

**课 程 实 验 报 告**

**课程名称： 计算机系统基础实验**

**实验名称： 二进制程序分析**

**院 系 ： 计算机科学与技术**

**专业班级 ： 计算机图灵班202301班**

**学 号 ： U202315064**

**姓 名 ： 刘铖**

**指导教师 ： 王多强**

**2024年 10 月 8 日**

**一、实验目的与要求**

通过逆向分析一个二进制程序（称为“二进制炸弹”）的构成和运行逻辑，加深对理论课中关于程序的机器级表示各方面知识点的理解，增强反汇编、跟踪、分析、调试等能力。

实验环境：Ubuntu，GCC，GDB等。

**二、实验内容**

作为实验目标的二进制炸弹（binary bombs）可执行程序由多个“关”组成。每一个“关”（阶段）要求输入一个特定字符串，如果输入满足程序代码的要求，该阶段即通过，否则程序输出失败。实验的目标是设法得到得出解除尽可能多阶段的字符串。

为了完成二进制炸弹的拆除任务，需要通过反汇编和分析跟踪程序每一阶段的机器代码，从中定位和理解程序的主要执行逻辑，包括关键指令、控制结构和相关数据变量等等，进而推断拆除炸弹所需要的目标字符串。

实验源程序及相关文件：

bomb.c 主程序

phases.o 各个阶段的目标程序

support.c 完成辅助功能的目标程序

phases.h support.h 公共头文件

**阶段1： 串比较 phase\_1(char \*input);**

要求输出的字符串(input) 与程序中内置的某一特定字符串相同。提示：找到与input串相比较的特定串的地址，查看相应单元中的内容，从而确定input 应输入的串。

**阶段2：循环 phase\_2(char \*input);**

要求在一行上输入 6个整数数据，与程序自动产生的 6个数据进行比较，若一致，则过关。提示：将输入串input拆分成 6个数据由函数 read\_six\_numbers(input, numbers) 完成。之后是各个数据与自动产生的数据的比较，在比较中使用了循环语句。

**阶段3：条件分支 phase\_3(char \*input);**

要求输入一个整数数据，该数据与程序自动生成的 一个数据比较，相等则过关。提示：在自动生成数据时，使用了 switch … case 语句。

**阶段4：递归调用和栈 phase\_4(char \*input);**

要求在一行中输入两个数，第一个数表示在一个有序的数组（或者binary search tree）中需要搜索到的数，该数是在一定范围之内的；第二个数表示找到搜索数的路径（在树的左边搜索编码为二进制位0，在树的有边搜索编码为二进制位1）。

**阶段5：指针和数组访问 phase\_5(char \*input);**

要求在一行中输入一个串，该串与程序自动生成的串相同。在生成串和比较串时，使用了数组和指针。

**阶段6：链表、结构、指针的访问 phase\_6(char \*input);**

要求在一行中输入6个数，这6个数是一个链表中结点的顺序号（从 1 到6）。按照输入的顺序号，将对应链表结点中的值形成一个数组。若该数组是按照降序排列的，则过关。

1. **实验记录及问题回答**

首先对可执行文件进行反汇编得到汇编代码，便于进行静态阅读，理解代码内容。阅读汇编代码容易找到主函数<main>，与 C 源码的 main 函数对应。在主函数中，分别调用了 <phase\_1> 到 <phase\_6> 六个函数，因此我们只要分别阅读这六个函数的代码即可。

首先阅读 <phase\_1> 的源码，发现其调用了 <string\_not\_equal> 函数，并根据结果选择是否调用 <explode\_bomb>。使用 gdb 进行动态调试，输出传入 <string\_not\_equal> 的两个参数，发现一个是读入的字符串，另一个是 I am for medical liability at the federal level.故输入该字符串即可拆除炸弹。

再阅读 <phase\_2> 的源码，发现其调用了 <read\_six\_numbers> 函数，故容易猜到 phase2 的输入是 6 个数字。再阅读代码发现这段代码先判断了某个位置 pos 是否是 1，然后进行 5 次循环，每次判断对应位置是否是下一个位置的两倍。故输入 1 2 4 8 16 32 即可拆除炸弹。

阅读 <phase\_3> 的源码可以猜测这个代码是使用了一个 switch，每个分支对应着代码中的 <.L31>，<.L32> 等部分，且根据代码中 sscanf 的参数个数，猜出输入应为两部分。使用 gdb 进行调试，输出 sscanf 的第一个参数，即格式化字符串，发现输入是两个数字。代码比较了第一个数字与 7 的无符号大小关系，故输入应是 0到7。输入 1 选择分支1进入，发现最终会比较第二个参数与某个数进行一系列计算后的结果是否相等，那么第二个输入直接输入这个计算出的结果即可。

阅读 <phase\_4> 的源码，其调用了一个递归函数 <fun4>，这个函数传入三个参数，其中第一个参数是第一个输入。我们需要这个函数返回 7。观察到当直接传入 7 是，函数不递归直接返回 7，故第一个输入为 7。接着 phase4 比较了第二个输入和 7 的大小关系，那么第二个输入使用 7 即可。

阅读 <phase\_5> 的源码，输入是一个字符串，根据对函数 <string\_length> 的返回值的处理，判断输入字符串长度为 6。接着一个循环，把每个字符 and 上 15，再在对应地址找到对应的字母，组成新的字符串，判断与目标串是否相等。使用 gdb 从内存中直接读取目标串，然后倒推出原串即可。

阅读 <phase\_6> 源码，输入是六个数。接着是一个双重循环，用以判断输入的六个数是否是排列。接着再进入一个循环，里面使用了一个未知的位置。使用 gdb 读出对应地址的内容，发现是一系列结构体，分别是值、编号和指针。容易看出这是一个链表。重新阅读循环代码，发现是用读入的排列对链表的顺序进行重整。接下来一个循环遍历链表，要求链表的值单调递增，按要求找出重整链表的排列，输入即可。

接下来尝试进入 <secret\_phase>。根据提示在四阶段结束后加入一个字符串，跟踪程序进程发现有一个字符串比较，直接 gdb 读取出来即可。然后即可进入 secret\_phase。其中调用了一个 fun7，有两个参数,第一个是地址，第二个是值。进入fun7后读取对应地址，发现内部存储的是一个结构体，内部包含一个值以及两个指针。结合函数实现，容易看出这是一个二叉查找树上找值的函数，返回值对应查找的路径。那么容易得知需要的返回值7对应的路径，直接在 gdb 中读取内存之后模拟查找即可找到对应的值。作为输入即可拆除炸弹。

**四、体会**

通过本次“二进制炸弹”实验，我对程序的逆向分析、汇编语言的阅读与调试、以及二进制程序的运作机制有了更深刻的理解。在实验过程中，我学到了以下几点重要的知识与技能：

**1.反汇编与动态调试的实践**  
这是我第一次深入使用反汇编工具（如GDB）对二进制程序进行逐步调试。在逐阶段拆解二进制炸弹的过程中，我切实感受到了汇编语言的强大之处。通过GDB的动态调试，跟踪每一阶段的代码流，特别是函数调用和寄存器操作，使我更加明确了机器级指令与源代码的关联。这种对程序运行的细粒度控制，也加深了我对编译器和底层执行环境的理解。

**2.对函数调用机制和内存管理的深入理解**  
实验中的各个阶段展示了不同的程序结构：从简单的字符串比较、数字循环，到复杂的递归调用和二叉查找树的操作。这使我对函数调用、栈帧管理和递归机制有了更直观的认识。例如，在递归和二叉树阶段的分析中，我必须对内存中的数据结构和指针进行细致的分析，才能准确地理解程序的行为并找到正确的输入。

**3.综合运用编程技能和分析能力**  
每个阶段都有不同的挑战，涉及循环、条件分支、递归、数组和链表等多种编程结构。我不仅需要正确理解每个阶段的代码逻辑，还要根据实际的程序行为推断出合理的输入。通过这些操作，我增强了对复杂程序逻辑的推断能力和问题解决的思维。

4.**工具的高效使用**  
通过本次实验，我熟悉了GDB调试工具的使用，包括设置断点、逐步调试和内存查看等功能。这些工具帮助我更加精确地定位程序中的问题，并有效地推断出解决方案。此外，反汇编工具和静态分析工具也极大地辅助了我对程序的静态结构理解。

总的来说，这次实验不仅提升了我的二进制程序分析和调试技能，也让我认识到解决复杂问题时条理化思维的重要性。这些经验将对我未来的学习和开发工作产生深远的影响。