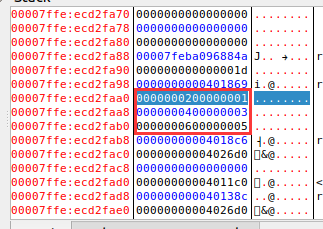


从第6关刚开始执行的过程来看，第6关是要输入6个数字并检验其正确性。

初次尝试输入六个数1 2 3 4 5 6，观察堆栈内容，可以发现6个数分别放在rsp偏移为0x30, 0x34, 0x38, 0x3C, 0x40, 0x44的位置。



通过分析汇编代码发现，在第6关函数的栈帧中，不光有6个输入的整数，还存在6个8字节的指针数据，位于rsp偏移0x0开始的6个位置。

通过对第6关的代码反汇编，发现其中存在一个链表，对其代码进行手工反编译后得到如下C语言代码：

#include <stdio.h>

void bomb();

int main()

{

    int a[6];   *//rsp+0x30, rsp+0x34, rsp+0x38, rsp+0x3C, rsp+0x40, rsp+0x44*

    long \*l[6]; *//rsp+0x0, rsp+0x8, rsp+0x10, rsp+0x18, rsp+0x20, rsp+0x28*

    scanf("%d %d %d %d %d %d", &a[0], &a[1], &a[2], &a[3], &a[4], &a[5]);

    long rax, rbx, rbp, rsi, rdi, r8, r9, r10, r11, r12, r13, r14, r15;

    long rcx;

    long rdx;

*//0x4015fe*

*//这段代码要求输入的6个数都小于等于6，且两两不同*

    for (rbp = 0; rbp <= 5; rbp++)

    {

        if (a[rbp] - 1 > 5)

            bomb();

        for (rbx = rbp + 1; rbx <= 5; rbx++)

        {

            if (a[rbx] == a[rbp])

                bomb();

        }

    }

*//0x40164b*

    for (rsi = 0; rsi <= 5; rsi++)

    {

        rax = 1;  *//rax是一个计数器*

        rdx = 0x4052d0; *//这是一个全局的    链表的地址*

*//其内容为：*

*/\*\**

*\* 1：004052d0: 00 00 00 00 00 40 52 e0 00 00 00 01 00 00 01 a6*

*\* 2：0x4052e0: 00 00 00 00 00 40 52 f0 00 00 00 02 00 00 03 d5*

*\* 3：0x4052f0: 00 00 00 00 00 40 53 00 00 00 00 03 00 00 03 a2*

*\* 4：0x405300: 00 00 00 00 00 40 53 10 00 00 00 04 00 00 03 6c*

*\* 5：0x405310: 00 00 00 00 00 40 53 20 00 00 00 05 00 00 01 92*

*\* 6：0x405320: 00 00 00 00 00 00 00 00 00 00 00 06 00 00 01 80*

*\**

*\*/*

*//可以推断这是一个链表*

*/\**

*对于一个rsi, 下面的循环执行a[rsi]次，之后把指针l[rsi]赋为第a[rsi]个节点(头节点为第1个而不是第0个节点)*

*\*/*

        while (a[rsi] > rax)

        {

            rdx = \*((long \*)(rdx + 8));  *//使rdx指向下一个节点*

            rax++;

        }

        l[rsi] = rdx;

    }

*//0x40167a*

    rbx = l[0];

    rcx = rbx;

*//这部分代码是从l[0]指针开始，按照l[]指针数组把链表结点重新排序*

    for (rax = 1; rax <= 5; rax++)

    {

        rdx = rax;

        rdx = l[rdx];

        \*((long \*)(rcx + 8)) = rdx;

        rcx = rdx;

    }

*//这部分代码要求前一个结点的值小于或等于后一个结点的值，因此排序后的结点应当为：*

*//0x405320->0x405310->004052d0->0x405300->0x4052f0->0x4052e0*

*//由此可以判断输入的六个数为：6 5 1 4 3 2*

    \*((long \*)(rcx + 8)) = 0;

    for (rbp = 0, rbp <= 4; rbp++)

    {

        rbp = 0;

        rax = \*((long \*)(rcx + 8));

        rax = \*((long \*)(rax));

        if (\*((long \*)(rbx)) < rax)

            bomb();

    }

}

这个函数的栈帧中包含两个数组，其中一个是输入的6个整数的数组，另一个是包含6个指针的数组，这个指针数组分别用来指向链表的每一个结点。

其中，第一个循环要求输入的6个数两两不同且都不大于6。

第2个循环修改了函数的指针数组，将指针数组的第i个元素赋为第a[i]个结点的地址，其中a[i]即为输入的第i个整数。

第3个循环修改了链表结构，按照函数分配的指针数组中的6个元素(6个地址)的顺序，重新排列链表元素。

第4个循环依次检验链表元素，要求链表中前一个结点的数据必须不大于其后继节点的数据值。

通过EDB动态调试，可以发现程序中链表初始状态如：

*\* 1：004052d0: 00 00 00 00 00 40 52 e0 00 00 00 01 00 00 01 a6*

*\* 2：0x4052e0: 00 00 00 00 00 40 52 f0 00 00 00 02 00 00 03 d5*

*\* 3：0x4052f0: 00 00 00 00 00 40 53 00 00 00 00 03 00 00 03 a2*

*\* 4：0x405300: 00 00 00 00 00 40 53 10 00 00 00 04 00 00 03 6c*

*\* 5：0x405310: 00 00 00 00 00 40 53 20 00 00 00 05 00 00 01 92*

*\* 6：0x405320: 00 00 00 00 00 00 00 00 00 00 00 06 00 00 01 80*

因此，只需要输入6个正整数，使得链表重新排列后，能够按照从小到大的吮血排序即可。六个结点的数据从小到大排列依次是：6 5 1 2 3 4，每一个数字对应着一个结点的序号。

在输入这六个数之后，函数的指针数组l[]会依次存储第6个结点、第5个结点、第1个结点、第2个结点、第3个结点、第4个结点的地址。经过第3个循环之后，结点6的后继节点变为第5个结点，结点5的后继节点变为第1个结点，……，结点3的后继节点变为结点4，结点4的后继节点变为空。然后，再以结点6为头节点依次遍历链表，即可得到一个单调不减的序列，至此，第6关可以顺利通过。

综上所述，第6关需要输入的数据是：6 5 1 2 3 4