

# 网鼎杯第一场wp

转载

[weixin\\_30482181](#) 于 2018-08-21 20:41:00 发布 255 收藏 1

文章标签: [运维](#)

原文链接: <http://www.cnblogs.com/liang2580/p/9513850.html>

版权

网鼎杯第一场WriteUP—china H.L.B 团队

文章地址: <https://www.o2oxy.cn/1661.html>

## 1、签到题

青龙之战

📌 签到

- \* 欢迎来答题
- \* 推荐视频《密码学基础概述》: <https://www.ichunqiu.com/course/63762>
- \* 回复正确选项的[ ]内字符串进入下一题
- \* 提示: 请以正确格式回复, 例如[aaa111]。如果以错误格式回复, 例如[aaa111]、[aaa111]答案, 则无法进入下一题。

(一)在数字签名中, 签名值的长度与被签名消息的长度有关。  
[ca02f7]正确  
[1f5f2e]错误

回复 1f5f2e 进入下一关

1f5f2e

📌 签到

(二)弱碰撞自由的Hash函数比强碰撞自由的Hash函数的安全性高。  
[3c47f0]正确  
[4c361b]错误

4c361b

📌 签到

(三)1949年, ( ) 发表题为《保密系统的通信理论》的文章, 为密码系统建立了理论基础, 从此密码学成了一项科学。  
[7642a4]Shannon  
[716f94]Diffie  
[a930ab]Hellman  
[6fcb56]Shamir

7642a4

(四)下列密码体制可以抗量子攻击的是( )。

- [6f9c31]ECC
- [7bd470]RSA
- [4ef7e8]AES
- [aac333]NTRU

aac333

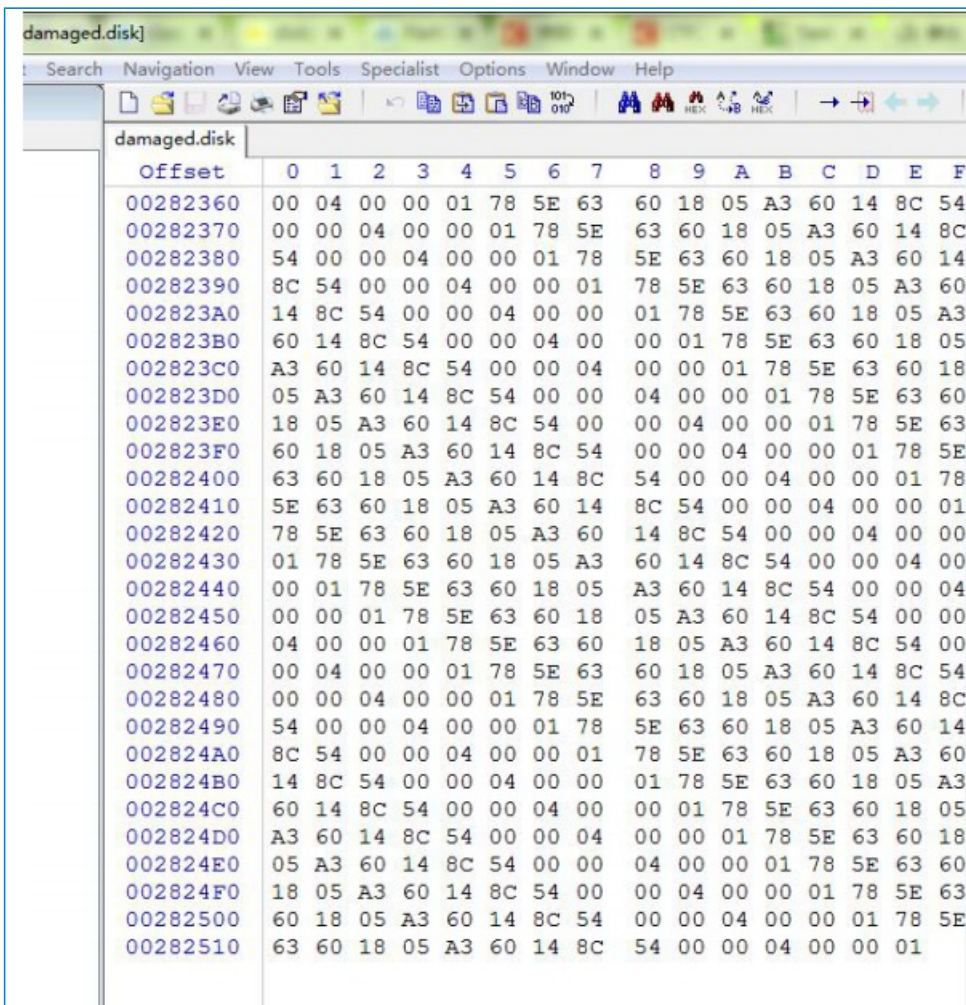


\* flag{hello\_wangdingbei}  
\* 想了解更多精彩赛事,请关注“永信至诚 (INT-GROUP)” 公众号

## 2、Clip

下载题目是Disk 文件。第一反应是linux虚拟硬盘。

用winhex 打开如图:



在winhex中第196280 发现了png的头文件如图:

```
00196280 00 38 61 AD 49 78 01 01 00 04 FF FB 89 50 4E 47 8a-Ix yûPNG
00196290 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 03 4F IHDR 0
```

png 16进制文件头以 89504E47开头

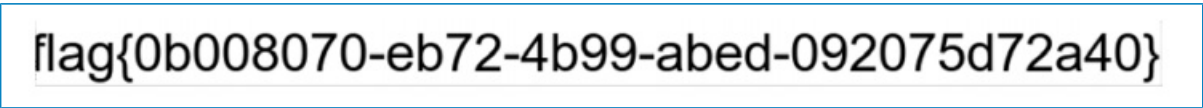
第一张图片:



第二张图片

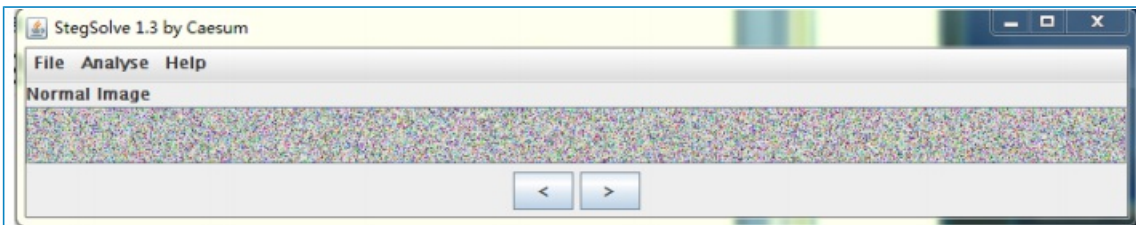


使用PS拼接如下如:

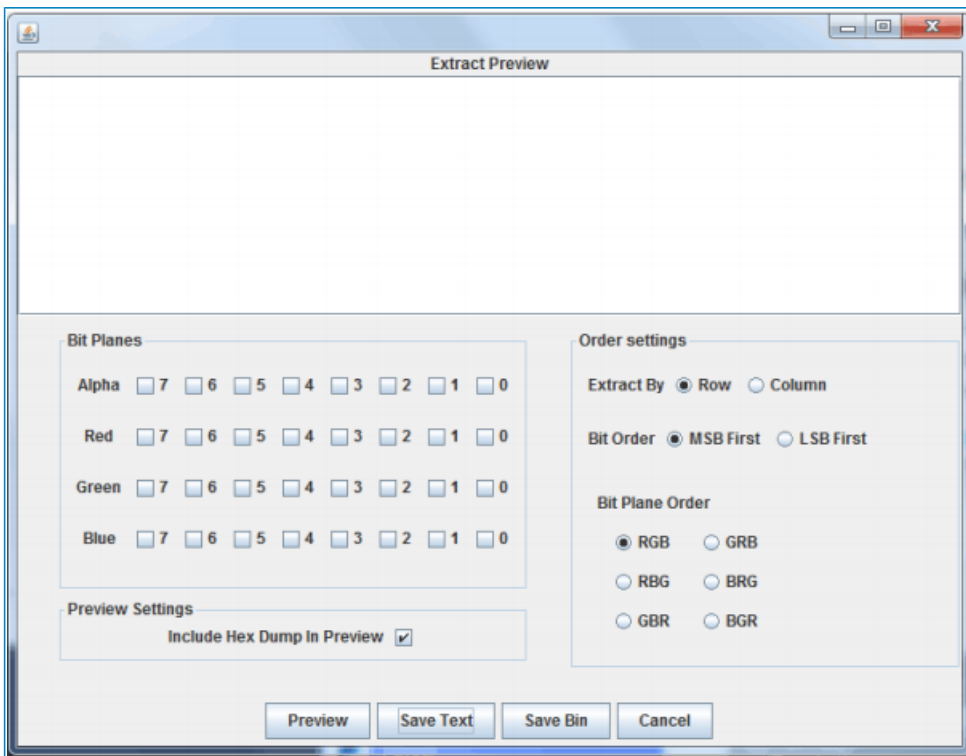


### 3. minified

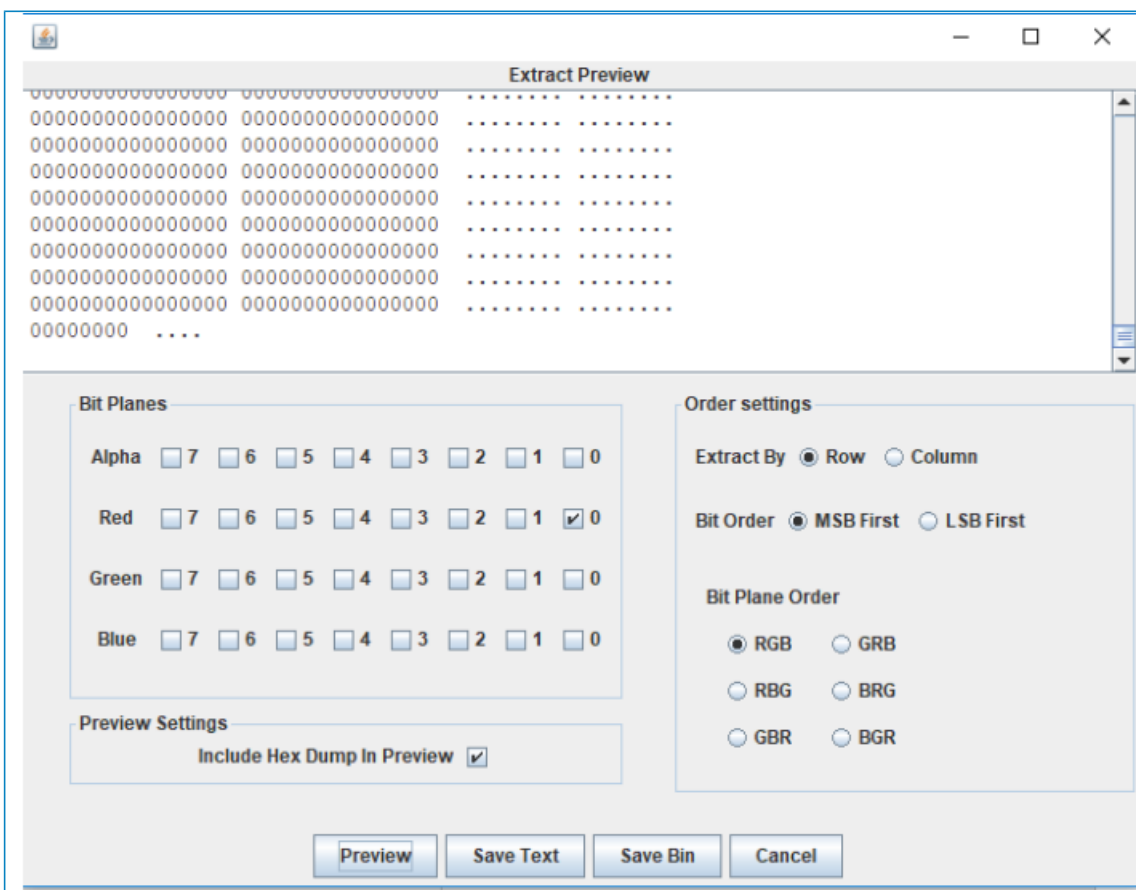
用Stegsolve 打开图片



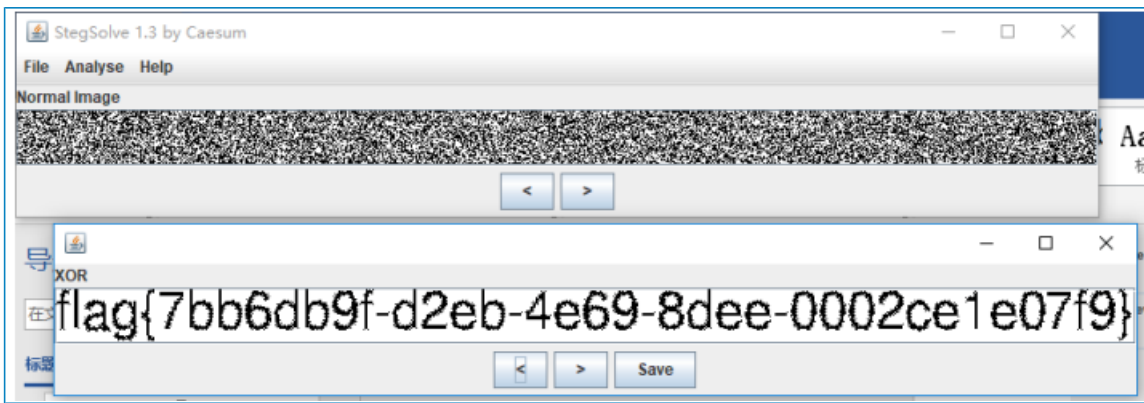
打开 Stegsolve 选择Data Extract 查看图片通道, 如图,



选择0 通道发现是LSB 隐写。



分别把 alpha green 和blue 的0通道另存为 再进行异或处理 最终在alpha 和green 中发现flag



#### 4、beijing

题目给了我们一个linux 下可执行的程序、放在虚拟机执行如下：

```
listennter@ubuntu:~/Desktop/studying$ ls -l beijing
-rwxr-xr-x 1 listennter listennter 5708 Aug 17 09:06 beijing
listennter@ubuntu:~/Desktop/studying$ ./beijing
R[09]>-@-@,o[10]>@,>
listennter@ubuntu:~/Desktop/studying$
```

拖到IDA 里面进行尝试分析一下程序的逻辑、经过分析程序主要处理的两个函数、主要逻辑如下”

main 函数21次调用了encode 函数 然后将返回的结果按照字符打印如下

```

22 char v19; // a1
23 char v20; // a1
24
25 v0 = encode(dword_804A03C);
26 printf("%c", v0);
27 fflush(stdout);
28 v1 = encode(dword_804A044);
29 printf("%c", v1);
30 fflush(stdout);
31 v2 = encode(dword_804A0E0);
32 printf("%c", v2);
33 fflush(stdout);
34 v3 = encode(dword_804A050);
35 printf("%c", v3);
36 fflush(stdout);
37 v4 = encode(dword_804A058);
38 printf("%c", v4);
39 fflush(stdout);
40 v5 = encode(dword_804A0E4);
41 printf("%c", v5);
42 fflush(stdout);
43 v6 = encode(dword_804A064);
44 printf("%c", v6);
45 fflush(stdout);
46 v7 = encode(dword_804A0E8);
47 printf("%c", v7);
48 fflush(stdout);
49 v8 = encode(dword_804A070);
50 printf("%c", v8);
51 fflush(stdout);

```

encode 函数按照a1 的数值进行对饮的亦或运算、返回char 类型结果

```

1 int __cdecl encode(int a1)
2 {
3     char v2; // [esp+Fh] [ebp-1h]
4
5     switch ( a1 )
6     {
7     case 0:
8         v2 = byte_804A021 ^ byte_804A020;
9         break;
10    case 1:
11        v2 = byte_804A023 ^ byte_804A022;
12        break;
13    case 2:
14        v2 = byte_804A025 ^ byte_804A024;
15        break;
16    case 3:
17        v2 = byte_804A027 ^ byte_804A026;
18        break;
19    case 4:
20        v2 = byte_804A029 ^ byte_804A028;
21        break;
22    case 5:
23        v2 = byte_804A02B ^ byte_804A02A;
24        break;
25    case 6:
26        v2 = byte_804A02D ^ byte_804A02C;
27        break;
28    case 7:
29        v2 = byte_804A02F ^ byte_804A02E;
30        break;

```

查看或部分对饮的数据段和对应的hex 数据

```
.data:0804A020 byte_804A020 db 61h ; DATA XREF: encode:loc_804848C↑r
.data:0804A021 byte_804A021 db 4Ch ; DATA XREF: encode+33↑r
.data:0804A022 byte_804A022 db 67h ; DATA XREF: encode:loc_80484A6↑r
.data:0804A023 byte_804A023 db 59h ; DATA XREF: encode+4D↑r
.data:0804A024 byte_804A024 db 69h ; DATA XREF: encode:loc_80484C0↑r
.data:0804A025 byte_804A025 db 29h ; DATA XREF: encode+67↑r
.data:0804A026 byte_804A026 db 6Eh ; DATA XREF: encode:loc_80484DA↑r
.data:0804A027 byte_804A027 db 42h ; DATA XREF: encode+81↑r
.data:0804A028 byte_804A028 db 62h ; DATA XREF: encode:loc_80484F4↑r
.data:0804A029 byte_804A029 db 0Dh ; DATA XREF: encode+9B↑r
.data:0804A02A byte_804A02A db 65h ; DATA XREF: encode:loc_804850E↑r
.data:0804A02B byte_804A02B db 71h ; DATA XREF: encode+85↑r
.data:0804A02C byte_804A02C db 66h ; DATA XREF: encode:loc_8048528↑r
.data:0804A02D byte_804A02D db 34h ; DATA XREF: encode+CF↑r
.data:0804A02E byte_804A02E db 6Ah ; DATA XREF: encode:loc_8048542↑r
.data:0804A02F byte_804A02F db 0C6h ; DATA XREF: encode+E9↑r
.data:0804A030 byte_804A030 db 6Dh ; DATA XREF: encode:loc_804855C↑r
.data:0804A031 byte_804A031 db 8Ah ; DATA XREF: encode+103↑r
.data:0804A032 byte_804A032 db 6Ch ; DATA XREF: encode:loc_8048576↑r
.data:0804A033 byte_804A033 db 7Fh ; DATA XREF: encode+11D↑r
.data:0804A034 byte_804A034 db 7Bh ; DATA XREF: encode:loc_8048590↑r
.data:0804A035 byte_804A035 db 0AEh ; DATA XREF: encode+137↑r
.data:0804A036 byte_804A036 db 7Ah ; DATA XREF: encode:loc_80485AA↑r
.data:0804A037 byte_804A037 db 92h ; DATA XREF: encode+151↑r
.data:0804A038 byte_804A038 db 7Dh ; DATA XREF: encode:loc_80485C4↑r
.data:0804A039 byte_804A039 db 0ECh ; DATA XREF: encode+16B↑r
.data:0804A03A byte_804A03A db 5Fh ; DATA XREF: encode:loc_80485DE↑r
.data:0804A03B byte_804A03B db 57h ; DATA XREF: encode+185↑r
.data:0804A03C dword_804A03C dd 6 ; DATA XREF: main+10↑r
```

数据段数据hex数据

```
0804A020 61 4C 67 59 69 29 6E 42 62 0D 65 71 66 34 6A C6 aLgYi)nBb.eqf4j.
0804A030 6D 8A 6C 7F 7B AE 7A 92 7D EC 5F 57 06 00 00 00 m.l.{.z.}.....
-----
```

这里可以看到这段数据大部分都是可见字符、因此可以假设flag 就在这段数据中、但是顺序是被打乱的、而正确的main 函数的中的顺序

即

encode :

```
return flag[i]^xor[i]
```

main:

```
list[] ← 记录正确的flag打印顺序
```

```
print encode(list[i])
```

按照上面的理论 、可以得到如下的分组:

```
flag = ['a','g','i','n','b','e','f','j','m','l','{','z','}','_']
xor = ['L','Y','B','','q','4','','','','',''] #不可见字符没有打印出来
list = [6, 9, 0, 1, 0xa, 0, 8, 0, 0xb, 2, 3, 1, 0xd, 4, 5, 2, 7, 2, 3, 1, 0xc]
...
```

最后运算脚本:

```
python:
```

```
result=""
```

```
for i in range(0,20):
```

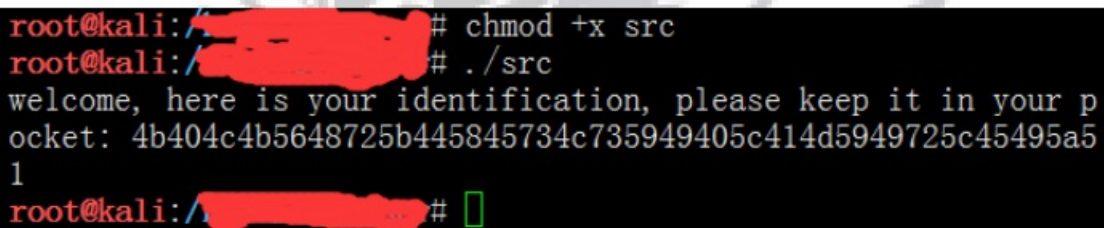
```
    result +=flag[list[i]]
```

```
print result
```

Flag: flag{amazing\_beijing}

## 5、advanced

把题目放入linux 中尝试运行了一下:



```
root@kali:~/... # chmod +x src
root@kali:~/... # ./src
welcome, here is your identification, please keep it in your pocket: 4b404c4b5648725b445845734c735949405c414d5949725c45495a51
root@kali:~/... #
```

得到的数值进行ASCII 转换如下



## ASCII在线转换器-十六进制，十进制、二进制

ASCII转换到 ASCII (例: a b c)

```
K@LKVHr [DXEsLsYI@\AMYIr\EIZQ
```

添加空格

删除空格

将空白字符转换

十六进制转换到十六进制 (例: 0x61或61或61/62)  删除 0

```
0x4b0x400x4c0x4b0x560x480x720x5b0x440x580x450x730  
x4c0x730x590x490x400x5c0x410x4d0x590x490x720x5c0x  
450x490x5a0x51
```

十进制转换到 十进制 (例: 97 98 99)

```
7564767586721149168886911576115897364926577897311  
49269739081
```

二进制转换到 二进制 (例: 01100001 01100010 01100011)

```
01001011010000000100110001001011010101100100100  
00111001001011011010001000101100001000101011100  
11010011000111001101011001010010010100000001011  
10001000001010011010101100101001001011100100101
```

解密得到内容使用脚本得到flag如下:

Flag{d\_with\_a\_template\_phew}

PDF 下载: <http://www.o2oxy.cn/wp-content/uploads/2018/08/China-H.L.B-网鼎杯部分WriteUp.pdf>

转载于:<https://www.cnblogs.com/liang2580/p/9513850.html>