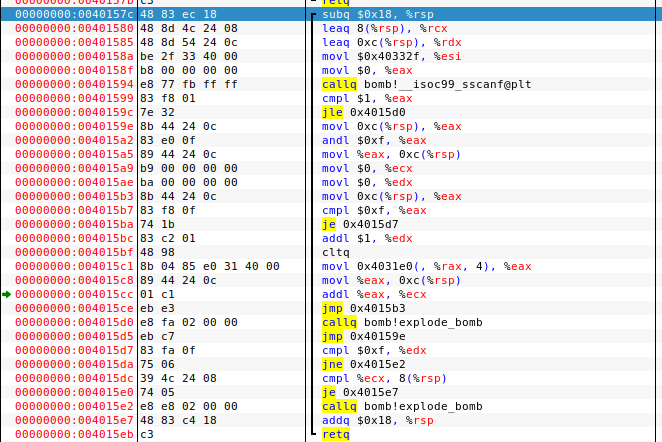


结合前面几关的经验，从这段神秘代码中很容易发现依然是读取两个整数并检验其输入的值是否符合预期。



将第5关的这部分代码手动反编译可以得到如下C代码：

int a, b;

scanf("%d %d", &a, &b); //a in %0xc(%rsp), b in 0x8(%rsp)

int ret = a & 0xf; //0b0000 1111即取a的低4位，0～15

a = ret; //ret in %eax

int n = 0; //n in %ecx

int m = 0; //m in %edx

do{

ret = a;

if(ret == 0xf)

break;

m += 1;

ret = array[ret]; //array starts from 0x4031e0

/\*Array:0xa, 0x2, 0xe, 0x7, 0x8, 0xc, 0xf, 0xb, 0x0, 0x4, 0x1, 0xd, 0x3, 0x9, 0x6, 0x5

Array[0] = 0xa;

Array[1] = 0x2;

Array[2] = 0xe;

Array[3] = 0x7;

Array[4] = 0x8;

Array[5] = 0xc;

Array[6] = 0xf;

Array[7] = 0xb;

Array[8] = 0x0;

Array[9] = 0x4;

Array[0xa] = 0x1;

Array[0xb] = 0xd;

Array[0xc] = 0x3;

Array[0xd] = 0x9;

Array[0xe] = 0x6;

Array[0xf] = 0x5

\*/

a = ret;

n += ret;

}while(1);

if(m != 0xf || b != n)

bomb();

由此代码可知，输入的第一个数(也就是代码中的a变量)只有低4位有效，因此无论如何输入a的值，只有0~15这16个数是有效的。在do-while循环体中，只有当ret的值为0xf(也就是十进制的15)时才会break，也就是说要让a的值最终被修改为0xf。同时，要让m的值为0xf，也就是说要执行15次循环体才行。Array数组形成的链式结构如下：

0x0🡪0xa🡪0x1🡪0x2🡪0xe🡪0x6🡪0xf🡪0x5🡪0xc🡪0x3🡪0x7🡪0xb🡪0xd🡪0x9🡪0x4🡪0x8🡪0x0

因此让初始时的a为0xf之后的0x5，即可让循环体执行15次。

同时要求第2个输入的数字等于最终的n，通过分析可知最终n = 0x0+0x1+0x2+0x3+…+0xf-0x5 = 115。之所以要减去0x5，是因为第一次输入的0x5赋给ret之后执行了ret = Array[ret]，此时的ret已经被修改过一次，因此不需要加0x5。

因此可以分析出输入的两个数分别是：5, 115。

经过测试这两个数结果正确。