

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	3 -
1.1 实验目的	3 -
1.2 实验环境与工具	3 -
1.2.1 硬件环境.....	3 -
1.2.2 软件环境.....	3 -
1.2.3 开发工具.....	3 -
1.3 实验预习	3 -
第 2 章 实验环境建立	5 -
2.1 UBUNTU 下 CODEBLOCKS 反汇编（10 分）	5 -
2.2 UBUNTU 下 EDB 运行环境建立（10 分）	5 -
第 3 章 各阶段炸弹破解与分析	7 -
3.1 阶段 1 的破解与分析.....	7 -
3.2 阶段 2 的破解与分析.....	7 -
3.3 阶段 3 的破解与分析.....	8 -
3.4 阶段 4 的破解与分析.....	9 -
3.5 阶段 5 的破解与分析.....	10 -
3.6 阶段 6 的破解与分析.....	11 -
3.7 阶段 7 的破解与分析(隐藏阶段).....	13 -
第 4 章 总结	15 -
4.1 请总结本次实验的收获.....	15 -
4.2 请给出对本次实验内容的建议	15 -
参考文献	16 -

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。

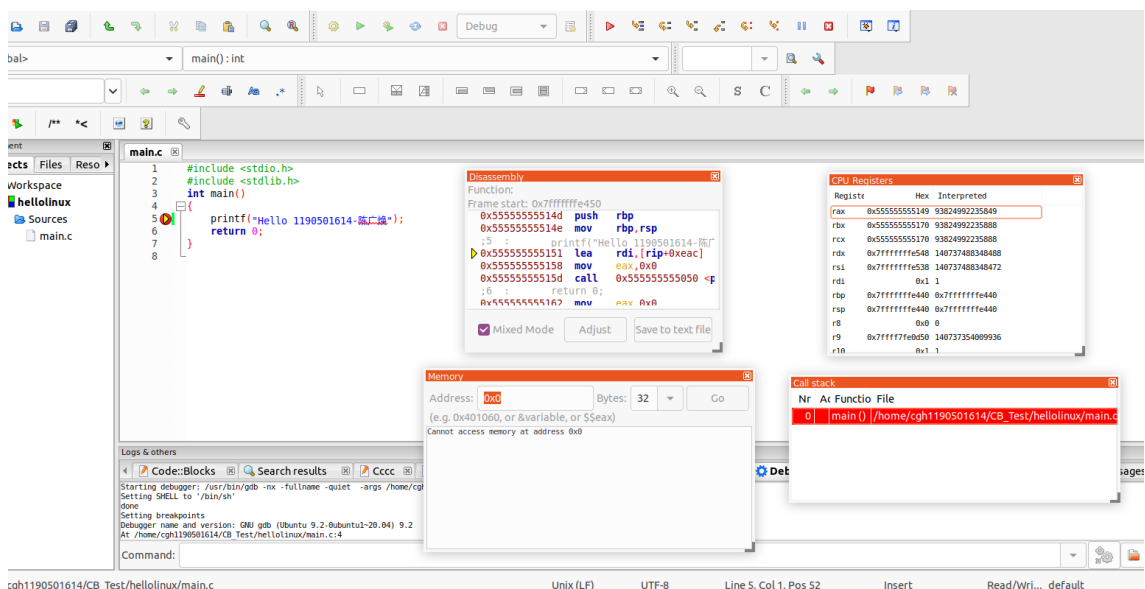


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

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

计算机系统实验报告

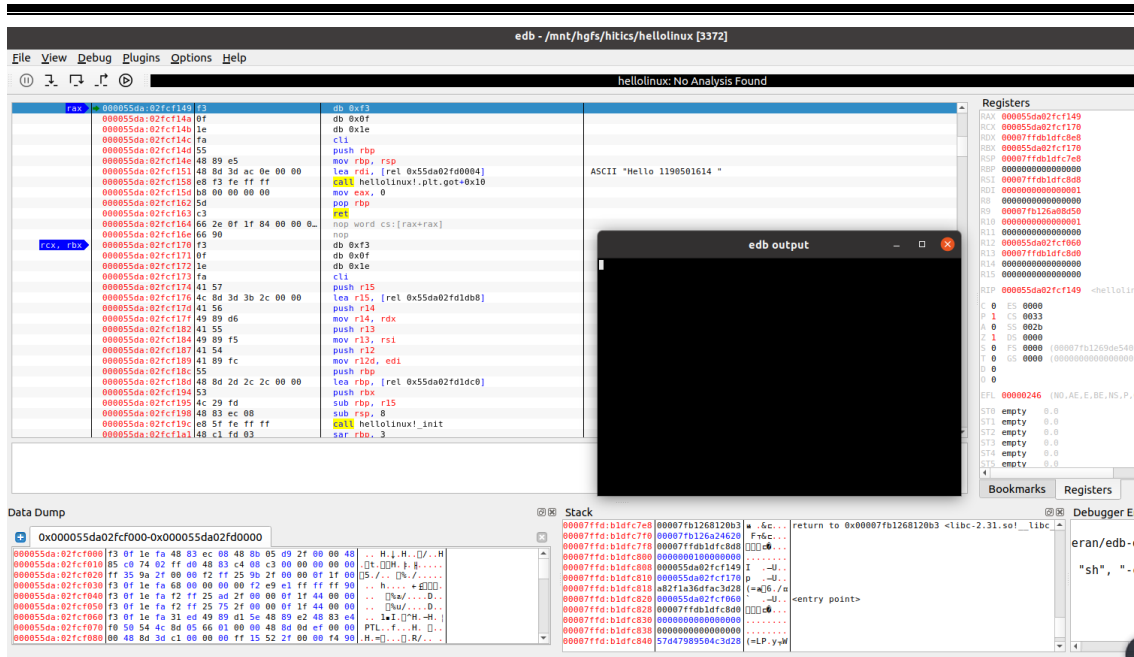


图 2-2 Ubuntu 下 EDB 截图

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.

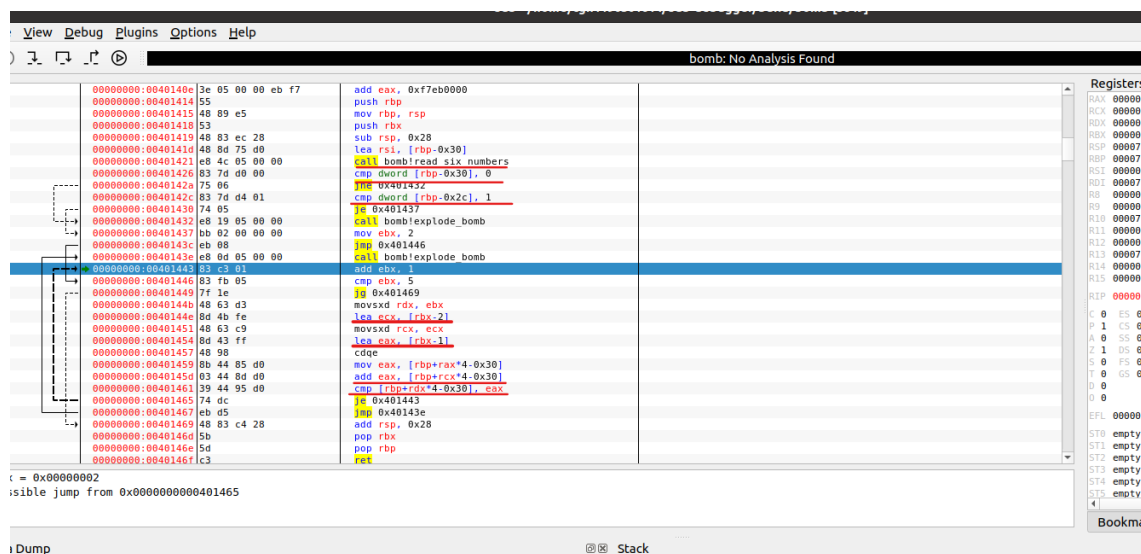
破解过程：随便输入一个字符串，进入 phase_1，找到要匹配的字符串，如下图所示红色标注，即为答案。

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

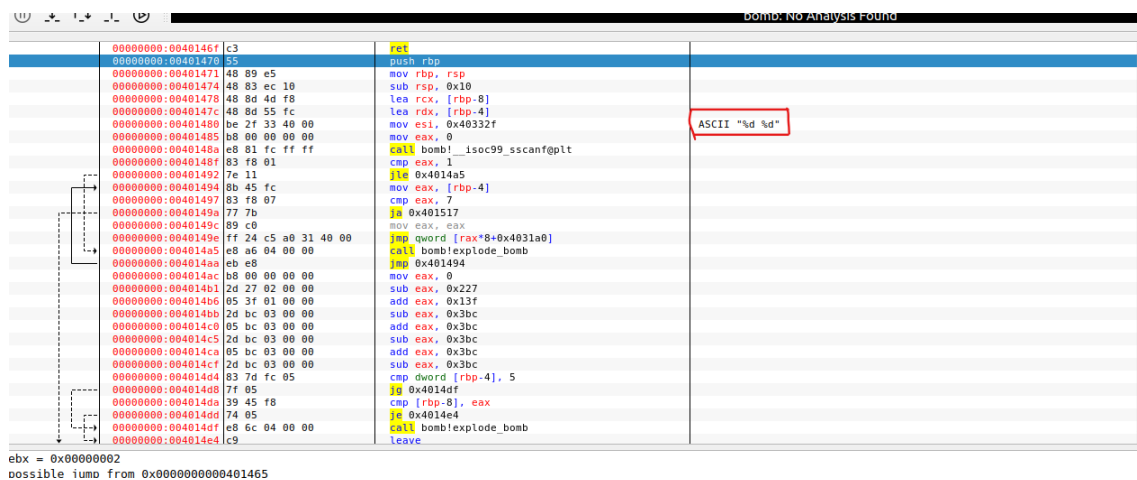


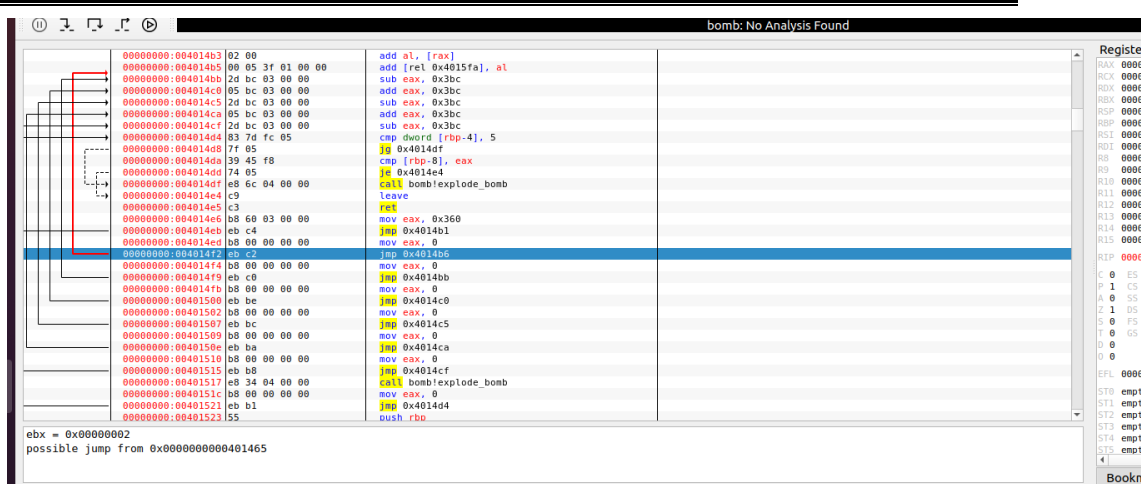
3.3 阶段 3 的破解与分析

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。



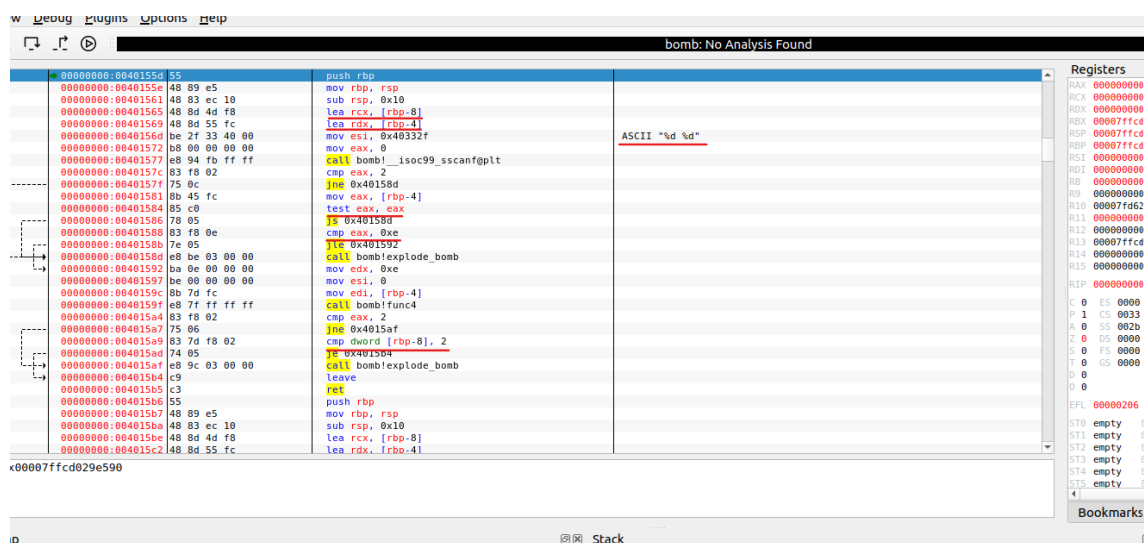


3.4 阶段 4 的破解与分析

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。



00000000:00401523	55	push rbp	
00000000:00401524	48 89 e5	mov rbp, rsp	
00000000:00401527	89 d1	mov ecx, edx	
00000000:00401529	29 f1	sub ecx, esi	
00000000:0040152b	89 c8	mov eax, ecx	
00000000:0040152d	c1 e8 1f	shr eax, 0x1f	
00000000:00401530	01 c8	add ecx, ecx	
00000000:00401532	d1 f8	sar eax, 1	
00000000:00401534	01 f0	add ecx, esi	
00000000:00401536	39 f8	cmp eax, edi	
00000000:00401538	7f 09	jg 0x401543	
00000000:0040153a	7c 13	jl 0x40154f	
00000000:0040153c	b8 00 00 00 00	mov ecx, 0	
00000000:00401541	5d	pop rbp	
00000000:00401542	c3	ret	
00000000:00401543	8d 50 ff	lea edx, [rax-1]	
00000000:00401545	e8 48 ff ff ff	call bomb!func4	
00000000:0040154b	01 c0	add ecx, eax	
00000000:0040154d	eb f2	jmp 0x401541	
00000000:0040154f	8d 70 01	lea esi, [rax+1]	
00000000:00401552	e8 cc ff ff ff	call bomb!func4	
00000000:00401557	8d 44 00 01	lea ecx, [rax+rax+1]	
00000000:0040155b	eb e4	jmp 0x401541	
00000000:0040155d	55	push rbp	
00000000:0040155e	48 89 e5	mov rbp, rsp	
00000000:00401561	48 83 ec 10	sub rsp, 0x10	
00000000:00401565	48 8d 4d f8	lea rcx, [rbp-8]	
00000000:00401569	48 8d 55 fc	lea rdx, [rbp-4]	
00000000:0040156d	be 2f 33 40 00	mov esi, 0x40332f	
00000000:00401572	b8 00 00 00 00	mov ecx, 0	
00000000:00401577	e8 94 fb ff ff	call bomb!_isoc99_sscanf@plt	
00000000:0040157c	83 f8 02	cmp ecx, 2	

ax = 0x00000007

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。

00000000:004015b7	48 89 e5	mov rbp, rsp	
00000000:004015ba	48 83 ec 10	sub rsp, 0x10	
00000000:004015be	48 8d 4d f8	lea rcx, [rbp-8]	
00000000:004015c2	48 8d 55 fc	lea rdx, [rbp-4]	
00000000:004015c6	be 2f 33 40 00	mov esi, 0x40332f	
00000000:004015cb	b8 00 00 00 00	mov ecx, 0	
00000000:004015d0	e8 3b fb ff ff	call bomb!_isoc99_sscanf@plt	
00000000:004015d5	83 f8 01	cmp ecx, 1	
00000000:004015d8	7e 2e	jle 0x401608	
00000000:004015da	8b 45 fc	mov eax, [rbp-4]	
00000000:004015dd	83 e0 0f	and eax, 0xf	
00000000:004015e0	89 45 fc	mov [rbp-4], eax	
00000000:004015e3	b9 00 00 00 00	mov ecx, 0	
00000000:004015e8	ba 00 00 00 00	mov edx, 0	
00000000:004015ed	8b 45 fc	mov eax, [rbp-4]	
00000000:004015f0	83 f8 0f	cmp ecx, 0xf	
00000000:004015f3	74 1a	je 0x40160f	
00000000:004015f5	83 c2 01	add ecx, 1	
00000000:004015f8	48 98	cdqe	
00000000:004015fa	8b 04 85 e0 31 40 00	mov eax, [rax*4+0x4031e0]	
00000000:00401601	89 45 fc	mov [rbp-4], eax	
00000000:00401604	01 c1	add ecx, ecx	
00000000:00401606	eb e5	jmp 0x4015ed	
00000000:00401608	e8 43 03 00 00	call bomb!explode_bomb	
00000000:0040160d	eb cb	jmp 0x4015da	
00000000:0040160f	83 fa 0f	cmp edx, 0xf	
00000000:00401612	75 05	jne 0x401619	
00000000:00401614	39 4d f8	cmp [rbp-8], ecx	
00000000:00401617	74 05	je 0x40161e	
00000000:00401619	e8 32 03 00 00	call bomb!explode_bomb	
00000000:0040161e	c9	leave	
00000000:0040161f	c3	ret	

x00000007

x00000004

Data Dump

0x0000000000403000-0x0000000000404000

00000000:004031c4	00 00 00 00 02 15 40 00 00 00 00 00 09 15 40 00@.....@.
00000000:004031d4	00 00 00 00 10 15 40 00 00 00 00 00 0a 00 00 00@.....
00000000:004031e4	02 00 00 00 0e 00 00 00 07 00 00 00 08 00 00 00@.....
00000000:004031f4	0c 00 00 00 0f 00 00 00 0b 00 00 00 00 00 00 00@.....
00000000:00403204	04 00 00 00 01 00 00 00 0d 00 00 00 03 00 00 00@.....
00000000:00403214	09 00 00 00 06 00 00 00 05 00 00 00 53 6f 20 79So y
00000000:00403224	6f 75 20 74 68 69 6e 6b 20 79 6f 75 20 63 61 6e	ou think you can
00000000:00403234	20 73 74 6f 70 20 74 68 65 20 62 6f 6d 62 20 77	stop the bomb w
00000000:00403244	69 74 68 20 63 74 72 6c 2d 63 2c 20 64 6f 20 79	ith ctrl-c, do y

Assembly View

bomb: No Analysis Found

00000000:004015be	48 0d 4d f8	lea rcx, [rbp-8]	
00000000:004015c2	48 0d 55 fc	lea rdx, [rbp-4]	
00000000:004015c5	b8 2f 33 40 00	mov esi, 0x40332f	ASCII "%d %d"
00000000:004015cb	b8 00 00 00 00	mov eax, 0	
00000000:004015d0	e8 3b fb ff ff	call bomb!_isoc99_sscanf@plt	
00000000:004015d5	83 f8 01	cmp eax, 1	
00000000:004015d8	74 2e	jle 0x401600	
00000000:004015da	8b 45 fc	mov eax, [rbp-4]	
00000000:004015dd	83 e0 0f	and eax, 0xf	
00000000:004015e0	89 45 fc	mov [rbp-4], eax	
00000000:004015e3	b9 00 00 00 00	mov ecx, 0	
00000000:004015e8	ba 00 00 00 00	mov edx, 0	
00000000:004015ed	8b 45 fc	mov eax, [rbp-4]	
00000000:004015f0	83 f8 0f	cmp eax, 0xf	
00000000:004015f3	74 1a	jle 0x40160f	
00000000:004015f5	83 c2 01	add ecx, 1	
00000000:004015f8	48 98	cdqe	
00000000:004015fa	8b 04 85 e0 31 40 00	mov eax, [rax*4+0x4031e0]	
00000000:00401601	89 45 fc	mov [rbp-4], eax	
00000000:00401604	01 c1	add ecx, eax	
00000000:00401606	eb c5	jnp 0x4015ed	
00000000:00401608	e8 43 03 00 00	call bomb!explode_bomb	
00000000:0040160d	eb cb	jnp 0x4015da	
00000000:0040160f	83 fa 0f	cmp edx, 0xf	
00000000:00401612	75 05	jne 0x401619	
00000000:00401614	39 4d f8	cmp [rbp-8], ecx	
00000000:00401617	74 05	jle 0x40161e	
00000000:00401619	e8 32 03 00 00	call bomb!explode_bomb	
00000000:0040161e	c9	leave	
00000000:0040161f	c3	ret	
00000000:00401620	55	push rbp	
00000000:00401621	48 89 e5	mov rbp, rbp	

eax = 0x00000007
edi = 0x00000004

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。

计算机系统实验报告

```

497 00000000 401620: -phase_6:
498 401620: 55          push  %rbp
499 401621: 48 89 e5    mov   %rsp,%rbp
500 401624: 41 55      push  %r13
501 401626: 41 54      push  %r12
502 401628: 53         push  %rbx
503 401629: 48 83 ec 58 sub   %0x58,%rsp
504 40162d: 48 86 75 c0 lea   -0x40(%rbp),%rsi
505 401631: e8 3c 03 00 00 callq 401972 <read_six_numbers>
506 401636: 41 bc 00 00 00 00 mov   %0x0,%r12d
507 401637: e8 47 03 00 00 jmp   401667 <-phase_6+0x47>
508 40163e: e8 0d 03 00 00 callq 401950 <explode_bomb>
509 401643: eb 37      jmp   40167c <-phase_6+0x5c>
510 401645: e8 00 03 00 00 callq 401950 <explode_bomb>
511 40164a: 83 c3 01    add   %0x1,%ebx
512 40164d: 83 fb 05    cmp   %0x5,%ebx
513 401650: 7f 12      jg    401664 <-phase_6+0x44>
514 401652: 49 03 c4    movslq %r12d,%rax
515 401655: 48 03 d3    movslq %ebx,%rdx
516 401658: 8b 7c 95 c0 mov   -0x40(%rbp,%rdx,4),%edi
517 40165c: 39 7c 85 c0 cmp   %edi,-0x40(%rbp,%rax,4)
518 401660: 75 e8      jne   40164a <-phase_6+0x2a>
519 401662: eb e1      jmp   401645 <-phase_6+0x25>
520 401664: 45 89 ec    mov   %r13d,%r12d
521 401667: 41 83 fc 05 cmp   %0x5,%r12d
522 40166b: 7f 19      jg    401686 <-phase_6+0x66>
523 40166d: 49 03 c4    movslq %r12d,%rax
524 401670: 8b 44 85 c0 mov   -0x40(%rbp,%rax,4),%eax
525 401674: 83 e8 01    sub   %0x1,%eax
526 401677: 83 fb 05    cmp   %0x5,%eax
527 40167a: 77 c2      ja    40163e <-phase_6+0x1e>
528 40167c: 45 8d 6c 24 01 lea   0x1(%r12),%r13d
529 401681: 44 89 eb    mov   %r13d,%ebx
530 401684: eb c7      jmp   40164d <-phase_6+0x2d>
531 401686: b8 00 00 00 00 mov   %0x0,%eax
532 40168b: eb 13      jmp   4016a0 <-phase_6+0x80>
533 40168d: 48 03 c8    movslq %eax,%rcx
534 401690: ba 07 00 00 00 mov   %0x7,%edx
535 401695: 2b 54 8d c0 sub   -0x40(%rbp,%rcx,4),%edx
536 401699: 89 54 8d c0 mov   %edx,-0x40(%rbp,%rcx,4)
537 40169d: 83 c0 01    add   %0x1,%eax
538 4016a0: 83 fb 05    cmp   %0x5,%eax
539 4016a3: 7e e8      jle   40168d <-phase_6+0x6d>
540 4016a5: b8 00 00 00 00 mov   %0x0,%esi
541 4016aa: eb 18      jmp   4016c4 <-phase_6+0xa4>
542 4016ac: 48 8b 52 08 mov   0x8(%rdx),%rdx
543 4016b0: 83 c0 01    add   %0x1,%eax
544 4016b3: 48 03 ce    movslq %esi,%rcx
545 4016b6: 39 44 8d c0 cmp   %eax,-0x40(%rbp,%rcx,4)
546 4016ba: 7f f0      jg    4016ac <-phase_6+0x8c>
547 4016bc: 48 89 54 cd 90 mov   %rdx,-0x70(%rbp,%rcx,8)
548 4016c1: 83 c0 01    add   %0x1,%esi
549 4016c4: 83 fe 05    cmp   %0x5,%esi
550 4016c7: 7f 0c      jg    4016d5 <-phase_6+0xb5>
551 4016c9: b8 01 00 00 00 mov   %0x1,%eax
552 4016ce: ba d0 52 40 00 mov   %0x4052d0,%edx
553 4016d3: eb de      jmp   4016b3 <-phase_6+0x93>
554 4016d5: 48 8b 5d 90 mov   -0x70(%rbp),%rbx
555 4016d9: 48 89 49    mov   %rbx,%rcx
556 4016dc: b8 01 00 00 00 mov   %0x1,%eax
557 4016e1: eb 12      jmp   4016f5 <-phase_6+0xd5>
558 4016e3: 48 03 d9    movslq %eax,%rdx
559 4016e6: 48 8b 54 d5 90 mov   -0x70(%rbp,%rdx,8),%rdx
560 4016eb: 48 89 51 08 mov   %rdx,0x8(%rcx)
561 4016ef: 83 c0 01    add   %0x1,%eax
562 4016f2: 48 89 d1    mov   %rdx,%rcx
563 4016f5: 83 fb 05    cmp   %0x5,%eax
564 4016f8: 7e e9      jle   4016e3 <-phase_6+0xc3>
565 4016fa: 48 c7 41 08 00 00 00 movq   %0x8,%rcx
566 401701: 00        nop
567 401702: 41 bc 00 00 00 00 mov   %0x0,%r12d

```

```

525 401674: 83 c0 01    sub   %0x1,%eax
526 401677: 83 fb 05    cmp   %0x5,%eax
527 40167a: 77 c2      ja    40163e <-phase_6+0x1e>
528 40167c: 45 8d 6c 24 01 lea   0x1(%r12),%r13d
529 401681: 44 89 eb    mov   %r13d,%ebx
530 401684: eb c7      jmp   40164d <-phase_6+0x2d>
531 401686: b8 00 00 00 00 mov   %0x0,%eax
532 40168b: eb 13      jmp   4016a0 <-phase_6+0x80>
533 40168d: 48 03 c8    movslq %eax,%rcx
534 401690: ba 07 00 00 00 mov   %0x7,%edx
535 401695: 2b 54 8d c0 sub   -0x40(%rbp,%rcx,4),%edx
536 401699: 89 54 8d c0 mov   %edx,-0x40(%rbp,%rcx,4)
537 40169d: 83 c0 01    add   %0x1,%eax
538 4016a0: 83 fb 05    cmp   %0x5,%eax
539 4016a3: 7e e8      jle   40168d <-phase_6+0x6d>
540 4016a5: b8 00 00 00 00 mov   %0x0,%esi
541 4016aa: eb 18      jmp   4016c4 <-phase_6+0xa4>
542 4016ac: 48 8b 52 08 mov   0x8(%rdx),%rdx
543 4016b0: 83 c0 01    add   %0x1,%eax
544 4016b3: 48 03 ce    movslq %esi,%rcx
545 4016b6: 39 44 8d c0 cmp   %eax,-0x40(%rbp,%rcx,4)
546 4016ba: 7f f0      jg    4016ac <-phase_6+0x8c>
547 4016bc: 48 89 54 cd 90 mov   %rdx,-0x70(%rbp,%rcx,8)
548 4016c1: 83 c0 01    add   %0x1,%esi
549 4016c4: 83 fe 05    cmp   %0x5,%esi
550 4016c7: 7f 0c      jg    4016d5 <-phase_6+0xb5>
551 4016c9: b8 01 00 00 00 mov   %0x1,%eax
552 4016ce: ba d0 52 40 00 mov   %0x4052d0,%edx
553 4016d3: eb de      jmp   4016b3 <-phase_6+0x93>
554 4016d5: 48 8b 5d 90 mov   -0x70(%rbp),%rbx
555 4016d9: 48 89 49    mov   %rbx,%rcx
556 4016dc: b8 01 00 00 00 mov   %0x1,%eax
557 4016e1: eb 12      jmp   4016f5 <-phase_6+0xd5>
558 4016e3: 48 03 d9    movslq %eax,%rdx
559 4016e6: 48 8b 54 d5 90 mov   -0x70(%rbp,%rdx,8),%rdx
560 4016eb: 48 89 51 08 mov   %rdx,0x8(%rcx)
561 4016ef: 83 c0 01    add   %0x1,%eax
562 4016f2: 48 89 d1    mov   %rdx,%rcx
563 4016f5: 83 fb 05    cmp   %0x5,%eax
564 4016f8: 7e e9      jle   4016e3 <-phase_6+0xc3>
565 4016fa: 48 c7 41 08 00 00 00 movq   %0x8,%rcx
566 401701: 00        nop
567 401702: 41 bc 00 00 00 00 mov   %0x0,%r12d

```

```

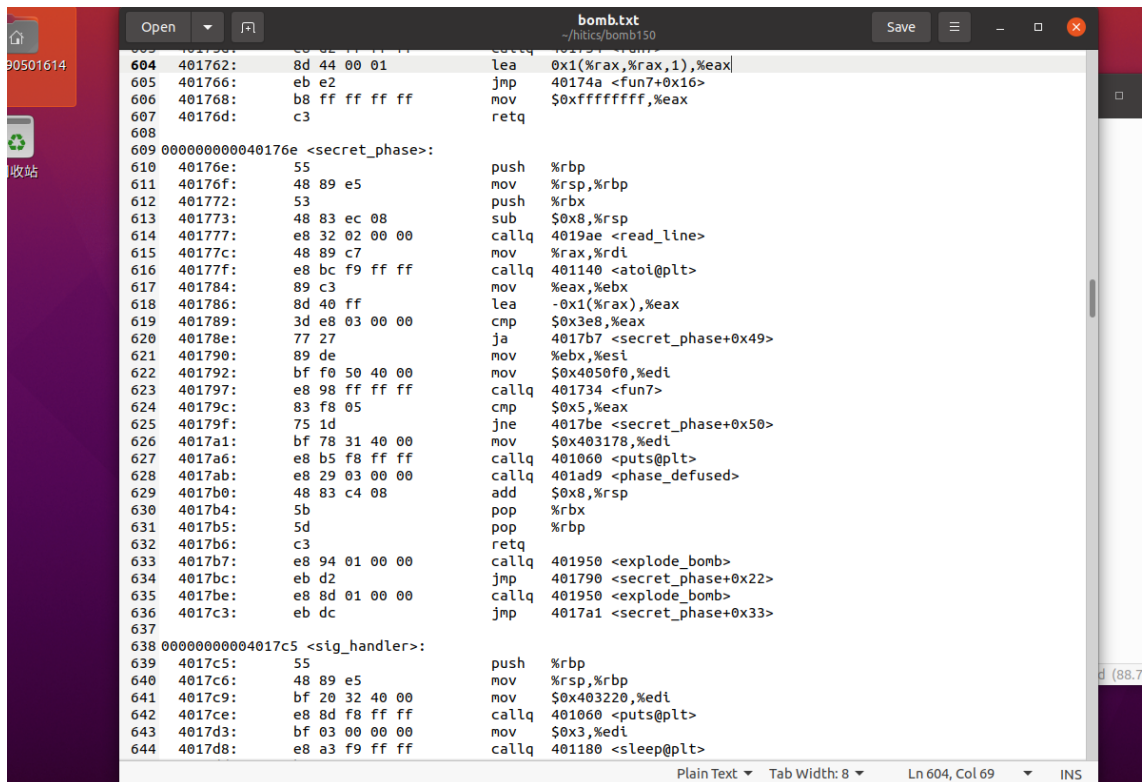
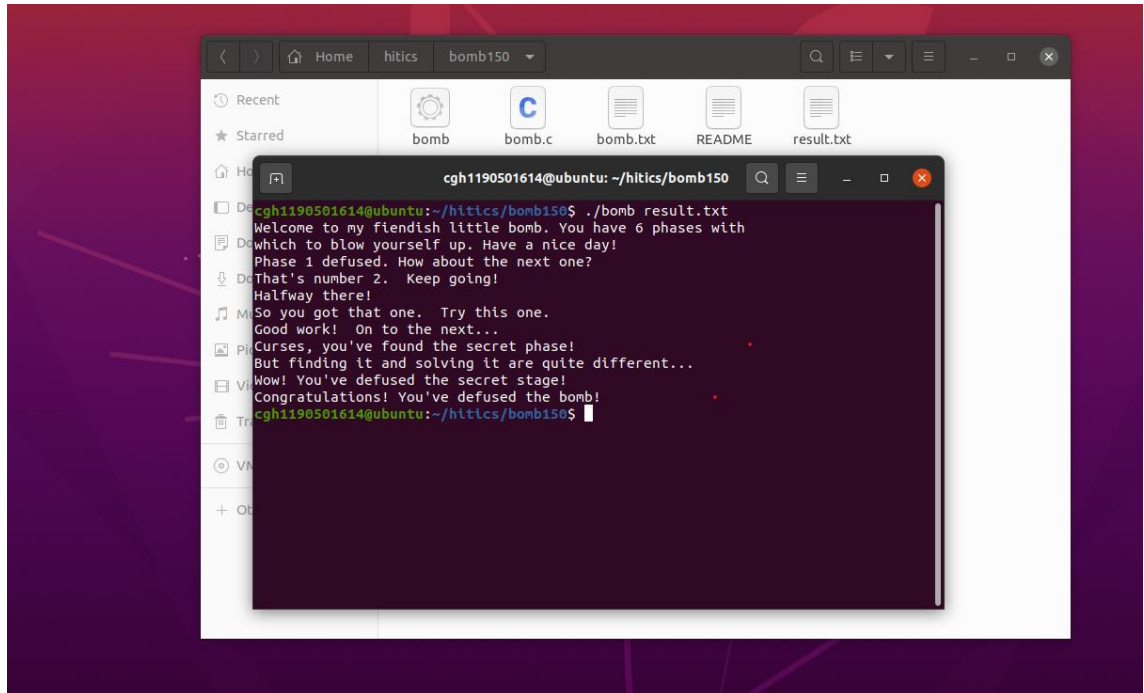
cgh1190501614@ubuntu: ~/hitcs/bomb150
(gdb) b phase_6
Breakpoint 2 at 0x401620: file phases.c, line 284.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) x/3x 0x4052d0
0x4052d0 <node1>: 0x00000118 0x00000001 0x004052e0
(gdb) 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：




```

4月 24 17:42
bomb.txt
~/hitics/bomb150
578 401727: eb e1 jmp 40170a <phase_6+0xea>
579 401729: 48 83 c4 58 add $0x58,%rsp
580 40172d: 5b pop %rbx
581 40172e: 41 5c pop %r12
582 401730: 41 5d pop %r13
583 401732: 5d pop %rbp
584 401733: c3 retq
585
586 0000000000401734 <fun7>:
587 401734: 48 85 ff test %rdi,%rdi
588 401737: 74 2f je 401768 <fun7+0x34>
589 401739: 55 push %rbp
590 40173a: 48 89 e5 mov %rsp,%rbp
591 40173d: 8b 07 mov (%rdi),%eax
592 40173f: 39 f0 cmp %esi,%eax
593 401741: 7f 09 jg 40174c <fun7+0x18>
594 401743: 75 14 jne 401759 <fun7+0x25>
595 401745: b8 00 00 00 00 mov $0x0,%eax
596 40174a: 5d pop %rbp
597 40174b: c3 retq
598 40174c: 48 8b 7f 08 mov 0x8(%rdi),%rdi
599 401750: e8 df ff ff ff callq 401734 <fun7>
600 401755: 01 c0 add %eax,%eax
601 401757: eb f1 jmp 40174a <fun7+0x16>
602 401759: 48 8b 7f 10 mov 0x10(%rdi),%rdi
603 40175d: e8 d2 ff ff ff callq 401734 <fun7>
604 401762: 8d 44 00 01 lea 0x1(%rax,%rax,1),%eax
605 401766: eb e2 jmp 40174a <fun7+0x16>
606 401768: b8 ff ff ff ff mov $0xffffffff,%eax
607 40176d: c3 retq
608
609 000000000040176e <secret_phase>:
610 40176e: 55 push %rbp
611 40176f: 48 89 e5 mov %rsp,%rbp
612 401772: 53 push %rbx
613 401773: 48 83 ec 08 sub $0x8,%rsp
614 401777: e8 32 02 00 00 callq 4019ae <read_line>
615 40177c: 48 89 c7 mov %rax,%rdi
616 40177f: e8 bc f9 ff ff callq 401140 <atoi@plt>
617 401784: 89 c3 mov %eax,%ebx
618 401786: 8d 40 ff lea -0x1(%rax),%eax
619 401789: 3d e8 83 00 00 cmp $0x3e8,%eax

```

```

cgh1190501614@ubuntu: ~/hitics/bomb150
(gdb) x /100wx 0x4050f0
0x4050f0 <n1>: 0x00000024 0x00000000 0x00405110 0x00000000
0x405100 <n1+16>: 0x00405130 0x00000000 0x00000000 0x00000000
0x405110 <n21>: 0x00000008 0x00000000 0x00405190 0x00000000
0x405120 <n21+16>: 0x00405150 0x00000000 0x00000000 0x00000000
0x405130 <n22>: 0x00000032 0x00000000 0x00405170 0x00000000
0x405140 <n22+16>: 0x004051b0 0x00000000 0x00000000 0x00000000
0x405150 <n32>: 0x00000016 0x00000000 0x00405270 0x00000000
0x405160 <n32+16>: 0x00405230 0x00000000 0x00000000 0x00000000
0x405170 <n33>: 0x0000002d 0x00000000 0x004051d0 0x00000000
0x405180 <n33+16>: 0x00405290 0x00000000 0x00000000 0x00000000
0x405190 <n31>: 0x00000006 0x00000000 0x004051f0 0x00000000
0x4051a0 <n31+16>: 0x00405250 0x00000000 0x00000000 0x00000000
0x4051b0 <n34>: 0x0000006b 0x00000000 0x00405210 0x00000000
0x4051c0 <n34+16>: 0x004052b0 0x00000000 0x00000000 0x00000000
0x4051d0 <n45>: 0x00000028 0x00000000 0x00000000 0x00000000
0x4051e0 <n45+16>: 0x00000000 0x00000000 0x00000000 0x00000000
<RET> for more, q to quit, c to continue without paging--RET
00
0x4051f0 <n41>: 0x00000001 0x00000000 0x00000000 0x00000000
0x405200 <n41+16>: 0x00000000 0x00000000 0x00000000 0x00000000
0x405210 <n47>: 0x00000063 0x00000000 0x00000000 0x00000000
0x405220 <n47+16>: 0x00000000 0x00000000 0x00000000 0x00000000
0x405230 <n44>: 0x00000023 0x00000000 0x00000000 0x00000000
0x405240 <n44+16>: 0x00000000 0x00000000 0x00000000 0x00000000
0x405250 <n42>: 0x00000007 0x00000000 0x00000000 0x00000000
0x405260 <n42+16>: 0x00000000 0x00000000 0x00000000 0x00000000
0x405270 <n43>: 0x00000014 0x00000000 0x00000000 0x00000000
(gdb) x /1xg 0x405290
0x405290 <n46>: 0x000000000000002f
(gdb)

```

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