XCTF easyCpp

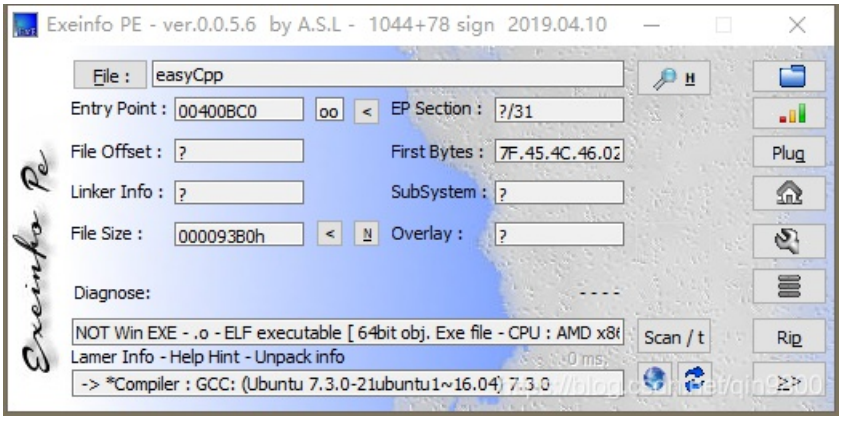
原创

夏了茶糜 于 2020-03-18 12:05:19 发布 283 收藏 1

分类专栏： CTF-REVERSE 文章标签： 安全

本文链接：https://blog.csdn.net/qin9800/article/details/104939043

查壳



无壳，64位Linux程序

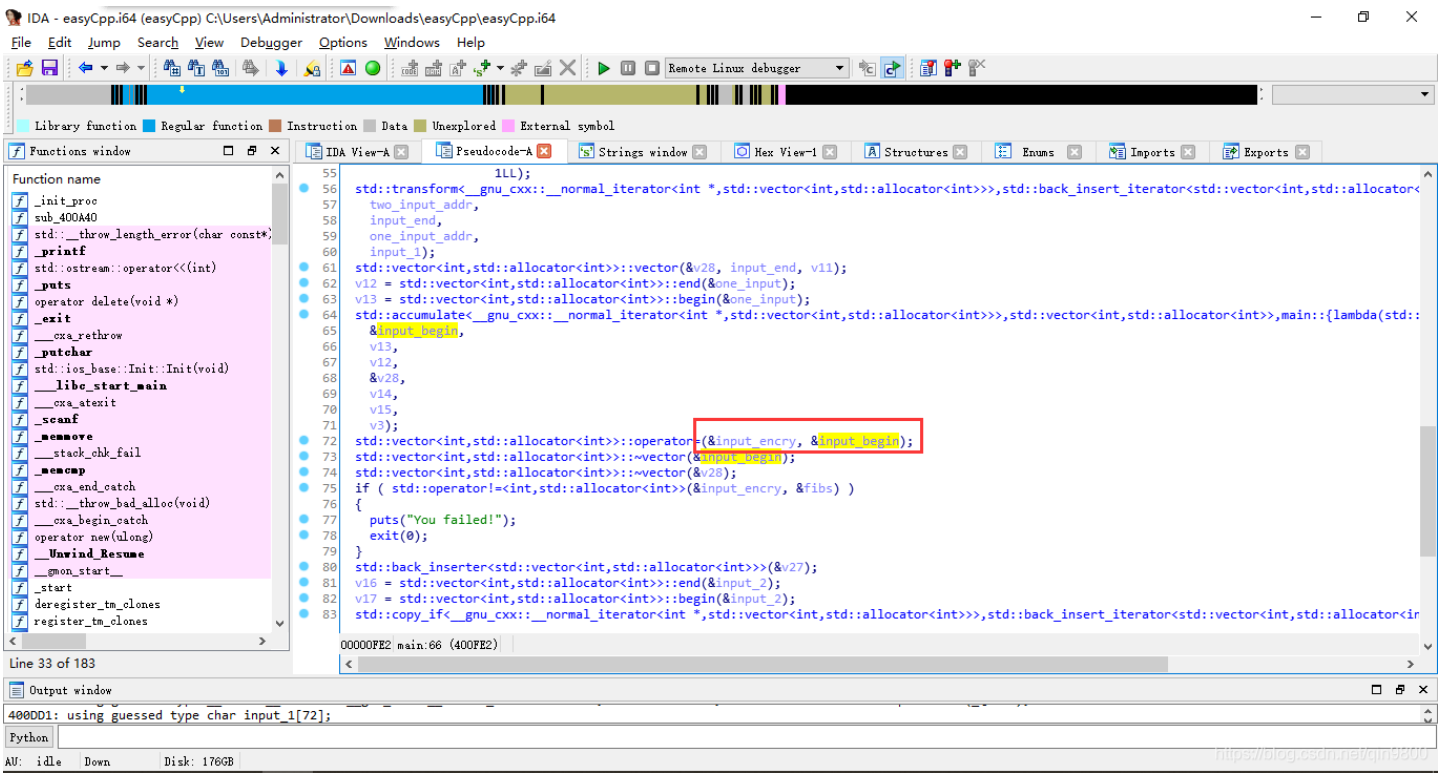
放入IDA中分析

查看Main函数

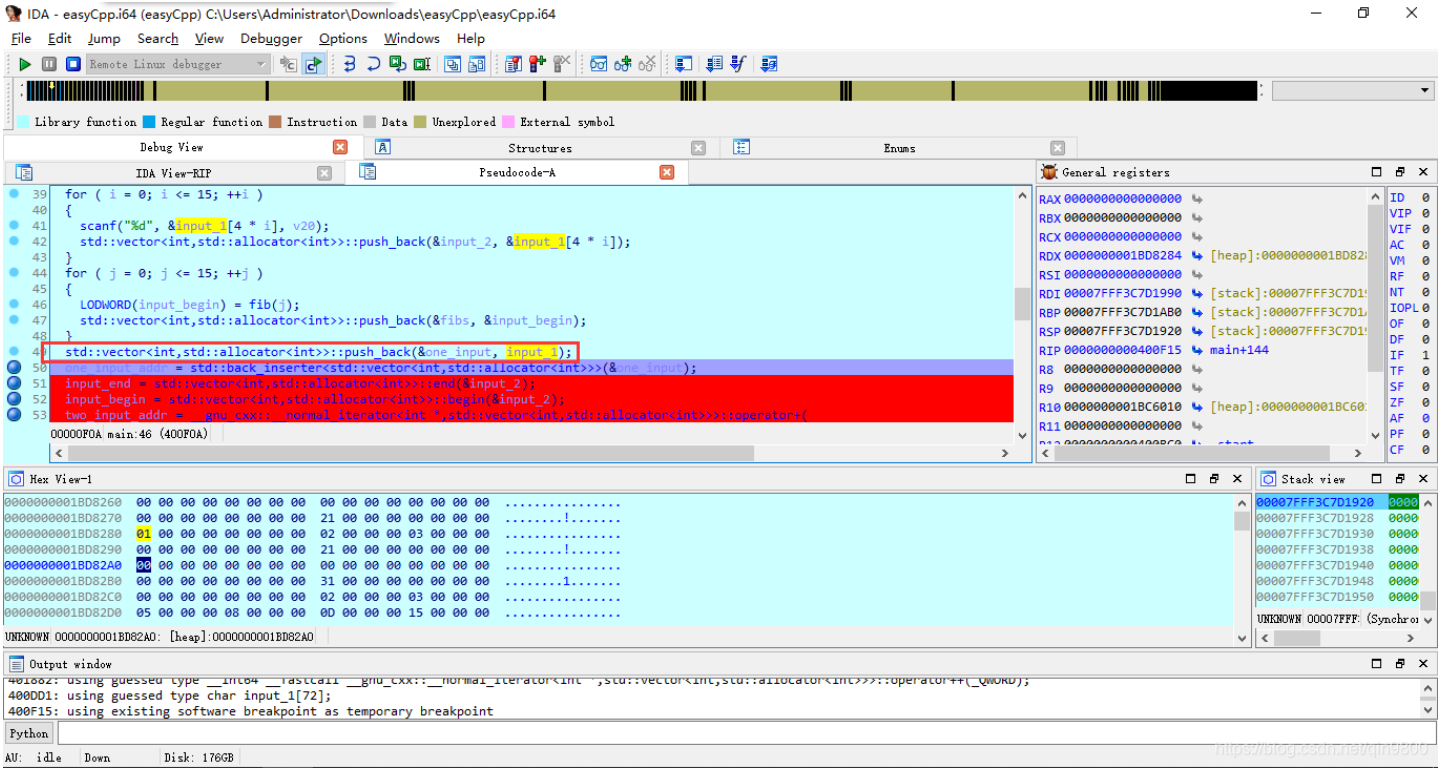
```
std::accumulate<__gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>,std::vector<int,std::alloca
  &input_begin,
  v13,
  v12,
  &v28,
  v14,
  v15,
  v3);
std::vector<int,std::allocator<int>>::operator=(&input_encry, &input_begin);
std::vector<int,std::allocator<int>>::~vector(&input_begin);
std::vector<int,std::allocator<int>>::~vector(&v28);
if ( std::operator!=<int,std::allocator<int>>(&input_encry, &fibs) )
{
  puts("You failed!");
  exit(0);
}
std::back_inserter<std::vector<int,std::allocator<int>>>(&v27);
v16 = std::vector<int,std::allocator<int>>::end(&input_2);
v17 = std::vector<int,std::allocator<int>>::begin(&input_2);
std::copy_if<__gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>,std::back_insert_iterator<std:
puts("You win!");
printf("Your flag is:flag{", v16, v20);
v28 = std::vector<int,std::allocator<int>>::begin(&v27);
input_begin = std::vector<int,std::allocator<int>>::end(&v27);
while ( __gnu_cxx::operator!=<int *,std::vector<int,std::allocator<int>>>(&v28, &input_begin) )
{
  v18 = __gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>::operator*(&v28);
  std::ostream::operator<<(&std::cout, *v18);
  __gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>::operator++(&v28);
```

```
0000101C main:78 (40101C)
```

这里拿加密后的用户输入和程序生成的前16个斐波那契数列进行对比
，不相等则退出程序，相等的话则继续往下执行
所以说只需要逆推出input_encry是如何生成的，既可以得到flag

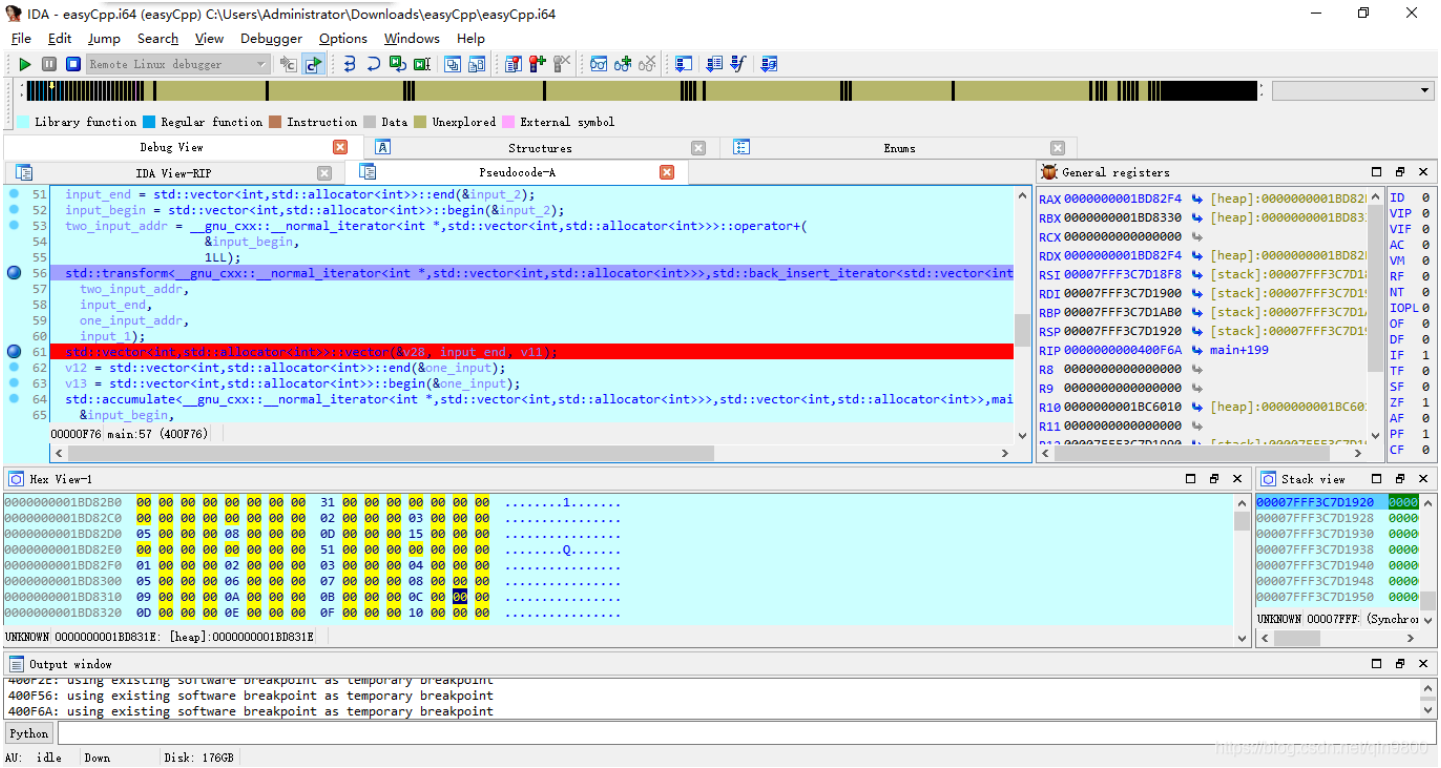


这里是把&input_begin存到&input_encry
从头开始分析&input_begin是如何生成的

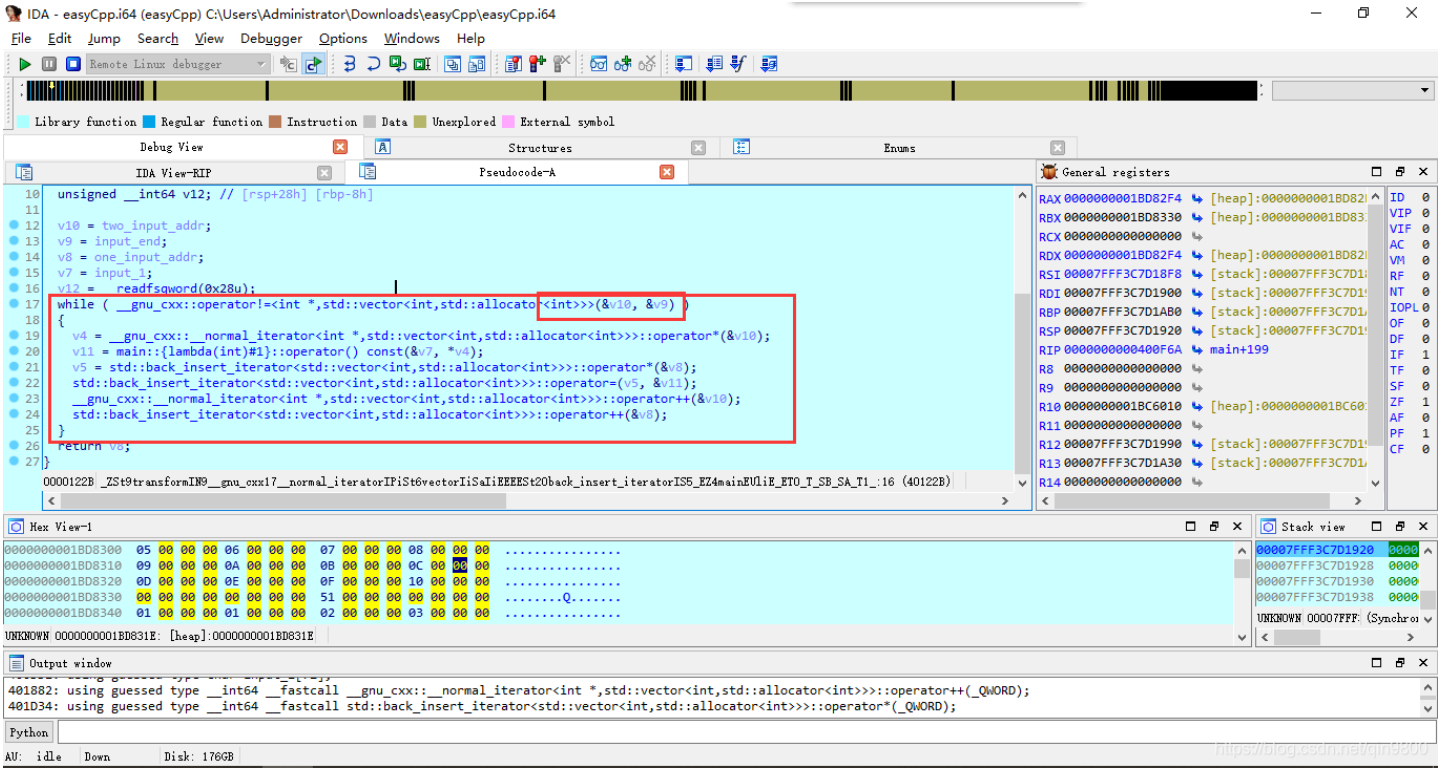


这里是把用户输入的字符串首地址放到&one_input中，
one_input_addr也执指向用户输入的字符串首地址

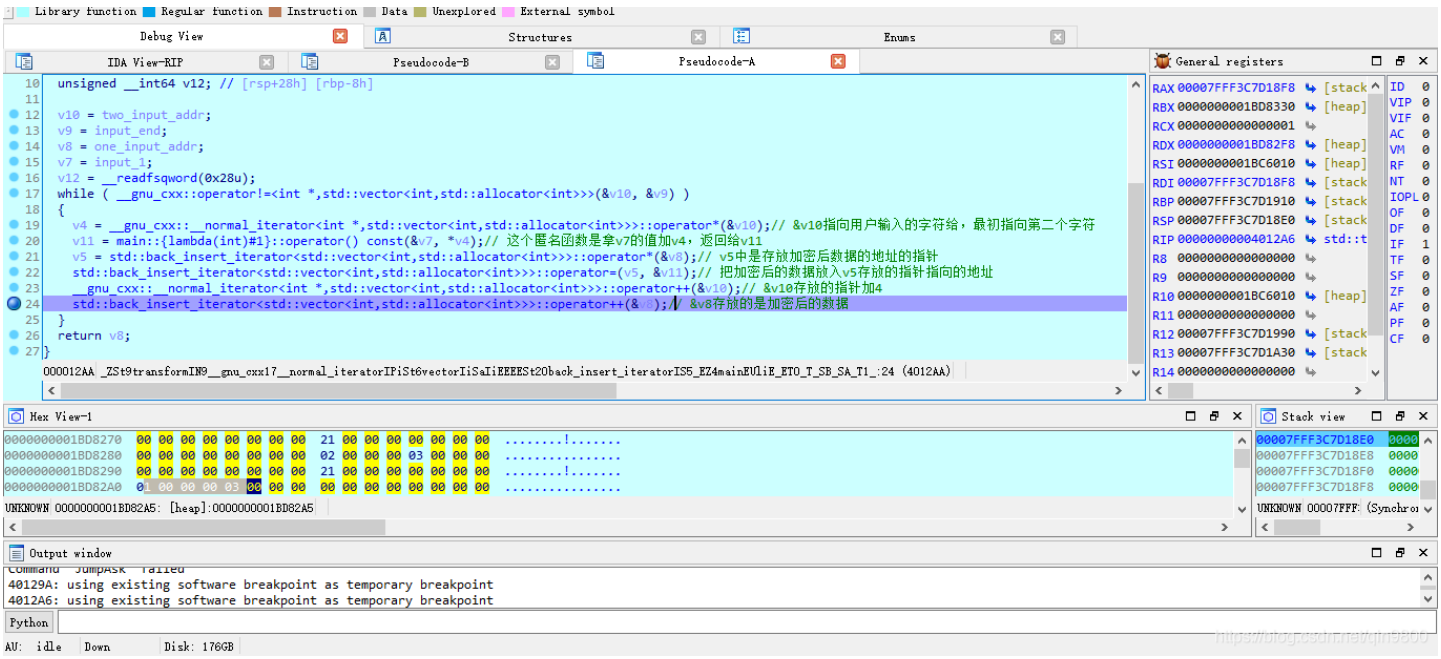
input_end指向用户输入数据的最后一个的地址
input_begin指向用户输入数据的第一个的地址
two_input_addr指向用户输入数据的第二个的地址



跟入transform函数分析函数作用

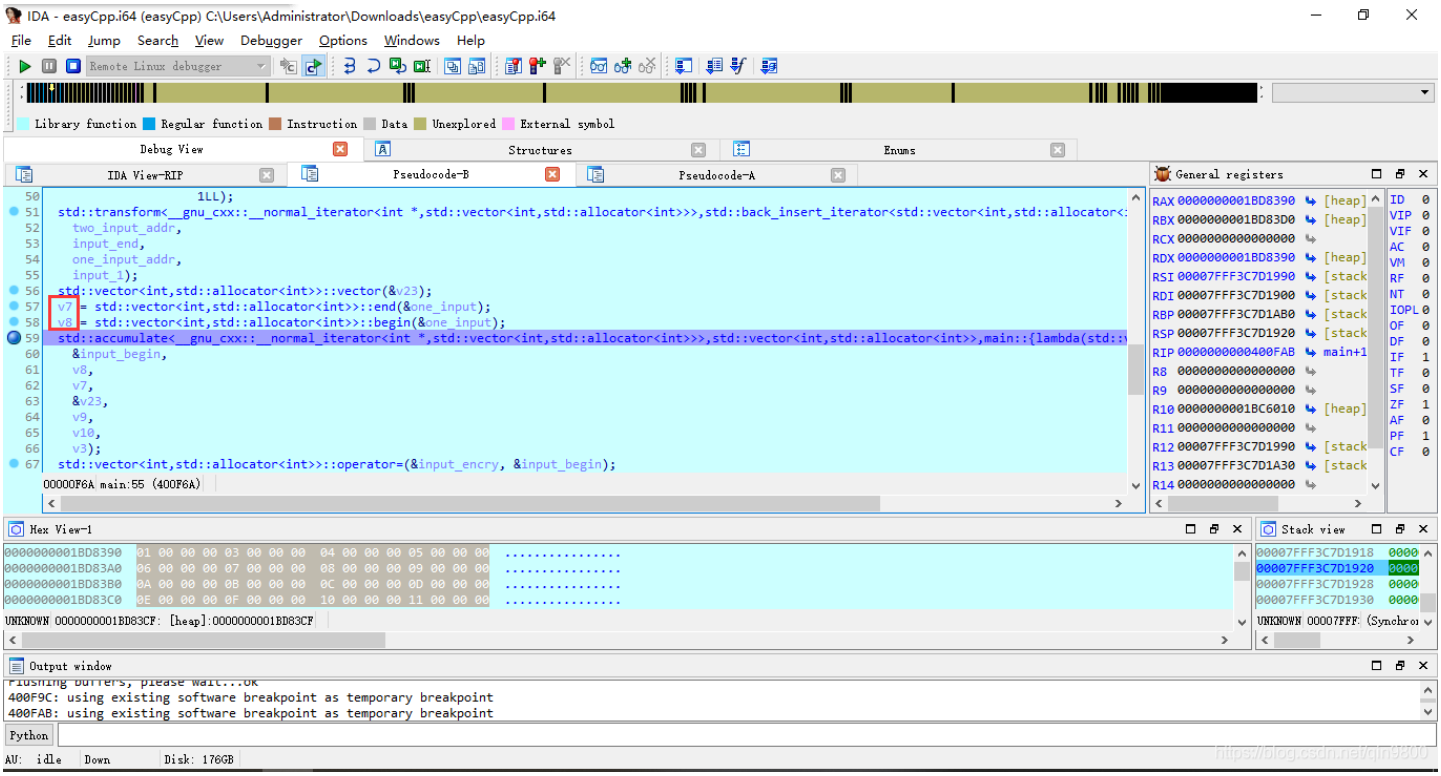


这个循环是对用户输入的数据第二个开始一直到最后一个，进行加密，详细分析写在图中注释，这个函数实现的功能就是把用户输入的整形数组从第二个开始到最后一个依次加上第一个数字



这里的v7指向用户输入加密后的最后一个字符

这里的v8指向用户输入加密后的第一个字符



接下来分析accumulate函数

通过动态调试分析，发现这个函数是对用户输入加密后的数据进行逆序。

```
v7 = std::vector<int,std::allocator<int>>::end(&one_input);
v8 = std::vector<int,std::allocator<int>>::begin(&one_input);
std::accumulate<__gnu_cxx::__normal_iterator<int *,std::vector<int,std::allocator<int>>>,std::vector<int,std::allocator<int>>,main::{lambda(std::
  &input_begin,
  v8,
  v7,
  &v23,
  v9,
  v10,
  v3);
std::vector<int,std::allocator<int>>::operator=(&input_encry, &input_begin);// 把加密后的数据地址给input_encry
std::vector<int,std::allocator<int>>::~vector(&input_begin);
std::vector<int,std::allocator<int>>::~vector(&v23);
if ( std::operator!=<int,std::allocator<int>>(&input_encry, &fibs) )// 对比数据
{
  puts("You failed!");
```

```
    exit(0);
  }
00000FE2 main:67 (400FE2)
```


input_encry的生成算法已经基本清晰了

Python实现

```python
# -*- coding: utf-8 -*-
# @Author: 夏了茶糜
# @Date:   2020-03-17 21:42:19
# @email: sxin0807@qq.com
# @Last Modified by:   夏了茶糜
# @Last Modified time: 2020-03-17 22:20:48
def fibs(n):
 if not n or n == 1:
  return 1
 return fibs(n-1) + fibs(n-2)
tmp = []
for i in range(0,16):
 tmp.append(fibs(i))
tmp = tmp[::-1]
print(tmp)
print(tmp[0])
for i in range(1,16):
 tmp[i] = tmp[i] - tmp[0]
 print(tmp[i])
```

```
[987, 610, 377, 233, 144, 89, 55, 34, 21, 13, 8, 5, 3, 2, 1, 1]
987
-377
-610
-754
-843
-898
-932
-953
-966
-974
-979
-982
-984
-985
-986
-986
```

得到flag

```
flag{987-377-843-953-979-985}
```