

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**专业班级： CS1409**

**学 号： U201414808**

**姓 名： 王林**

**指导教师： 谭志虎**

**报告日期： 2016年 月 日**

**计算机科学与技术学院**

**目录**

[实验2： 二进制炸弹实验 1](#_Toc450861540)

[2.1 实验概述 1](#_Toc450861541)

[2.1.1实验简介 1](#_Toc450861542)

[2.1.2实验目的 1](#_Toc450861543)

[2.1.3实验环境 1](#_Toc450861544)

[2.2 实验内容 1](#_Toc450861545)

[2.3 实验过程和记录 2](#_Toc450861546)

[2.3.1 阶段1 字符串比较 2](#_Toc450861547)

[2.3.1.1.任务描述： 2](#_Toc450861548)

[2.3.1.2.实验设计： 2](#_Toc450861549)

[2.3.1.3.实验过程： 2](#_Toc450861550)

[2.3.1.4.实验结果： 3](#_Toc450861551)

[2.3.2 阶段2循环 3](#_Toc450861552)

[2.3.2.1.任务描述： 3](#_Toc450861553)

[2.3.2.2.实验设计： 3](#_Toc450861554)

[2.3.2.3.实验过程： 3](#_Toc450861555)

[2.3.2.4.实验结果： 4](#_Toc450861556)

[2.3.3 阶段3 条件分支 switch语句 4](#_Toc450861557)

[2.3.3.1.任务描述： 4](#_Toc450861558)

[2.3.3.2.实验设计： 4](#_Toc450861559)

[2.3.3.3.实验过程： 4](#_Toc450861560)

[2.3.3.4.实验结果： 4](#_Toc450861561)

[2.3.4. 阶段4 递归调用和栈 5](#_Toc450861562)

[2.3.4.1.任务描述： 5](#_Toc450861563)

[2.3.4.2.实验设计： 5](#_Toc450861564)

[2.3.4.3.实验过程： 5](#_Toc450861565)

[2.3.4.4.实验结果： 5](#_Toc450861566)

[2.3.5. 阶段5指针 5](#_Toc450861567)

[2.3.5.1.任务描述： 5](#_Toc450861568)

[2.3.5.2.实验设计： 5](#_Toc450861569)

[2.3.5.3.实验过程： 5](#_Toc450861570)

[2.3.5.4.实验结果： 6](#_Toc450861571)

[2.3.6. 阶段6链表/指针/结构 6](#_Toc450861572)

[2.3.6.1.任务描述： 6](#_Toc450861573)

[2.3.6.2.实验设计： 6](#_Toc450861574)

[2.3.6.3.实验过程： 6](#_Toc450861575)

[2.3.6.4.实验结果： 7](#_Toc450861576)

[2.3.7 阶段7 隐藏关卡 7](#_Toc450861577)

[2.3.7.1.任务描述： 7](#_Toc450861578)

[2.3.7.2.实验设计： 7](#_Toc450861579)

[2.3.7.3.实验过程： 7](#_Toc450861580)

[2.3.7.4.实验结果： 7](#_Toc450861581)

[2.4 实验小结 8](#_Toc450861582)

# 

# 实验2： 二进制炸弹实验

2.1 实验概述

2.1.1实验简介

本次试验中，需要运用所学的知识来拆除一个“binary bombs”, 从而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。

一个“binary bombs”（二进制炸弹，下文将简称为炸弹）是一个Linux可执行C程序，包含了6个阶段（phase1~phase6）。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。实验的目标是拆除尽可能多的炸弹层次。

2.1.2实验目的

通过拆弹实验，可以巩固课上所学的知识，并且增强对计算机系统程序执行原理的理解。可以熟悉程序的机器级表示，以及熟悉汇编语言，熟悉对gdb的使用，了解逆向工程和相关的技能。

2.1.3实验环境

实验语言：C语言

实验环境：linux

2.2 实验内容

每个炸弹阶段考察了机器级语言程序的一个不同方面，难度逐级递增：

\* 阶段1：字符串比较

\* 阶段2：循环

\* 阶段3：条件/分支

\* 阶段4：递归调用和栈

\* 阶段5：指针

\* 阶段6：链表/指针/结构

另外还有一个隐藏阶段，但只有当你在第4阶段的解之后附加一特定字符串后才会出现。

为了完成二进制炸弹拆除任务，需要使用gdb调试器和objdump来反汇编炸弹的可执行文件，并单步跟踪调试每一阶段的机器代码，从中理解每一汇编语言代码的行为或作用，进而设法“推断”出拆除炸弹所需的目标字符串。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。

本次实验中，需要完成拆弹实验。通过查看反汇编代码，以及GDB单步调试，观察内存中的值。

2.3 实验过程和记录

将下载的文件考到Ubuntu虚拟机中，打开终端，进入当前目录，“ls ”命令，查看当前目录文件，发现有可执行文件bomb和源程序文件bomb.c。

直接在终端下运行命令 “objdump –d bomb > asm.asm”，得到对应的反汇编代码，可以发现在当前目录下多了一个asm.asm文件，执行 “vim asm.asm”命令，打开该文件，查看对应的反汇编代码。然后进入拆弹。

安装一个打印gdb调试信息的脚本:.gdb\_init，方便打印出调试时的相关信息，方便调试。

2.3.1 阶段1 字符串比较

2.3.1.1.任务描述：

本阶段是字符串比较，需要输入一个与内存中的某一个字符串进行匹配，匹配成功，则拆除炸弹，否则炸弹爆炸。

2.3.1.2.实验设计：

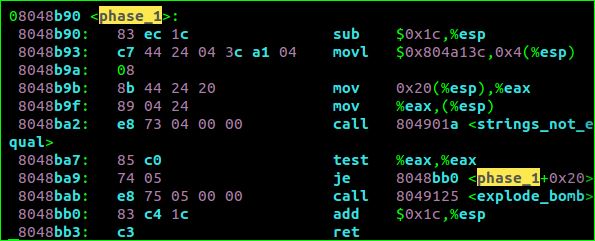
通过阅读反汇编代码的到储存对应字符串的内存地址，因为这个地址会作为一个参数传入字符串匹配函数，比较容易找到。然后再gdb下打印这个内存地址连续区域的信息，得到要输入的字符串。

2.3.1.3.实验过程：

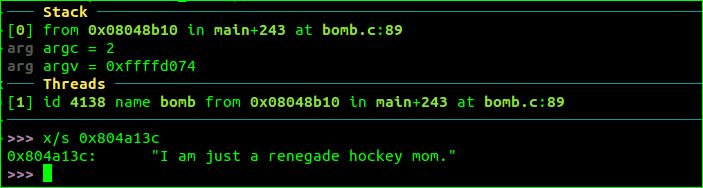
1、找到main函数的代码，找到调用phase\_1的部分

2、找到phase\_1函数代码，可以看到read\_line函数，即读入输入字符串的函数

3、在phase\_1函数中，有一个strings\_not\_equal函数，她有两个参数，其中一个参数是内存地址$0x804a13c。



4、在终端下运行gdb进行单步调试，直接执行“x/s 0x804a13c”，得到打印的字符串为'I am just a renegade hocke mom.'，即要求输入的字符串。



2.3.1.4.实验结果：

在终端下运行./bomb程序，输入字符串'I am just a renegade hocke mom.'，显示成功拆除炸弹，则字符串正确。

2.3.2 阶段2循环

2.3.2.1.任务描述：

本阶段是循环，通过阅读反汇编代码知道循环的次数，要输入6个正确的数，才能拆除炸弹，否则炸弹爆炸。

2.3.2.2.实验设计：

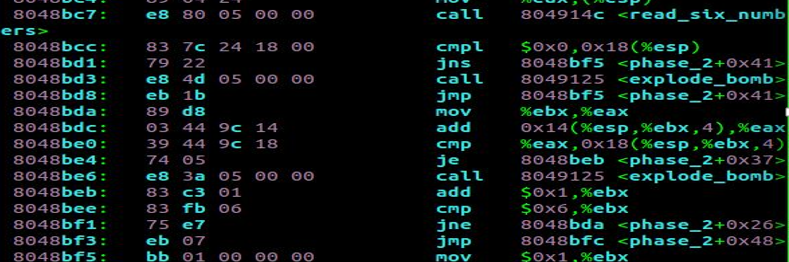
阅读反汇编代码，找到phase\_2函数，找到循环体，弄清楚循环体的内容，通过计算得到正确的值。

2.3.2.3.实验过程：

1、找到main函数的代码，找到调用phase\_2的部分

2、找到phase\_3函数代码，可以看到read\_six\_numbers函数，即读入输入整数的函数，可以发现是读入6个整数。

3、在phase\_2函数中，找到循环体，发现循环内容是前一个数先加上计数器的值，再与后一个数比较，相同继续循环，否则炸弹爆炸。



4、计数器从1到6，则输入的6的数的间隔应该分别是1~6，取第一个数是0，则后面的数分别为1、3、6、10、15。

2.3.2.4.实验结果：

在终端下运行./bomb程序，输入6个整数：0、1、3、6、10、15，显示成功拆除炸弹，则整数正确。

2.3.3 阶段3 条件分支 switch语句

2.3.3.1.任务描述：

本阶段是条件分支，通过阅读反汇编代码找到对应的条件，通过计算找出一种符合条件的输入，即可拆除炸弹。

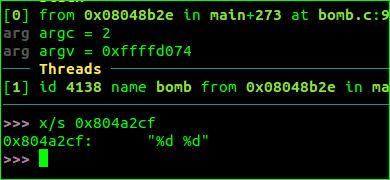
2.3.3.2.实验设计：

阅读反汇编代码，找到phase\_3函数，找到对应的分支结构，计算条件的内容，得到正确的值。

2.3.3.3.实验过程：

1、找到main函数的代码，找到调用phase\_3的部分

2、找到phase\_3函数代码，可以看到scanf函数，读入输入的值，执行“x/s 0x804a2cf”,得到要求输入两个整数。



3、在gdb中单步调试，发现要求输入的第一个整数小于等于7，则把第一个数定为7。

4、 然后打印，执行“x/wx 0x804a1ac”，得到值为0x8048c6e，跳转到对应代码行，计算第二个参数值为770。

注：这个结果只是一种情况。

2.3.3.4.实验结果：

在终端下运行./bomb程序，输入2个整数：7 770，显示成功拆除炸弹，则答案正确。

2.3.4. 阶段4 递归调用和栈

2.3.4.1.任务描述：

本阶段是递归调用和栈，通过阅读反汇编代码找到对应的递归终止条件，通过计算，计算出符合条件的输入，即可拆除炸弹。

2.3.4.2.实验设计：

阅读反汇编代码，找到phase\_4函数，找到对应的递归函数，找到计算终止条件，得到正确的值。

2.3.4.3.实验过程：

1、找到main函数的代码，找到调用phase\_4的部分

2、找到phase\_4函数代码，可以看到scanf函数，读入输入的值，执行“x/s 0x804a2cf”,得到要求输入两个整数。

3、阅读反汇编代码，得到第一个输入必须为1。

4、找到递归函数fanc4,并且fun4函数的操作数个数为3，并且递归最终返回值必须是1。

5、通过阅读反汇编代码，可以发现是二叉排序树算法，则画出对应的二叉树。并且递归调用的返回值必须为1，而递归终止时最后一个返回时0.从右边返回是×2+1操作，左边是×2操作。所以是递归了两次，且第二次是从右边返回。所以输入的第一个参数是11。

2.3.4.4.实验结果：

在终端下运行./bomb程序，输入2个整数：1 11，显示成功拆除炸弹，则答案正确

2.3.5. 阶段5指针

2.3.5.1.任务描述：

本阶段是指针，通过阅读反汇编代码找到对应的处理地址的地方，通过计算，计算出符合条件的输入，即可拆除炸弹。

2.3.5.2.实验设计：

阅读反汇编代码，找到phase\_5函数，找到对应的地址，打印地址和值，得到正确的值。

2.3.5.3.实验过程：

1、找到main函数的代码，找到调用phase\_5的部分

2、找到phase\_5函数代码，看到有一个string\_length函数和string\_not\_equal函数，可以推测要求我们输入一个确定长度的字符串。并且观察string\_length函数的返回值可以确定长度是6。

3、阅读反汇编代码可以看到一个循环，是依次取出输入字符的低四位并且和内存的某些字符运算，得到新的字符串，存到堆栈区。

4、然后调用字符串比较函数，另一个参数是内存地址0x804a186,打印出来是字符串“bruins”。然后反推每次从输入的字符串中取出的值，从而找到应该输入的字符串。

2.3.5.4.实验结果：

在终端下运行./bomb程序，输入字符串：‘mfcdhw’，显示成功拆除炸弹，则答案正确

2.3.6. 阶段6链表/指针/结构

2.3.6.1.任务描述：

本阶段是指针和链表，通过阅读反汇编代码找到对应的链表结构，通过打印出地址和对应的值，计算出符合条件的输入，即可拆除炸弹。

2.3.6.2.实验设计：

阅读反汇编代码，找到phase\_6函数，找到对应的内存地址，打印所有的地址，得到正确的值。

2.3.6.3.实验过程：

1、找到main函数的代码，找到调用phase\_6的部分

2、找到phase\_6函数代码，可以发现有一个read\_six\_numbers函数，知道要输入6个数。

3、阅读反汇编代码，在输入之后有一串循环，可以读出是要求输入的6个数互不相等，且范围是1~6。

4、后面又是一串循环，可以发现循环中执行的是交换地址，即交换指针，而且是按照前面输入的顺序改变对应的地址。通过循环打印出对应6个地址中的值分别是：

1、0x0804c13c:0x000002f4

2、0x0804c148:0x0000006a

3、0x0804c154:0x0000038d

4、oxo8o4c160:0x00000200

5、0x0804c16c:0x000000b6

6、0x0804c178:0x0000008e

5、阅读源码，知道最后满足的条件是从大到小，则对应的输入应该是：3 1 4 5 6 2

2.3.6.4.实验结果：

在终端下运行./bomb程序，输入6个整数：3 1 4 5 6 2，显示成功拆除炸弹，则答案正确

2.3.7 阶段7 隐藏关卡

2.3.7.1.任务描述：

本阶段是隐藏关卡，只有咋的4个炸弹后输入正确的字符串，才能进入隐藏关卡，在隐藏关卡输入正确的解，才能解开炸弹。

2.3.7.2.实验设计：

阅读反汇编代码，找到secret\_phase函数，找到对应的调用函数，找到进入隐藏关卡的条件。然后到secret\_phase函数中，单步调试，找出正确的解。

2.3.7.3.实验过程：

1、找到main函数的代码，找到调用secret\_phase函数的部分

2、找到secret\_phase函数的调用者是是phase\_defused。可以分析是在phase\_defused函数中进入隐藏关卡。

3、在phase\_defused中，有一个strings\_not\_equal函数，一个参数是读入的字符串，另一个参数是在内存0x0804a332中，打印出来是“DrEvil”，可知是输入 “DrEvil” 来进入隐藏关卡。

4、进入secret\_phase函数，gdb单步调试，有输入。而且对输入有限制，发现函数strtol@plt。它的返回值是整型。通过尝试，发现输入字符串返回值就是0，就会引爆炸弹，输入整数就不会,所以应该输入整数，并且要小于1000。

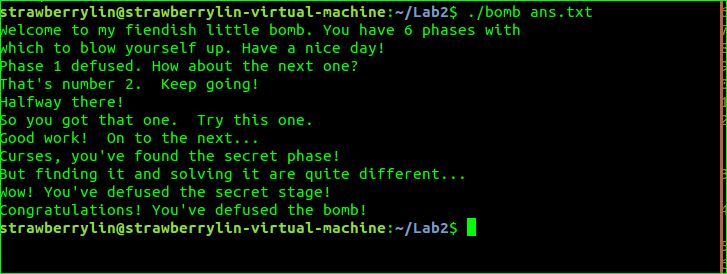
5、阅读源码，可以发现是递归调用，终止的条件是输入的值与内存中的某个值相等。可以发现是树的结构，返回值是0从左边返回是\*2操作，右边是\*2+1操作。

6、在secret\_phase函数中，要求最终返回值是3，则要3次递归，并且后两次返回从右边返回，打印出对应的内存值 “x/d 0x804c0d0”，值为 107.

2.3.7.4.实验结果：

在终端下运行./bomb程序，输入107，显示成功拆除炸弹，则答案正确

测试结果截图：



2.4 实验小结

通过本次拆弹实验，首先巩固课上所学的知识，并且增强对计算机系统程序执行原理的理解。而且熟悉程序的机器级表示，以及汇编语言，学习到了逆向工程和相关的技能。

而且学习了部分有关gdb的使用，熟悉了Linux编程环境，了解了一些基本的命令以及用法，掌握了一些基本技能。

通过阅读反汇编代码，感受到了汇编语言和C语言的区别，而且了解了有关函数栈帧的设计，了解了函数调用过程中的参数传递以及栈帧变化情况。