

**实验报告**

**实 验（三）**

题 目 Binary Bomb

二进制炸弹

专 业 计算机类

学　　 号 1190200523

班　　 级 1903002

学 生 石翔宇

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.4.23

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc70085699)

[1.1 实验目的 - 3 -](#_Toc70085700)

[1.2 实验环境与工具 - 3 -](#_Toc70085701)

[1.2.1 硬件环境 - 3 -](#_Toc70085702)

[1.2.2 软件环境 - 3 -](#_Toc70085703)

[1.2.3 开发工具 - 3 -](#_Toc70085704)

[1.3 实验预习 - 3 -](#_Toc70085705)

[第2章 实验环境建立 - 5 -](#_Toc70085706)

[2.1 Ubuntu下CodeBlocks反汇编（10分） - 5 -](#_Toc70085707)

[2.2 Ubuntu下EDB运行环境建立（10分） - 5 -](#_Toc70085708)

[第3章 各阶段炸弹破解与分析 - 6 -](#_Toc70085709)

[3.1 阶段1的破解与分析 - 6 -](#_Toc70085710)

[3.2 阶段2的破解与分析 - 7 -](#_Toc70085711)

[3.3 阶段3的破解与分析 - 8 -](#_Toc70085712)

[3.4 阶段4的破解与分析 - 9 -](#_Toc70085713)

[3.5 阶段5的破解与分析 - 10 -](#_Toc70085714)

[3.6 阶段6的破解与分析 - 12 -](#_Toc70085715)

[3.7 阶段7的破解与分析(隐藏阶段) - 14 -](#_Toc70085716)

[第4章 总结 - 17 -](#_Toc70085717)

[4.1 请总结本次实验的收获 - 17 -](#_Toc70085718)

[4.2 请给出对本次实验内容的建议 - 17 -](#_Toc70085719)

[参考文献 - 18 -](#_Toc70085720)

# 第1章 实验基本信息

## 1.1 实验目的

* 熟练掌握计算机系统的ISA指令系统与寻址方式
* 熟练掌握Linux下调试器的反汇编调试跟踪分析机器语言的方法
* 增强对程序机器级表示、汇编语言、调试器和逆向工程等的理解

## 1.2 实验环境与工具

### 1.2.1 硬件环境

* Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
* 16GB RAM
* 1TB HDD + 512G SSD

### 1.2.2 软件环境

* Windows 10 21H1
* Ubuntu 20.04 LTS

### 1.2.3 开发工具

* VSCode，CodeBlocks，gcc+gdb，edb

## 1.3 实验预习

* 上实验课前，必须认真预习实验指导书（PPT或PDF）
* 了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。
* 请写出C语言下包含字符串比较、循环、分支（含switch）、函数调用、递归、指针、结构、链表等的例子程序sample.c。
* 生成执行程序sample.out。
* 用gcc –S或CodeBlocks或GDB或OBJDUMP等，反汇编，比较。
* 列出每一部分的C语言对应的汇编语言。
* 修改编译选项-O (缺省2)、O0、O1、O2、O3，-m32/m64。再次查看生成的汇编语言与原来的区别。
* 注意O1之后无栈帧，EBP做别的用途。-fno-omit-frame-pointer加上栈指针。
* GDB命令详解 –tui模式 ^XA切换 layout改变等等
* 有目的地学习: 看VS的功能GDB命令用什么？

# 第2章 实验环境建立

## 2.1 Ubuntu下CodeBlocks反汇编（10分）

CodeBlocks运行hellolinux.c。反汇编查看printf函数的实现。

要求：C、ASM、内存(显示hello等内容)、堆栈（call printf前）、寄存器同时在一个窗口。

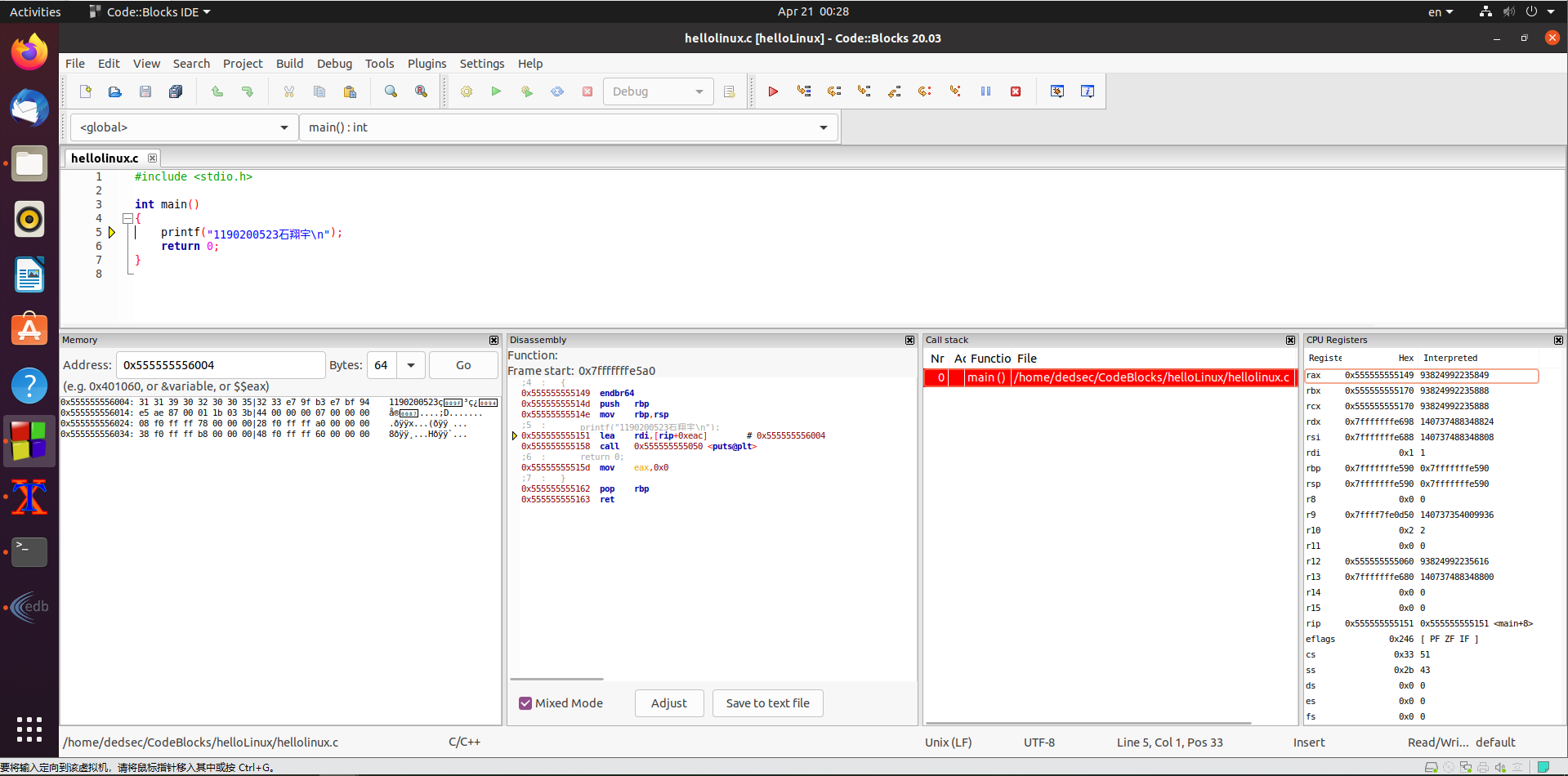


图2-1 Ubuntu下CodeBlocks反汇编截图

## 2.2 Ubuntu下EDB运行环境建立（10分）

用EDB调试hellolinux.c的执行文件，截图，要求同2.1

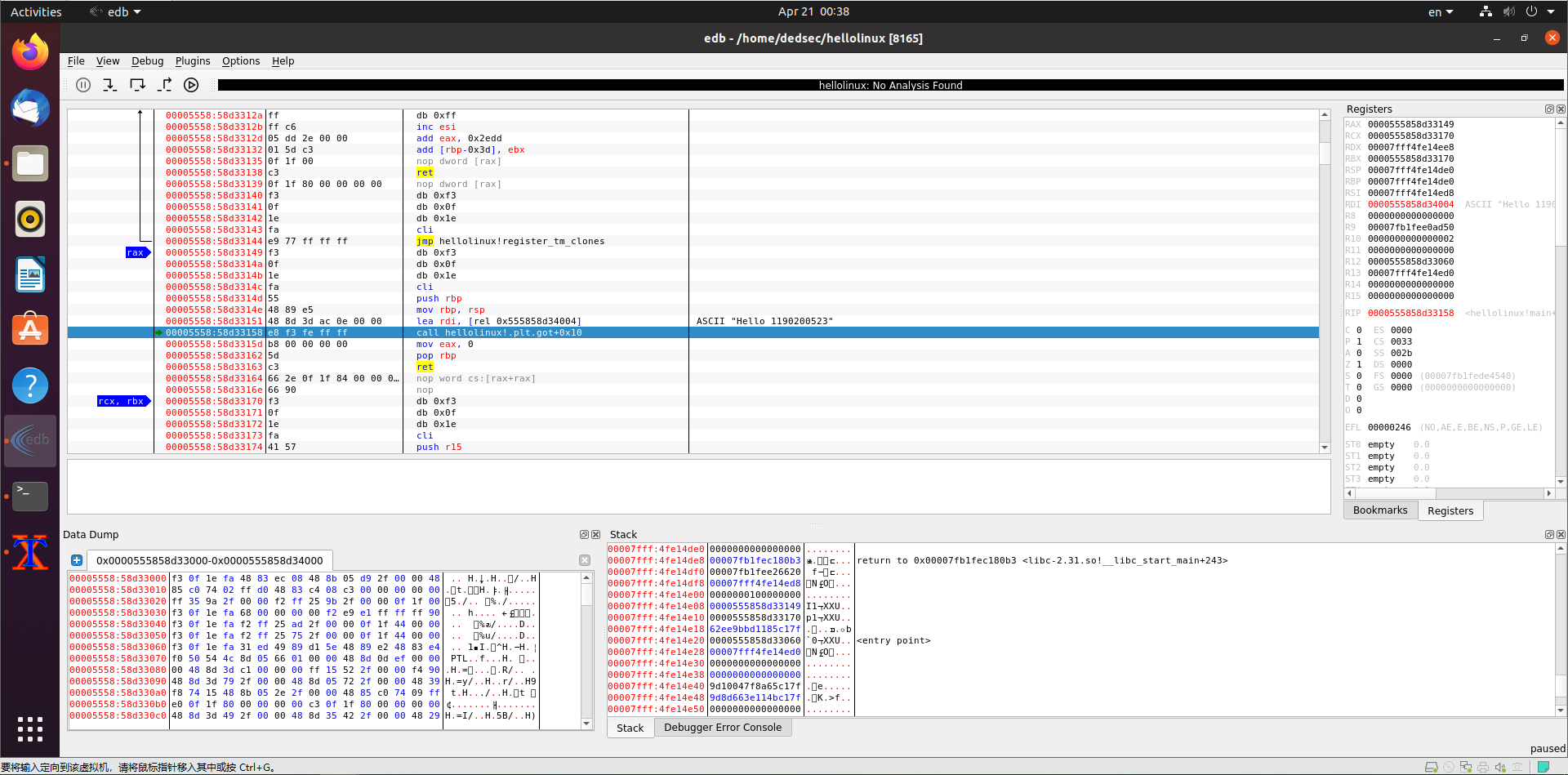


图2-2 Ubuntu下EDB截图

# 第3章 各阶段炸弹破解与分析

每阶段15分（密码10分，分析5分），总分不超过80分

## 3.1 阶段1的破解与分析

密码如下：You can Russia from land here in Alaska.

破解过程：

1. 查看反汇编代码得知此阶段比较字符串是否相等。
2. 由反汇编代码 4013fd: be 50 31 40 00 mov $0x403150,%esi得知答案字符串存在0x403150处
3. 提取得字符串16进制表示59 6f 75 20 63 61 6e 20 52 75 73 73 69 61 20 66 72 6f 6d 20 6c 61 6e 64 20 68 65 72 65 20 69 6e 20 41 6c 61 73 6b 61 2e
4. 由程序：

int main()

{

    freopen("phase1.txt", "r", stdin);

    char a[2];

    while(scanf("%s", a) != EOF)

    {

        int x = 0;

        if(a[0] >= '0' && a[0] <= '9')

            x = a[0] - '0';

        else if(a[0] >= 'a' && a[0] <= 'f')

            x = a[0] - 'a' + 10;

        x = x \* 16;

        if(a[1] >= '0' && a[1] <= '9')

            x += a[1] - '0';

        else if(a[1] >= 'a' && a[1] <= 'f')

            x += a[1] - 'a' + 10;

        printf("%c", x);

    }

    return 0;

}

转换成字符串“You can Russia from land here in Alaska.”，得到答案

## 3.2 阶段2的破解与分析

密码如下：1 2 4 7 11 16（满足第一个数为非负整数，其余依次递增1,2,3,4,5的六个整数即可）

破解过程：

1. 查看反汇编代码

401421: e8 ef 04 00 00 callq 401915 <read\_six\_numbers>

及read\_six\_numbers的有关内容得知此阶段首先输入6个整数，存储于%rbp-0x30处。

1. 由反汇编代码

401426: 83 7d d0 00 cmpl $0x0,-0x30(%rbp)

40142a: 78 07 js 401433 <phase\_2+0x1f>

可知第一个数字不能为负数

1. 40142c: bb 01 00 00 00 mov $0x1,%ebx

首先设置%ebx为1，%ebx代表当前进行到第几个数字（下标从0开始），

401442: 83 fb 05 cmp $0x5,%ebx

401445: 7f 17 jg 40145e <phase\_2+0x4a>

当%ebx达到6时跳出，

401447: 48 63 c3 movslq %ebx,%rax

令%rax = %ebx，%rax表示当前第几个数字，

40144a: 8d 53 ff lea -0x1(%rbx),%edx

令%edx = %rbx(%ebx) – 1，%edx表示当前数字的前一个，

40144d: 48 63 d2 movslq %edx,%rdx

401450: 89 d9 mov %ebx,%ecx

令%ecx = %ebx，表示当前第几个数，也即当前数字需要比前一个数字大多少，

401452: 03 4c 95 d0 add -0x30(%rbp,%rdx,4),%ecx

-0x30(%rbp,%rdx,4)表示前一个数字的值，加到%ecx上，现在的%ecx就表示当前数字期望的值，

401456: 39 4c 85 d0 cmp %ecx,-0x30(%rbp,%rax,4)

-0x30(%rbp,%rax,4)表示输入的当前数字的值，%ecx表示当前数字期望的值，将两个数字进行比较，

40145a: 74 e3 je 40143f <phase\_2+0x2b>

若两数字相等则回到0x40143f，继续循环，

40145c: eb dc jmp 40143a <phase\_2+0x26>

否则炸弹爆炸。

1. 由此可知，满足第一个数为非负整数，其余依次递增1,2,3,4,5的六个整数即为答案。

## 3.3 阶段3的破解与分析

密码如下：0 168或1 967或2 408或3 811或4 82或5 323或6 90或7 319

破解过程：

1. 查看反汇编代码

401475: be 4f 33 40 00 mov $0x40334f,%esi

及0x40334f处的有关内容得知此阶段首先输入2个整数，存储于%rbp-0x4和%rbp-0x8处。

1. 由反汇编代码

401489: 8b 45 fc mov -0x4(%rbp),%eax

40148c: 83 f8 07 cmp $0x7,%eax

40148f: 77 46 ja 4014d7 <phase\_3+0x72>

可知第一个输入应小于或者等于7

1. 由反汇编代码

401493: ff 24 c5 c0 31 40 00 jmpq \*0x4031c0(,%rax,8)

由对应位置的内容可知，第一个数分别等于0，1，2，3，4，5，6，7时分别跳到00 00 00 00 00 40 14 a1，00 00 00 00 00 40 14 e3，00 00 00 00 00 40 14 ad，00 00 00 00 00 40 14 b4，00 00 00 00 00 40 14 bb，00 00 00 00 00 40 14 c2，00 00 00 00 00 40 14 c9，00 00 00 00 00 40 14 d0处

1. 由各个对应位置内容可知，第一个数分别等于0，1，2，3，4，5，6，7时，且第二个数分别等于0xa8，0x3c7，0x198，0x32b，0x52，0x143，0x5a，0x13f，即为答案。
2. 将对应的十六进制转为十进制得到答案：0 168或1 967或2 408或3 811或4 82或5 323或6 90或7 319

## 3.4 阶段4的破解与分析

密码如下：176 2或264 3或352 4

破解过程：

1. 查看反汇编代码

40154c: be 4f 33 40 00 mov $0x40334f,%esi

及0x40334f处的有关内容得知此阶段首先输入2个整数，存储于%rbp-0x8和%rbp-0x4处。

1. 查看反汇编代码

401563: 83 f8 01 cmp $0x1,%eax

401566: 7e 05 jle 40156d <phase\_4+0x31>

401568: 83 f8 04 cmp $0x4,%eax

40156b: 7e 05 jle 401572 <phase\_4+0x36>

可知第二个整数必须大于1，小于等于4，即只能为2，3，4

1. 查看反汇编代码

401572: 8b 75 fc mov -0x4(%rbp),%esi

401575: bf 09 00 00 00 mov $0x9,%edi

40157a: e8 72 ff ff ff callq 4014f1 <func4>

向func4中传入两个参数，一个是第二个整数，一个是0x9

1. 查看func4的反汇编代码易写出其C代码

int func4(int x, int y)

{

    if(x == 1)

        return y;

    else if(x == 0)

        return 0;

    return func4(x - 1, y) + y + func4(x - 2, y);

}

1. 可根据其c代码得出要求的第一个参数，当第二个参数分别为2，3，4时，第一个参数分别为176，264，352

## 3.5 阶段5的破解与分析

密码如下：012345(123450，234501等，包含0~5这6个数字的任意排列组合成的字符串均可)

破解过程：

1. 查看反汇编代码

401599: e8 45 02 00 00 callq 4017e3 <string\_length>

40159e: 83 f8 06 cmp $0x6,%eax

4015a1: 75 25 jne 4015c8 <phase\_5+0x3b>

可知此阶段首先输入一个长度为6的字符串

1. 由0x4015a3到0x4015c6的反汇编代码可知，程序中有一个累加器x（%eax），一个总和y（%ecx）。按照输入的字符串，依次计算地址0x403200(,%rdx,4)，到相应的地址取数字。累加器x记录字符串的位置，总和y记录当前的和。
2. 其中各地址储存的数字分别为：

403200：0x02

403204：0x0a

403208：0x06

40320c：0x01

403210：0x0c

403214：0x10

403218：0x09

40321c：0x03

403220：0x04

403224：0x07

403228：0x0e

40322c：0x05

403230：0x0b

403234：0x08

403238：0x0f

40323c：0x0d

1. 由反汇编代码

4015cf: 83 f9 2f cmp $0x2f,%ecx

4015d2: 75 07 jne 4015db <phase\_5+0x4e>

可知程序要求总和y在累加完6个数字后必须等于0x2f

1. 由此我们构造一个长度为6的字符串：由数字组成，各个数字对应的地址储存的数字总和为0x2f。
2. 我们得到答案012345。

## 3.6 阶段6的破解与分析

密码如下：2 3 5 1 6 4

破解过程：

1. 查看反汇编代码

4015ef: 48 8d 75 c0 lea -0x40(%rbp),%rsi

4015f3: e8 1d 03 00 00 callq 401915 <read\_six\_numbers>

易知此阶段首先读入6个整数，存储于-0x40(%rbp)处

1. 阅读0x4015f8到0x401646的代码可大概写出原始的C代码：

    while (r12d <= 5)

    {

        rax = \*(a\_40 + r12d);

        rax--;

        if(rax > 5)

            explode();

        ebx = r12d + 1;

        while(ebx <= 5)

        {

            rax = r12d;

            if(\*(a\_40 + ebx) == \*(a\_40 + rax))

                explode();

            ebx++;

        }

    }

此段代码的功能主要为判断输入6个数字两两互不相等。

1. 阅读0x401648到0x401676的代码可大概写出原始的C代码：

    for (int esi = 0; esi <= 5; esi++)

    {

        rax = 1;

        rdx = 0x4052d0;

        while(a\_40[esi] > rax)

        {

            rdx = \*(rdx + 1);

            rax++;

        }

        a\_70[esi] = rdx;

    }

此段代码功能主要为按照输入的数字，找到以0x4052d0开始的链表相应位置的数字的地址（例如输入为3，则找到以0x4052d0开始的链表的第3个数字的地址），存储在-0x70(%rbp,%rcx,8)处（代码中为a\_70）

也就是将原始的链表重新排序组合，存储在-0x70(%rbp,%rcx,8)处

1. 阅读0x 401678到0x 4016ab的代码可大概写出原始的C代码：

    int \*rcx = a\_70[0];

    rax = 1;

    while(rax <= 5)

    {

        rdx = a\_70[rax];

        \*(rcx + 1) = rdx;

        rax++;

        rcx = rdx;

    }

此代码的主要功能是为新的链表重新建立从前到后的链接

1. 阅读0x4016ad到0x4016d6的代码可大概写出原始的C代码：

int \*rbx = a\_70[0];

    r12d = 0;

    while(r12d <= 4)

    {

        if(\*rbx > \*(\*(rbx + 1)))

            explode();

        else

        {

            rbx = \*(rbx + 1);

            r12d++;

        }

    }

此代码的主要功能是在新的链表上从前到后比较相邻的两个数，需满足前面的数不大于后面的数，即新生成的数字序列是单调不降的。

1. 查看原始链表中的数字依次为：

1：00 00 01 f8

2：00 00 00 92

3：00 00 01 00

4：00 00 02 82

5：00 00 01 0e

6：00 00 02 7f

1. 欲将其从小到大排列，则对应下标为2 3 5 1 6 4，得到答案。

## 3.7 阶段7的破解与分析(隐藏阶段)

密码如下：进入隐藏阶段的字符串：DrEvil，解开需要的密码：47

破解过程：

1. 观察到反汇编代码段phase\_defused中0x401aea出现了对于secret\_phase的引用，向上查找到

401abf: be a2 33 40 00 mov $0x4033a2,%esi

401ac4: 48 8d 7d b0 lea -0x50(%rbp),%rdi

401ac8: e8 2a fd ff ff callq 4017f7 <strings\_not\_equal>

401acd: 85 c0 test %eax,%eax

401acf: 75 e2 jne 401ab3 <phase\_defused+0x37>

表明地址0x4033a2存储着进入隐藏阶段需要输入的字符串，查找得44 72 45 76 69 6c，变成字符串的形式为DrEvil

1. 观察secret\_phase的反汇编代码，由0x401716到0x401731可知，本阶段首先输入一个整数，要求这个整数小于等于0x3e9
2. 0x401733和0x401735的代码负责传递fun7的参数，%edi被赋为地址0x4050f0，%esi是输入的整数。
3. 阅读fun7的代码可大概写出原始的C代码：

int fun7(int \*edi)

{

    if(\*edi > esi)

    {

        edi = \*(edi + 1);

        return fun7(edi) \* 2;

    }

    else if(\*edi != esi)

    {

        edi = \*(edi + 2);

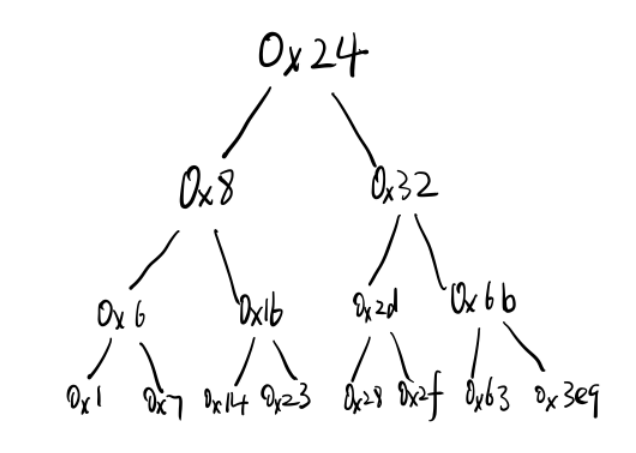
        return fun7(edi) \* 2 + 1;

    }

    return 0;

}

由代码可知，数据在以地址0x4050f0开头的结构中是以链表的形式存在。Fun7中，\*edi以位于0x4050f0的数据为开始，再以树形结构依次跳转到相应位置的数据，最后返回值。由相应的数据得出的树形结构如下：



1. 由反汇编代码

40173f: 83 f8 05 cmp $0x5,%eax

401742: 75 1d jne 401761 <secret\_phase+0x50>

可知fun7的返回值需要为5，构造答案为0x2f，即47

# 第4章 总结

## 4.1 请总结本次实验的收获

1. 学会了gdb、edb、objdump等工具的使用，更加熟练地利用这些工具来调试程序
2. 对汇编语言更加熟悉，对于各种汇编指令有了更深的理解

## 4.2 请给出对本次实验内容的建议

1. 使用CMU原版实验收获很大，建议以后章节也能继续使用原版实验

注：本章为酌情加分项。

# 参考文献

[1] 大卫 R.奥哈拉伦，兰德尔 E。布莱恩特. 深入理解计算机系统[M]. 机械工业

出版社.2017.7