

**实验报告**

**实 验（三）**

题 目 Binary Bomb

二进制炸弹

专 业 计算机类

学　　 号 1190202405

班　　 级 1903005

学 生 林逸灏

指 导 教 师 史先俊

实 验 地 点 G709

实 验 日 期 2021.04.21

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc19846)

[1.1 实验目的 - 3 -](#_Toc19505)

[1.2 实验环境与工具 - 3 -](#_Toc15894)

[1.2.1 硬件环境 - 3 -](#_Toc16679)

[1.2.2 软件环境 - 3 -](#_Toc1640)

[1.2.3 开发工具 - 3 -](#_Toc311)

[1.3 实验预习 - 3 -](#_Toc20293)

[第2章 实验环境建立 - 4 -](#_Toc3866)

[2.1 Ubuntu下CodeBlocks反汇编（10分） - 4 -](#_Toc24906)

[2.2 Ubuntu下EDB运行环境建立（10分） - 4 -](#_Toc18077)

[第3章 各阶段炸弹破解与分析 - 5 -](#_Toc4144)

[3.1 阶段1的破解与分析 - 5 -](#_Toc24389)

[3.2 阶段2的破解与分析 - 5 -](#_Toc25931)

[3.3 阶段3的破解与分析 - 5 -](#_Toc21845)

[3.4 阶段4的破解与分析 - 5 -](#_Toc11024)

[3.5 阶段5的破解与分析 - 5 -](#_Toc29641)

[3.6 阶段6的破解与分析 - 5 -](#_Toc18838)

[3.7 阶段7的破解与分析(隐藏阶段) - 6 -](#_Toc20276)

[第4章 总结 - 7 -](#_Toc9339)

[4.1 请总结本次实验的收获 - 7 -](#_Toc16563)

[4.2 请给出对本次实验内容的建议 - 7 -](#_Toc27793)

[参考文献 - 8 -](#_Toc10577)

# 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

### X64 CPU；2GHz；2G RAM；256GHD Disk 以上

### 1.2.2 软件环境

Windows7 64位以上；VirtualBox/Vmware 11以上；Ubuntu 16.04 LTS 64位/优麒麟 64位；

### 1.2.3 开发工具

GDB/OBJDUMP；EDB；KDD等

## 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、O3、Og、-m32/m64。再次查看生成的汇编语言与原来的区别。**
* **注意O1之后缺省无栈帧，RBP为普通寄存器。用 -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。

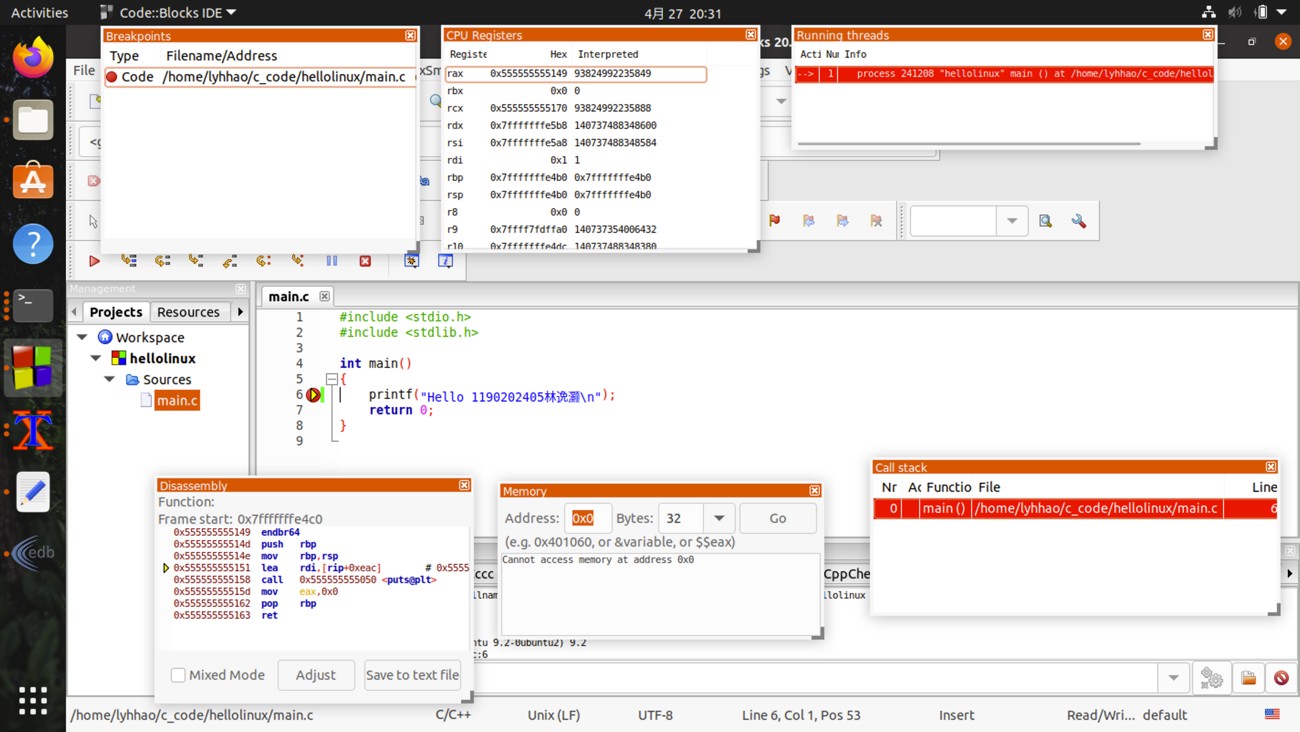


图2-1 Ubuntu下CodeBlocks反汇编截图

## 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

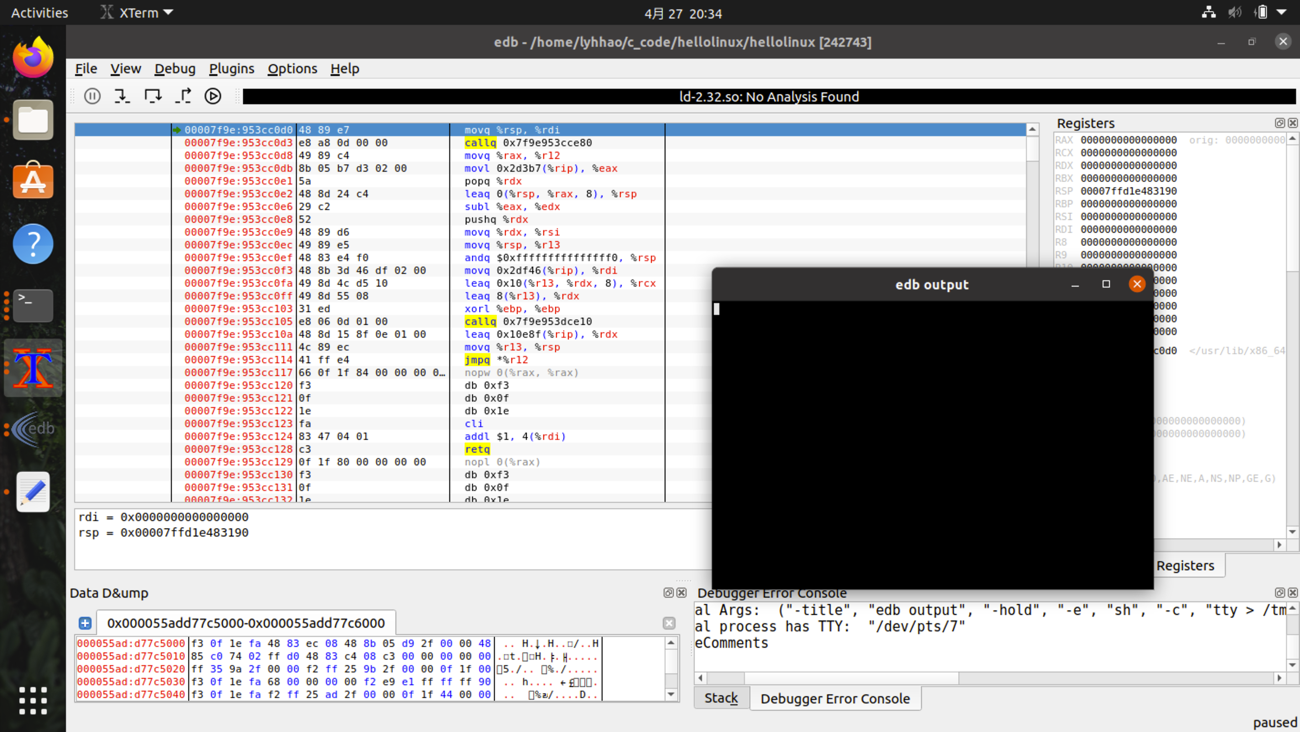


图2-2 Ubuntu下EDB截图

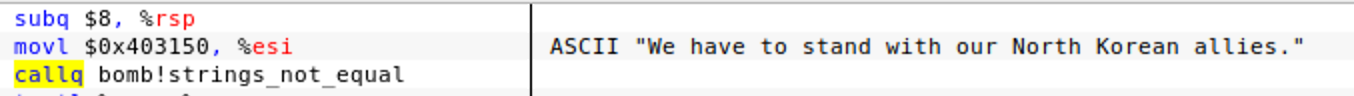
# 第3章 各阶段炸弹破解与分析

每阶段40分，密码20分，分析20分，总分不超过80分

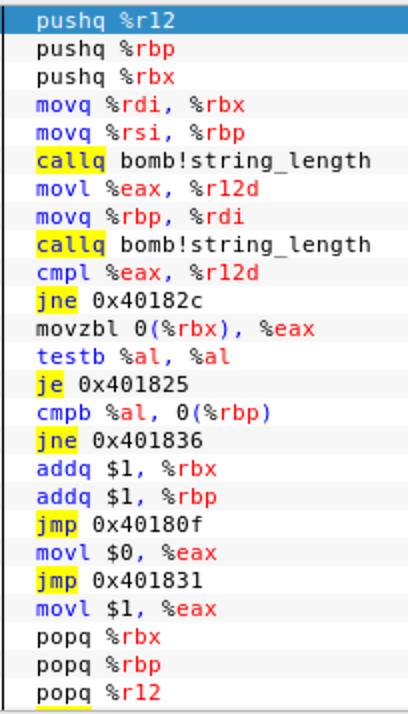
## 3.1 阶段1的破解与分析

密码如下：We have to stand with our North Korean allies.

破解过程：



反汇编代码中，将 0x403150 地址存储到%esi中，0x403150地址中存储的为字符串: “We have to stand with our North Korean allies.” 然后调用 bomb!strings\_not\_equal 函数，该函数部分反汇编代码如下：

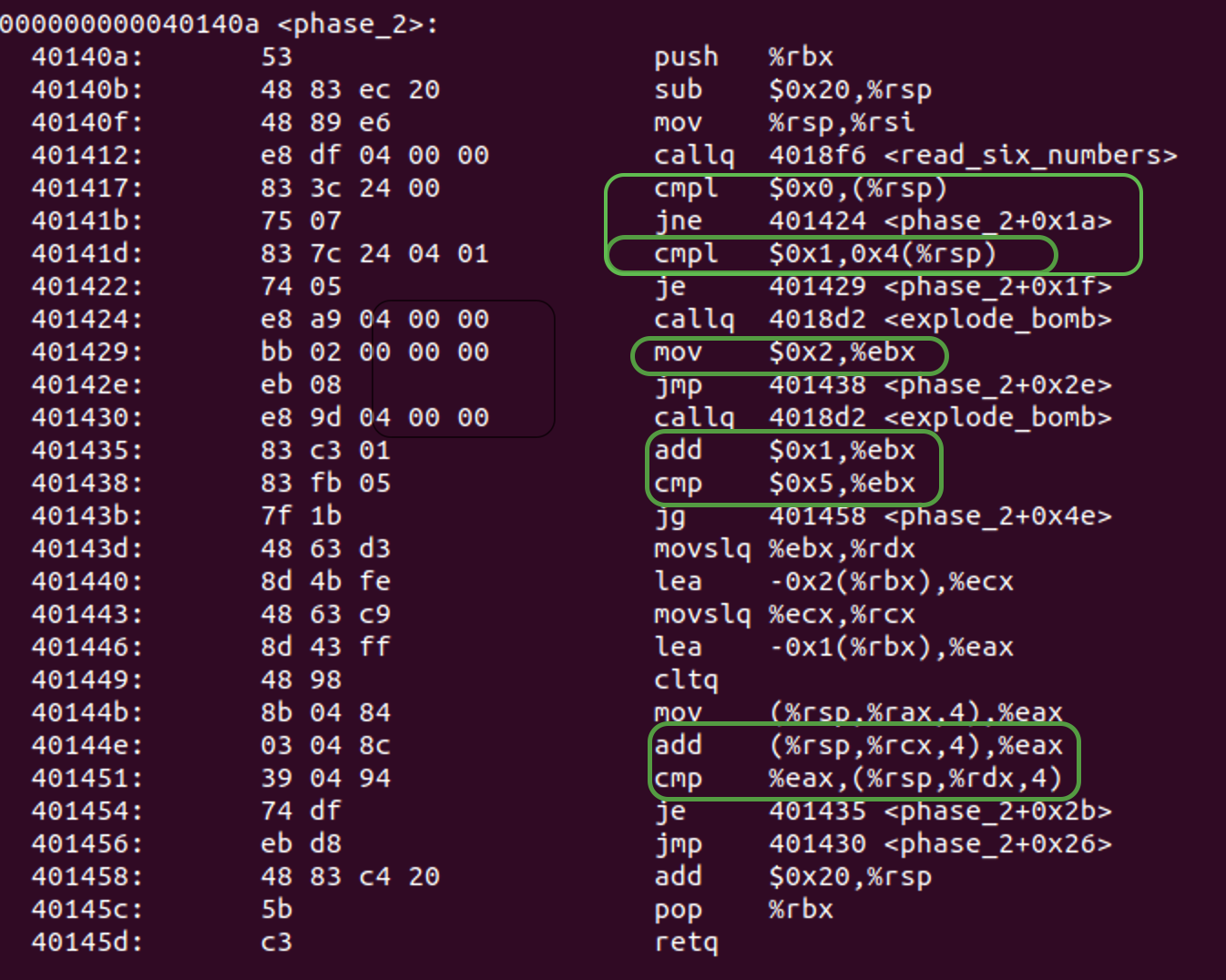


可知，该函数用于判断输入字符串与程序内设置的字符串是否相等。所以阶段1的密码为：We have to stand with our North Korean allies.

## 3.2 阶段2的破解与分析

密码如下：00 01 01 02 03 05

破解过程：



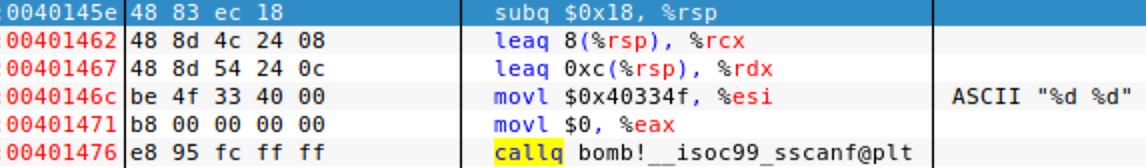
分析 phase\_2 得，由函数 read\_six\_numbers 可猜测会输入六个 int 型的数值。首先 cmpl $0x0, (%rsp)即比较 0 和(%rsp)且不为0 时爆炸，故第一个数为0。然后cmpl $0x1, 0x4(%rsp)即比较 0 和0x4(%rsp)且不为1 时爆炸，为1时跳转到0x401429。往后面分析得，%ebx的初始值为 2，然后到 401438 判断其是否小于等于 5，并将此条件作为循环条件，每次循环一次，%ebx 都加 1，到%ebx=5 时跳出循环。

在每次循环内，经过movslq与lea的赋值语句，可知%rcx与%rax中存的值相差为1。通过 mov (%rsp,%rax,4),%eax，add (%rsp,%rcx,4), %eax 可知每次循环时计算的当前循环体的变量的值为该数列中的前两个的值之和，即斐波那契数列，初始的两个值为00，01，由此可以推断出剩下4个数字的数值。这六个数为：00，01，01，02，03，05.

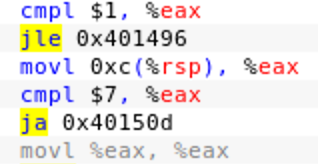
## 3.3 阶段3的破解与分析

密码如下： 4 0

破解过程：



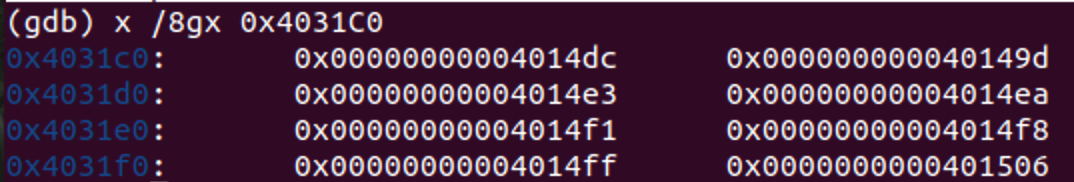
先为 phase\_3部分的函数分配栈空间，从反汇编代码中可以查看 0x40334f 位置，得知输入类型为”%d %d”两个整型数据。



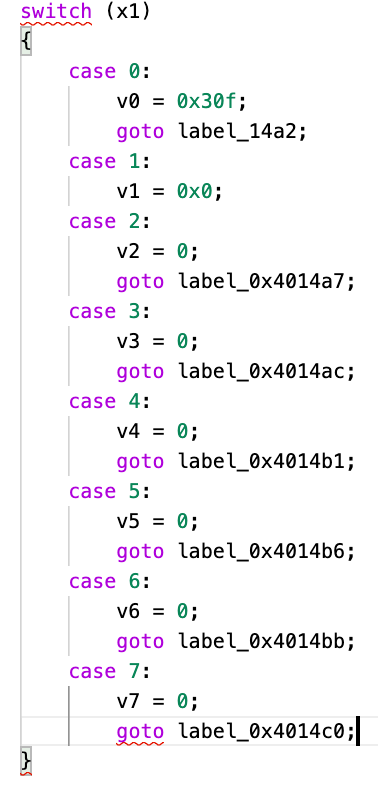
内存地址 0x401496 与 0x40150d 位置执行的操作都为，即爆炸，这段代码对输入的第一个参数做了限制，即 1< x1<=7.



然后根据 %rax 中的数据进行跳转，跳转表为：



根据此跳转表，结合对应的反汇编代码，写出对应的伪代码



再按这段反汇编代码进行计算，判断，对输入的 x1 进行加减运算，若得出的结果等于 x2，则拆弹成功。此处代码有 x1<=5的限制条件。

取 x1 = 4，按反汇编代码计算的 x2=0，为正解，拆弹成功。

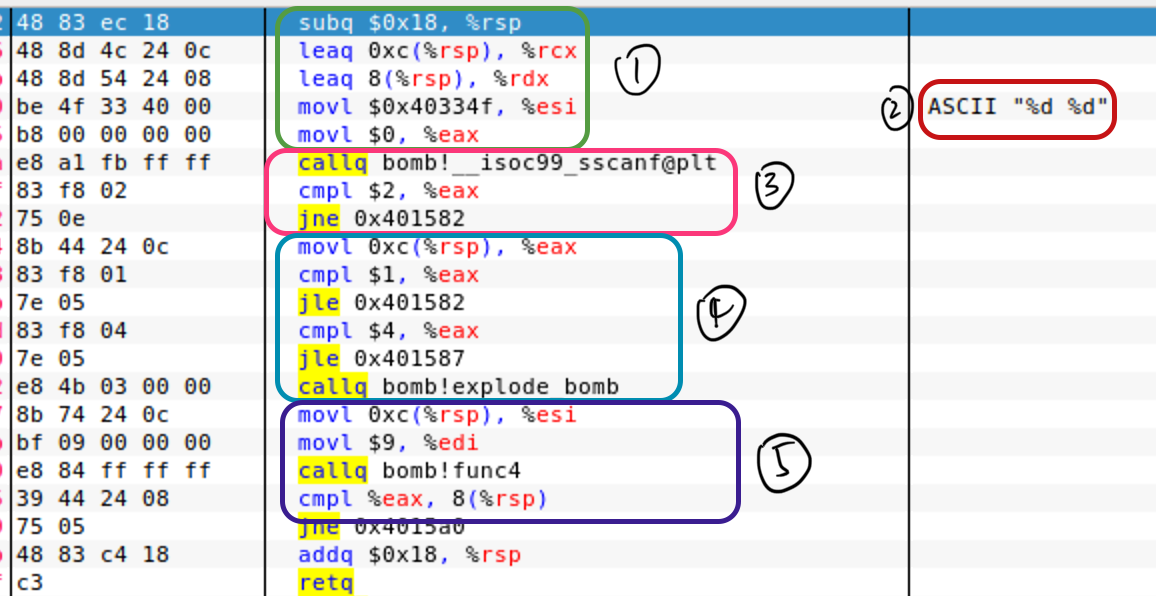
## 3.4 阶段4的破解与分析

密码如下：176 2

264 3

352 4

破解过程：



Phase\_4的汇编代码如上图所示，首先可以看到：

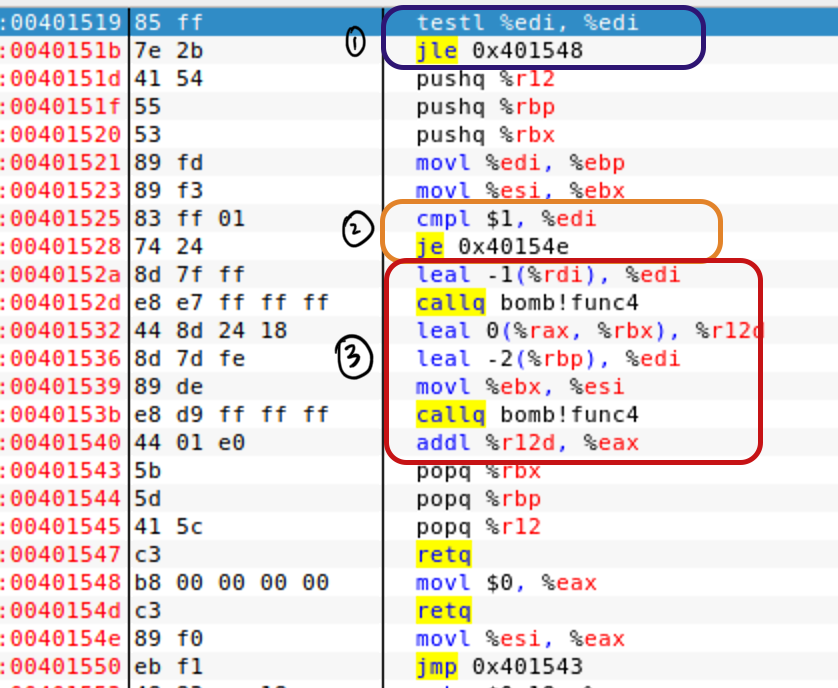
①部分的代码在栈里分配了存储空间，参数1在 8(%rsp) 中，参数2在0xc(%rsp) 中。

②处为内存地址0x40334f 处的内容，表示输入为两个整型变量。

③处的Call bomb！\_\_isoc99\_sscanf@plt函数说明调用函数读入参数。cmpl $2, %eax表明如果%eax不等于2，则调用explode\_bomb函数，直接爆炸，因此phase\_4必须读入两个参数。

④处的代码表示把参数2赋值给寄存器%eax，并分别于1和4比较，当%eax小于等于1和大于4时都会触发explode\_bomb，说明参数2应大于1，并小于等于4，即 1<x2<=4。

⑤处的代码表示把参数2赋值给寄存器%esi，把$9赋值给寄存器%edi，并将这两个寄存器的值作为参数传递给func4 （callq bomb!func4），然后将func4函数返回值与输入的参数1（8(%rsp)）比较，若相等，则通过阶段4。



func4部分的汇编代码如上图所示：

该函数有两个参数，设%edi的参数为n，%esi的参数为m（即输入的参数2）

①处的代码判断n是否为0，若为0，直接返回0。

②处的代码将n与1比较，若等于1，则返回m。

③处代码为一段递归，调用了两次func4。第一次调用时，参数为func4(n-1, m)；第二次调用时，参数为func4(n-2, m)。该段递归代码返回func4(n-1, m)+ func4(n-2, m)+m。

因此，func4的对应C语言代码为：

|  |
| --- |
| int func4(int n, int m) {  if (n == 0) {  return 0;  }  if (n == 1) {  return m;  }  if (n > 1) {  return func4(n-1, m) + func4(n-2, m) + m;  }  } |

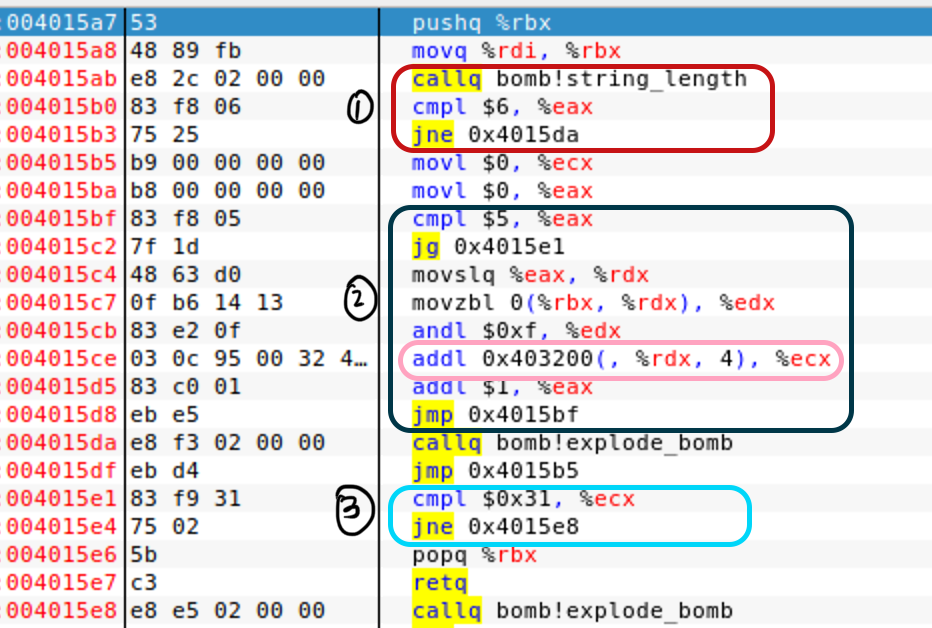
其中m=9. 代入n=2，得176；代入n=3，得264；代入n=4，得352。

因此，阶段四的通关密码为176 2或264 3或352 4三者任一。

## 3.5 阶段5的破解与分析

密码如下：444103

破解过程：



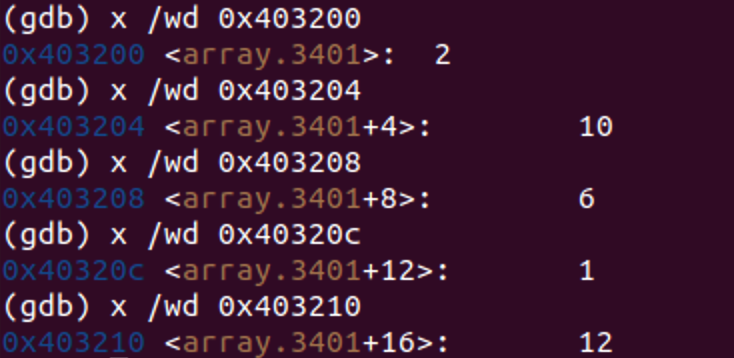
上图为phase\_5的反汇编代码，代码分析如下：

①处代码为判断输入的字符数目，若不为6则爆炸。

②处的代码为循环体，先判断了循环条件，%eax的值从0开始，每次循环结束时加1，共循环6次，即遍历了输入的字符串。

movslq指令将%eax的值做符号扩展传递给%rdx。movzbl 0(%rbx, %rdx), %ebx和andl $0xf, %ebx将输入做了零扩展传送到%ebx中，再取%ebx中的后四位。

addl 0x403200(, %rdx, 4), %ecx，将%ecx中的值与地址0x403200(, %rdx, 4)中的值进行相加，查看这部分的地址的值，如下图所示：



③处的代码表示经过循环体后%ecx中的值应为0x31即十进制中的49，否则将会爆炸。为了使六个数的值为49，可以取12+12+12+10+2+1，此时每次循环中的%rdx分别对应444103，即为阶段5的密码。

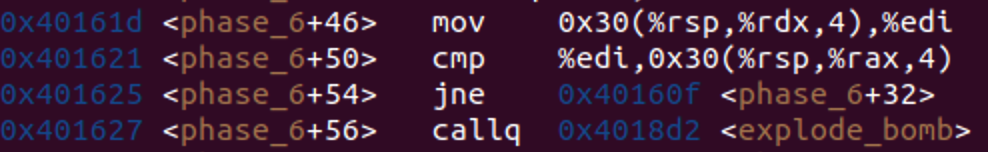
## 3.6 阶段6的破解与分析

密码如下：6 1 5 4 2 3

破解过程：



由read\_six\_numbers可知，本阶段读取6个数字。



该段代码判断输入的六个数字中是否有相同的数字，若有相同的，则直接爆炸。说明输入的六个数据应均不相等。

对之后的代码分析可知，phase\_6的作用是对链表的排序，通过用户的输入数据对原来的链表进行从大到小的顺序排列，因此，我们去查看原有链表的数据大小。



该段代码表明内存地址0x4052d0可能为原有链表的存储地址，我们查看这部分的值，如下图所示：

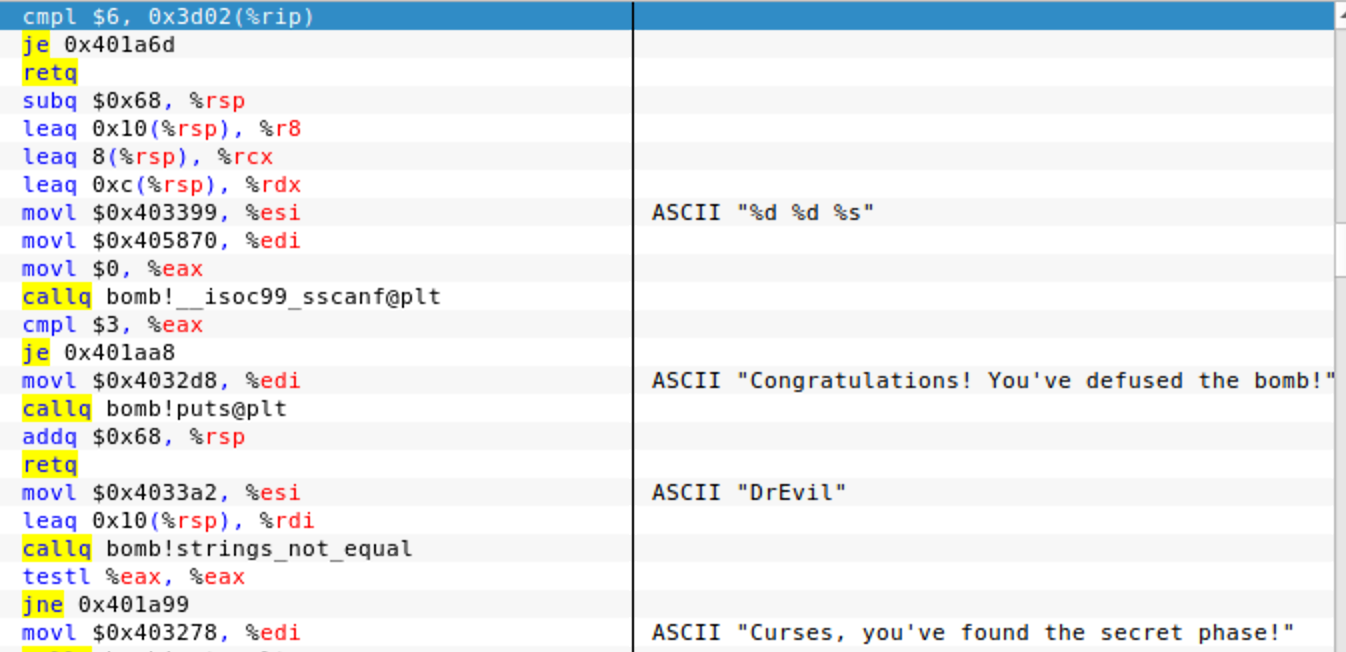


由该图可知，node6>node1>node5>node4>node2>node3

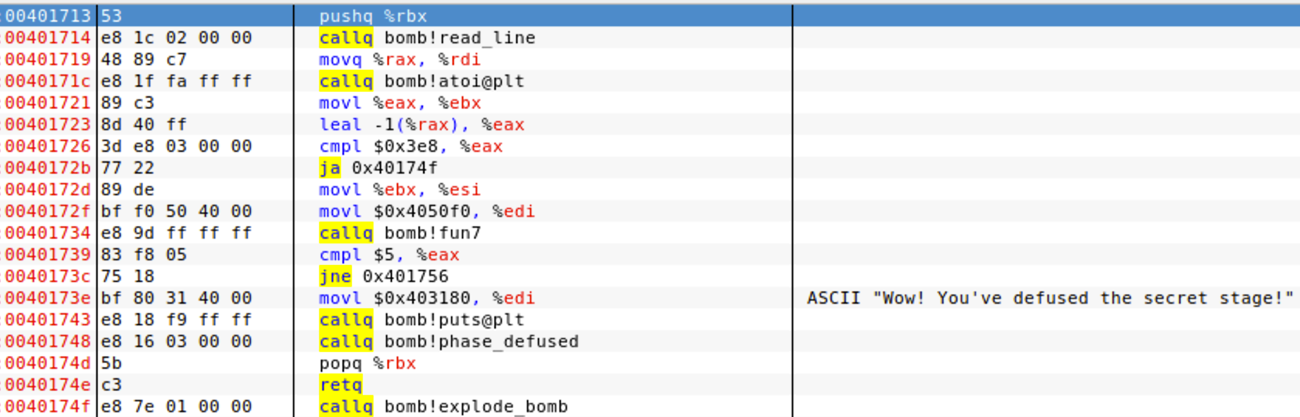
## 3.7 阶段7的破解与分析(隐藏阶段)

密码如下：47

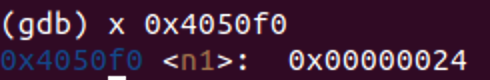
破解过程：



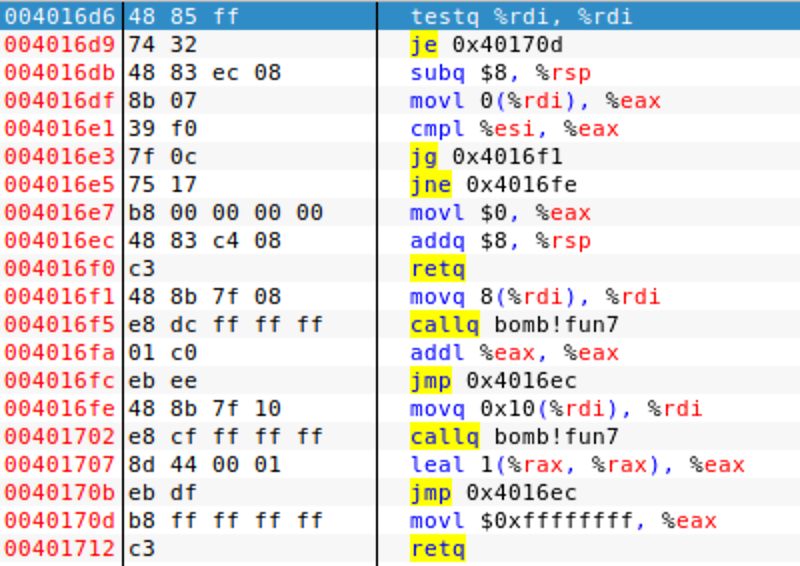
由phase\_defused处的反汇编代码可知，要想进入隐藏阶段，需要解决完前六个阶段，并且需要在阶段4输入的两个数字后面加上一串字符串：”DrEvil”，输入后即可进入隐藏阶段。



然后分析secret\_phase，汇编代码如上图所示，输入为一个字符串，并转化为十进制数字。



然后将内存地址0x4050f0中的值0x24（即36），赋值给%edi，调用fun7函数，然后判断fun7函数返回值结果是否5，若为5，则程序运行结束。



上图为fun7的反汇编代码，是一个递归调用。由以上代码可知

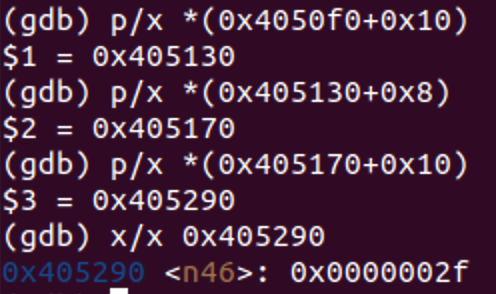
若\*a1>a2, a1=\*(a1+8), call fun7, %eax=%eax\*2;

若\*a1<a2, a1=\*(a1+16), call fun7, %eax=%eax\*2+1;

若\*a1==a2, return。

由此可知，递归树中最深层的%eax=0，往上弹栈递归。由fun7得出的结果需要为5，而计算结果为5的产生过程为：5=((0\*2+1)\*2)\*2+1

则递归时a2=\*(\*(\*(a1+0x10)+0x8)+0x10)



可得结果为0x2f，十进制为47，即为隐藏阶段的破译密码。

# 第4章 总结

## 4.1 请总结本次实验的收获

1.学会了edb，gdb的基本操作，了解了他们的相关命令。

2. 对C语言下字符串比较、循环、分支（含switch）、函数调用、递归、指针、结构、链表等有了更深刻的理解。

3.对C语言程序以及汇编语言有了更深层次的理解。

## 4.2 请给出对本次实验内容的建议

老师讲解gdb、edb时，由于时间仓促，不太细致，很多内容需要上网检索，但网上的内容鱼龙混杂且与教材内容有较大出入，希望能增加一些相关的较为权威的网站的链接。

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.