**武汉大学国家网络安全学院**

**《软件逆向分析》课程报告**

|  |  |
| --- | --- |
| **报告题目** | bomb-whu报告 |
| **专 业** | 网络空间安全 |
| **姓 名** | 彭馨勇 |
| **学 号** | **2017301510012** |
| **指导教师** | 赵 磊 |

**目 录**

目录

[1 第一关 3](#_Toc18533047)

[2 第二关 3](#_Toc18533048)

[3 第三关 4](#_Toc18533049)

[4 第四关 6](#_Toc18533050)

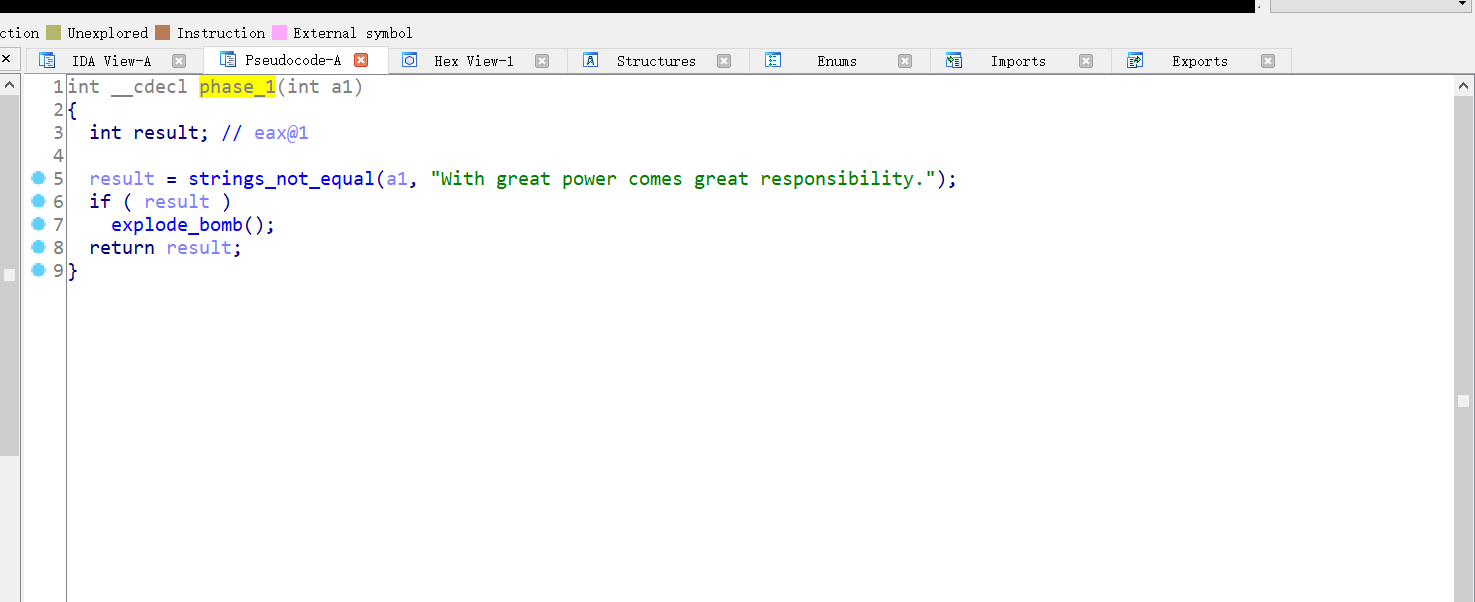
[5 第五关 8](#_Toc18533051)

[6 第六关 10](#_Toc18533052)

[7 秘密关卡 17](#_Toc18533053)

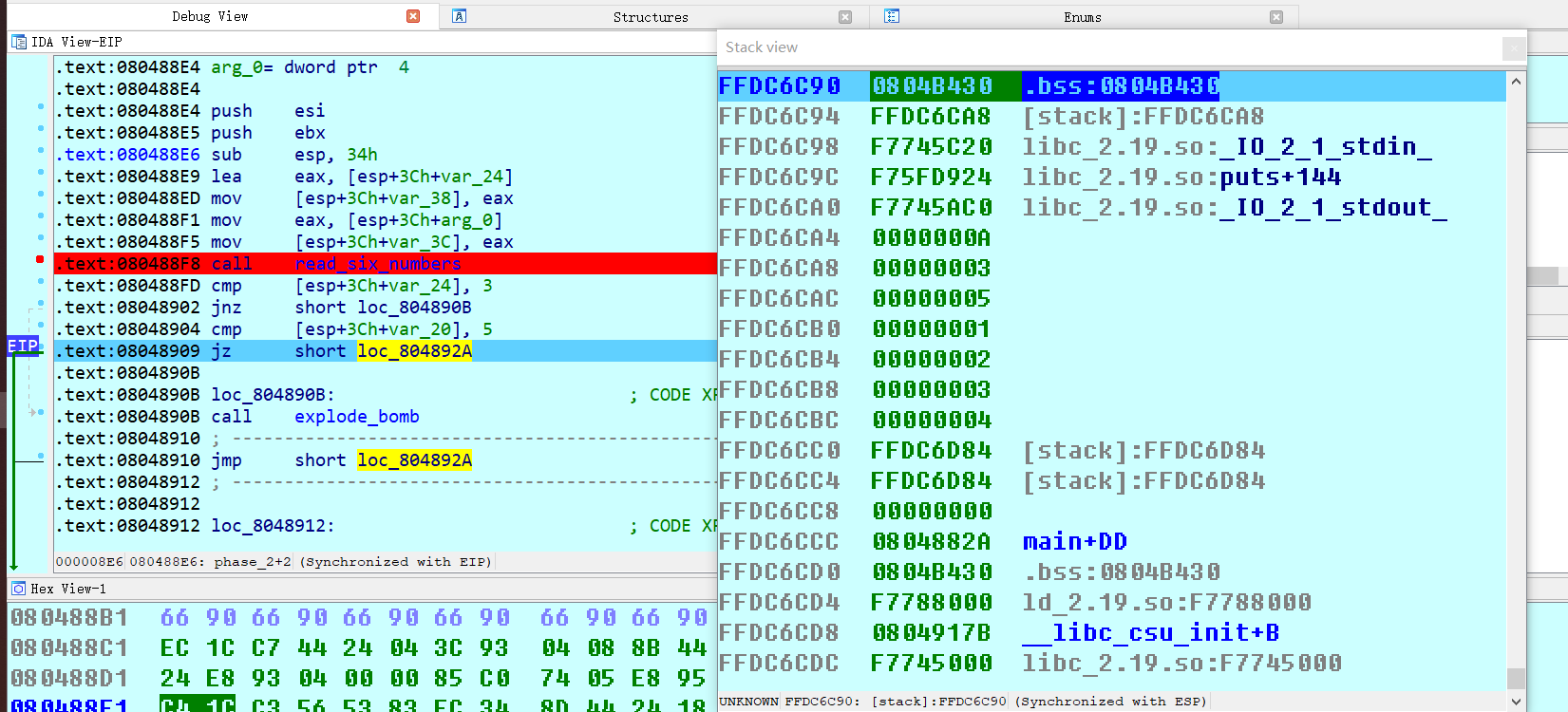
[参考 21](#_Toc18533054)

## 1 第一关



With great power comes great responsibility.

## 2 第二关

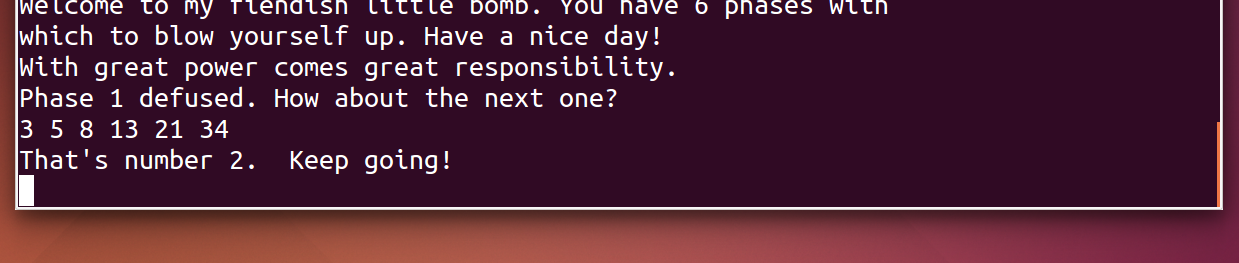


先绕过第一个限制，即前两个必须是3 和5试了半天

看一下源代码：

int \_\_cdecl phase\_2(int a1)  
{  
 int result; // eax@4  
 char \*v2; // ebx@6  
 int v3; // [sp+18h] [bp-24h]@1  
 int v4; // [sp+1Ch] [bp-20h]@2  
 char v5; // [sp+20h] [bp-1Ch]@8  
 char v6; // [sp+30h] [bp-Ch]@6  
  
 read\_six\_numbers(a1, &v3);  
 if ( v3 != 3 || v4 != 5 )  
 explode\_bomb();  
 v2 = &v5;  
 do  
 {  
 result = \*((\_DWORD \*)v2 - 1) + \*((\_DWORD \*)v2 - 2);  
 if ( \*(\_DWORD \*)v2 != result )  
 explode\_bomb();  
 v2 += 4;  
 }  
 while ( v2 != &v6 );  
 return result;  
}

试一下 3 5 8 13 21 34 果然真的是斐波那契数列

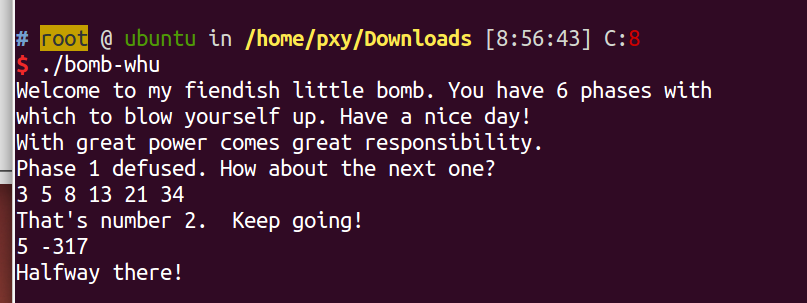


## 3 第三关

仔细分析一下逻辑

int \_\_cdecl phase\_3(int a1)  
{  
 signed int v1; // eax@4  
 int v2; // eax@6  
 int v3; // eax@8  
 int v4; // eax@10  
 int v5; // eax@12  
 int v6; // eax@14  
 int v7; // eax@16  
 int result; // eax@18  
 int v9; // [sp+18h] [bp-14h]@1  
 int v10; // [sp+1Ch] [bp-10h]@1  
  
 if ( \_\_isoc99\_sscanf(a1, "%d %d", &v9, &v10) <= 1 )// v9要小于等于5  
 explode\_bomb();  
 switch ( v9 )  
 {  
 case 1:  
 v1 = 0;  
 goto LABEL\_6;  
 case 0:  
 v1 = 985;  
LABEL\_6:  
 v2 = v1 - 329;  
 goto LABEL\_8;  
 case 2:  
 v2 = 0;  
LABEL\_8:  
 v3 = v2 + 191;  
 goto LABEL\_10;  
 case 3:  
 v3 = 0;  
LABEL\_10:  
 v4 = v3 - 317;  
 goto LABEL\_12;  
 case 4:  
 v4 = 0;  
LABEL\_12:  
 v5 = v4 + 317;  
 goto LABEL\_14;  
 case 5:  
 v5 = 0;  
LABEL\_14:  
 v6 = v5 - 317;  
 goto LABEL\_16;  
 case 6:  
 v6 = 0;  
LABEL\_16:  
 v7 = v6 + 317;  
 break;  
 case 7:  
 v7 = 0;  
 break;  
 default:  
 explode\_bomb();  
 return result;  
 }  
 result = v7 - 317;  
 if ( v9 > 5 || result != v10 ) //小于等于5且result必须等于v10  
 explode\_bomb();  
 return result;  
}

然后走一个case就行



## 4 第四关

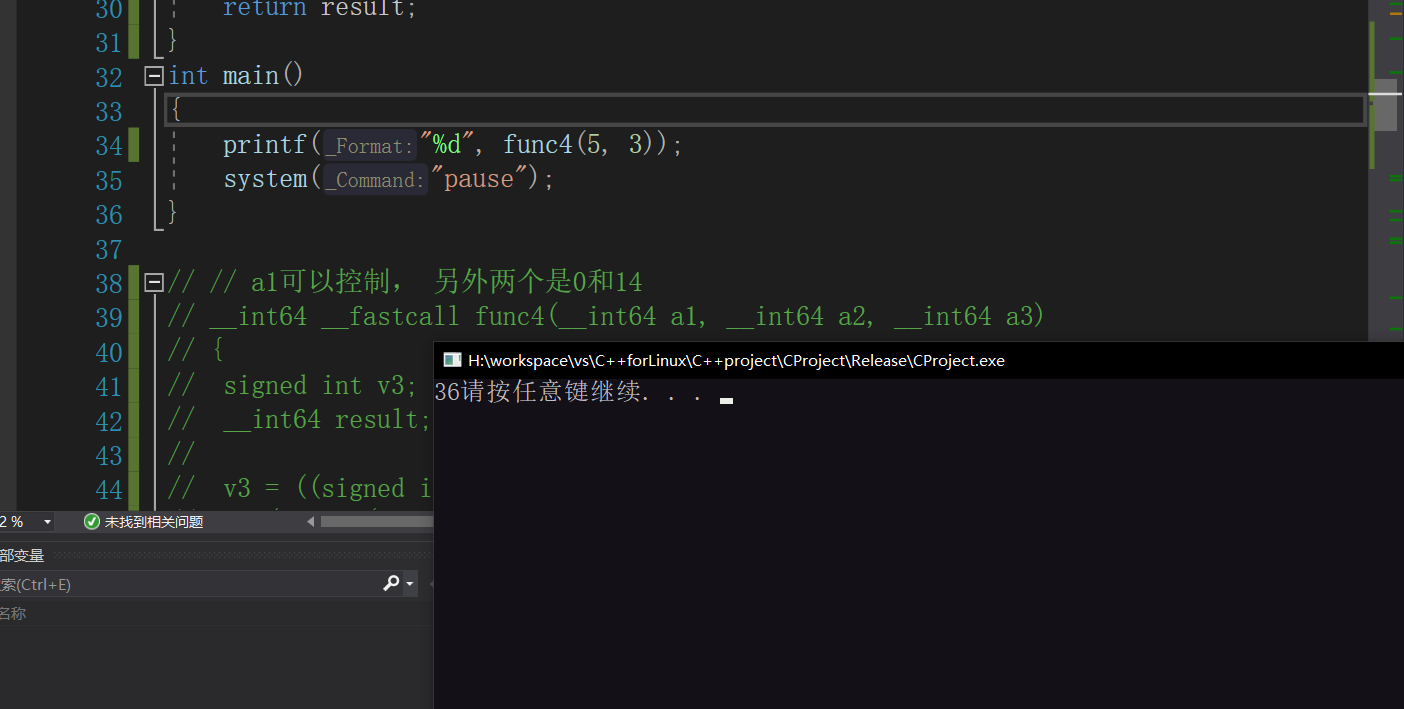
关键代码

int \_\_cdecl phase\_4(int a1)  
{  
 int result; // eax@4  
 int v2; // [sp+18h] [bp-14h]@1  
 int v3; // [sp+1Ch] [bp-10h]@1  
  
 if ( \_\_isoc99\_sscanf(a1, "%d %d", &v3, &v2) != 2 || (unsigned int)(v2 - 2) > 2 )  
 explode\_bomb(); //必须是2个参数且v2-2 <= 2  
 result = func4(5, v2);  
 if ( result != v3 ) //result必须为v3，也就是说传入一个v2计算出来的值要和v3相等  
 explode\_bomb();  
 return result;  
}

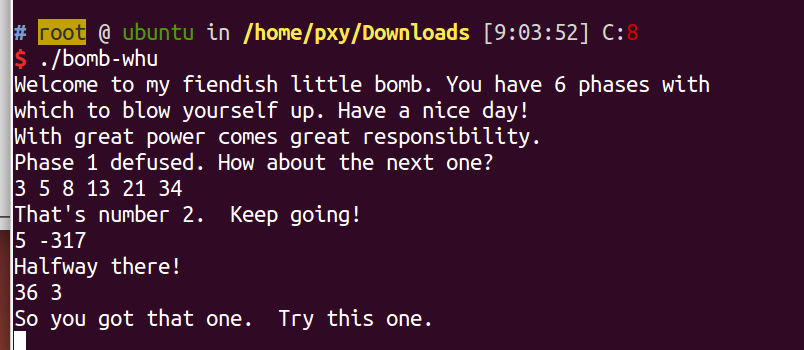
分析一下func4

int \_\_cdecl func4(int a1, int a2) //a2是可以控制的参数  
{  
 int result; // eax@2  
 int v3; // edi@3  
  
 if ( a1 <= 0 )  
 {  
 result = 0;  
 }  
 else  
 {  
 result = a2;  
 if ( a1 != 1 )  
 {  
 v3 = func4(a1 - 1, a2) + a2;  
 result = v3 + func4(a1 - 2, a2);  
 }  
 }  
 return result;  
}

注意此处(unsigned int)(v2 - 2) > 2是无符号数比较，负数会变成很大的数字



输入36 和 3即可

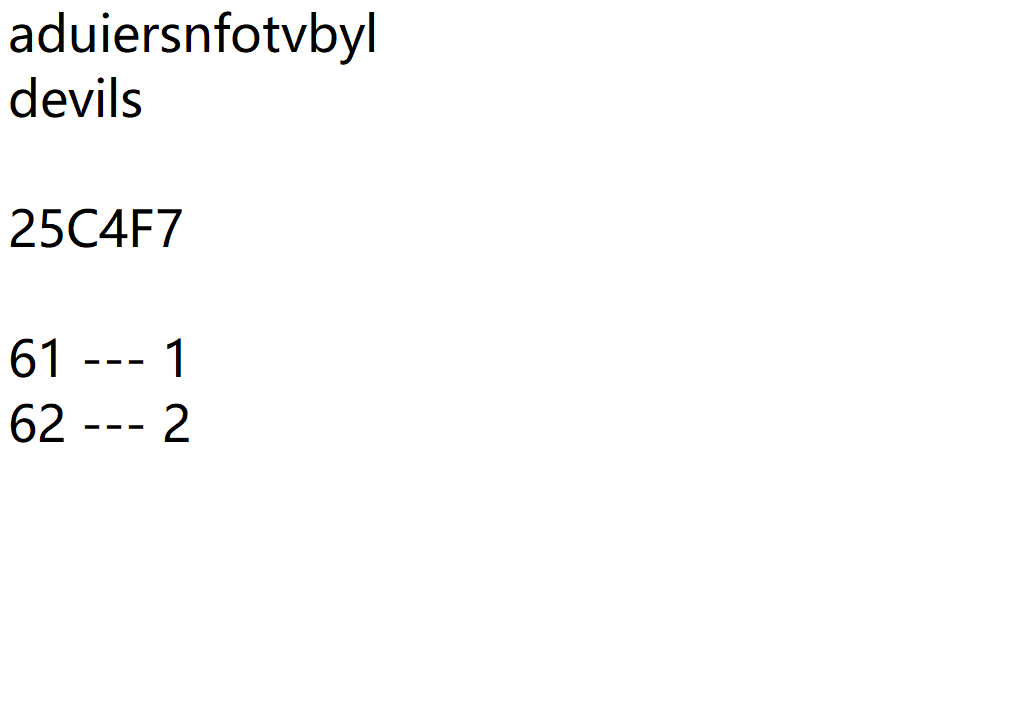


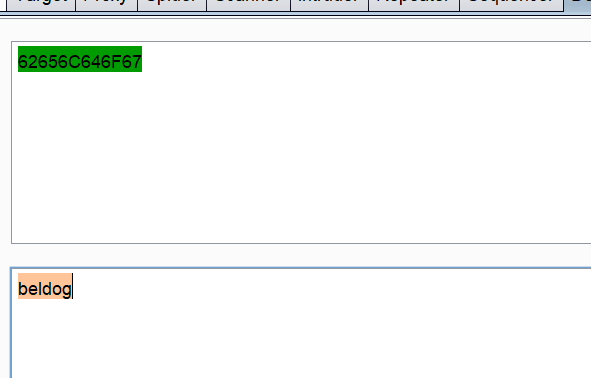
## 5 第五关

关键代码中有一处：

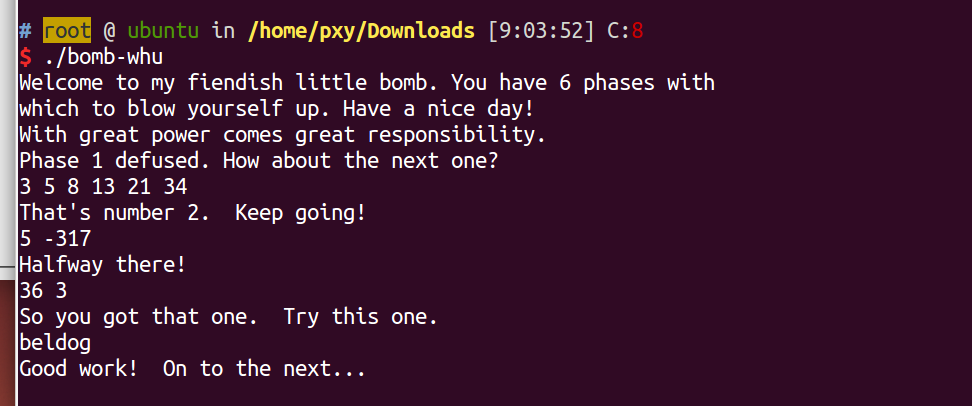
v3[v1] = array\_3143[\*(\_BYTE \*)(a1 + v1) & 0xF];

其中array\_3143数组是确定的，并且之后是跟字符串if ( strings\_not\_equal(v3, "devils") ) 比较，那么就是从中选字符就是了，调试的时候发现是从1开始的被坑了





输入beldog即可



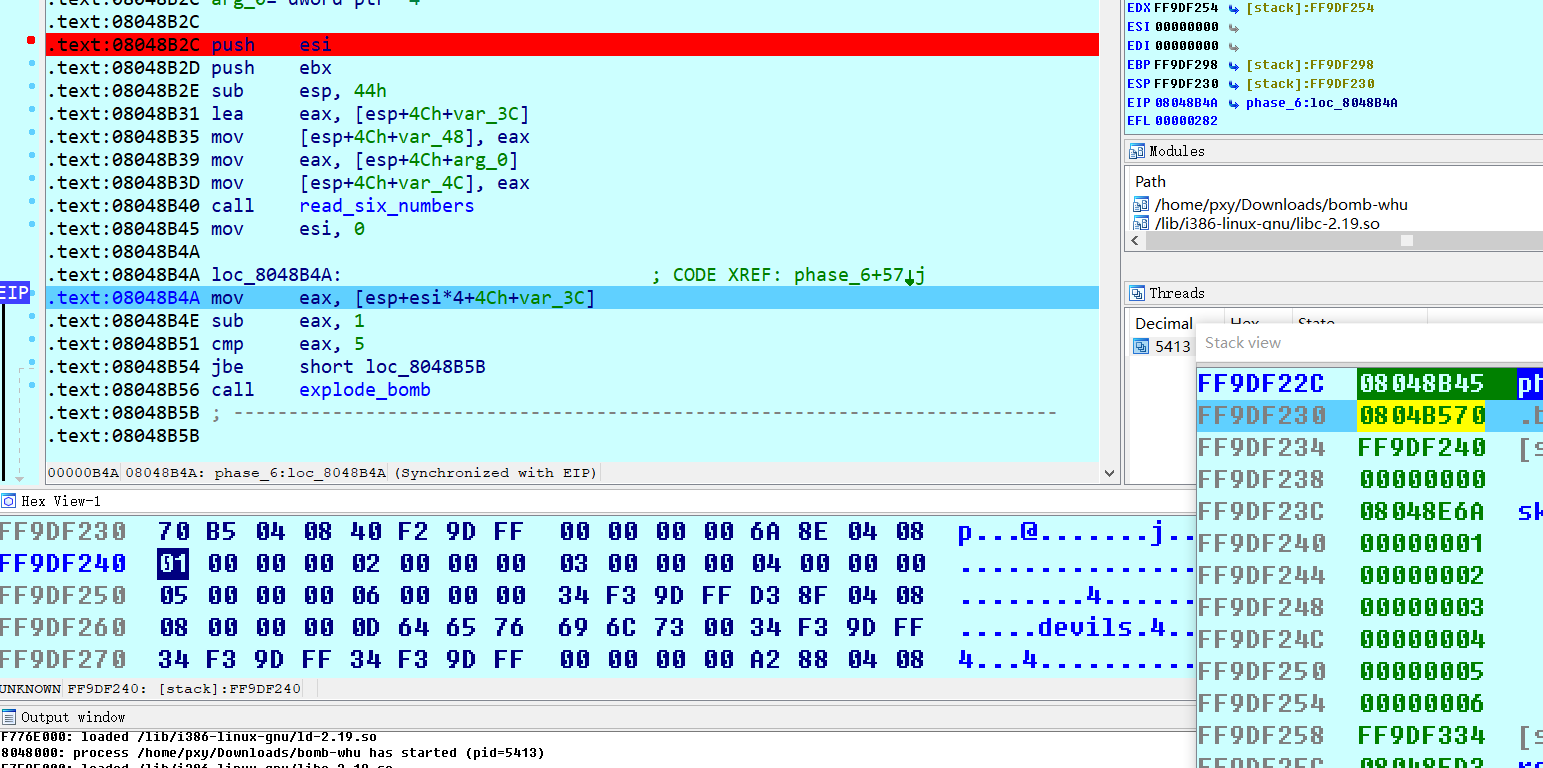
## 6 第六关

int \_\_cdecl phase\_6(int a1)  
{  
 signed int v1; // esi@1  
 int v2; // ebx@5  
 signed int v3; // ebx@6  
 \_DWORD \*v4; // edx@11  
 signed int v5; // eax@11  
 int v6; // ecx@15  
 int v7; // ebx@17  
 char \*v8; // eax@17  
 int i; // ecx@17  
 int v10; // edx@18  
 signed int v11; // esi@20  
 int result; // eax@21  
 int v13; // [sp+Ch] [bp-40h]@7  
 int v14[6]; // [sp+10h] [bp-3Ch]@1  
 int v15; // [sp+28h] [bp-24h]@14  
 char v16; // [sp+2Ch] [bp-20h]@17  
 char v17; // [sp+40h] [bp-Ch]@18  
  
 read\_six\_numbers(a1, v14);  
 v1 = 0;  
 while ( 1 )  
 {  
 if ( (unsigned int)(v14[v1] - 1) > 5 )  
 explode\_bomb();  
 if ( ++v1 == 6 )  
 break;  
 v3 = v1;  
 do  
 {  
 if ( \*(&v13 + v1) == v14[v3] )  
 explode\_bomb();  
 ++v3;  
 }  
 while ( v3 <= 5 );  
 }  
 v2 = 0;  
 do  
 {  
 v6 = v14[v2];  
 if ( v6 <= 1 )  
 {  
 v4 = &node1;  
 }  
 else  
 {  
 v5 = 1;  
 v4 = &node1;  
 do  
 {  
 v4 = (\_DWORD \*)v4[2];  
 ++v5;  
 }  
 while ( v5 != v6 );  
 }  
 \*(&v15 + v2++) = (int)v4;  
 }  
 while ( v2 != 6 );  
 v7 = v15;  
 v8 = &v16;  
 for ( i = v15; ; i = v10 )  
 {  
 v10 = \*(\_DWORD \*)v8;  
 \*(\_DWORD \*)(i + 8) = \*(\_DWORD \*)v8;  
 v8 += 4;  
 if ( v8 == &v17 )  
 break;  
 }  
 \*(\_DWORD \*)(v10 + 8) = 0;  
 v11 = 5;  
 do  
 {  
 result = \*\*(\_DWORD \*\*)(v7 + 8);  
 if ( \*(\_DWORD \*)v7 > result )  
 explode\_bomb();  
 v7 = \*(\_DWORD \*)(v7 + 8);  
 --v11;  
 }  
 while ( v11 );  
 return result;  
}

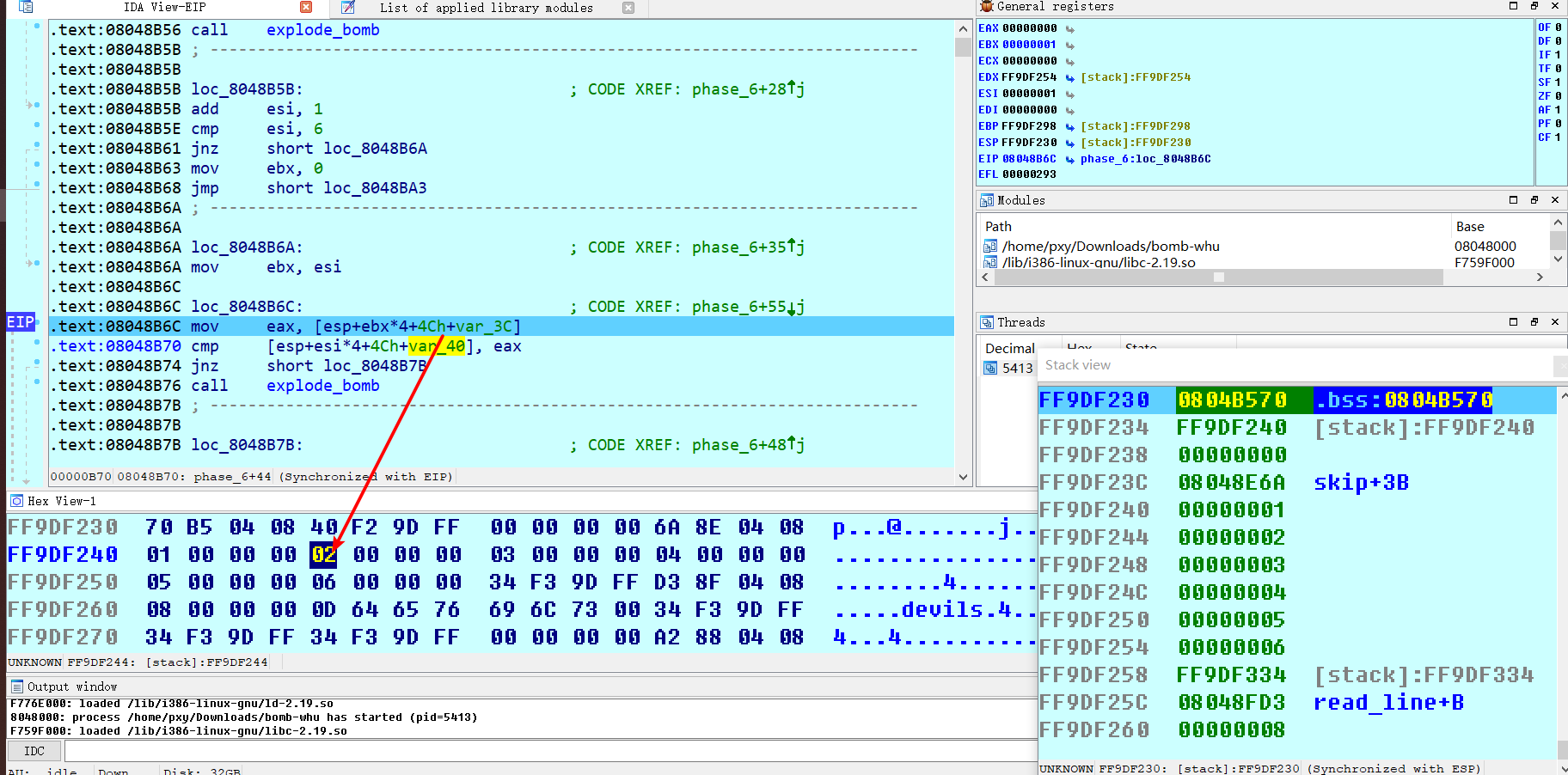
输入 1 2 3 4 5 6存入的数据

学习一下，此时可以通过esi寄存器进行变址寻址

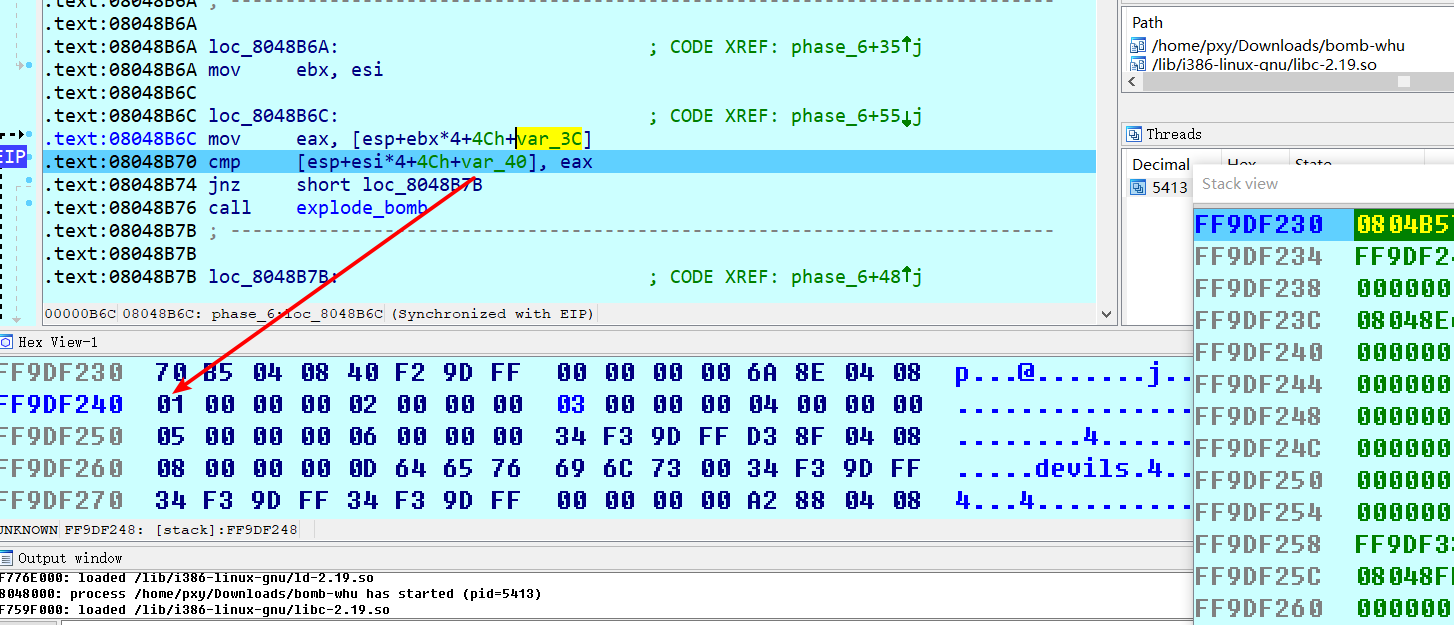
第一个循环的v13这个变量很神奇，之前根本没定义啊，自己调试来看一下



然后将数据存入



而此时



那大概就是取之前的那个数字了于是这段代码的含义：

就是每个数必须都小于6并且互不相等

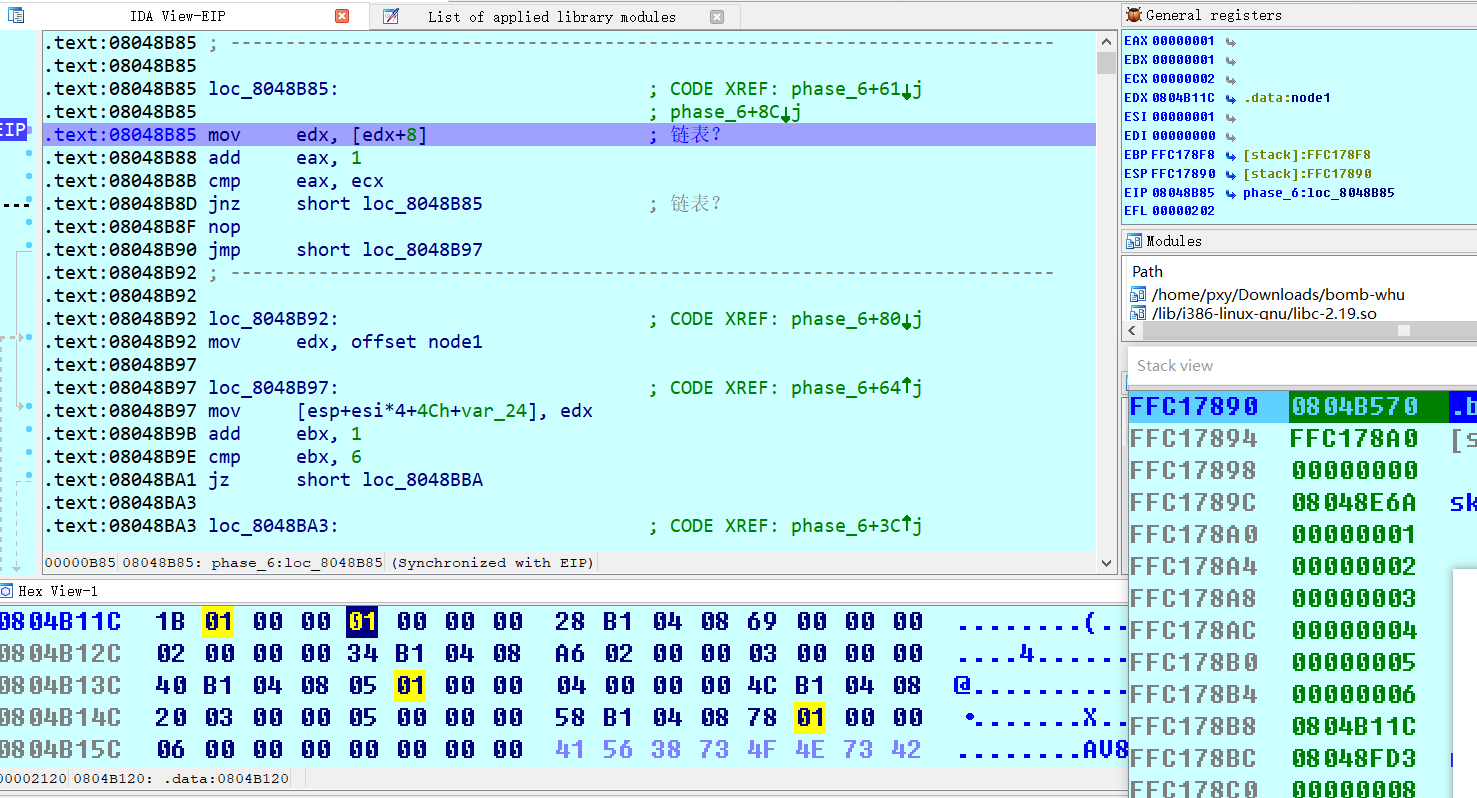
接下来看下一个循环

v2 = 0;  
 do  
 {  
 v6 = v14[v2]; //取出v14数组第v2个相当于遍历数组  
 if ( v6 <= 1 )  
 {  
 v4 = &node1;  
 }  
 else  
 {  
 v5 = 1;  
 v4 = &node1;  
 do  
 {  
 v4 = (\_DWORD \*)v4[2]; //指针+2，v4在不断地变化  
 ++v5;   
 }  
 while ( v5 != v6 );  
 }  
 \*(&v15 + v2++) = (int)v4; //v15又没有定义   
 }  
 while ( v2 != 6 ); //进行6次循环

下一个循环

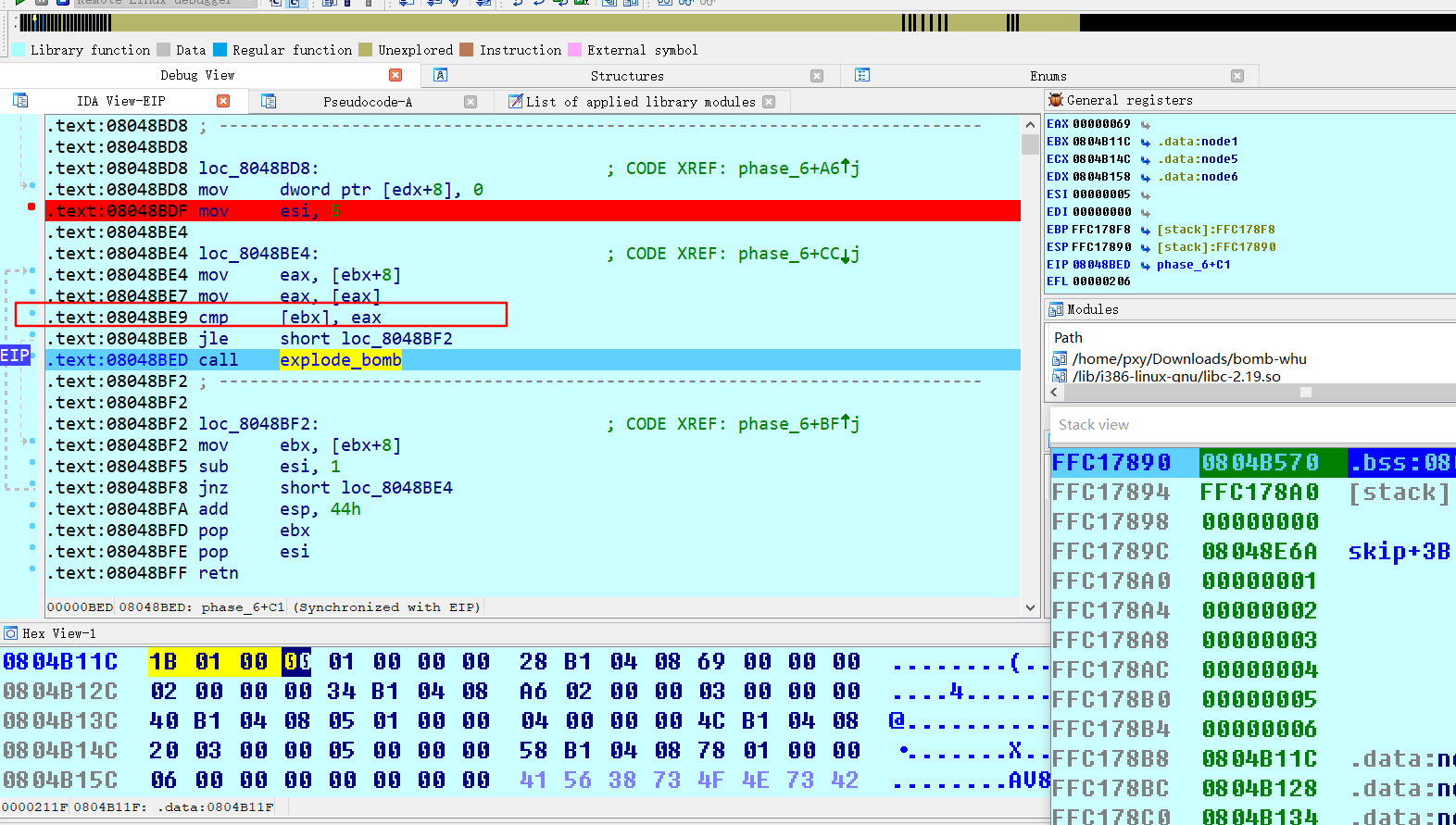
v7 = v15;  
 v8 = &v16;  
 for ( i = v15; ; i = v10 )  
 {  
 v10 = \*(\_DWORD \*)v8;  
 \*(\_DWORD \*)(i + 8) = \*(\_DWORD \*)v8;  
 v8 += 4;  
 if ( v8 == &v17 )  
 break;  
 }

此时看到edx的值

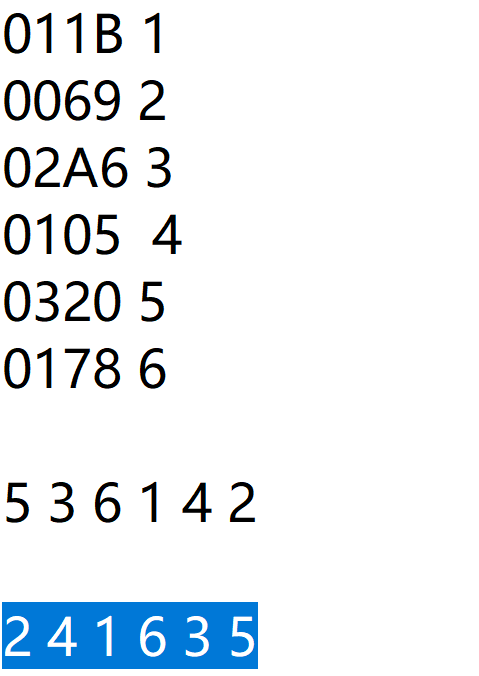


恍然大悟这里是链表。。

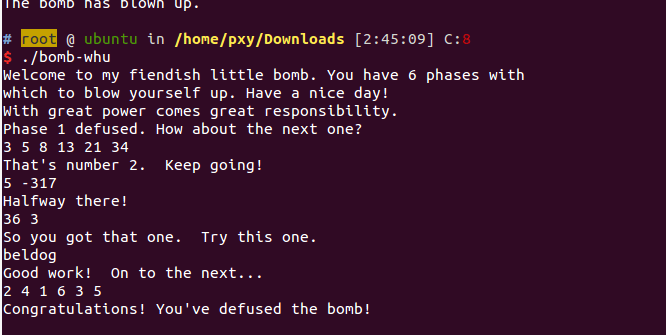
重点是这个比较



大概明白就是比较链表中每个元素的前8个字节，于是按照顺序排列一下，这时候发现IDA太过聪明了直接把小端序给倒过来了搞得我排反了



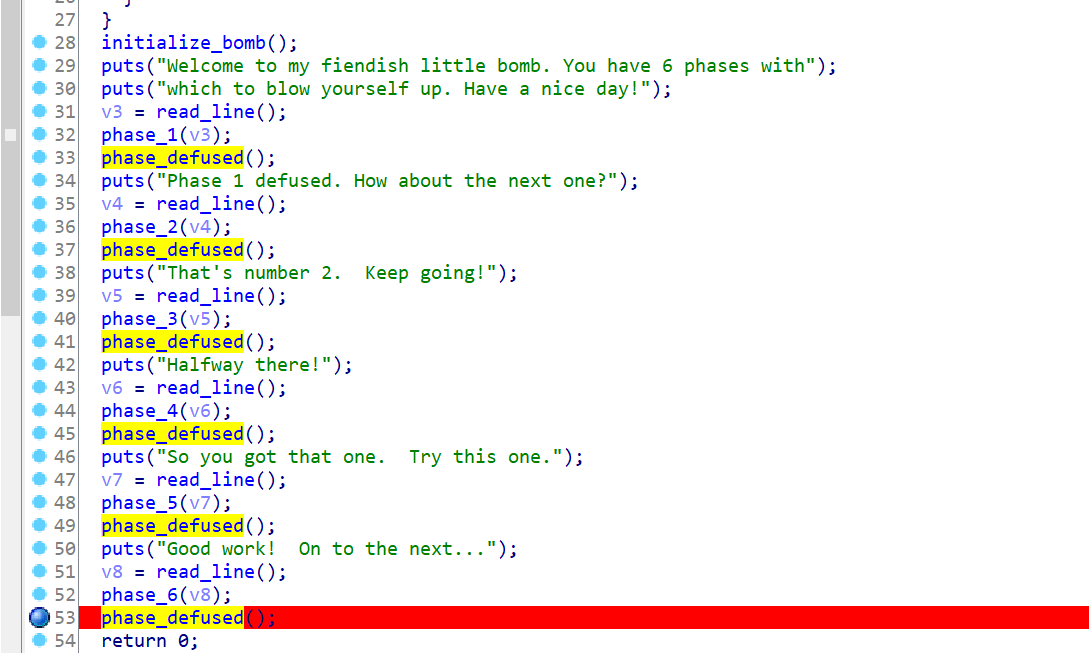
输入即可



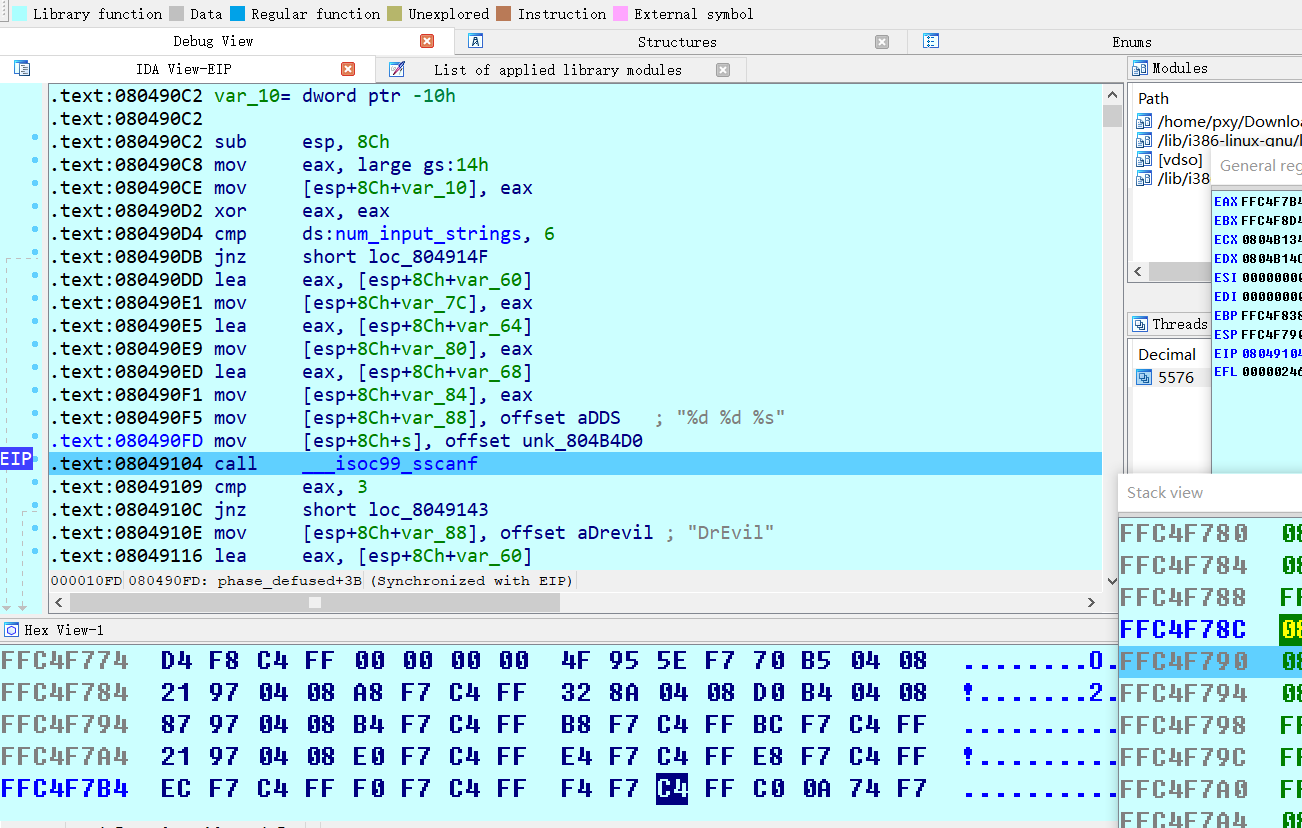
## 7 秘密关卡

突然发现还有一个密码的关卡。。

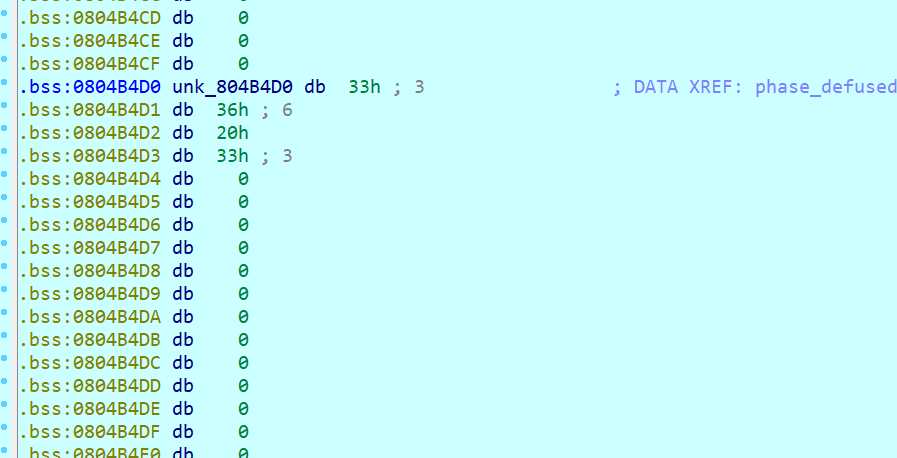
怎么进去是一个问题，于是我在最后调用那个phase\_defused函数的地方下个断点



此时在调用sscanf函数之前，我多次调试后发现返回值都是2不管怎么传值，于是想知道这里是从哪里读入的



双击进入之后



发现36 3这几个数据，这不就是前面的关卡输入的吗

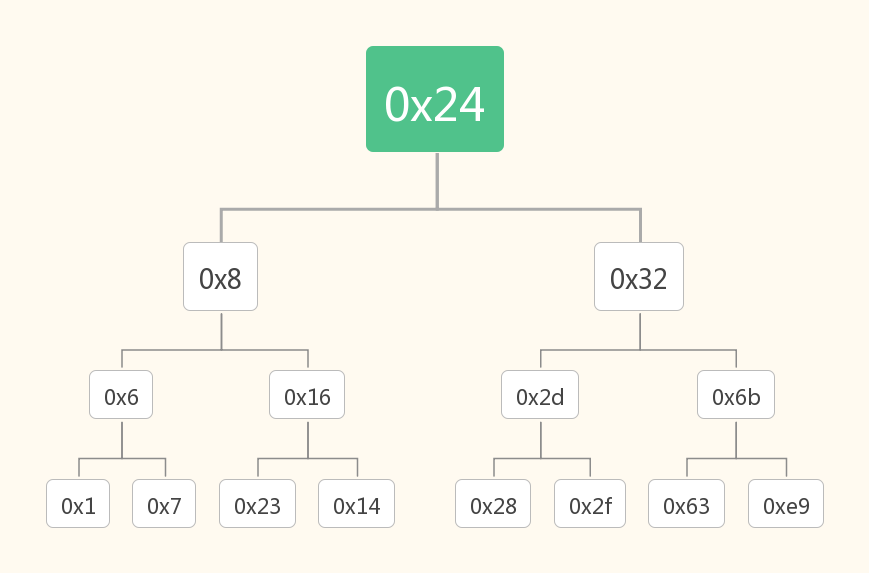
原来还需要修改之前的输入23333

接下来分析数据结构

查看edx发现了一些规律

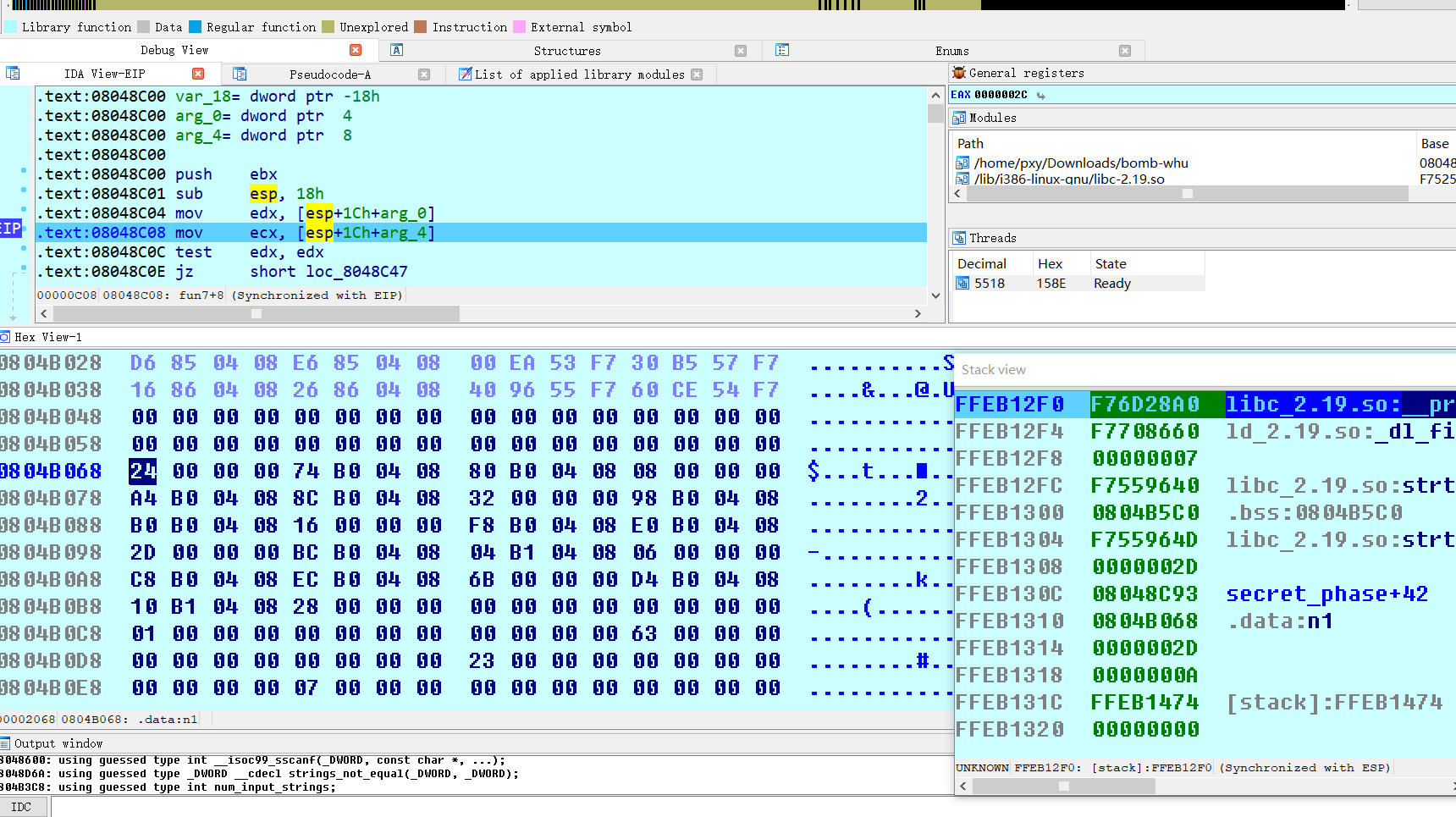
是一个二叉树，里面存放了数值，会根据我们传入的数字进行比较然后相应的跳转(这部分没有自己分析出来是看了师傅的blog)

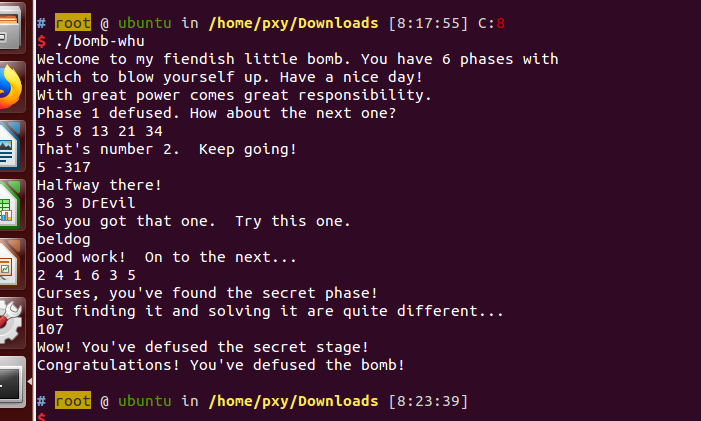
借师傅一张图：



我们要让函数的结果返回3，按照递归来分析，最后一次返回0是肯定的，第二次调用返回2\*eax+1为1，第一次调用返回为2\*eax+1为3就行了

于是就知道在第三层，并且是两次都需要返回2\*eax+1说明必须两次比较都是大于，所以确定答案就是 0x6b十进制107





## 参考

还是参考了师傅们的blog才搞出来的

https://www.viseator.com/2017/06/21/CS\_APP\_BombLab/