

CTF-MISC总结

原创

[Atkxor](#) 于 2020-04-20 21:40:24 发布 2864 收藏 51

分类专栏: [CTF](#) 文章标签: [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/qq_46150940/article/details/105620013

版权



[CTF 专栏收录该内容](#)

39 篇文章 2 订阅

订阅专栏

这里写自定义目录标题

1. 看图片属性

2. 隐藏在字节中

3. 补充头部

4. ZIP

ZIP文件格式

真加密

伪加密

5. LSB隐写

6. 双图

7. 音频分析

8. 压缩包暴力破解

9. 掩码攻击

10. 明文攻击

11. 图片高宽修改

计算图片正确宽高

爆破图片高度

JPG格式

PNG格式

12. 文件分离

foremost

binwalk

dd命令

13. 隐藏文件加密

F5隐写

steghide

outguess

JPHS

cloacked-pixel

14. 提取电话拨号音

15. Base64隐写

16. CRC碰撞

17. Base转图片

18. 盲水印

BlindWaterMark

频域盲水印

JAVA盲水印

19. gnuplot画图

20. 零宽字符加密

21. gaps拼图

22. 十六进制文件字节左右翻转

23. 十六进制文件头尾逆序

24. zlib解压

25. 爆破hash

26. 哈夫曼编码

27. RAR

RAR文件格式

RAR伪加密

RAR文件受损

29. USB流量分析

记录做题遇见的不同分类的杂项题，持续更新...

1.看图片属性

根据提示，找到图片属性,flag就出来了



2.隐藏在字节中

把图片放入010editor中，在文本中搜索 SL{

File Edit Search View Format Scripts Templates Tools Window Help

Startup 409649817-128-d85231de79516fdc5f9adf1c1a7ba94e.uc 67-191105111644Z6_爱奇艺.jpg

Edit As: Hex Run Script Run Template: JPG.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
8:6B80h:	9C	11	A3	E9	07	73	DB	3F	10	A4	C6	7C	69	95	A2	B9	E.é.sÛ?.Æ i·ç¹
8:6B90h:	F9	B8	BB	CF	49	EC	D5	5B	C9	A7	7B	69	05	BB	D3	84	ù.»ÏiïÖ[ÉS{i.»Ó,
8:6BA0h:	50	A8	73	64	56	BB	A0	1D	37	FE	69	19	68	2F	B6	B9	F`sdV» .7pi.h/¶¹
8:6BB0h:	86	70	6D	5C	E4	98	38	B6	B3	03	28	19	17	51	C8	77	tpm\ä~8¶³.(.QËw
8:6BC0h:	67	AA	50	3E	9A	29	AE	6A	07	BA	8E	96	E2	55	10	B6	gªP>š)øj.ºŽ-âU.¶¹
8:6BD0h:	5E	C4	16	D5	A0	EB	A1	AC	BB	CF	A0	34	58	AB	6F	FD	^Ä.Ö ë;~»Ï 4X«oy
8:6BE0h:	88	62	CF	03	4F	7C	AB	5C	B6	B0	B1	75	D9	83	A1	D8	^bÏ.O «\¶¹±uÛf;ø
8:6BF0h:	F9	FF	00	EF	7F	48	E1	01	B6	E5	82	0D	33	5C	FF	00	ùÿ.i.Há.¶¹á,.3\ÿ.
8:6C00h:	F9	63	07	BE	60	1D	FC	F7	2E	0B	B4	39	1E	B0	1A	AE	ùc.¾\..ù÷..º9.º.º
8:6C10h:	CC	87	DE	26	1A	CD	5D	9A	F1	55	53	4C	7B	69	61	6D	Ï+P&.Í}šñU{SL{iam
8:6C20h:	66	6C	61	67	7D	B3	53	23	AE	4C	5D	15	28	46	76	76	flag)ªS#¶¹. (Fvv
8:6C30h:	4E	4F	11	6F	04	65	CB	F1	B5	2D	1A	D0	F1	74	4F	01	NO.o.e¶¹p-.ĐñtO.
8:6C40h:	91	2E	5C	11	A3	FC	80	DC	AE	DA	7F	62	1D	1B	F2	DC	\. \.úeÛóú.b..òÛ
8:6C50h:	B9	70	78	61	CA	5D	F8	0D	23	AD	23	18	22	B6	3A	DA	¶¹xaË]ø.#-#. "¶¹:Û
8:6C60h:	A3	5C	11	C9	01	FA	35	E7	7C	E5	3A	EF	B4	16	B6	46	é\..É.ú5ç á:i'.¶¹F
8:6C70h:	FD	AC	22	F0	BA	53	49	3C	05	CA	52	46	B8	32	63	15	ÿ~"ø°SI<.ÉRF,2c.
8:6C80h:	00	BA	05	86	6F	34	AD	19	0B	34	EB	02	80	71	D2	52	.º.to4-..4ë.eqÒR
8:6C90h:	68	31	C1	0B	11	11	D1	1C	27	A9	29	C5	DB	B6	52	CF	h1Á...Ñ.'©)ÄÛ¶¹RÏ
8:6CA0h:	73	3D	61	89	61	20	0A	85	46	0D	B1	DC	D5	98	C9	DE	s=aªa ...F.±ÛÖ~ÉP

Template Results - JPG.bt

Name	Value	Start	Size	Color	Comment
> struct APP14 app14		14h	10h	Fg: Bg:	
> struct APP1 app1[0]		24h	107Ch	Fg: Bg:	
> struct APP12 app12		10A0h	13h	Fg: Bg:	
> struct APP1 app1[1]		10B3h	8DFh	Fg: Bg:	
> struct DQT dqt[0]		1992h	45h	Fg: Bg:	
> struct DQT dqt[1]		19D7h	45h	Fg: Bg:	
> struct SOFx sof2		1A1Ch	13h	Fg: Bg:	
> struct DHT dht[0]		1A2Fh	20h	Fg: Bg:	
> struct DHT dht[1]		1A4Fh	3Fh	Fg: Bg:	
> struct DHT dht[2]		1A8Eh	1Eh	Fg: Bg:	
> struct DHT dht[3]		1AACCh	76h	Fg: Bg:	
> struct DHT dht[4]		1B22h	4Fh	Fg: Bg:	
> struct DHT dht[5]		1B71h	2Eh	Fg: Bg:	

Output

Address	Value
Found 1 occurrences of 'SL{':	
86C1Ah	SL{

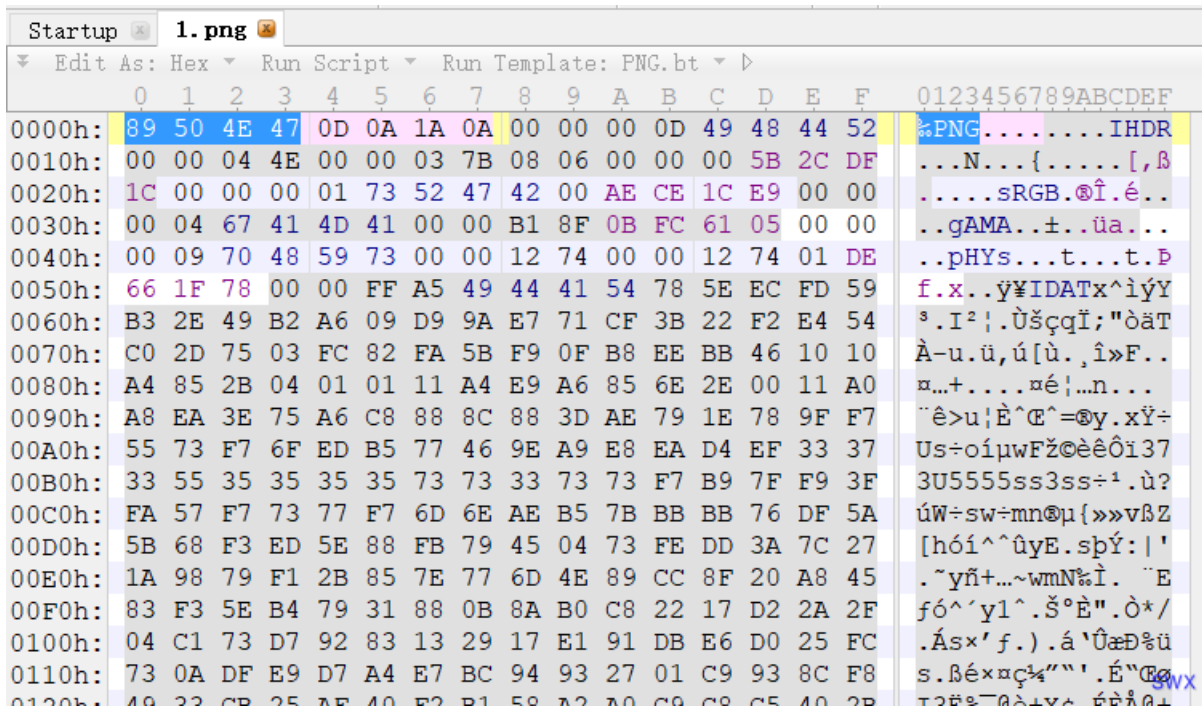
SWX

3.补充头部

常见文件文件头文件尾格式		
文件名	文件头	文件尾
JPEG (jpg)	FFD8FF	FF D9
PNG (png)	89504E47	AE 42 60 82
GIF (gif)	47494638	00 3B
ZIP Archive (zip)	504B0304	FF D9

常见文件文件头文件尾格式		
RAR Archive (rar)	50 4B	

将该图片以十六进制格式打开，发现该png图片缺失文件头，补充png文件头89504E47



4. ZIP

ZIP文件格式

一个 ZIP 文件由三个部分组成：

压缩源文件数据区+压缩源文件数据区+压缩源文件目录结束标志

压缩源文件数据区：

50 4B 03 04: 这是头文件标记 (0x04034b50)
 14 00: 解压文件所需 pkware 版本
 00 00: 全局方式位标记 (有无加密)
 08 00: 压缩方式
 20 9E: 最后修改文件时间
 66 4F: 最后修改文件日期
 F2 1B 0F 4A: CRC-32校验 (4A0F1BF2)
 0E 00 00 00: 压缩后尺寸
 0C 00 00 00: 未压缩尺寸
 08 00: 文件名长度
 00 00: 扩展记录长度
 66 6C 61 67 2E 74 78 74: 文件名 (不定长)
 4B CB 49 4C AF 36 34 32 36 31 35 AB 05 00: 文件flag.txt压缩后的数据

压缩源文件数据区：

50 4B 01 02: 目录中文件文件头标记(0x02014b50)
 1F 00: 压缩使用的 pkware 版本
 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)
 08 00: 压缩方式 20 9E: 最后修改文件时间 66 4F: 最后修改文件日期
 F2 1B 0F 4A: CRC-32校验 (4A0F1BF2)
 0E 00 00 00: 压缩后尺寸
 0C 00 00 00: 未压缩尺寸
 08 00: 文件名长度
 24 00: 扩展字段长度
 00 00: 文件注释长度
 00 00: 磁盘开始号
 00 00: 内部文件属性
 20 00 00 00: 外部文件属性
 00 00 00 00: 局部头部偏移量

压缩源文件目录结束标志:

50 4B 05 06: 目录结束标记
 00 00: 当前磁盘编号
 00 00: 目录区开始磁盘编号
 01 00: 本磁盘上纪录总数
 01 00: 目录区中纪录总数
 5A 00 00 00: 目录区尺寸大小
 34 00 00 00: 目录区对第一张磁盘的偏移量
 00 00: ZIP 文件注释长度

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 50 4B 03 04 14 00 00 00 08 00 70 02 01 4B B7 EF PK.....p..K·i
00000010 DC 83 03 00 00 00 01 00 00 00 05 00 00 00 30 2F Üf.....0.
00000020 74 78 74 33 04 00 50 4B 01 02 1F 00 14 00 00 00 txt3..PK.....
00000030 08 00 70 02 01 4B B7 EF DC 83 03 00 00 00 01 00 ..p..K·iÜf.....
00000040 00 00 05 00 24 00 00 00 00 00 00 00 20 00 00 00 ....$. ....
00000050 00 00 00 00 30 2E 74 78 74 0A 00 20 00 00 00 00 ....0.txt.. ....
00000060 00 01 00 18 00 2F EB D5 CB 18 0A D3 01 34 F1 41 .... /eÖË..Ó.4ñA
00000070 C9 18 0A D3 01 34 F1 41 C9 18 0A D3 01 50 4B 05 É..Ó.4ñAÉ..Ó.PK.
00000080 06 00 00 00 01 00 01 00 57 00 00 00 26 00 00 .....W...&..
00000090 00 00 00
```

加密注意点

https://blog.csdn.net/qq_46150940

区分

无加密

压缩源文件数据区的全局加密应当为00 00
且压缩源文件目录区的全局方式位标记应当为00 00

伪加密

压缩源文件数据区的全局加密应当为00 00
且压缩源文件目录区的全局方式位标记应当为09 00

真加密

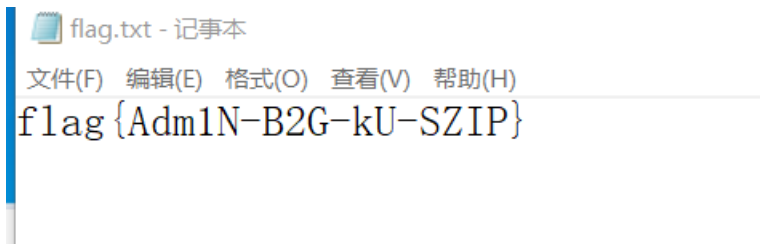
压缩源文件数据区的全局加密应当为09 00
且压缩源文件目录区的全局方式位标记应当为09 00

真加密

把这个zip文件拖入winhex中，由图中的信息可以知道是真加密

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	50	4B	03	04	14	00	09	00	08	00	50	A3	A5	4A	21	38	PK P%J!8
00000010	76	65	19	00	00	09	17	00	00	00	08	00	00	00	66	6C	ve fl
00000020	61	67	2E	74	78	74	4B	CB	49	4C	AF	76	4C	C9	35	F4	ag.txtK覆L痢L??
00000030	D3	75	32	72	D7	CD	0E	D5	0D	8E	F2	0C	A8	05	00	50	觔2r淄?床?P
00000040	4B	01	02	1F	00	14	00	09	00	08	00	50	A3	A5	4A	21	K P%J!
00000050	38	76	65	19	00	00	09	17	00	00	00	08	00	24	00	00	8ve \$
00000060	00	00	00	00	00	20	00	00	00	00	00	00	00	66	6C	61	fla
00000070	67	2E	74	78	74	0A	00	20	00	00	00	00	00	01	00	18	g.txt
00000080	00	0F	F5	04	D5	9A	C5	D2	01	46	1F	CB	8A	9A	C5	D2	?謂兵 F 葛毯?
00000090	01	46	1F	CB	8A	9A	C5	D2	01	50	4B	05	06	00	00	00	F 葛毯?PK
000000A0	00	01	00	01	00	5A	00	00	00	3F	00	00	00	00	00	00	Z ?

把09 00修改为00 00保存后即可得到flag



伪加密

伪加密，把全方位标记区的 01 00 或 09 00 改为 00 00

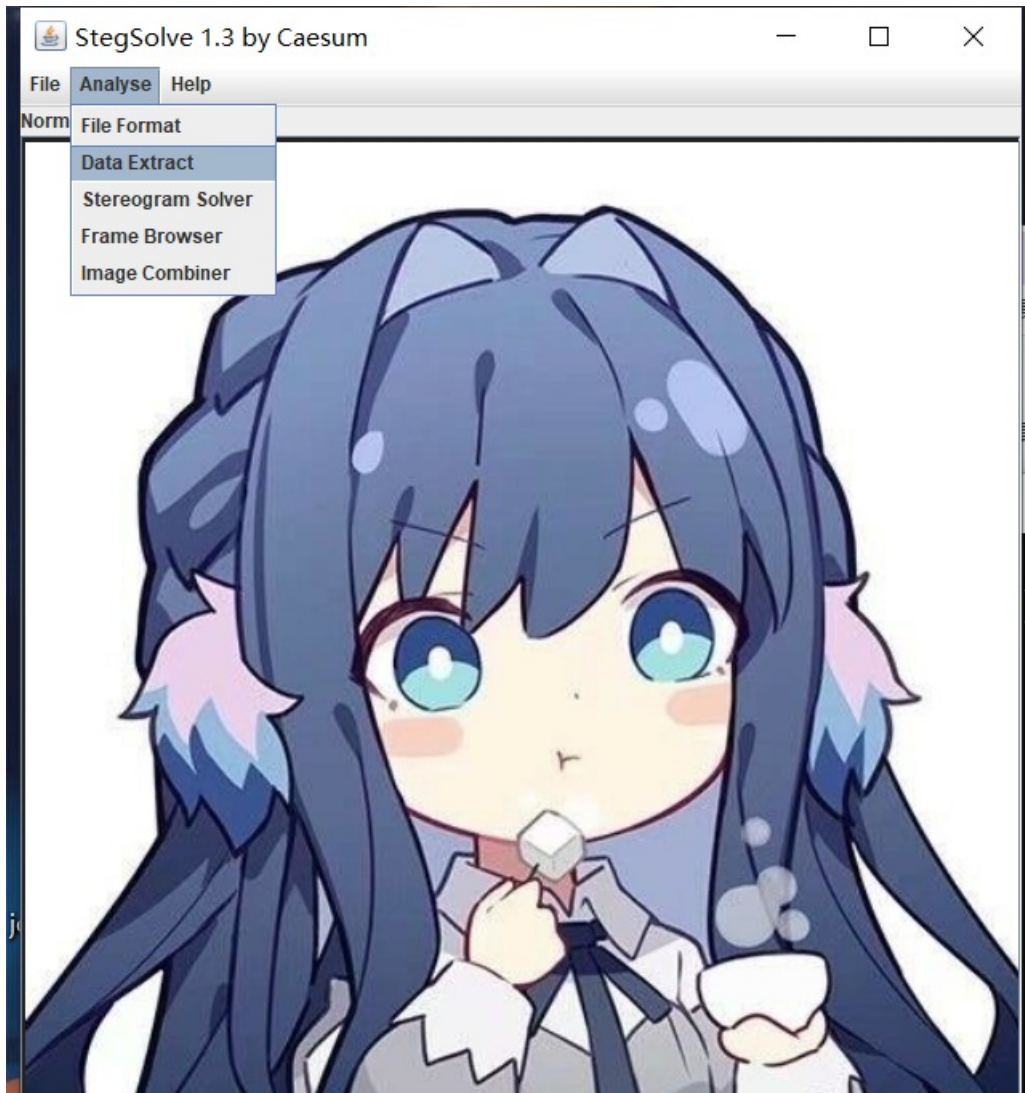
Startup secret.zip* x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
000h:	50	4B	03	04	14	00	09	00	08	00	98	A4	61	51	36	98	PK.....~paQ6~
010h:	49	60	88	00	00	00	71	01	00	00	0C	00	19	00	CE	D2	I`^...q.....ÎÒ
020h:	CB	A7	B2	BB	CB	A7	2E	74	78	74	75	70	15	00	01	43	Ë\$²»Ë\$.txtup...C
030h:	BF	FB	39	E6	88	91	E5	B8	85	E4	B8	8D	E5	B8	85	2E	¿09æ^`â...ä...ä...`
040h:	74	78	74	55	50	41	0E	C0	20	08	BB	9B	F8	07	EE	4B	txtUPA.À .»»ø.îK
050h:	79	01	F1	23	C6	FF	7F	63	AD	30	33	4D	D6	38	68	69	y.#Eÿ.c-03M08hi
060h:	F1	D1	B1	C4	89	71	EE	FC	62	19	FF	FD	AB	78	14	07	ñÑ±Ã%qîüb.ÿý«x..
070h:	30	C4	1A	44	2F	A6	47	6F	E2	4C	C3	D8	25	B6	AB	21	0Ä.D/!GoâLÄ0%¶«!
080h:	9C	30	8A	10	46	91	53	A6	43	F4	8D	28	03	CF	7A	6F	œ0\$.F'S!Cò.(.Ïzo
090h:	2A	1D	99	62	D8	CA	A1	0A	60	8F	02	90	4A	87	6F	50	*.™b0Ëj.`...J‡oP
0A0h:	D6	0F	BF	EC	99	A8	1A	13	49	DD	FE	F0	9F	E7	C6	E3	Ö.¿i™".IÿpðÿçÆä
0B0h:	1F	B5	81	86	6A	3F	E4	CA	BD	D5	BB	DC	4B	1D	6A	BD	.µ.†j?ãÊ½Ö»ÜK.j½
0C0h:	C5	35	D4	E3	4E	24	A9	12	B1	F1	02	50	4B	01	02	1F	Å50ãN\$@.±ñ.PK...
0D0h:	00	14	00	09	00	08	00	98	A4	61	51	36	98	49	60	88~paQ6~I`^
0E0h:	00	00	00	71	01	00	00	0C	00	3D	00	00	00	00	00	00	...q.....=.....
0F0h:	00	20	00	00	00	00	00	00	00	CE	D2	CB	A7	B2	BB	CBÎÒË\$²»Ë
100h:	A7	2E	74	78	74	0A	00	20	00	00	00	00	00	01	00	18	\$.txt..

emplate Results - ZIP.bt ↻

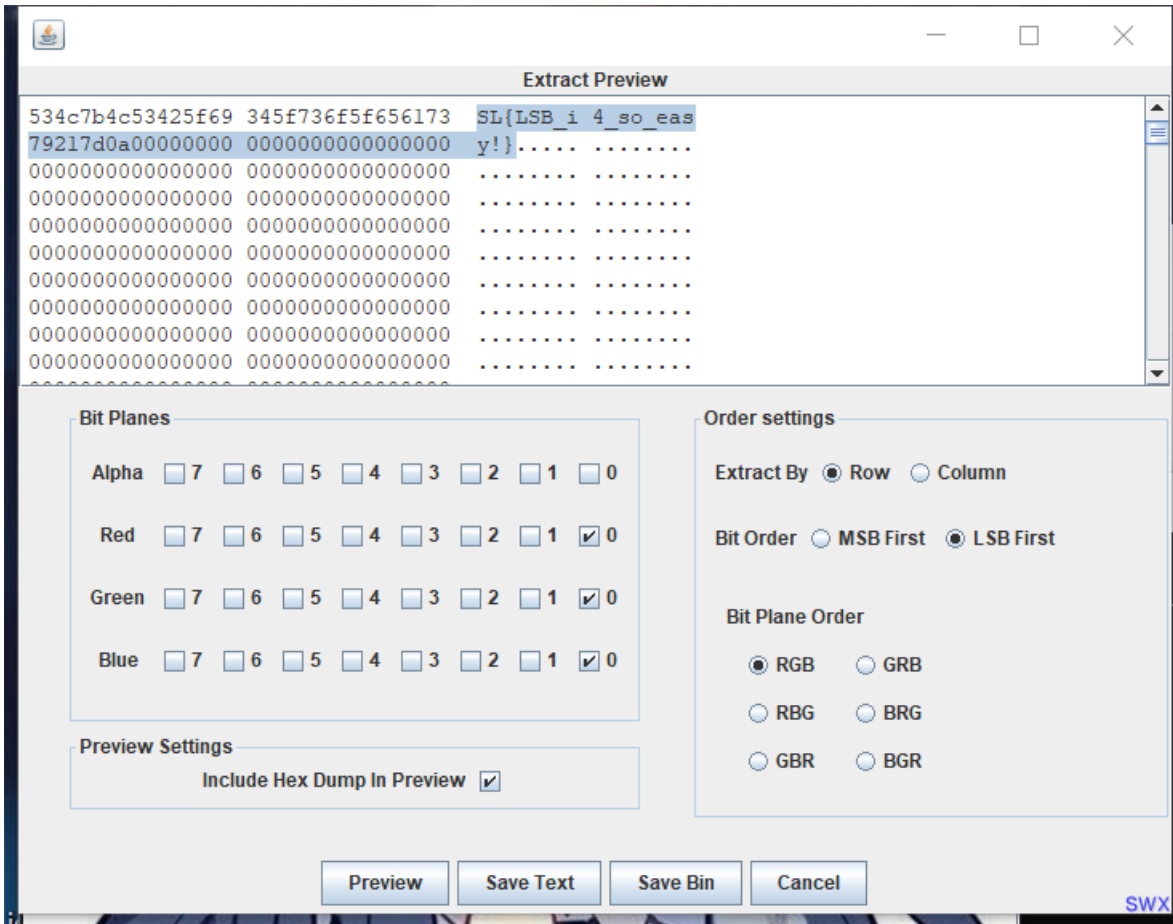
Name	Value	Start	Size	Color	Comment
struct ZIPFILERECD re...	ÎÒË\$²»Ë\$.txt	0h	CBh	Fg: Bg:	
struct ZIPDIRENTRY dirE...	ÎÒË\$²»Ë\$.txt	CBh	77h	Fg: Bg:	https://blog.csdn.net/qq_46150940

5. LSB隐写

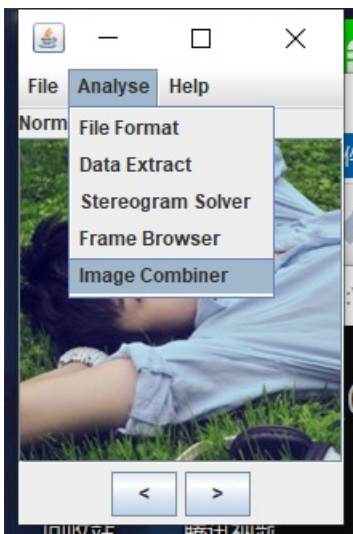




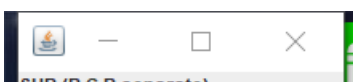
简单的LSB隐写，在最低有效位隐藏信息。可以使用Stegsolve提取。



6. 双图



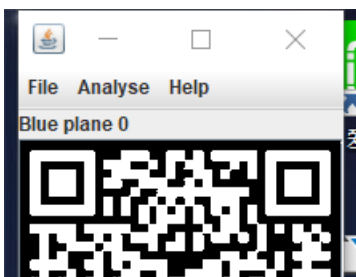
将两张图重叠后，进行异或，出现了疑似二维码的图片





点击Save，将该图片保存下来。

打开该图片继续进行异或，可以得到三张二维码图片





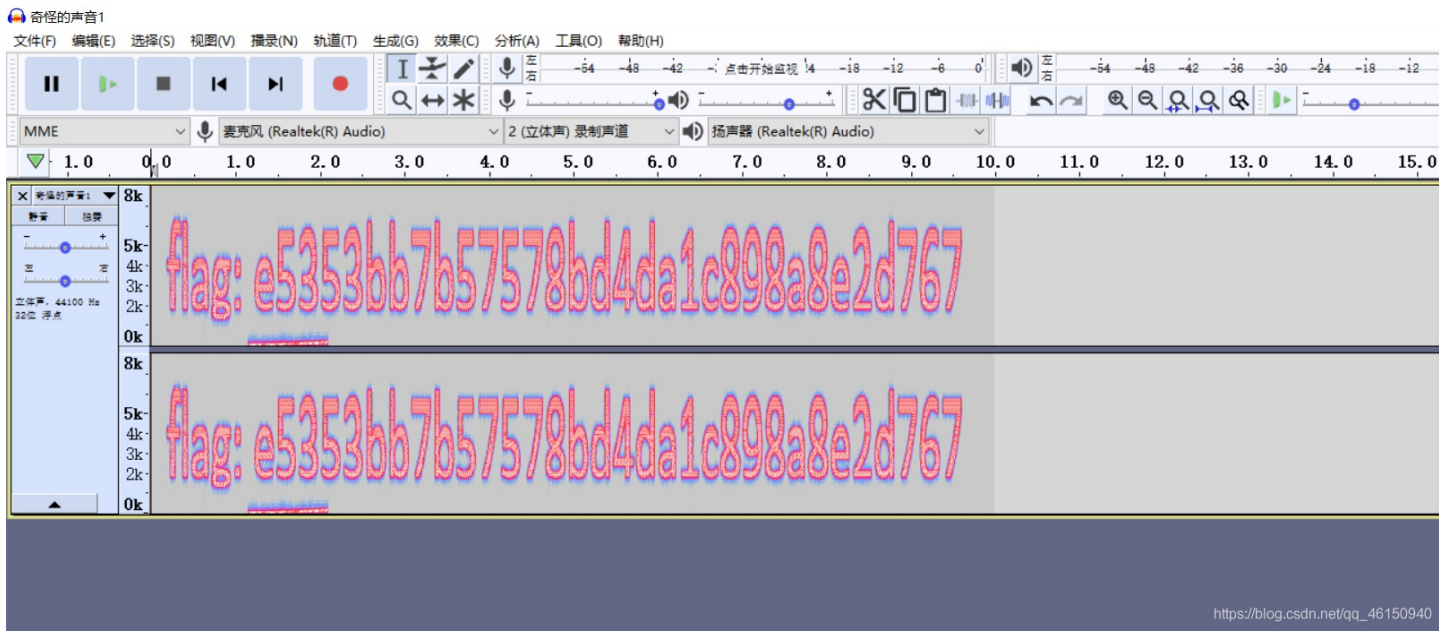
依次扫描这三张二维码可得到以下信息

第一张: DES
第二张: 6XaMMbM7
第三张: U2FsdGVkX18IBeATgMBe8NqjIqp65CxRjjMxXIiUxIjBnAODJQRkSLQ/+lHBsjpv1BwwEawMo1c=

根据该信息, 可以知道是DES加密, 进行DES解密

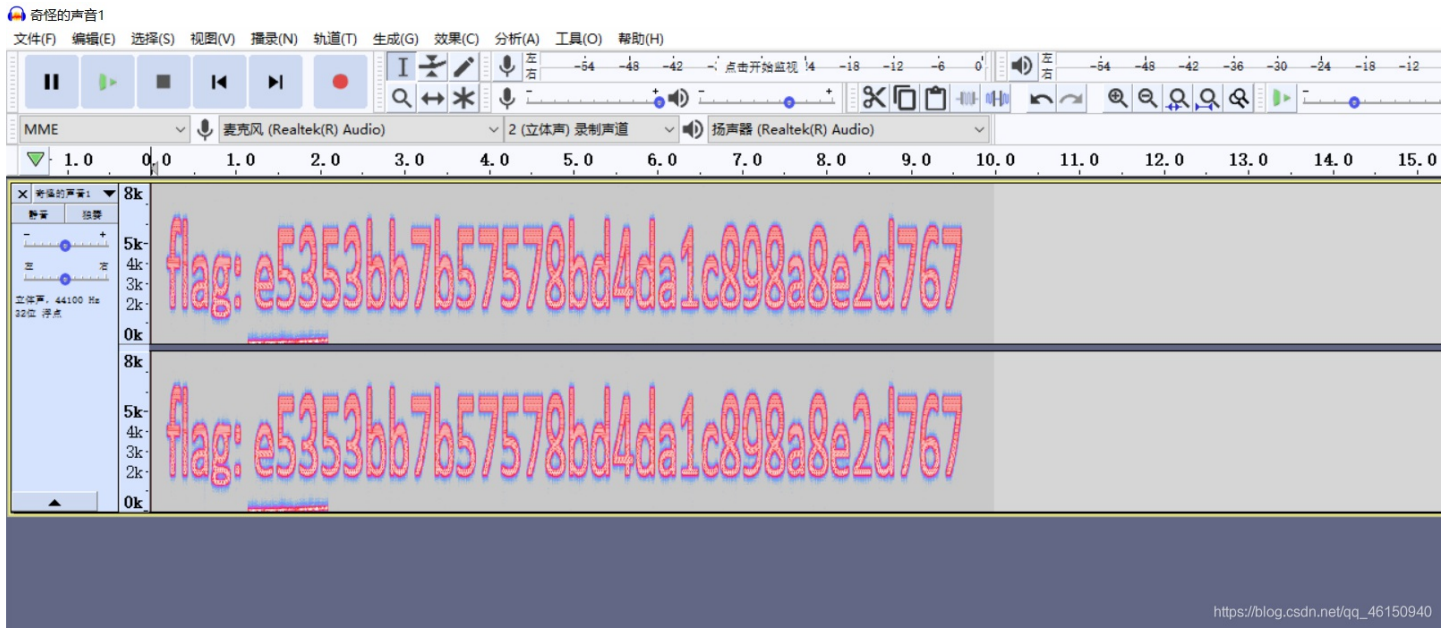
<code>ctf{67a166801342415a6da8f0dbac591974}</code>	6XaMMbM7 密码是可选项, 也就是可以不填。 <input type="button" value="解密"/> <input type="button" value="加密"/>	<code>U2FsdGVkX18IBeATgMBe8NqjIqp65CxRjjMxXIiUxIjBnAODJQRkSLQ/+lHBsjpv1BwwEawMo1c=</code>
--	---	---

查看频谱图



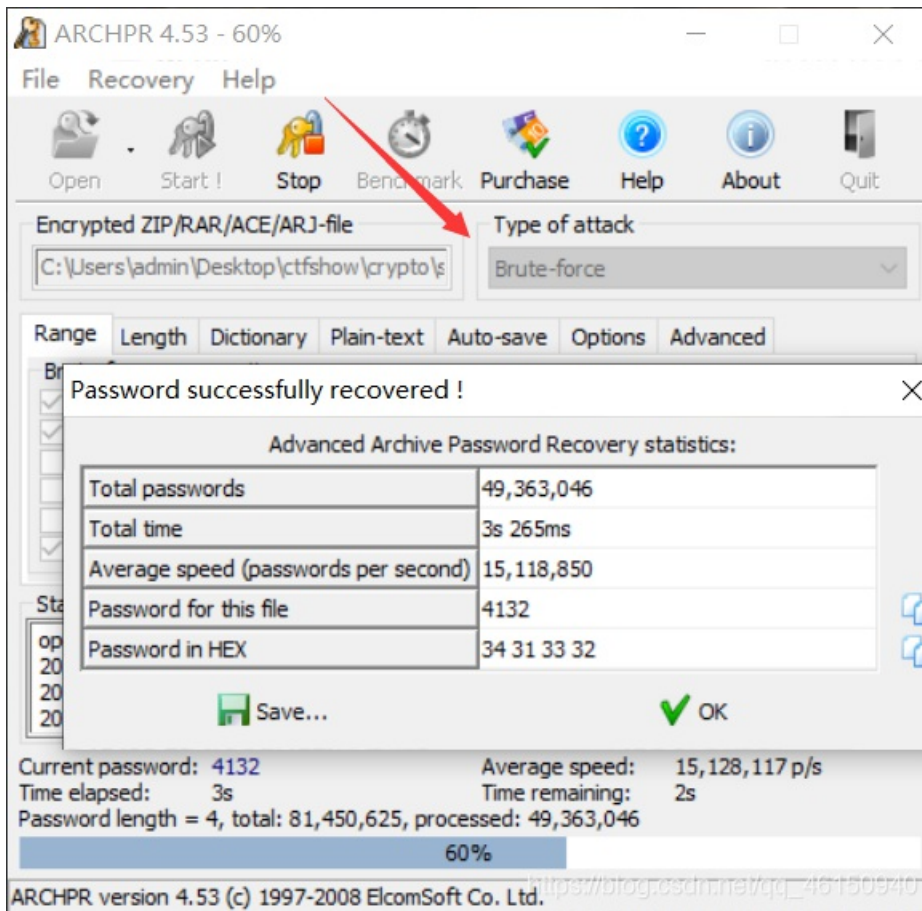
7. 音频分析

查看频谱图



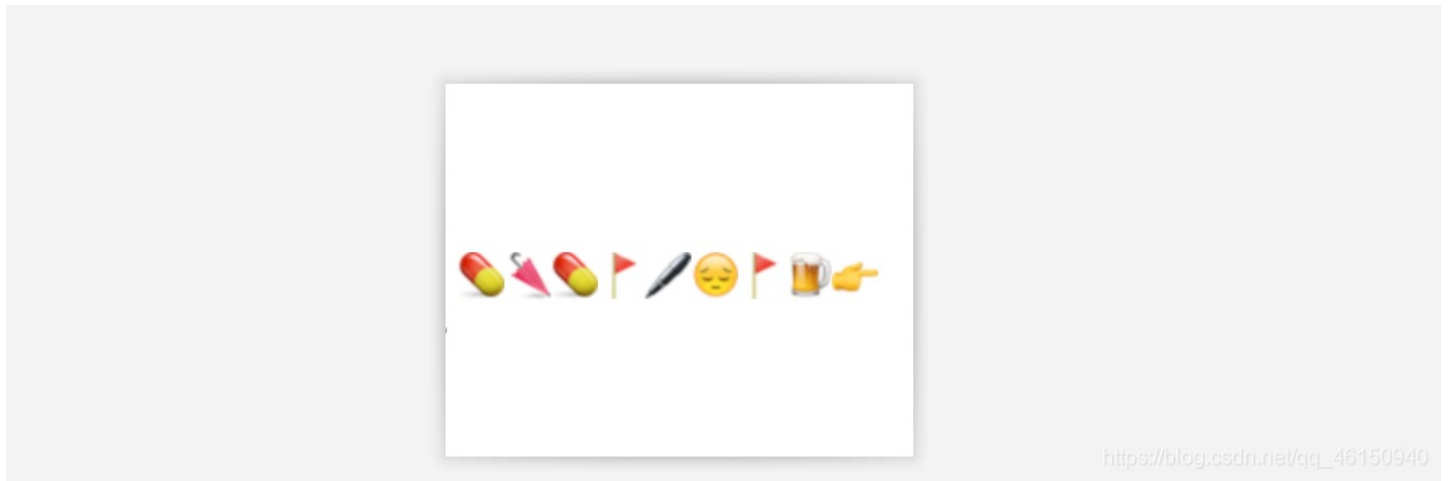
8. 压缩包暴力破解

使用ARCHPR工具，选择Brute-force模式，进行暴力破解



9. 掩码攻击

下载附件，是一个加密的压缩包和图片



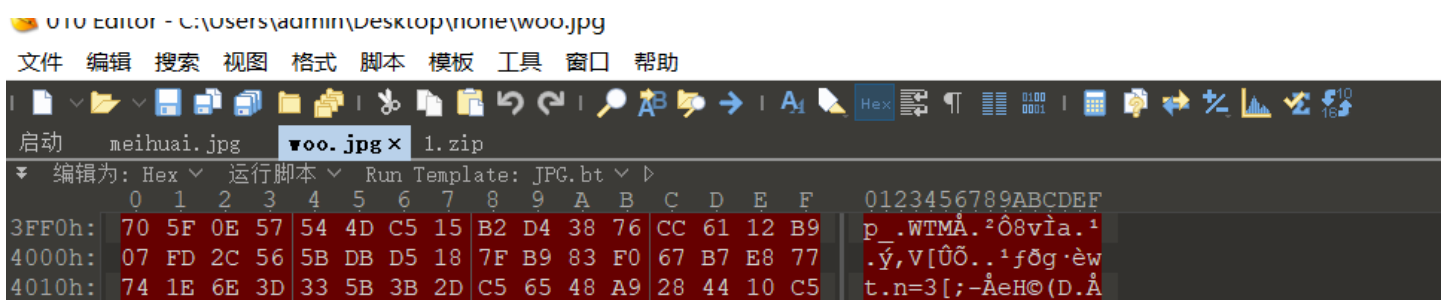
猜测图片里面的内容是压缩包密码，内容是 1317???7p9? ,进行 掩码攻击 ，结果提示错误，尝试了很多次，结果将掩码修改成 1317???7p9? 结果成功了。



10. 明文攻击

[ACTF新生赛2020]明文攻击

把图片拖入010Editor中，发现一个flag.txt




```

4020h: A6 FF 00 DD 8B 5F 03 E3 8B D3 08 1B E3 68 0A E1 |ÿ.Y<_.ä<Ö..äh.á
4030h: 63 F2 BB 85 85 71 85 71 84 92 58 58 58 58 49 2E |cò».....q...q,, 'XXXXXI.
4040h: 5B 7C 65 33 7A 5F 86 AD 6B 0B 47 27 8D 7D 3C C6 |[|e3z †-k.G'..)<E
4050h: 35 98 03 38 A5 F8 (9A) 46 0A 3E F1 3C 69 EF E5 8B |5~.8¥øšF.>ñ<iiá<
4060h: 23 76 97 2A E4 6E A6 D2 FF D9 32 32 32 32 32 32 |#v-*än!ÖÿÜ222222
4070h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 |2222222222222222
4080h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 |2222222222222222
4090h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 |2222222222222222
40A0h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 |2222222222222222
40B0h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 |2222222222222222
40C0h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 |2222222222222222
40D0h: 14 00 00 00 08 00 CB A2 82 4F D8 30 C5 B0 11 00 |.....Ëç,ø0Å°
40E0h: 00 00 11 00 00 00 08 00 00 00 66 6C 61 67 2E 74 |.....flag.t
40F0h: 78 74 2B C9 C8 2C 56 00 A2 92 8C 54 85 B4 9C C4 |xt+ÉÈ,V.ç'ËT...œÄ
4100h: 74 3D 00 50 4B 01 02 14 00 14 00 00 00 08 00 CB |t=.PK.....Ë
4110h: A2 82 4F D8 30 C5 B0 11 00 00 11 00 00 00 08 |ç,ø0Å°.....
4120h: 00 24 00 00 00 00 00 00 00 20 00 00 00 00 00 00 |.$.
4130h: 00 66 6C 61 67 2E 74 78 74 0A 00 20 00 00 00 00 |.flag.txt...
4140h: 00 01 00 18 00 01 02 2B 25 0B A9 D5 01 1D 7B 6F |.....+%.©Ö..{o
4150h: 54 0B A9 D5 01 79 58 D8 1C 0B A9 D5 01 50 4B 05 |T.©Ö.yXø..©Ö.PK.
4160h: 06 00 00 00 00 01 00 01 00 5A 00 00 00 37 00 00 |.....Z...7..
4170h: 00 00 00

```

https://blog.csdn.net/qq_46150940

复制完后加上 50 4B, 变成一个新的压缩包1.zip

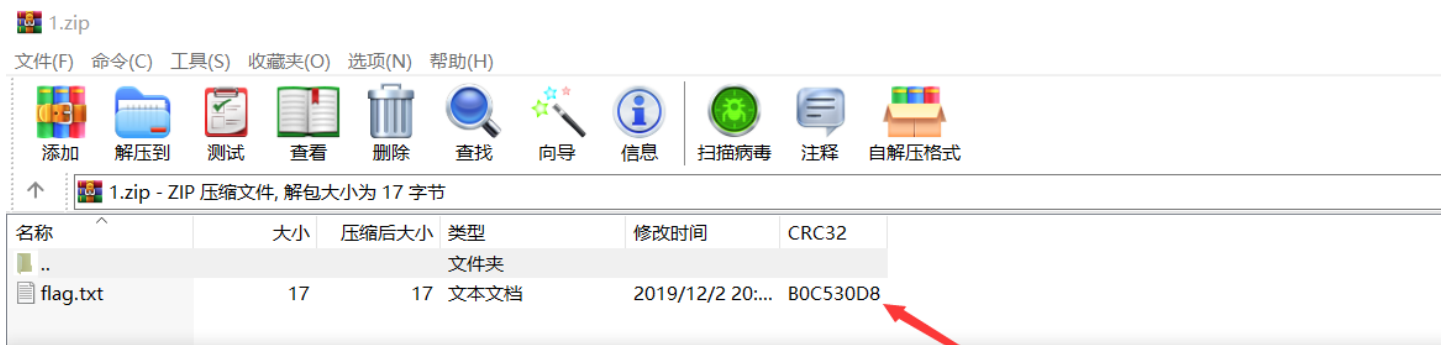
```

启动 meihuai.jpg 1.zip x 无标题2
编辑为: Hex 运行脚本 Run Template: JPG.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 50 4B 03 04 14 00 00 00 08 00 CB A2 82 4F D8 30 |PK.....Ëç,ø0Å°
0010h: C5 B0 11 00 00 00 11 00 00 00 08 00 00 00 66 6C |Å°.....fl
0020h: 61 67 2E 74 78 74 2B C9 C8 2C 56 00 A2 92 8C 54 |ag.txt+ÉÈ,V.ç'ËT
0030h: 85 B4 9C C4 74 3D 00 50 4B 01 02 14 00 14 00 00 |...œÄt=.PK.....
0040h: 00 08 00 CB A2 82 4F D8 30 C5 B0 11 00 00 00 11 |...Ëç,ø0Å°.....
0050h: 00 00 00 08 00 24 00 00 00 00 00 00 00 20 00 00 |.....$.
0060h: 00 00 00 00 00 66 6C 61 67 2E 74 78 74 0A 00 20 |.....flag.txt..
0070h: 00 00 00 00 00 01 00 18 00 01 02 2B (25) 0B A9 D5 |.....+%.©Ö
0080h: 01 1D 7B 6F 54 0B A9 D5 01 79 58 D8 1C 0B A9 D5 |..{oT.©Ö.yXø..©Ö
0090h: 01 50 4B 05 06 00 00 00 00 01 00 01 00 5A 00 00 |.PK.....Z...
00A0h: 00 37 00 00 00 00 00 00 |.7.....

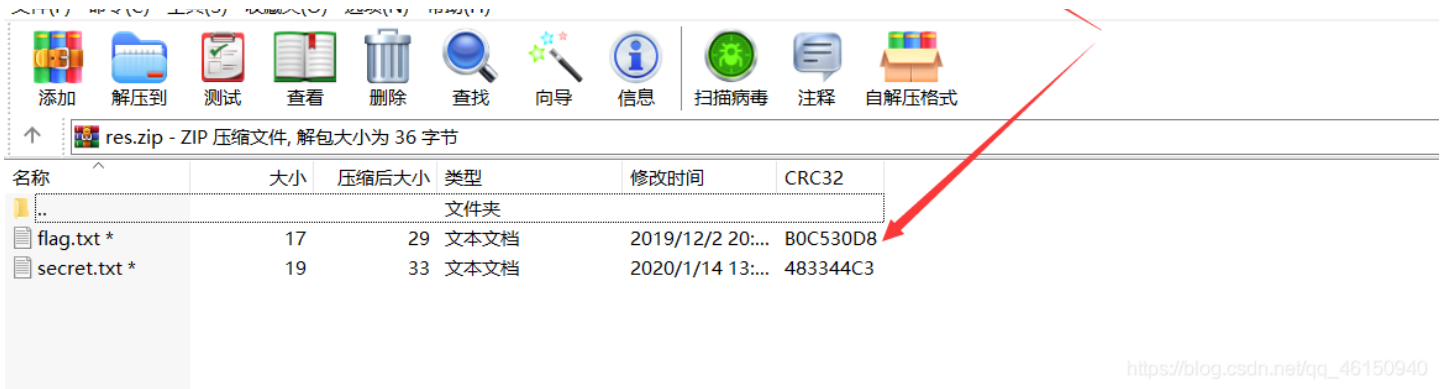
```

https://blog.csdn.net/qq_46150940

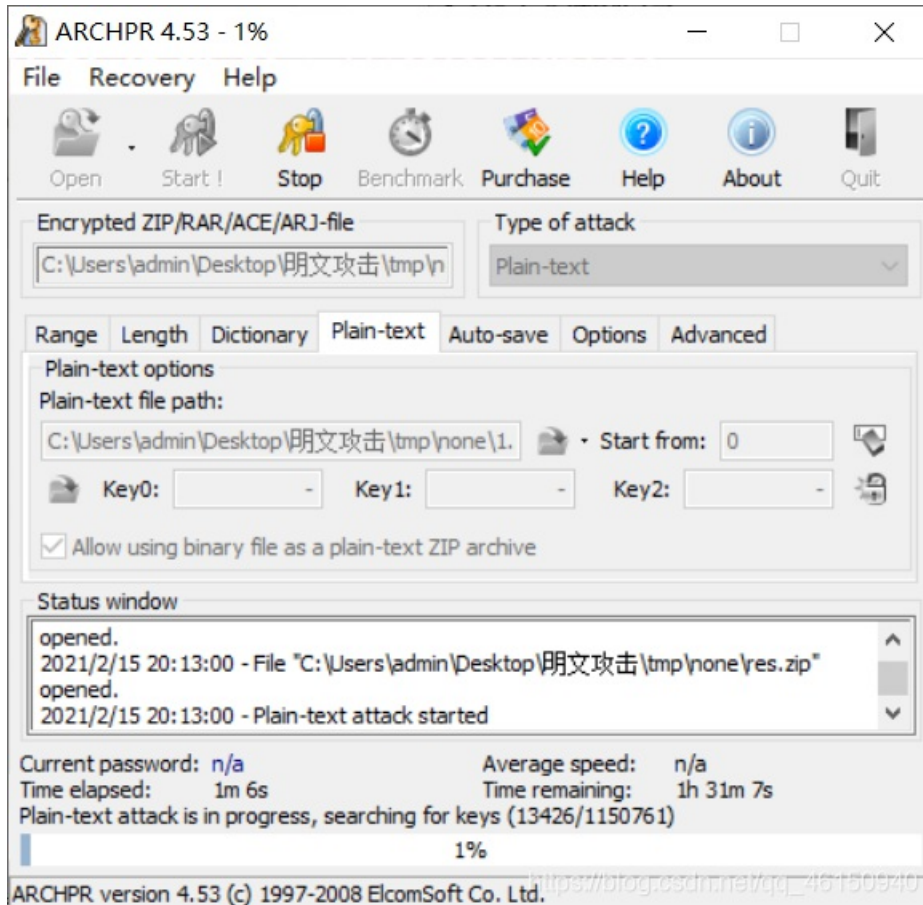
压缩包1.zip里面的flag.txt与加密压缩包里面的flag.txt的CRC值一样



res.zip
文件(F) 命令(C) 工具(S) 收藏夹(O) 选项(N) 帮助(H)



进行明文攻击



11. 图片高宽修改

计算图片正确宽高

查看图片crc值

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	%PNG.....IHDR
0010h:	00	00	05	56	00	00	05	B7	08	06	00	00	00	AB	21	2A	...V...<!*>
0020h:	35	00	00	00	16	74	45	58	74	41	72	74	69	73	74	00	5...tEXtArtist.
0030h:	23	41	5F	6B	33	79	5F	31	6E	5F	65	78	69	66	23	7F	#A_k3y_l_n_exif#.
0040h:	FA	C3	E3	00	00	02	3E	69	54	58	74	58	4D	4C	3A	63	úÃã...>iTXtXML:c
0050h:	6F	6D	2E	61	64	6F	62	65	2E	78	6D	70	00	00	00	00	om.adobe.xmp...
0060h:	00	3C	3F	78	70	61	63	6B	65	74	20	62	65	67	69	6E	.<?xpacket begin
0070h:	3D	27	EF	BB	BF	27	20	69	64	3D	27	57	35	4D	30	4D	= 'i»¿' id='W5M0M
0080h:	70	43	65	68	69	48	7A	72	65	53	7A	4E	54	63	7A	6B	pCehiHzreSzNTczk
0090h:	63	39	64	27	3F	3E	0A	3C	78	3A	78	6D	70	6D	65	74	c9d'?'>.<x:xmpmet
00A0h:	61	20	78	6D	6C	6E	73	3A	78	3D	27	61	64	6F	62	65	a xmlns:x='adobe
00B0h:	3A	6E	73	3A	6D	65	74	61	2F	27	20	78	3A	78	6D	70	:ns:meta/' x:xmp
00C0h:	74	6B	3D	27	49	6D	61	67	65	3A	3A	45	78	69	66	54	tk='Image::ExifT
00D0h:	6F	6F	6C	20	31	31	2E	39	38	27	3E	0A	3C	72	64	66	ool 11.98'>.<rdf
00E0h:	3A	52	44	46	20	78	6D	6C	6E	73	3A	72	64	66	3D	27	:RDF xmlns:rdf='
00F0h:	68	74	74	70	3A	2F	2F	77	77	77	2E	77	33	2E	6F	72	http://www.w3.or
0100h:	67	2F	31	39	39	39	2F	30	32	2F	32	32	2D	72	64	66	g/1999/02/22-rdf
0110h:	2D	73	79	6E	74	61	78	2D	6E	73	23	27	3E	0A	0A	20	-syntax-ns#'>..
0120h:	3C	72	64	66	3A	44	65	73	63	72	69	70	74	69	6F	6E	<rdf:Description
0130h:	20	72	64	66	3A	61	62	6F	75	74	3D	27	27	0A	20	20	rdf:about=''
0140h:	78	6D	6C	6E	73	3A	70	68	6F	74	6F	73	68	6F	70	3D	xmlns:photoshop=
0150h:	27	68	74	74	70	3A	2F	2F	6E	73	2E	61	64	6F	62	65	'http://ns.adobe
0160h:	2E	63	6F	6D	2F	70	68	6F	74	6F	73	68	6F	70	2F	31	.com/photoshop/1
0170h:	2E	30	2F	27	3E	0A	20	3C	70	68	6F	74	6F	73	68	6F	.0/'>.<photosho
0180h:	70	3A	44	6F	63	75	6D	65	6E	74	41	6E	63	65	73	74	p:DocumentAncest

https://blog.csdn.net/qq_46150940

所以CRC值为0xab212a35，计算图片正确宽高

```
import struct
import binascii
import os

m = open("ctfshow.png", "rb").read()
k=0
for i in range(5000):
    if k==1:
        break
    for j in range(5000):
        c = m[12:16] + struct.pack('>i', i) + struct.pack('>i', j)+m[24:29]
        crc = binascii.crc32(c) & 0xffffffff
        if crc == 0xab212a35:
            k = 1
            print(hex(i),hex(j))
            break
```

运行脚本得到

0x556 0x5b7

爆破图片高度

```

import os
import binascii
import struct
misc = open("ctfshow.png", "rb").read()
#print(misc[0x0c:0x0f+1])
# 爆破高

crc32_bytes = misc[0x1d:0x20+1]# 读出bytes
crc32_hex_eval = eval('0x' + crc32_bytes.hex())#bytes串 -> hex串 -> 值
print(crc32_hex_eval)
for i in range(4096):
    data = misc[0x0c:0x0f+1] + misc[0x10:0x13+1] + struct.pack('>i',i)+ misc[0x18:0x1c+1] #IHDR数据
    crc32 = binascii.crc32(data) & 0xffffffff
    if crc32 == crc32_hex_eval : #IHDR块的crc32值
        print(i)
        print("height_hex:"+ hex(i))

```

运行脚本得到

```

2871077429
1463
height_hex:0x5b7

```

JPG格式

打开图片，搜索 `FFC0`，在001180 后面4个字节分别是是高宽，把00 80改成02 80

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0h: FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	60	ÿøÿà..JFIF.....`
0h: 00	60	00	00	FF	DB	00	43	00	03	02	02	03	02	02	03	..ÿÛ.C.....
0h: 03	03	03	04	03	03	04	05	08	05	05	04	04	05	0A	07
0h: 07	06	08	0C	0A	0C	0C	0B	0A	0B	0B	0D	0E	12	10	0D
0h: 0E	11	0E	0B	0B	10	16	10	11	13	14	15	15	15	0C	0F
0h: 17	18	16	14	18	12	14	15	14	FF	DB	00	43	01	03	04ÿÛ.C...
0h: 04	05	04	05	09	05	05	09	14	0D	0B	0D	14	14	14	14
0h: 14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
0h: 14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
0h: 14	14	14	14	14	14	14	14	14	14	14	14	14	14	FF	C0ÿÀ
0h: 00	11	08	00	88	02	80	03	01	22	00	02	11	01	03	11	...^€. ".
0h: 01	FF	C4	00	1F	00	00	01	05	01	01	01	01	01	01	00	ÿÄ.....
0h: 00	00	00	00	00	00	00	01	02	03	04	05	06	07	08	09
0h: 0A	0B	FF	C4	00	B5	10	00	02	01	03	03	02	04	03	05	..ÿÄ.µ.....
0h: 05	04	04	00	00	01	7D	01	02	03	00	04	11	05	12	21}
0h: 31	41	06	13	51	61	07	22	71	14	32	81	91	A1	08	23	1A. .Qa. "q.2. ' j. #
0h: 42	B1	C1	15	52	D1	F0	24	62	72	82		09	0A	16	17	B±Ä.RÑó\$3br, ...
0h: 18	19	高	A	25	26	27	28	29	2A	34	35	36	37	38	3A	...%&'()*456789:
0h: 43	44	45	46	47	48	49	4A	53	54	55	56	57	58	59	5A	CDEFGHIJSTUVWXYZ
0h: 63	64	65	66	67	68	69	6A	73	74	75	76	77	78	79	7A	cdefghijstuvwxyz
0h: 83	84	85	86	87	88	89	8A	92	93	94	95	96	97	98	99	f...îî^%S'""•_~™

PNG格式

第二行前四位表示图片的宽，后四位表示图片的高。

十六进制的045F为十进制的1119，十六进制的200为十进制的512
发现像素大小与属性一致，于是把高跟宽改成同一大小保存数据。

Startup 修改图片宽高1.png																	
Edit As: Hex Run Script Run Template: PNG.bt																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	%PNG.....IHDR
0010h:	00	00	04	5F	00	00	04	5F	08	02	00	00	00	BE	0C	A5 ¼.¥
0020h:	9B	00	00	20	00	49	44	41	54	78	01	EC	BD	F9	76	24	>...IDATx.i¼ùv\$
0030h:	39	96	E6	47	D2	49	3A	F7	88	C8	8C	CA	A5	96	AC	AE	9-æGÒI:÷^ÈÈ¥--@
0040h:	E9	6D	BA	CE	51	4B	7A	04	3D	87	FE	D7	13	4B	EA	51	ém°îQKz.=+p×.KêQ
0050h:	1F	A9	A7	6B	E6	74	F7	F4	92	99	91	8C	E0	E6	4E	27	.@škat÷ô'™`EàæN'
0060h:	F5	FB	BE	0B	C0	60	E6	66	BE	D1	C9	88	AC	0A	04	C3	õù¼.À`æf¼ÑÉ^-.Ä
0070h:	1C	CB	C5	DD	70	01	C3	35	C0	60	BB	BF	FB	CB	FF	65	.ÈÁÝp.Ã5À`»¿ûËËye
0080h:	72	F3	61	77	67	E7	71	67	6F	77	F7	71	67	47	FF	1F	róawgçqggow÷qgGÿ.
0090h:	1F	77	C8	D9	DD	25	93	C0	EF	CE	83	23	7B	94	50	FE	.wÈÛÝ%`ÀiÎf#{"Pp
00A0h:	B8	BB	BB	BB	07	0C	B0	06	20	B2	B3	FB	20	B0	54	8D	,»»»...°.²³ù °T.
00B0h:	9F	C7	87	47	32	F9	EF	38	B8	85	34	63	16	B2	08	BB	ÝÇ+G2ùì8, ...4c.².»
00C0h:	7B	A2	0C	74	02	DE	11	09	8A	12	AE	14	13	0C	D9	C0	{ç.t.Đ...Š.®...ÙÀ
00D0h:	40	77	06	A7	C4	A9	18	41	D5	15	7B	DC	73	2D	45	55	@w.sÄ®.AÕ.{Ûs-EU
00E0h:	DB	E1	11	C6	61	56	6C	C0	75	FE	51	11	55	CC	B0	EB	Ûá.ÆavlÀupQ.Uì°è
00F0h:	AA	00	FE	2D	7E	D4	23	03	29	85	08	04	81	9F	5F	31	ª.p~÷Ô#.)....ÿ_1
0100h:	FB	10	EC	25	79	61	D7	15	04	9F	6A	BA	2A	0C	22	12	û.i%ya×..ÿj°*.".
0110h:	1C	A1	A2	02	21	80	AC	53	03	0B	7F	AA	E7	5F	25	F5	.jç.!€-š...ªç_°õ
0120h:	3F	32	A5	12	97	46	96	B2	89	B9	69	02	65	82	55	81	?2¥.-F-²%¹i.e,šwx

12. 文件分离

foremost

使用foremost命令 `foremost 1.jpg`

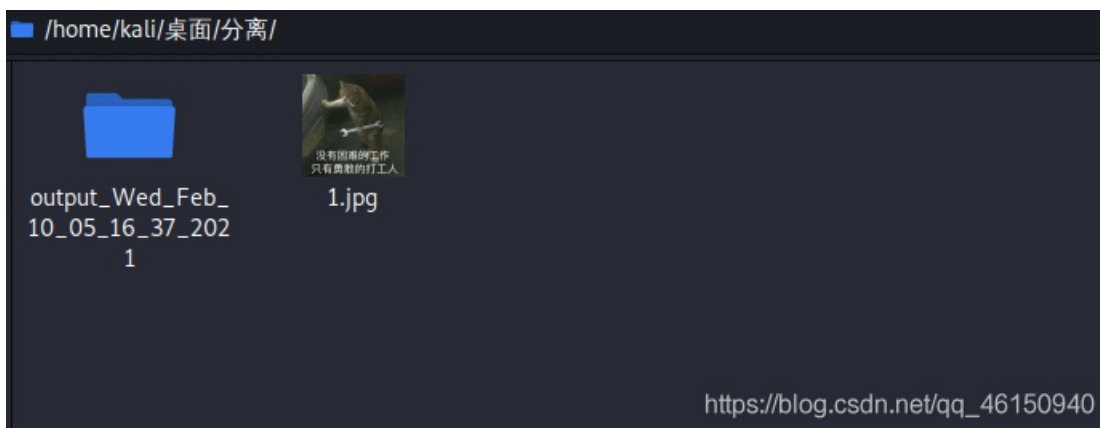
```
kali@kali: ~/桌面/分离
└─(kali@kali)-[~/桌面/分离]
└─$ foremost 1.jpg
Processing: 1.jpg
|*|
└─(kali@kali)-[~/桌面/分离]
└─$ █
```

https://blog.csdn.net/qq_46150940

得到



带 `-T` 参数, `foremost -T 1.jpg` 会生成以时间为名的文件夹名字, 可以避免重复



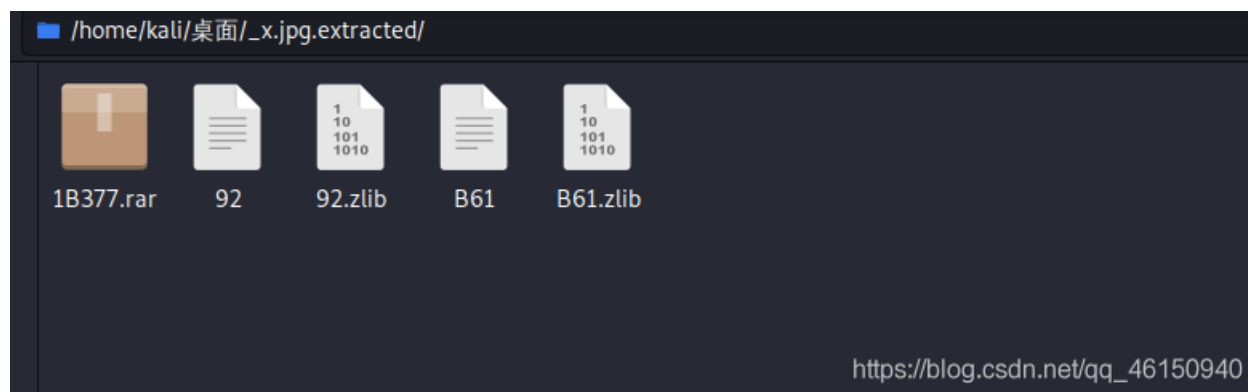
binwalk

binwalk一下，发现有隐藏文件

```
(kali㉿kali)-[~/桌面]
└─$ binwalk 小火龙.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 658 x 658, 8-bit/color RGBA, non-interlaced
146	0x92	Zlib compressed data, best compression
2913	0xB61	Zlib compressed data, best compression
111479	0x1B377	RAR archive data, version 5.x

分离文件，分离出1B377.rar



dd命令

binwalk图片后得到的信息，jpeg文件是从271007开始

```
└─$ dd if=binwalk,foremost,dd.png of=out.jpg bs=1 skip=271007
记录了29053+0 的读入
记录了29053+0 的写出
29053字节 (29 kB, 28 KiB) 已复制, 0.175658 s, 165 kB/s
```

可以得到png图片

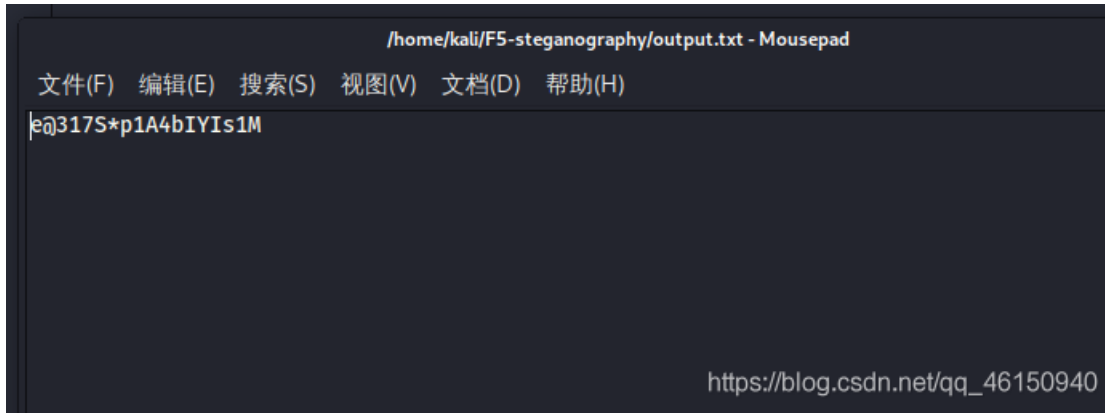
13. 隐藏文件加密

F5隐写

需要java环境，进入F5-steganography/目录下

```
java Extract Matryoshka.jpg -p '!LyJJ9bi&M7E72*JyD'
```

生成out.txt



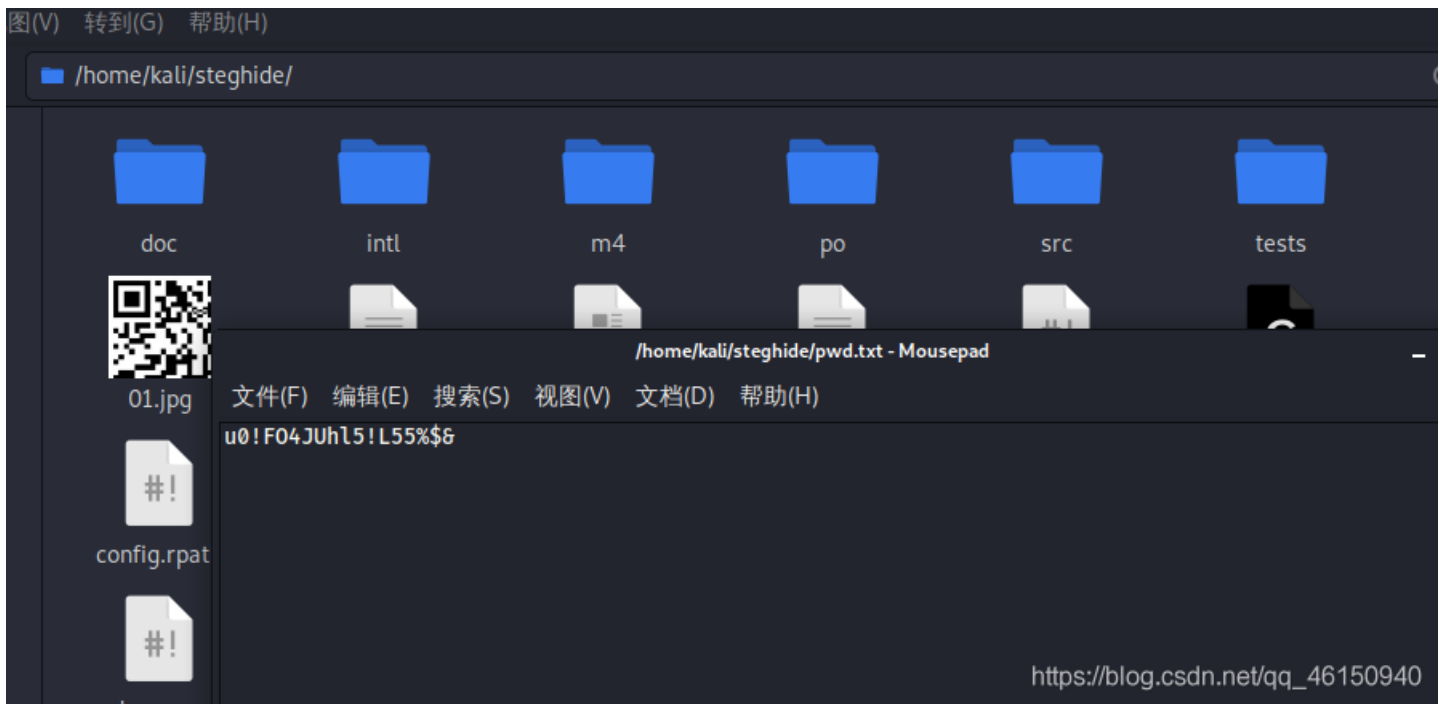
```
/home/kali/F5-steganography/output.txt - Mousepad
文件(F) 编辑(E) 搜索(S) 视图(V) 文档(D) 帮助(H)
e@317S*p1A4bIYIs1M
https://blog.csdn.net/qq_46150940
```

steghide

进入目录下

```
steghide extract -sf 01.jpg -p A7SL9nHRJXLh@$EbE8
```

生成pwd.txt



```
图(V) 转到(G) 帮助(H)
/home/kali/steghide/
doc intl m4 po src tests
01.jpg
config.rpat
/home/kali/steghide/pwd.txt - Mousepad
文件(F) 编辑(E) 搜索(S) 视图(V) 文档(D) 帮助(H)
u0!F04JUhl5!L55%$8
https://blog.csdn.net/qq_46150940
```

outguess

进入目录下，需要新建一个txt文档

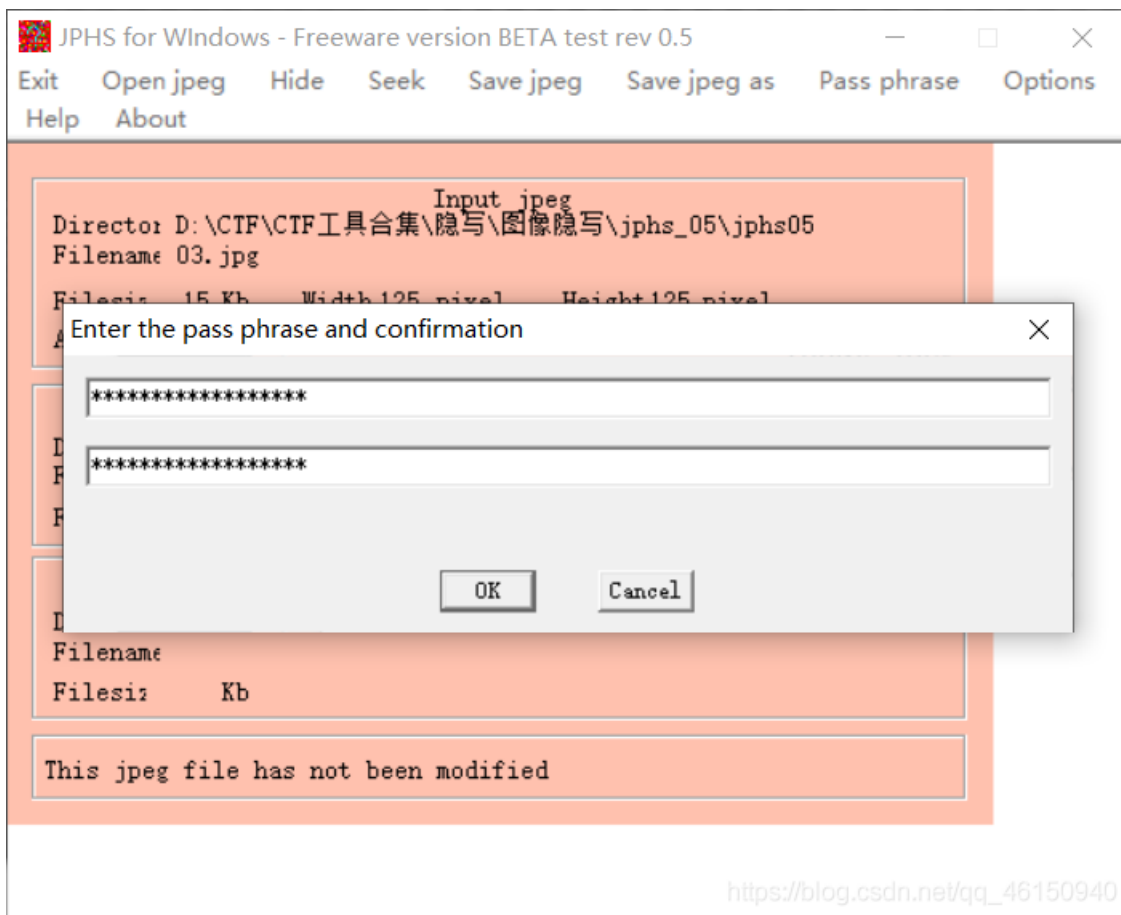
```
outguess -k "z0GFieYAee%gdf0%1F" -r 02.jpg flag.txt
```



JPHS

参考文章: <https://blog.csdn.net/drondong/article/details/79326385>

只能打开jpg格式的图片, 输入两次相同密码, 提取隐藏文件



cloacked-pixel

cloacked-pixel安装

```
git clone https://codechina.csdn.net/mirrors/cyberinc/cloacked-pixel.git
```

安装需要用到的库


```
pip install numpy-1.16.6+mkl-cp27-cp27m-win_amd64.whl
pip install matplotlib
pip install pillow
#卸载陈旧版本的包
pip uninstall crypto
pip uninstall pycrypto
#安装新版本
pip install pycryptodome
```

使用

```
E:\cloacked-pixel>python lsb.py extract lsb (123456) .png flag.txt 123456
[+] Image size: 640x640 pixels.
[+] Written extracted data to flag.txt.
```



https://blog.csdn.net/qq_46150940

14. 提取电话拨号音

工具下载地址: <https://www.cr173.com/soft/17040.html>


```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19041.685]
(c) 2020 Microsoft Corporation. 保留所有权利。

D:\CTF\MISC\音频分析\dtmf2num>dtmf2num.exe 拨号音.wav

DTMF2NUM 0.1c
by Luigi Aurieremma
e-mail: aluigi@autistici.org
web: aluigi.org

- open 拨号音.wav
  wave size      35200
  format tag     1
  channels:      1
  samples/sec:   8000
  avg/bytes/sec: 16000
  block align:   2
  bits:          16
  samples:       17600
  bias adjust:   -3
  volume peaks:  -29471 29471
  normalize:     3296

- MF numbers:    74
- DTMF numbers: 15975384265
```



https://blog.csdn.net/qq_46150940

15. Base64隐写

[GXCYCTF2019]SXMgdGhpcyBiYXNIPw=

打开文件是多行base64密文

```
Q2V0dGUgbnVpdCwK
SW50ZW5hYmx1IGluc29tbm1lLAp=
TGEgZm9saWUgbWUgZ3VldHR1LAo=
SmUgc3VpcyBjZSBxdWUgamUgZnVpcwP=
SmUgc3ViaXMsCt==
Q2V0dGUgY2Fjb3Bob25pZSwK
UXVpIG11IHJjaWUgbGEgdOmUmnR1LAp=
QXNzb21tYW50ZSB0YXJtb25pZSwK
RWxsZSBtZSBkaXQsCo==
VHUgcGFpZXJhcyB0ZXMGZGVsaXRzLAp=
UXVvaSBxdSdpbCBhZHZpZW5uZSwK
T24gdHJhY2Y2vbmUgc2VzIGNoYewNr251cywK
U2VzIHBlaw51cywK
SmUgdm91ZSBtZXMgbnVpdHMscm==
QSBsJ2Fzc2FzeW1waG9uYWUsc1==
QXV4IHJlcXVpZW1zLAr=
VHVhbnQgcGFyIGRlcG10LAq=
Q2UgcXV1IGp1IHN1bWUsc2==
SmUgdm91ZSBtZXMgbnVpdHMsc3==
QSBsJ2Fzc2FzeW1waG9uYWUsc3==
RXQgYXV4IGJsYXNwaGVtZXMsc4==
Sldhdm91ZSBqZSBtYXVkaXMsC1==
VG91cyBjZXV4IHV1aSBzJ2FpbWUdCwK
TCdlbm51bWksCu==
VGFwaSBkYW5zIG1vbiBlc3ByaXQsc5==
```

```
RumUmnRlIG1lcyBkZWZhaXRlcywK
U2FucyByZXBpdCBtZSBkZWZpZSsK
SmUgcmVuaWUsCq==
TGEgZmF0YWxlIGhlcmVzawUsCh==
UXVpIHJvbmdlIG1vbiDpIjJp0cmUsCo==
SmUgdmV1eCByZW5h5Y2vdHJlLAp=
UmVuYeWnr3RyZSsK
SmUgdm9lZSBtZXMgbnVpdHMsCn==
QSBsJ2Fzc2FzeW1waG9uawUsCq==
QXV4IHJlcXVpZW1zLAp=
VHVhbnQgcGFyIGRlcG10LAq=
Q2UgcXVlIGp1IHNIbWUsCo==
SmUgdm9lZSBtZXMgbnVpdHMsCm==
QSBsJ2Fzc2FzeW1waG9uawUsCl==
RXQgYXV4IGJsYXNwaGVtZXMsCm==
Sidhdm9lZSBqZSBtYXVkaXMsCu==
VG91cyBjZXV4IHFlaSBzJ2FpbWVudCwK
UGxlZXV4IHJlbnQgbGVzIHZpb2xvbnMgZGUgbWEgdm1lLAp=
TGEgdm1vbGVuY2UgZGUgbWVzIGVudm1lcywK
U2lwaG9ubmVlIHNIbXBob25pZSsK
RGVjb25jZXJ0YW50IGNvbmlcnRvLAq=
SmUgam9lZSBzYW5zIHRvdWNoZXIgbGUgRG8sCo==
TW9uIHRhbGVudCBzb25uZSBmYXV4LAp=
SmUgdm9pZSBtY24gZW5udWksCo==
RGFucyBsYSBtZWxvbnVuaWUsCl==
SmUgdHVlIG1lcyBwaG9iaWVzLAq=
RGFucyBsYSBkZXNoYXJtb25pZSsK
SmUgdm9lZSBtZXMgbnVpdHMsCv==
QSBsJ2Fzc2FzeW1waG9uawUsCn==
QXV4IHJlcXVpZW1zLAp=
VHVhbnQgcGFyIGRlcG10LAo=
Q2UgcXVlIGp1IHNIbWUsCm==
SmUgdm9lZSBtZXMgbnVpdHMsCp==
QSBsJ2Fzc2FzeW1waG9uawUsCm==
RXQgYXV4IGJsYXNwaGVtZXMsCu==
Sidhdm9lZSBqZSBtYXVkaXMsCm==
VG91cyBjZXV4IHFlaSBzJ2FpbWVudCwK
SmUgdm9lZSBtZXMgbnVpdHMsCn==
QSBsJ2Fzc2FzeW1waG9uawUgKGwnYXNzYXN5bXBob25pZSksCn==
Sidhdm9lZSBqZSBtYXVkaXMsCt==
VG91cyBjZXV4IHFlaSBzJ2FpbWVudA==
```

脚本1:

```
#python2
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('flag.txt', 'rb') as f:
    bin_str = ''
    for line in f.readlines():
        stegb64 = ''.join(line.split())
        rowb64 = ''.join(stegb64.decode('base64').encode('base64').split())
        offset = abs(b64chars.index(stegb64.replace('=', '')[-1]) - b64chars.index(rowb64.replace('=', '')[-1]))
        equalnum = stegb64.count('=') #no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
    print ''.join([chr(int(bin_str[i:i + 8], 2)) for i in xrange(0, len(bin_str), 8)])
```

脚本2:

```

#python2
def get_base64_diff_value(s1, s2):
    base64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
    res = 0
    for i in xrange(len(s2)):
        if s1[i] != s2[i]:
            return abs(base64chars.index(s1[i]) - base64chars.index(s2[i]))
    return res

def solve_stego():
    with open('flag.txt','rb') as f:
        file_lines = f.readlines()
        bin_str = ''
        for line in file_lines:
            steg_line = line.replace('\n', '')
            norm_line = line.replace('\n', '').decode('base64').encode('base64').replace('\n', '')
            diff = get_base64_diff_value(steg_line, norm_line)
            print diff
            pads_num = steg_line.count('=')
            if diff:
                bin_str += bin(diff)[2:].zfill(pads_num * 2)
            else:
                bin_str += '0' * pads_num * 2
            print goflag(bin_str)

def goflag(bin_str):
    res_str = ''
    for i in xrange(0, len(bin_str), 8):
        res_str += chr(int(bin_str[i:i + 8], 2))
    return res_str

if __name__ == '__main__':
    solve_stego()

```

最终得到flag

GXY{fazhazhenhaoting}

16. CRC碰撞

下载附件，都是压缩包

jdb' > 五瓶药水

名称	修改日期	类型	大小
 flag.zip	2020/9/8 14:50	WinRAR ZIP 压缩...	34 KB
 橙色.zip	2020/9/8 14:13	WinRAR ZIP 压缩...	1 KB
 红色.zip	2020/9/8 14:12	WinRAR ZIP 压缩...	1 KB
 黄色.zip	2020/9/8 14:13	WinRAR ZIP 压缩...	1 KB
 绿色.zip	2020/9/8 14:13	WinRAR ZIP 压缩...	1 KB
 青色.zip	2020/9/8 14:13	WinRAR ZIP 压缩...	1 KB

https://blog.csdn.net/qq_46150940

CRC碰撞，根据彩虹排序颜色排序，红橙黄绿青，CRC32值分别为

```
红 555FA1A2
橙 E5C67F46
黄 6E957E45
绿 76D6A31A
青 2B042586
```

补充上0x开头，使用脚本进行碰撞

```
import binascii
import string

dic=string.printable
crc1 = 0xe5c67f46
crc2 = 0x555fa1a2
crc3 = 0x6e957e45
crc4 = 0x76d6a31a
crc5 = 0x2b042586

def CrackCrc4(crc):
    for i in dic :
        for j in dic:
            for p in dic:
                for q in dic:
                    s=i+j+p+q
                    if crc == (binascii.crc32(s.encode("ascii"))):
                        print (s)
                        return 1

CrackCrc4(crc1)
CrackCrc4(crc2)
CrackCrc4(crc3)
CrackCrc4(crc4)
CrackCrc4(crc5)
```

脚本跑出来

```
cG90
aW9u
Z2Vu
YjEy
Mw==
```

在进行压缩包内为五字节或六字节爆破时，使用上面的脚本需要的时间太长了，在github上找到两个新的脚本

项目1:

<https://github.com/theonlypwner/crc32.git>

```
└─$ python crc32.py reverse 0x3dacac6b
4 bytes: {0x47, 0x18, 0x87, 0xce}
verification checksum: 0x3dacac6b (OK)
5 bytes: DCr4m (OK)
6 bytes: 1QhloU (OK)
6 bytes: 3mmr6H (OK)
6 bytes: 49Gqqk (OK)
6 bytes: 5Uumn6 (OK)
6 bytes: 8Gpbyp (OK)
6 bytes: 9G1Sbi (OK)
6 bytes: EHZxWz (OK)
6 bytes: F93jxv (OK)
6 bytes: I6mk_a (OK)
6 bytes: KGEHkt (OK)
6 bytes: O3d8oG (OK)
6 bytes: Uy1jKa (OK)
6 bytes: XJju5k (OK)
6 bytes: Zvoklv (OK)
6 bytes: a3H1hL (OK)
6 bytes: g6AbXj (OK)
6 bytes: kHvqPq (OK)
6 bytes: lQqOzZ (OK)
6 bytes: vwW0Z8 (OK)
```

当然由于crc32.py中字符集中只包含 `[A-Za-z0-9_]`,遇到特殊字符可能无法碰撞出来,可以对脚本进行改进,把字符集换成dic,即可打印字符。

```
dic=string.printable
#0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
```

然后在进行CRC32碰撞,可以得到更多的结果

```

└─(kali㉿kali)-[~/桌面/Python/CRC32]
└─$ python3 crc32.py reverse 0x3dacac6b
4 bytes: {0x47, 0x18, 0x87, 0xce}
verification checksum: 0x3dacac6b (OK)
5 bytes: (0_0) (OK)
5 bytes: DCr4m (OK)
6 bytes: 1QhloU (OK)
6 bytes: 3mmr6H (OK)
6 bytes: 49Gqqk (OK)
6 bytes: 5Uumn6 (OK)
6 bytes: 7ips7+ (OK)
6 bytes: 8Gpbyp (OK)
6 bytes: 9G1Sbi (OK)
6 bytes: EHZxWz (OK)
6 bytes: F93jxv (OK)
6 bytes: I6mk_a (OK)
6 bytes: J+wTt) (OK)
6 bytes: K+6eo0 (OK)
6 bytes: KGEHkt (OK)
6 bytes: N/jUuJ (OK)
6 bytes: O/+dnS (OK)
6 bytes: O3d8oG (OK)
6 bytes: TX.K94 (OK)
6 bytes: Uy1jKa (OK)
6 bytes: XJju5k (OK)
6 bytes: YJ+D.r (OK)
6 bytes: Zvoklv (OK)
6 bytes: a3H1hL (OK)
6 bytes: dG(pwf (OK)
6 bytes: e7U0i/ (OK)
6 bytes: fgb2/o (OK)
6 bytes: g6AbXj (OK)
6 bytes: kHvqPq (OK)
6 bytes: kT9-Qe (OK)
6 bytes: lQq0zZ (OK)
6 bytes: vwW0Z8 (OK)

```

项目2:

<https://github.com/kmyk/zip-crc-cracker>

```

└─$ python3 crack.py key.zip
reading zip files...
file found: key.zip / key.txt: crc = 0x3dacac6b, size = 5
compiling...
searching...
crc found: 0x3dacac6b: "yR>\x5c5"
crc found: 0x3dacac6b: "DCr4m"
crc found: 0x3dacac6b: "X\x0c.5y"
crc found: 0x3dacac6b: "(0_0)"
crc found: 0x3dacac6b: "\x09n0Ye"
done

key.zip / key.txt : 'yR>\\5'
key.zip / key.txt : 'DCr4m'
key.zip / key.txt : 'X\x0c.5y'
key.zip / key.txt : '(0_0)'
key.zip / key.txt : '\tn0Ye'

```


17. Base转图片

打开flag.txt, 发现 `data:image/png;base64` 开头,

```
flag.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H) 修改日期 类型 大小
data:image/jpg;base64,/9j/4AAQSkZJRgABAQEASABIAAD//gA7Q1JFQVRPUjogZ2QtanBlZyB2MS4wLWCh1c2luZyBJSkcgSIBFRyB2N
yEuAdxBjgPmRwPfb15yJ67Y2qNn96MdwF2xuJAzNuMzWMbESGujy32++OS7xjt98BqPvl+LPINVC0lu6V45OUck/GbXRLh2oKWql
qVMZQUmyresq3uC9R6g4su/EfJJPYzw5p0xjHMSSie5cORkjpJHeiNnuiVaq7DhO3F5Qqyerc3VtbUyeJnGwZyEhq1DXrFmG2dhLPs5
sjk2VNie78ZTi3xlc3T2q1uwuqlafA1C5BxjnM/YWRuUZUqd2VPNI7bWuo21lcWSXC1Le7P6wcJz8ps/yo1QpLRmNVR6jLxoA3DnAln:
0F0Dqm+/3zX/R9ZQ9e3N0w2peFfjMpgqpbobmejdhlP2fs4tc4DV3ZvlK6iawjbNA7c998RgOQeU4+y8ukbk+vOciOyqJgR+MkXC
/Jh/4Js9BqdodJpHD31LPImQN2t0VG19BPoMzCmnRwfAv8MWEVtkUYblwILNo3bPSFwA9RiRyCSKp22sEB4KFdz6LMhxKFywEb3
adGmFRQvGd8zj69qtTlv0pnP1UlaDkfKdLjy6y0j2F/pHVG97UH+AAEYbq7f3ryufg8H4/CBEX5Q0Fs7UR6h8dxXYermNW2oKcFC
WocurnB5+sqLpBjr1iVXoHErGUxpfBxH1MgHhQsfiDMELv3r16tTYHPCw3EtGbVBiOR1xJUccjvtAqb46wim/5R0QW2i3lO3rvp12M2l
hRZmC4PKXdnZBaW6yOSvP1qNfLp7kYgBkVcRiae5sFK5wczP6lRakcKD16TZshx2S21tRqgs3CT5QtdMRW46aqD5yv0rTKlduP2f
NV6VV6qVH4DTbHxgNR6fEAhyvrKoQ0nYzquQMZiAT7WfWLNmnhBYt6dYxMKSo7Yyx5+clpP4Rv0kdrpWp3Kq9vp9d1JyPDzj7i2
OKk5+qZLQPs9JVLdVdTIW3E4xMGeUoOy2qG6t6lpcArWtnKkHniXjEiQM4x26SNsb+eY9m29cSNiRmMGMdsbyM89/KOYyM49MYh
YL+U0uyGqOvRgdhykilfKOpsHG+MyZUHMDMwWwWyW3HARK5PnDqTjg3PSBpgHAzzj+9wvTYS4qODQY78KtjpbHfPPYxj1yMj5-
HDx4WbJV0cUnsgdTjzL3QVqaRqdRvXOFVroQpzyHrkSpgDZwSekuNWQnRtNrcQ3QrtNE6MzQdp9SoCgtp3iszNxPw7Ez3FaXNM
joYQlrntvM9IAMGO5j5QtLMHGt1kq2oHwk6ABK/nOcDsX9mAnfZ8uUq0CRXhN/8obQXI28h+UY9MK8l1wcc/Jlkyi/7O5CVgftc
tpF+V4hj0gtSkAcesmpuGUb52nXwQY+mTdgmcDznMeHn0kzKM7SNvCD5Y5SyBuMffft/rrFxDi+YMRcfJa0F+Bzx1E6SM8xykZf
aSrwb1+rQV8t67dYXb/o1b9nvKVb2Mg8Cqd43UP0e12DpoqrRK7rV5851+DmYItNnzgA4mn7ONQRUj1EXvVBZcj4Sqs7dXAB5m'
5J4+imKHpGlcDG/KH1bYgAyl0vIR8goHC5bA55kqHAEeKZUn4xBceohyIJKbkfdDre5rW1xTurWp3dakcg/a9JXKMfHfKEU6oBxn8Y
pak7IwByQefKKE21Mce/PH8opnyEeuPEY3EkfHFuRI2dE95IHxMwsSERzzzjQM/DEhe8t0PiqqN/ORHU7RV/uAaxzdI6HoMwd8zu
hc27itbsch13x8fWZpRnZxnPnDtPv7nSqhe3YtRZv1lI7honoGg5UOMjbacqllfwVgMEY4j0lglO3v7Q3enniHN6P1kgqgOvu9NwZnTAj
Fp44R77HYAR9Clc6tccTtimv8ACqzt/foKbabprcNjW/XVR9aS9lWNv9Rp0UbTdlYhQMvbgfWpPkaCUUIUcWTuTz6MiiikhVnto135
xjKMf3prW8SNtZez/bRaJ7PVnqHDqW7vHMtnaW+nioNOTxW3qd0vGfXea0FmKvadem1SmVZOJiD98Js7OkIv1Uma25062u1lqUv
B/OOCY+OJ2wZSJKKfVxyhKDHDtIU/+oQg5HnNHlaJOGMScbDEipryOZKRhcEQTKERk8jzkLsBjp6yRjw50OsBro9aaoqU989DyMUm
Mv7Ls8lPuvFmXFhot3SjZqDEkcpaUtPvGBAT9sdTOVpvoVkn3eg0a68NOMAc8+Uzl12fvKDsBTZgPIT0j9FagTkU6fP7UA1mnX0nT
Z5/wDSLqHHVoaehwAvE+JsPbbo1FRbJscXvF/WeX9p7prR77mox3U8OPKT2OimcZdUHXYAqjVX4VtPTKqnfHU4is0FbUqKN7obit
l3dUrekOJqh4QBPWdE0waNpSWuBxYy5A5zLJKkOKLNWY2OkIQHP84NTP8Aoc4Sh2+c5m9mr0LHh5dJ08z0A5HltEw8941ohCUff-
nG3KHcJiZc52G11oLKGtZ+L3cZqpsM81ztNE9EH74O1ttt1E1iKhlOthiO33vVLfhA2Aln7POeznErkiESmoAHlCOXHwnVoeSVUxKUqo4
<
```

在线网站Base64转图片

在线调色板 网页常用色彩 中日传统色彩 传图上色 WEB安全色 网页颜色选择器 颜色代码查询、RGB颜色值 base64图片在线转换工具



```
JZas7mPmKSCiSKkaEj1p7z1x091ozumKkog06b0n5wKce55000e0E3g5zV0c-
G+RFFGhC7s8PMe7EUQMjc4iigBzh6fOdCb422ijGOCk9Y7gwcbc4oomlbwct5wqc
dOUUUXgZxllOM8zlypO5O3lFFAH0IU8deU6Ke/P0iiiYkPFPO4ON/KdCnbeKKD7Gloz
tuNxllypB2PWKKUMcqqk7+7ykoT4RRSWNGa+kOkW7NKAQMVRid05i1tp1lBhsozj
bMUU6l/oVDsLUVK1UqzjC7S0tLcFNzyEUUxkbeQzuAq5B3z5RAYO+8UUhAdKZ8Od
JlqiZUnbJiighsr69HIJ6yv7vGzYIMUuJnlIudNtqq5ZStRtgybYfS1V+zmvNV7hLruR4
RUJ5fziilrowfrtdl+kGz1NcVNOqo55lXGJcVe0VKiMLak5Gd2iimcuiUDv2mu/Etk0oAj
qzH+UjGvaxVLhKdkuDtKMYopkuwFZ1LzX6yDgurajkFUQyQprlZStq5TP2FxFFCQED
6fqJJ77Wrth5K2l06lzZ73Ubx+m9UxRSojXaEez1tsHqVn+NQyl9ntOPhNmSfNjmKKQ
zZJUPTs/pyEYtaZyd8rO/ojTUXis6WcfYEUUoTCP0dbIMJQpqrVrkLOC3ROSrv6RRSflx
1OmCONziiijA/ZZmxhZ3tjX2Fsd2F5c190YWtlXzVfcG90aW9uX3doZW5faV9nMF8w
dXR9aGpraGpkZmAAZGdmZHNmZGRmZGZkZmRrZmtoZ2pqa2hzZmpoY2pka2hm
amtkaGZofSagLcAGIA==
```

*请上传小于300KB的.jpg/jpeg/.gif/.bmp/.png/.ico格式图片, 不建议将大图转换。

图片转成Base64 Base64还原图片 清空结果

18. 盲水印

BlindWaterMark

下载地址: <https://github.com/chishaxie/BlindWaterMark>

需要安装cv2库, `pip install opencv-python`

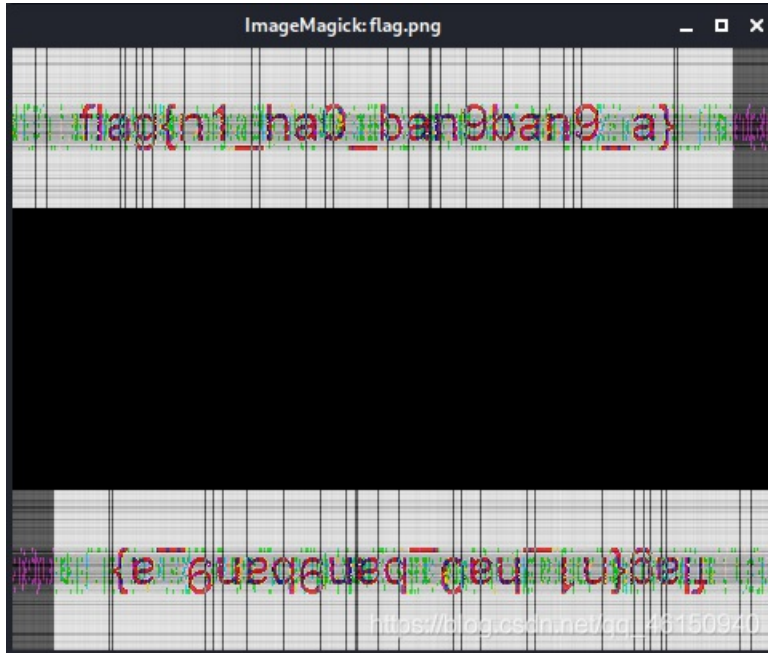
```
python2 -m pip install opencv-python
```

合成盲水印

```
python bwm.py encode hui.png wm.png hui_with_wm.png
```

提取图中的盲水印

```
python2 bwm.py decode 2.png 1.png flag.png  
或者  
python3 bwmforpy3.py decode 2.png 1.png flag.png  
#2.png --original <原始图像文件>  
#1.png --image <图像文件>  
#flag.png --result <结果文件>
```



频域盲水印

参考文章: <https://xz.aliyun.com/t/1875/#toc-9>

[GWCTF2019]huyao

频域盲水印脚本: BlindWaterMarkplus.py

```

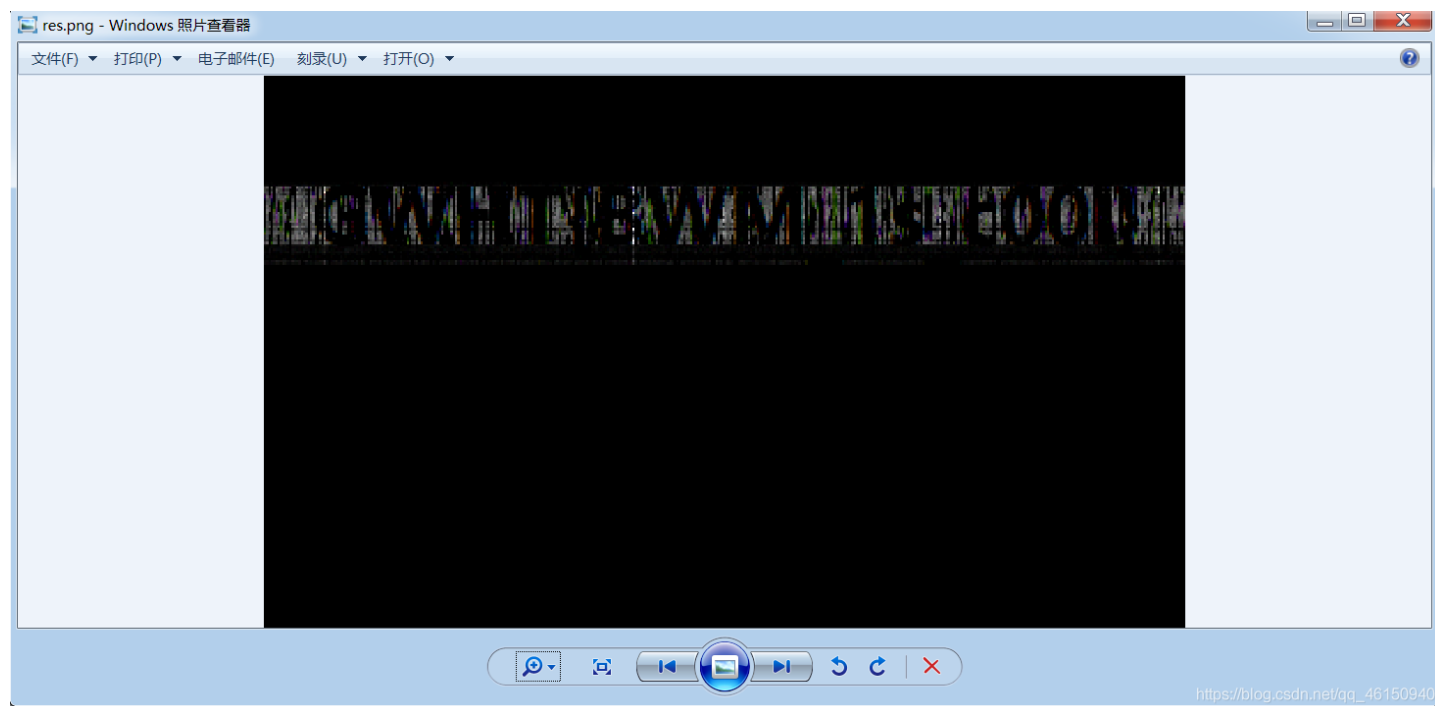
# coding=utf-8
import cv2
import numpy as np
import random
import os
from argparse import ArgumentParser
ALPHA = 5
def build_parser():
    parser = ArgumentParser()
    parser.add_argument('--original', dest='ori', required=True)
    parser.add_argument('--image', dest='img', required=True)
    parser.add_argument('--result', dest='res', required=True)
    parser.add_argument('--alpha', dest='alpha', default=ALPHA)
    return parser
def main():
    parser = build_parser()
    options = parser.parse_args()
    ori = options.ori
    img = options.img
    res = options.res
    alpha = options.alpha
    if not os.path.isfile(ori):
        parser.error("original image %s does not exist." % ori)
    if not os.path.isfile(img):
        parser.error("image %s does not exist." % img)
    decode(ori, img, res, alpha)
def decode(ori_path, img_path, res_path, alpha):
    ori = cv2.imread(ori_path)
    img = cv2.imread(img_path)
    ori_f = np.fft.fft2(ori)
    img_f = np.fft.fft2(img)
    height, width = ori.shape[0], ori.shape[1]
    watermark = (ori_f - img_f) / alpha
    watermark = np.real(watermark)
    res = np.zeros(watermark.shape)
    random.seed(height + width)
    x = range(height / 2)
    y = range(width)
    random.shuffle(x)
    random.shuffle(y)
    for i in range(height / 2):
        for j in range(width):
            res[x[i]][y[j]] = watermark[i][j]
    cv2.imwrite(res_path, res, [int(cv2.IMWRITE_JPEG_QUALITY), 100])
if __name__ == '__main__':
    main()
python BlindWaterMarkplus.py --original huyao.png --image stillhuyao.png --result res.png

```

然后，使用命令

```
python BlindWaterMarkplus.py --original huyao.png --image stillhuyao.png --result res.png
```

得到res.png

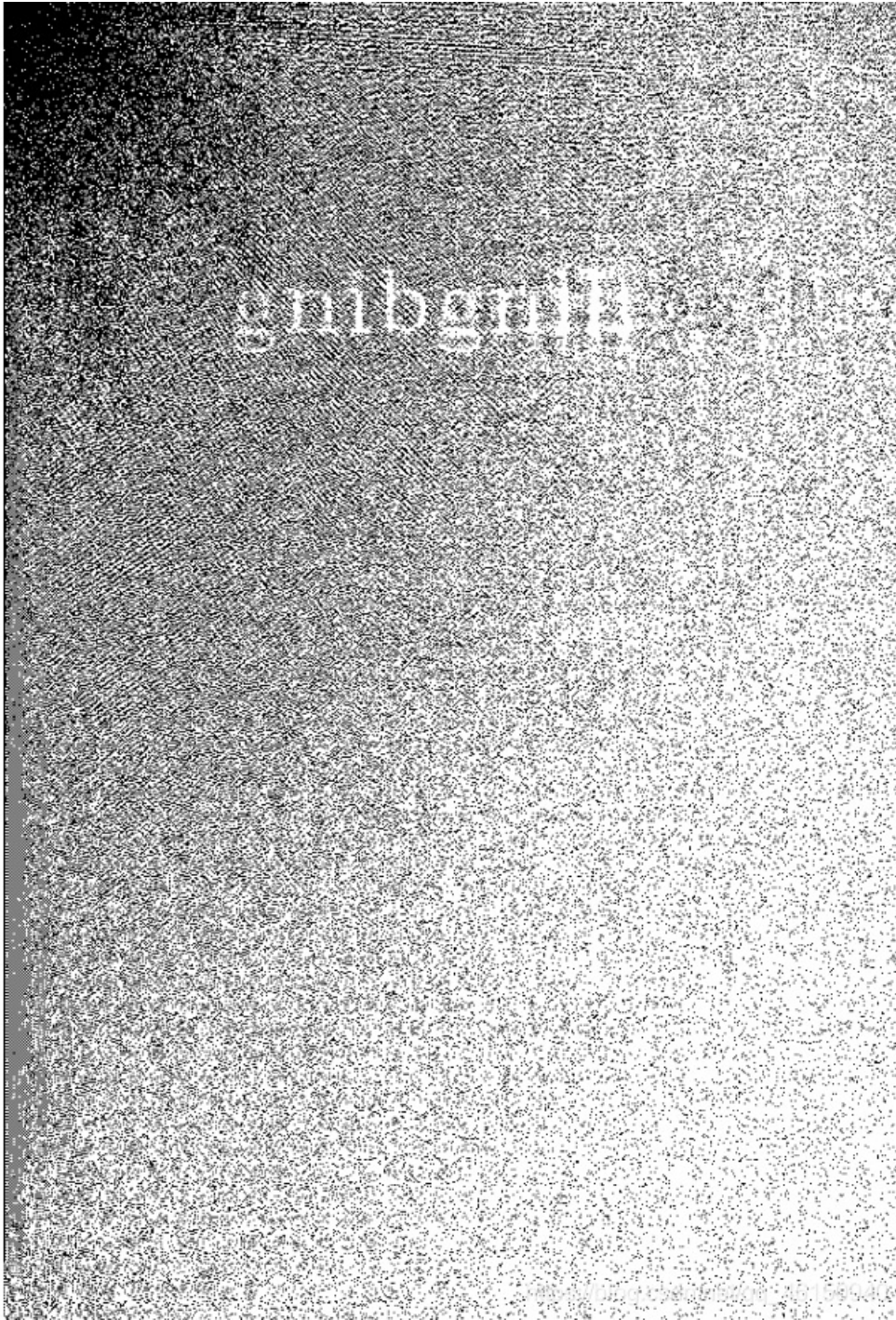


JAVA盲水印

项目地址: <https://github.com/ww23/BlindWaterMark/releases>

执行命令

```
java -jar BlindWatermark.jar decode -c bingbing.jpg decode.jpg
```

19. gnuplot画图

[BUUCTF]- 梅花香之苦寒来

用010 editor打开图片，在文件尾发现大量数据

```
启动 meihuai.jpg x
编辑为: Hex 运行脚本 Run Template: JPC.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
52B0h: 42 7E 03 98 52 EE 68 42 80 3F 2A 63 72 84 20 A6 B~.~RihBe?*cr,, |
52C0h: EC 87 21 0B 47 E9 1D D5 72 42 13 F4 09 0E 48 42 i+!.Gé.ÖrB.ó..HB
52D0h: 06 A2 A7 2F 54 21 11 68 42 15 02 10 85 00 84 21 .¢$/T!.hB....."!
52E0h: 00 84 21 50 24 84 28 1A 10 84 02 10 85 40 84 21 ..!P$,,(..,..@..!
52F0h: 00 84 21 07 FF D9 32 38 33 37 32 63 33 37 32 39 ..!.yÜ28372c3729
5300h: 30 61 32 38 33 37 32 63 33 38 32 39 30 61 32 38 0a28372c38290a28
5310h: 33 37 32 63 33 39 32 39 30 61 32 38 33 37 32 63 372c39290a28372c
5320h: 33 31 33 30 32 39 30 61 32 38 33 37 32 63 33 31 3130290a28372c31
5330h: 33 31 32 39 30 61 32 38 33 37 32 63 33 31 33 32 31290a28372c3132
5340h: 32 39 30 61 32 38 33 37 32 63 33 31 33 33 32 39 290a28372c313329
5350h: 30 61 32 38 33 37 32 63 33 31 33 34 32 39 30 61 0a28372c3134290a
5360h: 32 38 33 37 32 63 33 31 33 35 32 39 30 61 32 38 28372c3135290a28
5370h: 33 37 32 63 33 31 33 36 32 39 30 61 32 38 33 37 372c3136290a2837
5380h: 30 61 32 38 33 37 32 63 33 38 32 39 30 61 32 38 0a28372c38290a28
```

```
5380h: 32 63 33 31 33 37 32 39 30 61 32 38 33 37 32 63 2c3137290a28372c
5390h: 33 31 33 38 32 39 30 61 32 38 33 37 32 63 33 31 3138290a28372c31
53A0h: 33 39 32 39 30 61 32 38 33 37 32 63 33 32 33 30 39290a28372c3230
53B0h: 32 39 30 61 32 38 33 37 32 63 33 32 33 31 32 39 290a28372c323129
53C0h: 30 61 32 38 33 37 32 63 33 32 33 32 32 39 30 61 0a28372c3232290a
53D0h: 32 38 33 37 32 63 33 32 33 33 32 39 30 61 32 38 28372c3233290a28
53E0h: 33 37 32 63 33 32 33 34 32 39 30 61 32 38 33 37 372c3234290a2837
53F0h: 32 63 33 32 33 35 32 39 30 61 32 38 33 37 32 63 2c3235290a28372c
5400h: 33 32 33 36 32 39 30 61 32 38 33 37 32 63 33 32 3236290a28372c32
5410h: 33 37 32 39 30 61 32 38 33 37 32 63 33 32 33 38 37290a28372c3238
5420h: 32 39 30 61 32 38 33 37 32 63 33 32 33 39 32 39 290a28372c323929
5430h: 30 61 32 38 33 37 32 63 33 33 33 30 32 39 30 61 0a28372c3330290a
5440h: 32 38 33 37 32 63 33 33 33 31 32 39 30 61 32 38 28372c3331290a28
5450h: 33 37 32 63 33 33 33 32 32 39 30 61 32 38 33 37 372c3332290a2837
5460h: 32 63 33 33 33 33 32 39 30 61 32 38 33 37 32 63 2c3333290a28372c
5470h: 33 33 33 34 32 39 30 61 32 38 33 37 32 63 33 33 3334290a28372c33
5480h: 33 35 32 39 30 61 32 38 33 37 32 63 33 33 36 35290a28372c3336
```

https://blog.csdn.net/qq_46150940

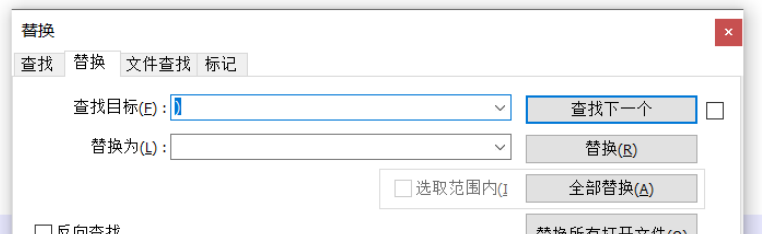
复制到Notepad++, 使用 hex转ascll插件

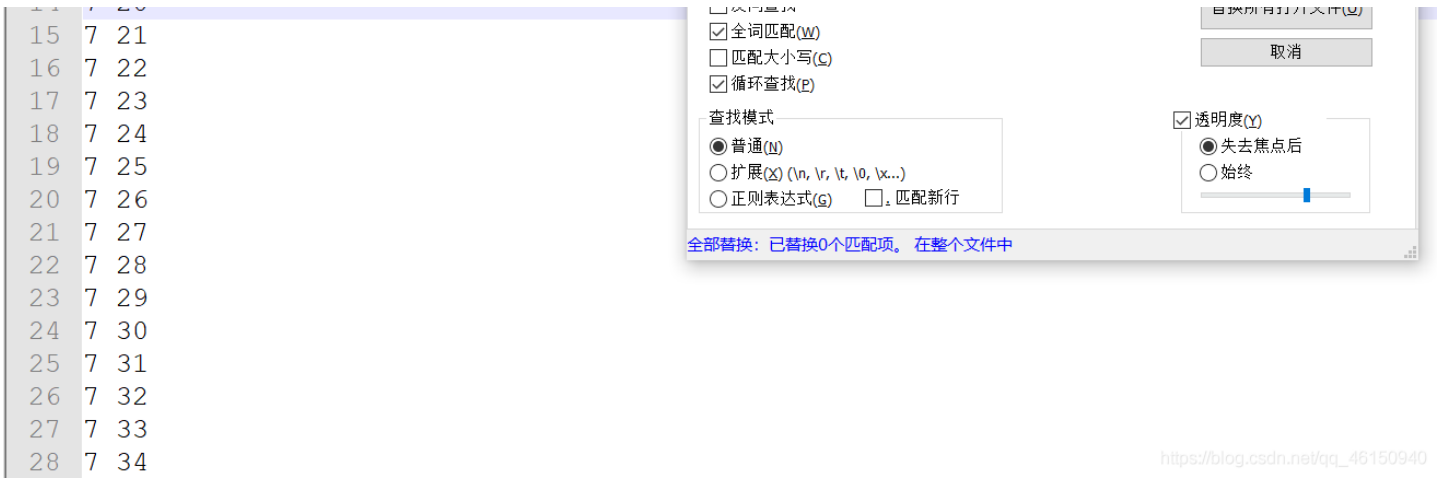
```
m. txt
1 (7,7)
2 (7,8)
3 (7,9)
4 (7,10)
5 (7,11)
6 (7,12)
7 (7,13)
8 (7,14)
9 (7,15)
10 (7,16)
11 (7,17)
12 (7,18)
13 (7,19)
14 (7,20)
15 (7,21)
16 (7,22)
17 (7,23)
18 (7,24)
19 (7,25)
20 (7,26)
21 (7,27)
22 (7,28)
23 (7,29)
24 (7,30)
25 (7,31)
```

https://blog.csdn.net/qq_46150940

转换为gnuplot能识别的内容

```
m. txt
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
14 7 20
```





使用gnuplot工具画图

```
plot "m.txt"
```



扫码即可得到flag

看了其他人的解法，大师傅的python脚本


```
import matplotlib.pyplot as plt
import numpy as np

with open('m.txt', 'r') as h:
    h = h.read()
with open('ascii.txt', 'a') as a:
    for i in range(0, len(h), 2):
        tmp = '0x'+h[i]+h[i+1]
        tmp = int(tmp, base=16)
        if chr(tmp) != '(' and chr(tmp) != ')':
            a.write(chr(tmp))

x, y = np.loadtxt('ascii.txt', delimiter=',', unpack=True)
plt.plot(x, y, '.')
plt.show()
```

20. 零宽字符加密

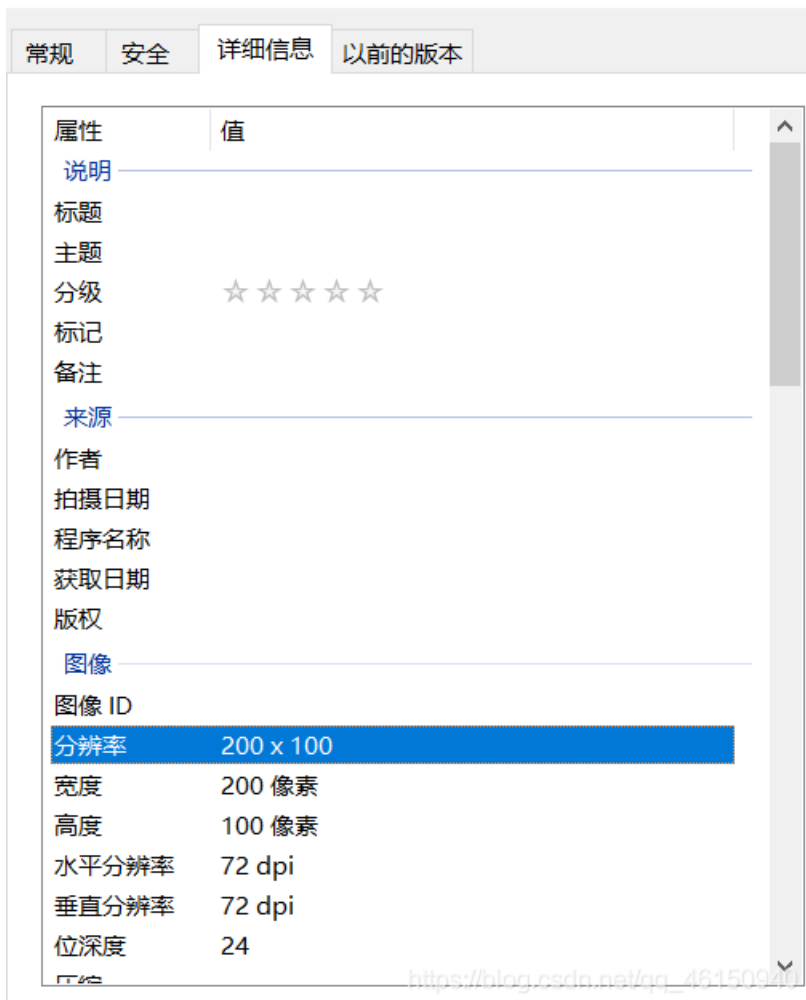
下载附件，删去不必要的文件，查看文件夹属性，一共120张图片

不眠夜之WA 属性



再查看图片属性，分辨率为200x100

00fd5b9.jpg 属性



1.使用 `montage` 命令将碎片合成为一张图片,得到flag.jpg

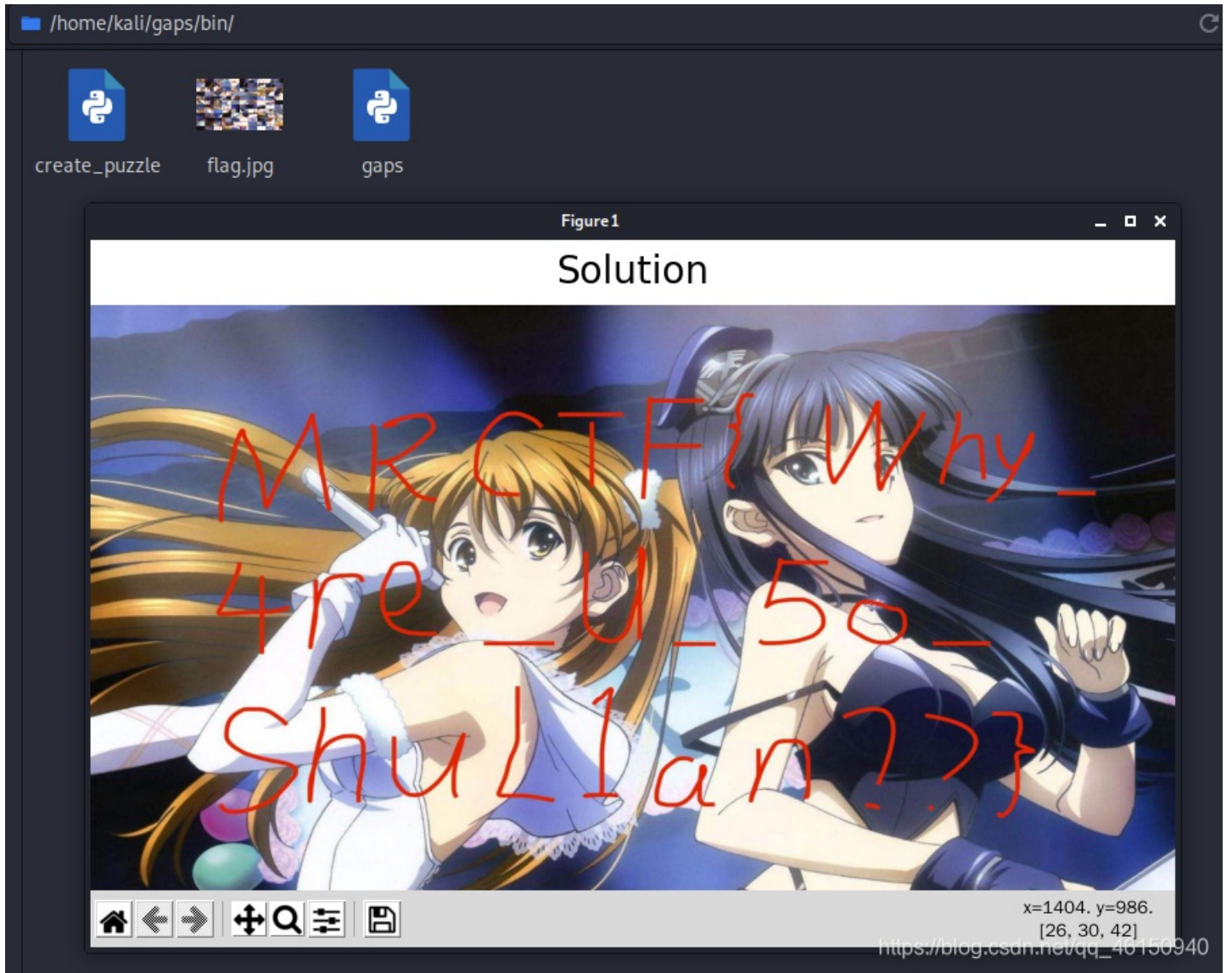
```
montage *.jpg -tile 10x12 -geometry 200x100+0+0 flag.jpg
```



2. 将生成的flag.jpg拖到gaps的bin目录下执行

```
python3 gaps --image=flag.jpg --generations=40 --population=120 --size=100
```

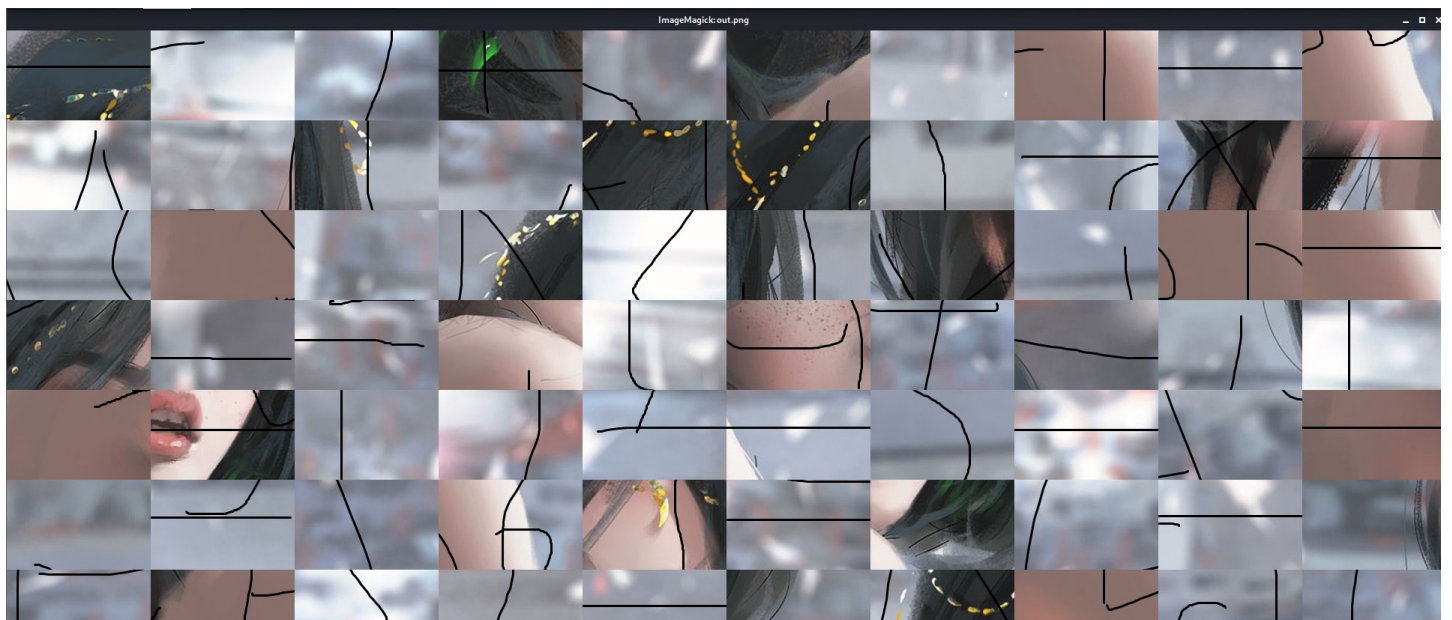
#population为图片数量, size为每个图片高度



[第一届SXC CTF]拼图

将100张碎片合成，每个图片分辨率为192x120，生成的out.png分辨率为1920x1200

```
montage *png -tile 10x10 -geometry 192x120+0+0 out.png
```

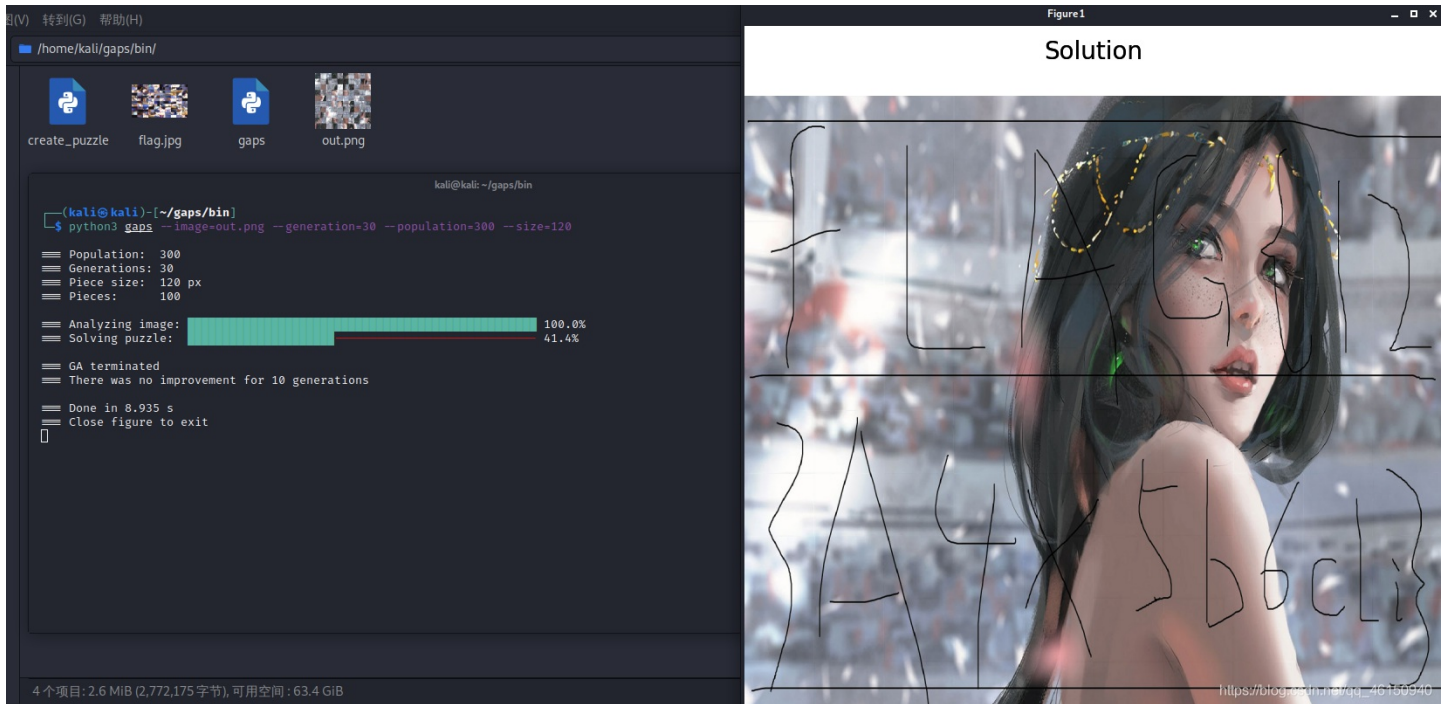


修改拼接后的图片大小，改为1200x1200



还原拼图

```
python3 gaps --image=out.png --generation=30 --population=300 --size=120
```

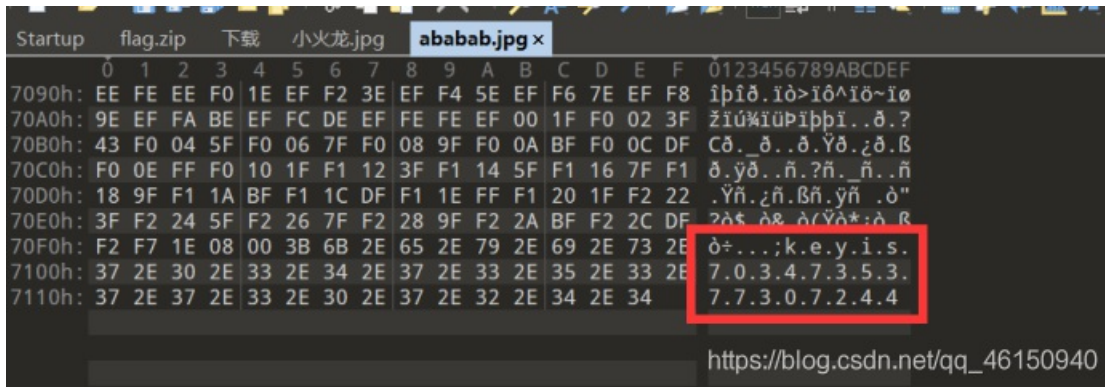
22.十六进制文件字节左右翻转

[2020金盾信安杯]-我和十六有个约定

内容：链接：<https://pan.baidu.com/s/1LJiUaVyANUZBN4bb1SHiCQ> 密码：u5sr

下载附件，得到ababab.png和flag.zip

010打开ababab.png，发现其实是GIF，不过没什么用，拖到最后，看到keys7034735377307244



十六进制转字符串，得到p4sSw0rD，这就是压缩包密码解压文件flag.txt和splice.txt

flag.txt内容为

发现第一行 **FF D8 FF E0**，倒着读是jpg的文件头，找到一个脚本将flag.txt反转，生成output.txt

```
#coding=utf-8
import re
f=open("flag.txt")
out = open("output.txt","w")
out.close()
out = open("output.txt","a")

for fl in f.readlines():
    temp=re.findall(r'.{2}',fl)
    order=[]
    for i in temp:
        order.append(i)
    order.reverse()
    print(''.join(order))
    out.write(''.join(order)+'\n')

f.close()
out.close()
```



去掉空格

```
*output.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
FFD8FFE00010A4469446000110010090
00900000FFE10082547896660000D44D
002A0000000800051012000300000001
00010000101A0005000000010000004A
101B0005000000010000005210280003
00000001000200007869000400000001
0000005A000000000000009000000001
0000009000000000100020A0200040000
0001000010100A030004000000010000
1016000000000000FFED002C0568F674
F673866F0720332E03008342944D4025
0000000000104D1DC8D9F8002B049E80
9098CEF8247EF2F0F09443345F0552
F446944C540010010000F0E01670076C
20100000D66E4772254724208559A520
70E40008001900150031001D16633770
https://blog.csdn.net/qq\_46150940
```

再贴上一位大佬的脚本

```
re = []
s = open("./flag1.txt", 'w') #以覆盖的方式进行写文件
f = open('./flag.txt'); #打开文件,进行读取内容

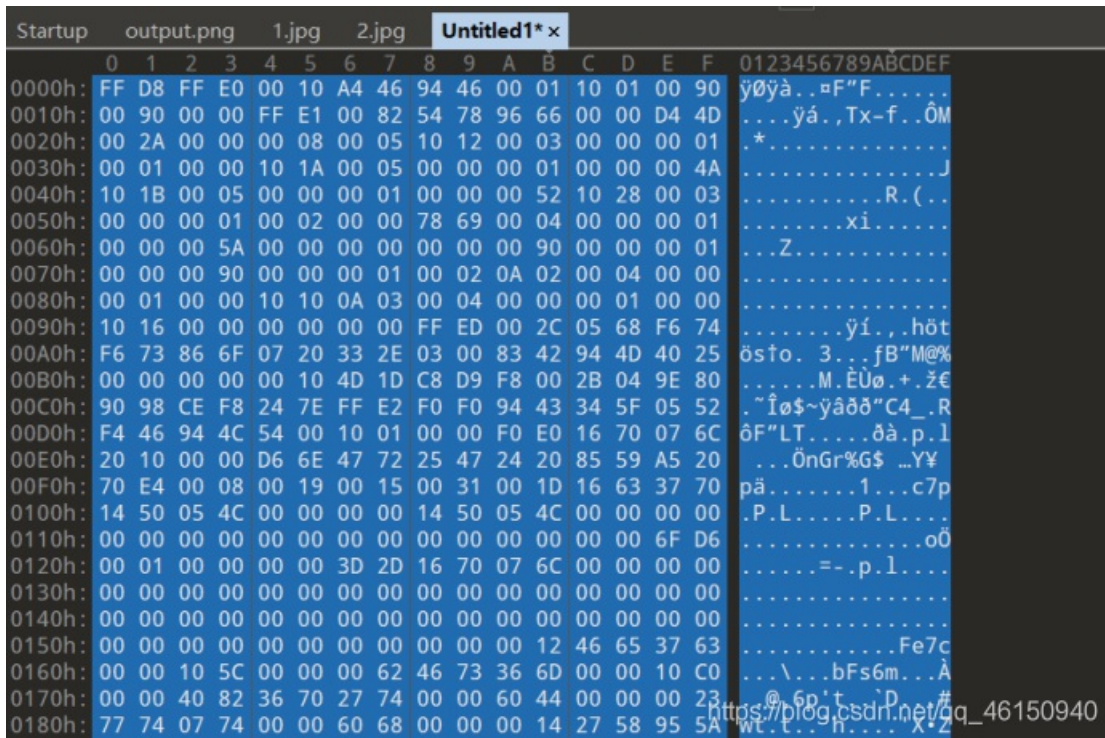
for line in f.readlines(): #调用readlines()一次读取所有内容并按行返回list
    re = line.split(' ') #split()函数以空格为划分生成了一个新的列表并放入re[]中
    re = re[:-1] #因为readlines返回的每一行最后都有\n换行符,所以要去掉
    re = re[::-1] #使列表中的值逆序
    s.write(" ".join(re) + '\n') #join()函数用于列表格式化输出,用空格相连接

s.close()
```


得到

```
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 90
00 90 00 00 FF E1 00 82 45 78 69 66 00 00 4D 4D
00 2A 00 00 00 08 00 05 01 12 00 03 00 00 00 01
00 01 00 00 01 1A 00 05 00 00 00 01 00 00 00 4A
01 1B 00 05 00 00 00 01 00 00 00 52 01 28 00 03
00 00 00 01 00 02 00 00 87 69 00 04 00 00 00 01
00 00 00 5A 00 00 00 00 00 00 90 00 00 00 01
00 00 00 90 00 00 00 01 00 02 A0 02 00 04 00 00
00 01 00 00 01 10 A0 03 00 04 00 00 00 01 00 00
01 16 00 00 00 00 00 00 FF ED 00 2C 50 68 6F 74
6F 73 68 6F 70 20 33 2E 30 00 38 42 49 4D 04 25
00 00 00 00 00 10 D4 1D 8C D9 8F 00 B2 04 E9 80
09 98 EC F8 42 7E FF E2 0F F0 49 43 43 5F 50 52
4F 46 49 4C 45 00 01 01 00 00 0F E0 61 70 70 6C
02 10 00 00 6D 6E 74 72 52 47 42 20 58 59 5A 20
07 E4 00 08 00 19 00 15 00 31 00 1D 61 63 73 70
```

打开010，新建一个Hex file，把编辑后的output.txt内容粘贴进去



文件头是FF D8 FF E0，是jpg文件，另存为output.jpg，得到一张残缺的二维码



https://blog.csdn.net/qq_46150940

splice.txt内容为

```
splice.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAEsAAABHCAYAAABPjLqRAAAHqUIEQVR4Xu1cyS5sQRiuvglymdDYiu
boG24UxAVMRgCPj4/m9vbWoHRdRQ8WAGlvbzcyjyOmq6srvTIBkDDzWBxuegAA7SLiwuzublpS4LI+2ij6MFCp7u7u63JhMIkGSg
FEbAqdRTHUjrGLWXiiA5nl0itMbwz9PTUTExMGJTrNxDR8Jmtc4sf76jlL047CNFI/ts+cLB4aYECv20kELH0agTNyfn5uk8fOzs5MbW1t
```

Base64转换为图片

补全二维码，扫码得到flag

23.十六进制文件头尾逆序

[HGAME2021]-Hallucigenia

用Stegsolve打开进行异或

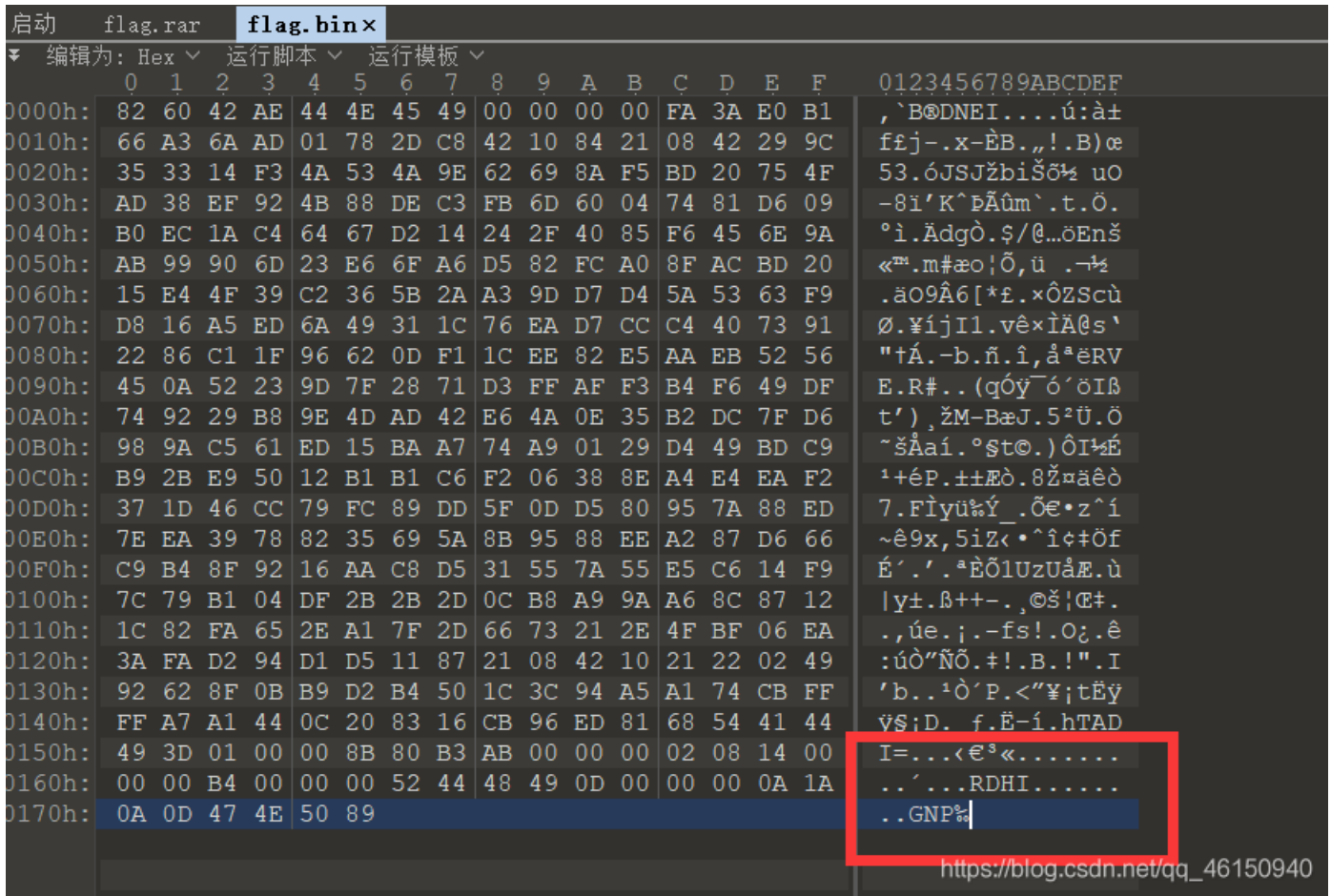


扫码，是二进制文件

```
gmBCrkRORUkAAAAA+jrgswajaq0BeC3IQhCEIQhCKZw1MxTzS1NKnmJpivW9IHVPrTjvkkuI3sP7bWAEdIHWCbDsGsRkZ9IUJC9AhfZFbpqrmZBt  
I+ZvptWC/KCPrL0gFeRPOcI2WyqjndfUW1Nj+dgWpe1qSTEdurXzMRAc5EihsEflmIN8RzuguWq61JWRQpSI51/KHHT/6/ztpZJ33SSKbieTa1C  
5ko0NbLcf9aYmsVh7RW6p3SpASnUSb3JuSvpUBKxscbyBjiOp0Tq8jcdRxs5/IndXw3VgJV6i01+6jl4gjVpWouVi06ih9ZmybSPkhaqyNUxVXpV  
5cYU+Xx5sQTfKystDLipmqAMhxIcgvplLqF/LWZzIS5Pvwbq0vrS1NHVEYchCEIQISICSZJijwu50rRQHDyUpaF0y///p6FEDCCDFsuW7YFoVEFE  
ST0BAACLgLOrAAAAAggUAAAAtAAAAFJESEkNAAAChoKDUdOUIk=
```

使用脚本生成二进制文件flag.bin

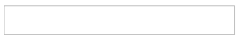
```
from base64 import b64decode  
open('flag.bin', 'wb+').write(b64decode(open('flag_inv_b64.txt', 'rb').read()))
```



□ 16 进制编辑器打开可以发现 PNG、IHDR 等字样，文件头尾逆序

```
from base64 import b64decode
open('flag.png', 'wb+').write(b64decode(open('h.txt', 'rb').read()[::-1]))
```

得到flag.png



上下翻转镜像，得到flag

```
hgame{tenchi_souzou_dezain_bu}
```

使用十六进制文件头尾逆序脚本，颠倒十六进制文件flag.bin也可以得到flag.png

```
input = open('flag.bin', 'rb')
input_all = input.read()
ss = input_all[::-1]
output = open('flag.png', 'wb')
output.write(ss)
input.close()
output.close()
```

24.zlib解压

例题1:


```
D:\CTF\pngcheck>pngcheck.exe -v pngcheck.png
File: pngcheck.png (135570 bytes)
  chunk IHDR at offset 0x0000c, length 13
    600 x 379 image, 24-bit RGB, non-interlaced
  chunk pHYS at offset 0x00025, length 9: 3827x3827 pixels/meter (97 dpi)
  chunk tEXt at offset 0x0003a, length 17, keyword: Title
  chunk tEXt at offset 0x00057, length 19, keyword: Author
  chunk zTXt at offset 0x00076, length 45, keyword: Description
  chunk IDAT at offset 0x000af, length 135317
    zlib: deflated, 32K window, maximum compression
  chunk IDAT at offset 0x21150, length 92: EOF while reading data
ERRORS DETECTED in pngcheck.png
```

010 editor搜索IDAT，找到最后那段IDAT块，读取数据时发生错误,可能存在zlib压缩

脚本1:

```
import zlib
s = '789C4BCB494CAF4E4B36324D4E4A324FB230B534B54C4B35303637B63032314CB4344B4A33324EA90500E9E20B5FD01C6808'
b = bytes.fromhex(s)
flag = zlib.decompress(b)
print(flag)
```

运行得到flag

```
flag{fc25cbb7b85959fe03738241a96bf23d}
```

例题2:

sctf.png



先用pngcheck检查一下

```

D:\CTF\pngcheck-3.0.2-win32>pngcheck.win64.exe -v test/sctf.png
File: test/sctf.png (1421461 bytes)
  chunk IHDR at offset 0x0000c, length 13
    1000 x 562 image, 32-bit RGB+alpha, non-interlaced
  chunk sRGB at offset 0x00025, length 1
    rendering intent = perceptual
  chunk gAMA at offset 0x00032, length 4: 0.45455
  chunk pHYs at offset 0x00042, length 9: 3780x3780 pixels/meter (96 dpi)
  chunk IDAT at offset 0x00057, length 65445
    zlib: deflated, 32K window, fast compression
  chunk IDAT at offset 0x10008, length 65524
  chunk IDAT at offset 0x20008, length 65524
  chunk IDAT at offset 0x30008, length 65524
  chunk IDAT at offset 0x40008, length 65524
  chunk IDAT at offset 0x50008, length 65524
  chunk IDAT at offset 0x60008, length 65524
  chunk IDAT at offset 0x70008, length 65524
  chunk IDAT at offset 0x80008, length 65524
  chunk IDAT at offset 0x90008, length 65524
  chunk IDAT at offset 0xa0008, length 65524
  chunk IDAT at offset 0xb0008, length 65524
  chunk IDAT at offset 0xc0008, length 65524
  chunk IDAT at offset 0xd0008, length 65524
  chunk IDAT at offset 0xe0008, length 65524
  chunk IDAT at offset 0xf0008, length 65524
  chunk IDAT at offset 0x100008, length 65524
  chunk IDAT at offset 0x110008, length 65524
  chunk IDAT at offset 0x120008, length 65524
  chunk IDAT at offset 0x130008, length 65524
  chunk IDAT at offset 0x140008, length 65524
  chunk IDAT at offset 0x150008, length 45027
  chunk IDAT at offset 0x15aff7, length 138
  chunk IEND at offset 0x15b08d, length 0
No errors detected in test/sctf.png (28 chunks, 36.8% compression).

```

正常的块的length是65524，而倒数第二个IDAT块长度是45027，最后一个长度是138，最后一个IDAT块异常

010 editor找到最后一个IDAT块的十六进制，使用脚本解压

```

import zlib
s = '''
78 9C 5D 91 01 12 80 40 08 02 BF 04 FF FF 5C 75
29 4B 55 37 73 8A 21 A2 7D 1E 49 CF D1 7D B3 93
7A 92 E7 E6 03 88 0A 6D 48 51 00 90 1F B0 41 01
53 35 0D E8 31 12 EA 2D 51 C5 4C E2 E5 85 B1 5A
2F C7 8E 88 72 F5 1C 6F C1 88 18 82 F9 3D 37 2D
EF 78 E6 65 B0 C3 6C 52 96 22 A0 A4 55 88 13 88
B3 A1 70 A2 07 1D DC D1 82 19 DB 8C 0D 46 5D 8B
69 89 71 96 45 ED 9C 11 C3 6A E3 AB DA EF CF C0
AC F0 23 E7 7C 17 C7 89 76 67 D9 CF A5 A8
'''
s = s.replace(' ', '').replace('\n', '')
b = bytes.fromhex(s)
flag = zlib.decompress(b)
print(flag)

```

运行得到

```
b'11111100010000110111111100000101110010110100000110111010100000000101110110110100100000001011101101101011
101101001011101100000101010110110100000111111110101010101011111100000000101110111000000001101001100000101001110
110111101010100100001110000000000101000000001001001101000100111001111011100111100001110111110001100101000110011
100001010100011010001111010110000010100010110000011011101100100001110011100100001011111101000000001101010010001
111011111101110000110101101110000010000110011000111101011101000110100111110000101110101100011101001110011100111010
0100111011011000110000010110001101000110001111111011010110111011011'
```

得到的01 串的长度是625， $625 = 25*25$ ，是个正方形的形状

```
from PIL import Image

MAX = 25
pic = Image.new("RGB", (MAX, MAX))
str = "111111000100001101111111000001011100101101000001101110101000000001011101101110100100000000101110110111
010111011010010111011000001010101101101000001111111101010101010111111000000001011101110000000011010011000001010
011101101111010101001000011100000000001010000000010010011010001001110011110111001111000011101111100011001010001
10011100001010100011010001111010110000010100010110000011011101100100001110011100100001011111101000000011010100
100011110111111011100001101011011100000100001100110001111010111010001101001111100001011101011000111010011100101
1101001001110110110001100000101100011010001100011111110110101101110110111"
i=0
for y in range (0,MAX):
    for x in range (0,MAX):
        if(str[i] == '1'):
            pic.putpixel([x,y],(0, 0, 0))
        else:
            pic.putpixel([x,y],(255, 255, 255))
        i = i+1
pic.show()
pic.save("flag.png")
```

得到一张二维码，扫码得到flag

```
SCTF{(121.518549, 25.040854)}
```

25. 爆破hash

```
import hashlib
def md5(s):
    return hashlib.md5(s).hexdigest()
for i in range(1, 999999):
    if md5(str(i)).startswith('bf5ede'):
        print i
#找到md5前6位是bf5ede的值
```

26. 哈夫曼编码

```
Surprise
Wabby Wabby:
j 29
z 31
7 25
e 31
l 23
6 37
4 32
p 38
h 27
g 26
x 28
```


i 25
u 27
n 25
8 36
0 24
o 23
c 28
y 24
l 29
b 26
m 27
2 28
v 25
d 33
f 28
9 33
t 21
w 22
a 31
r 24
s 16
k 32
5 25
q 23
3 32
{ 1
- 4
} 1

Wabby Wabbo:

01111100010000110010100011110111101010011011011110100000110010111101000010010010001100001110010000011110011101
10111101100111110100000011101010000010110100100011110000000001010011010010100111011100100011000111000100101110
011000111001100110100110001010101000110111100011111111011100101110001010010111110000101101100100100100001011111
01011101110101111000101110110000110010110011010010100101111100111010100011000100100111011101111000110010
0101111110001111100001010011001001000010011101001010111110111110011101011101000000100100100011111110010001011
101010010011011100010111011010010010010110100001011111110010111111001100101001111111000100110010001001001111
01111101101100011010000100101101100001011010000100011010111101011100001100000100011111110000101000100101101111
000111100101101011001100010101100011001001111100101001111010010001100010111110110110110000110110101000110111
00010001010001010000000011010010100101001111110100101101100111101001010100101010100010101101001111000100
00110001000010101110011100011001011000010101110111011011011010001111111010011110011100110011
101111100111100101101101010100110001100100110101011110000011111110001111001110101001111010101111001111000
0100011111011111010001001111001100001000010000110010111110101011010110001110001001010000111000100101011001001001
0100010101101101001110000101111110101010110110110000010011000111000010001001101101101101100111000011000011010101
1110101011001010000110110010110001011011101000111100011001111011101100010011011000011101010110111101001111111
111000010001110000010010111110111100101101010111100011100011010100110001011111000011111101110011010100100011111
1101111111011001111110001110111101100101110001110110111100101010110011001110110011110001111010000011010101000
111110111011100101100100100001111101010011101111001100111000000101001010001111001000110010111110000001111111
1100000001111111101110111111001110100100000100000011011111010000000111101011101111011011001111011010111100
00101100011010001110001110000011101101110001000111101011001001000111001111001011010101111110011100100
000111011011010101101110111000001001100110111001000111001000000110001100101100001001000100011110101010001011
01111000000110101110011101001011100110111101101111000011110001100101000011110010001111000110011011100110101
110001010101110111111100101100101010001101110110110101010011101000110101100010001111011011100101011000
0010010000110110001110111011100110011011101000000101000001011110100000010000110011011011110100111010000001011011
010111010011011100000100111100011101001110001011111010101101110100110100110000110001101100101100010010110111
10000100100010111101000101110101001011101010000111011100000100101101011110010110001000111111001000000101
11001011101000110110111101110000010101100100001010100101001000101110100110010101010011111100000100110100
11110101001001001110010110100111011101100001111010000100111110001111100111101001110101101001110001000111110100
11100111101011111111111101101010000001010001001001111010011001101110101110110100000100111110111100100000101
0110001101100000101100010011111111101110101100001010111110111001011101111111001110111010010001110111101111
01001011110011000110011000010010110010011000100101111000011010000111011100110110100101001011100100110010111
101001000100111111100001011110101011000000111010100011101101011100110101001001110001110001001111110001000010011

```
011110100111011111000101111100110000110100010001010001100111000110010010110001110111001011010001100011100110111
0101010100101001110111010010011110101110101001101010101011110111010110100000111110011111101001101011110100010101
1111101011100101101101001100011001111101111100100111101101101110111111010111010100100101110111000011100001001000
0111000101011101001111100110011001011111101101001111010000100010000110111100000110101101110101100011100111111100
000111100100010110100101111111010101011100100000010100111110111001010001011010101010110110100010100011001110110101
011000110010101110111011110000001010000011110011010011000011111110100111011100100111000001101001110111100000101
01011000001000010000111011100001111110010010100111111101010110000000001110110100001011001001110011100000010111
0110000011011010101100101100011100111111001010111100101101110100001000110001110111001010011100001111100111000110
0111110110111101010101100100010101101000110000001000111111001100110111111010110010001111001100111110001110011
1000100110111001000100110110001100001001011111001111101111010100100011010100111001100010110011110001000110111101
000111010111010101111111000001111011011101111000001011110011001110001101011110111111010000001000111110010110001
11100011010111111011111101111101110101000110100100011100010111111010100011001100011101111110111110100001111011
1100101000111011101111110101011001110001011001000100111010010111100111110011111011100011101111110111001111000101
1001001101101010001110010101010101011000000101011100110111110011111001010011100001010111100111001101101111100110
111001111100100000000011110101100011111000110101001101100001010010010011101111111001000000010100111111110101100
00101000000111010010100111100101101100100100101110010111110
```

surprise message len: 1000

[参考Python|Huffman编码的python代码实现](#)

```

# 节点类
class Node(object):
    def __init__(self, name=None, value=None):
        self._name = name
        self._value = value
        self._left = None
        self._right = None

# 哈夫曼树类
class HuffmanTree(object):
    # 根据Huffman树的思想: 以叶子节点为基础, 反向建立Huffman树
    def __init__(self, char_weights):
        self.a = [Node(part[0], part[1]) for part in char_weights] # 根据输入的字符及其频数生成叶子节点
        while len(self.a) != 1:
            self.a.sort(key=lambda node: node._value, reverse=True)
            c = Node(value=(self.a[-1]._value + self.a[-2]._value))
            c._left = self.a.pop(-1)
            c._right = self.a.pop(-1)
            self.a.append(c)

        self.root = self.a[0]
        self.b = list(range(10))

    # 用递归的思想生成编码
    def pre(self, tree, length):
        node = tree
        if (not node):
            return
        elif node._name:
            x = str(node._name) + '的编码为:'
            for i in range(length):
                x += str(self.b[i])
            print(x)
            return
        self.b[length] = 0
        self.pre(node._left, length + 1)
        self.b[length] = 1
        self.pre(node._right, length + 1)

    # 生成哈夫曼编码
    def get_code(self):
        self.pre(self.root, 0)

if __name__ == '__main__':
    # 输入的是字符及其频数
    char_weights = [('j', 29), ('z', 31), ('7', 25), ('e', 31), ('l', 23), ('6', 37), ('4', 32), ('p', 38), ('h', 27), ('g', 26),
                    ('x', 28), ('i', 25), ('u', 27), ('n', 25), ('8', 36), ('0', 24), ('o', 23), ('c', 28), ('y', 24), ('1', 29), ('b', 26), ('m', 27), ('2',
                    28), ('v', 25), ('d', 33), ('f', 28), ('9', 33), ('t', 21), ('w', 22), ('a', 31), ('r', 24), ('s', 16), ('k', 32), ('5', 25), ('q', 23),
                    ('3', 32), ('{', 1), ('-', 4), ('}', 1)]
    tree = HuffmanTree(char_weights)
    tree.get_code()

```

运行脚本得到

0的编码为:00000
5的编码为:00001
i的编码为:00010
7的编码为:00011
v的编码为:00100
n的编码为:00101
b的编码为:00110
g的编码为:00111
m的编码为:01000
u的编码为:01001
h的编码为:01010
f的编码为:01011
2的编码为:01100
c的编码为:01101
x的编码为:01110
1的编码为:01111
j的编码为:10000
a的编码为:10001
e的编码为:10010
z的编码为:10011
3的编码为:10100
k的编码为:10101
4的编码为:10110
9的编码为:10111
d的编码为:11000
8的编码为:11001
6的编码为:11010
p的编码为:11011
t的编码为:111000
)的编码为:111001000
{的编码为:111001001
-的编码为:11100101
s的编码为:1110011
w的编码为:111010
q的编码为:111011
o的编码为:111100
l的编码为:111101
r的编码为:111110
y的编码为:111111

27. RAR

RAR文件格式

参考文章: <https://www.freebuf.com/column/199854.html>

RAR是有四个文件块组成的，分别是分别是标记块、归档头部块、文件块、结束块，这些块之间没有固定地先后顺序，但要求第一个块必须是标志块并且其后紧跟一个归档头部块。每个块都包含以下内容：

名称	大小 (字节)	描述	翻译
HEAD_CRC	2	CRC of total block or block part	整个块或者块某个部分的CRC (根据块类型而有不同)
HEAD_TYPE	1	Block type	块类型
HEAD_FLAGS	2	Block flags	块标志
HEAD_SIZE	2	Block size	块大小
ADD_SIZE	4	Optional field - added block size	添加块的大小 (可选字段)

归档头部块和文件块的内容较多，斗哥仅列出每个块头部内容：

块名称	标记块	归档头部块	文件块	结束块
HEAD_CRC	0x6152	从头域HEAD_TYPE到头域RESERVED2的CRC	文件名从头域HEAD_TYPE到头域FILEATTR的CRC	0x3DC4
HEAD_TYPE	0x72	0x73	0x74	0x7b
HEAD_FLAGS	0x1A21	位标志	位标志 (伪加密)	0x4000
HEAD_SIZE	0x0007	归档头部块大小, 包括归档注释	文件块大小, 包括文件名的内容	0x0007

RAR的标记块和结束块都是固定的7字节序列，分别为0x52 61 72 21 1A 07 00和0xC4 3D 7B 00 40 07 00。文件块这边要注意一下HEAD_FLAGS这个头部，其中HEAD_FLAGS的低三位代表加密标志，此位若被置为1，则文件使用了基于密钥的加密。

RAR伪加密

[参考L1near师傅的博客](#)

我们可以看到RAR第24字节这个地方，对应着一个PASSWORD_ENCRYPTED，所以如果某RAR文件没有被加密，那么这一行中的数字为0，将其修改为1

```
0000h: 52 61 72 21 1A 07 00 CF 90 73 00 00 0D 00 00 00 Rar!...s.....
0010h: 00 00 00 00 AD 20 74 24 94 36 00 C0 B4 01 00 96 .....(f$.6.....
0020h: BA 01 00 02 78 BB 78 F4 32 56 12 51 1D 33 09 00 ...x.x.2V.Q.3..
0030h: 20 00 00 00 67 69 61 61 2E 64 6F 63 78 26 8A 05 ...giao.docx&..
0040h: 8B 51 78 D1 93 00 F0 D0 0C 86 2D 58 7A 28 84 A3 .Qx.....?z...
0050h: 88 21 47 74 20 13 16 60 48 01 4E 16 08 0A 0E 08 100...&..8.....
0060h: 78 00 00 37 F6 14 82 20 3D 51 4C 08 4E 22 64 18 1...?.....&.8*d.
0070h: 07 76 0F 08 4A 08 10 38 78 0F 00 7A E5 A9 04 45 .ve.d...=...&...B
0080h: 2F 5B 0E BD 42 18 FF 31 FD 46 AF BE 87 E0 84 AD /1..B...3..F.....
0090h: AF 7C 37 97 92 26 03 08 2A A6 06 78 18 09 80 87 .17..&...?..&....
00A0h: 18 58 F4 82 08 5A 33 51 00 28 92 6E 04 3E 08 07 .x...238..+..6...
00B0h: 70 3D 28 90 05 80 03 07 80 39 A7 2A 98 8D A9 29 p=&.....&..?..1
00C0h: 54 25 32 2A 82 08 18 04 00 2A 55 00 58 05 9E 4E 9&?.....0...?..B
00D0h: 7C 8C 08 18 54 51 80 07 80 08 01 80 89 17 0A 68 11..?.....?..k
00E0h: FF 00 F9 AC 6A 99 04 20 38 38 51 50 88 08 93 0B .....3...?..?..P
00F0h: AF 03 0A 06 46 08 74 03 98 39 15 F1 8E A6 02 8B .....&...&...1..
0100h: 8A A1 82 88 38 00 13 78 47 53 86 31 13 76 D8 F0 .....?..&...1..1..
0110h: 87 81 88 80 87 87 85 08 28 51 90 78 18 FA 04 88 .....1Q0?.....
0120h: 5F 86 C7 AF 72 4B 4F 51 AF 26 78 8F 0A 89 AD E5 ...&MMQ..k?.....
0130h: 1E 95 F3 C9 48 98 85 C3 4C 28 AF 75 E2 40 87 F0 ...1..&..&...&..
0140h: 87 48 11 78 28 0E 8D A5 28 8C 7E 78 80 11 35 A7 .B..?.....&..&..
```

模板结果 - RAR.bt

名称	值	开始	大小	颜色	注释
struct RarBlock Marker		0h	7h	Fg: Bg:	
struct RarBlock ArcHeader		7h	Dh	Fg: Bg:	
uint16 HEAD_CRC	90CFh	7h	2h	Fg: Bg:	
enum RarBlockType HeadType	ARCHIVE (115)	9h	1h	Fg: Bg:	
struct MainHeadFlags HEAD_FLAGS		Ah	2h	Fg: Bg:	
uint16 HeaderSize	13	Ch	2h	Fg: Bg:	
uint16_reserved1	0	Eh	2h	Fg: Bg:	
uint32_reserved2	0	10h	4h	Fg: Bg:	
struct RarBlock block[0]		14h	1B4F6h	Fg: Bg:	
uint16 HEAD_CRC	28ADh	14h	2h	Fg: Bg:	
enum RarBlockType HeadType	FILE_OR_DIR (1...	16h	1h	Fg: Bg:	
struct FileHeadFlags HEAD_FLAGS		17h	2h	Fg: Bg:	
ubyte from_PREV_VOLUME : 1	0	17h	1h	Fg: Bg:	
ubyte to_NEXT_VOLUME : 1	0	17h	1h	Fg: Bg:	
ubyte PASSWORD_ENCRYPTED : 1	1	17h	1h	Fg: Bg:	
ubyte FILE_COMMENT_PRESENT : 1	0	17h	1h	Fg: Bg:	

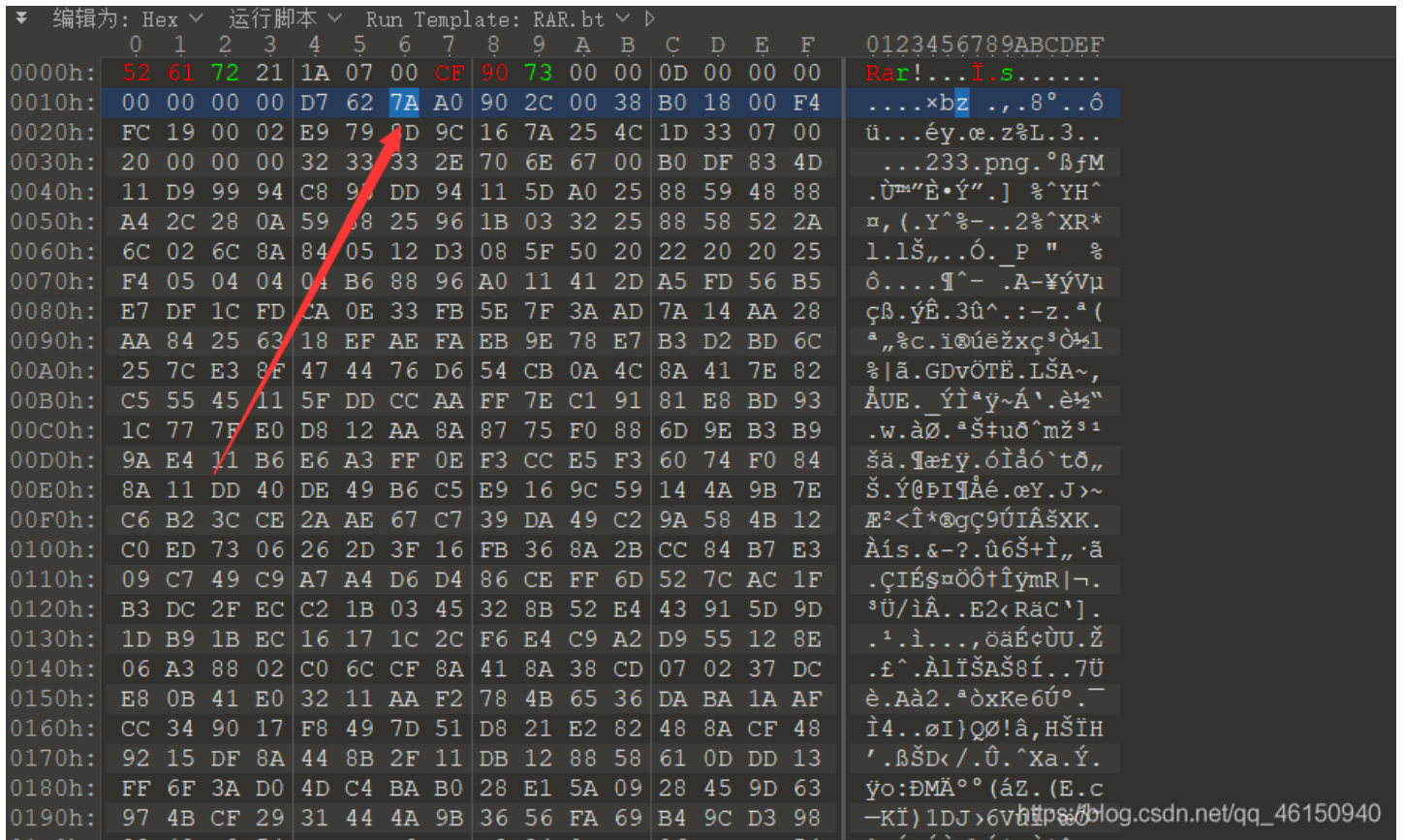
https://blog.csdn.net/qq_46150940

将其从1改为0就可以正常打开文件了。

也可以直接winhex或者010editor打开，把第24字节的后面一个4改成0也可以。（即从24变成20）

RAR文件受损

分析RAR文件结构，文件块的HEAD_TYPE应该是0x74而不是0x7A，修改为74后保存。



28. 凯撒密码爆破脚本

```

s = "Hello_Atkx"

def kaisa(k):
    t = ""
    for c in s:
        if 'a' <= c <= 'z':
            t += chr(ord('a') + ((ord(c) - ord('a')) + int(k)) % 26)
        elif 'A' <= c <= 'Z':
            t += chr(ord('A') + ((ord(c) - ord('A')) + int(k)) % 26)
        else:
            t += c
    print(t)

for i in range(0,26):
    kaisa(i)
    
```

运行结果:


```
Hello_Atkx  
Ifmmp_Buly  
Jgnnq_Cvmz  
Khoor_Dwna  
Lipps_Exob  
Mjqqt_Fypc  
Nkrru_Gzqd  
Olssv_Hare  
Pmttw_Ibsf  
Qnuux_Jctg  
Rovvy_Kduh  
Spwvz_Levi  
Tqxxa_Mfwj  
Uryyb_Ngkx  
Vszzc_Ohy1  
Wtaad_Pizm  
Xubbe_Qjan  
Yvccf_Rkbo  
Zwddg_Slcp  
Axeeh_Tmdq  
Byffi_Uner  
Czggj_Vofs  
Dahhk_Wpgt  
Ebiil_Xqhu  
Fcjfm_Yriv  
Gdkkn_Zsjw
```

29. USB流量分析

详见之前写过的[USB流量分析](#)