哈爾濱工業大學

实验报告

实 验（三）

题　　目 Binary Bomb

二进制炸弹

专　　业 计算机类

学　　号 1190501614

班　　级 1903006

学　　生 cgh

指 导 教 师 史先俊

实 验 地 点 G709

实 验 日 期 2021.05.02

计算机科学与技术学院

目　录

第 1 章 实验基本信息

1.1 实验目的

熟练掌握计算机系统的 ISA 指令系统与寻址方式
熟练掌握 Linux 下调试器的反汇编调试跟踪分析机器语言的方法
增强对程序机器级表示、汇编语言、调试器和逆向工程等的理解

1.2 实验环境与工具

1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

1.2.2 软件环境

Windows7 64 位以上；VirtualBox/Vmware 11 以上；Ubuntu 16.04 LTS 64 位/优麒麟 64 位

1.2.3 开发工具

GDB/OBJDUMP；EDB；KDD 等

1.3 实验预习

- 上实验课前，必须认真预习实验指导书（**PPT 或 PDF**）
- 了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。
- 请写出 **C** 语言下包含字符串比较、循环、分支（含 **switch**）、函数调用、递归、指针、结构、链表等的例子程序 **sample.c**。
- 生成执行程序 **sample.out**。
- 用 **gcc –S** 或 **CodeBlocks** 或 **GDB** 或 **OBJDUMP** 等，反汇编，比较。
- 列出每一部分的 **C** 语言对应的汇编语言。

- **修改编译选项-O (缺省 2)、O0、O1、O2、O3，-m32/m64。再次查看生成的汇编语言与原来的区别。**
- **注意 O1 之后无栈帧，EBP 做别的用途。-fno-omit-frame-pointer 加上栈指针。**
- **GDB 命令详解 –tui 模式 ^XA 切换 layout 改变等等**
- **有目的地学习: 看 VS 的功能 GDB 命令用什么**

第 2 章 实验环境建立

2.1 Ubuntu 下 CodeBlocks 反汇编（10 分）

CodeBlocks 运行 hellolinux.c。反汇编查看 printf 函数的实现。

要求：C、ASM、内存(显示 hello 等内容)、堆栈（call printf 前）、寄存器同时在一个窗口。

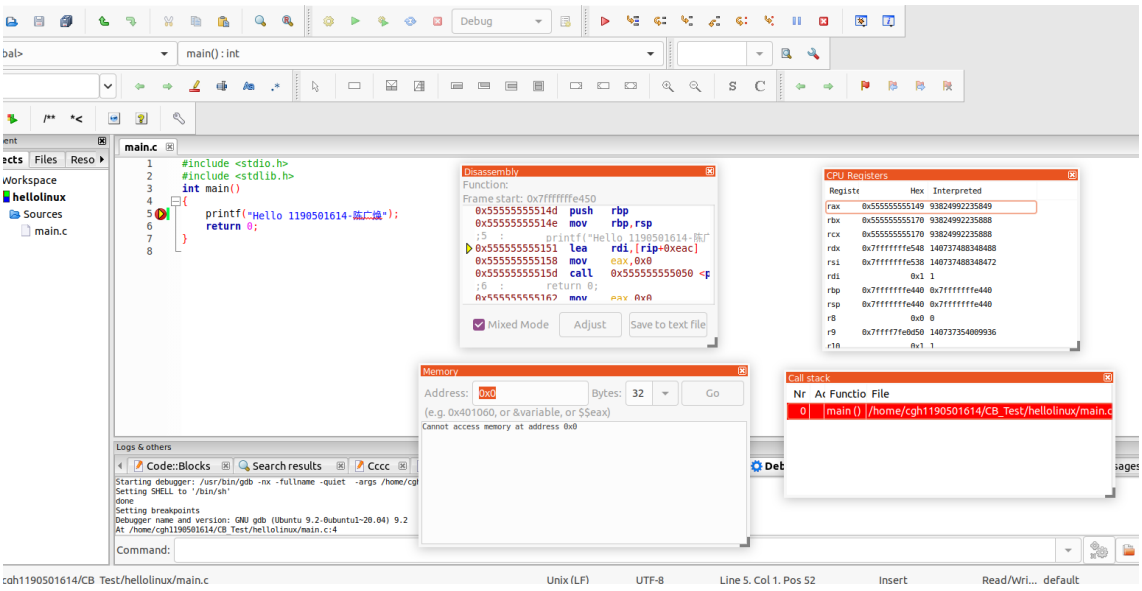


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

2.2 Ubuntu 下 EDB 运行环境建立（10 分）

用 EDB 调试 hellolinux.c 的执行文件，截图，要求同 2.1

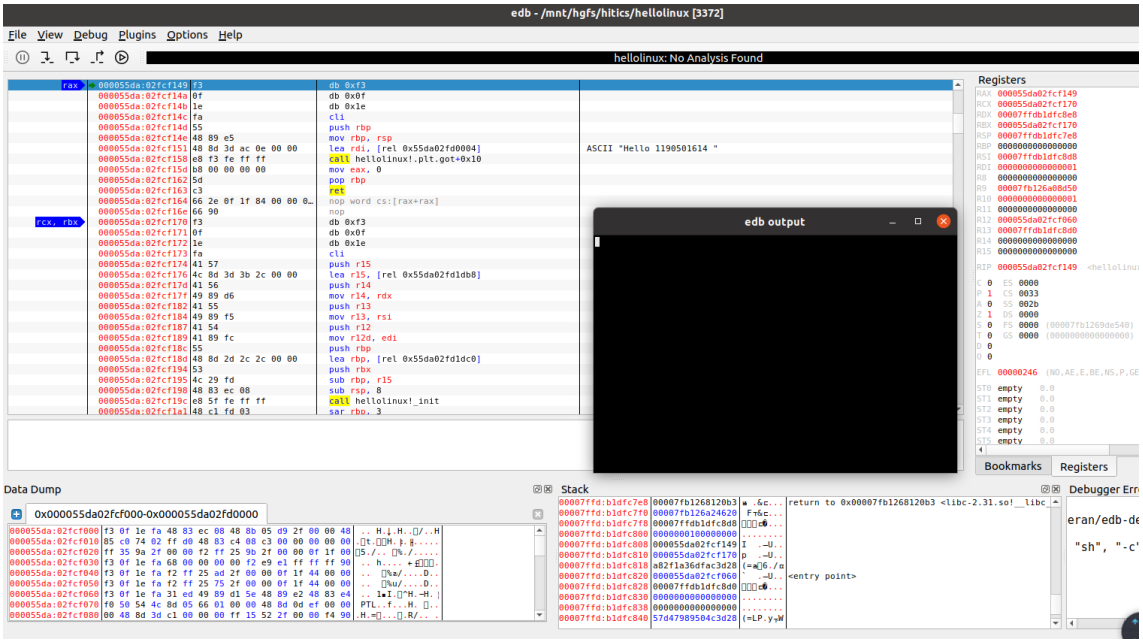


图 2-2　Ubuntu 下 EDB 截图

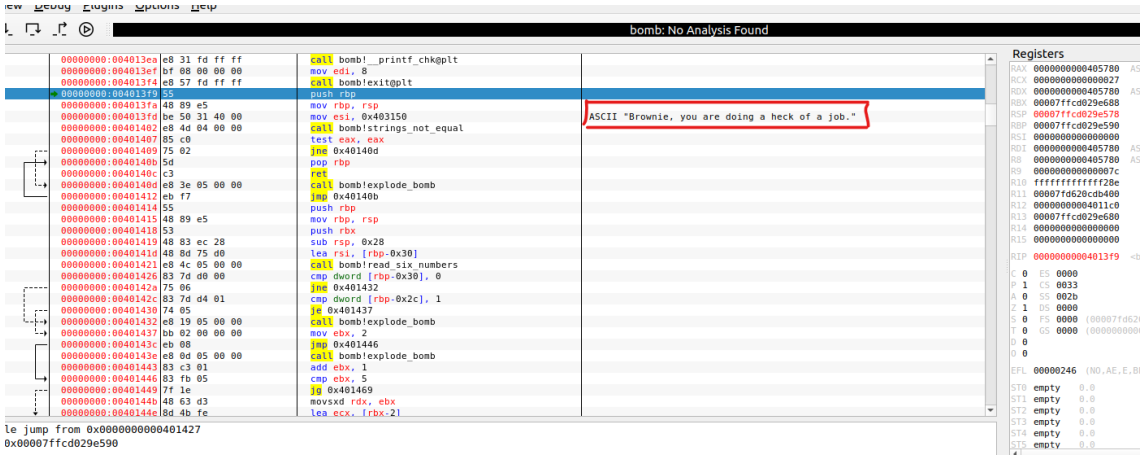
第 3 章 各阶段炸弹破解与分析

每阶段 40 分，密码 20 分，分析 20 分，总分不超过 80 分

3.1 阶段 1 的破解与分析

密码如下：Brownie, you are doing a heck of a job.

破解过程：随便输入一个字符串，进入 phase_1，找到要匹配的字符串，如下图红色标注，即为答案。

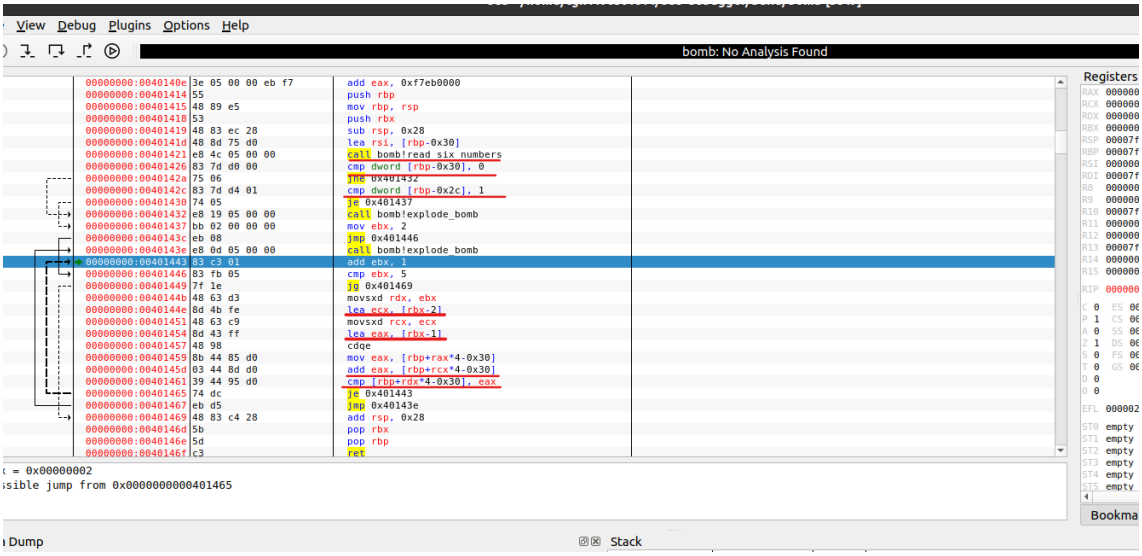


3.2 阶段 2 的破解与分析

密码如下：0 1 1 2 3 5

破解过程：

通过 read_six_numbers 这个函数可以看出要输入 6 个数字，%ebx 相当于一个计数器，计到 5 退出循环。lea ecx,[rbx-2]和 lea eax,[rbx-1]这两条指令是得到 A_{n-2} 和 A_{n-1} 项的地址，然后将其值累加到 eax，而 cmp [rbp+rdx*4-0x30],eax 这条指令则是将 A_n 与累加器的值比较（$A_{n-2}+A_{n-1}$），很明显这是一个斐波那契数列，刚开始是 0 和 1，所以答案是 0 1 1 2 3 5

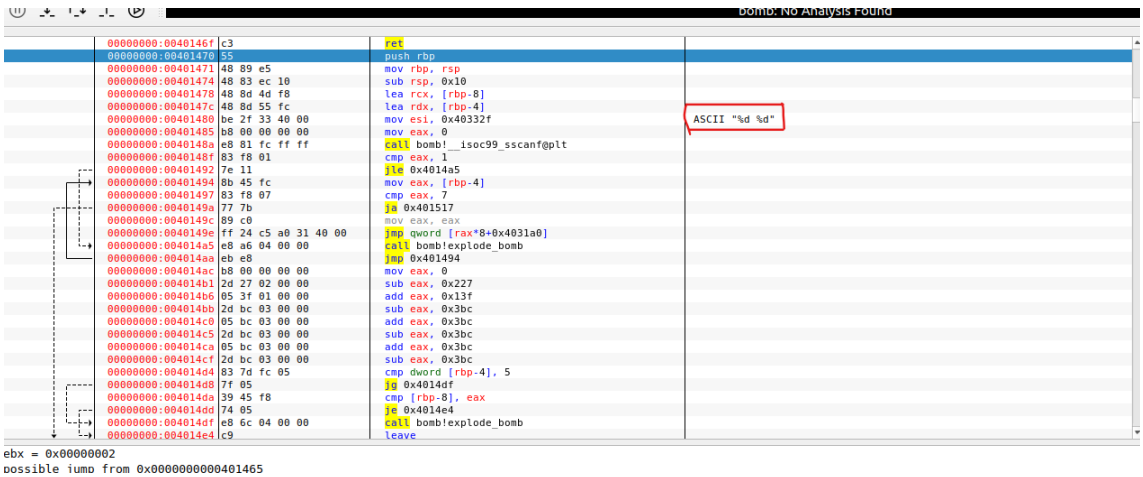


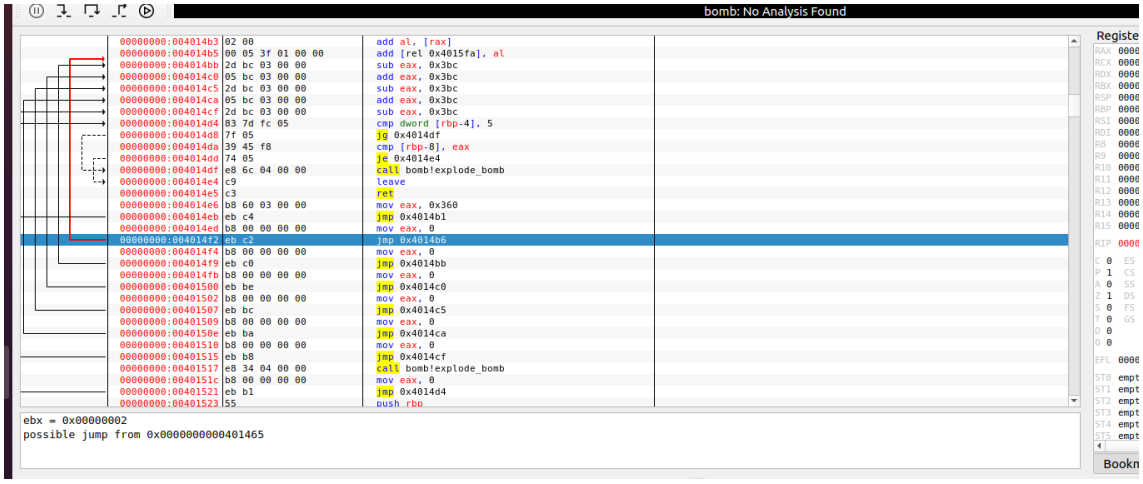
3.3 阶段 3 的破解与分析

密码如下：0　-324

破解过程：

随便输入进入 phase_3，首先发现输入格式为“%d　%d”，找到跳转表，选择其中一条路径计算即可得到其中一个答案为 0 -324，输入格式和跳转表如下所示。



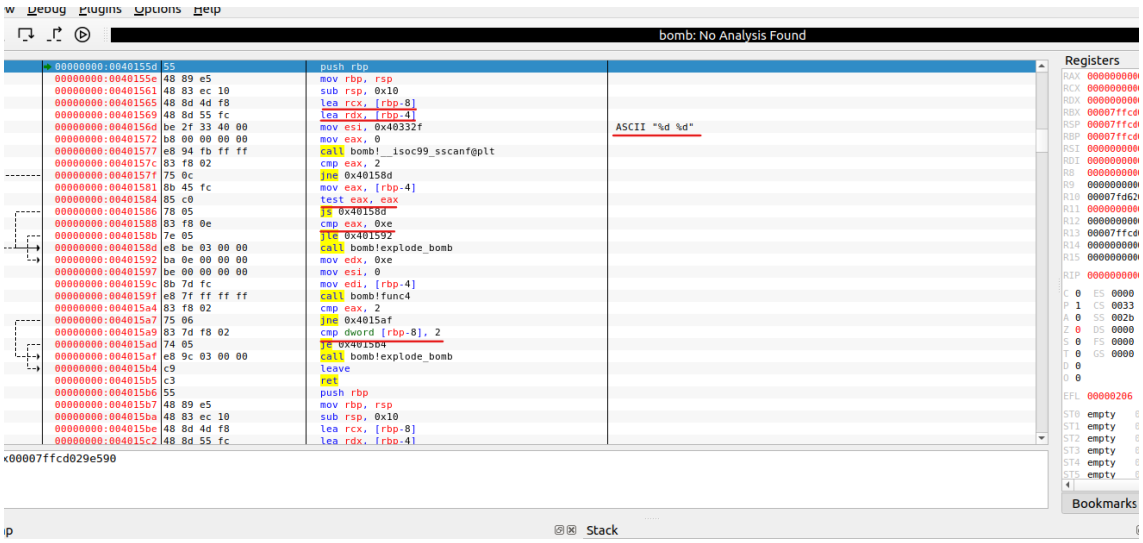


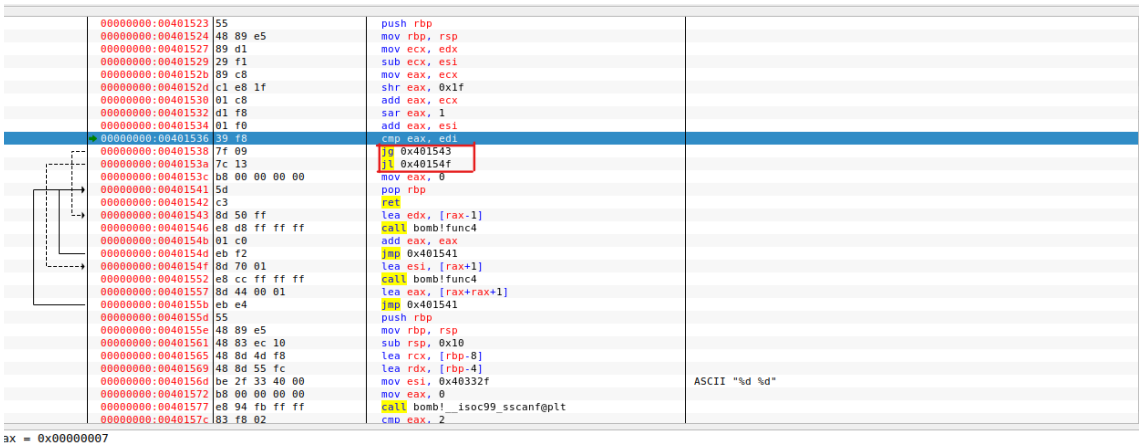
3.4 阶段 4 的破解与分析

密码如下：4 2 DrEvil

破解过程：

首先随便输入进入 phase_4,发现输入格式为 “%d %d”，两个整数分别存在%rbp-8 和%rbp-4，第一个数要大于 0 且小于 14，第二个数要等于 2。进入 func4 后发现是一个递归，当第一个参数小于 7 的时候进入另一条分支，大于 7 的时候进入另一条分支，等于 7 的时候退出递归，经过一番推理得到 4 2 是满足其中的一个答案。



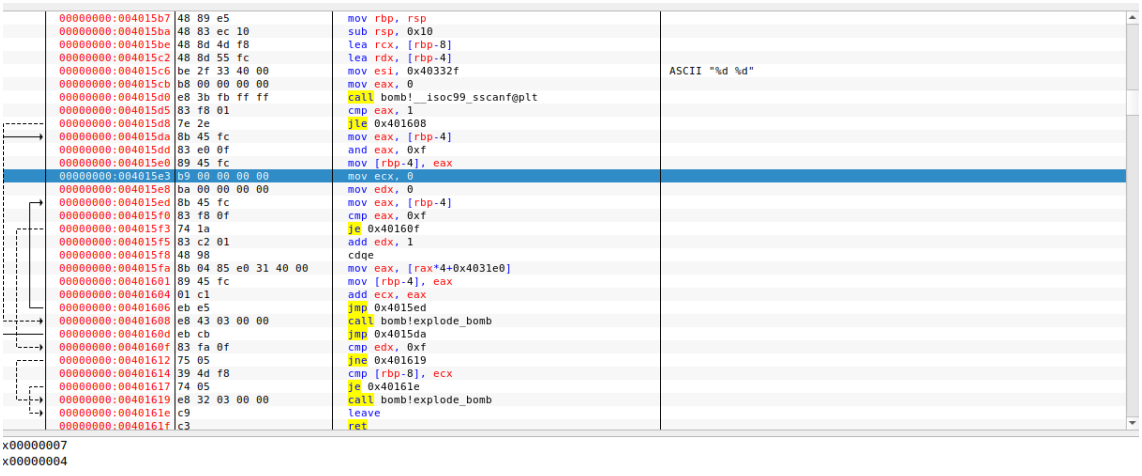


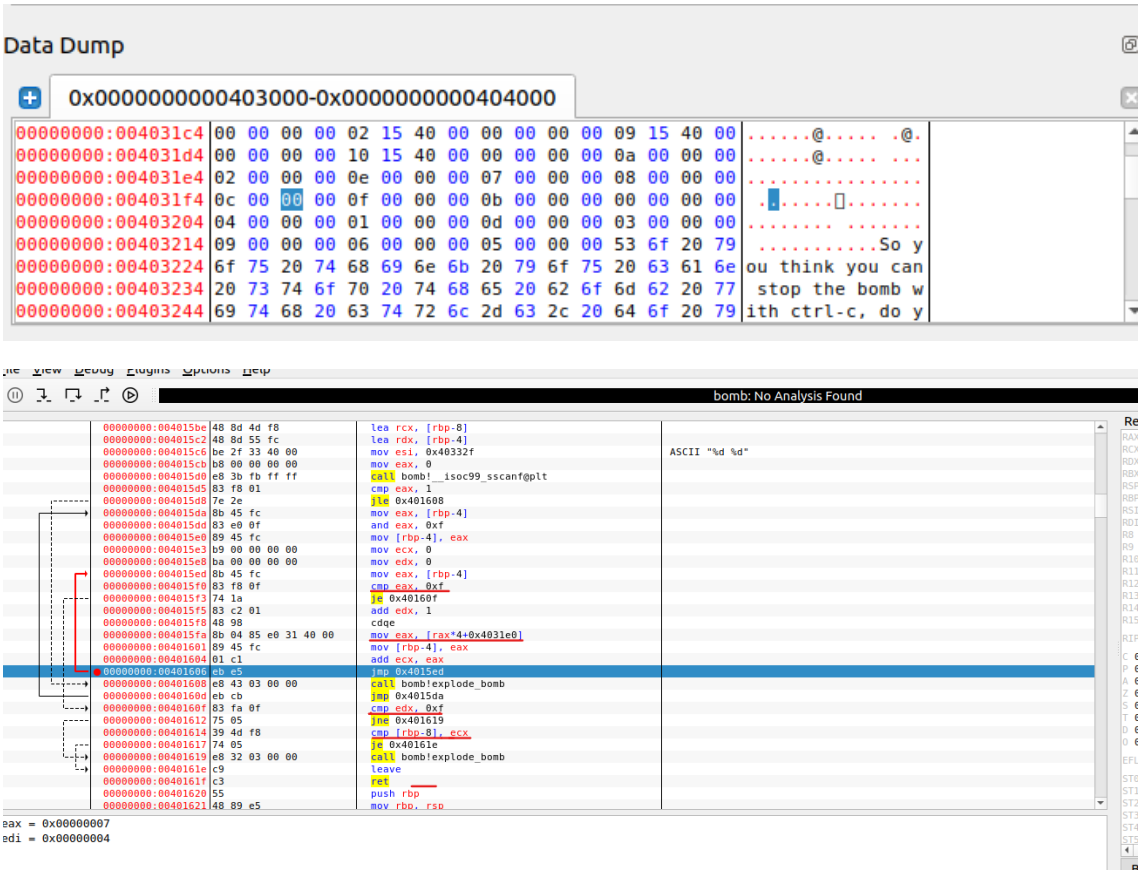
3.5 阶段 5 的破解与分析

密码如下：5 115

破解过程：

首先和之前一样进入 phase_5,输入格式为两个整数，存在%rbp-8 和%rbp-4 处。累加器%edx 从 0 开始，每次循环加 1，直到 15，每次循环将%eax 的值加到%ecx，当%eax 的值等于 15 并且%edx 累加到 15 退出循环，所以可以知道最后一次加的是 15，将该内存里的 15 个元素打印出来，反推回去可以知道第二个数是 115。



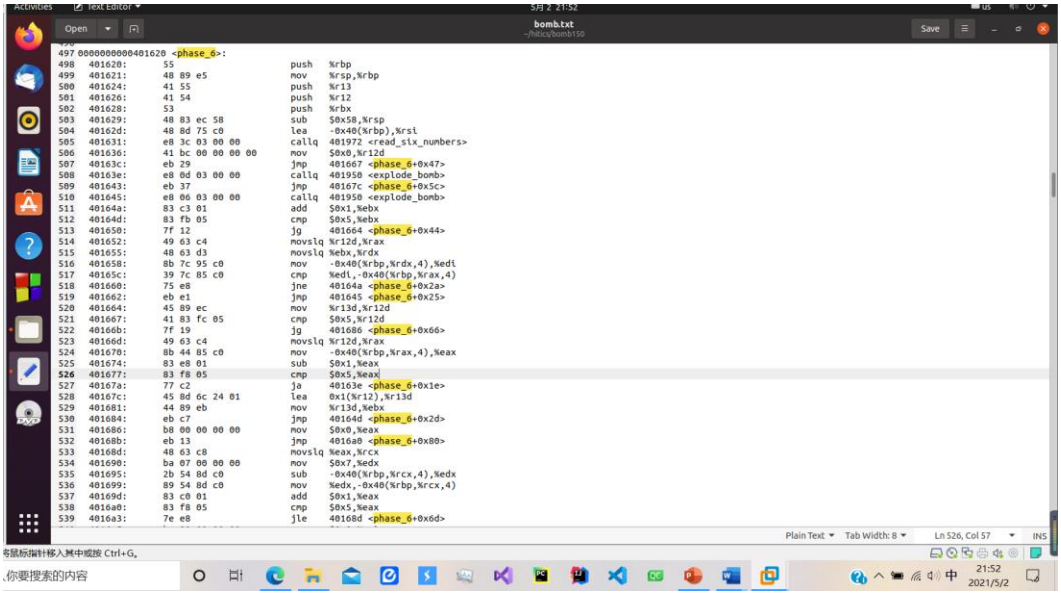


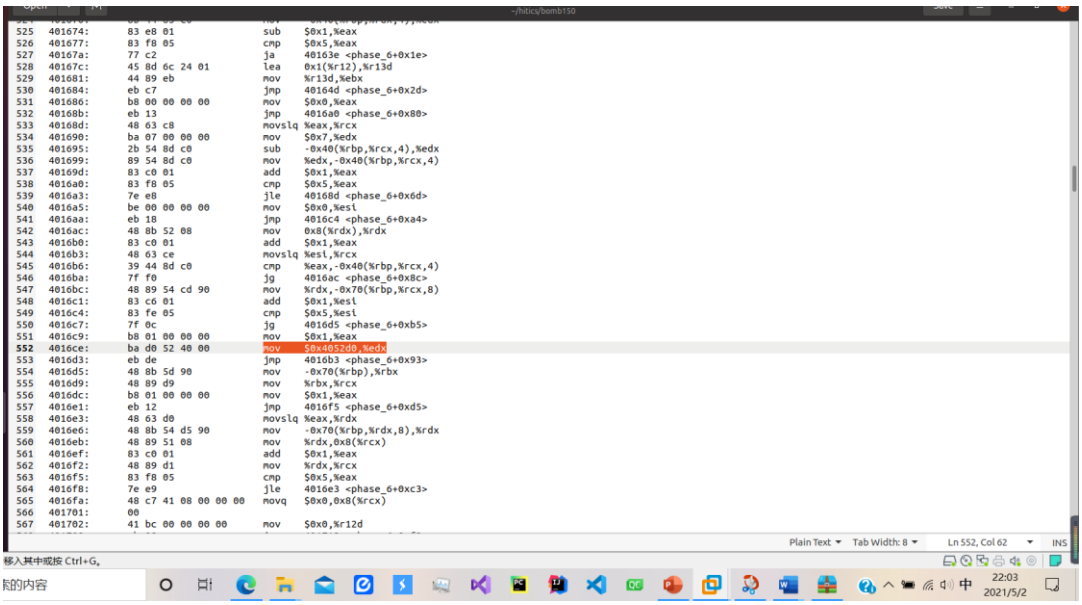
3.6 阶段 6 的破解与分析

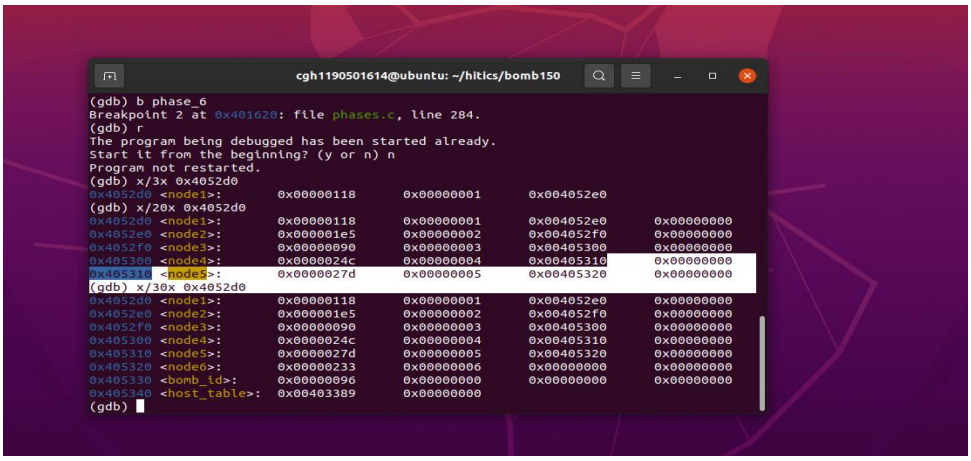
密码如下：2 3 1 5 6 4

破解过程：

首先进入 phase_6 里面，通过反汇编代码可知这是对链表里的元素从大到小排序，同时做了 7-i 的处理，我们找到链表存储的地址 0x4052d0，将里面的内容打印出来，得到下标排序为 5 4 6 2 1 3，由于做了 7-i 的处理，还原回去是 2 3 1 5 6 4 即为答案。


bomb.txt
~/hitics/bomb150
Save
0000000000401620 <phase_6>:
401620: 55 push %rbp
401621: 48 89 e5 mov %rsp,%rbp
401624: 41 55 push %r13
401626: 41 54 push %r12
401628: 53 push %rbx
401629: 48 83 ec 58 sub $0x58,%rsp
4016 2d: 48 8d 75 c0 lea -0x40(%rbp),%rsi
401631: e8 3c 03 00 00 callq 401972 <read_six_numbers>
401636: 41 bc 00 00 00 00 mov $0x0,%r12d
40163c: eb 29 jmp 401667 <phase_6+0x47>
40163e: e8 0d 03 00 00 callq 401950 <explode_bomb>
401643: eb 37 jmp 40167c <phase_6+0x5c>
401645: e8 06 03 00 00 callq 401950 <explode_bomb>
40164a: 83 c3 01 add $0x1,%ebx
40164d: 83 fb 05 cmp $0x5,%ebx
401650: 7f 12 jg 401664 <phase_6+0x44>
401652: 49 63 c4 movslq %r12d,%rax
401655: 48 63 d3 movslq %ebx,%rdx
401658: 8b 7c 95 c0 mov -0x40(%rbp,%rdx,4),%edi
40165c: 39 7c 85 c0 cmp %edi,-0x40(%rbp,%rax,4)
401660: 75 e8 jne 40164a <phase_6+0x2a>
401662: eb e1 jmp 401645 <phase_6+0x25>
401664: 45 89 ec mov %r13d,%r12d
401667: 41 83 fc 05 cmp $0x5,%r12d
40166b: 7f 19 jg 401686 <phase_6+0x66>
40166d: 49 63 c4 movslq %r12d,%rax
401670: 8b 44 85 c0 mov -0x40(%rbp,%rax,4),%eax
401674: 83 e8 01 sub $0x1,%eax
401677: 83 f8 05 cmp $0x5,%eax
40167a: 77 c2 ja 40163e <phase_6+0x1e>
40167c: 45 8d 6c 24 01 lea 0x1(%r12),%r13d
401681: 44 89 eb mov %r13d,%ebx
401684: eb c7 jmp 40164d <phase_6+0x2d>
401686: b8 00 00 00 00 mov $0x0,%eax
40168b: eb 13 jmp 4016a0 <phase_6+0x80>
40168d: 48 63 c8 movslq %eax,%rcx
401690: ba 07 00 00 00 mov $0x7,%edx
401695: 2b 54 8d c0 sub -0x40(%rbp,%rcx,4),%edx
401699: 89 54 8d c0 mov %edx,-0x40(%rbp,%rcx,4)
40169d: 83 c0 01 add $0x1,%eax
4016a0: 83 f8 05 cmp $0x5,%eax
4016a3: 7e e8 jle 40168d <phase_6+0x6d>
4016a5: be 00 00 00 00 mov $0x0,%esi
4016aa: eb 18 jmp 4016c4 <phase_6+0xa4>
4016ac: 48 8b 52 08 mov 0x8(%rdx),%rdx
4016b0: 83 c0 01 add $0x1,%eax
4016b3: 48 63 ce movslq %esi,%rcx
4016b6: 39 44 8d c0 cmp %eax,-0x40(%rbp,%rcx,4)
4016ba: 7f f0 jg 4016ac <phase_6+0x8c>
4016bc: 48 89 54 cd 90 mov %rdx,-0x70(%rbp,%rcx,8)
4016c1: 83 c6 01 add $0x1,%esi
4016c4: 83 fe 05 cmp $0x5,%esi
4016c7: 7f 0c jg 4016d5 <phase_6+0xb5>
4016c9: b8 01 00 00 00 mov $0x1,%eax
4016ce: ba d0 52 40 00 mov $0x4052d0,%edx
4016d3: eb de jmp 4016b3 <phase_6+0x93>
4016d5: 48 8b 5d 90 mov -0x70(%rbp),%rbx
4016d9: 48 89 d9 mov %rbx,%rcx
4016dc: b8 01 00 00 00 mov $0x1,%eax
4016e1: eb 12 jmp 4016f5 <phase_6+0xd5>
4016e3: 48 63 d0 movslq %eax,%rdx
4016e6: 48 8b 54 d5 90 mov -0x70(%rbp,%rdx,8),%rdx
4016eb: 48 89 51 08 mov %rdx,0x8(%rcx)
4016ef: 83 c0 01 add $0x1,%eax
4016f2: 48 89 d1 mov %rdx,%rcx
4016f5: 83 f8 05 cmp $0x5,%eax
4016f8: 7e e9 jle 4016e3 <phase_6+0xc3>
4016fa: 48 c7 41 08 00 00 00 movq $0x0,0x8(%rcx)
401701: 00
401702: 41 bc 00 00 00 00 mov $0x0,%r12d
Plain Text
Tab Width: 8
Ln 526, Col 57
INS
21:52
2021/5/2

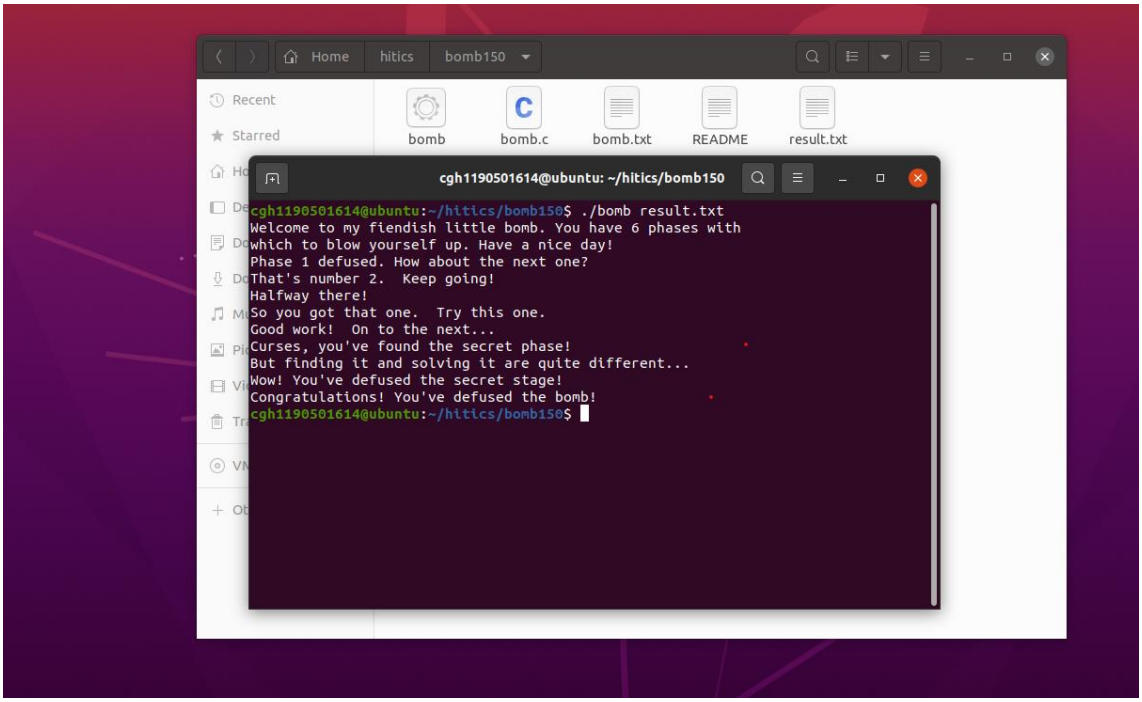

Plain Text
Tab Width: 8
Ln 552, Col 62
INS
22:03
2021/5/2

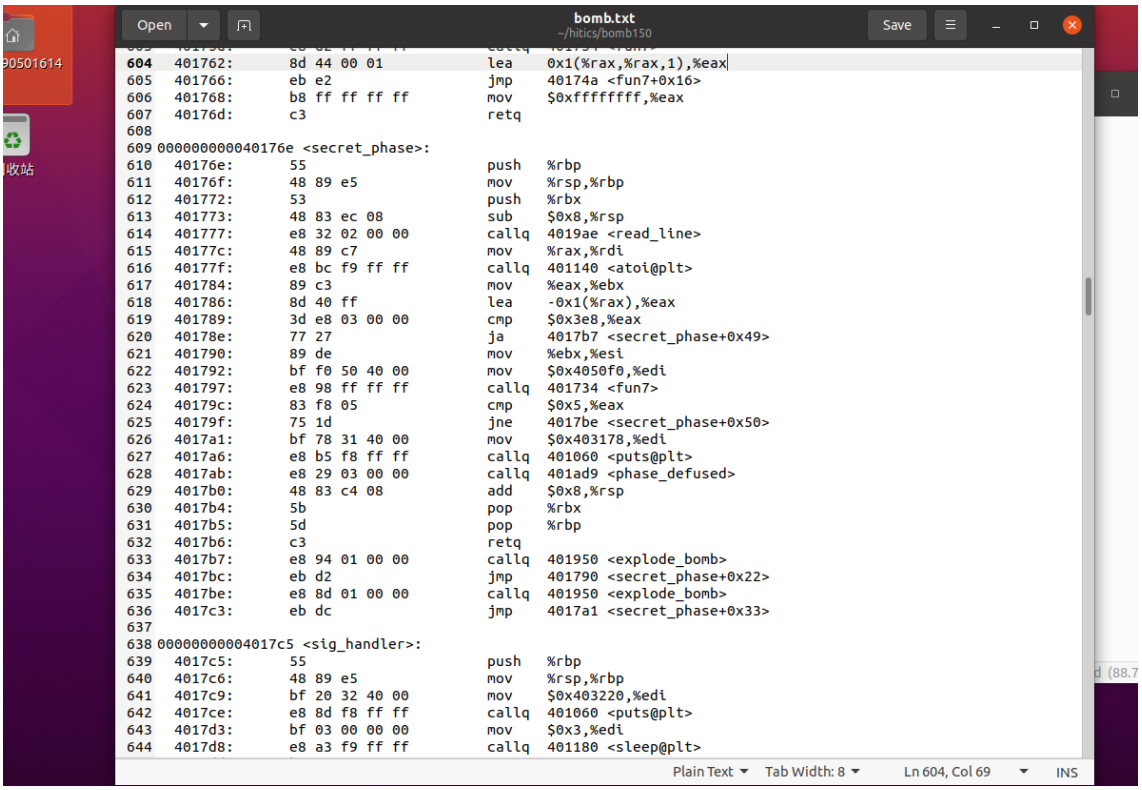

cgh1190501614@ubuntu: ~/hitics/bomb150
(gdb) b phase_6
Breakpoint 2 at 0x401620: file phases.c, line 284.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) x/3x 0x4052d0
0x4052d0 <node1>: 0x00000118 0x00000001 0x004052e0
(gdb) x/20x 0x4052d0
0x4052d0 <node1>: 0x00000118 0x00000001 0x004052e0 0x00000000
0x4052e0 <node2>: 0x000001e5 0x00000002 0x004052f0 0x00000000
0x4052f0 <node3>: 0x00000090 0x00000003 0x00405300 0x00000000
0x405300 <node4>: 0x0000024c 0x00000004 0x00405310 0x00000000
0x405310 <node5>: 0x0000027d 0x00000005 0x00405320 0x00000000
(gdb) x/30x 0x4052d0
0x4052d0 <node1>: 0x00000118 0x00000001 0x004052e0 0x00000000
0x4052e0 <node2>: 0x000001e5 0x00000002 0x004052f0 0x00000000
0x4052f0 <node3>: 0x00000090 0x00000003 0x00405300 0x00000000
0x405300 <node4>: 0x0000024c 0x00000004 0x00405310 0x00000000
0x405310 <node5>: 0x0000027d 0x00000005 0x00405320 0x00000000
0x405320 <node6>: 0x00000233 0x00000006 0x00000000 0x00000000
0x405330 <bomb_id>: 0x00000096 0x00000000 0x00000000 0x00000000
0x405340 <host_table>: 0x00403389 0x00000000
(gdb)

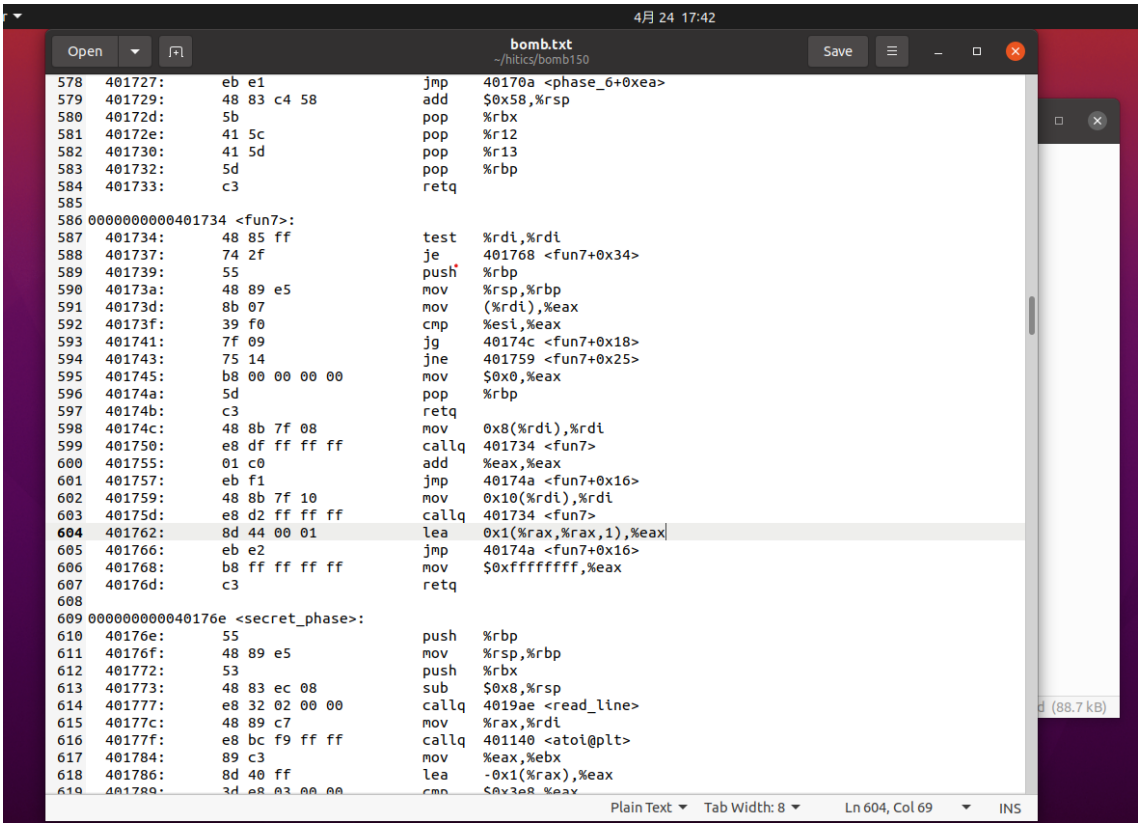
3.7 阶段 7 的破解与分析(隐藏阶段)

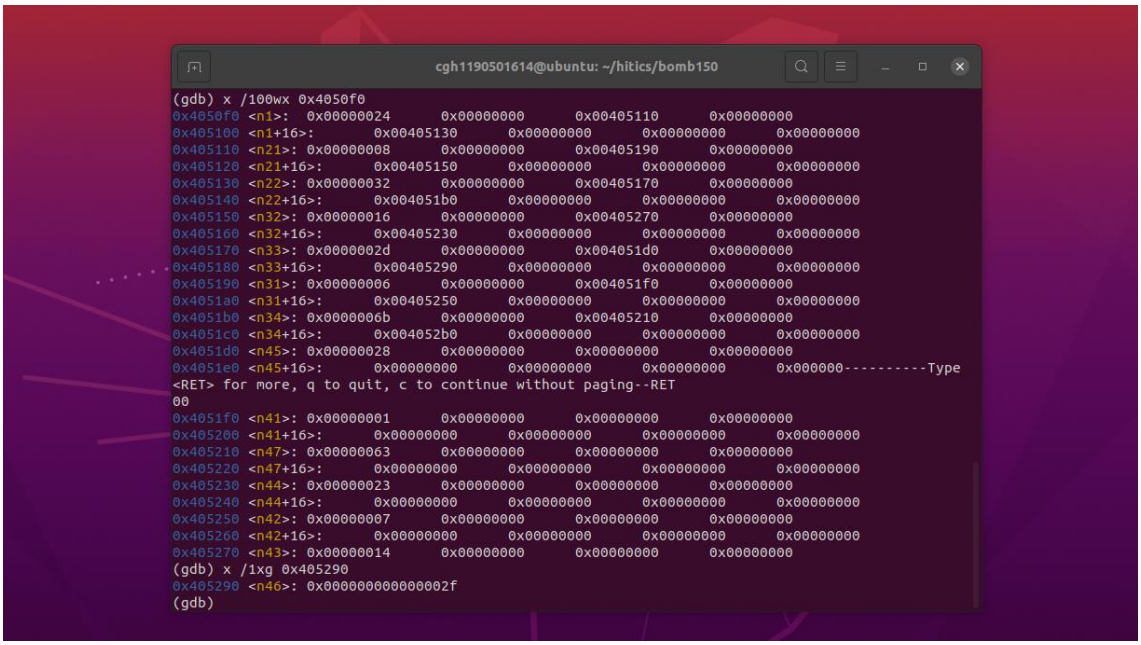
密码如下：47

破解过程：









通过 gdb 找到答案为 0x2f,对应 10 进制的 47。

第 4 章 总结

4.1 请总结本次实验的收获

本次实验对 gdb 运用更加娴熟，同时开始学习 edb 的使用，对反汇编代码的理解能力也得到提升。

4.2 请给出对本次实验内容的建议

希望老师能提前一天发实验 ppt,让我们提前安装好环境，以便上课同步进行操作。

参考文献