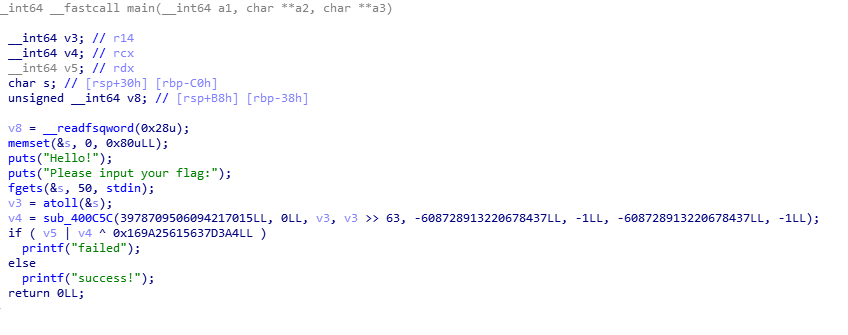
# 题目分析

拖进IDA进行反编译后



点进函数sub\_400C5C后会发现里面是一个很复杂的运算。

发但是进行动态调试的话，发现函数并不会走这里。

核心代码应该不在这里

查找init函数 调用main函数之前会先调用这个函数

发现init调用的函数中 有一个很可疑sub\_400976。

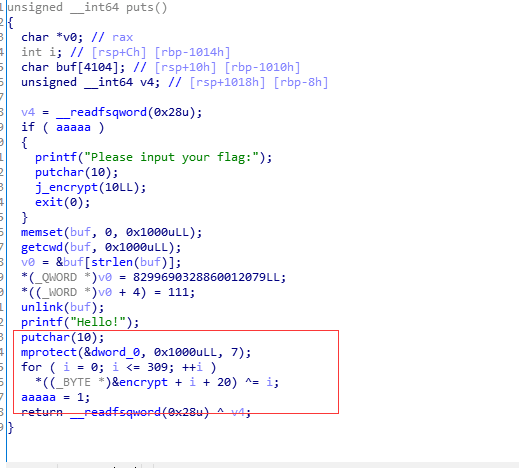


这代码的意思是

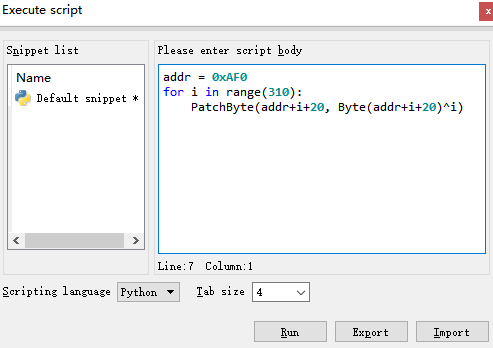
从源文件读取倒数12936个字节出来当作文件运行

也就是说 源文件可能有两个elf文件，用winhex一查果然如此，用winhex抠出来分析。

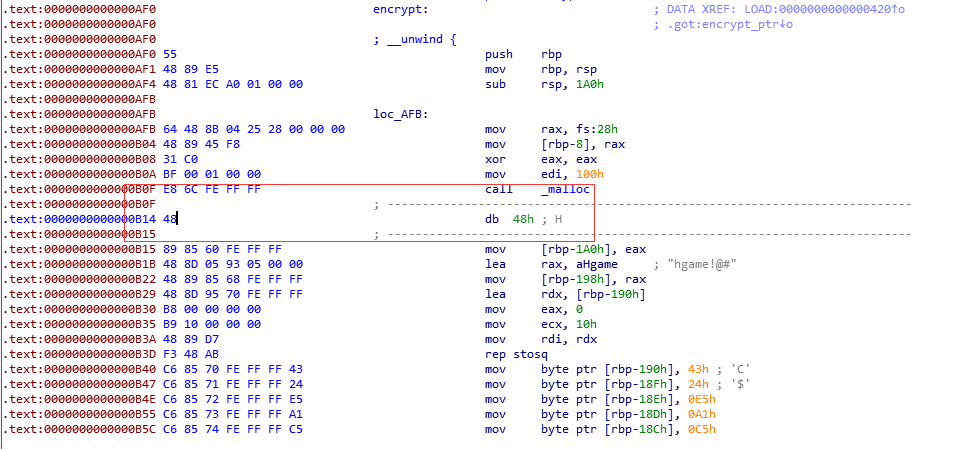
分析新的ELF文件，然后发现了一段SMC自解密。



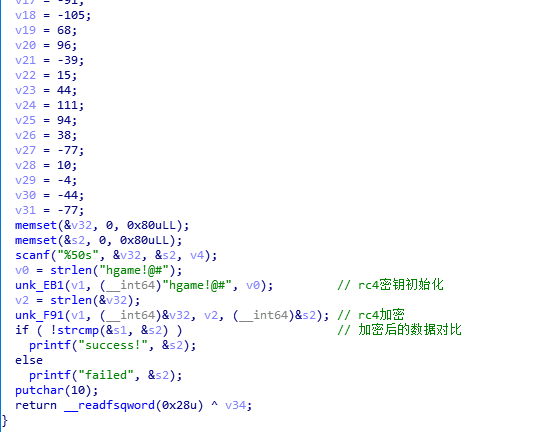
我们来写IDAPython脚本进行解密。



解密后发现有个流氓字节干扰了反编译，我们给nop掉。



然后在函数名处按U再按P即可反编译了，反编译后可以轻松分析处事rc4加密。



密钥是”hgame!@#”。

# 脚本

#include<stdio.h>

#include<string.h>

void rc4\_init(unsigned char \*s,unsigned char \*key, unsigned long Len)

{

int i=0,j=0;

char k[256]={0};

unsigned char tmp=0;

for(i=0;i<256;i++)

{

s[i]=i;

k[i]=key[i%Len];

}

for(i=0;i<256;i++)

{

j=(j+s[i]+k[i])%256;

tmp=s[i];

s[i]=s[j]; //交换s[i]和s[j]

s[j]=tmp;

}

}

void rc4\_crypt(unsigned char \*s,unsigned char \*Data,unsigned long Len)

{

int i=0,j=0,t=0;

unsigned long k=0;

unsigned char tmp;

for(k=0;k<Len;k++)

{

i=(i+1)%256;

j=(j+s[i])%256;

tmp=s[i];

s[i]=s[j]; //交换s[x]和s[y]

s[j]=tmp;

t=(s[i]+s[j])%256;

Data[k]^=s[t];

}

}

int main()

{

unsigned char s[256] = { 0 }, s2[256] = { 0 };//S-box

char key[256] = { "hgame!@#" };

char pData[512] = {67,36,-27,-95,-59,29,114,-46,40,-17,-66,-22,-91,-105,68,96,-39,15,44,111,94,38,-77,10,-4,-44,-77};

unsigned long len = strlen(pData);

int i;

//printf("pData=%s\n", pData);

printf("key=%s,length=%d\n", key, strlen(key));

rc4\_init(s, (unsigned char\*)key, strlen(key)); //已经完成了初始化

for (i = 0; i<256; i++) //用s2[i]暂时保留经过初始化的s[i]，很重要的！！！

{

s2[i] = s[i];

}

//rc4\_init(s,(unsignedchar\*)key,strlen(key));//初始化密钥

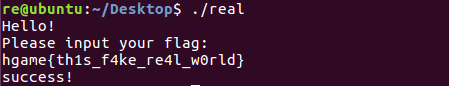
rc4\_crypt(s2, (unsigned char\*)pData, len);//解密

printf("pData=%s\n", pData);

return 0;

}

C:\Users\asus\Documents\Tencent Files\2929812422\Image\C2C\Image1\UYP2{W$0WE}(2P2X23H6R_X.png



## Flag：hgame{th1s\_f4ke\_re4l\_w0rld}