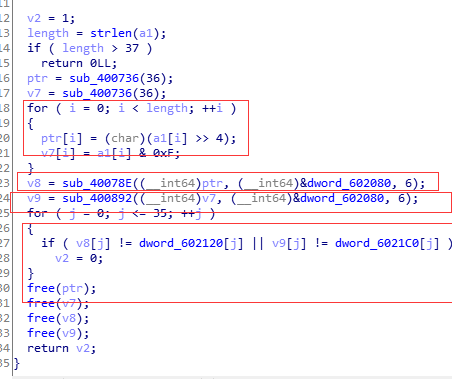
## 题目分析

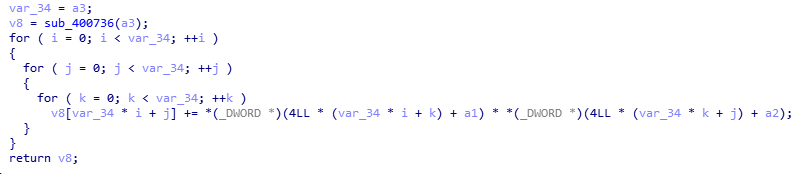
拖进IDA进行分析，可以发现sub\_40094C对输入进行了处理。

点进IDA，可以发现程序的逻辑也比较清晰



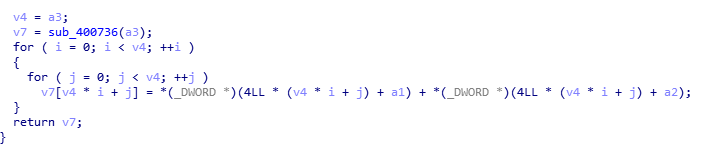
首先for循环对输入的字符进行处理，然后得到ptr数组和v7数组，都有32个值。

然后点进sub\_40078E函数进行分析。

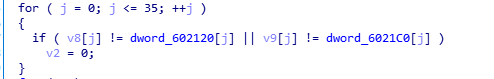


可以发现有三个for循环，是对传进的两个数组进行了矩阵相乘的运算。

然后点进sub\_400892函数进行分析



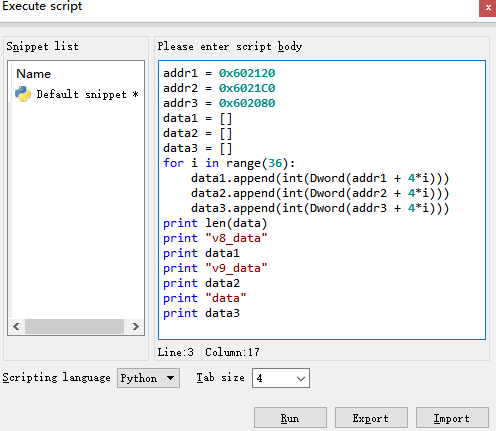
可以发现是传进函数的两个数组中的值进行相加。



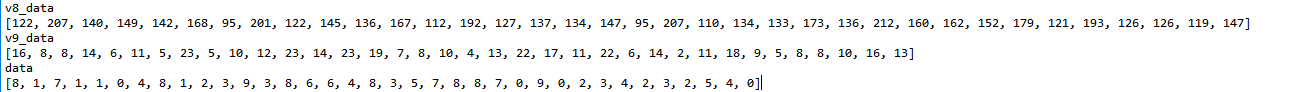
最后就是将两个函数得到的数组中的值与程序中数据的值进行对比。

# 脚本

首先我们写出IDAPython脚本将传进两个函数的数组数据和对比的两个数组数据打印出来。

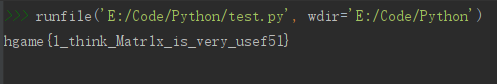


addr1 = 0x602120  
addr2 = 0x6021C0  
addr3 = 0x602080  
data1 = []  
data2 = []  
data3 = []  
for i in range(36):  
 data1.append(int(Dword(addr1 + 4\*i)))  
 data2.append(int(Dword(addr2 + 4\*i)))  
 data3.append(int(Dword(addr3 + 4\*i)))  
print len(data)  
print "v8\_data"  
print data1  
print "v9\_data"  
print data2  
print "data"  
print data3



然后剩下的运算我们通过python的Z3库来进行运算

#coding:utf-8  
from z3 import \*  
v8\_data = [122, 207, 140, 149, 142, 168, 95, 201, 122, 145, 136, 167, 112, 192, 127, 137, 134, 147, 95, 207, 110, 134, 133, 173, 136, 212, 160, 162, 152, 179, 121, 193, 126, 126, 119, 147]  
v9\_data = [16, 8, 8, 14, 6, 11, 5, 23, 5, 10, 12, 23, 14, 23, 19, 7, 8, 10, 4, 13, 22, 17, 11, 22, 6, 14, 2, 11, 18, 9, 5, 8, 8, 10, 16, 13]  
data = [8, 1, 7, 1, 1, 0, 4, 8, 1, 2, 3, 9, 3, 8, 6, 6, 4, 8, 3, 5, 7, 8, 8, 7, 0, 9, 0, 2, 3, 4, 2, 3, 2, 5, 4, 0]  
  
solver = Solver()  
flag = [BitVec("flag%d" % i, 8) for i in range(36)]  
  
for i in range(len(flag)):  
 solver.add(flag[i] < 127)  
 solver.add(flag[i] > 32)  
  
ptr = [(i >> 4) for i in flag]  
v7 = [(i & 15) for i in flag]  
  
v8 = [0 for i in range(36)]  
v9 = [0 for i in range(36)]  
  
#sub\_40078E  
for i in range(6):  
 for j in range(6):  
 for k in range(6):  
 v8[6 \* i + j] += ptr[6 \* i + k] \* data[6 \* k + j]  
  
#sub\_400892  
for i in range(6):  
 for j in range(6):  
 v9[6 \* i + j] = v7[6 \* i + j] + data[6 \* i + j]  
  
for i in range(36):  
 solver.add(v8[i] == v8\_data[i])  
 solver.add(v9[i] == v9\_data[i])  
  
#check一下看看有没有解  
if solver.check() == sat:  
 model = solver.model()  
 s = []  
 #s = [chr(model[flag[i]].as\_long().real) for i in range(36)]  
 for i in range(36):  
 s += chr(model[flag[i]].as\_long().real)  
 print "".join(s)  
else:  
 print "unsat"



### FLAG：hgame{1\_think\_Matr1x\_is\_very\_usef5l}