**课 程 实 验 报 告**

**课程名称： 计算机组成原理**

**专业班级： 计算机Ⅱ类2103班**

**学 号： 8008121077**

**姓 名： 陈映融**

**指导教师： 胡勇**

**报告日期： 2022/12/06**

# 实验2： 拆弹实验

## 2.1 实验概述

实验目的：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。

实验目标：需要拆除尽可能多的炸弹。

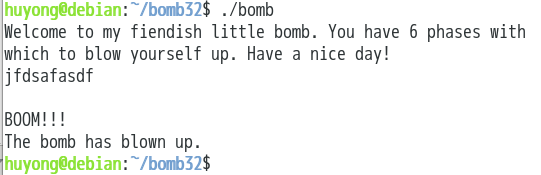
实验要求：使用gdb调试器和objdump来反汇编炸弹的可执行文件，并单步跟踪调试每一阶段的机器代码，从中理解每一汇编语言代码的行为或作用，进而设法“推断”出拆除炸弹所需的目标字符串。

实验语言：c。

实验环境：linux

## 2.2 实验内容

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。



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

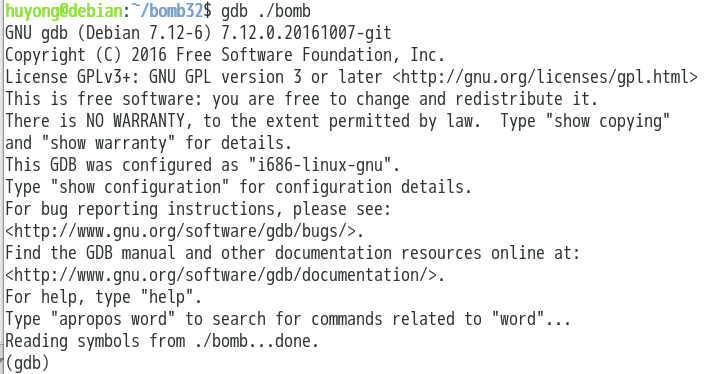
\* 阶段5：指针

\* 阶段6：链表/指针/结构

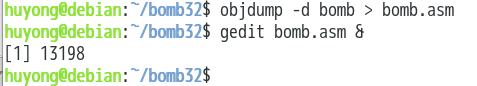
另外还有一个隐藏阶段，但只有当你在第4阶段的解之后附加一特定字符串后才会出现。

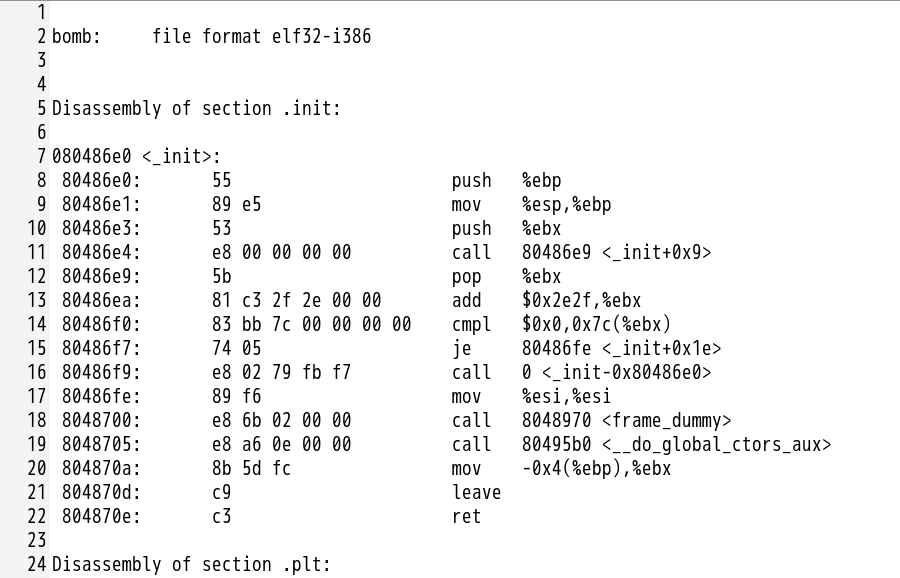
为了完成二进制炸弹拆除任务，你需要使用gdb调试器和objdump来反汇编炸弹的可执行文件，并单步跟踪调试每一阶段的机器代码，从中理解每一汇编语言代码的行为或作用，进而设法“推断”出拆除炸弹所需的目标字符串。这可能需要你在每一阶段的开始代码前和引爆炸弹的函数前设置断点，以便于调试。

通过gdb调试bomb程序：



通过执行以下指令，可以反汇编bomb，并将结果保存到bomb.asm文件中，然后通过gedit或more可以方便地察看汇编代码：





**2.2.1 阶段1 字符串比较**

1. 任务描述：通过phase\_1的反汇编代码找出要输入的字符串。
2. 实验设计：利用gdb结合断点来动态地分析。
3. 实验过程：

观察phase\_1的反汇编代码，如图2.1.1所示：

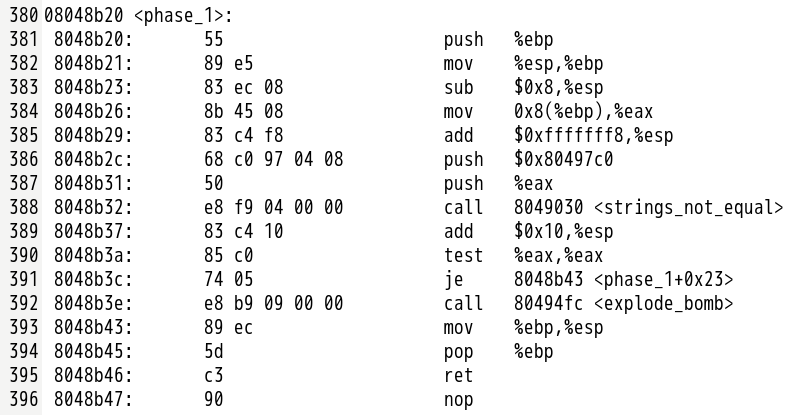


图2.1.1

通过分析，发现在调用strings\_not\_equal对比字符串之前，有两个压栈的指令，其中一个是将函数入参送入栈中，还有一个地址送入了栈，函数入参应该就是输入的字符串所在地址，猜测另一个地址就是正确字符串的首址，于是在phase\_1处下个断点，然后运行，随意输入一个字符，触发断点，再用x命令查看字符串。如图2.1.2所示：

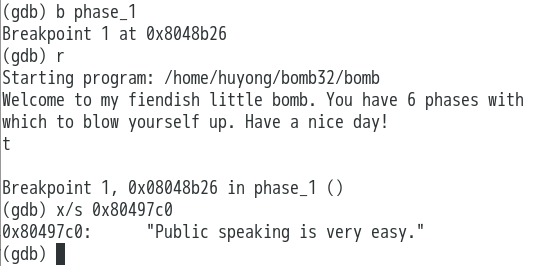


图2.1.2

故猜测“Public speaking is very easy.”就是所需字符串。重新执行该程序，直接输入该字符串，观察结果。如图2.1.3所示：

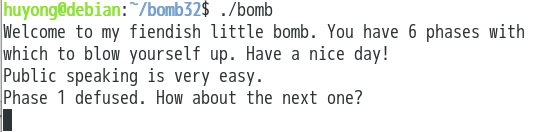


图2.1.3

1. 实验结果：如图2.1.3所示，阶段一拆弹成功！

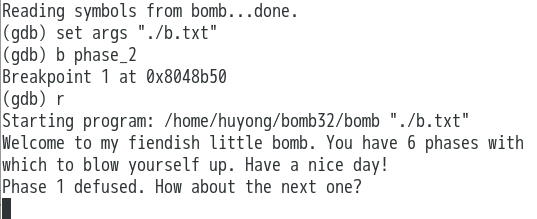
阶段一很简单，找出字符串首地址就可以了，没有什么大的难点。

为了提高调试效率，可以将解析的结果保存到文件中，在调试时以参数方式提供给程序运行，方法如下：

第一步：建立一个参数文件，比如“b.txt”文件，然后用vim或gedit编辑该文件，写入“Public speaking is very easy.”：



然后在gdb中调用该参数文件：



**2.2.2 阶段2 循环**

任务描述：通过phase\_2的反汇编代码推断第二阶段要输入的数据

**2.2.3 阶段3 条件/分支**

任务描述：通过phase\_3的反汇编代码推断第三阶段要输入的数据

**2.2.4 阶段4 递归调用和栈**

任务描述：通过phase\_4以及func4的反汇编代码推断第四阶段要输入的数据

**2.2.5 阶段5 指针**

任务描述：通过phase\_5的反汇编代码推断第五阶段要输入的数据

**2.2.6 阶段6 链表/指针/结构**

任务描述：通过phase\_6的反汇编代码推断第五阶段要输入的数据

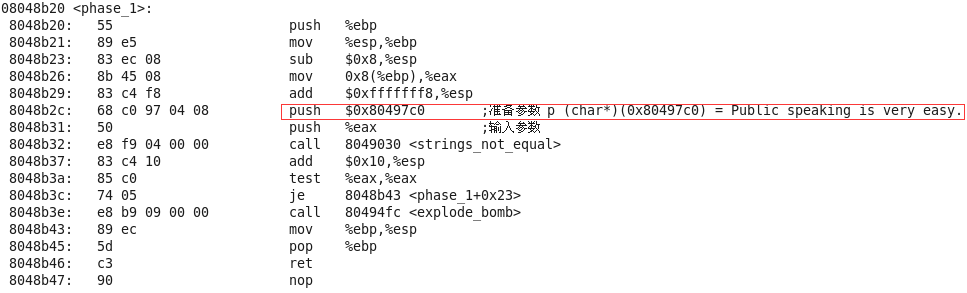
**2.2.7 阶段7 隐藏阶段**

任务描述：找出隐藏阶段开启方式并且拆除隐藏阶段的炸弹。

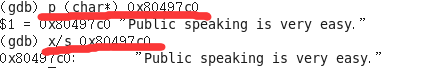
## 2.3实验过程

2.3.1 : 字符串比较

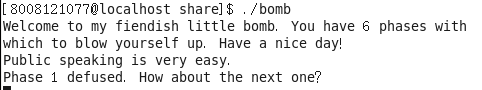
阅读汇编代码，call命令的参数准备阶段用命令查看立即数表示地址



两种查看方法



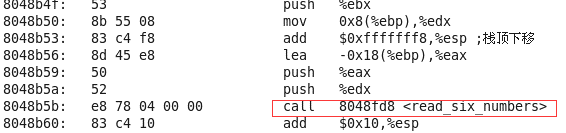
得出答案：Public speaking is very easy.



进入第二关

2.3.2 循环

分析汇编代码有提示读6个数 我们输入6个数字



我们以call 80494fc <explode\_bomb>为突破点开始分析。

ebp为栈底指针同时可以，ebp由下向上为参数1····n

所以第一次比较的是第一个参数。第一个数就是1

往后R[ebx] = 1 后面有R[ebx]++

同时看到寄存器esi参数地址给到了esi

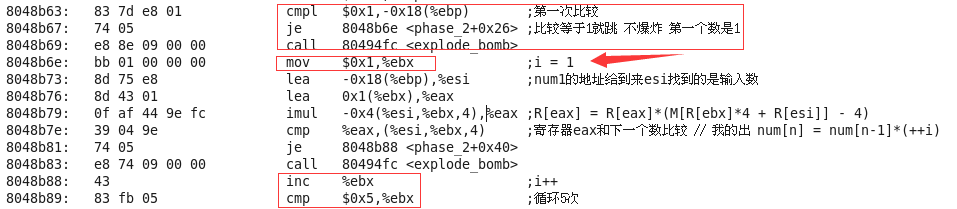
eax比ebx的值大一

最后最难分析的就是imul指令



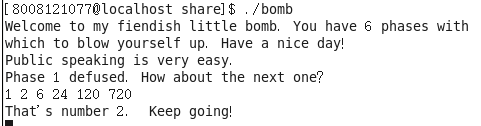
-0x4(%esi,%ebx,4),%eax 第一个源操作找到的是esi指向的输入参数的下一个数（int类型）

得出num[n] = num[n – 1] \* [i + 1]

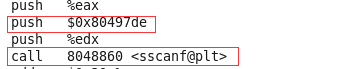


有第一个数是1 后一个数是前一个数\*（++i）

的出答案1 2 6 24 120 720

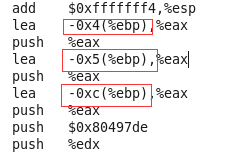


2.3.3 条件分支



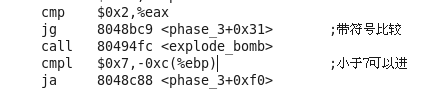
从scanf入手可知调用输入函数，要准备参数。猜测非寄存器值的功能  


查看的知要输入的数据为两整数和一个字符第二个就是看数据特点了



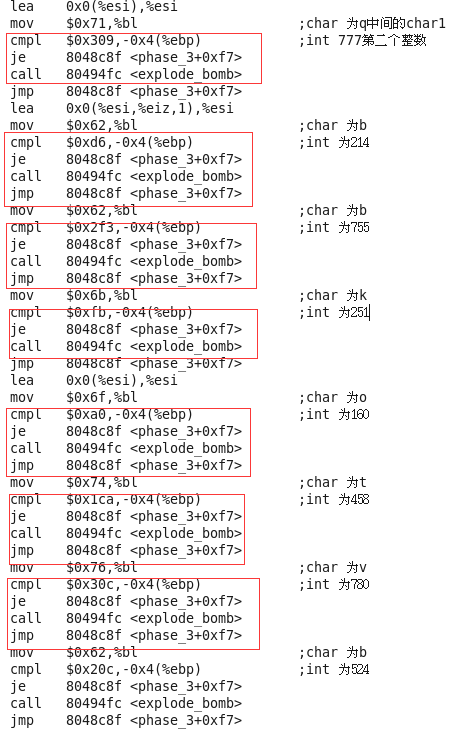
esp栈顶指针下划了12Byte一个为4B一个为1B最后一个为4B 字节对齐所以有从C开始进一步验证了数据输入的格式

继续



cmpl加ja可知第一个参数必须要小于7才可以使炸弹不爆炸

下面就是多个重复度很高的代码并且还有一个8位的寄存器bl来



## 2.4实验小结

//写出对此实验过程的总结