

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**专业班级： ACM1901班**

**学 号： U201915035**

**姓 名： 邹雅**

**指导教师： 刘海坤**

**报告日期： 2021年 6月 8日**

**计算机科学与技术学院**

**目录**

[实验2： 2](#_Toc73983883)

[实验3： 4](#_Toc73983884)

[实验总结 5](#_Toc73983885)

# 

# 实验2： Binary Bombs

**2.1 实验概述**

本实验中，你要使用课程所学知识拆除一个“Binary Bombs”来增强对程序的机器级表示、汇编语言、调试器和逆向工程等方面原理与技能的掌握。

为了完成二进制炸弹拆除任务,你需要：

1 使用gdb调试器和objdump来反汇编炸弹的可执行文件;

2 单步跟踪调试每一阶段的机器代码；

3 理解每一汇编语言代码的行为或作用；

4 进而设法“推断”出拆除炸弹所需的目标字符串。

5 这可能需要你在每一阶段的开始代码前和引爆炸弹的函数前设置断点,以便于调试。

**2.2 实验内容**

* 一个“Binary Bombs”（二进制炸弹，简称炸弹）是一个Linux可执行C程序，包含phase1~phase6共6个阶段。
* 炸弹运行的每个阶段要求你输入一个特定的字符串，若你的输入符合程序预期的输入，该阶段的炸弹就被“拆除”，否则炸弹“爆炸”并打印输出 "BOOM!!!"字样。
* 实验的目标是你要拆除尽可能多的炸弹阶段。

**2.2.1 阶段1 拆除<phase\_1>**

1.任务描述：

第一阶段主要检测反汇编中的基础部分，通过对于字符串比较过程的解析来拆除炸弹的第一阶段。

2.实验设计：

利用objdump对bomb进行反汇编并将汇编代码输出到asm.txt中。vim查看汇编源代码。利用gdb来查看对应地址存放的内容。

3.实验过程：

观察phase\_1的汇编源代码

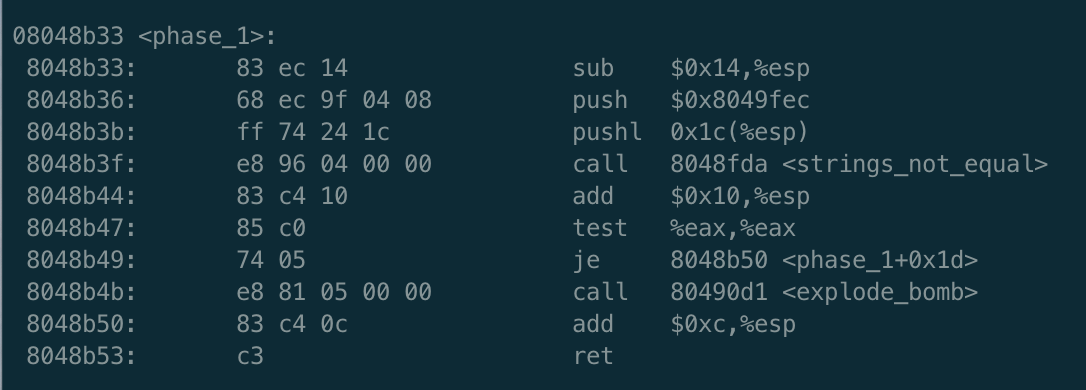


图2.2.1.1 phase\_1汇编源代码

可以看到在调用“strings\_not\_equal”之前push了一个地址0x8049fec，可见该字符串保存地址就是该地址。利用gdb查看该地址内容：

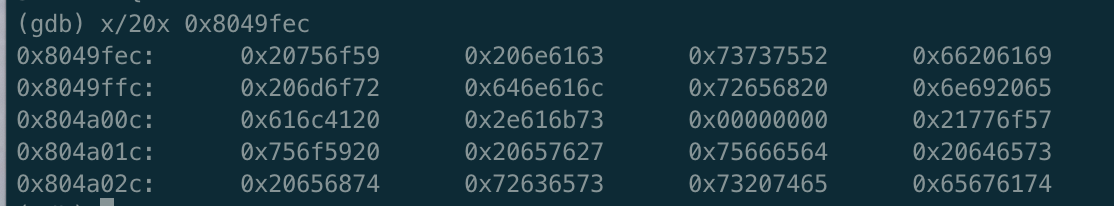


图2.2.1.2 0x8049fec地址内容

解析字符ASCII码值，得到结果为：You can Russia from land here in Alaska.

输入测试可见：

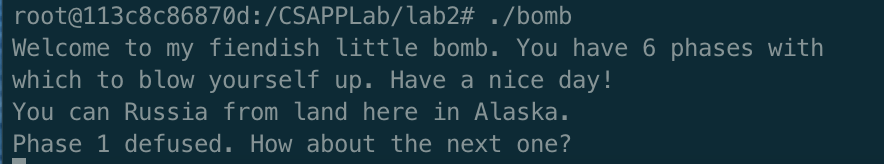


图2.2.1.3 phase\_1测试结果

4.实验结果：phase\_1的解为字符串：You can Russia from land here in Alaska.

**2.2.2 阶段2 拆除<phase\_2>**

1.任务描述：

phase\_2涉及汇编源程序的循环。

2.实验设计：

通过源代码观察分析源程序的循环思路、功能，进而推断结果。

3.实验过程：

观察phase\_2的汇编源代码（已添加必要英文注释）

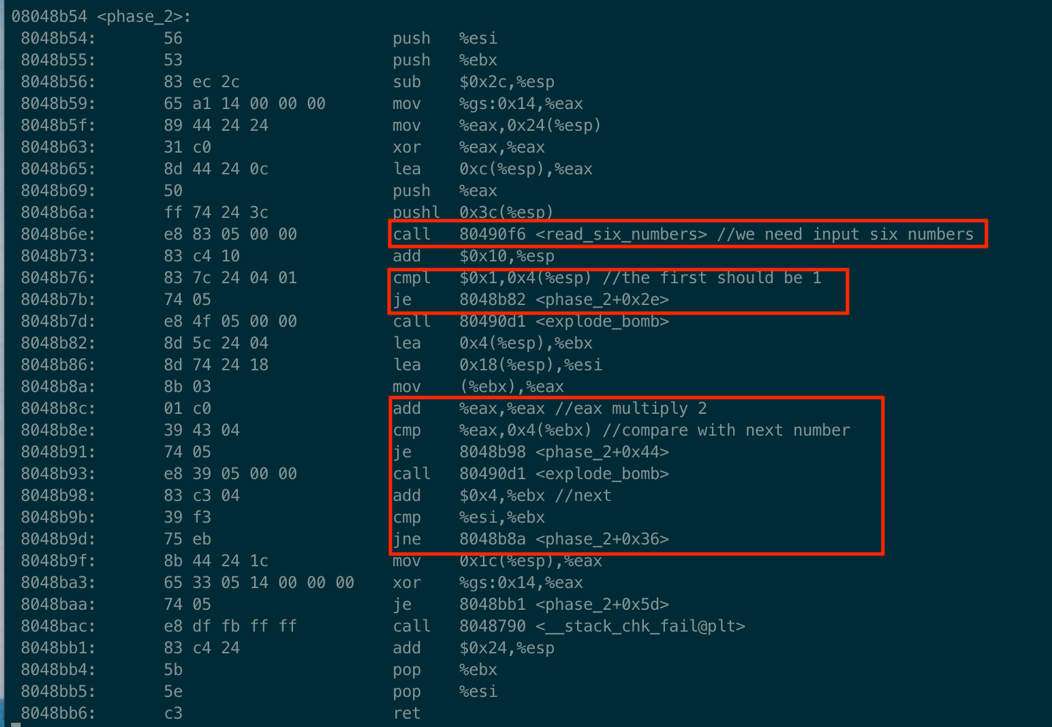


图2.2.2.1 phase\_2源代码

观察可得调用了“read\_six\_numbers”函数，可以知道本阶段的密码是六个数字。且把保存在堆栈中的第一个数字和0x1比较，如果不相等就爆炸，所以第一个是1.

在后续比较中，因为语句“add %eax,%eax”说明把第一个数变为两倍再做后续的比较，如此循环直到读完全部数字结束。由此可以得出这六个数字应该是：1 2 4 8 16 32.

输入测试可见：

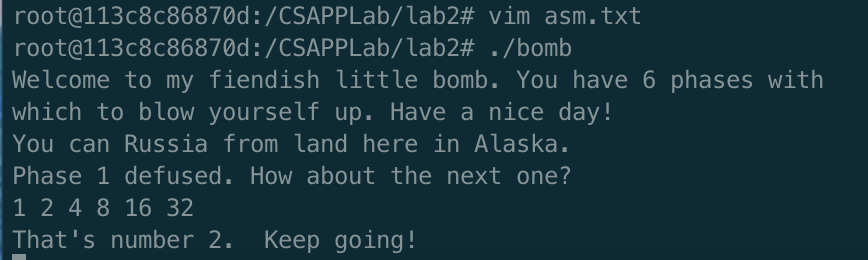


图2.2.2.2 phase\_2测试结果

4.实验结果：

phase\_2的解为1 2 4 8 16 32.

**2.2.3 阶段3 拆除<phase\_3>**

1.任务描述：

phase\_3涉及含switch语句的条件/分支语法。

2.实验设计：

通过源代码观察分析源程序的思路、功能，进而推断结果。利用gdb来查看对应地址存放的内容。

3.实验过程：

观察phase\_3的汇编源代码，截取关键语句进行分析说明。

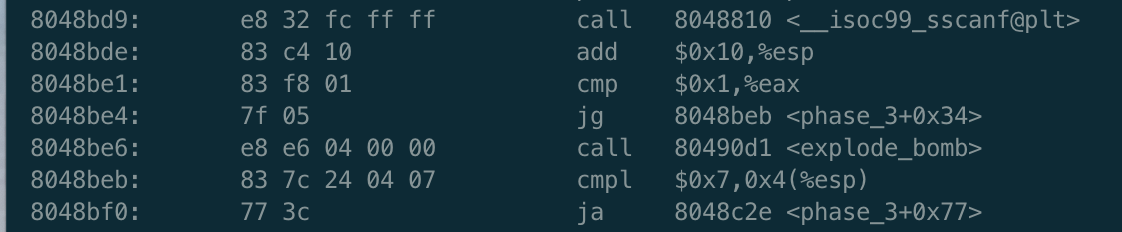


图2.2.3.1 phase\_3源代码1

比较scanf的输入结果，通过跳转指令的对应语句分析输入的参数数量应该大于1. ja跳转到的语句是调用“explode\_bomb”函数，所以推断出第一个参数应该要小于等于7.

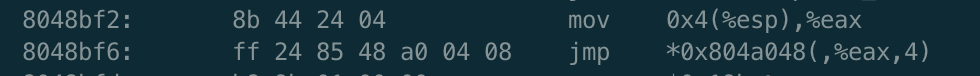


图2.2.3.2 phase\_3源代码2

这两句指令的意思是，根据输入的第一个参数来寻找地址0x804a048中保存的内容，并跳转到该内容对应的指令。通过gdb查看0x804a048的内容。

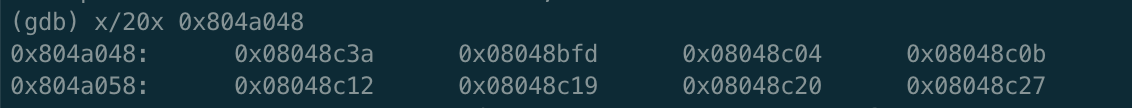


图2.2.3.3 0x804a048的内容

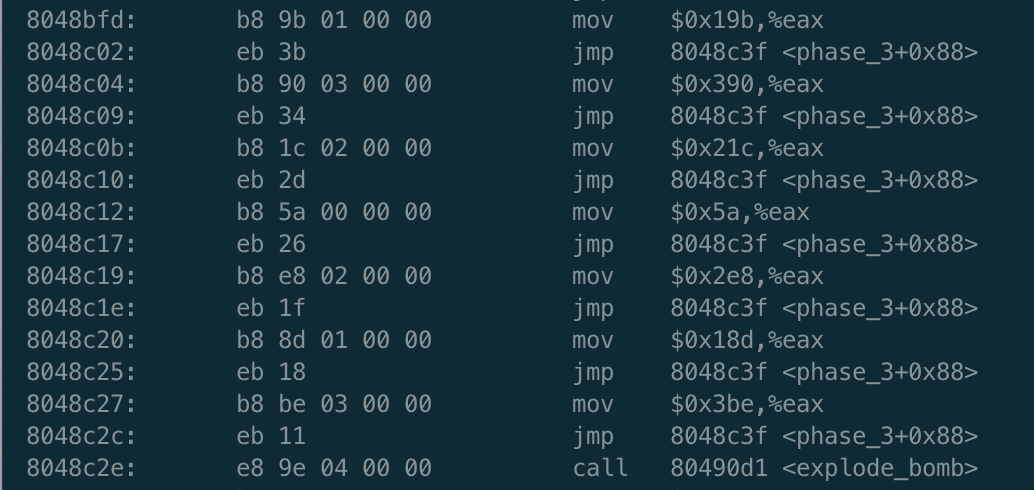


图2.2.3.4 phase\_3源代码3

可见这是一个switch语句。根据第一个参数来选择另一个参数的值。以第一个参数为1来分析，其他类似。当第一个参数为1时，程序跳转到0x08048bfd，eax等于0x19b.

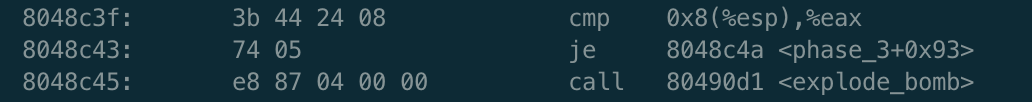


图2.2.3.5 phase\_3源代码4

上述跳转到0x8048c3f后是把第二个参数的值和eax中保存的值比较，不相等则爆炸。如此我们可以得到一组解为1 411，411为0x19b的十进制对应数值。类似我们可以得到剩余六组解2 912、3 540、4 90、5 744、6 397、7 958.

输入测试可见：

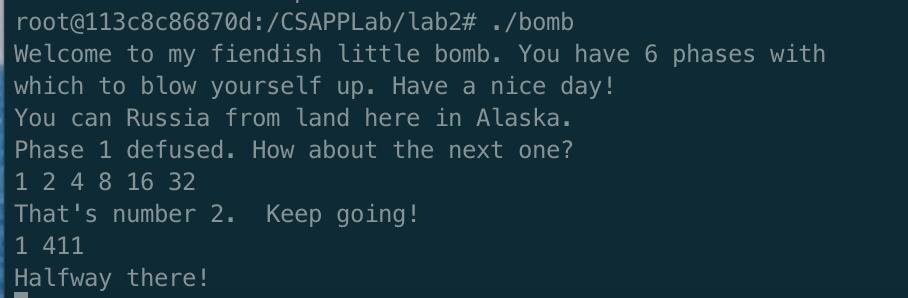


图2.2.3.6 phase\_3测试

4.实验结果：

由上分析可得，phase\_3有七组解，分别为1 411、2 912、3 540、4 90、5 744、6 397、7 958。

**2.2.4 阶段4 拆除<phase\_4>**

1.任务描述：

phase\_4涉及递归调用和栈的内容。

2.实验设计：

通过源代码观察分析源程序的思路、功能，进而推断结果。利用gdb来查看寄存器的值。

3.实验过程：

观察phase\_4的汇编源代码，截取关键语句进行分析说明。

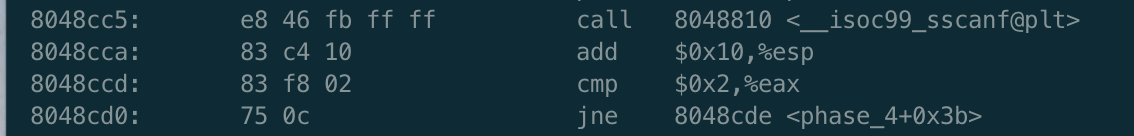


图2.2.4.1 phase\_4源代码1

jne的跳转语句为调用“explode\_bomb”函数，所以可以知道输入的一定是两个参数。

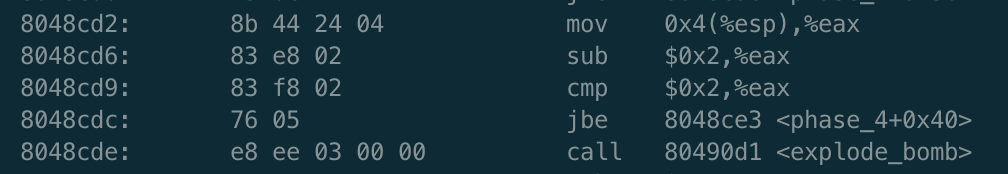


图2.2.4.2 phase\_4源代码2

输入的第二个参数先减2，再和2比较，如果小于等于2则满足条件，否则爆炸。根据数值的机器码表示，可以推出第二个参数应该是一个大于等于2且小于等于4的值，即可能是2、3、4.

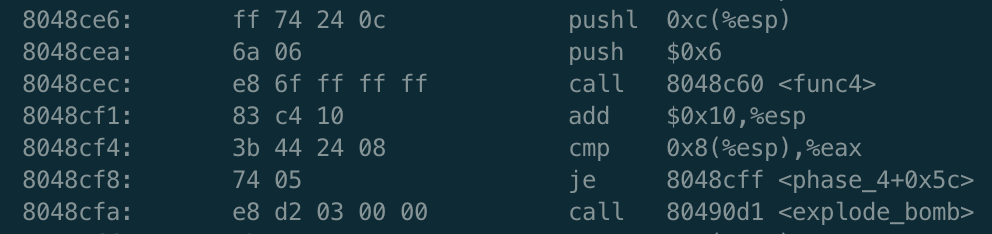


图2.2.4.3 phase\_4源代码3

把输入的第二个参数和6作为形式参数调用func4，调用完之后把返回值和输入的第一个参数比较，如果相等则满足条件，否则爆炸。

利用gdb直接观察寄存器的值来得到func4调用完之后的值。先输入第二个参数为2的情况，观察寄存器。

先在phase\_4函数打上断点，run到该位置时停下来，再利用disassemble指令来获取phase\_4下的汇编源代码来观察打断点的语句位置，在调用func4后的语句打上断点。

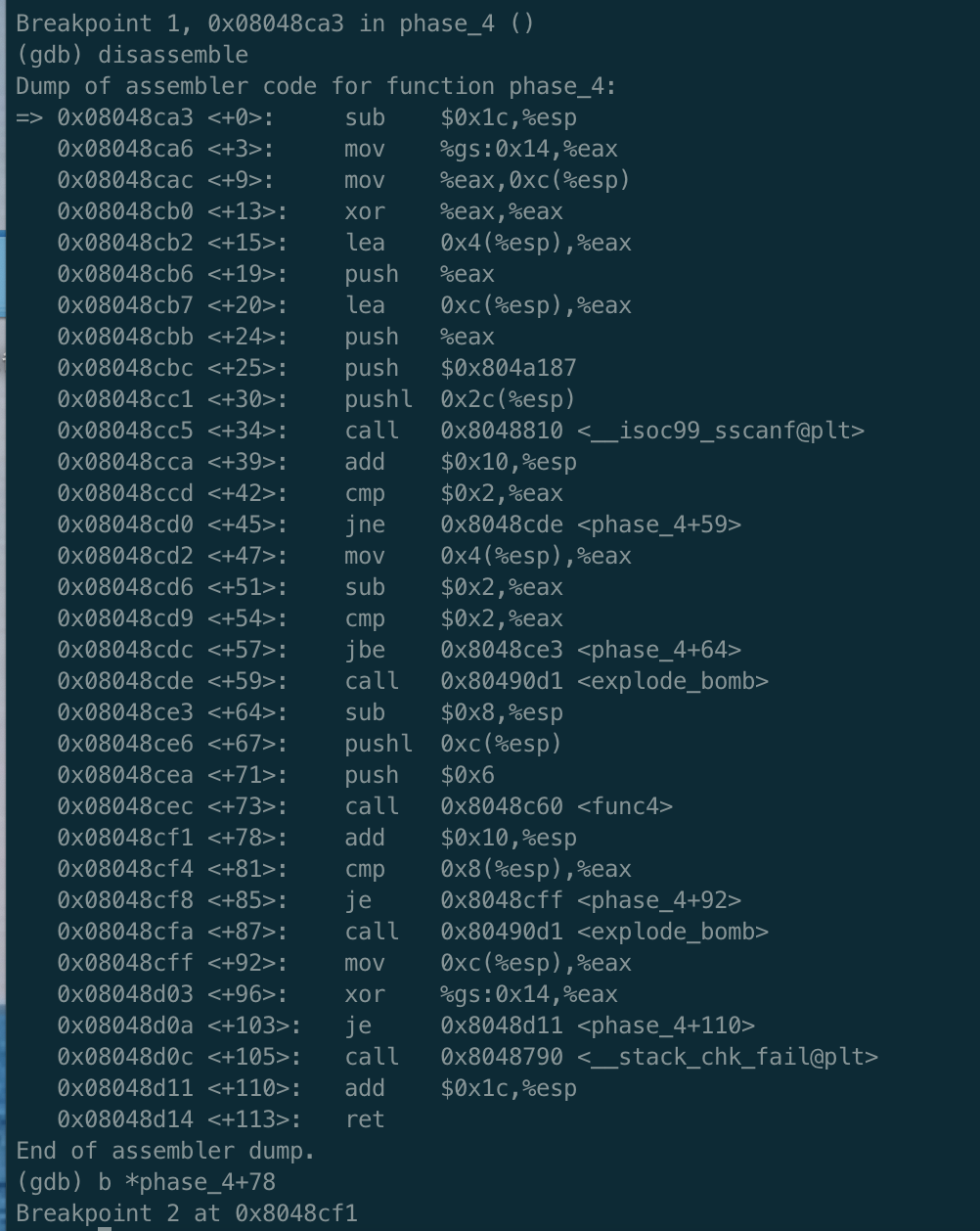


图2.2.4.4 disassemble语句观察并打断点

继续运行到第二个断点并观察寄存器的值。

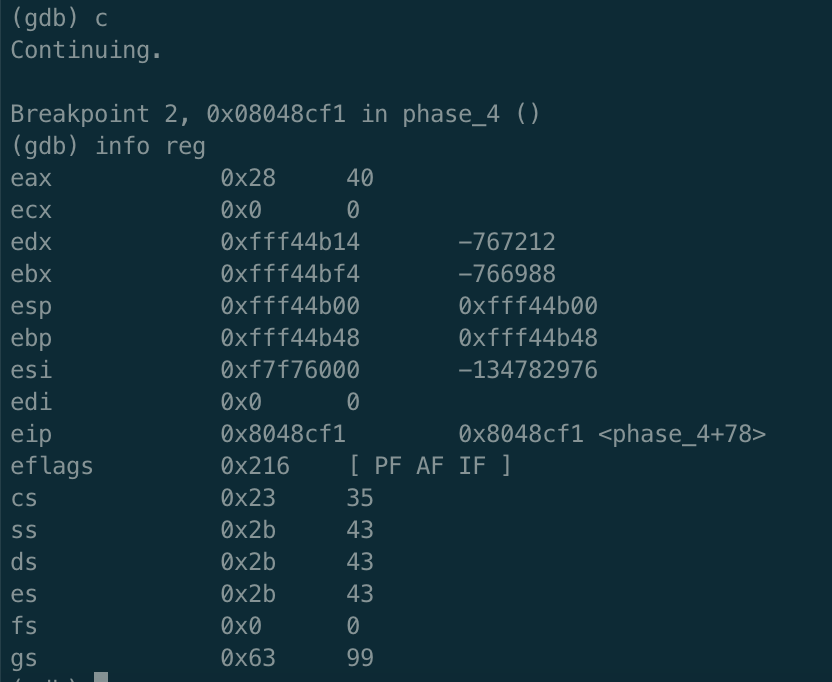


图2.2.4.5 观察寄存器的值

所以第二个参数为2时，第一个参数为40.同理可得另两组解为60 3、80 4.

将结果写入ans.txt文件，并把先前三个阶段的解写入，测试本阶段：

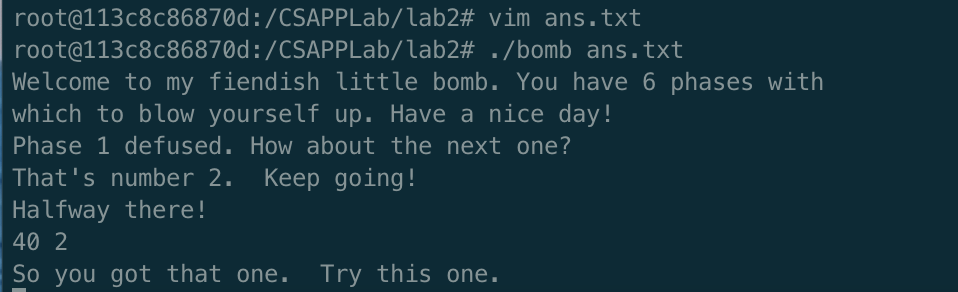


图2.2.4.6 phase\_4测试结果

4.实验结果：

phase\_4有三组解，分别为40 2、60 3、80 4。

**2.2.5 阶段5 拆除<phase\_5>**

1.任务描述：

phase\_5涉及指针的内容。

2.实验设计：

通过源代码观察分析源程序的思路、功能，进而推断结果。利用gdb来查看对应地址存放的内容。

3.实验过程：

观察phase\_5的汇编源代码，截取关键语句进行分析说明。

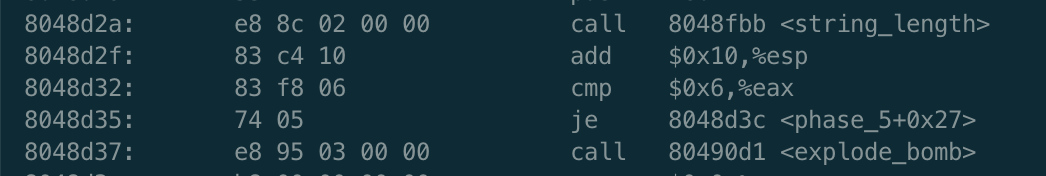


图2.2.5.1 phase\_5源代码1

可得应该输入一个字符串，字符串的长度为6.

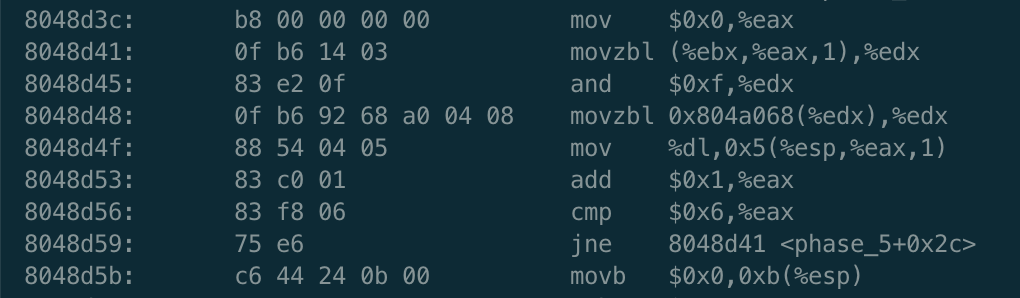


图2.2.5.2 phase\_5源代码2

这是理解整个函数的关键。此时的ebx指向的是输入的字符串在堆栈中的位置，通过eax把每个字符取出来，在“and $0xf,%edx”指令中，截取字符的低4位，以这个为偏移地址在基址0x804a068里取对应字符保存到堆栈中。

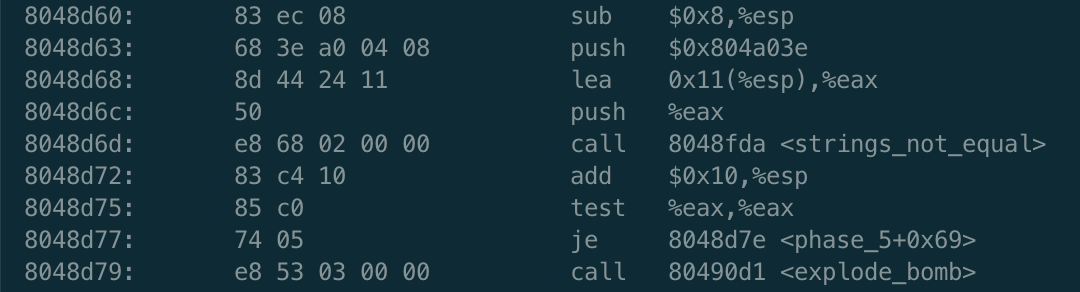


图2.2.5.3 phase\_5源代码3

把上述处理后的字符串地址和0x804a03e压入堆栈，再调用“strings\_not\_equal”，返回值为0时满足结果，否则爆炸，也就是这两个字符串必须要相等。

利用gdb来观察0x804a03e地址中的内容。

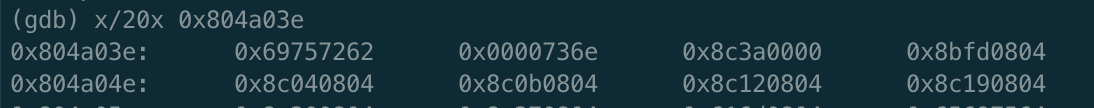


图2.2.5.4 0x804a03e地址中的内容

解析得这六个字符为“bruins”。利用gdb来观察0x804a068地址中的内容。

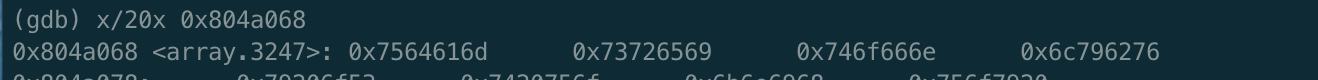


图2.2.5.5 0x804a068地址中的内容

在这块内存中找到对应“bruins”各个字符的位置，得出相对于0x804a068的偏移地址为13、6、3、4、8、7，这是输入字符串ASCII码值的低四位值，转化为ASCII码后其中一个解为mfcdhg（0x6d、0x66、0x63、0x64、0x68、0x67）。

输入测试结果：

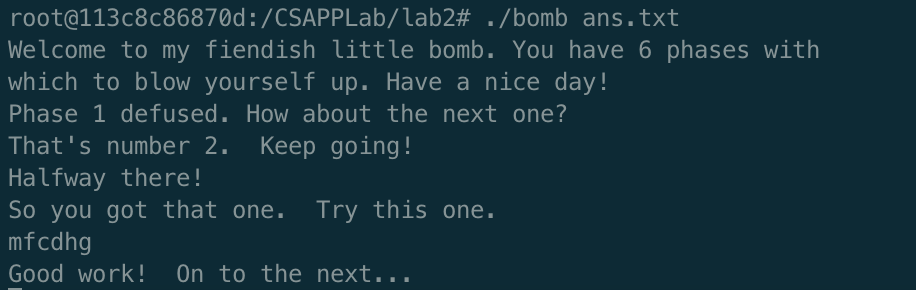


图2.2.5.6 phase\_5测试结果

4.实验结果：

phase\_5的解为mfcdhg。

**2.2.6 阶段6 拆除<phase\_6>**

1.任务描述：

phase\_6涉及链表/指针/结构的内容。

2.实验设计：

通过源代码观察分析源程序的思路、功能，进而推断结果。利用gdb来查看对应地址存放的内容。

3.实验过程：

观察phase\_6的汇编源代码，截取关键语句进行分析说明。

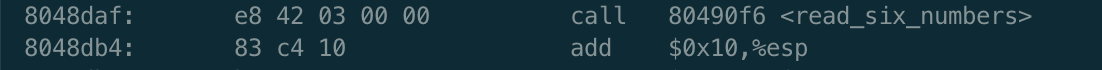


图2.2.6.1 phase\_6源代码1

由此可以得出输入为六个数字。

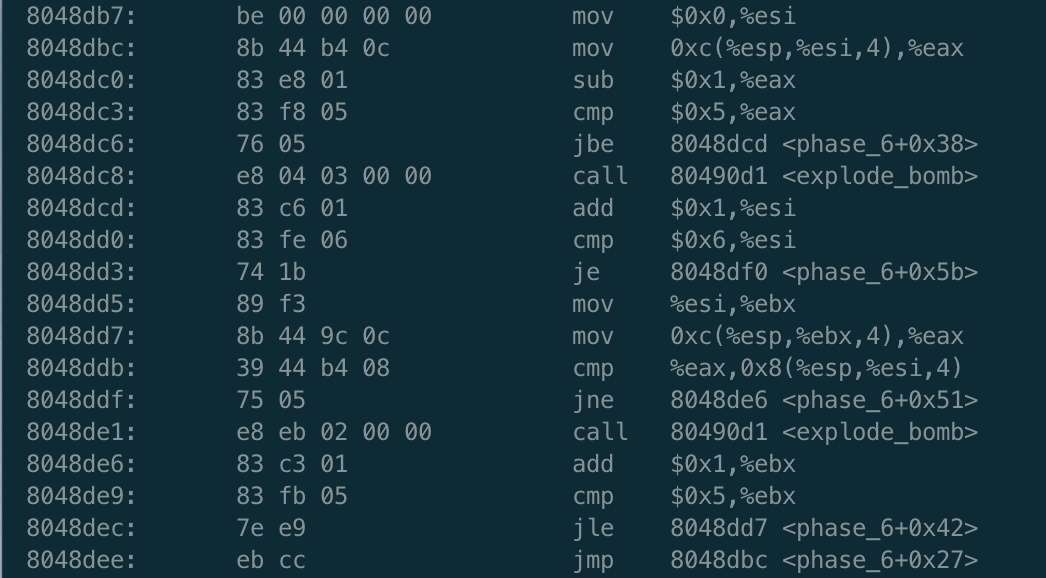


图2.2.6.2 phase\_6源代码2

循环判断。第一重输入的六个值都必须大于等于1且小于等于6，第二重循环判断六个值是否两两之间相等，如果相等则爆炸。所以可以推断这六个数是1、2、3、4、5、6的排序。

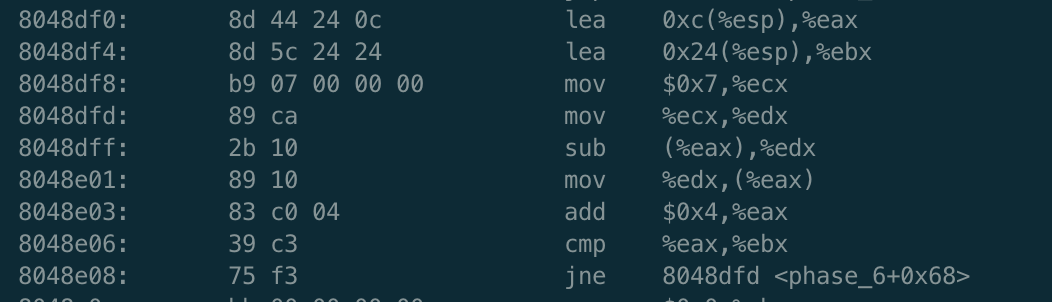


图2.2.6.3 phase\_6源代码3

循环读取每一个数，计算用7减去读入的数并保存至堆栈中，为后续表述方便称之为第二组数据。

后续一部分涉及到一个地址0x804c13c，我们先去查看这个地址中的内容。

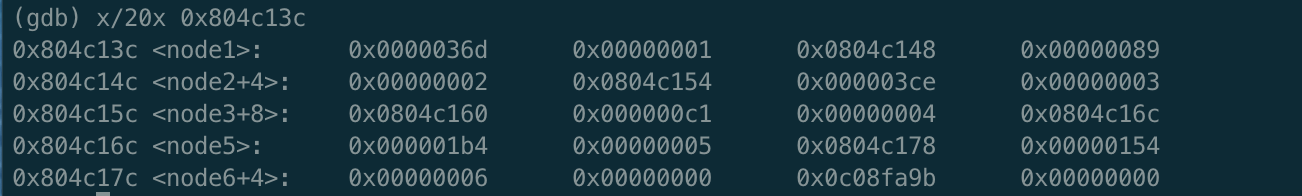


图2.2.6.4 0x804c13c地址中的内容

这是名为node的struct构成的链表内容。每个节点占12个字节，前四个字节保存了一个数值，第二个四个字节保存了索引顺序，第三个四个字节保存了next指针的值。

阅读剩下的汇编源代码（代码较长在此省略截图），可得其语意为把链表按照第一个数值进行排序，排序后的索引值与上述处理过的第二组数据进行比较，如果相等则满足条件，否则爆炸。

将0x36d、0x089、0x3ce、0x0c1、0x1b4、0x154进行排序，得到排序后的索引值为3 1 5 6 4 2。由此可以得到输入的数据应为4 6 2 1 3 5。

输入测试结果：

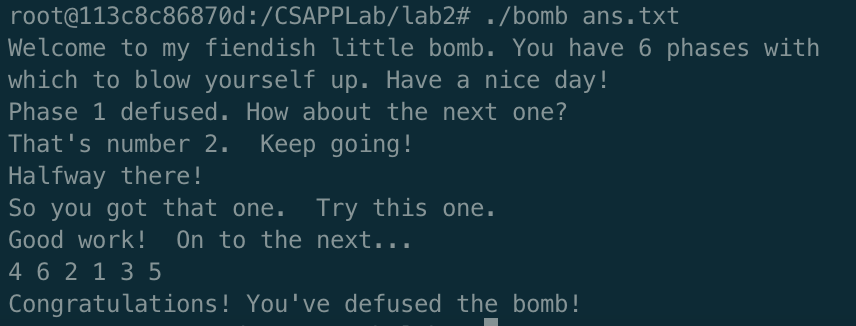


图2.2.6.5 phase\_6测试结果

4.实验结果：

phase\_6的解为4 6 2 1 3 5。

**2.2.7 阶段7 拆除<secret\_phase>**

1.任务描述：隐藏关卡

2.实验设计：

通过源代码观察分析源程序的思路、功能，进而推断结果。利用gdb来查看对应地址存放的内容。

3.实验过程：

在phase\_6下看到两个函数fun7和secret\_phase，说明存在隐藏关卡，检索secret\_phase发现其入口在phase\_defused中。

在phase\_defused有如下语句：

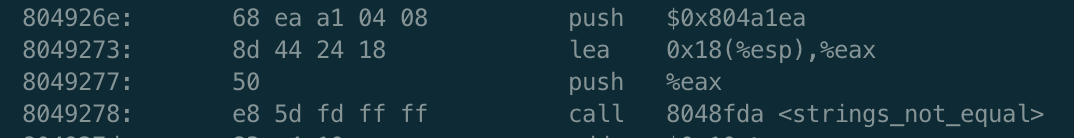


图2.2.7.1 phase\_defused源代码

可以看到涉及一个地址0x804a1ea,访问该地址的内容。

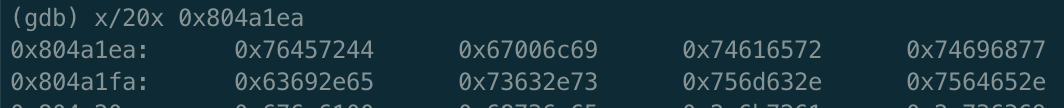


图2.2.7.2 0x804a1ea内容

解析该字符串为“DrEvil”,说明这是进入secret\_phase的字符串，根据提示将该字符串加在phase\_4的解后面，即可进入secret\_phase。

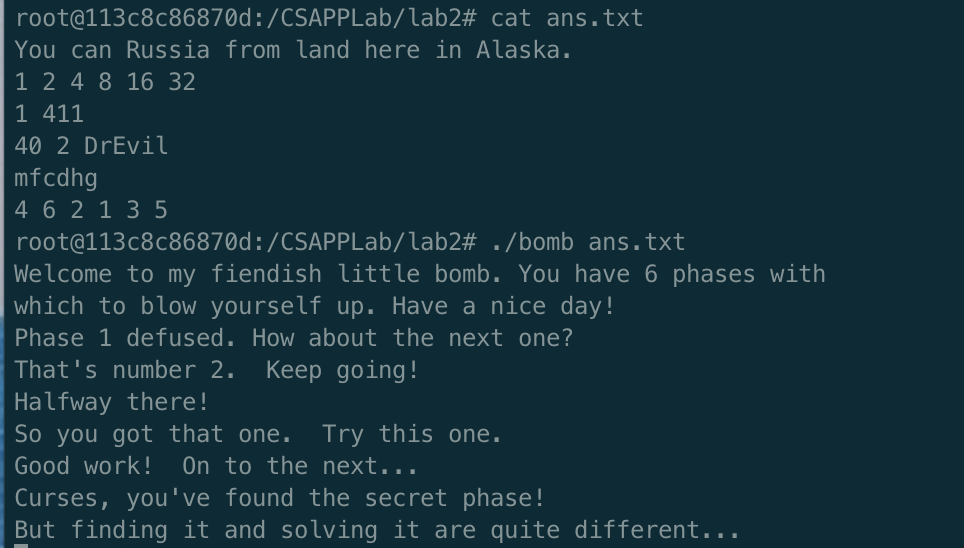


图2.2.7.3 进入隐藏关卡

观察secret\_phase的代码并分析。secret\_phase首先读入一行字符串，然后转化为数字保存。接着以这个数字为参数和另一个地址0x804c088调用fun7，并要求fun7的返回值为5.

查看0x804c088地址中的内容。

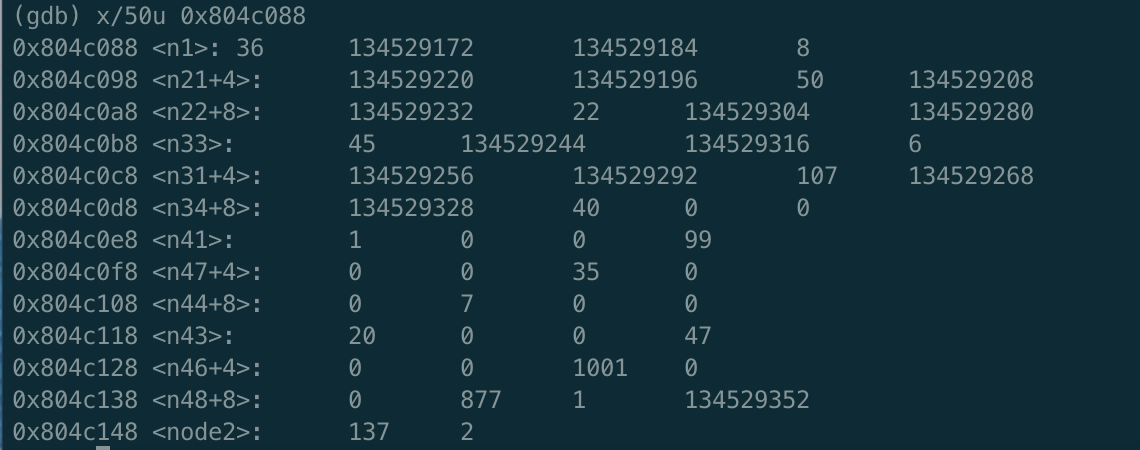


图2.2.7.4 0x804c088内容

这是一个二叉树结构，如下图所示：

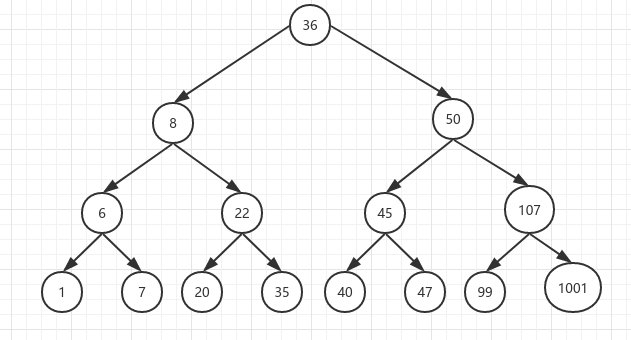


图2.2.7.5 二叉树内容

接着观察fun7的源代码.

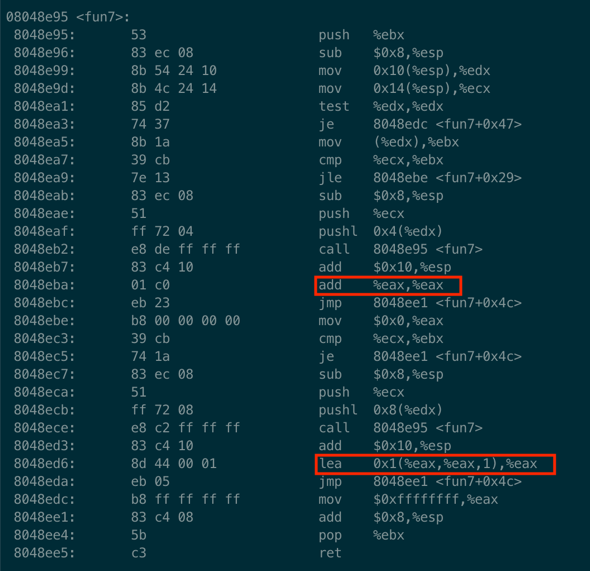


图2.2.7.6 fun7源代码

发现fun7的内容为对输入的数字在二叉树中进行搜索，如果搜索边为左子树的边则将返回值eax的值乘以2（对应“add %eax,%eax”语句），如果搜索边为右子树，则将返回值乘以2加1（对应“lea 0x1(%eax,%eax,1),%eax”语句）。

而我们所要得到的结果为5，那么（（0\*2+1）\*2）\*2+1=5，所有先到右子树再到左子树再到右子树，答案为47.

输入测试：

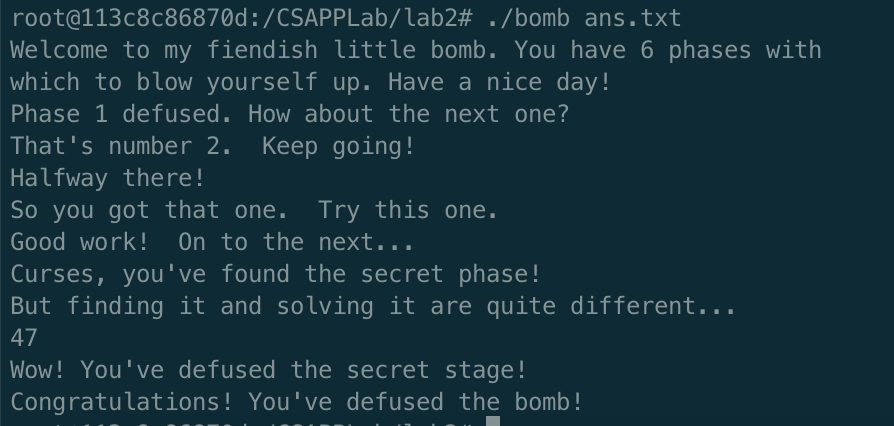


图2.2.7.7 secret\_phase测试

4.实验结果：

secret\_phase的解为47。

**1.3 实验小结**

通过本次实验，我对汇编语言有了更深程度的掌握，对C语言如何转换成汇编语言也有了更加深入的了解。特别是第四阶段的递归、第六阶段的链表和排序、隐藏阶段的二叉树，这些在机器级的表述让我学到了很多，对它们的工作方式也有了更深刻的了解。

本次实验中用到了objdump和gdb两个工具。其中objdump用于反汇编输出bomb程序的汇编源代码。整个实验做完我对gdb中的命令语句掌握了更多：

1. gdb bomb ——gdb运行某个程序
2. b +地址值 ——在某个地址处打断点
3. r ——run
4. ni ——单步执行机器语句
5. c ——continue
6. x/20x +地址值 ——查看某一个地址处的内存内容
7. info registers ——查看当前寄存器的值
8. disassemble ——查看当前所属地址下接着的汇编语句
9. q ——退出

# 实验3：

**3.1 实验概述**

介绍本次实验的目的意义、目标、要求及安排等

**3.2 实验内容**

介绍本次实验的总体主要内容

**3.2.1 阶段1 XXXXXXX**

1.任务描述：给出阶段1的任务描述

2.实验设计：给出解题思路分析和拟采用的技术和方法等

3.实验过程：详细描述实验的具体过程

4.实验结果：给出阶段1的实验结果和必要的结果分析

……

**3.2.x 阶段x XXXXXXX**

1.任务描述：给出阶段x的任务描述

2.实验设计：给出解题思路分析和拟采用的技术和方法等

3.实验过程：详细描述实验的具体过程

4.实验结果：给出阶段x的实验结果和必要的结果分析

**3.3实验小结**

对本次实验使用的理论、技术、方法和结果进行总结。描述一下通过实验你有哪些收获。

# 实验总结

全面总结实验成果，描述通过实验得到收获（400字以上）