

**实验报告**



**题目： 拆解二进制炸弹**

**班 级： 2023211307**

**学 号： 2023211075**

**姓 名： 魏生辉**

**学 院： 计算机学院**

**2024 年 10 月 26 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. Ubuntu 2204.2.33.0
3. Objdump命令反汇编
4. GDB调试工具
5. GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1

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。

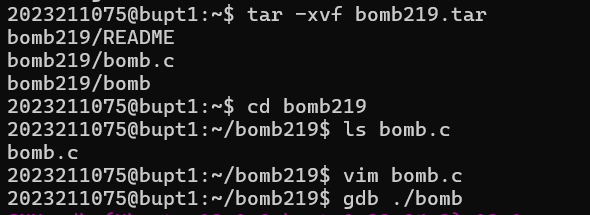
为完成二进制炸弹拆除任务，需要使用gdb调试器和objdump来反汇编bomb文件，可以单步跟踪调试每一阶段的机器代码，也可以阅读反汇编代码，从中理解每一汇编语言代码的行为或作用，进而设法推断拆除炸弹所需的目标字符串。实验2的具体内容见实验2说明。

四、实验步骤及实验分析

**（六个阶段+隐藏阶段）**

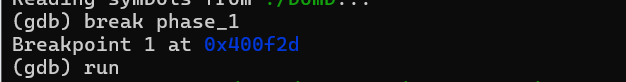
**准备工作**

**先获取炸弹**

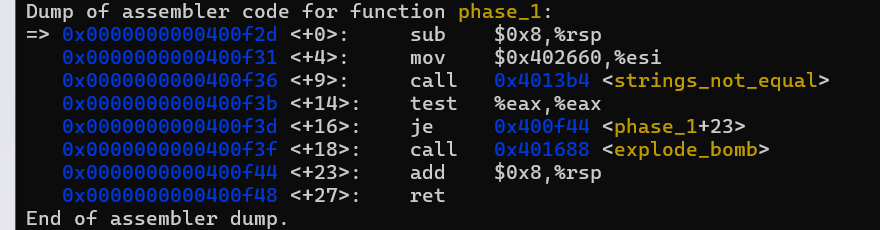


**阶段一**

**①先打个断点**



**②反汇编看一下代码**



**③确定预期字符串**

为了找到 phase\_1 期望的字符串，可以直接检查地址 0x402660 中的内容：

在 gdb 中输入以下命令：

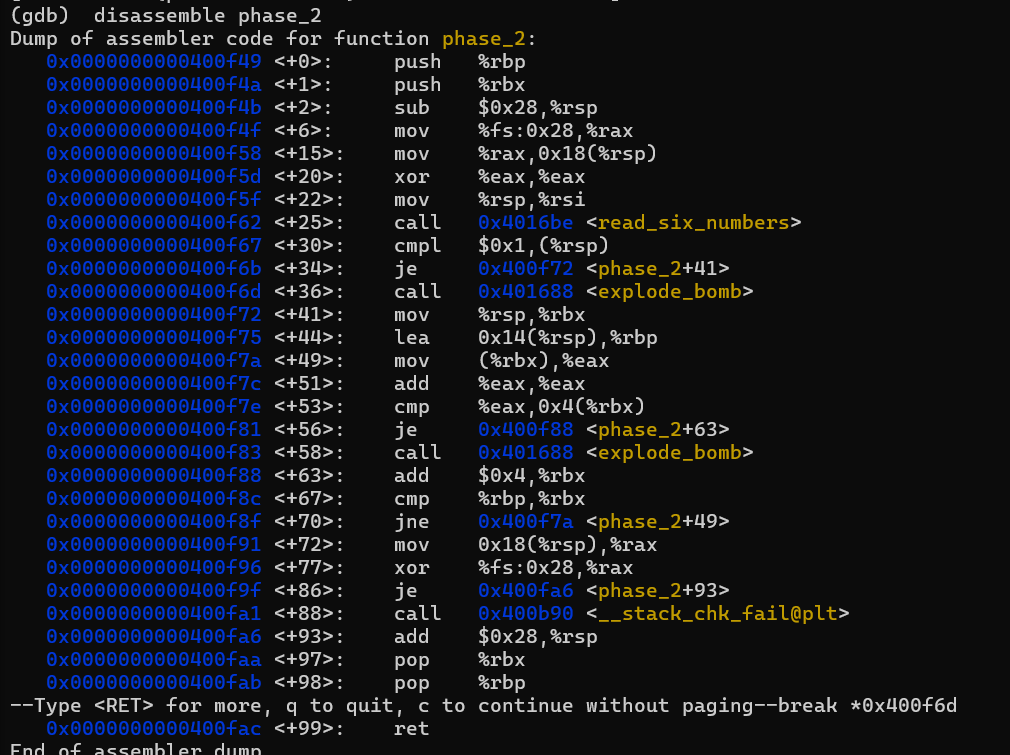


**④通过**



**阶段二**

**①先设置断点并反汇编**



**②分析汇编代码**

·调用 read\_six\_numbers：

在 <+25> 处调用 read\_six\_numbers，读取用户输入的六个整数，并将其存储在栈上，从地址 %rsp 开始。

·第一个数字检查：

在 <+30> 处比较栈上的第一个整数与 1 是否相等：cmpl $0x1,(%rsp)。

如果第一个数字不是 1，则会调用 explode\_bomb 使炸弹爆炸。

·循环检查后续数字：

代码将第一个数字 (%rbx) 取出，并乘以 2 后与第二个数字 0x4(%rbx) 比较：add %eax,%eax 和 cmp %eax,0x4(%rbx)。

如果相等，则继续检查下一个数字。否则，调用 explode\_bomb。

使用循环（由 cmp %rbp,%rbx 和 jne 0x400f7a 控制）逐个检查每个数字。每次循环中，当前数字应为前一个数字的两倍。

**③总结 phase\_2 输入要求**

第一个数字为 1。

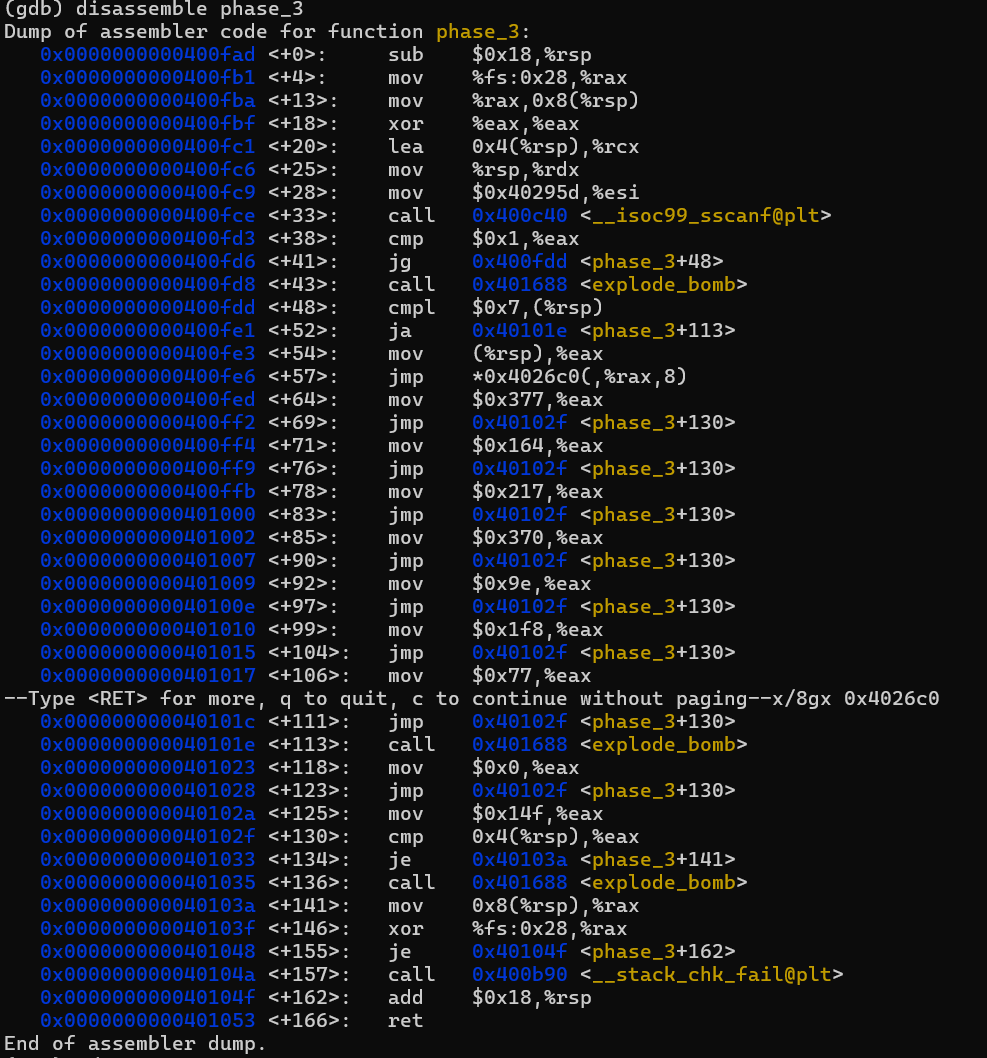
从第二个数字开始，每个数字应为前一个数字的两倍。

**④通过**

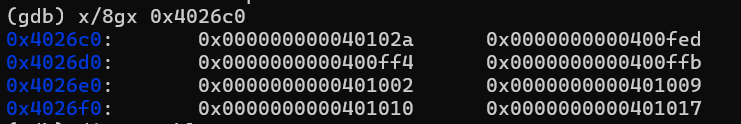


**阶段三**

**①先设置断点并反汇编**



**②查看跳转表**



**③分析跳转表**

输入 0 跳转到 0x40102a，期望第二个输入为 0x14f（335）。

输入 1 跳转到 0x400fed，期望第二个输入为 0x377（887）。

输入 2 跳转到 0x400ff4，期望第二个输入为 0x164（356）。

输入 3 跳转到 0x400ffb，期望第二个输入为 0x217（535）。

输入 4 跳转到 0x401002，期望第二个输入为 0x370（880）。

输入 5 跳转到 0x401009，期望第二个输入为 0x9e（158）。

输入 6 跳转到 0x401010，期望第二个输入为 0x1f8（504）。

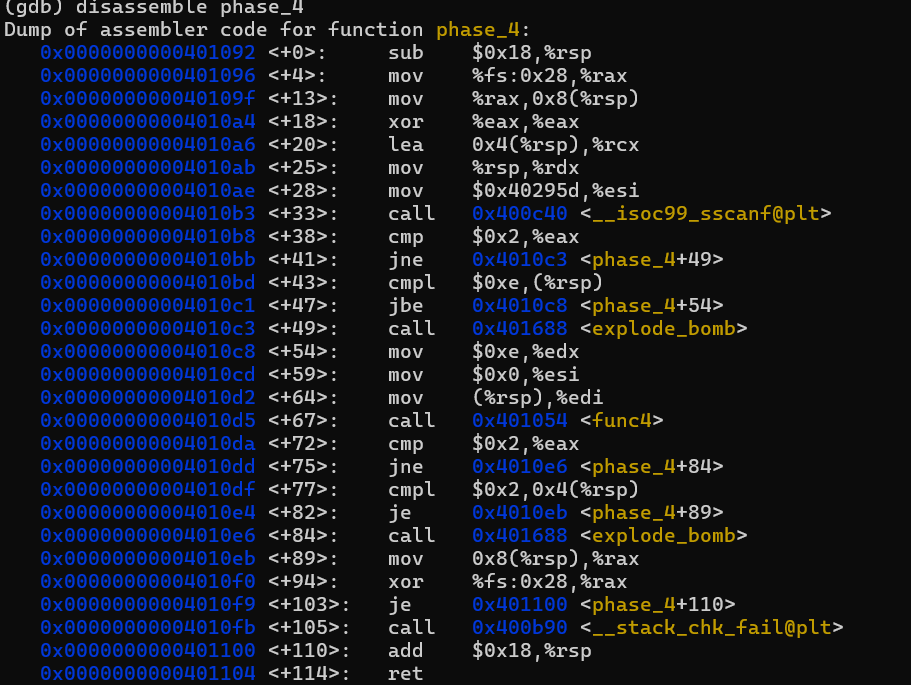
输入 7 跳转到 0x401017，期望第二个输入为 0x77（119）。

**④通过**



**阶段四**

**①设置断点和反汇编**



**②分析汇编代码**

·调用 sscanf 读取输入：

在 <+33> 处，phase\_4 使用 sscanf 从用户输入中读取两个整数，并将其存储在栈上。

·检查输入数量：

在 <+38> 处，phase\_4 将 sscanf 的返回值（存储在 eax 中）与 2 比较，以确保读取到两个输入值。

如果 sscanf 的返回值不等于 2，则调用 explode\_bomb。

·第一个数字范围检查：

在 <+43> 处，检查第一个数字是否小于或等于 0xe（即 14）。如果超出该范围，调用 explode\_bomb。

·调用 func4 函数：

在 <+67> 处，调用 func4 函数，并传递以下参数：edx = 0xe（即 14）

esi = 0

edi（第一个输入的值）

·检查 func4 的返回值：

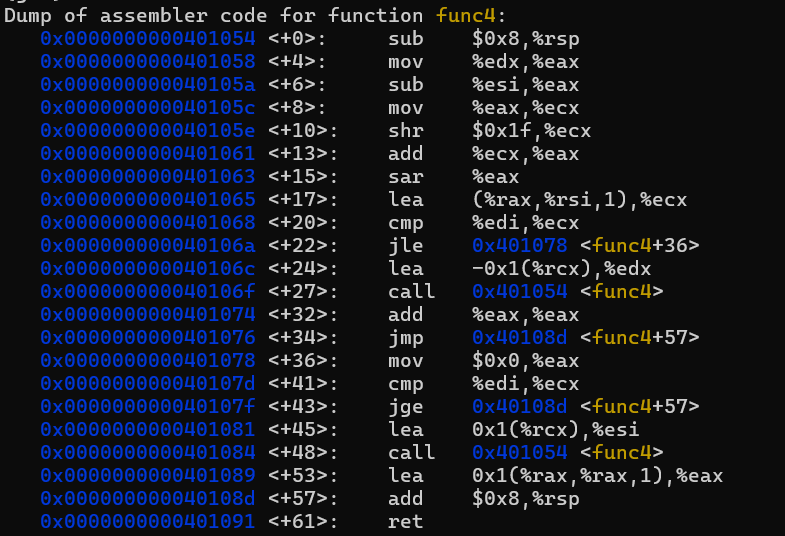
在 <+72> 处，将 func4 的返回值与 2 比较，如果不等于 2，则调用 explode\_bomb。

·第二个数字检查：

在 <+77> 处，将第二个输入与 2 进行比较，如果不等于 2，则调用 explode\_bomb。

**所以，第二输入一定是2。**

**③分析func4代码**



·参数传递：

edi 是目标值（即第一个输入）。

esi 和 edx 是上下限（esi = 0, edx = 14）。

·二分查找逻辑：

func4 计算 esi 和 edx 的中间值 ecx。

ecx 与 edi 比较，根据比较结果递归调用自身。

如果 edi 大于 ecx，则缩小范围至上半部分并递归调用；如果小于 ecx，则递归调用下半部分。

·递归返回条件：

当 ecx 等于 edi 时，停止递归。

每次递归返回时，会累加 eax 的值。

**③计算满足条件的第一个输入**

我们需要找到value1，使得func4(value1, 0, 14) == 2。

·尝试 value1 = 4：

·第一次调用： func4(4, 0, 14)

mid = 7,4 < 7，递归调用func4(4, 0, 6)

返回值= 2 \* result

·第二次调用： func4(4, 0, 6)

mid = 3,4 > 3，递归调用func4(4, 4, 6)

返回值= 2 \* result + 1

·第三次调用： func4(4, 4, 6)

mid = 5,4 < 5，递归调用func4(4, 4, 4)

返回值= 2 \* result

·第四次调用： func4(4, 4, 4)

mid = 4,4 == 4，返回0

·回溯计算：

第四次调用返回0，第三次调用返回2 \* 0 = 0

第三次调用返回0，第二次调用返回2 \* 0 + 1 = 1

第二次调用返回1，第一次调用返回2 \* 1 = 2

·结论：

func4(4, 0, 14) == 2

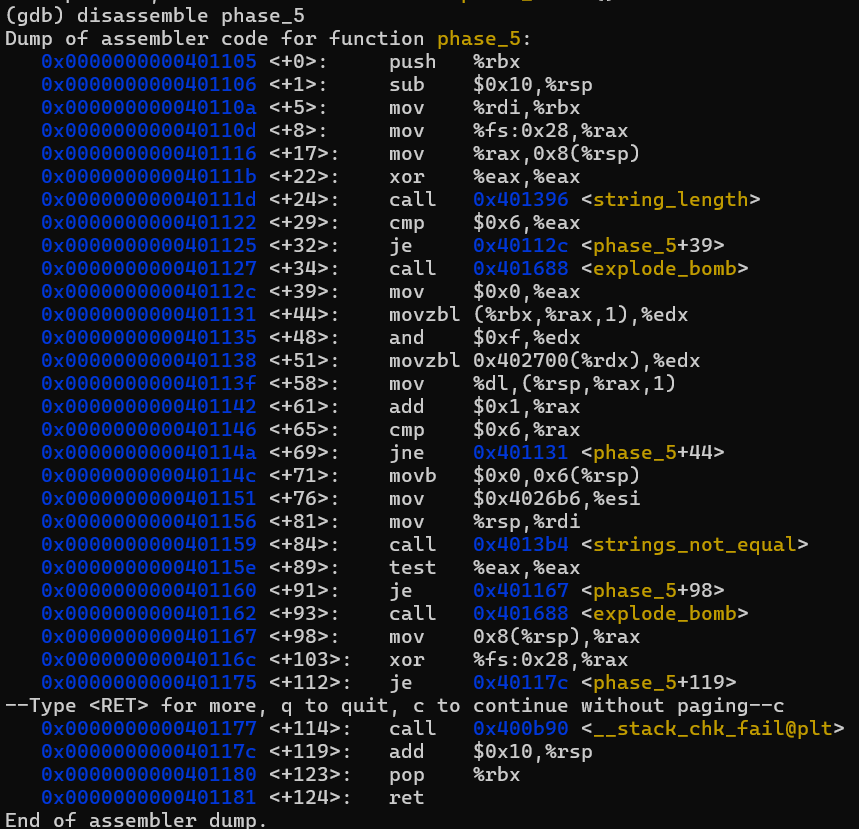
value1 = 4 满足条件。

**④通过**



**阶段五**

**①设置断点和反汇编**



**②分析代码**

·字符串长度检查：

在 <+24> 调用了 string\_length 函数，计算输入字符串的长度，并将结果保存在 eax 中。

在 <+29> 处将 eax 与 6 进行比较，确保输入字符串长度为 6。如果长度不为 6，调用 explode\_bomb 引发爆炸。

·字符映射：

从 <+39> 开始的循环将输入字符串的每个字符逐个映射到新的字符。

每次循环：movzbl (%rbx,%rax,1),%edx：取出输入字符串的一个字符。

and $0xf,%edx：将字符与 0xf 进行按位与操作，取出字符的最低 4 位（使字符值在 0 到 15 之间）。

movzbl 0x402700(%rdx),%edx：使用此值作为索引，从 0x402700 地址处读取一个字符。

结果字符被存储在栈中，构成一个新的字符串。

·字符串比较：

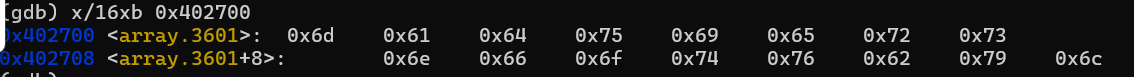
经过映射后的新字符串与地址 0x4026b6 处的字符串进行比较：在 <+84> 处调用 strings\_not\_equal 函数比较栈中的字符串和 0x4026b6 的字符串。

如果它们不相等，则调用 explode\_bomb。

**③确定目标字符串**



④**构建映射关系**：



**⑤利用索引关系寻找答案**

·根据目标字符串sabres，查找对应的索引：

's'：索引为7

'a'：索引为1

'b'：索引为13

'r'：索引为6

'e'：索引为5

's'：索引为7

·因此，我们需要找到6个字符，使得它们的低4位对应以下索引：

[7, 1, 13, 6, 5, 7]

·找到对应的输入字符

步骤1：列出可能的字符

对于每个需要的索引，列出可能的字符（取ASCII值的低4位等于索引）：

索引7

ASCII码低4位为0x7的字符：0x67：'g'

索引1

ASCII码低4位为0x1的字符：0x61：'a'

索引13

ASCII码低4位为0xd的字符：0x6d：'m'

索引6

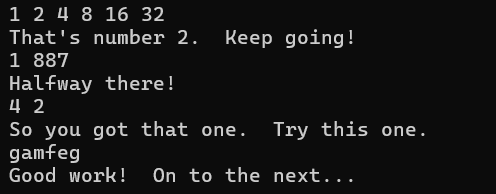
ASCII码低4位为0x6的字符：0x66：'f'

索引5

ASCII码低4位为0x5的字符：0x65：'e'

**得到字符串："gamfeg"**

**⑥全部通过**



**隐藏阶段六**



提取节点的值

将每个节点的值提取出来：

节点1（地址0x6042f0）：值部分：0x0000000100000048

低4字节：0x00000048（十进制72）

节点2（地址0x604300）：值部分：0x00000002000001ad

低4字节：0x000001ad（十进制429）

节点3（地址0x604310）：值部分：0x0000000300000155

低4字节：0x00000155（十进制341）

节点4（地址0x604320）：值部分：0x00000004000002b8

低4字节：0x000002b8（十进制696）

节点5（地址0x604330）：值部分：0x000000050000006a

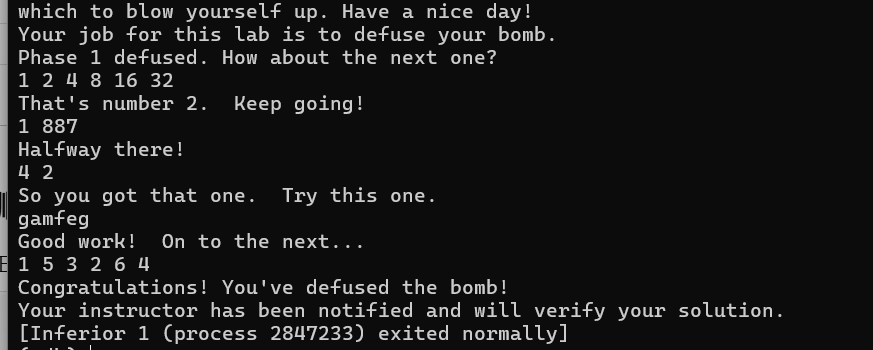
低4字节：0x0000006a（十进制106）

节点6（地址0x604340）：值部分：0x00000006000001d7

低4字节：0x000001d7（十进制471）

**建立节点位置与值的对应关系可得：1 5 3 2 6 4**

**全部通过！！！！**

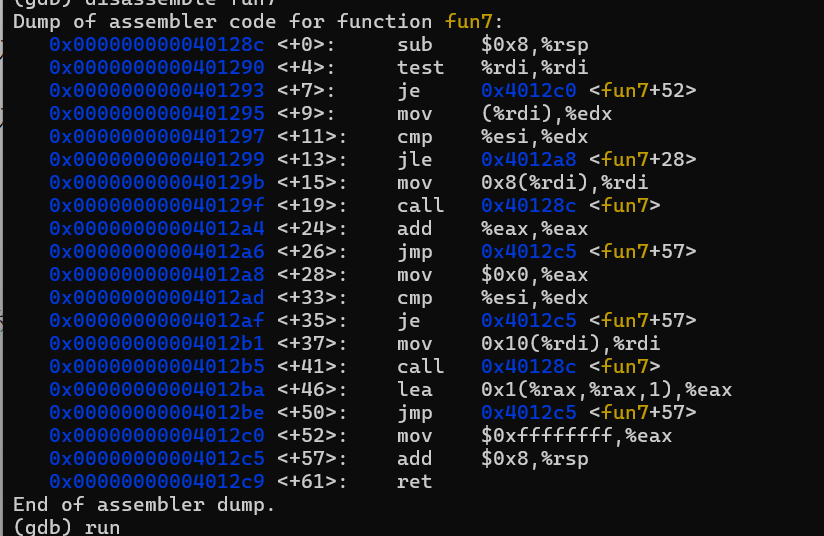


**隐藏阶段**

·首先找到入口

第四个输入 ：4 2 DrEvil

·分析fun7和二叉树



·构造二叉树

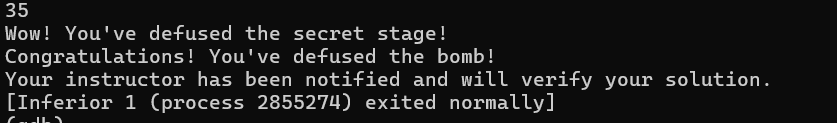
36

8 50

6 22

1 7 20 35

要求返回值是6 所以 倒推反演 答案是35



五、总结体会

1.不细心：字符串居然加了引号；

2.对汇编指令不熟悉：断点几次都没设置好；

3.最开始不知道怎么输入，**直接气急败坏，连续爆炸4次**，后来平和心情，第二个完成所有任务。

