网鼎杯第一场wp

转载

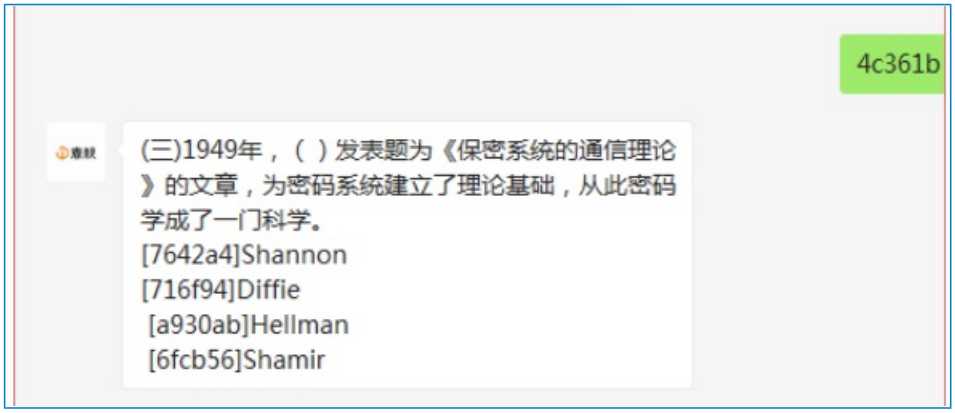
weixin_30482181 于 2018-08-21 20:41:00 发布 255 收藏 1
文章标签： 运维
原文链接：http://www.cnblogs.com/liang2580/p/9513850.html

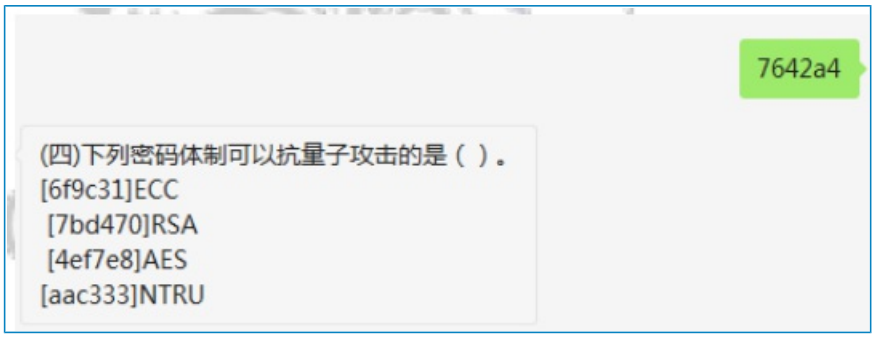
网鼎杯第一场WriteUP—china H.L.B 团队

文章地址：https://www.o2oxy.cn/1661.html

1、签到题

回复1f5f2e进入下一关

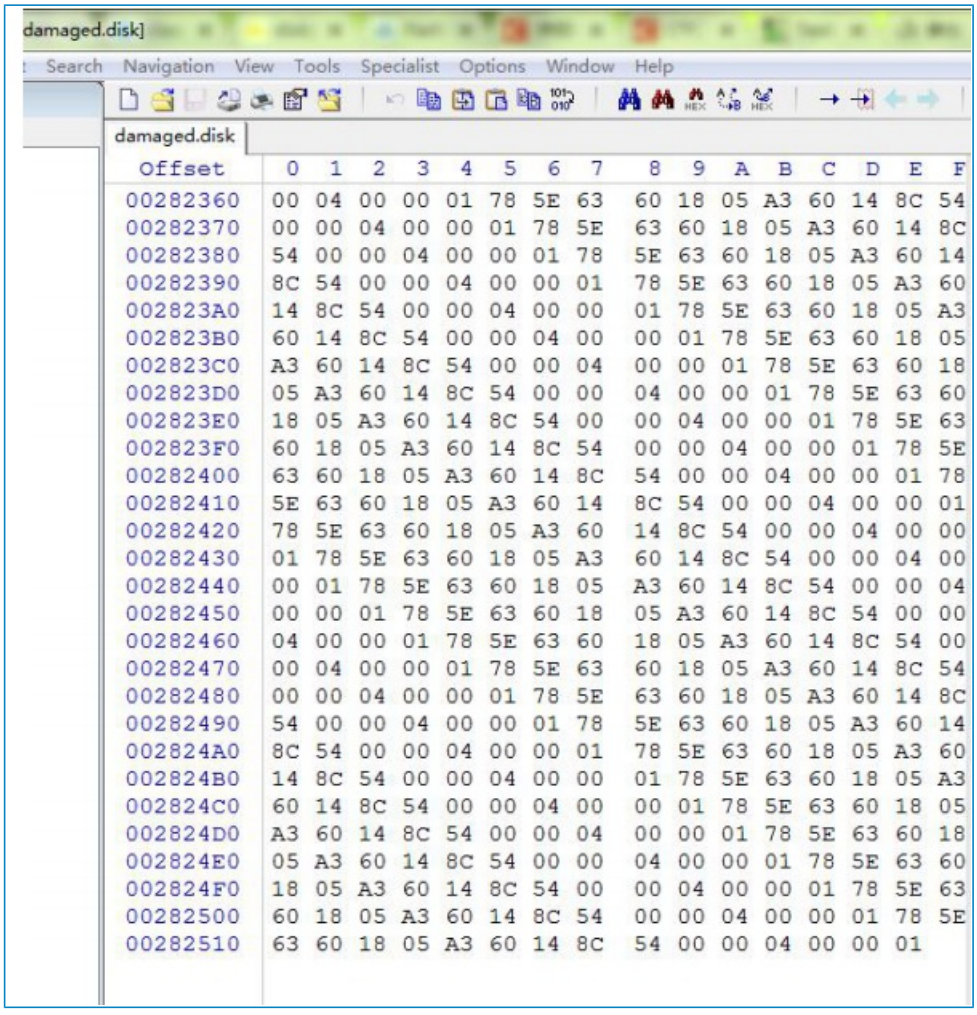




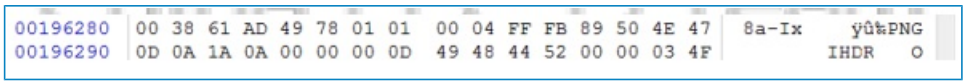
2、Clip

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：

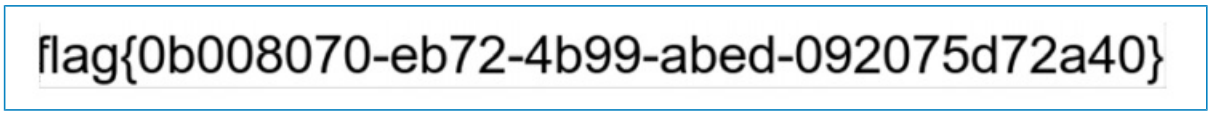


png 16进制文件头以 89504E47开头

第一张图片：

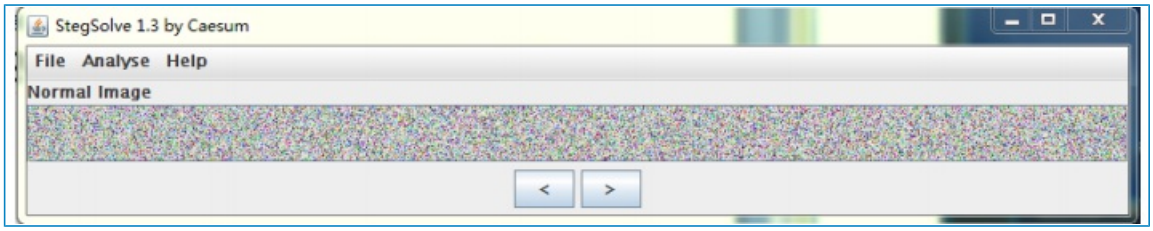
第二张图片

使用PS拼接如下如：

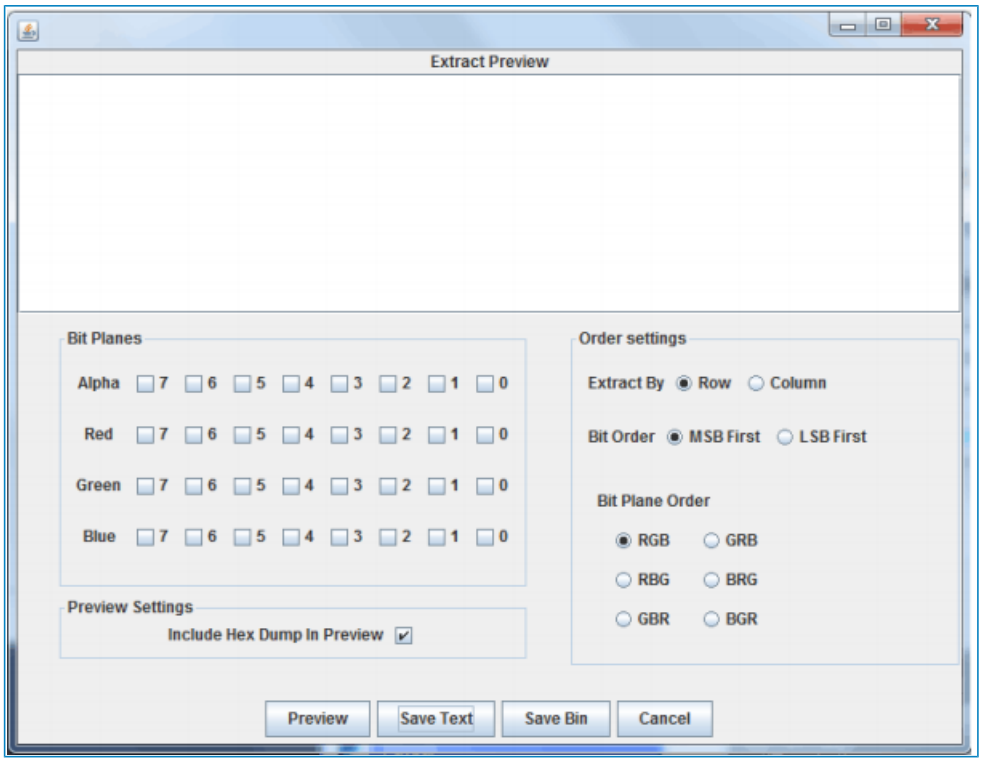


3. minified

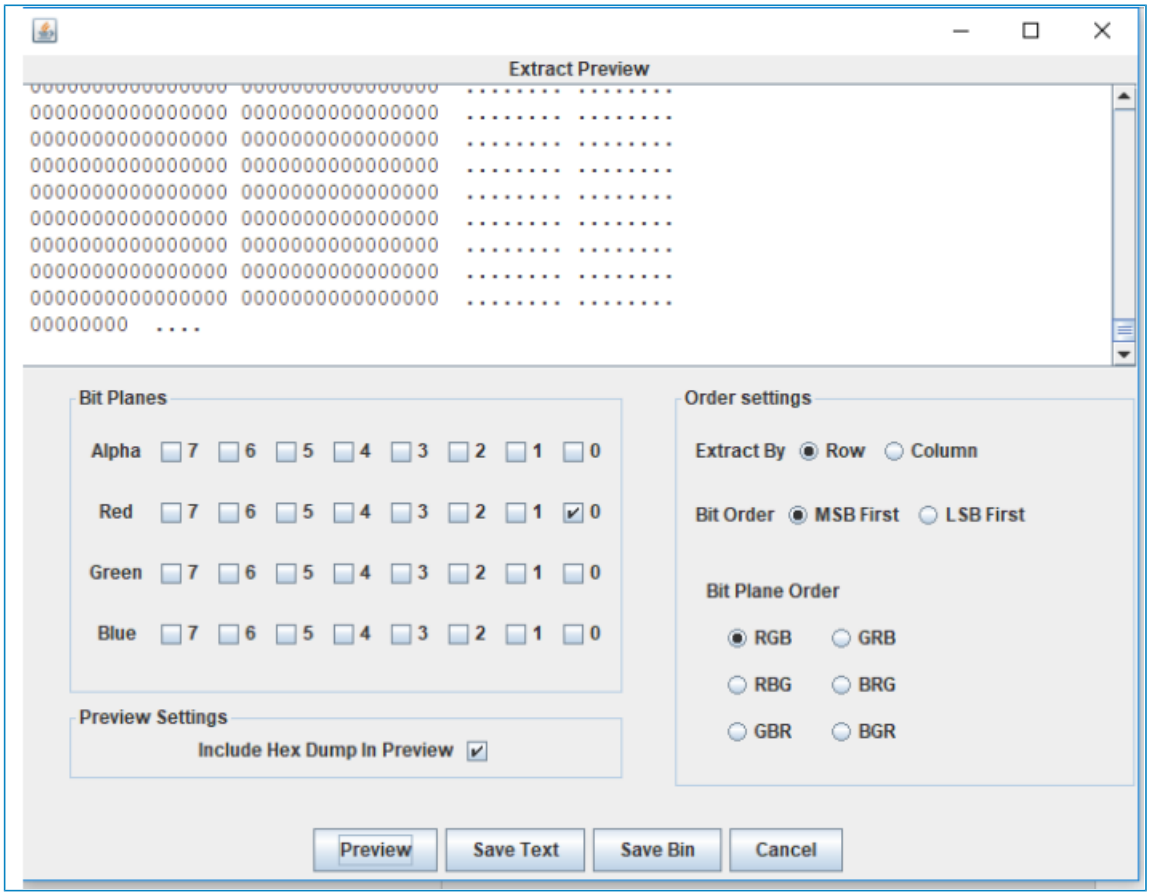
用Stegsolve 打开图片



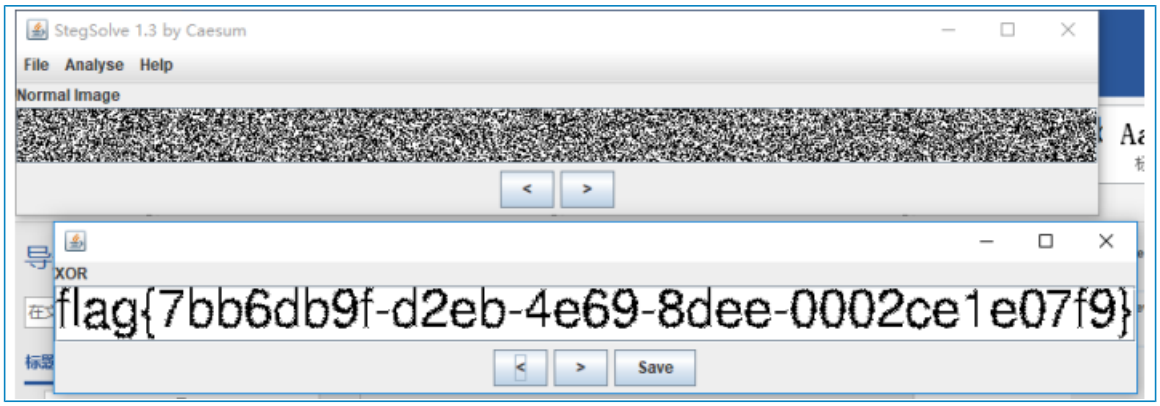
打开 Stegsolve 选择Data Extract 查看图片通道，如图，



选择0 通道发现是LSB 隐写。

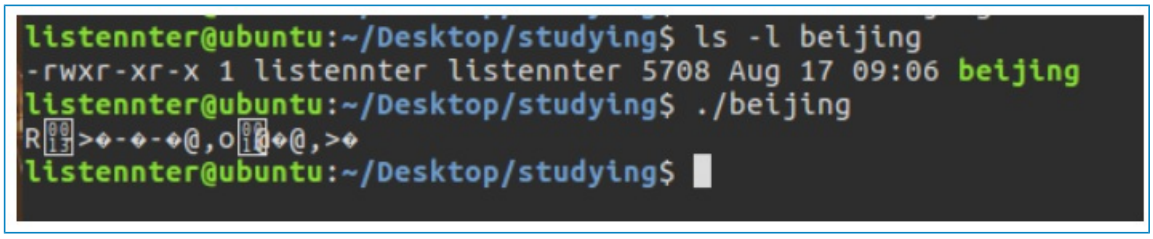


分别把 alpha green 和blue 的0通道另存为 再进行异或处理 最终在alpha 和green 中发现flag



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：



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; // al
23  char v20; // al
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+B5↑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):

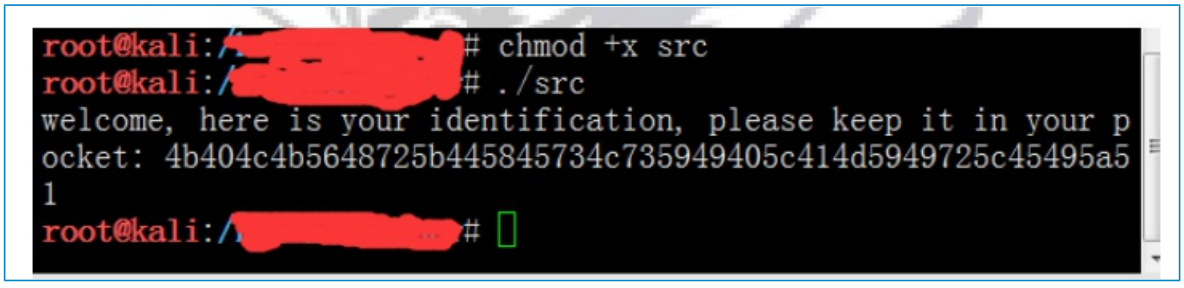
result +=flag[list[i]]

print result

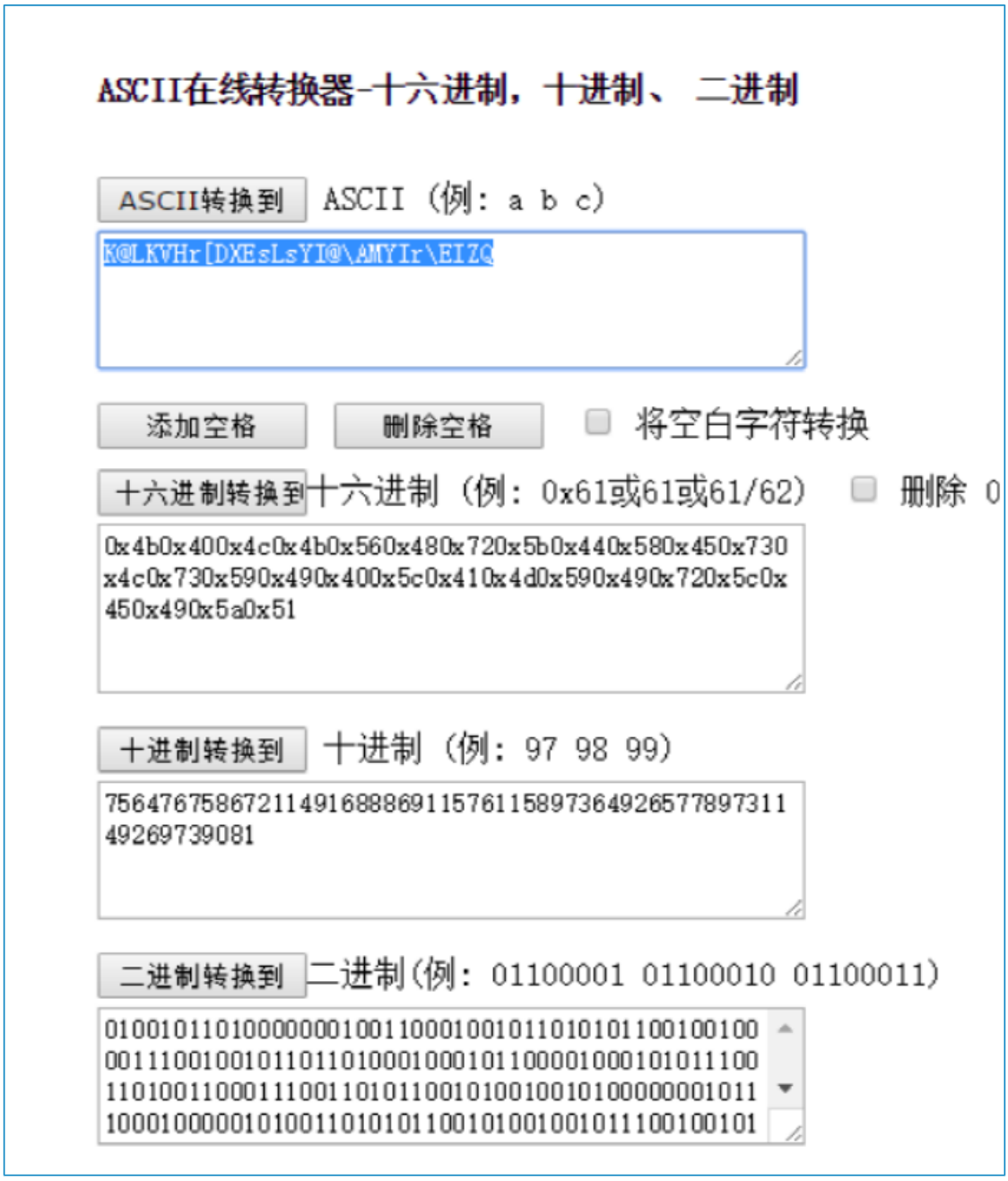
Flag: flag{amazing_beijing}

5、advanced

把题目放入linux 中尝试运行了一下：



得到的数值进行ASCII 转换如下



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