

# 哈尔滨工业大学

# 实验报告

## 实验（三）

题 目 Binary Bomb

二进制炸弹

专 业 计算机科学与技术

学 号 1180300315

班 级 1836101

学 生 周牧云

指 导 教 师 史先俊

实 验 地 点 \_\_\_\_\_

实 验 日 期 2019/10/24

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） .....	6 -
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.....	8 -
3.3 阶段 3 的破解与分析.....	9 -
3.4 阶段 4 的破解与分析.....	10 -
3.5 阶段 5 的破解与分析.....	11 -
3.6 阶段 6 的破解与分析.....	12 -
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。

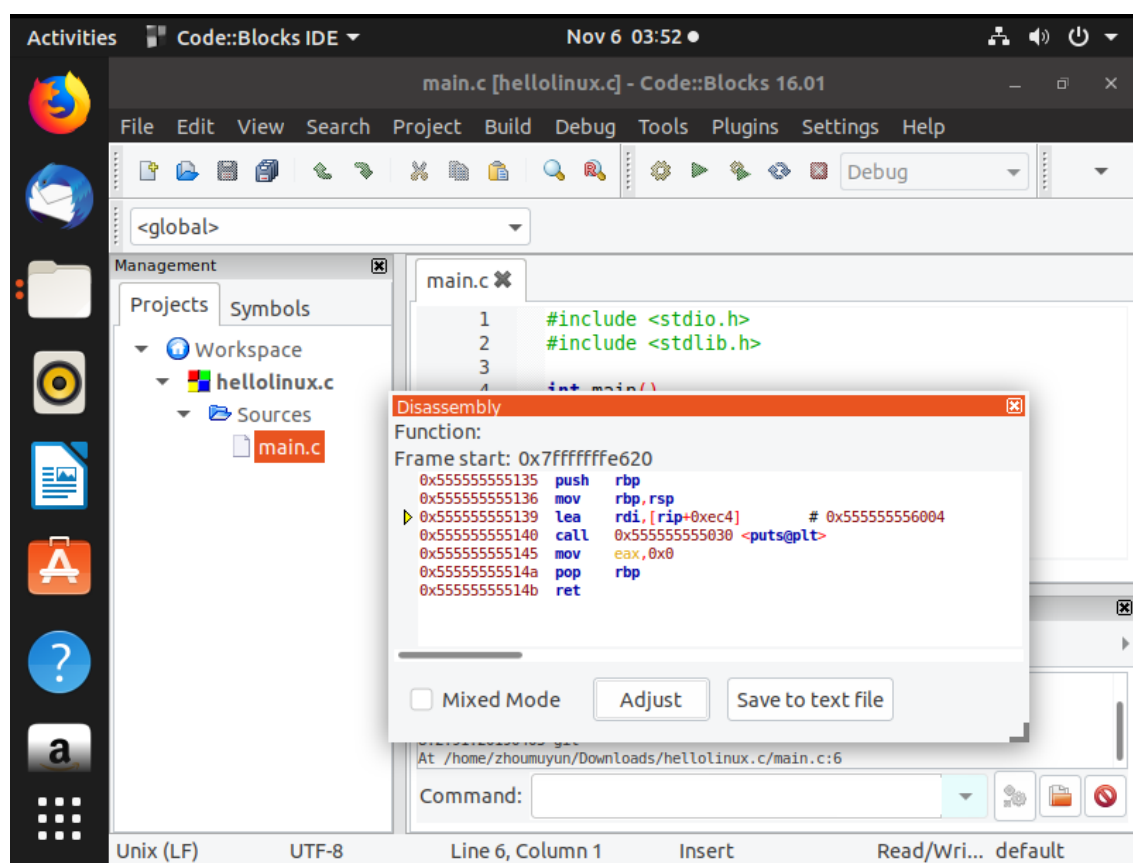


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

## 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

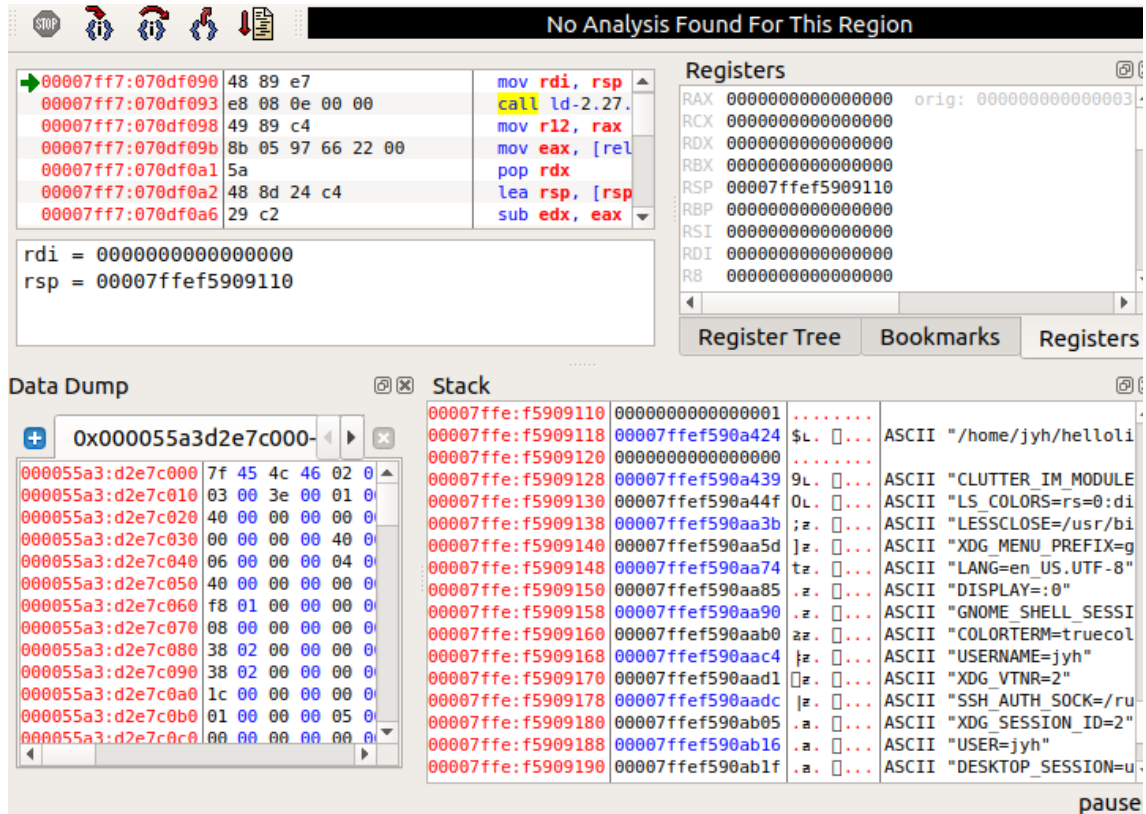


图 2-2 Ubuntu 下 EDB 截图

## 第 3 章 各阶段炸弹破解与分析

每阶段 15 分（密码 10 分，分析 5 分），总分不超过 80 分

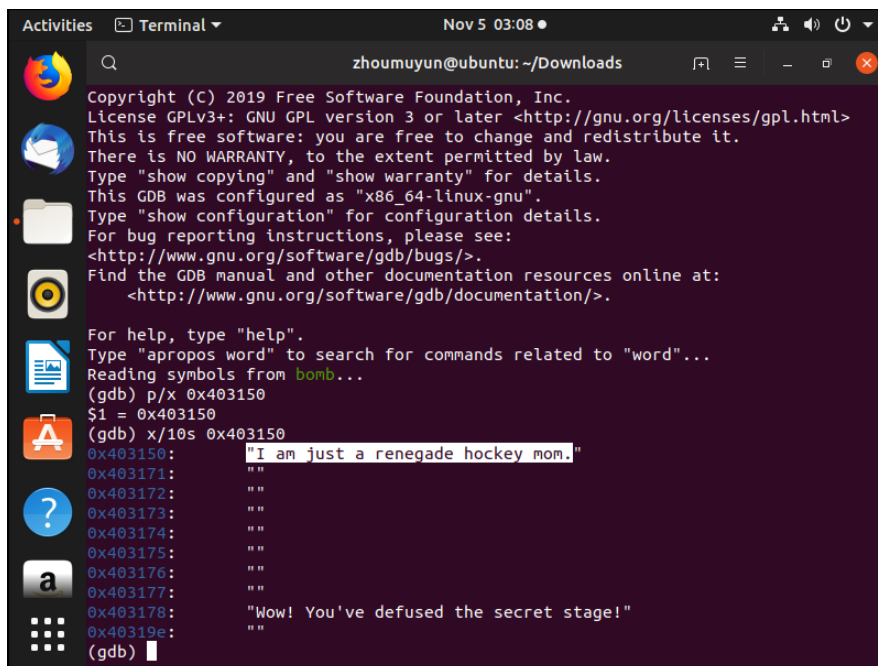
### 3.1 阶段 1 的破解与分析

密码如下：I am just a renegade hockey mom.

破解过程：

```
00000000004013f5 <phase_1>:
4013f5: 55                push    %rbp
4013f6: 48 89 e5          mov     %rsp,%rbp
4013f9: be 50 31 40 00    mov     $0x403150,%esi
4013fe: e8 19 04 00 00    callq  40181c <strings_not_equal>
401403: 85 c0            test    %eax,%eax
401405: 75 02            jne     401409 <phase_1+0x14>
401407: 5d              pop     %rbp
401408: c3              retq
401409: e8 0d 05 00 00    callq  40191b <explode_bomb>
40140e: eb f7            jmp     401407 <phase_1+0x12>
```

由反汇编得知密码就在地址 0x403150 中，用 gdb 查案该地址中的内容即可得知密码。



```
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) p/x 0x403150
$1 = 0x403150
(gdb) x/10s 0x403150
0x403150: "I am just a renegade hockey mom."
0x403171: ""
0x403172: ""
0x403173: ""
0x403174: ""
0x403175: ""
0x403176: ""
0x403177: ""
0x403178: "Wow! You've defused the secret stage!"
0x40319e: ""
(gdb)
```

## 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：

```

0000000000401410 <phase_2>:
401410: 55                push    %rbp
401411: 48 89 e5          mov     %rsp,%rbp
401414: 53                push    %rbx
401415: 48 83 ec 28       sub     $0x28,%rsp
401419: 48 8d 75 d0       lea     -0x30(%rbp),%rsi
40141d: e8 1b 05 00 00   callq  40193d <read_six_numbers>
401422: 83 7d d0 00       cmpl    $0x0,-0x30(%rbp)           //第一位和0比较
401426: 75 06            jne     40142e <phase_2+0x1e>      //不等于就爆炸
401428: 83 7d d4 01       cmpl    $0x1,-0x2c(%rbp)           //第二位与1进行比较
40142c: 74 05            je      401433 <phase_2+0x23>      //相等继续
40142e: e8 e8 04 00 00   callq  40191b <explode_bomb>       //不等于就爆炸
401433: bb 02 00 00 00   mov     $0x2,%ebx                  //ebx = 2 计数
401438: eb 08            jmp     401442 <phase_2+0x32>       //进入循环
40143a: e8 dc 04 00 00   callq  40191b <explode_bomb>       //没进循环就爆炸
40143f: 83 c3 01         add     $0x1,%ebx                  // ebx += 1
401442: 83 fb 05         cmp     $0x5,%ebx                  // 若ebx > 5
401445: 7f 1e            jg      401465 <phase_2+0x55>       // 结束循环，共循环四次
401447: 48 63 d3         movslq  %ebx,%rdx                  // 把ebx赋给rdx
40144a: 8d 4b fe         lea     -0x2(%rbx),%ecx            //
40144d: 48 63 c9         movslq  %ecx,%rcx                  //
401450: 8d 43 ff         lea     -0x1(%rbx),%eax            //
401453: 48 98            cltq                                     //
401455: 8b 44 85 d0       mov     -0x30(%rbp,%rax,4),%eax    // 把c[i-2]赋给eax
401459: 03 44 8d d0       add     -0x30(%rbp,%rcx,4),%eax    // eax += c[i-1]
40145d: 39 44 95 d0       cmp     %eax,-0x30(%rbp,%rdx,4)    // eax 与c[i]比较
401461: 74 dc            je      40143f <phase_2+0x2f>      // 相等的话就循环
401463: eb d5            jmp     40143a <phase_2+0x2a>      // 爆炸
401465: 48 83 c4 28       add     $0x28,%rsp
401469: 5b                pop     %rbx
40146a: 5d                pop     %rbp
40146b: c3                retq

```

由反汇编得知密码为六位数字，第一位是 0，第二位是 1，后面的四位每一位的值为它前面两位的和。



### 3.3 阶段3的破解与分析

密码如下：0 166 或 1 163 或 2 721 或 3 888 或 4 776 或 5 829 或 6 830 或 7 175

破解过程：

```

000000000040146c <phase_3>:
40146c: 55                push    %rbp
40146d: 48 89 e5          mov     %rsp,%rbp
401470: 48 83 ec 10        sub     $0x10,%rsp
401474: 48 8d 4d f8        lea     -0x8(%rbp),%rcx
401478: 48 8d 55 fc        lea     -0x4(%rbp),%rdx
40147c: be 2f 33 40 00     mov     $0x40332f,%esi
401481: b8 00 00 00 00     mov     $0x0,%eax                //eax赋值0
401486: e8 95 fc ff ff     callq  401120 <__isoc99_sscanf@plt>
40148b: 83 f8 01          cmp     $0x1,%eax                //eax与1比较
40148e: 7e 10             jle     4014a0 <phase_3+0x34>      //若小于, 则爆炸
401490: 83 7d fc 07        cmpl    $0x7,-0x4(%rbp)          //第一位与7比较
401494: 77 42             ja      4014d8 <phase_3+0x6c>      //大于则爆炸
401496: 8b 45 fc          mov     -0x4(%rbp),%eax          //小于则把它赋给eax
401499: ff 24 c5 a0 31 40 00 jmpq     *0x4031a0(,%rax,8)        //跳转到指定位置
4014a0: e8 76 04 00 00     callq  40191b <explode_bomb>      //爆炸指令
4014a5: eb e9             jmp     401490 <phase_3+0x24>      //回到比较
4014a7: b8 a6 00 00 00     mov     $0xa6,%eax              //若eax = 0,  eax赋值166
4014ac: eb 3b             jmp     4014e9 <phase_3+0x7d>      //跳
4014ae: b8 d1 02 00 00     mov     $0x2d1,%eax             //若eax = 2,  eax赋值721
4014b3: eb 34             jmp     4014e9 <phase_3+0x7d>      //跳
4014b5: b8 78 03 00 00     mov     $0x378,%eax             //若eax = 3,  eax赋值888
4014ba: eb 2d             jmp     4014e9 <phase_3+0x7d>      //跳
4014bc: b8 08 03 00 00     mov     $0x308,%eax             //若eax = 4,  eax赋值776
4014c1: eb 26             jmp     4014e9 <phase_3+0x7d>      //跳
4014c3: b8 3d 03 00 00     mov     $0x33d,%eax             //若eax = 5,  eax赋值829
4014c8: eb 1f             jmp     4014e9 <phase_3+0x7d>      //跳
4014ca: b8 3e 03 00 00     mov     $0x33e,%eax             //若eax = 6,  eax赋值830
4014cf: eb 18             jmp     4014e9 <phase_3+0x7d>      //跳
4014d1: b8 af 00 00 00     mov     $0xaf,%eax              //若eax = 7,  eax赋值175
4014d6: eb 11             jmp     4014e9 <phase_3+0x7d>      //跳
4014d8: e8 3e 04 00 00     callq  40191b <explode_bomb>      //爆炸指令
4014dd: b8 00 00 00 00     mov     $0x0,%eax              //eax赋值0
4014e2: eb 05             jmp     4014e9 <phase_3+0x7d>      //跳
4014e4: b8 a3 00 00 00     mov     $0xa3,%eax              //若eax = 1,  eax赋值163
4014e9: 39 45 f8          cmp     %eax,-0x8(%rbp)          //比较第二位与eax
4014ec: 75 02             jne     4014f0 <phase_3+0x84>      //不等于就爆炸
4014ee: c9               leaveq  %eax,%rbp
4014ef: c3               retq
4014f0: e8 26 04 00 00     callq  40191b <explode_bomb>      //爆炸指令
4014f5: eb f7             jmp     4014ee <phase_3+0x82>

```

由反汇编得知程序先读入第一个数，由第一个数进入分支判断第二个数是多少，所以有多个答案

### 3.4 阶段4的破解与分析

密码如下：40 2 或 60 3 或 80 4

破解过程：

```

00000000004014f7 <func4>:
4014f7: 85 ff          test    %edi,%edi                //若edi等于0
4014f9: 7e 3d          jle     401538 <func4+0x41>      //返回0
4014fb: 83 ff 01       cmp     $0x1,%edi               //若edi等于1
4014fe: 74 3e          je      40153e <func4+0x47>      //返回第二个数
401500: 55             push    %rbp                    //入栈
401501: 48 89 e5       mov     %rsp,%rbp               //rbp赋值rsp
401504: 41 55          push    %r13                    //入栈
401506: 41 54          push    %r12                    //入栈
401508: 53             push    %rbx                    //入栈
401509: 48 83 ec 08     sub     $0x8,%rsp               //扩展栈
40150d: 89 f3          mov     %esi,%ebx               //ebx赋值esi (初始为第二个数)
40150f: 41 89 fc       mov     %edi,%r12d              //r12赋值edi (初始为6)
401512: 8d 7f ff       lea     -0x1(%rdi),%edi          //edi-1
401515: e8 dd ff ff ff callq   4014f7 <func4>           //递归
40151a: 44 8d 2c 18     lea     (%rax,%rbx,1),%r13d      //r13赋值 (rax + rbx)
40151e: 41 8d 7c 24 fe  lea     -0x2(%r12),%edi          //edi = r12 - 2
401523: 89 de          mov     %ebx,%esi               //esi赋值ebx
401525: e8 cd ff ff ff callq   4014f7 <func4>           //递归
40152a: 44 01 e8       add     %r13d,%eax              //eax += r13
40152d: 48 83 c4 08     add     $0x8,%rsp               //出栈
401531: 5b             pop     %rbx                    //出栈
401532: 41 5c          pop     %r12                    //出栈
401534: 41 5d          pop     %r13                    //出栈
401536: 5d             pop     %rbp                    //出栈
401537: c3             retq                                |
401538: b8 00 00 00 00 mov     $0x0,%eax               //eax赋值0
40153d: c3             retq                                |
40153e: 89 f0          mov     %esi,%eax               //eax赋值esi
401540: c3             retq                                |

0000000000401541 <phase_4>:
401541: 55             push    %rbp
401542: 48 89 e5       mov     %rsp,%rbp
401545: 48 83 ec 10     sub     $0x10,%rsp
401549: 48 8d 4d fc     lea     -0x4(%rbp),%rcx
40154d: 48 8d 55 f8     lea     -0x8(%rbp),%rdx
401551: b6 2f 33 40 00 mov     $0x40332f,%esi
401556: b8 00 00 00 00 mov     $0x0,%eax               //eax赋值0
40155b: e8 c0 fb ff ff callq   401120 <__isoc99_sscanf@plt> //读入
401560: 83 f8 02       cmp     $0x2,%eax               //将eax与2比较
401563: 75 0d          jne     401572 <phase_4+0x31>     //不等于就爆炸
401565: 8b 45 fc       mov     -0x4(%rbp),%eax         //将第一个数赋值给eax
401568: 83 f8 01       cmp     $0x1,%eax               //比较eax与1
40156b: 7e 05          jle     401572 <phase_4+0x31>     //小于等于就爆炸
40156d: 83 f8 04       cmp     $0x4,%eax               //比较eax与4
401570: 7e 05          jle     401577 <phase_4+0x36>     //小于等于就继续
401572: e8 a4 03 00 00 callq   40191b <explode_bomb>      //否则爆炸 //第二个数是2或3 4
401577: 8b 75 fc       mov     -0x4(%rbp),%esi         //esi赋值第二个数
40157a: bf 06 00 00 00 mov     $0x6,%edi               //edi赋值6
40157f: e8 73 ff ff ff callq   4014f7 <func4>           //调用函数
401584: 39 45 f8       cmp     %eax,-0x8(%rbp)         //比较eax与第一个数
401587: 75 02          jne     40158b <phase_4+0x4a>     //不等于就爆炸
401589: c9             leaveq  %r13,%rbp
40158a: c3             retq
40158b: e8 8b 03 00 00 callq   40191b <explode_bomb>      //爆炸指令
401590: eb f7          jmp     401589 <phase_4+0x48>

```

由反汇编得密码为两个数，第二个数大于 1 小于等于 4，第一个数由递归算得是第二个数的 20 倍。

### 3.5 阶段5的破解与分析

密码如下：5 115

破解过程：

```

0000000000401592 <phase_5>:
401592: 55                push    %rbp
401593: 48 89 e5          mov     %rsp,%rbp
401596: 48 83 ec 10        sub     $0x10,%rsp
40159a: 48 8d 4d f8        lea     -0x8(%rbp),%rcx
40159e: 48 8d 55 fc        lea     -0x4(%rbp),%rdx
4015a2: be 2f 33 40 00     mov     $0x40332f,%esi          //esi初始化
4015a7: b8 00 00 00 00     mov     $0x0,%eax              //eax初始化
4015ac: e8 6f fb ff ff     callq   401120 <__isoc99_sscanf@plt> //读入函数
4015b1: 83 f8 01          cmp     $0x1,%eax              //如果eax <= 1
4015b4: 7e 2e            jle     4015e4 <phase_5+0x52>    //就爆炸
4015b6: 8b 45 fc          mov     -0x4(%rbp),%eax        //eax赋值第一个数 //
4015b9: 83 e0 0f          and     $0xf,%eax              //eax和1111进行操作 //
4015bc: 89 45 fc          mov     %eax,-0x4(%rbp)        //第一个数赋值eax //只取第一个数后四位
4015bf: b9 00 00 00 00     mov     $0x0,%ecx              //ecx赋值0
4015c4: ba 00 00 00 00     mov     $0x0,%edx              //edx赋值0
4015c9: 8b 45 fc          mov     -0x4(%rbp),%eax        //eax赋值第一个数
4015cc: 83 f8 0f          cmp     $0xf,%eax              //比较eax与1111
4015cf: 74 1a            je      4015eb <phase_5+0x59>    //相等跳转,此时若edx等于1111, ecx等于第二个数,则成功
4015d1: 83 c2 01          add     $0x1,%edx              //否则edx += 1
4015d4: 48 98            cltq
4015d6: 8b 04 85 e0 31 40 00 mov     0x4031e0(,%rax,4),%eax //eax赋值 (0x4031e0 + 4*%rax) 中的值
4015dd: 89 45 fc          mov     %eax,-0x4(%rbp)        //第一个数赋值eax
4015e0: 01 c1            add     %eax,%ecx              //ecx += eax
4015e2: eb e5            jmp     4015c9 <phase_5+0x37>    //跳转回去
4015e4: e8 32 03 00 00     callq   40191b <explode_bomb>    //爆炸指令
4015e9: eb cb            jmp     4015b6 <phase_5+0x24>    //跳转
4015eb: 83 fa 0f          cmp     $0xf,%edx              //比较edx与1111
4015ee: 75 05            jne     4015f5 <phase_5+0x63>    //不等于就爆炸 //从1到1111共取出了15个数
4015f0: 39 4d f8          cmp     %ecx,-0x8(%rbp)        //比较ecx与第二个数
4015f3: 74 05            je      4015fa <phase_5+0x68>    //相等就结束
4015f5: e8 21 03 00 00     callq   40191b <explode_bomb>    //否则爆炸
4015fa: c9              leaveq
4015fb: c3              retq

```

```

(gdb) x/80x 0x4031e0
0x4031e0 <array.3511>: 0x0a 0x00 0x00 0x00 0x02 0x00 0x00 0
x00
0x4031e8 <array.3511+8>: 0x0e 0x00 0x00 0x00 0x00 0x07 0x00 0
x00 0x00
0x4031f0 <array.3511+16>: 0x08 0x00 0x00 0x00 0x00 0x0c 0x00 0
x00 0x00
0x4031f8 <array.3511+24>: 0x0f 0x00 0x00 0x00 0x00 0x0b 0x00 0
x00 0x00
0x403200 <array.3511+32>: 0x00 0x00 0x00 0x00 0x00 0x04 0x00 0
x00 0x00
0x403208 <array.3511+40>: 0x01 0x00 0x00 0x00 0x00 0x0d 0x00 0
x00 0x00
0x403210 <array.3511+48>: 0x03 0x00 0x00 0x00 0x00 0x09 0x00 0
x00 0x00
0x403218 <array.3511+56>: 0x06 0x00 0x00 0x00 0x00 0x05 0x00 0
x00 0x00

```

此题密码共两个数，第一个数用来寻址，第二个数是所有寻址地址的和。每次寻址后，找到的数加给 ecx（初值为 0），edx += 1（edx 初值为 0），找到的数继续用来寻址。由终止条件 edx = 15，eax = 15 得，共寻址 15 次，且最后一个找到的数必须是 15。可倒推 15 次得，第一次寻址的值为 5，即密码第一位是 5，第二位可由加法算得为 115。

### 3.6 阶段6的破解与分析

密码如下：4 6 2 3 1 5

破解过程：

```

0000000004015fc <phase_6>:
4015fc: 55          push  %rbp
4015fd: 48 89 e5    mov   %rsp,%rbp
401600: 41 55       push  %r13
401602: 41 54       push  %r12
401604: 53         push  %rbx
401605: 48 83 ec 58 sub   $0x58,%rsp
401609: 48 8d 75 c0 lea    -0x40(%rbp),%rsi          //rsi赋第一个数
40160d: e8 2b 03 00 00 callq  40193d <read_six_numbers> //读取六个数
401612: 41 bc 00 00 00 00 mov   $0x0,%r12d              //r12 = 0
401618: eb 29       jmp    401643 <phase_6+0x47>    //跳到401643
40161a: e8 fc 02 00 00 callq  40191b <explode_bomb>    //爆炸指令
40161f: eb 37       jmp    401658 <phase_6+0x5c>    //跳到401658
401621: e8 f5 02 00 00 callq  40191b <explode_bomb>    //爆炸指令
401626: 83 c3 01    add   $0x1,%ebx              //ebx += 1
401629: 83 fb 05    cmp   $0x5,%ebx             //比较ebx与5
40162c: 7f 12       jg     401640 <phase_6+0x44>    //大于则跳401640
40162e: 49 63 c4    movslq %r12d,%rax           //否则 rax = r12
401631: 48 63 d3    movslq %ebx,%rdx            //rdx = ebx
401634: 8b 7c 95 c0 mov    -0x40(%rbp,%rdx,4),%edi //edi = (rbp - 0x40 + 4 * rdx)
401638: 39 7c 85 c0 cmp    %edi,-0x40(%rbp,%rax,4) //edi 与 (rbp - 0x40 + 4 * rax)比较
40163c: 75 e8       jne    401626 <phase_6+0x2a>    //不等于则跳401626
40163e: eb e1       jmp    401621 <phase_6+0x25>    //否则跳到401621
401640: 45 89 ec    mov   %r13d,%r12d           //r12 = r13
401643: 41 83 fc 05 cmp   $0x5,%r12d            //比较r12与5
401647: 7f 19       jg     401662 <phase_6+0x66>    //若大于则跳401662
401649: 49 63 c4    movslq %r12d,%rax           //否则rax = r12
40164c: 8b 44 85 c0 mov    -0x40(%rbp,%rax,4),%eax //eax = (rbp - 0x40 + 4 * rax)
401650: 83 e8 01    sub   $0x1,%eax             //eax -= 1
401653: 83 f8 05    cmp   $0x5,%eax            //eax 与 5 比较
401656: 77 c2       ja     40161a <phase_6+0x1e>    //若大于则爆炸
401658: 45 8d 6c 24 01 lea    0x1(%r12),%r13d        //r13 = r12 + 1
40165d: 44 89 eb    mov   %r13d,%ebx            //ebx = r13
401660: eb c7       jmp    401629 <phase_6+0x2d>    //跳转401629
401662: be 00 00 00 00 mov   $0x0,%esi              //esi = 0
401667: eb 18       jmp    401681 <phase_6+0x85>    //跳401681
401669: 48 8b 52 08 mov    0x8(%rdx),%rdx        //rdx = rdx + 8
40166d: 83 c0 01    add   $0x1,%eax             //eax += 1
401670: 48 63 ce    movslq %esi,%rcx            //rcx = esi
401673: 39 44 8d c0 cmp    %eax,-0x40(%rbp,%rcx,4) //比较eax与-0x40(%rbp,%rcx,4)
401677: 7f f0       jg     40166a <phase_6+0x6a>    //大于则跳40166a

```

```

Q zhomuyun@ubuntu: ~/Downloads
(gdb) p/x *(0x4052d0)
$1 = 0x314
(gdb) p/x *(0x4052d8)
$2 = 0x4052e0
(gdb) p/x *(0x4052e0)
$3 = 0x206
(gdb) p/x *(0x4052e8)
$4 = 0x4052f0
(gdb) p/x *(0x4052f0)
$5 = 0x240
(gdb) p/x *(0x4052f8)
$6 = 0x405300
(gdb) p/x *(0x405300)
$7 = 0x176
(gdb) p/x *(0x405308)
$8 = 0x405310
(gdb) p/x *(0x405310)
$9 = 0x38e
(gdb) p/x *(0x405318)
$10 = 0x405320
(gdb) p/x *(0x405320)
$11 = 0x1ca
(gdb) p/x *(0x405328)
$12 = 0x0

```

此题密码是内存中的六个数由小到大的顺序。每个数后面跟着下一个数的地址，由此不断寻址可按顺序得到六个数，比较大小则可得密码。

### 3.7 阶段 7 的破解与分析(隐藏阶段)

密码如下：7

破解过程：

```
000000000401aaa <phase_defused>:
401aaa: 83 3d bb 3c 00 00 06    cmpb $0x6,0x3cbb(%rip)      # 40576c <num_input_strings>
401ab1: 74 01                  je 401ab4 <phase_defused+0xa>
401ab3: c3                    retq
401ab4: 55                    push %rbp
401ab5: 48 89 e5              mov %rsp,%rbp
401ab8: 48 83 ec 60           sub $0x60,%rsp
401abc: 4c 8d 45 b0           lea -0x50(%rbp),%r8
401ac0: 48 8d 4d a8           lea -0x58(%rbp),%rcx
401ac4: 48 8d 55 ac           lea -0x54(%rbp),%rdx
401ac8: be 79 33 40 00        mov $0x403379,%esi
401acd: bf 70 58 40 00        mov $0x405870,%edi
401ad2: b8 00 00 00 00        mov $0x0,%eax
401ad7: e8 44 f6 ff ff       callq 401120 <__isoc99_sscanf@plt>
401adc: 83 f8 03              cmp $0x3,%eax
401adf: 74 0c                  je 401aed <phase_defused+0x43>
401ae1: bf b8 32 40 00        mov $0x4032b8,%edi
401ae6: e8 75 f5 ff ff       callq 401060 <puts@plt>
401aeb: c9                    leaveq
401aec: c3                    retq
401aed: be 82 33 40 00        mov $0x403382,%esi
401af2: 48 8d 7d b0           lea -0x50(%rbp),%rdi
401af6: e8 21 fd ff ff       callq 40181c <strings_not_equal>
401afb: 85 c0                 test %eax,%eax
401afd: 75 e2                  jne 401ae1 <phase_defused+0x37>
401aff: bf 58 32 40 00        mov $0x403258,%edi
401b04: e8 57 f5 ff ff       callq 401060 <puts@plt>
401b09: bf 80 32 40 00        mov $0x403280,%edi
401b0e: e8 4d f5 ff ff       callq 401060 <puts@plt>
401b13: b8 00 00 00 00        mov $0x0,%eax
401b18: e8 0e fc ff ff       callq 40172b <secret_phase>
401b1d: eb c2                  jmp 401ae1 <phase_defused+0x37>
```

```
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
```

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) b phase_4
Breakpoint 1 at 0x401541: file phases.c, line 130.
(gdb) in
Ambiguous command "in": inf, inferior, info, init-if-undefined, inspect, internal, interpreter-exec, interrupt.
(gdb) in
Ambiguous command "in": inf, inferior, info, init-if-undefined, inspect, internal, interpreter-exec, interrupt.
(gdb) in
Ambiguous command "in": inf, inferior, info, init-if-undefined, inspect, internal, interpreter-exec, interrupt.
(gdb) p (char *) 0x403382
$1 = 0x403382 "DrEvil"
(gdb)
```

```

000000000040172b <secret_phase>:
40172b: 55          push    %rbp
40172c: 48 89 e5    mov     %rsp,%rbp
40172f: 53          push    %rbx
401730: 48 83 ec 08  sub     $0x8,%rsp
401734: e8 40 02 00 00 callq   401979 <read_line>
401739: ba 0a 00 00 00 mov     $0xa,%edx
40173e: be 00 00 00 00 mov     $0x0,%esi
401743: 48 89 c7    mov     %rax,%rdi
401746: e8 a5 f9 ff ff callq   4010f0 <strtol@plt>
40174b: 48 89 c3    mov     %rax,%rbx
40174e: 8d 40 ff    lea     -0x1(%rax),%eax
401751: 3d e8 03 00 00 cmp     $0x3e8,%eax
401756: 77 27      ja      40177f <secret_phase+0x54>
401758: 89 de      mov     %ebx,%esi
40175a: bf f0 50 40 00 mov     $0x4050f0,%edi
40175f: e8 8d ff ff ff callq   4016f1 <fun7>
401764: 83 f8 04    cmp     $0x4,%eax
401767: 75 1d      jne     401786 <secret_phase+0x5b>
401769: bf 78 31 40 00 mov     $0x403178,%edi
40176e: e8 ed f8 ff ff callq   401060 <puts@plt>
401773: e8 32 03 00 00 callq   401aaa <phase_defused>
401778: 48 83 c4 08  add     $0x8,%rsp
40177c: 5b          pop     %rbx
40177d: 5d          pop     %rbp
40177e: c3          retq
40177f: e8 97 01 00 00 callq   40191b <explode_bomb>
401784: eb d2      jmp     401758 <secret_phase+0x2d>
401786: e8 90 01 00 00 callq   40191b <explode_bomb>
40178b: eb dc      jmp     401769 <secret_phase+0x3e>

```

```

00000000004016f1 <fun7>:
4016f1: 48 85 ff    test    %rdi,%rdi
4016f4: 74 2f      je      401725 <fun7+0x34>
4016f6: 55          push    %rbp
4016f7: 48 89 e5    mov     %rsp,%rbp
4016fa: 8b 07      mov     (%rdi),%eax
4016fc: 39 f0      cmp     %esi,%eax
4016fe: 7f 09      jg      401709 <fun7+0x18>
401700: 75 14      jne     401716 <fun7+0x25>
401702: b8 00 00 00 00 mov     $0x0,%eax
401707: 5d          pop     %rbp
401708: c3          retq
401709: 48 8b 7f 08 mov     0x8(%rdi),%rdi
40170d: e8 df ff ff ff callq   4016f1 <fun7>
401712: 01 c0      add     %eax,%eax
401714: eb f1      jmp     401707 <fun7+0x16>
401716: 48 8b 7f 10 mov     0x10(%rdi),%rdi
40171a: e8 d2 ff ff ff callq   4016f1 <fun7>
40171f: 8d 44 00 01 lea     0x1(%rax,%rax,1),%eax
401723: eb e2      jmp     401707 <fun7+0x16>
401725: b8 ff ff ff ff mov     $0xffffffff,%eax
40172a: c3          retq

```

在 phase\_defused 中找到进入隐藏关的特定字符串，加到第四关答案后，再研究 secret\_phase 函数，发现 fun7 返回值必须为 4，fun7 函数为一个二分查找，倒推可得最后所需密码为 7。

## 第 4 章 总结

### 4.1 请总结本次实验的收获

更加熟悉了反汇编代码  
更加熟练地掌握了 GDB 的使用

### 4.2 请给出对本次实验内容的建议

无

注：本章为酌情加分项。

## 参考文献

### 为完成本次实验你翻阅的书籍与网站等

- [1] 林来兴. 空间控制技术[M]. 北京: 中国宇航出版社, 1992: 25-42.
- [2] 辛希孟. 信息技术与信息服务国际研讨会论文集: A 集[C]. 北京: 中国科学出版社, 1999.
- [3] 赵耀东. 新时代的工业工程师[M/OL]. 台北: 天下文化出版社, 1998 [1998-09-26]. <http://www.ie.nthu.edu.tw/info/ie.newie.htm> (Big5) .
- [4] 湛颖. 空间交会控制理论与方法研究[D]. 哈尔滨: 哈尔滨工业大学, 1992: 8-13.
- [5] KANAMORI H. Shaking Without Quaking[J]. Science, 1998, 279 (5359): 2063-2064.
- [6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science , 1998 , 281 : 331-332[1998-09-23]. <http://www.sciencemag.org/cgi/collection/anatmorp>.