哈爾濱工業大學

实验报告

实 验（三）

题　　目 Binary Bomb

二进制炸弹

专　　业 计算学部

学　　号 120L021809

班　　级 2003006

学　　生 刘雨尘

指 导 教 师 吴锐

实 验 地 点 G712

实 验 日 期 2022.4.01

计算学部

目 录

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 指令系统和寻址方式
- 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

基于 X64 的处理器，64 位操作系统，2.9GHZ，16GRAM，512GDisk

1.2.2 软件环境

Windows10 64 位，VMWare 16，Ubuntu 16.04

1.2.3 开发工具

VS Code 64 位；Codeblocks 64 位；vim+gcc；gdb；edb

1.3 实验预习

1. 上实验课前，必须认真预习实验指导书（PPT 或 PDF）
2. 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。
3. 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。
4. 生成执行程序 sample.out。
5. 用 gcc －S 或 CodeBlocks 或 GDB 或 OBJDUMP 等，反汇编，比较。
6. 列出每一部分的 C 语言对应的汇编语言。
7. 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。

8. 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。
9. GDB 命令详解 –tui 模式 ^XA 切换 layout 改变等等
10. 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 hello.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。

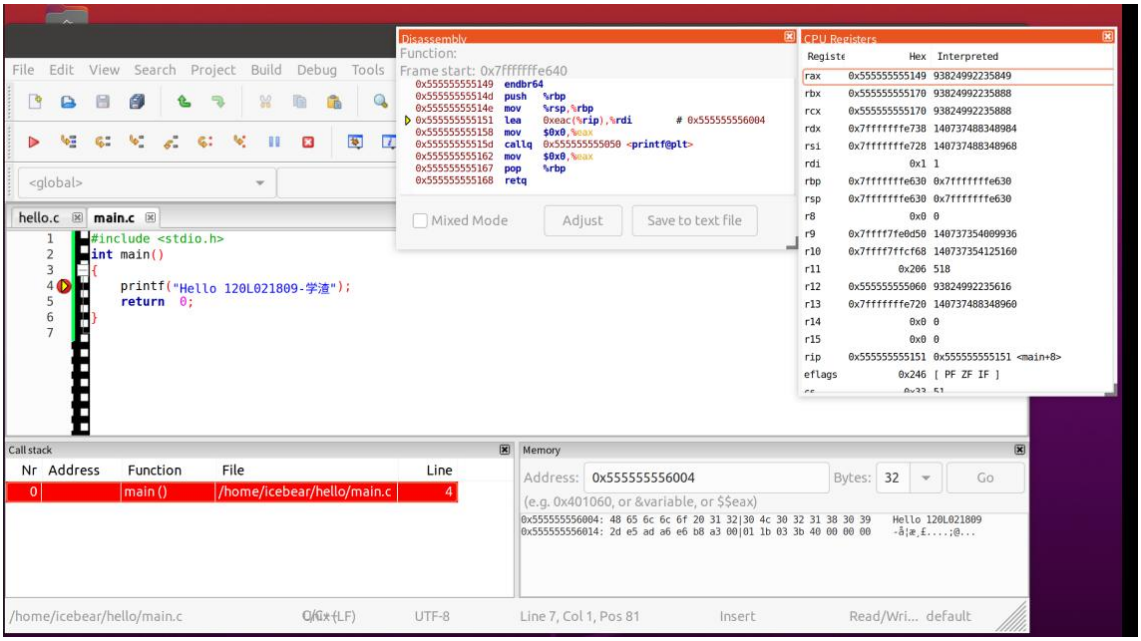


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

2.2 Ubuntu 下 EDB 运行环境建立（10 分）

用 EDB 调试 hello.c 的执行文件，截图，要求同 2.1

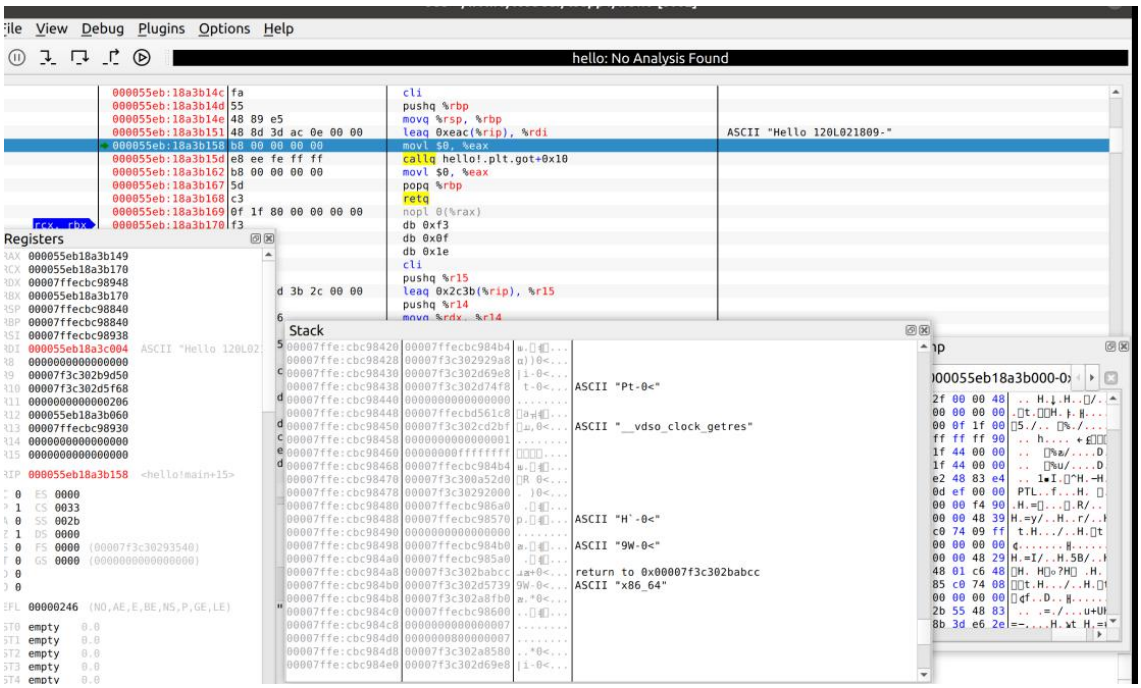
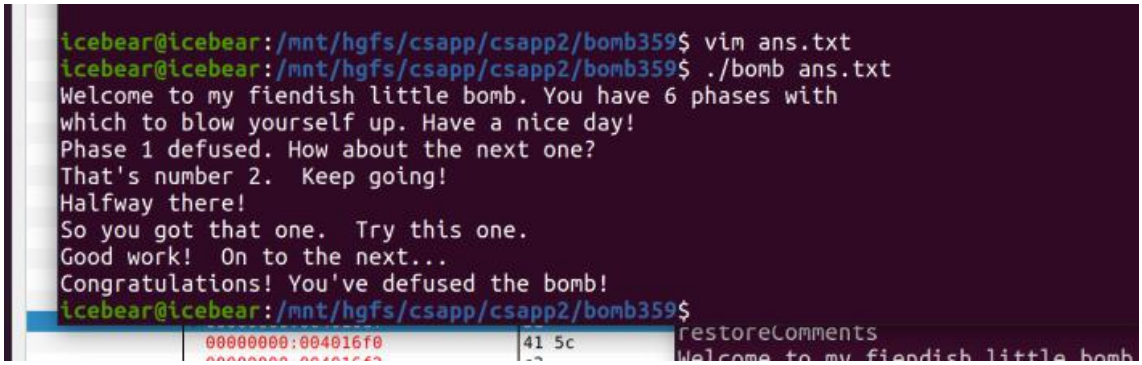


图 2-2　Ubuntu 下 EDB 截图

第 3 章 各阶段炸弹破解与分析

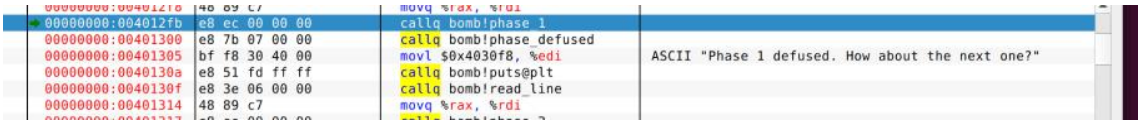
每阶段 30 分，密码 10 分，分析 20 分，总分不超过 80 分

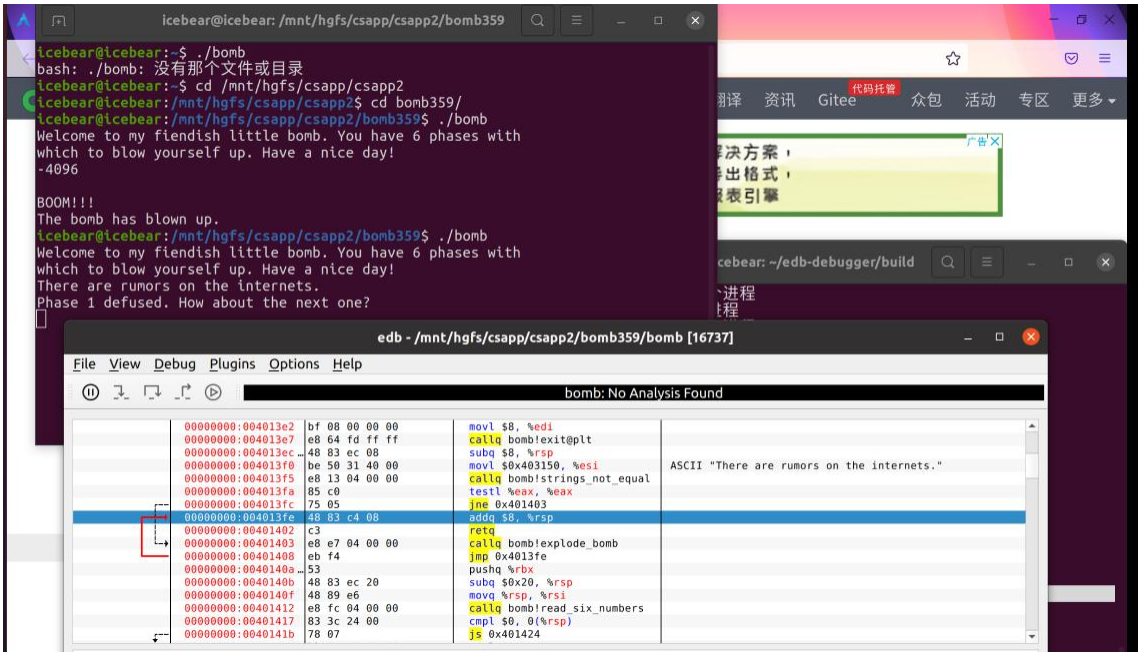


3.1 阶段 1 的破解与分析

密码如下：There are rumors on the internets.

破解过程：从 callq bomb!phase_1 处跳转，可以看到汇编语言中存在“movl $0x403150 %esi”的语句，即将被比较的字符串地址存入寄存器传递给字符串比较函数，如果输入的字符串与传入的字符串相等，则返回到“bomb!phase_defused”语句，意为拆弹成功；若字符串不相等，则返回到“bomb!exploded”函数，打印拆弹失败字样，结合后面的字符串相等判断即可看出 bomb1 的结果即为该字符串，输入密码验证成功。





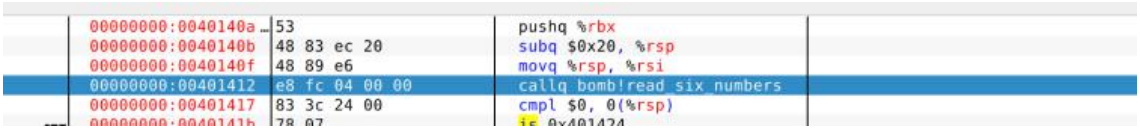
3.2 阶段 2 的破解与分析

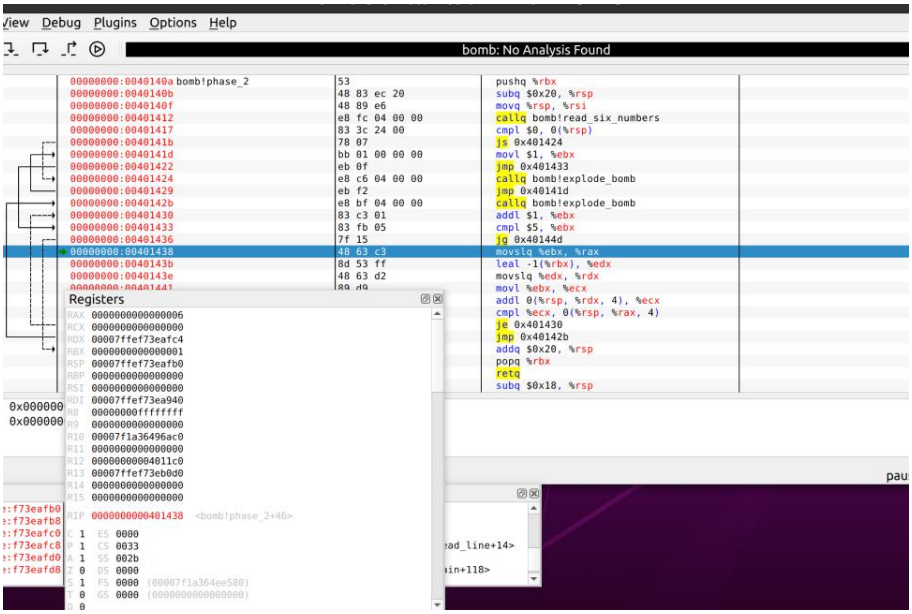
密码如下：0 1 3 6 10 15

破解过程：

反汇编中，follow“bomb！phase_2”函数得到如下结果，可以观察到存在对“bomb!read_six_numbers”函数的调用，即可明确密码为输入 6 个数字，查看函数可知格式为“%d %d %d %d %d %d”。该函数调用结束后，观察后续汇编程序可知，其后为一个循环判断语句。

紧随其后的第一行“cmpl $0, 0(%rsp)”可知 密码第一个数为 0，若输入不为 0 则跳转至地址 0x401424 即爆炸，为 0 则 ebx 赋值为 1，即循环标志 i=i+1。后面语言中，将寄存器 ecx 所保存的值与输入的第 i 位密码所比较，其中每次循环 ecx 中的值增加 1。



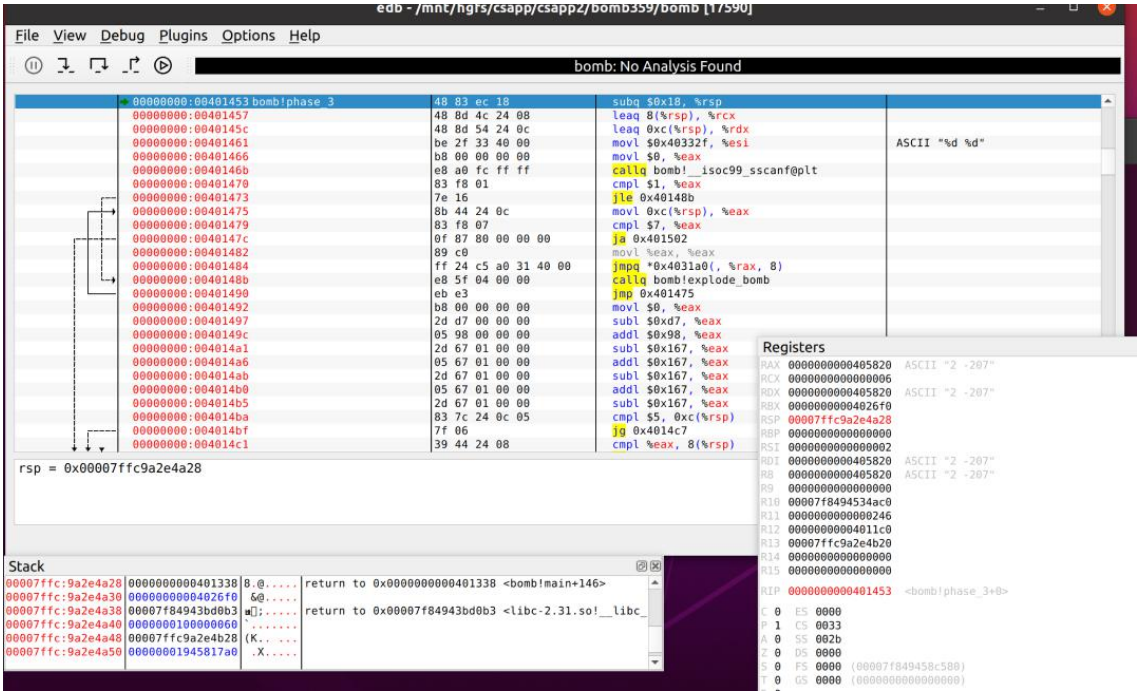


3.3 阶段 3 的破解与分析

密码如下：2 -207

破解过程：

首先输入两个数字使汇编程序得以运行，进入 phase_3 函数中，可以看到 ASCII 码提示输入格式“%d %d”，读入后程序首先会进行第一个数字的判断，若小于等于 1 则爆炸，随后进行第二个数字的判断，若大于等于 7 则炸弹爆炸，若条件都满足则继续。经过运行程序发现，最终结果将是有第一个数字计算后保存在 eax 中保存的数据，使第二个数字与之相同即可，由于是 fffffff31 所以输入-207，综上第三个密码为 2 -207.

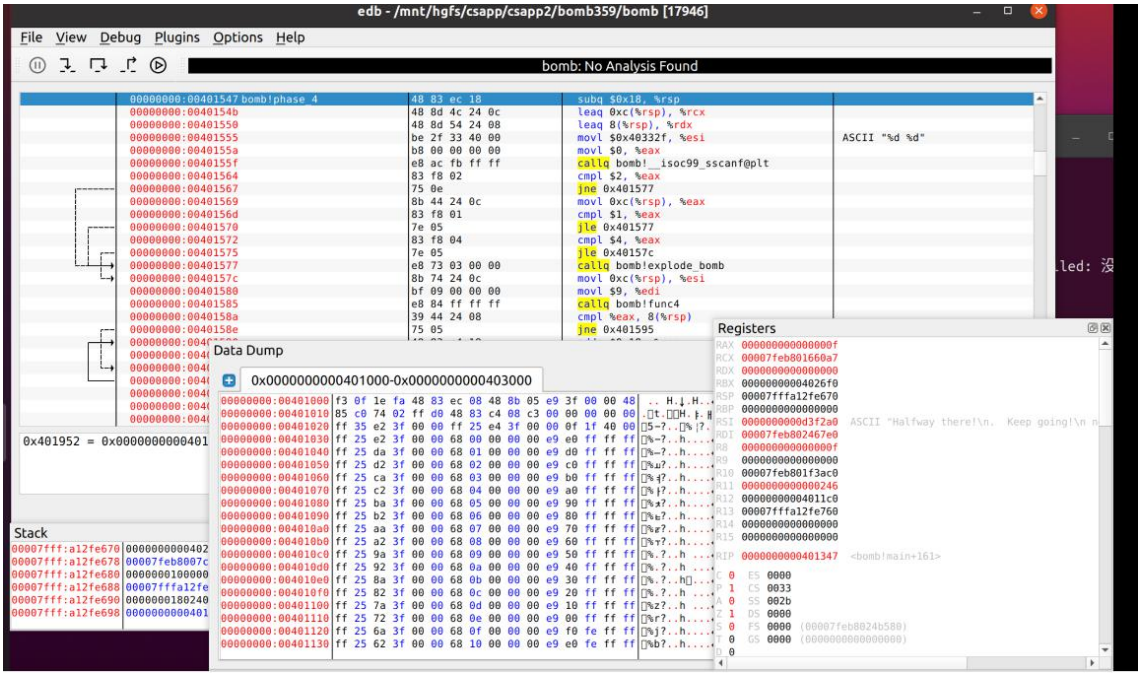


3.4 阶段 4 的破解与分析

密码如下：176 2

破解过程：

进入反汇编后得到的函数“bomb!phase_4”可以看到，输入格式要求为两个整数。其中保存在低位的数要求小于等于 4 且大于 1.高位数将被传送进函数“func4”中，并且由反汇编结果可得知，func4 为一个递归函数，其表达式可以理解为一个类斐波那契数列：f(n)=f(n-1)+f(n-2)+x，x、n 为传递的参数，本题中 x 为 2，递归次数为 9，计算可得结果为 176.输入 176 2 验证，结果正确。



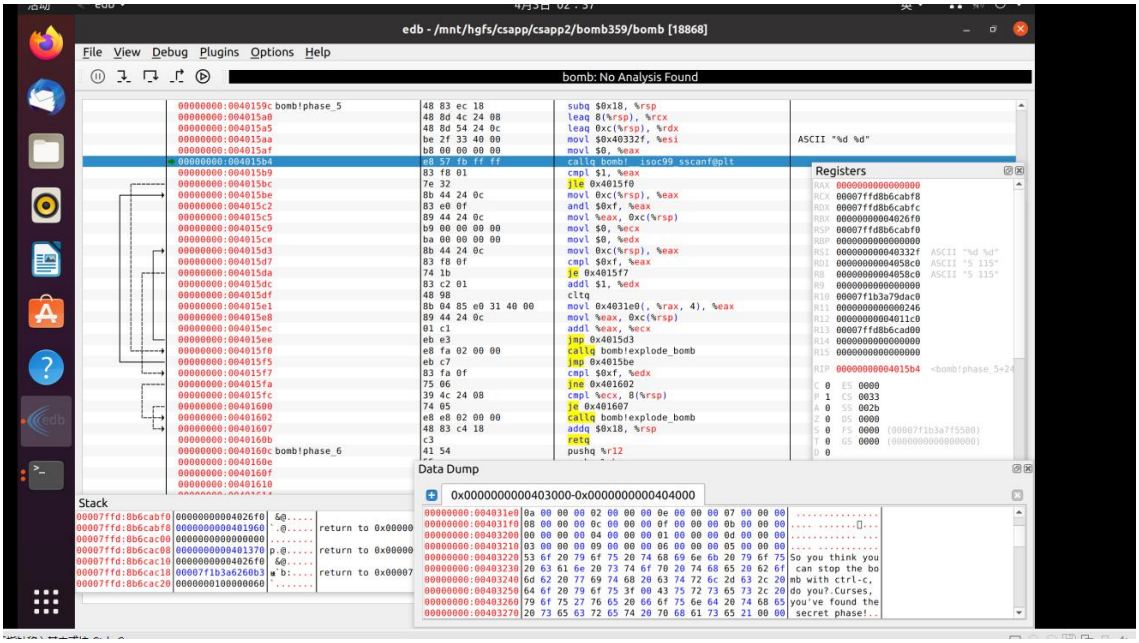
3.5 阶段 5 的破解与分析

密码如下：5 115

破解过程：

检查反汇编结果可以看到输入格式为输入两个整数，其中 0x4031e0 为指针指向一个 16 位数组，按照一定规律使用 0-15 排列，分析该函数可知是一个循环取值函数，定义 i 为循环次数，初始化为 0，循环一次增加 1.分析该函数可知函数安全退出的条件为最后取得的值恰为 15，且函数循环了 15 次，而当 eax 初值取 5 时，将按照 12→3→7→11→13→9→4→8→0→10→2→14→6→15 的顺序取得 15，而此时恰好循环了 15 次，取得的值之和为 115.

输入 5 115 验证正确。



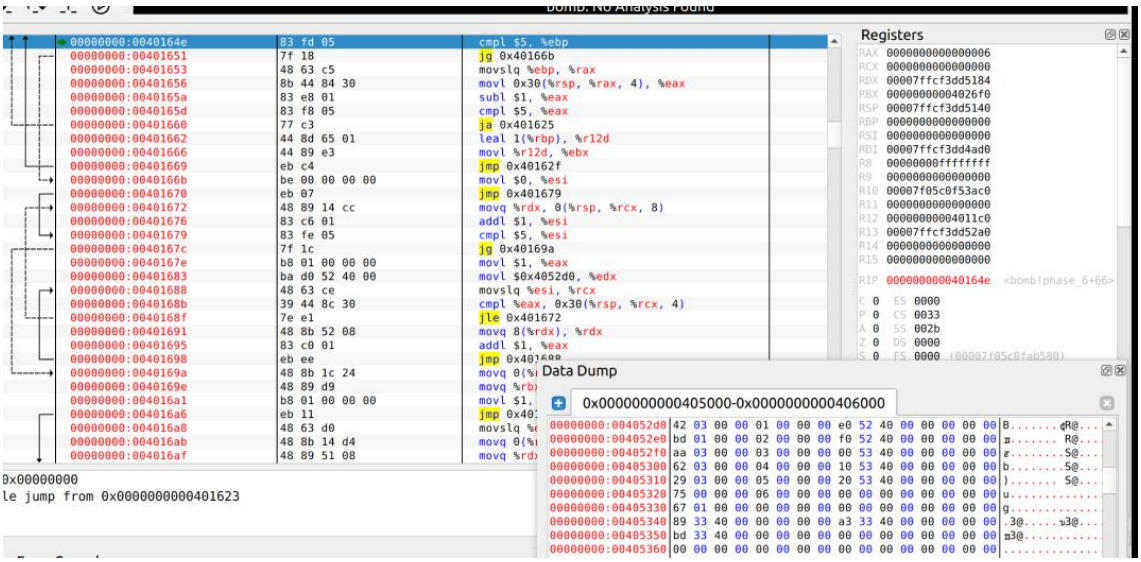
3.6 阶段 6 的破解与分析

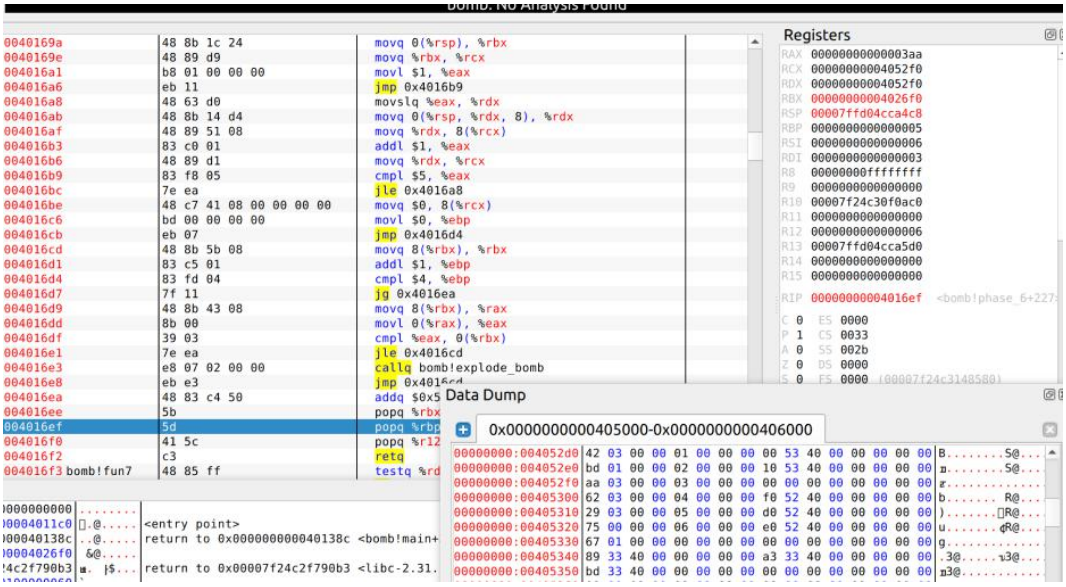
密码如下：6 2 5 1 4 3

破解过程：

观察第六个问题发现需要输入 6 个整数进入函数之后发现函数中需要调用内存中的数据。前往该部分内存，发现一段数据结构为{数据；数据；地址}，容易想到这是一个共有六个节点的链表。结合输入六个数，和其本身的序号 1-6 进行合理猜测，可以认为是一个链表排序，结合程序结尾处的“eax 与 rbx 中的数据进行比较，当小于等于时跳转”可以知道这是一个从小到大对链表进行排序判断的程序。

输入 6 2 5 1 4 3 进行验证，验证成功。


Registers
Data Dump
0x0000000000405000-0x0000000000406000


Registers
Data Dump
0x0000000000405000-0x0000000000406000

第 4 章 总结

4.1 请总结本次实验的收获

本次实验首先熟悉了 codeblocks 下的反汇编操作，并且在虚拟机上自主下载并安装了 EDB-debugger, 初步了解了如何通过 EDB 和 GDB 进行 c 可执行文件的反汇编分析操作。

本次实验的核心内容是拆除炸弹—即通过对 bomb 文件的反汇编操作，破解使各函数可以正常运行的密码，成功拆除炸弹。在拆除过程中，得益于逐步递增的破解难度，我首先熟悉了 EDB 的操作界面，很轻易的完成了第一个密码的破解：输入字符串。紧接着，对于函数输入六个整数的密码，我在破解中逐渐可以理解循环操作在汇编语言中的表现形式；对于递归调用函数，我学会了观察传参时如何判断参数在整个数组中的位置，如 a[n],a[n-1]等等，并且尝试建立起递归函数的抽象表达式（见 func4）；在接下来的问题中，对于比较难的汇编程序，我发现理解整个函数功能有些困难，但是我从中抓住了链表的关键信息，并根据合理猜测得出了正确密码。

4.2 请给出对本次实验内容的建议

进行反汇编操作过程中，由于经验不足以及对汇编语言熟悉程度不够，解读代码过程中我感觉到了吃力和晦涩，不能很好的和曾经所学习的 C 语言做出联系。希望作业完成后可以给出源 C 程序作为参照，也可以进一步加深对编程语言和汇编语言之间的理解。

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.