

MISC常见题型整理

原创

[Angel~Yan](#) 于 2020-10-04 10:32:47 发布 4602 收藏 99

分类专栏: [MISC](#)

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/mcmuyanga/article/details/108904983>

版权



[MISC 专栏收录该内容](#)

2 篇文章 0 订阅

订阅专栏

题目打包在[这里](#)

提取码: fhkb

MISC

流量包分析

[流量包_1](#)

[流量包_2](#)

[流量包_3](#)

图片隐写

[图片隐写_1](#)

[图片隐写_2](#)

[图片隐写_3](#)

[图片隐写_4](#)

[图片隐写_5](#)

[图片隐写_6](#)

音频隐写

[音频隐写_1](#)

[音频隐写_3](#)

[音频隐写_4](#)

[音频隐写_5](#)

取证分析

[取证分析_1](#)

[取证分析_2](#)

神秘的文件

[disk](#)

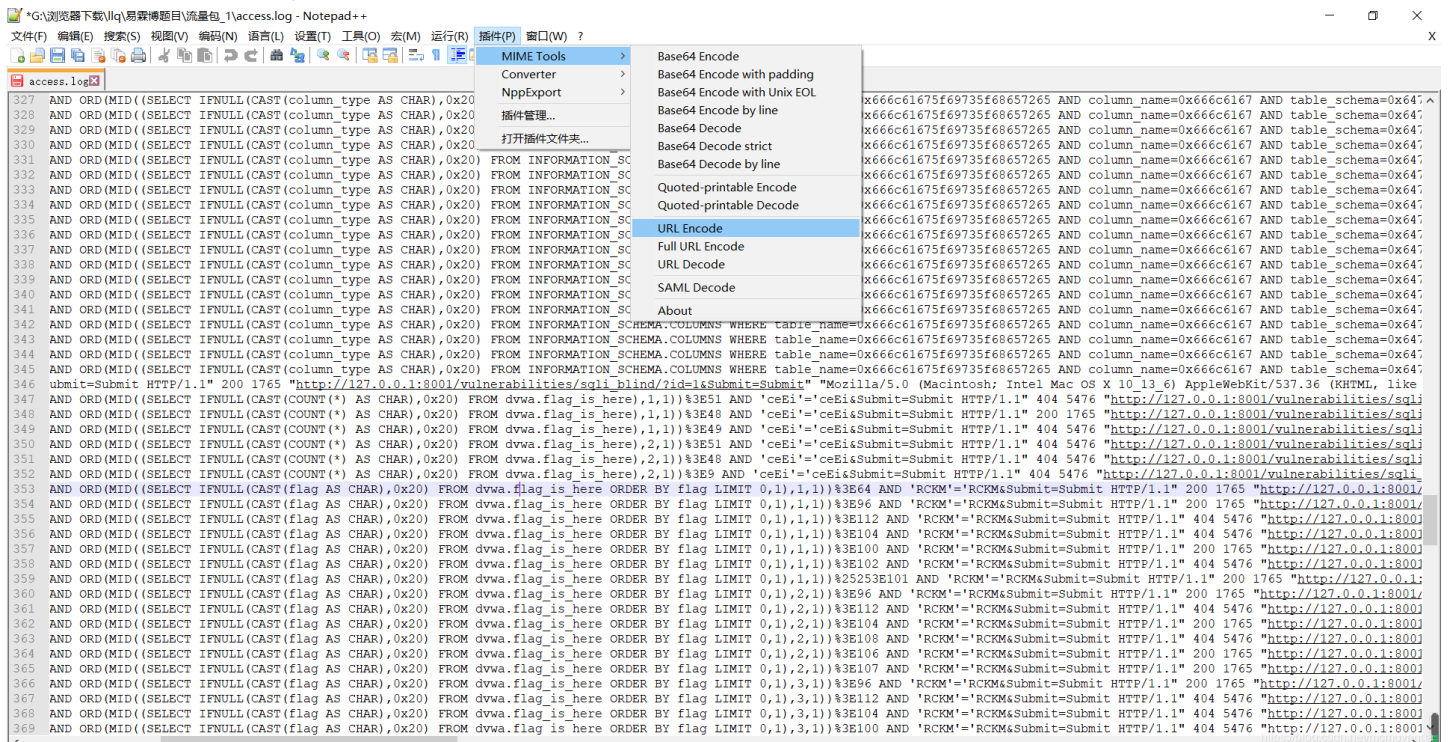
流量包分析

流量包_1

```
access.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

172.17.0.1 -- [03/Nov/2018:02:50:02 +0000] "GET /vulnerabilities/sqli_blind/?id=2&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "M
172.17.0.1 -- [03/Nov/2018:02:50:02 +0000] "GET /vulnerabilities/sqli_blind/?id=2&Submit=Submit&DieL=3713 AND 1=1 UNION ALL SELECT 1,NULL,'<script%3Ealert('XSS')</script%3E'.table_nam
172.17.0.1 -- [03/Nov/2018:02:50:03 +0000] "GET /vulnerabilities/sqli_blind/?id=2&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "M
172.17.0.1 -- [03/Nov/2018:02:50:03 +0000] "GET /vulnerabilities/sqli_blind/?id=2,...(&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "M
172.17.0.1 -- [03/Nov/2018:02:50:03 +0000] "GET /vulnerabilities/sqli_blind/?id=2&PHjkC<"%3EwOmGYB&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 3939=8029 AND (3898=3898&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sc
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6355=6355 AND (1179=1179&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sc
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 1813=7956&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Sul
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6355=6355&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Sul
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 3659=4375-- DkNq&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?i
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6355=6355-- Pedd&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6927=9421 AND ('gMkv'='gMkv&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilit
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6355=6355 AND ('WGPX'='WGPX&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilit
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 8078=8565 AND 'Wfgl'='Wfgl&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sc
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6355=6355 AND 'bOit'='bOit&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sc
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 6739=3560 AND 'qUXP'='qUXP&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sc
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT CHR(67)||CHR(65)||CHR(114)||CHR(114) FROM MSysAccessObjects)=CHR(67)||CHR(65)||CHR(114)
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT CHR(87)||CHR(107)||CHR(114)||CHR(109) FROM SYSIBM.SYSDUMMY1)=CHR(87)||CHR(107)||CHR(114)
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT 'SkRh' FROM RDBSDATABASE)='SkRh' AND 'Jvsf'='Jvsf&Submit=Submit HTTP/1.1" 404 5476 "http://
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT CHR(75)||CHR(106)||CHR(111)||CHR(70) FROM INFORMATION_SCHEMA.SYSTEM_USERS)=CHA
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT 'tkVkv' FROM SYSMASTER.SYSDUAL)='tkVkv' AND 'KcXq'='KcXq&Submit=Submit HTTP/1.1" 404 547
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT 'wlr' FROM VERSIONS)='wlr' AND 'JTCY'='JTCY&Submit=Submit HTTP/1.1" 404 5476 "http://127.0
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT CHR(75)+CHAR(90)+CHAR(102)+CHAR(74))=CHAR(75)+CHAR(90)+CHAR(102)+CHAR(74) AND 'N
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT CHR(75)+CHAR(90)+CHAR(102)+CHAR(74))=CHAR(121)+CHAR(73)+CHAR(65)+CHAR(117) AND 'N
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT 0x515a424a)=0x515a424a AND 'Gtzm'='Gtzm&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0
172.17.0.1 -- [03/Nov/2018:02:50:04 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (SELECT 0x515a424a)=0x546e5776 AND 'yzVs'='yzVs&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 25=25 AND 'PHUU'='PHUU&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 25=76 AND 'SYdD'='SYdD&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 76=64 AND 'KhrF'='KhrF&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 64=64 AND 'goL'='goL&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 76 64 AND 'gaGn'='gaGn&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND 9796=
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (2759)=2759 AND 'Jzff'='Jzff&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli
172.17.0.1 -- [03/Nov/2018:02:50:06 +0000] "GET /vulnerabilities/sqli_blind/?id=2) AND (2759)=2759 AND 'Jzff'='Jzff&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli"
```

是一个日志文件，用 notepad++ 打开查看，更清晰明了一些，利用插件解码



大致看了一下是二分法的sql盲注，每一条语句返回的状态码有200和404

我们往后继续看，看到有 flag_is_here 的记录，%3E通过url解码后是>

```
352 AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM dwva.flag_is_here),2,1))%3E9 AND 'ceBi'='ceBi&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli
353 AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM dwva.flag_is_here ORDER BY flag LIMIT 0,1,1))%3E64 AND 'RCkM'='RCkM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001
354 AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM dwva.flag_is_here ORDER BY flag LIMIT 0,1,1))%3E96 AND 'RCkM'='RCkM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001"
```

如何判断flag的第一个字符呢？

在sql盲注里测试的字符的 最后一条状态码为 200 的语句的 ASCII值再加1就是猜解正确的 ASCII值，转换成字符就是我们需要的答案的其中一个字符。

举个例子

```
52 '='%ceisSubmit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML
53 .1.1)%3E112 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) Ap
54 .1.1)%3E96 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) Ap
55 .1.1)%3E112 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) A
56 .1.1)%3E104 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) A
57 .1.1)%3E100 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) A
58 .1.1)%3E102 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) A
59 .1.1)%25253E101 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_
60 .2.1)%3E96 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 200 1765 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) Ap
61 .2.1)%3E112 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 404 5476 "http://127.0.0.1:8001/vulnerabilities/sqli_blind/?id=1&Submit=Submit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) A
```

flag_is_here ORDER BY flag LIMIT 0,1),1,1))>101 AND 'RCKM'='RCKM&Submit=Submit HTTP/1.1" 200 1765

我们可以看到当 LIMIT 0,1),1,1))>101 时，是第一个字符的最后一个状态码为 200 的语句，这时的 ASCII 码是 101，那么 101+1=102，ASCII 码为 102 对应的是 f，因此 flag 的第一个字符就为 f。

知道了原理，我们就可以去找到 flag 的字符串了，如果写不脚本的话就得要手工一个一个扣了

贴一下脚本 (python2.7)

```
import re
import urllib

f = open('D:/access.log', 'r') # 下载的access.log文件的绝对路径，记得写全英文
lines = f.readlines()
datas = []
for line in lines:
    t = urllib.unquote(line) # 就是将文本进行 urldecode 解码
    if '1765' in t and 'flag' in t: # 过滤出与flag相关，正确的猜测（只要200的）
        datas.append(t)

flag_ascii = {}
for data in datas:
    matchObj = re.search( r 'LIMIT 0,1\),(.*)\,1\)\)\>(.*?) AND', data) # 在date 中搜索符合正则表达的字符串并 将匹配的字符串存入变量 matchObj 中
    if matchObj:
        key = int(matchObj.group(1)) # 取变量matchObj 中的第一个括号里的内容（也就是上条语句中的 (.*) 中的内容）并转为10进制
        value = int(matchObj.group(2))+1 # 取变量matchObj中的第二个括号里的内容，并转为 10 进制
        flag_ascii[key] = value # 使用字典，保存最后一次猜测正确的ascii码

flag = ''
for value in flag_ascii.values():
    flag += chr(value)

print flag
```

```
-----
flag{sqlm4p_15_p0werful}
```

得到答案flag{sqlm4p_15_p0werful}

流量包_2

用wireshark打开流量包，这样设置检索一下有没有flag字符串，得到提示

The screenshot shows the Wireshark interface with the search filter 'flag' applied. The packet list shows several ARP and NBNS packets, and an ICMP Echo (ping) request at the bottom.

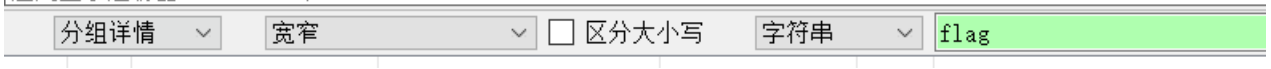
No.	Time	Source	Destination	Protocol	Length	Info
2...	26..	VMware_c0:00:...	Broadcast	ARP	60	who has 192.168.238.2? Tell 192.168.238.1
2...	27..	192.168.238.1..	192.168.238.2	NBNS	110	Refresh NB WORKGROUP<00>
2...	27..	VMware_c0:00:...	Broadcast	ARP	60	who has 192.168.238.2? Tell 192.168.238.1
2...	28..	VMware_c0:00:...	Broadcast	ARP	60	who has 192.168.238.2? Tell 192.168.238.1
2...	29..	VMware_c0:00:...	Broadcast	ARP	60	who has 192.168.238.2? Tell 192.168.238.1
2...	43..	192.168.238.1..	123.123.123.123	ICMP	55	Echo (ping) request id=0x0001, seq=0/0, ttl=64 (no response found!)

```

.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 41
Identification: 0x25ab (9643)
Flags: 0x4000, Don't fragment
 0... ..00 = Reserved bit: Not set
 .1... ..00 = Don't fragment: Set
 ..0... ..00 = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0x6eff [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.238.138
Destination: 123.123.123.123
> Internet Control Message Protocol
0000 00 50 56 f7 f5 6a 00 0c 29 3b 39 9a 08 00 45 00  ·PV·j·· );9··E·
0010 00 29 25 ab 40 00 40 01 6e ff c0 a8 ee 8a 7b 7b  ·)%·@·@· n····{{
0020 7b 7b 08 00 eb ec 00 01 00 00 66 6c 61 67 69 73  {{····· ·flagis
0030 68 65 72 65 00 00 00  here...

```

将检索设置改为这样，在查找，发现了几个很眼熟的东西



```

> Internet Control Message Protocol
0000 00 50 56 f7 f5 6a 00 0c 29 3b 39 9a 08 00 45 00  ·PV·j·· );9··E·
0010 00 29 25 ac 40 00 40 01 6e fe c0 a8 ee 8a 7b 7b  ·)%·@·@· n····{{
0020 7b 7b 08 00 91 fe 00 01 00 00 66 00 00 00 00 00  {{····· ·f····
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ······

```

backdoor++.pcap

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(I) 帮助(H)

应用显示过滤器: flag

No.	Time	Source	Destination	Protocol	Length	Info
2...	27...	VMware_c0:00:...	Broadcast	ARP	60	Who has 192.168.238.2? Tell 192.168.238.1
2...	28...	VMware_c0:00:...	Broadcast	ARP	60	Who has 192.168.238.2? Tell 192.168.238.1
2...	29...	VMware_c0:00:...	Broadcast	ARP	60	Who has 192.168.238.2? Tell 192.168.238.1
2...	43...	192.168.238.1...	123.123.123.123	ICMP	55	Echo (ping) request id=0x0001, seq=0/0, ttl=64 (no resp
2...	43...	192.168.238.1...	123.123.123.123	ICMP	55	Echo (ping) request id=0x0001, seq=0/0, ttl=64 (no resp
2...	43...	192.168.238.1...	123.123.123.123	ICMP	55	Echo (ping) request id=0x0001, seq=0/0, ttl=64 (no resp

```

0000 00.. = Differentiated Services Codepoint: Default (0)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 41
Identification: 0x25ad (9645)
Flags: 0x4000, Don't fragment
 0... ..00 = Reserved bit: Not set
 .1... ..00 = Don't fragment: Set
 ..0... ..00 = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0x6efd [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.238.138
Destination: 123.123.123.123
> Internet Control Message Protocol
0000 00 50 56 f7 f5 6a 00 0c 29 3b 39 9a 08 00 45 00  ·PV·j·· );9··E·
0010 00 29 25 ad 40 00 40 01 6e fd c0 a8 ee 8a 7b 7b  ·)%·@·@· n····{{
0020 7b 7b 08 00 8b fe 00 01 00 00 6c 00 00 00 00 00  {{····· ·1····
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ······

```

多点了几次，发现就是flag，整理一下
flag{icmp_backdoor_can_transfer-some_information}

流量包_3

打开附件是一个wifi.cap文件，这个是wifi握手包，通过握手包可以对密码进行暴力破解

flag格式: flag{你破解的WiFi密码}

tips: 密码为手机号，为了不为难你，大佬特地让我悄悄地把前七位告诉你

1391040**

Goodluck!!

用到了kali的两个软件,crunch 生成字典； aircrack-ng,爆破密码

利用kali里的crunch来生成字典

crunch命令格式: crunch <最小长度> <最大长度> [options]

```
crunch 11 11 -t 1391040%% %% >>wifi.txt
```

```
-t 指定模式
@ 插入小写字母
, 插入大写字母
% 插入数字
^ 插入特殊符号
```

得到手机号字典，之后用这个字典去爆破wifi密码

```
13910400000
13910400001
13910400002
13910400003
13910400004
13910400005
13910400006
13910400007
13910400008
13910400009
13910400010
13910400011
13910400012
13910400013
13910400014
13910400015
13910400016
13910400017
13910400018
13910400019
13910400020
13910400021
13910400022
13910400023
13910400024
13910400025
13910400026
13910400027
13910400028
13910400029
13910400030
13910400031
13910400032
13910400033
13910400034
13910400035
13910400036
13910400037
13910400038
13910400039
13910400040
13910400041
13910400042
13910400043
```

<https://blog.csdn.net/mcmuyanga>

-w后跟字典路径

```

Reading packets, please wait...

Aircrack-ng 1.2 rc4

[00:00:02] 7688/9999 keys tested (3829.18 k/s)

Time left: 0 seconds 76.89%

KEY FOUND! [ 13910407686 ]

Master Key : C4 60 FE 8B 14 7D 58 00 91 D7 0A 9C 3C DE 44 69
             0B E1 CD 81 07 F8 28 DB EA 76 1E ED 81 A3 FF FD

Transient Key : 0D 88 B3 F4 BC A3 C9 D2 06 12 28 43 FF 5E 21 3E
                F5 23 8E 0B 7A 9F 25 59 E9 7C 86 1E 7A 78 E4 D4
                D3 62 CD DD 4D 87 80 EE B9 E1 16 91 4A 6E 3E 09
                1E CE 5E 62 38 3C 05 35 34 A6 EB 16 31 D8 CE 96

EAPOL HMAC : 1C E7 D0 96 DE 87 93 56 88 1D 08 C8 B9 AA 93 B0

```

跑出来了答案，flag{13910407686}

图片隐写

图片隐写_1



得到一张图片，首先用winhex打开查看一下，首先ctrl+f，来搜索一下看看有没有flag字符串，发现没有，

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00001380	7E	FC	33	C6	1D	73	EB	CF	43	E8	47	22	8C	5F	5B	F8	~u3E seIC&g [e
00001390	02	DA	7E	AC	9A	BF	C3	CF	11	5D	E9	17	D1	31	78	E1	Ù--s;ãÏ jé Ñixá
000013A0	BA	63	2A	A9	DB	B4	2A	CA	3F	78	13	1C	6D	3B	EB	DE	*c*EÙ'·É?x m:ep
000013B0	68	A0	0F	9C	2E	FE	1D	7C	5D	97	57	B5	D7	B5	3D	63	h a.p)-Wu*xu=c
000013C0	40	D5	75	5B	58	BC	88	8C	57	57	16	AC	89	93	F7	4C	@Cu[X4'GWw -t"-L
000013D0	6A	8A	58	EE	60	4B	03	D4	67	22	A1	F1	47	C4	3D	63	j&Xi'K Ôg";AGÑ=c
000013E0	C0	AA	B1	F8	BB	FE	12	9D	1A	E9	C1	30	45	6B	AB	E9	À*æwp éA0Ekwé
000013F0	B7	FE	6A	FF	00	7B	12	C4	64	0B	90	79	DA	71	8F	AD	'bjÿ (Ad yÙq -
00001400	7D	2B	5C	DF	8C	2E	EE	34	A9	B4	9D	56	39	9C	59	43)+\B&.14&' V9æYC
00001410	74	B6	F7	B1	67	E5	68	A6	22	30	F8	F5	49	0C	6D	9E	tq:=g&h;"0æ&I m&Z
00001420	CB	BF	D6	80	3E	48	D4	3F	68	8F	12	48	87	FB	33	53	E¿Oe>H0?h H+03S
00001430	D6	AD	DB	1F	2F	DA	5E	C6	5C	F5	E7	E5	B3	5E	FF	00	ô-Ù /Ù^X\ôçá'^y
00001440	FE	BA	A3	7B	F1	7F	E2	55	DC	36	F1	49	E2	F4	58	EF	p'£(ã æU&ãI&0X1
00001450	18	22	25	AA	DA	C9	36	73	DF	C9	4D	EB	D4	75	03	3C	"*ÙÉ&æ&E&Me&Cu <
00001460	D6	8D	CB	25	F7	C4	9F	05	5A	A3	96	9E	2D	5A	0B	26	ô E%-AY Z&-Z-Z &
00001470	87	B0	8E	21	6E	D1	39	1F	5B	89	BF	01	8A	FA	EF	55	*Z!nÑ9 [t¿ SÙ1U
00001480	F0	F2	5F	34	62	0D	46	FF	00	4D	81	73	BE	1B	07	48	ô&4b Fÿ M s& H
00001490	44	A7	8E	59	B6	EE	C8	C7	1B	58	75	EF	40	1F	1F	6A	D&Zÿqie&C Xul& j
000014A0	3E	1A	D4	75	A1	34	BE	23	BC	F8	8F	E2	18	22	56	D9	> Cu;4N#æ & "VÙ
000014B0	23	E9	CD	6F	6A	EC	0F	0C	B3	CF	29	DA	87	D4	C6	08	#éioji 'I)Ù&Ö&Z
000014C0	CF	41	8A	66	85	E0	B9	AF	B1	17	87	BC	11	73	7F	71	I&ãf.â'^± #4 s q
000014D0	1F	31	BD	CE	B6	B7	F0	67	9E	0B	C1	12	C4	A7	A7	57	1&Ïq.ôg& Á Å&SSW
000014E0	53	C7	6E	DF	5D	5B	78	17	C3	D1	4E	93	DC	D8	7F	68	Sç&B)[x ÅNN"Ù& h
000014F0	DD	21	DC	B3	EA	72	BD	EC	88	7D	54	CA	5B	6F	FC	07	ÿ!Ù'æzti')TE[ou
00001500	15	D2	AA	85	01	54	00	07	40	05	00	7C	29	F1	1F	E1	ô'· T @)ã Á
00001510	BE	BD	E0	F1	A3	6B	DE	29	8E	CA	D7	4E	BB	98	59	CD	*4ã&ækb)Z&E"Ñ~'ÿf
00001520	1D	A6	FB	B7	81	48	24	E5	66	66	05	B6	87	20	06	C7	!ù· H&ãff q& Ç
00001530	CB	D4	1E	6B	E9	1F	08	FC	0B	F8	6D	6B	A4	43	3D	B6	E& k& ù æmk&C=ÿ
00001540	94	BA	B2	5D	41	95	BC	BB	99	A4	32	A3	8C	86	00	61	"*ù]A'4m"m&2&f+ a
00001550	14	E0	8C	32	A8	23	D7	35	D0	7C	6C	F0	7B	78	D7	E1	æ&2"#+5&B l&{x*â
00001560	F5	F6	99	02	EE	BB	46	5B	98	00	00	92	C8	4E	55	79	ô& m i&F[" 'ËNUy

```

00001570 1C B2 96 51 92 06 58 64 E2 B9 1F D9 5B C4 D2 6A  "Q' Xdã+ U[AC]
00001580 BE 01 97 41 D4 0B 8D 4F C3 F3 9B 49 12 4C 87 11  " -AC CÃó>I L+
00001590 12 4A 64 1E 98 21 D0 0E 38 41 40 1E B7 A3 69 56  "Jd '!B 8A@ `&iv
000015A0 1A 26 9F 15 86 91 67 05 9D 9C 43 09 0C 28 15 47  "ÿ t'g αC ( G
000015B0 A9 E3 BE 79 27 A9 35 7E 8A 28 00 A2 8A 28 00 A2  "eãNy'es-Š( cŠ( c
000015C0 8A 28 00 A2 8A 28 00 A2 8A 28 00 AC 3F 1B D9 41  "Š( cŠ( cŠ( -? ÚA
000015D0 A8 F8 37 5D B3 BB B8 6B 5B 79 EC 67 8D E7 54 DC  "a7}')»,k[yig çTÚ
000015E0 62 06 36 1B C0 EE 57 A8 FA 0A 28 A0 0F 9F BC 2D  "b é ãiW"ú ( Ÿ4-
000015F0 F0 EB 5A F1 0F 8A B4 5F 15 A5 90 B0 8B 4B 1A 72  "ëezñ Š' _ Y °<K r
00001600 05 68 FC A9 2E A4 5B 85 7B 92 E1 B0 7E 40 D2 26  "huc.µ[...'ã°~@ô&
00001610 71 F3 18 D4 2D 7D 3B 45 14 00 51 45 14 00 57 9E  "qó Œ-);E QE Wž
00001620 DA 78 3A 2D 0F E2 C3 EB FA 54 0D 0D BE AF 03 A5  "Úx:- ããëúT  " Y
00001630 E7 95 1E E5 33 0F 98 97 FE EE ED AA 43 72 01 57  "ç• áš "-pñi°Cr W
00001640 1D 64 06 8A 28 03 D0 A8 A2 8A 00 28 A2 8A 00 28  "d Š( D" cŠ( cŠ(
00001650 A2 8A 00 FF 26 23 31 30 37 3B 26 23 31 30 31 3B  "cŠ y&#107;ã#101;
00001660 26 23 31 32 31 3B 26 23 31 32 33 3B 26 23 31 32  "ã#121;ã#123;ã#12
00001670 31 3B 26 23 31 31 31 3B 26 23 31 31 37 3B 26 23  "1;ã#111;ã#117;ã#
00001680 33 32 3B 26 23 39 37 3B 26 23 31 31 34 3B 26 23  "32;ã#97;ã#114;ã#
00001690 31 30 31 3B 26 23 33 32 3B 26 23 31 31 34 3B 26  "101;ã#32;ã#114;ã
000016A0 23 31 30 35 3B 26 23 31 30 33 3B 26 23 31 30 34  "ã#105;ã#103;ã#104
000016B0 3B 26 23 31 31 36 3B 26 23 31 32 35 3B D9 D9  "ã#116;ã#129;ÚÚ

```

然后往下查看，在右边的代码中看到了一块与这群乱码格格不入的代码，猜测是一种编码
在CTF在线工具里，利用解码工具去尝试一下，在html编码里找到了flag

HTML编码

html

```

&#107;&#101;&#121;&#123;&#121;&#111;&#117;&#32;&#97;&#114;&#101;&#32;&#114;&#105;&#103;&#104;&#116;&#125;

```

编码 解码

```

key{you are right}

```

<https://blog.csdn.net/mcmuyanga>

key{you are right}

图片隐写_2



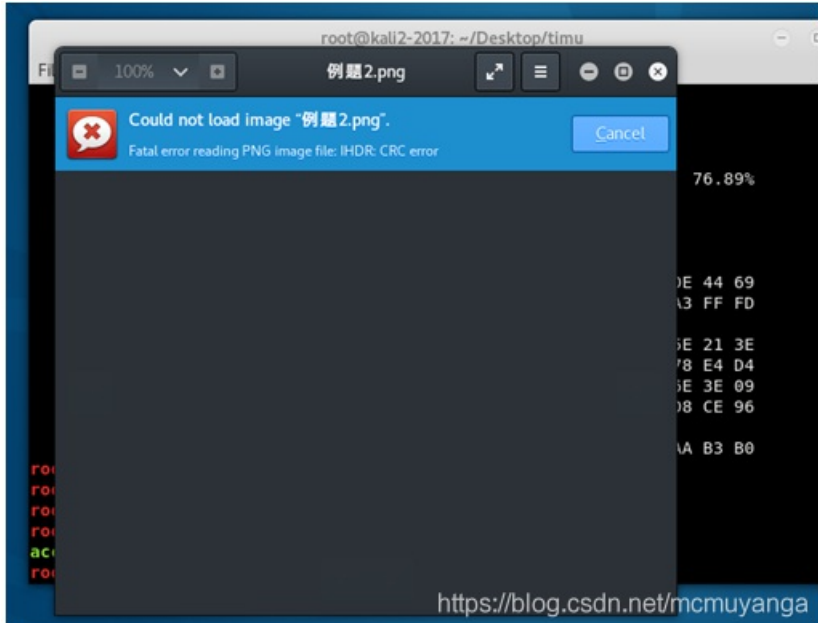
<https://blog.csdn.net/mcmuyanga>

先用winhex打开，老样子，先检索一下看看有没有flag关键字，没有找到

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	%PNG	IHDR	
00000010	00	00	01	F4	00	00	02	A4	08	06	00	00	00	CB	D6	DF	ó	µ	ËÖA
00000020	8A	00	00	00	09	70	48	59	73	00	00	12	74	00	00	12	Š	pHYs	t

00000030	74 01 DE 66 1F 78 00 00 0A 4D 69 43 43 50 50 68	t Bf x MiCCPPh
00000040	6F 74 6F 73 68 6F 70 20 49 43 43 20 70 72 6F 66	otoshop ICC prof
00000050	69 6C 65 00 00 78 DA 9D 53 77 58 93 F7 16 3E DF	ile xU SWX"> >B
00000060	F7 65 0F 56 42 D8 F0 B1 97 6C 81 00 22 23 AC 08	-e VB08i-1 "#-
00000070	C8 10 59 A2 10 92 00 61 84 10 12 40 C5 85 88 0A	È Yc ' a... @A...^
00000080	56 14 15 11 9C 48 55 C4 82 D5 0A 48 9D 88 E2 A0	V αHUA,Ō H ^á
00000090	28 B8 67 41 8A 88 5A 8B 55 5C 38 EE 1F DC A7 B5	(,gAS^Z<U\8i ŪSu
000000A0	7D 7A EF ED ED FB D7 FB BC E7 9C E7 FC CE 79 CF	
000000B0	0F 80 11 12 26 91 E6 A2 6A 00 39 52 85 3C 3A D8	È a'æçj 9R...<:Ø
000000C0	1F 8F 4F 48 C4 C9 BD 80 02 15 48 E0 04 20 10 E6	CHÄÉ:È Hà æ
000000D0	CB C2 67 05 C5 00 00 F0 03 79 78 7E 74 B0 3F FC	EÄg Ä ð yx~t°?ü
000000E0	01 AF 6F 00 02 00 70 D5 2E 24 12 C7 E1 FF 83 BA	-o pŌ.ç Çáýf°
000000F0	50 26 57 00 20 91 00 E0 22 12 E7 0B 01 90 52 00	F&W ' à" ç R
00000100	C8 2E 54 C8 14 00 C8 18 00 B0 53 B3 64 0A 00 94	È.TÈ È °s'd "
00000110	00 00 6C 79 7C 42 22 00 AA 0D 00 EC F4 49 3E 05	ly B" * iöI>
00000120	00 D8 A9 93 DC 17 00 D8 A2 1C A9 08 00 8D 01 00	ØE"Ü øc e
00000130	99 28 47 24 02 40 BB 00 60 55 81 52 2C 02 C0 C2	"(Gç @» `U R, ÅÅ
00000140	00 A0 AC 40 22 2E 04 C0 AE 01 80 59 B6 32 47 02	-@". Ås eyq2G
00000150	6C BD 05 00 76 8E 58 90 0F 40 60 00 80 99 42 2C	ε: vZx @` ePB,
00000160	CC 00 20 38 02 00 43 1E 13 CD 03 20 4C 03 A0 30	i s C í L 0
00000170	D2 BF E0 A9 5F 70 85 B8 48 01 00 C0 CB 95 CD 97	Ō:àæ p...,H ÅÈ-í-
00000180	4B D2 33 14 B8 95 D0 1A 77 F2 F0 E0 E2 21 E2 C2	KŌ3 ,.Đ wòðáá!ÅÅ
00000190	6C B1 42 61 17 29 10 66 09 E4 22 9C 97 9B 23 13	liBa) f ä"α->#
000001A0	48 E7 03 4C CE 0C 00 00 1A F9 D1 C1 FE 38 3F 90	Hç Lí ùÑÁp8?
000001B0	E7 E6 E4 E1 E6 66 E7 6C EF F4 C5 A2 FE 6B F0 6F	çääáafçl1óÄçpkóo
000001C0	22 3E 21 F1 DF FE BC 8C 02 04 00 10 4E CF EF DA	">!ñBp4G NiiÜ
000001D0	5F E5 E5 D6 03 70 C7 01 B0 75 BF 6B A9 5B 00 DA	_ääŌ pç °u;ke[Ū
000001E0	56 00 68 DF F9 5D 33 DB 09 A0 5A 0A D0 7A F9 8B	V hbù]3Ū Z Bzú<
000001F0	79 38 FC 40 1E 9E A1 50 C8 3C 1D 1C 0A 0B 0B ED	y8ú@ ž;PÈ< i
00000200	25 62 A1 BD 30 E3 8B 3E FF 33 E1 6F E0 8B 7E F6	ñb;40Ä<>y3áoä<~ò
00000210	FC 40 1E FE DB 7A F0 00 71 9A 40 99 AD C0 A3 83	ü@ pŪzð qš@P-Äšf
00000220	FD 71 61 6E 76 AE 52 8E E7 CB 04 42 31 6E F7 E7	yqanvSRZçÈ Bin+ç
00000230	23 FE C7 85 7F FD 8E 29 D1 E2 34 B1 5C 2C 15 8A	#pç... ýŽ)Ñá4±\, Š
00000240	F1 58 89 B8 50 22 4D C7 79 B9 52 91 44 21 C9 95	ñXñ,P"MCy'R'D!É•
00000250	E2 12 E9 7F 32 F1 1F 96 FD 09 93 77 0D 00 AC 86	á é 2ñ -ý "w -†
00000260	4F C0 4E B6 07 B5 CB 6C C0 7E EE 01 02 8B 0E 58	CÄNg µÈlÄ~i < X
00000270	D2 76 00 40 7E F3 2D 8C 1A 0B 91 00 10 67 34 32	Ōv @~ó-É ' g42
00000280	79 F7 00 00 93 BF F9 8F 40 2B 01 00 CD 97 A4 E3	y= "çù @+ í-mä
00000290	00 00 BC E8 18 5C A8 94 17 4C C6 08 00 00 44 A0	4è \'" LE D
000002A0	81 2A B0 41 07 0C C1 14 AC C0 0E 9C C1 1D BC C0	*°A Á -Ä çÁ 4Ä
000002B0	17 02 61 06 44 40 0C 24 C0 3C 10 42 06 E4 80 1C	a Dø çÄ< B äè
000002C0	0A A1 18 96 41 19 54 C0 3A D8 04 B5 B0 03 1A A0	; -A TÄ:Ø µ°
000002D0	11 9A E1 10 B4 C1 31 38 0D E7 E0 12 5C 81 EB 70	šá 'Á18 çà \ ep
000002E0	17 06 60 18 9E C2 18 BC 86 09 04 41 C8 08 13 61	` ZÄ 4† ÅÈ a
000002F0	21 3A 88 11 62 8E D8 22 CE 08 17 99 8E 04 22 61	!:' bZØ"í "Z "a
00000300	48 34 92 80 A4 20 E9 88 14 51 22 C5 C8 72 A4 02	H4'Ø'ä"Ō"ÄEwngä
00000310	28 42 6A 81 8D 48 23 F2 2D 72 14 28 8D 5C 40 F8	ØB411HFA...v 0 186

猜测图片修改了宽高，放到kali打开，如果修改了宽高，kali会报错



果真修改了宽高
 百度了一下知道了
 89 50 4E 47PE头是png照片的，就是说没有可能照片中嵌入了Exif信息
 给大家稍微介绍一下

89 50 4E 47 是文件的格式

00 00 00 0D 说明IHDR头块长为13

49 48 44 52 IHDR标识

00 00 01 F4 图像的宽

00 00 01 A4 图像的高

最后四位

CB D6 DF 8 A为CRC校验

在查看PNG文件格式时，IHDR后面的八个字节就是宽高的值，这边报错就是因为图片的宽高被修改了
将图片的宽高修改成一样看看

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	%PNG	IHDR
00000010	00	00	01	F4	00	00	01	F4	08	06	00	00	00	CB	D6	DF	ô õ	ËÖß
00000020	8A	00	00	00	09	70	48	59	73	00	00	12	74	00	00	12	š	pHYs t
00000030	74	01	DE	66	1F	78	00	00	0A	4D	69	43	43	50	50	68	t Þf x	MiCCPPh
00000040	6F	74	6F	73	68	6F	70	20	49	43	43	20	70	72	6F	66	otoshop	ICC prof
00000050	69	6C	65	00	00	78	DA	9D	53	77	58	93	F7	16	3E	DF	ile xÚ SwX"÷ >ß	
00000060	F7	65	0F	56	42	D8	F0	B1	97	6C	81	00	22	23	AC	08	÷e VB0ði-1 "#-	
00000070	C8	10	59	A2	10	92	00	61	84	10	12	40	C5	85	88	0A	È Yc ' a,, @Å...^	
00000080	56	14	15	11	9C	48	55	C4	82	D5	0A	48	9D	88	E2	AO	V	αHUA.Ö H "ä

BU

BUGKU{a1e5aSA}

<https://blog.csdn.net/micmuyanga>

BUGKU{a1e5aSA}

图片隐写_3

wc，这不是我们物联网课程的图标么





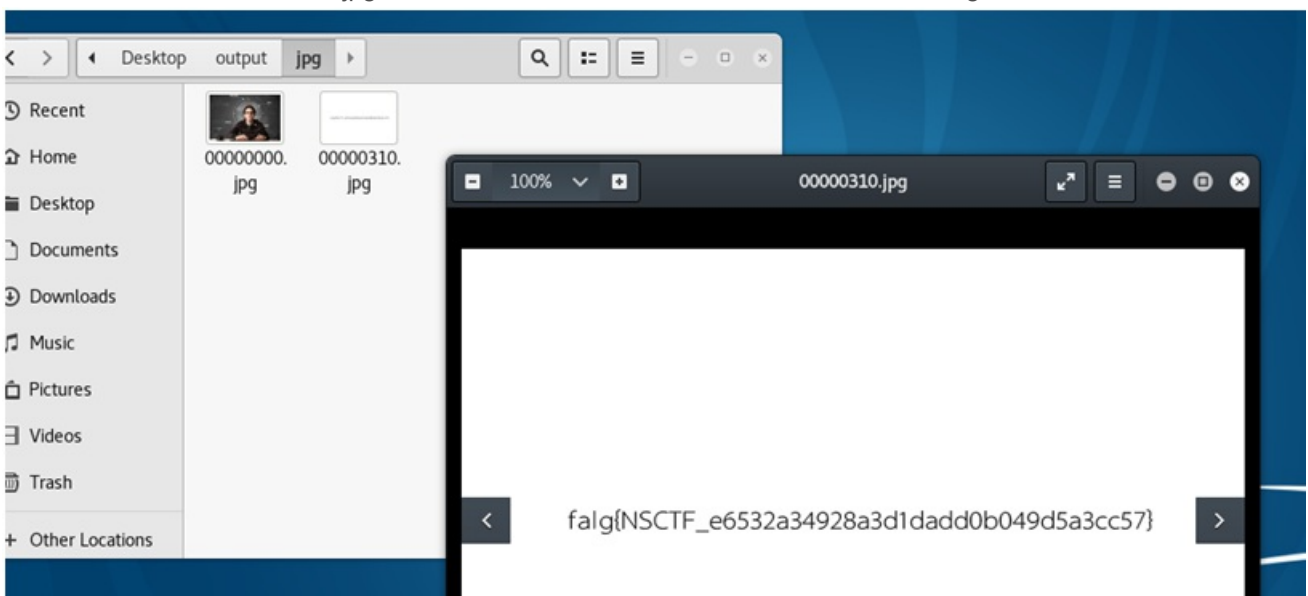
一样先用winhex打开，检索flag字符串，没有找到，放到kali里打开，能打开，说明没有修改宽高，在kali里用binwalk检查一下图片

```
root@kali2-2017:~/Desktop# binwalk -e 3.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian, offset of first image directory: 8
13017	0x32D9	Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#"><rdf:Description rdf:about="" xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/" xmlns
158792	0x26C48	JPEG image data, JFIF standard 1.02
158822	0x26C66	TIFF image data, big-endian, offset of first image directory: 8
159124	0x26D94	JPEG image data, JFIF standard 1.02
162196	0x27994	JPEG image data, JFIF standard 1.02
164186	0x2815A	Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#"><rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:xap="htt
168370	0x291B2	Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"

<https://blog.csdn.net/mcmuyanga>

有猫腻，隐藏了东西，用foremost 3.jpg对它进行文件恢复，然后多了一个图片，打开得到flag



flag{NSCTF_e6532a34928a3d1dadd0b049d5a3cc57}

图片隐写_4



先放winhex里，没有找到flag，kali打开，没有修改宽高，binwalk分析，没有隐藏文件，百度后发现一个神奇的工具zsteg

```
gem install zsteg #安装zsteg
```

[关于zstog工具看这里](#)

我们在kali里直接利用这个工具跑一下即可

```
root@kali2-2017:~/Desktop/tpyx_4_1# zsteg 例题4-1.png
imagedata          .. text: "fp\,RB886\"
b1,bgr,lsb,xy     .. text: "ZmxhZ3s1NDJmYjQzM2Y0Yzh1YmYwMmY3OGUwZja0MzZkMmU3Nm0="
b2,rgb,msb,xy     .. text: "@UUUUUUU"
b2,bgr,msb,xy     .. text: "@UUUUUUU"
b4,r,lsb,xy       .. text: "\"3\"$DEWTEgv"
b4,r,msb,xy       .. text: "n\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"fff"
b4,g,lsb,xy       .. text: "22\"5ETVUDfw"
b4,g,msb,xy       .. text: "n\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"ffff"
b4,b,lsb,xy       .. text: "$UUFDUgv"
b4,b,msb,xy       .. text: "n\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"\"f"
b4,rgb,lsb,xy     .. text: "!\#$%&'()*+,-./:;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_`{|}~!@#TDTUEUEUGfUDTDUEfggwvv"
b4,bgr,lsb,xy     .. text: "!\\"#$%&'()*+,-./:;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_`{|}~!@#TTTETUEEEVgETTTEEfggwvv"
root@kali2-2017:~/Desktop/tpyx_4_1#
```

<https://blog.csdn.net/mcmuyanga>

看到一个特殊的base64编码，拿去解码一下

base编码

base16, base32, base64

ZmxhZ3s1NDJmYjQzM2Y0Yzh1YmYwMmY3OGUwZja0MzZkMmU3Nm0=

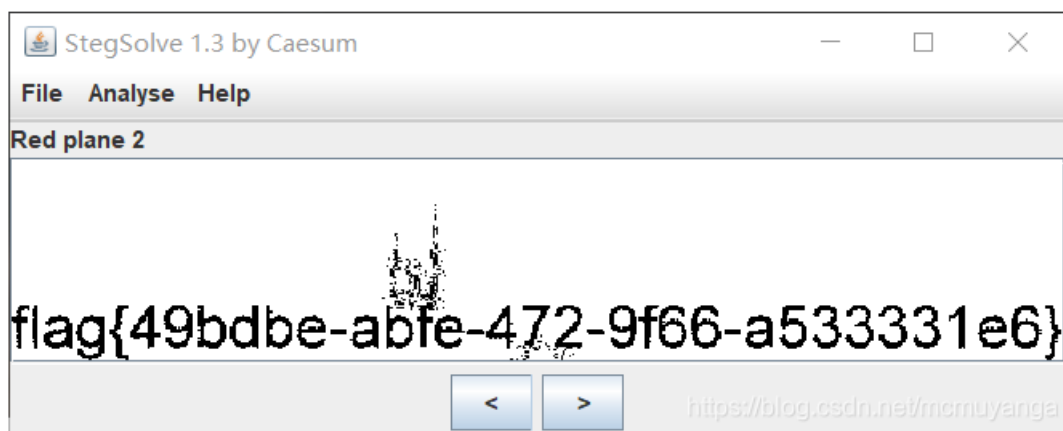
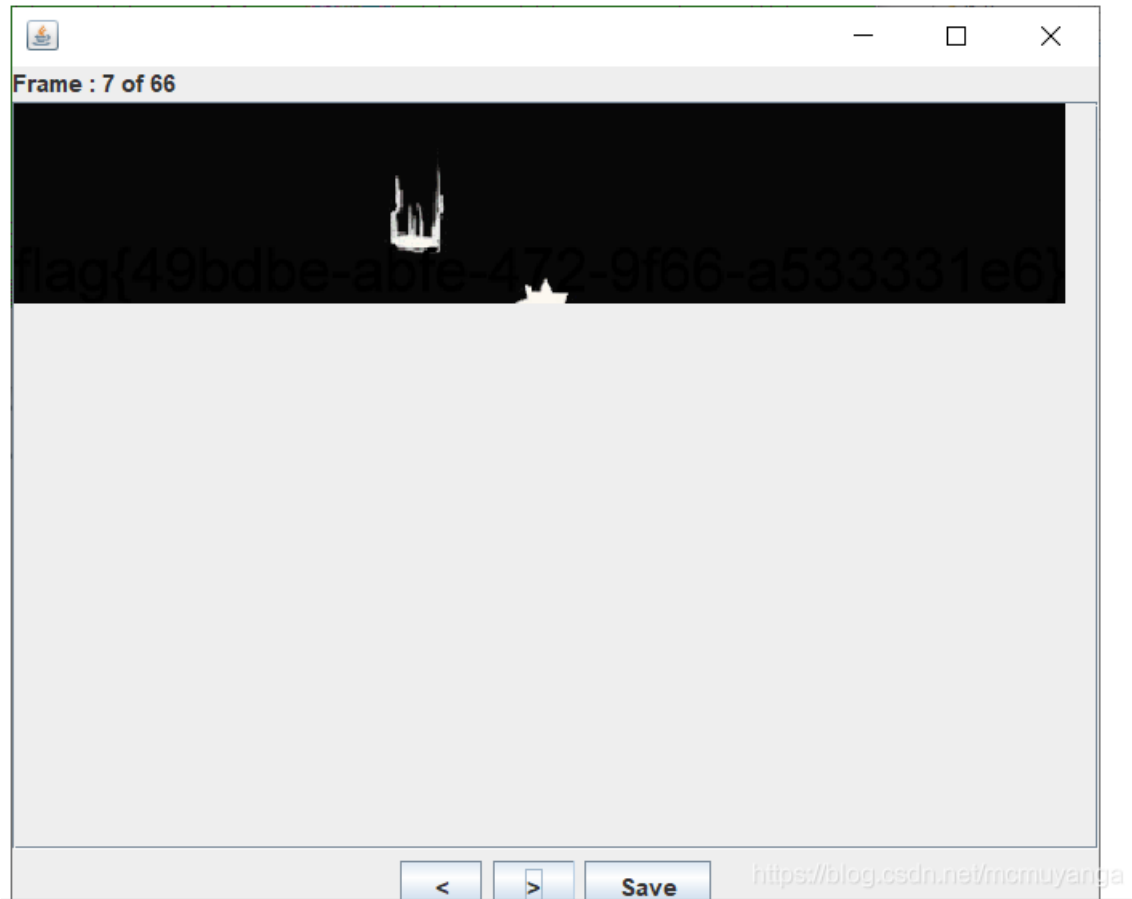
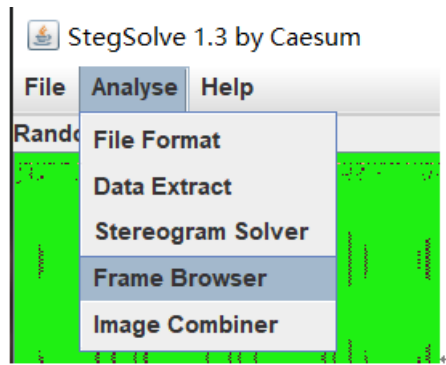
编码 字符集

flag{542fb433f4c8ebf02f78e0f0436d2e76}

<https://blog.csdn.net/mcmuyanga>

flag{542fb433f4c8ebf02f78e0f0436d2e76}

图片隐写_5

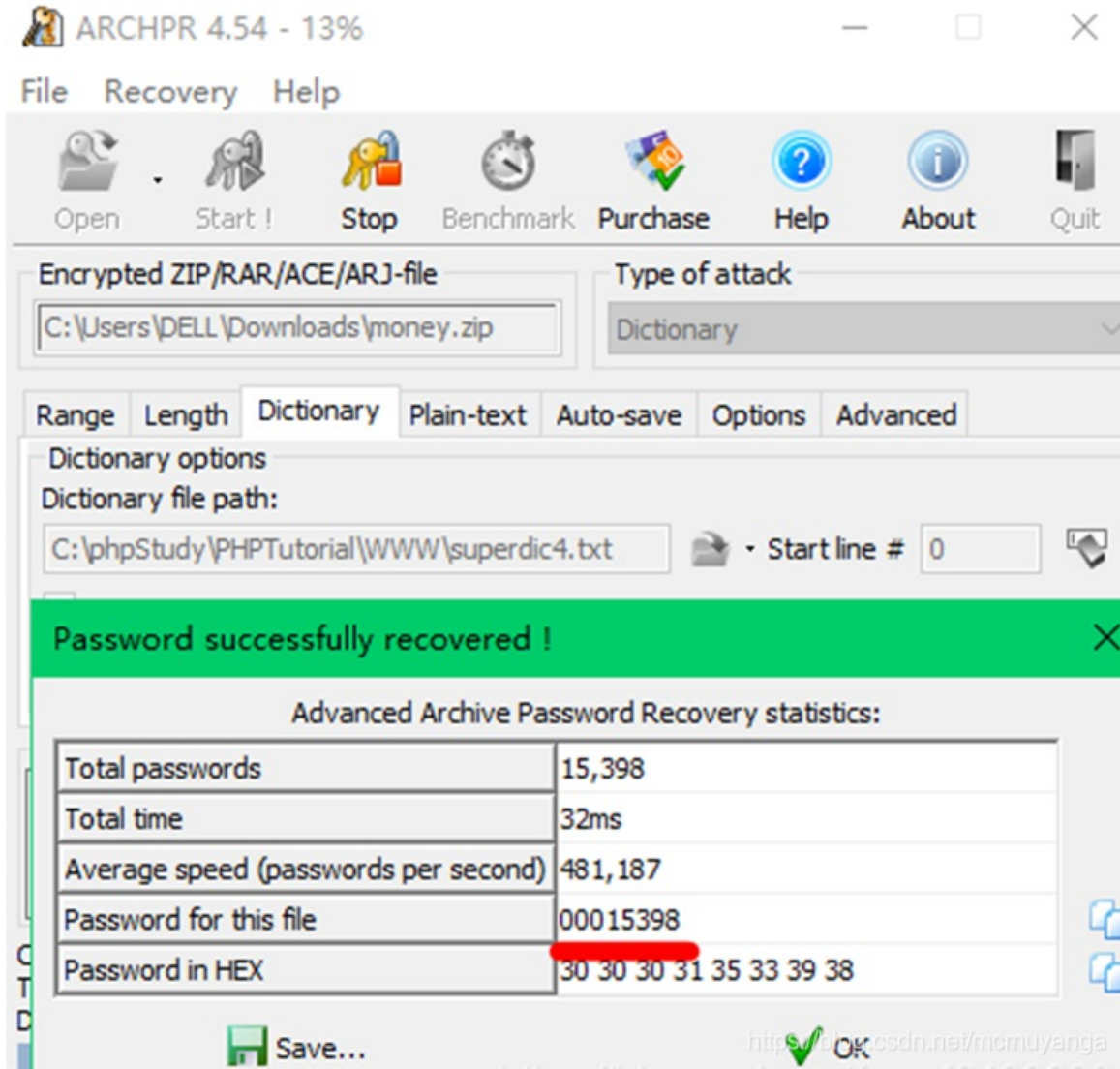


flag{49bdbe-abfe-472-9f66-a533331e6}

音频隐写

音频隐写_1

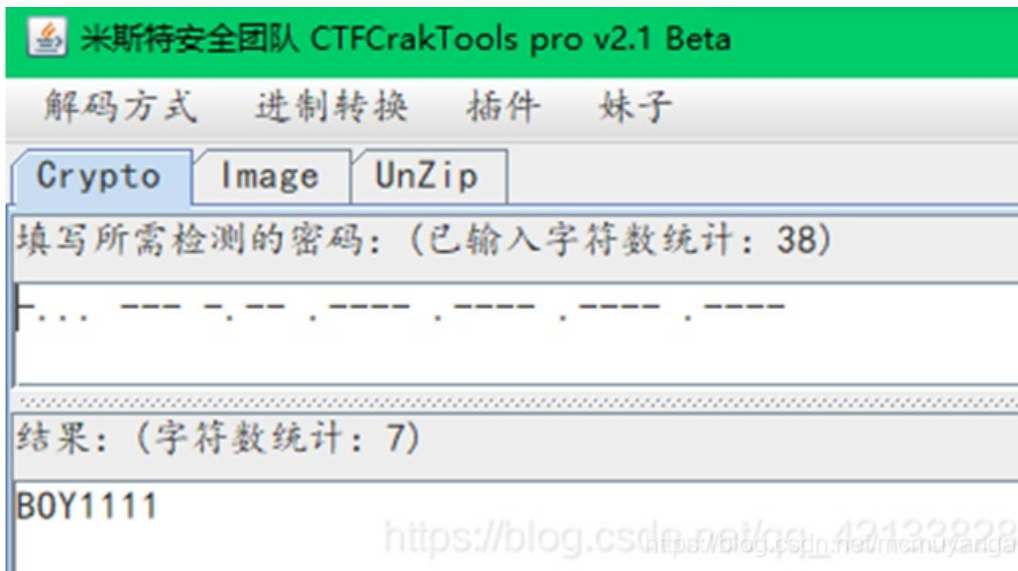
这题本来有两个文件的，那个用来破解的密码的压缩包找不到了，凑合看一下
原本打开下载的文件，会发现里面得两个文件都需要密码解密，意味着没有任何线索来获取密码
那么就只能粗暴一点，直接爆破出他的密码，即



解密money.zip，并打开里面得文件，在文件的最底部，即：

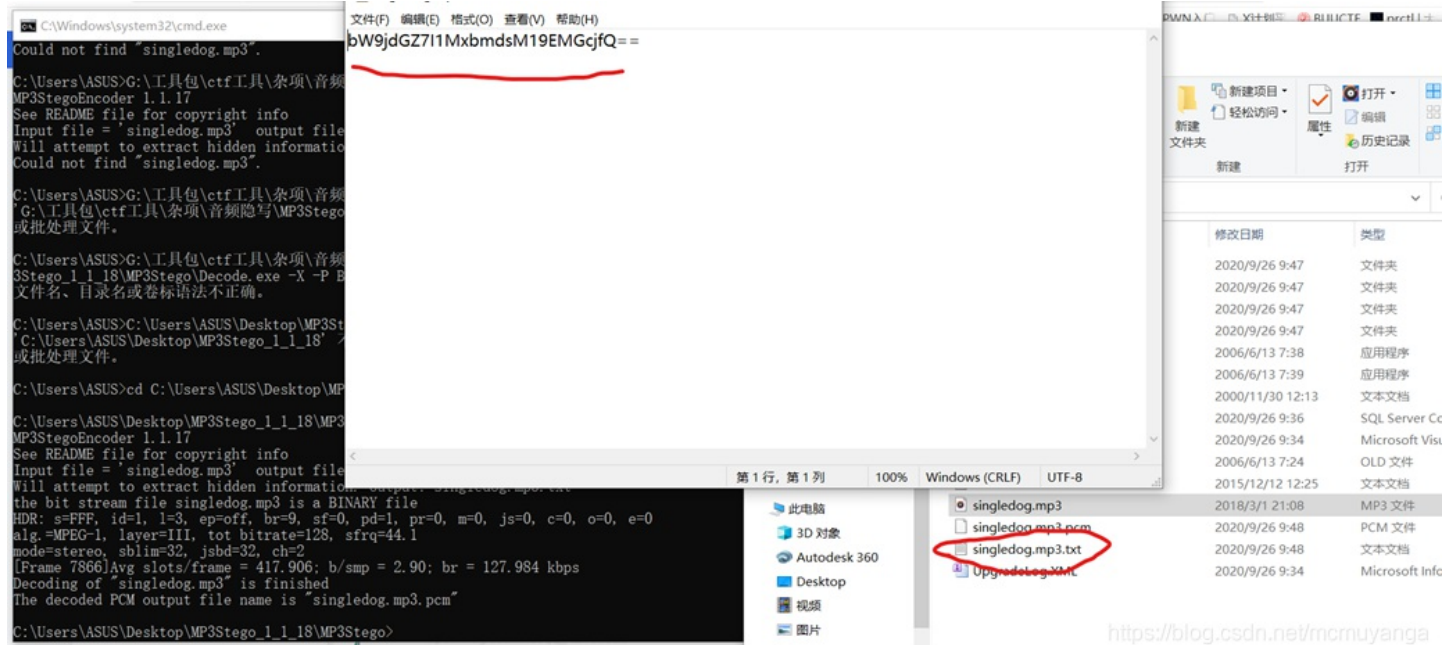


摩斯密码解密得：



解压包里面还有一个mp3的文件，又得到密码，那么使用工具MP3Stego

在cmd界面下，将decode.exe拖进去运行，之后我们会得到一个txt文件，里面是base64编码，拿去解码一下即可



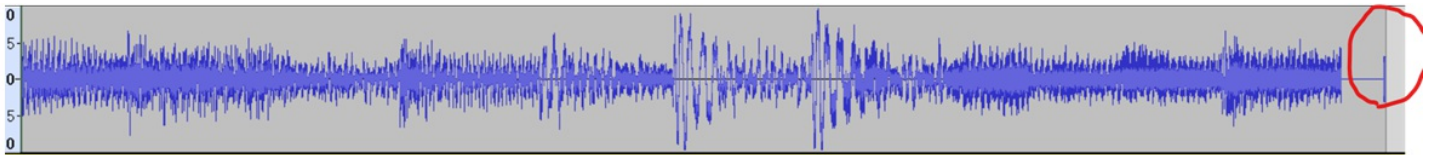
Dase1b、Dase3Z、Dase64



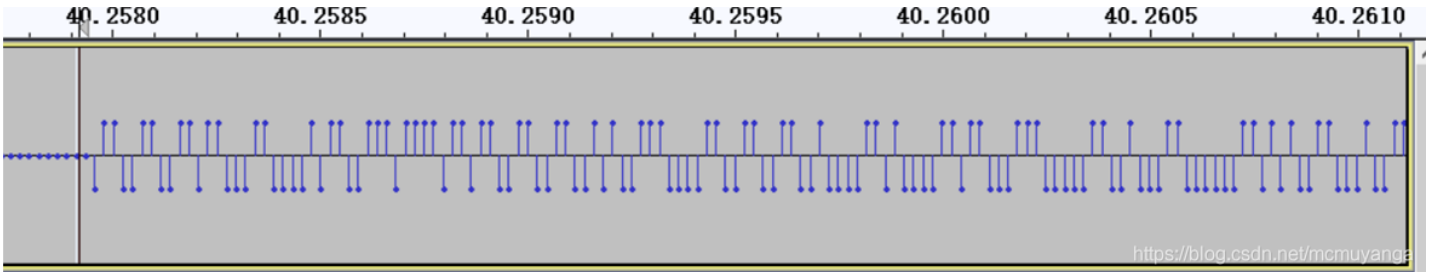
moctf{#S1ngl3_D0g#}

音频隐写_3

音频文件，首先用audacity打开，查看音谱，发现最好一部分有点奇怪



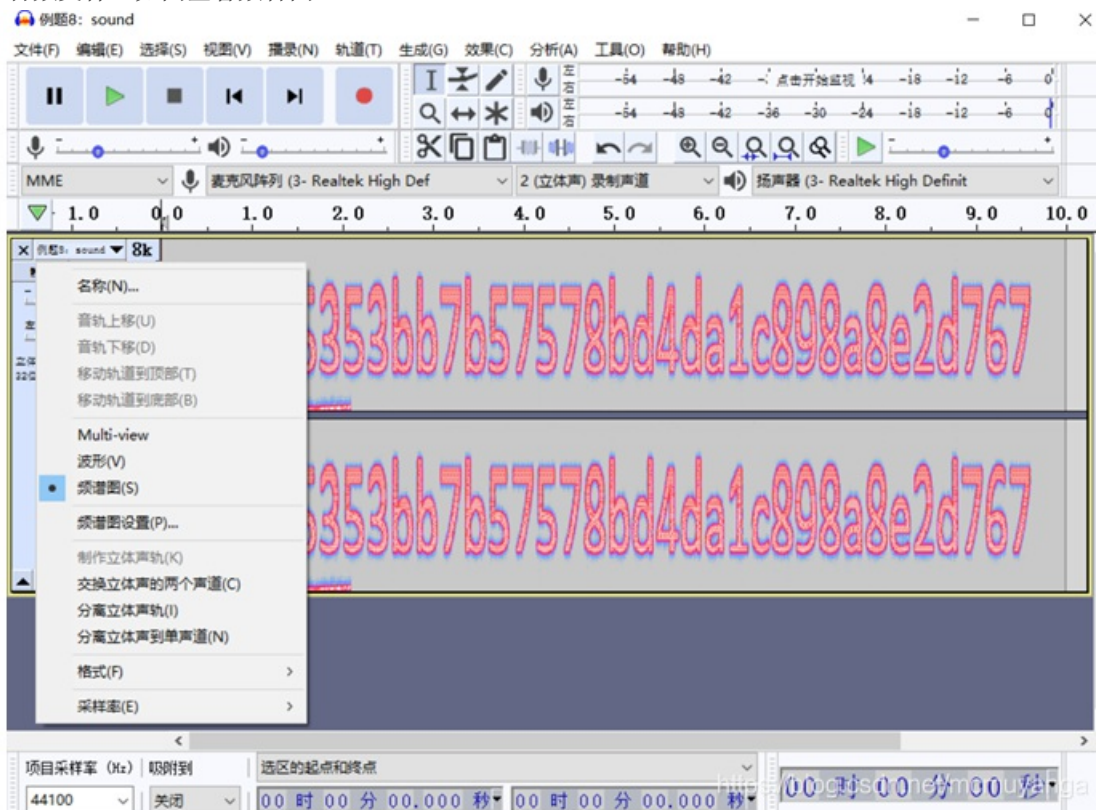
放大后发现后面这一块长这样



假设在上方是1，下方是0，整理一下得到一串01字符串，8位一组将得到的十进制数用ascii码替换这个一个一个数起来有点长，我数了两次，都数错了，没有做完，这边就介绍一下思路和方法

音频隐写_4

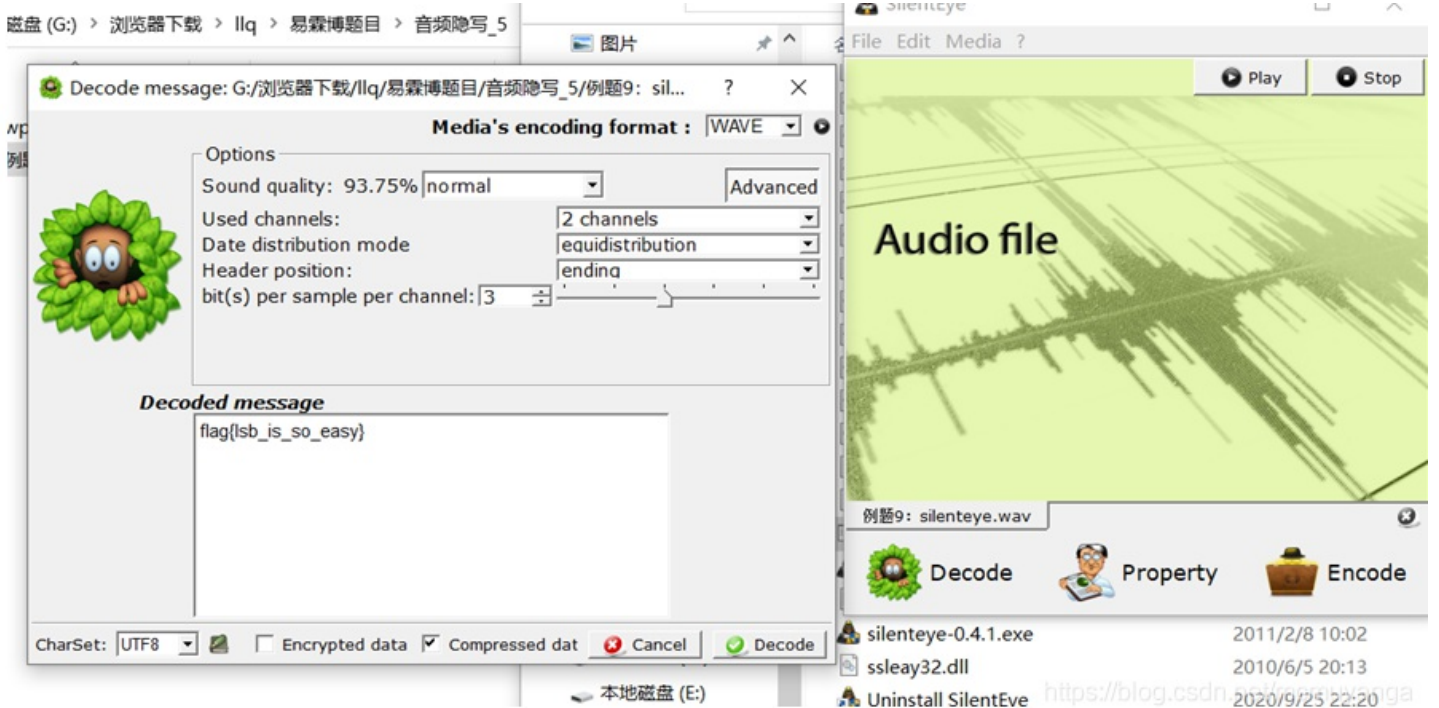
用Audacity载入音频文件，如图查看频谱图



flag{e5353bb7b57578bd4da1c898a8e0d767}

音频隐写_5

看附件提示是silenteye，使用工具打开，解密一下就得到了flag



flag{lsb_is_so_easy}

取证分析

取证分析_1

首先放到kali里file一下看看文件，是一个ext4文件

```
root@kali2-2017:~/Desktop/qzfx_1# file attachment.img
attachment.img: Linux rev 1.0 ext4 filesystem data, UUID=2d362e1b-69ae-4137-bdbb-4fde2775ac91 (extents) (huge files)
```

用extundelete对ext4文件系统进行数据恢复，恢复的目录保存在RECOVERED_FILES

```
root@kali2-2017:~/Desktop/qzfx_1# extundelete attachment.img --restore-all
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 2 groups loaded.
Loading journal descriptors ... 151 descriptors loaded.
Searching for recoverable inodes in directory / ...
2 recoverable inodes found.
Looking through the directory structure for deleted files ...
1 recoverable inodes still lost.
root@kali2-2017:~/Desktop/qzfx_1# ls
attachment.img RECOVERED_FILES
```

打开目录文件，读出flag

```
root@kali2-2017:~/Desktop/qzfx_1# ls
attachment.img RECOVERED_FILES
root@kali2-2017:~/Desktop/qzfx_1# cd RECOVERED_FILES/
root@kali2-2017:~/Desktop/qzfx_1/RECOVERED_FILES# ls
file.17
root@kali2-2017:~/Desktop/qzfx_1/RECOVERED_FILES# cat file.17
b0VIM 7.4
U3210#"! t p d f l a g { f u g l y _ c a t s _ n e e d _ l u v _ 2 } r
oot@kali2-2017:~/Desktop/qzfx_1/RECOVERED_FILES#
```

flag{fugly_cats_need_luv_2}

取证分析_2

打开是2个文件



volatility 使用:

volatility -f <文件名> --profile=<配置文件> <插件> [插件参数]

获取--profile的参数

使用imageinfo插件来猜测dump文件的profile值: WinXPSP2x86

volatility -f mem.vmem imageinfo

```
root@kali2-2017:~/Desktop/qzfx_2# volatility -f mem.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
AS Layer1 : IA32PagedMemoryPae (Kernel AS)
AS Layer2 : FileAddressSpace (/root/Desktop/qzfx_2/mem.vmem)
PAE type : PAE
DTB : 0xb18000L
KDBG : 0x80546ae0L
Number of Processors : 1
Image Type (Service Pack) : 3
KPCR for CPU 0 : 0xffdf000L
KUSER_SHARED_DATA : 0xffdf000L
Image date and time : 2016-05-03 04:41:19 UTC+0800
Image local date and time : 2016-05-03 12:41:19 +0800
```

列举进程, 可以发现一个TrueCryp.exe的进程。

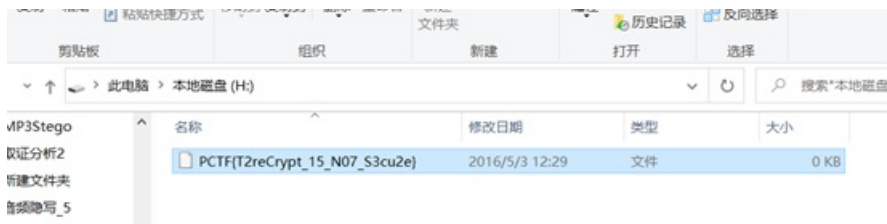
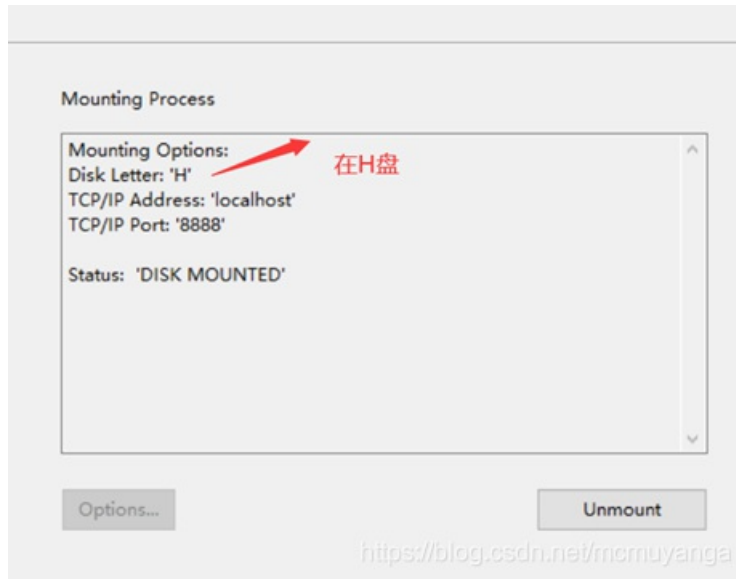
volatility -f mem.vmem --profile=WinXPSP2x86 pslist

```
root@kali2-2017:~/Desktop/qzfx_2# volatility -f mem.vmem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V) Name PID PPID Thds Hnds Sess Wow64 Start
-----
0x821b9830 System 4 0 62 253 ----- 0
0x81fb9210 smss.exe 552 4 3 19 ----- 0 2016-05-03 04:32:10 UTC+0
0x81c14da0 csrss.exe 616 552 10 328 0 0 2016-05-03 04:32:12 UTC+0
0x81f81880 winlogon.exe 640 552 18 449 0 0 2016-05-03 04:32:12 UTC+0
0x8208fda0 services.exe 684 640 16 260 0 0 2016-05-03 04:32:12 UTC+0
0x81c32b10 lsass.exe 696 640 18 333 0 0 2016-05-03 04:32:12 UTC+0
0x820a19a0 vmacthlp.exe 852 684 1 25 0 0 2016-05-03 04:32:13 UTC+0
0x81c30450 svchost.exe 864 684 18 201 0 0 2016-05-03 04:32:13 UTC+0
0x81c67020 svchost.exe 948 684 11 238 0 0 2016-05-03 04:32:13 UTC+0
0x81ce7da0 svchost.exe 1040 684 55 1103 0 0 2016-05-03 04:32:13 UTC+0
0x81c25020 svchost.exe 1096 684 4 66 0 0 2016-05-03 04:32:13 UTC+0
0x82002b28 svchost.exe 1256 684 13 194 0 0 2016-05-03 04:32:14 UTC+0
0x81fc988 explorer.exe 1464 1448 12 329 0 0 2016-05-03 04:32:14 UTC+0
0x82085550 spoolsv.exe 1576 684 13 140 0 0 2016-05-03 04:32:14 UTC+0
0x81f64560 vmttoolsd.exe 1712 1464 5 145 0 0 2016-05-03 04:32:15 UTC+0
0x820a3528 ctfmon.exe 1736 1464 1 78 0 0 2016-05-03 04:32:15 UTC+0
0x81f7d3c0 vmttoolsd.exe 2020 684 7 273 0 0 2016-05-03 04:32:23 UTC+0
0x8207db28 TPAutoConnSvc.e 512 684 5 99 0 0 2016-05-03 04:32:25 UTC+0
0x81c26da0 alg.exe 1212 684 6 105 0 0 2016-05-03 04:32:26 UTC+0
0x81f715c0 wscntfy.exe 1392 1040 1 39 0 0 2016-05-03 04:32:26 UTC+0
0x81e1f520 TPAutoConnect.e 1972 512 1 1 0 0 2016-05-03 04:32:26 UTC+0
0x81f9d3e8 TrueCrypt.exe 2012 1464 2 139 0 0 2016-05-03 04:33:36 UTC+0
```

TrueCryp.exe是一款加密程序, 而我们可以推出, suspicion为加密的结果。我们需要从内存dump出key来。dump出来的文件为1464.dmp。

```
root@kali2-2017:~/Desktop/qzfx_2# volatility -f mem.vmem --profile=WinXPSP2x86 memdump -p 1464 -D /root/Desktop/qzfx_2/
Volatility Foundation Volatility Framework 2.6
Writing explorer.exe [ 1464] to 1464.dmp
```


找到key保存

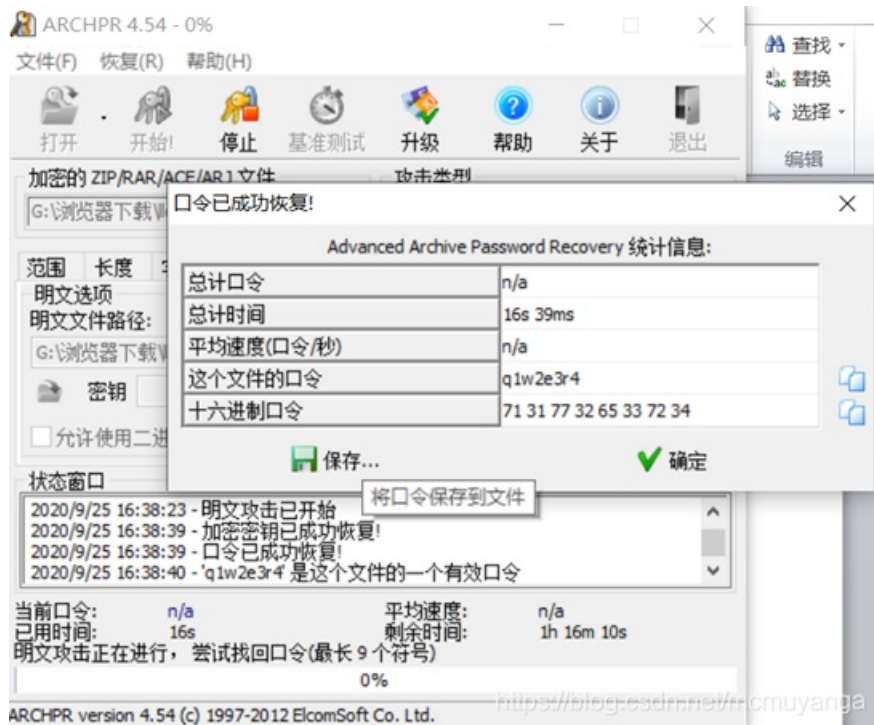


PCTF{T2reCrypt_15_N07_S3cu2e}

神秘的文件

附件是一个flag.zip压缩包（需要密码）和图片，双击压缩包发现图片也在那个压缩包里，是明文攻击，利用winrar将logo压缩成zip，记住一定要用winrar压缩

之后使用ARCHPR进行破解，得到密码



拿这个密码去打开加密的文件



哪有什么 WriteUP，别想了，老老实实做题吧！

将文件放到kali里用binwalk看一下，

```
root@kali2-2017:~/Desktop/tpyx_smwj# binwalk 2018山东省大学生网络安全技能大赛决赛writeup.docx
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Zip archive data, at least v2.0 to extract, compressed size: 362, uncompressed size: 695, name: docProps/app.xml
672	0x2A0	Zip archive data, at least v2.0 to extract, compressed size: 387, uncompressed size: 773, name: docProps/core.xml
1370	0x55A	Zip archive data, at least v1.0 to extract, compressed size: 36452, uncompressed size: 36452, name: docProps/thumbnail.jpeg
37875	0x93F3	Zip archive data, at least v2.0 to extract, compressed size: 1285, uncompressed size: 4056, name: word/document.xml
39207	0x9927	Zip archive data, at least v2.0 to extract, compressed size: 476, uncompressed size: 1529, name: word/fontTable.xml
39731	0x9B33	Zip archive data, at least v1.0 to extract, compressed size: 222845, uncompressed size: 222845, name: word/media/image1.png
262627	0x401E3	Zip archive data, at least v2.0 to extract, compressed size: 1117, uncompressed size: 2847, name: word/settings.xml
263791	0x4066F	Zip archive data, at least v2.0 to extract, compressed size: 2920, uncompressed size: 29509, name: word/styles.xml

果真有猫腻，foremost分离一下文件，在里面找到了flag.txt，打开发现是base64编码，解码一下



flag{d0cX_1s_ziP_file}

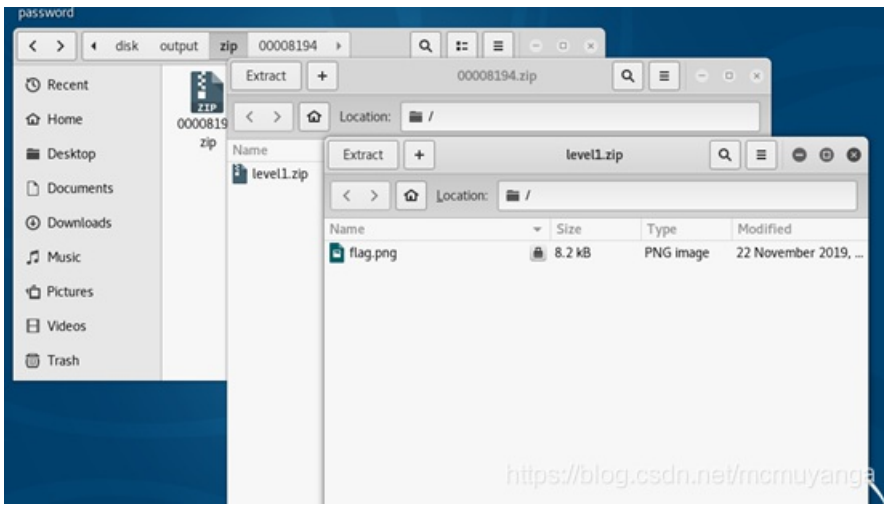
disk

放到kali先用binwalk检查一下

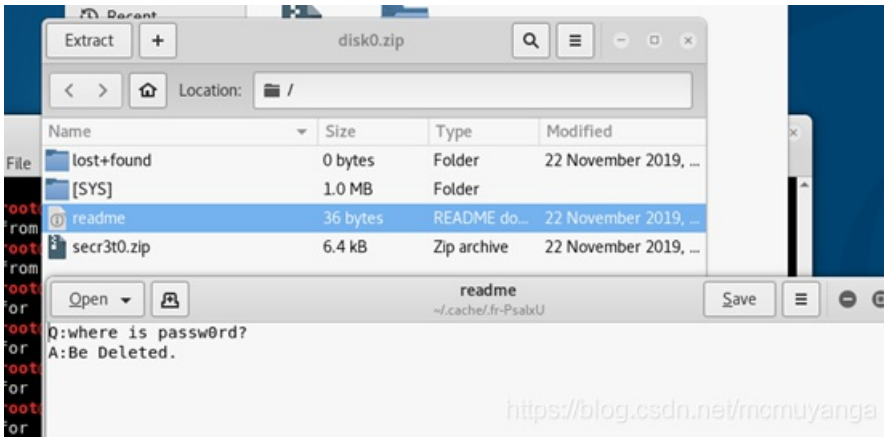
```
root@kali2-2017:~/Desktop/disk# binwalk disk0
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Linux EXT filesystem, rev 1.0, ext3 filesystem data
1160	0x488	Unix path: /root/MasterofZip-Hard/Generator/disk00
208896	0x33000	Linux EXT filesystem, rev 1.0, ext3 filesystem data
210056	0x33488	Unix path: /root/MasterofZip-Hard/Generator/disk00
224256	0x36C00	Linux EXT filesystem, rev 1.0, ext3 filesystem data
225416	0x37088	Unix path: /root/MasterofZip-Hard/Generator/test
231424	0x38800	Linux EXT filesystem, rev 1.0, ext3 filesystem data
232584	0x38C88	Unix path: /root/MasterofZip-Hard/Generator/test
253952	0x3E000	Linux EXT filesystem, rev 1.0, ext3 filesystem data
255112	0x3E488	Unix path: /root/MasterofZip-Hard/Generator/test
263168	0x40400	Linux EXT filesystem, rev 1.0, ext3 filesystem data
264328	0x40888	Unix path: /root/MasterofZip-Hard/Generator/test

有猫腻，用foremost恢复看一下



一个压缩包里还有一个压缩包，然后里面的压缩包里的flag.png上锁了，将原文件改为zip,发现了一个readme文件，打开提示密码被删了



将源文件改为.txt查看，发现了密码

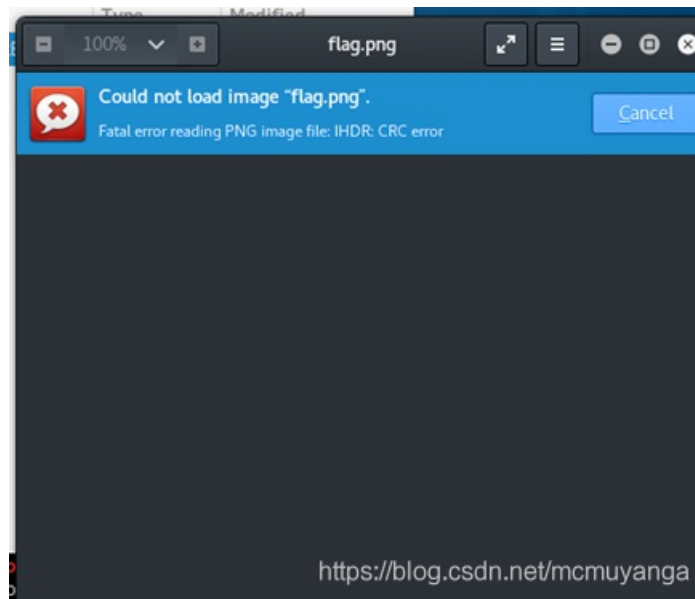
wd don0tgu355p@sswd

用这个密码去提取flag图片
Windows下打开是

???

<https://blog.csdn.net/mcmuyanga>

Kali打开的时候提示文件宽高存在问题



<https://blog.csdn.net/mcmuyanga>

用winhex修改图片高度，将之前的01改为02

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	%PNG IHDR
00000010	00	00	02	80	00	00	02	00	08	02	00	00	00	BA	B3	4B	e "K
00000020	B3	00	00	1F	A9	49	44	41	54	78	9C	ED	DD	79	90	14	' @IDATx!y
00000030	E5	DD	07	F0	59	59	8E	85	5D	40	14	44	34	78	A1	A8	áY 0VYž...j0 D4x;
00000040	78	10	F1	C4	78	20	DE	B7	41	43	D4	18	11	03	46	25	x nÄx P·ACÓ F%
00000050	A5	2B	F1	2C	DF	60	A2	46	8D	78	96	65	69	14	0F	34	W+ñ,B`cF x-ei 4
00000060	A5	26	F1	8C	89	C6	23	78	C4	F2	4E	79	21	20	A8	18	W!ñGhE#xñ0Nv! "

???

flag{3ae25f72880ac8ca7b1369a32e6c4edb34fdd886}

<https://blog.csdn.net/mcmuyanga>

flag{3ae25f72880ac8ca7b1369a32e6c4edb34fdd886}

题目包里还有一个js编码的题目，有师傅做出来了带带小弟呗