

**实验报告**



**题目： 拆解二进制炸弹**

**班 级： 2021211321**

**学 号： 2021213586**

**姓 名： 郭栩源**

**学 院： 计算机学院**

**2022 年 11 月 12日**

一、实验目的

1.理解C语言程序的机器级表示。

2.初步掌握GDB调试器的用法。

3.阅读C编译器生成的x86-64机器代码，理解不同控制结构生成的基本指令模式，过程的实现。

1. 实验环境
2. 服务器
   1. 服务器：10.120.11.12（X86版本）：**2022年11月10日23:59分关机（二周）；提交作业截止时间2022年11月13日00：00，延迟扣分（每晚一天扣1.5分）**
   2. 服务器：10.99.0.230（华为arm版本）：**2022年11月25日23:59分关机（三周）；提交作业截止时间2022年11月27日00：00，延迟扣分（每晚一天扣1.5分）**
   3. 服务器密码找回指南：<https://khbgo05wz2.feishu.cn/docx/doxcnioT927j2gDSTwHoAQLGVpg>
3. MobaXterm
4. Linux
5. Objdump命令反汇编
6. GDB调试工具
7. 积分榜（http://10.120.11.13:19240/scoreboard）

三、实验内容

登录bupt1服务器，在home目录下可以找到Evil博士专门为你量身定制的一个bomb，当运行时，它会要求你输入一个字符串，如果正确，则进入下一关，继续要求你输入下一个字符串；否则，炸弹就会爆炸，输出一行提示信息并向计分服务器提交扣分信息。因此，本实验要求你必须通过反汇编和逆向工程对bomb执行文件进行分析，找到正确的字符串来解除这个的炸弹。

本实验通过要求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。 “binary bombs”是一个Linux可执行程序，包含了5个阶段（或关卡）。炸弹运行的每个阶段要求你输入一个特定字符串，你的输入符合程序预期的输入，该阶段的炸弹就被拆除引信；否则炸弹“爆炸”，打印输出 “BOOM!!!”。炸弹的每个阶段考察了机器级程序语言的一个不同方面，难度逐级递增。

为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bomb文件，可以单步跟踪调试每一阶段的机器代码，也可以阅读反汇编代码，从中理解每一汇编语言代码的行为或作用，进而设法推断拆除炸弹所需的目标字符串。实验2的具体内容见实验2说明。

四、实验步骤及实验分析

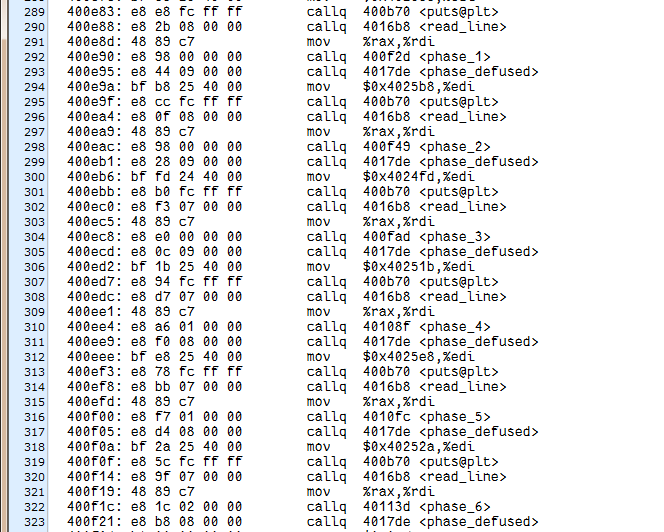
建议按照：准备工作、阶段1、阶段2、…等来组织内容

各阶段需要有操作步骤、运行截图、分析过程的内容

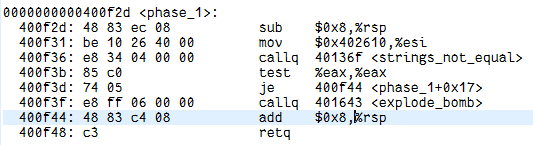
**准备工作：**使用Objdump反汇编bomb，生成bomb.txt，阅读汇编语言程序。使用gdb调试bomb，在explode\_bomb处设置断点，防止炸弹爆炸。



阅读bomb.txt的main函数部分，知每个阶段我们需要读入一个字符串，若读入正确字符串则进入下一阶段，否则炸弹引爆。



**阶段一：**阅读phase\_1函数的汇编语言代码，知：传入strings\_not\_equal函数的两个参数分别在寄存器%rdi和%rsi中。%rdi存储函数read\_line的返回值，即从stdin中读入的字符串，而%rsi中存储地址0x402610下的内容。

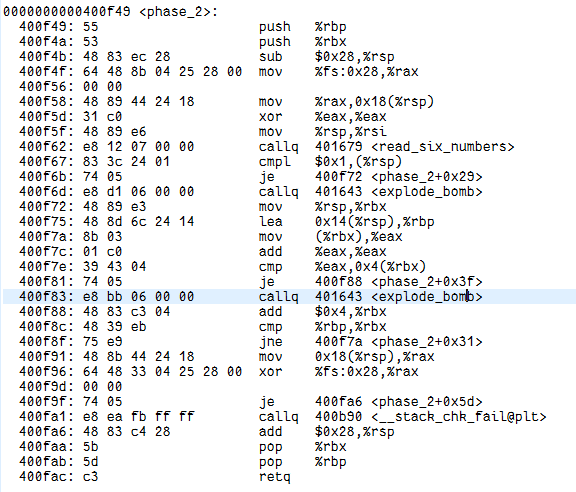


使用指令查看地址0x402610下的内容：



输入“There are many ways of defusing your bomb.”，阶段一结束。

**阶段二：**阅读phase\_2函数的汇编语言代码。用x/s 0x402941知：read\_six\_numbers读入六个整数。由400f67行中cmpl $0x1,(%rsp)知第一个整数为1。400f7a到400f8x行为循环，由cmp %rbp,%rbx，jne 400f7a <phase\_2+0x31>知终止条件为读到第六个数。在循环中由add %eax,%eax，cmp %eax,0x4(%rbx)可知读入的后一个整数是前一个整数的两倍。



于是输入：“1 2 4 8 16 32”，阶段二完成。



**阶段三：**首先查看sscanf参数表：



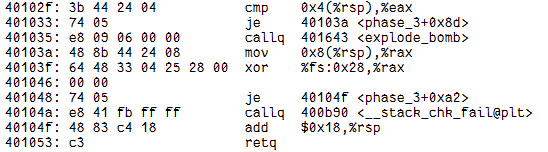
由400fdd: 83 3c 24 07 cmpl $0x7,(%rsp)400fe1和400fe1: 77 3b ja 40101e <phase\_3+0x71>知，第一个参数大于7会引爆炸弹，故第一个参数为0到6。不妨让第一个参数为6（不同参数会跳转到不同的位置，但本题有多组答案，我只需要求出其中一个）。由400fe6行jmpq \*0x402680(,%rax,8)，并用指令打印\*0x402680(,%rax,8)内容（此时%rax值为6，打印地址为0x4026b0）：



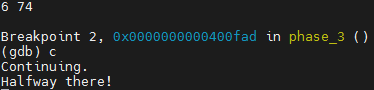
再查看401010内容：



知%eax被赋值为0x4a=74，然后跳转到40102f。0x4（%rsp）为读入的第二个参数，它要与%eax=74相等，否则引爆炸弹，故输入的两个参数为6、74。



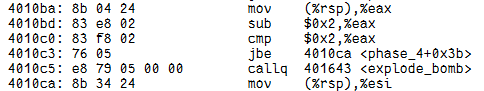
输入“6 74”，阶段三完成。



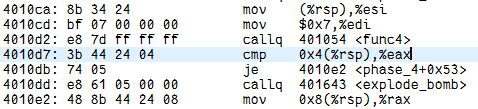
**阶段四：**阅读phase\_2函数的汇编语言代码，由下图知该阶段读入两个参数。



由下图代码知，第二个参数在无符号下大于等于2，小于4，此处选择3（同样的，此题此题可有多个答案，只要求出一个即可）。



由下图代码知，fun4的参数为7,（%rsp）=3，而输入的第一个参数等于fun4的返回值。



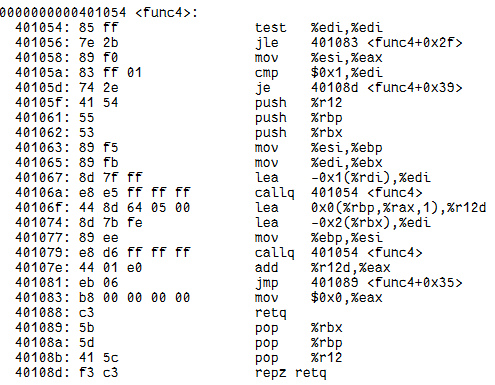
而由下图中代码知fun4是递归函数，可以总结到如下形式：

a=0时，fun4（a，b）=0；

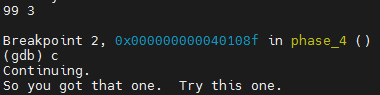
a=1时，fun4（a，b）=b；

a>=2时，fun4（a，b）=fun4（a-1，b）+fun4（a-2，b）+b

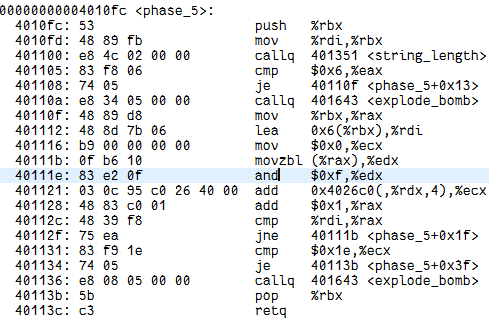
将7,3带入，有fun4（7,3）=99



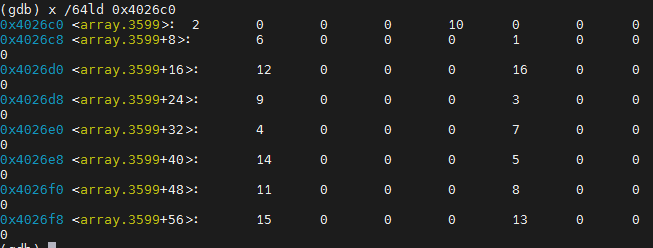
故输入“99 3”，阶段四结束。



**阶段五：**

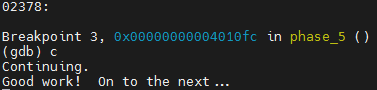


由phase\_5汇编代码知，阶段五读入一个长度为6的字符串，从0x40111b到0x40112f为循环，由0x401112行lea 0x6(%rbx),%rdi知循环执行6次，每次将读入字符串中的一位与0xf相与，相当于取其低位，并将其值存在%rdx中。然后取地址为0x4026c0+4\*%rdx的值，将其加给%rcx。6次循环后%rcx的值应为30。打印地址为0x4026c0+4\*%rdx的值的所有可能值如下：



故取%rdx的值为0,2,3,7,8,10。此处取string为02378：（对应ascii码为48,50,51,55,56,58,取低位后为0,2,3,7,8,10）。

输入“02378：”，阶段五结束。



**阶段六：**

**图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成**

阶段六实现的是链表操作，但是由于时间问题我并未能够完全解决。目前已知的第一个条件是阶段六要求输入六个整数a,b,c,d,e,f，且这六个整数在无符号下小于6，即1<=a,b,c,d,e,f<=6。第二个条件是这六个整数互不相同。

阶段六根据输入6个整数的顺序构造链表，其中1～6对应的地址分别为0x6042f0,0x604300,0x604310,0x604320,0x604330,0x604340。而这六个地址对应的内容分别为249,17f,05b,038,210,3d7如下图：

文本

描述已自动生成文本

描述已自动生成

阶段六要求按照地址对应内容从大到小顺序输入这六个整数，故猜测答案应为“6 1 5 2 3 4”。

五、总结体会

总结心得（包括实验过程中遇到的问题、如何解决的、过关或挫败的感受、实验投入的时间和精力、意见和建议等）

本实验投入精力大概16个小时左右，包括前期了解bomblab以及复习gdb相关指令，以及推导每一个炸弹的拆除过程。前三个炸弹和第五个炸弹我认为相对比较容易，而第四个炸弹（递归函数）和第六个炸弹（链表）则较难。尤其是第六个炸弹，阶段六的代码量很大，尽管我将其分为三个段落（第一个段落确定输入数据的限制条件，第二个段落确定链表的构造，最后一个段落判断链表是否合法），仍然不是很好理解。

实验中遇到的困难主要在于第六阶段，前五个阶段完成后还是比较有成就感的。感受就是其实单个指令还是比较好理解的，但是很多指令一起出现、不断跳转就让程序变得很难理解，而且没有变量这种抽象概念，可读性确实比高级语言差很多。

本次实验让我更深入地了解了汇编语言的运用，收获了很多。

六、诚信声明（不签扣10分）

需要填写如下声明，并在底部给出手写签名的电子版。

在完成本次实验过程中，我曾分别与以下各位同学就以下方面做过交流：

1、简单描述交流内容，例如：来自\*\*\*的建议，采用\*\*\*方式\*\*\*

2、

此外，我还参考了以下资料：

1. 网址等

在我提交的程序中，还在对应的位置以注释形式记录了具体的参考内容。

我独立完成了本次实验除以上方面之外的所有工作，包括分析、设计、编码、调试与测试。

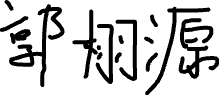
我清楚地知道，从以上方面获得的信息在一定程度上降低了实验的难度，可能影响起评分。

我从未使用他人代码，不管是原封不动地复制，还是经过某些等价转换。

我未曾也不会向同一课程（包括此后各届）的同学复制或公开我这份程序的代码，我有义务妥善保管好它们。

我编写这个程序无意于破坏或妨碍任何计算机系统的正常运行。

我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按照0分计。



（签名）