.31.98.199	192.155.81.86	TCP	66 46046→7890 [FIN, ACK]
81	73.620275	192.155.81.86	172.31.98.199	TCP	66 7890→46046 [FIN, ACK]

在第81条，右键tcp跟踪，得到flag

```
FLAG:385b87afc8671dee07550290d16a8071
```

Test-flag-please-ignore

下载附件发现是个zip,并没有伪加密，直接解压，并将打开的文件重命名为.txt
打开后发现了一串字符，英文最大不超过F

用进制转换工具，将16进制转为字符串，得到flag

```
flag{hello_world}
```

4-2

难度系数：
题目来源： WDCTF-2017

垃圾wp，根本无法得到，做出来的大佬请留言
WDCTF-2017:4-2
【原理】

字频分析
【目的】

简单了解字频分析
【环境】

windows,linux
【工具】

无
【步骤】

字频分析得到flag，

flag{classical-cipher_is_not_security_hs}

【总结】

无

glance-50

这个GIF皮的很啊，用了好多方法分离，最终还是败在了convert这个工具里了
在kali用分离命令将图片分离

```
convert glance.gif flag.png
```

得到了一大堆的.png图片，足足200多个
然后横向合成，用montage，

```
montage flag*.png -tile x1 -geometry +0+0 a.png
```

-tile是拼接时每行和每列的图片数，这里用x1，就是只一行

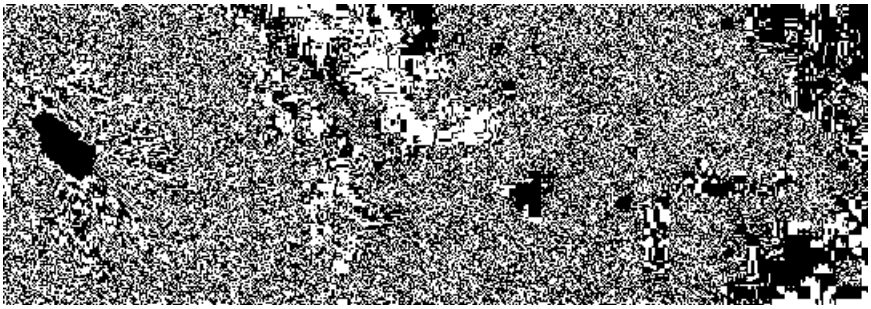
-geometry是首选每个图和边框尺寸，我们边框为0，图照原始尺寸即可
*的意思指的所有的.png
这里参考博客原创主
最后得到拼接好的图片

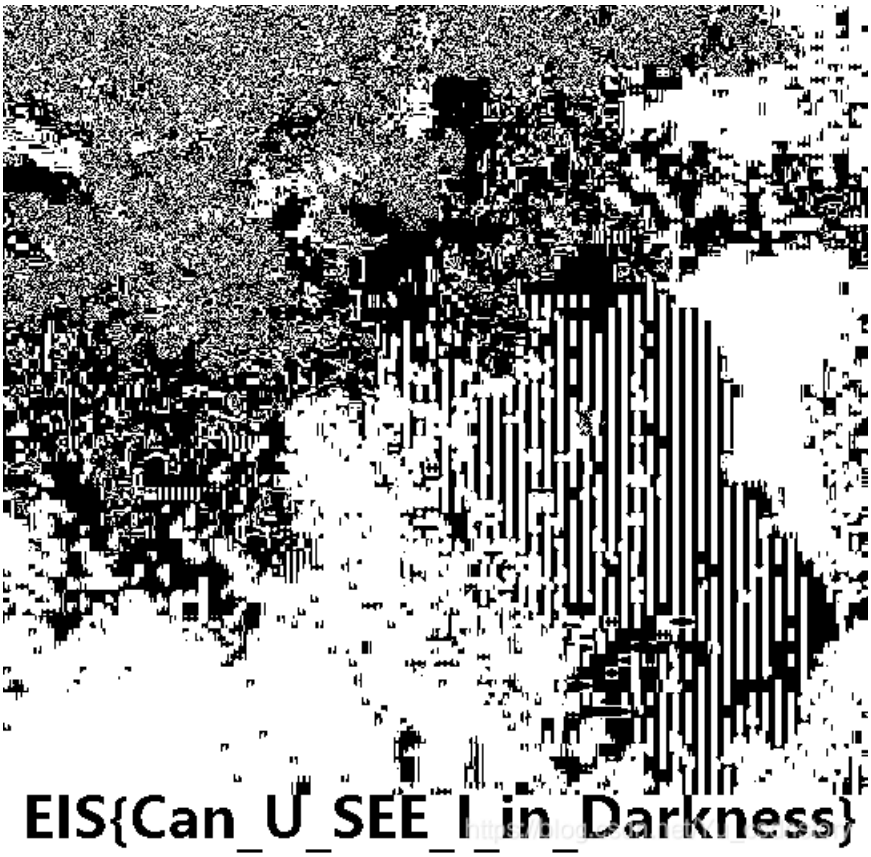
得到flag

```
TWCTF{Bliss by Charles O'Rear}
```

隐藏在黑夜里的秘密

下载附件发现是个压缩包，解压需要密码，首先想到的就是伪加密。
在winrar 修复之后，发现并不能将它解压出来，只好使用杀手锏，对它进行分析
在winhex下打开，看到16进制的编码，zip的开头是50 4B 03 04
50 4B 01 02：目录中文件文件头标记
3F 00：压缩使用的 pkware
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密，这个更改这里进行伪加密，改为09 00打开就会提示有密码了）
压缩源文件目录结束标志 ：50
4B 05 06：目录结束标记
在winhex里面搜索，50 4B
将全局方式位标记全部改为00 00
发现压缩包能够正常的打开
发现了一张图片和一个flag.txt
重点在图片上，进行stegsolve打开，得到flag

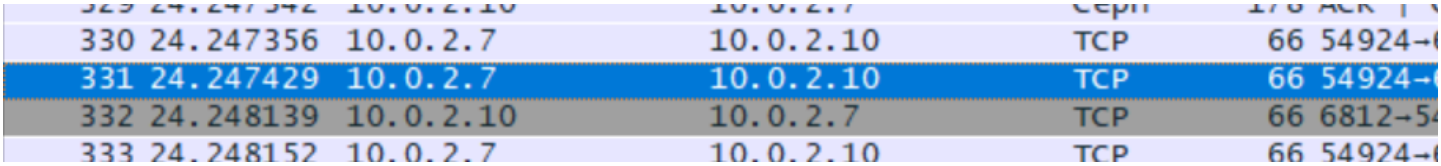




Cephalopod

打开附件后发现这是一个pcap文件，将它用winshark打开

搜索flag,发现了flag.png，几乎每个条目里都有，但是并不能分离出来



最终还是向writeup低了头，发现了一款新的工具

tcpxtract

在linux环境下，用binwalk对图片进行分析：

```
26441 0x6749 PNG image, 1754 x 2480, 8-bit/color RGBA, non-interlaced 26577 0x67D1 Zlib compressed data, best compression ``` 可以看到这里存在一个图片
```

用dd if 文件名 of 1.png skip=26441

分离出的图片不能被打开，才发现图片需要恢复，用简单的提取是不行的

Tcpxtract是一种基于文件签名从网络流量中提取文件的工具。基于文件类型的页眉和页脚（有时称为“雕刻”）提取文件是一种古老的数据恢复技术。Foremost这样的工具使用这种技术可以从任意数据流中恢复文件，其是专门用于通过网络传输的拦截文件的应用。填补类似需求的其他工具有流网和EtherPEG。driftnet和EtherPEG是用于在网络上监控和提取图形文件的工具，网络管理员通常使用它来警告用户的互联网活动。driftnet和EtherPEG的主要局限性在于它们只支持三种文件类型，不需要添加更多方法。他们使用的搜索技术也是不可扩展的，不会跨数据包边界搜索

由于它没有kali版本，只好安装在redhat里rpm安装包地址如下：

下载地址

分离命令：

```
tcpxtract -f 40150e85ac1b4952f1c35c2d9103d8a40c7bee55.pcap Found file of type "png" in session
```

分离出两张图片，还是不能查看，但是放入winhex发现，它的头部少了89

加上后保存，其中一张可以打开，出现了flag

2-1

难度系数：

题目来源： WDCTF-finals-2017

本题给出一张图片，但是并无法显示出内容，放到hex里面，发现它的数据头是错的

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
80 59 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  €YNG........IHDR
00 00 02 C5 00 00 02 F8 08 06 00 00 00 93 2F 8A  ...Å...ø......“/Š
6B 00 00 00 04 67 41 4D 41 00 00 9C 40 20 0D E4  k....gAMA..œ@ .ä
CB 00 00 00 20 63 48 52 4D 00 00 87 0F 00 00 8C  Ë... cHRM..‡.....
0F 00 00 FD 52 00 00 81 40 00 00 7D 79 00 00 E9  ...ýR...@..}y..é
8B 00 00 3C E5 00 00 19 CC 73 3C 85 77 00 00 0A  ‹..<å...Ìs<…w....
39 69 43 43 50 50 68 6F 74 6F 73 68 6F 70 20 49  9iCCPPhotoshop I
```

将它改为89 50，发现并没有显示出图片的内容。

试试放到tweakpng 工具中，发现它的检验值是错的，显示为0x932f8a6b

所以需要将它的校验值为0x932f8a6b,查看下，宽度是0，而高度760，所以这道题就需要改宽度

脚本如下：

```
import os
import binascii
import struct
misc = open("misc4.png","rb").read()
for i in range(1024):
 data = misc[12:16] + struct.pack('>i',i)+ misc[20:29]
 crc32 = binascii.crc32(data) & 0xffffffff
 if crc32 == 0x932f8a6b:
    print (i)
```

大致解释一下脚本：
爆破crc校验所需要了解到的PNG文件头知识

```
- （固定）八个字节89 50 4E 47 0D 0A 1A 0A为png的文件头
- （固定）四个字节00 00 00 0D（即为十进制的13）代表数据块的长度为13
- （固定）四个字节49 48 44 52（即为ASCII码的IHDR）是文件头数据块的标示（IDCH）
- （可变）13位数据块（IHDR)
    - 前四个字节代表该图片的宽
    - 后四个字节代表该图片的高
    - 后五个字节依次为：
    Bit depth、ColorType、Compression method、Filter method、Interlace method
- （可变）剩余四字节为该png的CRC检验码，由从IDCH到IHDR的十七位字节进行crc计算得到。
```

参考链接

binascii

bianscii库是一个进制转换库，可以实现二进制与ASCII的转换，将图片以二进制打开，存到misc变量中，
用struct.pack对i进行转换，转换为一层包装的python大端整型字节，用binascii.crc32得到校验值，与0xffffffff做与运算，得到16进制，与正确的0x932f8a6b比较，爆破出i,修改宽度，得到flag.

flag is wdflag{Png_

C2c_u_kn0W}

小小的pdf

下载附件，发现pdf文件，习惯的用linux的命令查看pdf的内容，所以命令：

```
pdfinfo 2333.pdf
```

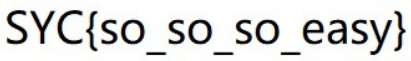
结果并没有什么东西，对它进行火狐的命令行
在控制台输入：

```
document.documentElement.textContent
```

查看信息，并没有出现隐藏flagd的内容
binwalk分离一下，发现了三张图片，有戏

```
root@kali:~/Desktop# binwalk 2333.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PDF document, version: "1.4"
452           0x1C4           JPEG image data, JFIF standard 1.01
73254         0x11E26         JPEG image data, JFIF standard 1.01
81606         0x13EC6         Zlib compressed data, default compression
82150         0x140E6         JPEG image data, JFIF standard 1.01
104469        0x19815         Zlib compressed data, default compression
105134        0x19AAE         Zlib compressed data, default compression
```


**用foremost 分离，得到隐函flag的图片 **



2017_Dating_in_Singapore

【原理】

日历中的日期隐藏flag
【目的】

了解日历中隐藏flag的方法
【环境】

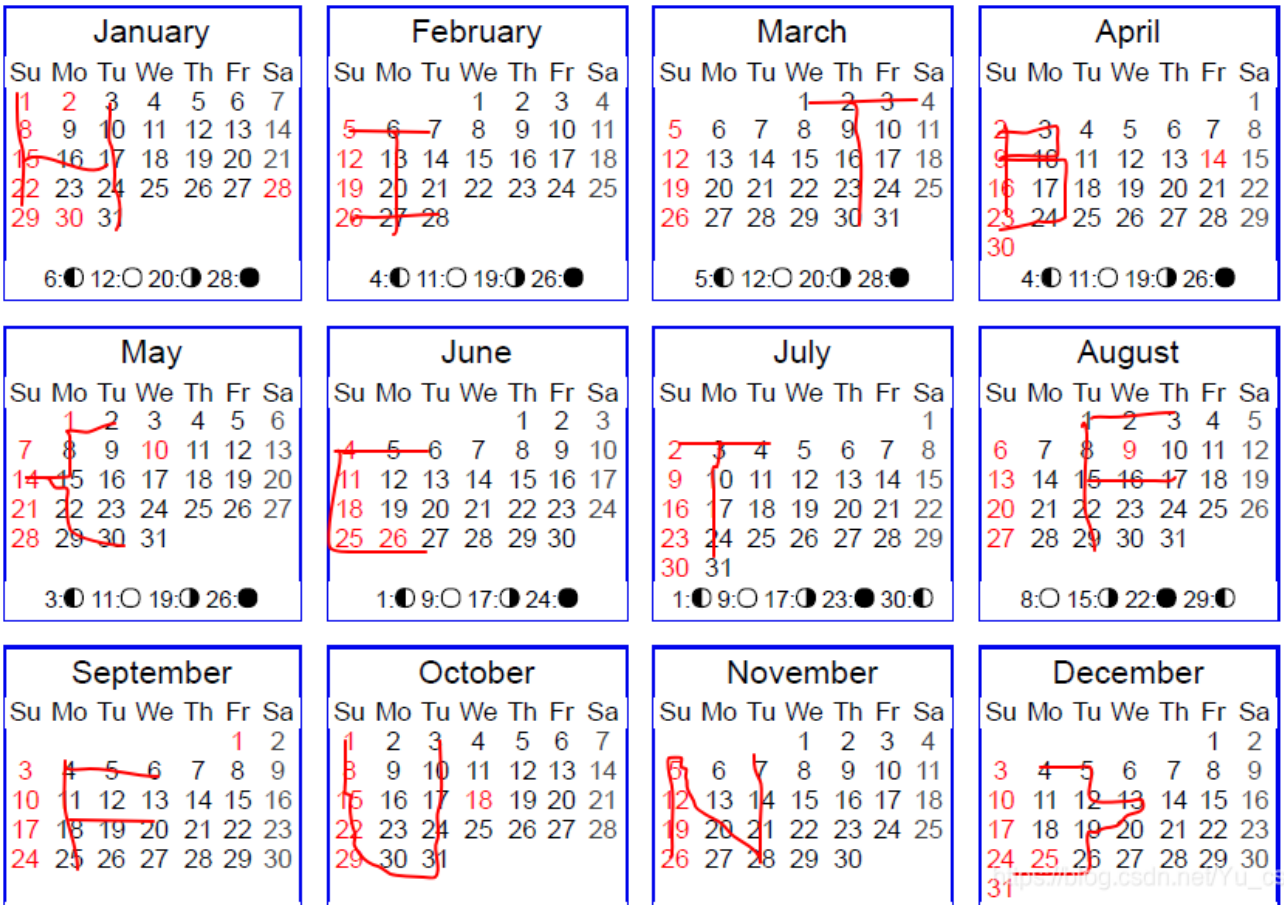
linux
【工具】

2017年新加坡日历
【步骤】

通过题目描述可以发现按-分割是12组，然后数字似乎都是两位的，于是按两位分割之后发现都是0-31范围内的，于是联想到月份，找出一份2017新加坡日历：

```
010815222915161703101724 31-050607132027162728-0102030209162330-020916230203100909101724 23-02010814222930-0605041
118252627-0203040310172431-0102030108152229151617-0405060411182518 1920-0108152229303124171003-261912052028211407
-04051213192625
```

将每个数字标记，然后连接得到flag：



4-1

用formost图片分离出压缩包，解压后为两张图片和一个tip.txt

看似一样，用stegsolve的拼图功能，对比不出结果，查了查，发现了盲水印的存在

gitHub地址

由于盲水印的脚本运行环境为python2，还得需要安装python2的库，我的过程如下：

```
python2 -m pip install --upgrade pip
```

由于用到cv2这个库，所以安装

```
pip install opencv-python
```

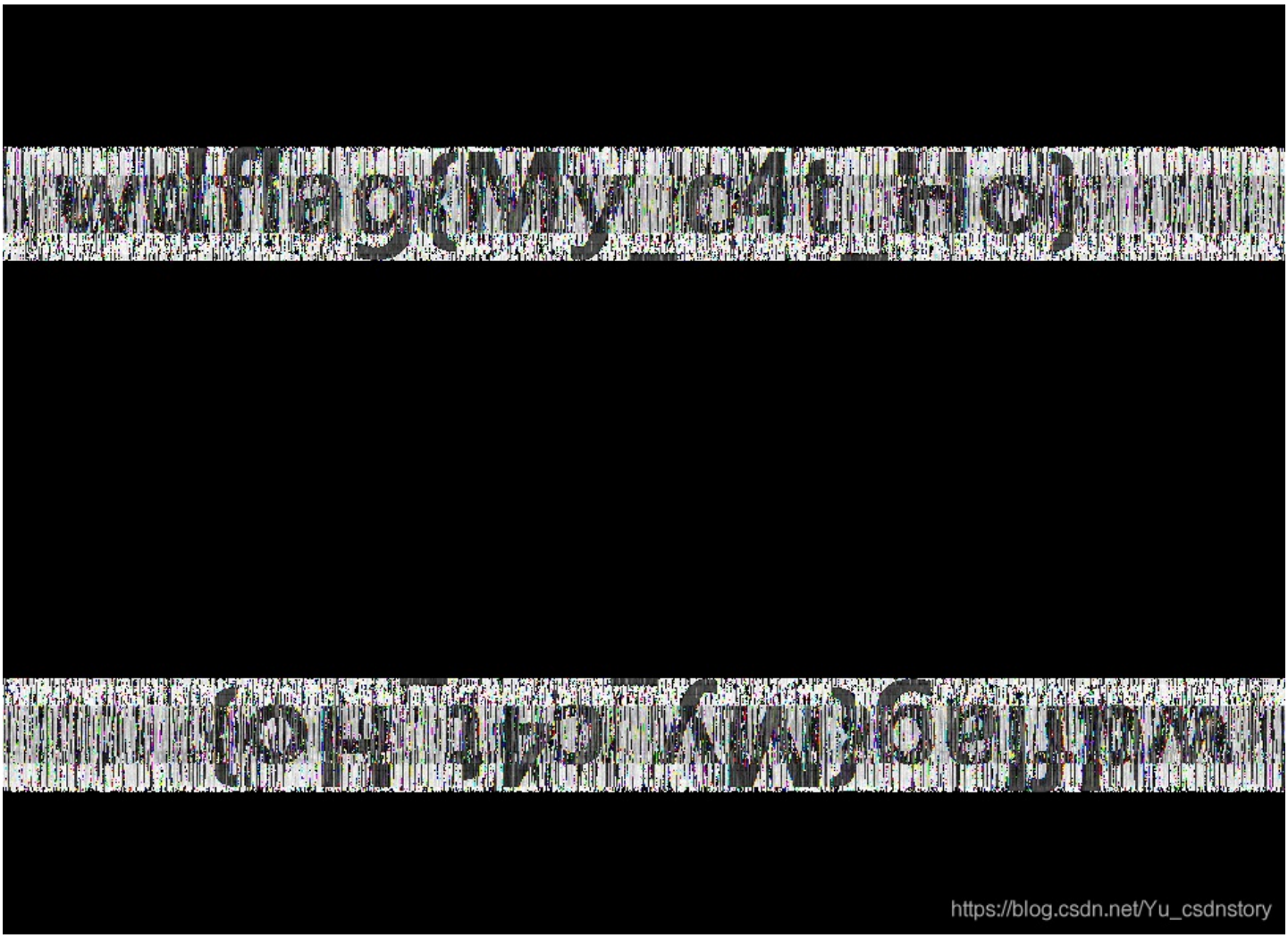
最后安装：

```
pip install matplotlib
```

之后就可以成功运行脚本，执行以下命令

```
python2 bwm.py decode day1.png day2.png flag.png
```

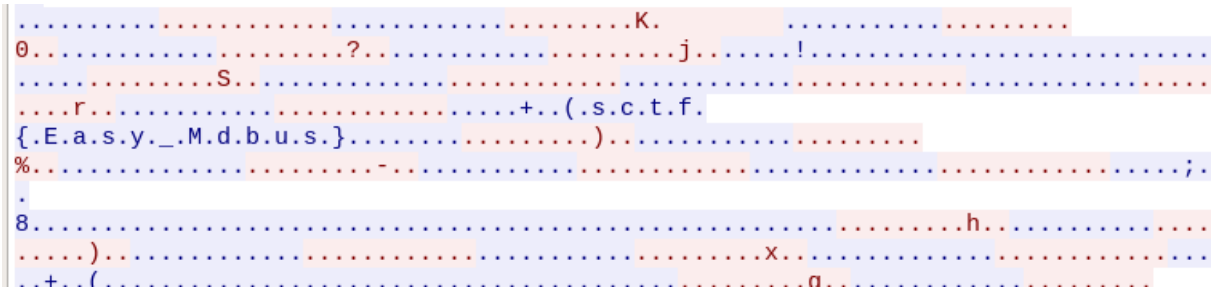
得到flag.png：



神奇的modbus

拿到了流量分析的文件，打开有很多包

找了个tcp流跟踪了一下，发现了flag



巨坑的是，提交不对，最终改为sctf{Easy_Modbus}成功通过

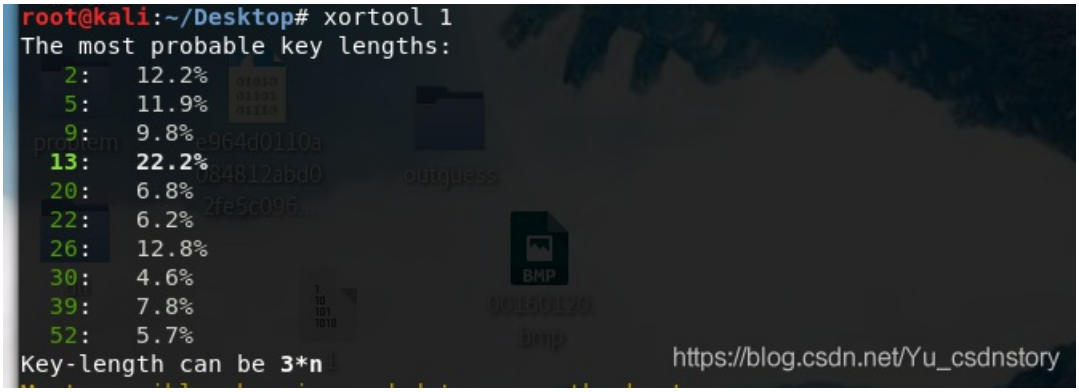
5-1

考察xortool工具的使用

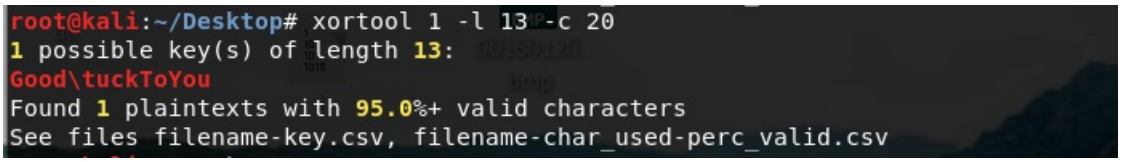
安装

linux下：

```
pip2 install xortool
```



可以看到13机率最大

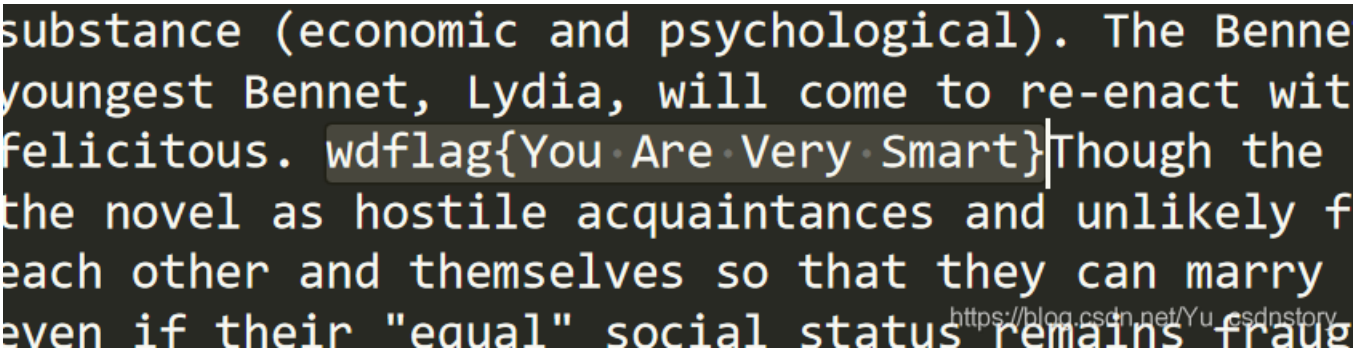


其中-l就是指定密钥长度，-c表示出现频率最高的字符。这个需要根据经验，比如文本内容一般是空格（20），二进制文件一般是00

解密脚本

```
import os

c = open("1",'rb').read()
key = "GoodLuckToYou"
def xor(c,k):
  keylen = len(k)
   res = ""
  for pos,c in enumerate(c):
   res +=chr(ord(c) ^ ord(k[pos % keylen]))
  return res
print xor(c,key)
```

在输出结果下找到flag



MISCall

得到未知名的文件，用binwalk分析下，发现是bz2压缩包，在linux下解压得到一个git仓库的文件夹

```
tar -xjvf bz2文件
```

解压缩命令

```
root@kali:~/Desktop# tar xjvf 2.bz2
ctf/
ctf/flag.txt
ctf/.git/
ctf/.git/description
ctf/.git/refs/
ctf/.git/refs/heads/
ctf/.git/refs/heads/master
ctf/.git/refs/stash
ctf/.git/refs/tags/
ctf/.git/ORIG_HEAD
ctf/.git/logs/
```


进入目录下，找到flag.txt

```
root@kali:~/Desktop# cd ctf
root@kali:~/Desktop/ctf# ls
flag.txt
root@kali:~/Desktop/ctf# cat flag.txt
Nothing to see here, moving along...
```

用git命令查看日志，但是没有找到这个文件

```
root@kali:~/Desktop/ctf# ls -a
.  ..  flag.txt  .git
root@kali:~/Desktop/ctf# cd .git
root@kali:~/Desktop/ctf/.git# ls
branches        config       HEAD   index  logs      ORIG_HEAD
COMMIT_EDITMSG  description  hooks  info   objects   refs
root@kali:~/Desktop/ctf/.git# git log
commit bea99b953bef6cc2f98ab59b10822bc42afe5abc (HEAD -> master)
Author: Linus Torvalds <torvalds@klaava.Helsinki.Fi>
Date:   Thu Jul 24 21:16:59 2014 +0200

    Initial commit
```


首先补充

git stash 的作用

git stash用于想要保存当前的修改,但是想回到之前最后一次提交的干净的工作仓库时进行的操作.git stash将本地的修改保存起来,并且将当前代码切换到HEAD提交上.

通过git stash存储的修改列表,可以通过git stash list查看.git stash show用于校验,git stash apply用于重新存储.直接执行git stash等同于git stash save.

原文链接：https://blog.csdn.net/zz_Caleb/article/details/89331985
https://www.jianshu.com/p/14afc9916dcb

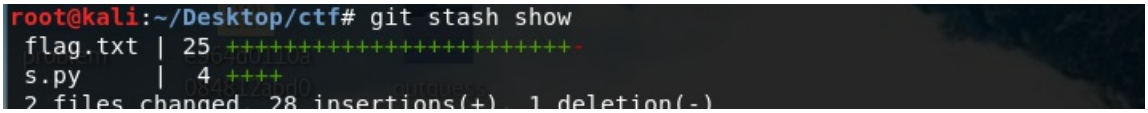
用如下命令查看修改的文件列表

```
git stash list
```

```
root@kali:~/Desktop/ctf# git stash list
stash@{0}: WIP on master: bea99b9 Initial commit
```

有东西，然后校验一下存储的文件列表

```
git stash show
```



发现了s.py,将他们重新储存，运行s.py得到flag

```
git stash apply
```

can_has_stdio?

得到一个用±<>这样符号组成的五角星，结合题目stdio，估计是c语言编译后的文件

查到BrianFuck语言，找个在线编译器或者找到编译码（c++）得到flag

BrainFuck语言

极简的一种图灵完备的语言，由Urban Müller在1993年创造，由八个指令组成（如下表）。工作机制与图灵机非常相似，有一条足够长的纸带，初始时纸带上的每一格都是0，有一个数据读写头指向纸带的初始位置，读写头的行为由指令指示。

指令	含义
>	指针向右移动一位
<	指针向左移动一位
+	指针所指位置的值增加1字节
-	指针所指位置的值减少1字节
.	将指针所指位置的值按ASCII表输出
,	接受1字节的输入，存储在当前指针所指位置
[	当指针当前处的值为0时，跳转到对应]之后；否则，顺序执行
]	跳转回对应[处

在线编译网站

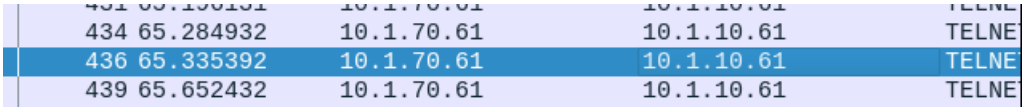
brainfuck

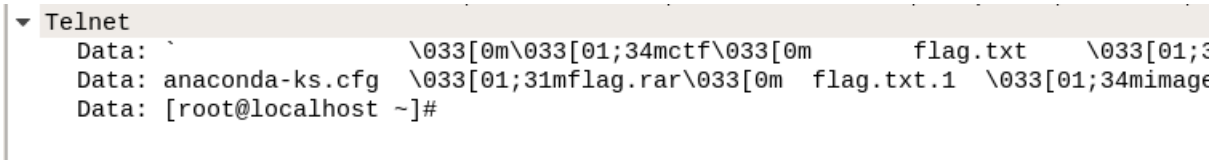
或者c的编译码

地址

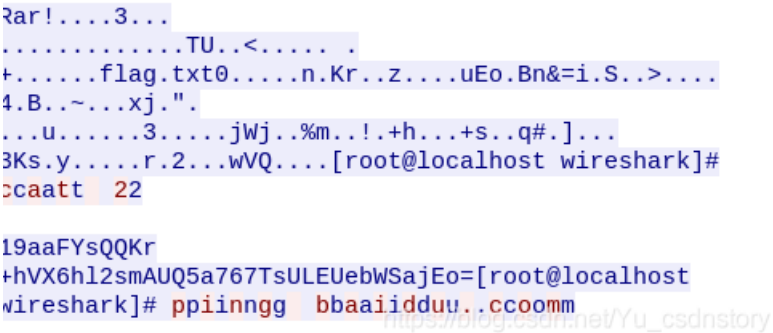
3-1

得到附件后解压，看出为rar文件，直接解压得到一个流量包，并搜寻flag





进行tcp流跟踪，在第六个流里看到了flag.rar，和一个加解密脚本，还有一串base64码



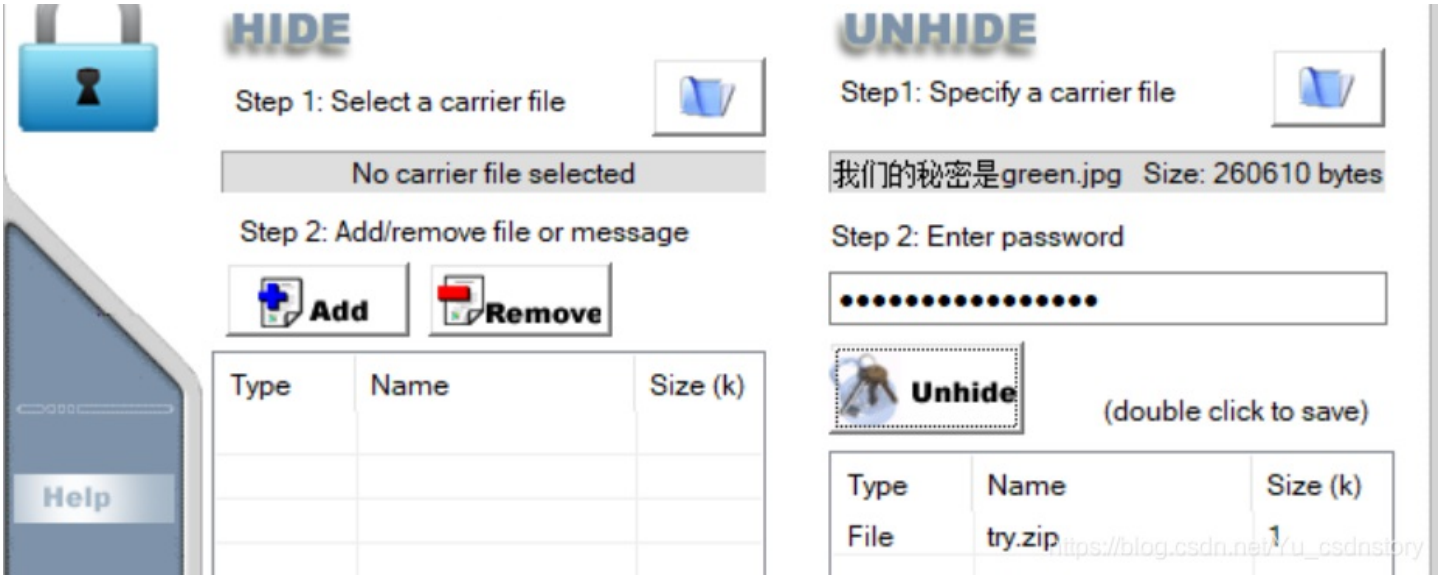
将rar文件提取出来，字符串码也复制出来。

```
# coding:utf-8
from Crypto import Random
from Crypto.Cipher import AES
import sys
import base64
IV = 'QWERTYUIOPASDFGH'
def decrypt(encrypted):

  aes = AES.new(IV, AES.MODE_CBC, IV)
  return aes.decrypt(encrypted)



def encrypt(message):

  length = 16

  count = len(message)

  padding = length - (count % length)

  message = message + '\0' * padding

  aes = AES.new(IV, AES.MODE_CBC, IV)

  return aes.encrypt(message)
str = '19aaFYsQQKr+hVX6hl2smAUQ5a767TsULEUebWSajEo='
example = decrypt(base64.b64decode(str))
print example
print decrypt(example)
```

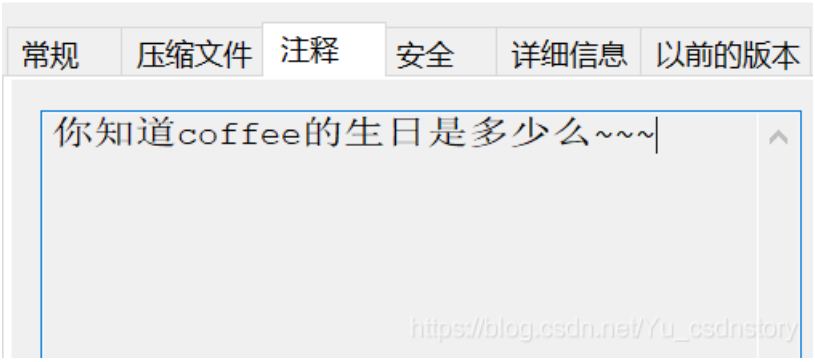
运行解密脚本，得到rar压缩包的密码，解压得到flag.txt,打开得到flag.

我们的秘密是绿色的

起初对图片进行了各种操作，但是仍然没有结果，最终在细看了题目，发现了图片上有日历内容是有特殊的显示，`0405111218192526`
这时，在工具oursecret中找到隐藏文件



拿到try.zip，在zip的属性中找到了提示信息



生日一般是8位数字，我们爆破密码可以得到

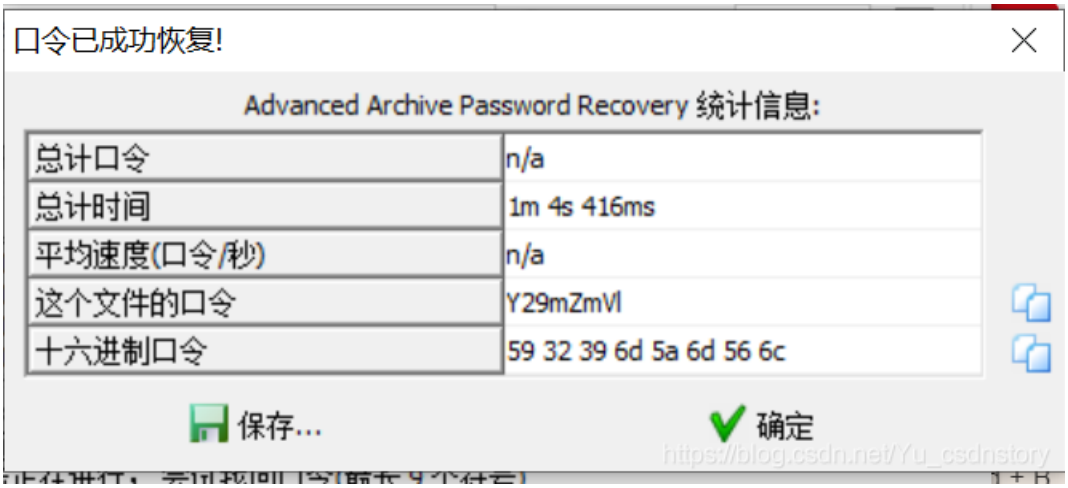
口令已成功恢复!

Advanced Archive Password Recovery 统计信息:

总计口令	19,950,820
总计时间	1s 246ms
平均速度(口令/秒)	16,011,894
这个文件的口令	19950822
十六进制口令	31 39 39 35 30 38 32 32

保存... 确定

解压后还是有密码的压缩包，readme.txt 压缩包和外部都有，我们进行明文掩码攻击



解压后又是密码，这次没有提示信息了，我们考虑压缩包伪加密。

```
50 4B 03 04 14 00 00 08 08 00 66 76 94 4A 7D AF  PK........fv"J}¯
72 9F 1E 00 00 00 1E 00 00 00 08 00 00 00 66 6C  rŸ.............fl
61 67 2E 74 78 74 2B 4C 49 29 28 2C CF 2B 48 2E  ag.txt+LI)(,Ï+H.
C8 49 CD 53 2D 28 02 B2 E3 AB E3 AB AA B4 52 1C  ÈIÍS-(.²ã«ã«ª´R.
D2 0B 6B 01 50 4B 01 02 3F 00 14 00 01 09 08 00  Ò.k.PK..?.....
66 76 94 4A 7D AF 72 9F 1E 00 00 00 1E 00 00 00  fv"J}¯rŸ........
08 00 24 00 00 00 00 00 00 00 20 00 00 00 00 00  ..$....... .....
00 00 66 6C 61 67 2E 74 78 74 0A 00 20 00 00 00  ..flag.txt.. ...
00 00 01 00 18 00 E6 FC D6 7E A2 B9 D2 01 2C E6  ......æüÖ~¢¹Ò.,æ
57 65 82 B9 D2 01 2C E6 57 65 82 B9 D2 01 50 4B  We,¹Ò.,æWe,¹Ò.PK
05 06 00 00 00 00 01 00 01 00 5A 00 00 00 44 00              Z   D
```

```
00 00 00 00 00 00 01 00 01 00 3A 00 00 00 44 00  ..........:...D.
00 00 00 00                                      ....
```


将01改为00,密码消失，解压得到flag.txt的内容

```
qddpqwnpcplen%prqwn_{_zz*d@gq}
```

很明显了，直接凯撒加栅栏解密，得到flag