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 1180300829

班　　级 1803008

学　　生 余涛

指 导 教 师 吴锐

实 验 地 点 G709

实 验 日 期 2019.10.19

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。

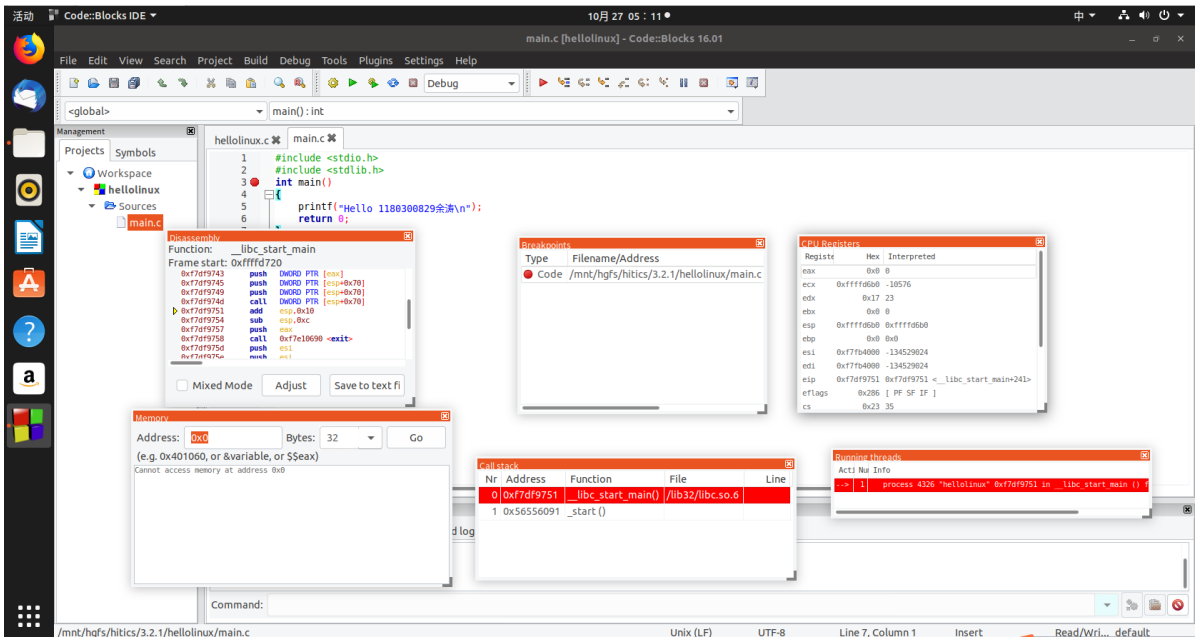


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

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

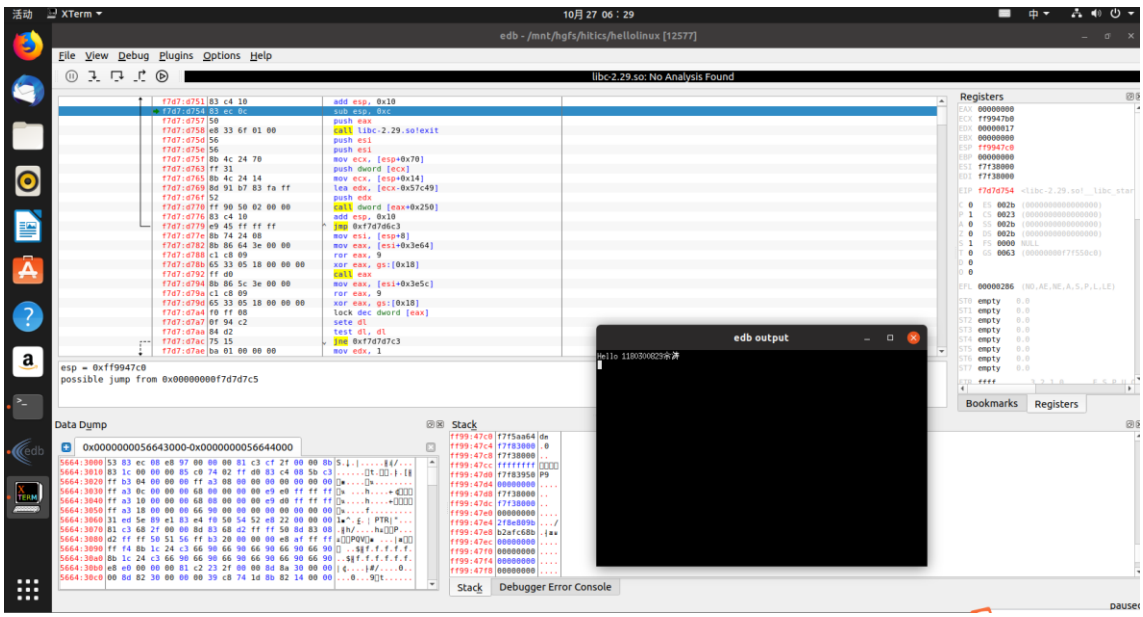


图 2-2　Ubuntu 下 EDB 截图

第 3 章 各阶段炸弹破解与分析

每阶段 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：The moon unit will be divided into two divisions.

破解过程：

（1）strings_not_equal 函数是判断输入的字符串和%rsi 中的字符串是否相等，如果相等则令%eax 为 0，否则为 1。

```
000000000040183f <strings_not_equal>:
  40183f:       55                      push   %rbp
  401840:       48 89 e5                mov    %rsp,%rbp
  401843:       41 55                   push   %r13
  401845:       41 54                   push   %r12
  401847:       53                      push   %rbx
  401848:       48 83 ec 08             sub    $0x8,%rsp
  40184c:       48 89 fb                mov    %rdi,%rbx
  40184f:       49 89 f4                mov    %rsi,%r12
  401852:       e8 d4 ff ff ff          callq  40182b <string_length>
  401857:       41 89 c5                mov    %eax,%r13d
  40185a:       4c 89 e7                mov    %r12,%rdi
  40185d:       e8 c9 ff ff ff          callq  40182b <string_length>
  401862:       41 39 c5                cmp    %eax,%r13d
  401865:       75 1e                   jne    401885 <strings_not_equal+0x46>
  401867:       0f b6 03                movzbl (%rbx),%eax
  40186a:       84 c0                   test   %al,%al
  40186c:       74 10                   je     40187e <strings_not_equal+0x3f>
  40186e:       41 38 04 24             cmp    %al,(%r12)
  401872:       75 21                   jne    401895 <strings_not_equal+0x56>
  401874:       48 83 c3 01             add    $0x1,%rbx
  401878:       49 83 c4 01             add    $0x1,%r12
  40187c:       eb e9                   jmp    401867 <strings_not_equal+0x28>
  40187e:       b8 00 00 00 00          mov    $0x0,%eax
  401883:       eb 05                   jmp    40188a <strings_not_equal+0x4b>
  401885:       b8 01 00 00 00          mov    $0x1,%eax
  40188a:       48 83 c4 08             add    $0x8,%rsp
  40188e:       5b                      pop    %rbx
  40188f:       41 5c                   pop    %r12
  401891:       41 5d                   pop    %r13
  401893:       5d                      pop    %rbp
  401894:       c3                      retq
  401895:       b8 01 00 00 00          mov    $0x1,%eax
  40189a:       eb ee                   jmp    40188a <strings_not_equal+0x4b>
```

```
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 3c 04 00 00          callq  40183f <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 30 05 00 00          callq  40193e <explode_bomb>
  40140e:       eb f7                   jmp    401407 <phase_1+0x12>
```

并且后面的 test %eax,%eax 和 jne 401409 则是判断若%eax 不为 0，就执行 explode_bomb。因此需要输入的密码必须与%rsi 中的字符串相等。

```
  4012f7:       e8 a0 06 00 00          callq  40199c <read_line>
  4012fc:       48 89 c7                mov    %rax,%rdi
  4012ff:       e8 f1 00 00 00          callq  4013f5 <phase_1>
  401304:       e8 c4 07 00 00          callq  401acd <phase_defused>
  401309:       bf f8 30 40 00          mov    $0x4030f8,%edi
  40130e:       e8 4d fd ff ff          callq  401060 <puts@plt>
```

read_line 将用户输入的字符串的返回值转移给%rdi，并且在 phase_1 中与 0x403150 中储存的密码相比较，所以只需找到 0x403150 中储存的密码即可

```
(gdb) x/10i phase_1
=> 0x4013f5 <phase_1>:  push   %rbp
   0x4013f6 <phase_1+1>:        mov    %rsp,%rbp
   0x4013f9 <phase_1+4>:        mov    $0x403150,%esi
   0x4013fe <phase_1+9>:        callq  0x40183f <strings_not_equal>
   0x401403 <phase_1+14>:       test   %eax,%eax
   0x401405 <phase_1+16>:       jne    0x401409 <phase_1+20>
   0x401407 <phase_1+18>:       pop    %rbp
   0x401408 <phase_1+19>:       retq
   0x401409 <phase_1+20>:       callq  0x40193e <explode_bomb>
   0x40140e <phase_1+25>:       jmp    0x401407 <phase_1+18>
(gdb) x/s 0x403150
0x403150:       "The moon unit will be divided into two divisions."
```

查看地址 0x403150 中的内容得到密码为 The moon unit will be divided into two divisions.

验证：

```
yt1180300829@ubuntu:~/hitics/bomb$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
The moon unit will be divided into two divisions.
Phase 1 defused. How about the next one?
```

3.2 阶段 2 的破解与分析

密码如下：0 1 3 6 10 15（只要第一个数大于 0，其他的数差值相同就满足，比如 1 2 4 7 11 16）

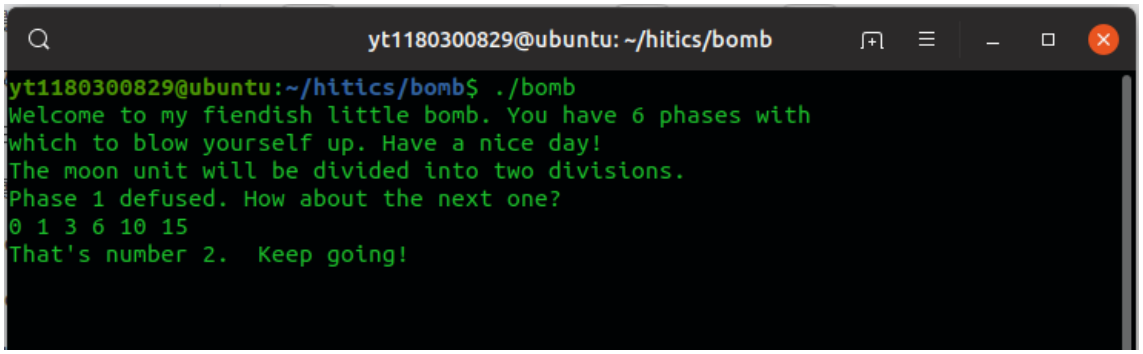
破解过程：

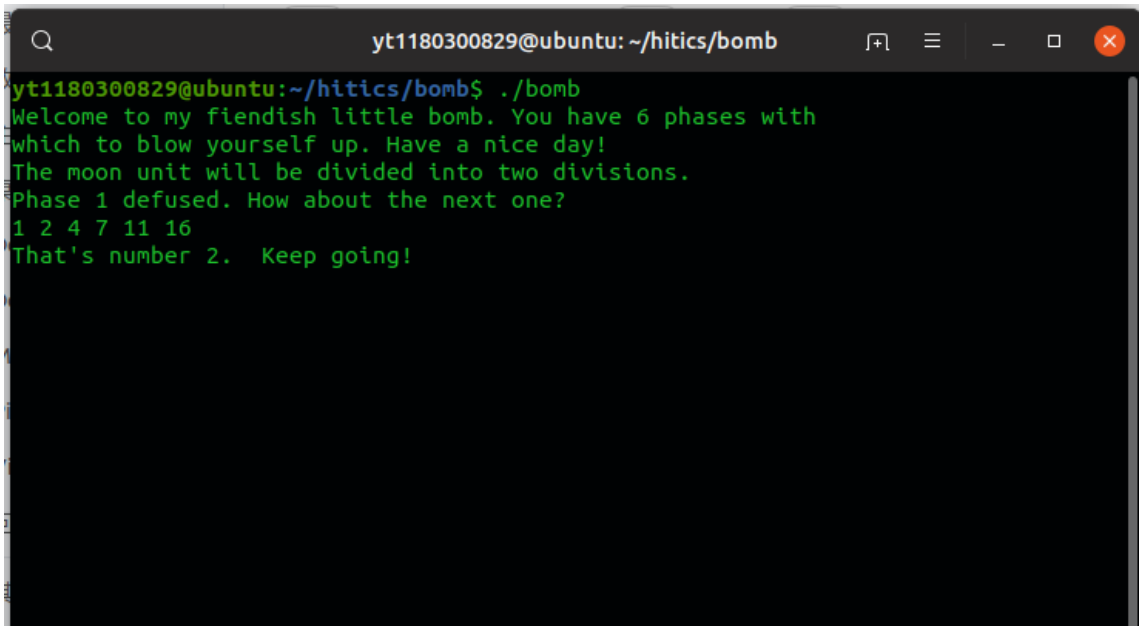
```
0000000000401410 <phase_2>:
  401410:       55                      push   %rbp
  401411:       48 89 e5                mov    %rsp,%rbp
  401414:       53                      push   %rbx
  401415:       48 83 ec 28             sub    $0x28,%rsp
  401419:       48 8d 75 d0             lea    -0x30(%rbp),%rsi
  40141d:       e8 3e 05 00 00          callq  401960 <read_six_numbers>
  401422:       83 7d d0 00             cmpl   $0x0,-0x30(%rbp)
  401426:       78 07                   js     40142f <phase_2+0x1f>
  401428:       bb 01 00 00 00          mov    $0x1,%ebx
  40142d:       eb 0f                   jmp    40143e <phase_2+0x2e>
  40142f:       e8 0a 05 00 00          callq  40193e <explode_bomb>
  401434:       eb f2                   jmp    401428 <phase_2+0x18>
  401436:       e8 03 05 00 00          callq  40193e <explode_bomb>
  40143b:       83 c3 01                add    $0x1,%ebx
  40143e:       83 fb 05                cmp    $0x5,%ebx
  401441:       7f 17                   jg     40145a <phase_2+0x4a>
  401443:       48 63 c3                movslq %ebx,%rax
  401446:       8d 53 ff                lea    -0x1(%rbx),%edx
  401449:       48 63 d2                movslq %edx,%rdx
  40144c:       89 d9                   mov    %ebx,%ecx
  40144e:       03 4c 95 d0             add    -0x30(%rbp,%rdx,4),%ecx
  401452:       39 4c 85 d0             cmp    %ecx,-0x30(%rbp,%rax,4)
  401456:       74 e3                   je     40143b <phase_2+0x2b>
  401458:       eb dc                   jmp    401436 <phase_2+0x26>
  40145a:       48 83 c4 28             add    $0x28,%rsp
  40145e:       5b                      pop    %rbx
  40145f:       5d                      pop    %rbp
  401460:       c3                      retq
```

分析 phase_2 得，由函数 read_six_numbers 可猜测会输入六个 int 型的数值。首先 cmpl $0x0,-0x30(%rbp)即比较 0 和-0x30(%rbp)且小于 0 时爆炸，故第一个数必须大于等于 0，我们假设第一个数为 0，往后面分析得，%ebx 的初始值为 1，然后到 40143e 判断其是否小于等于 5，并将此条件作为循环条件，每次循环一次，%ebx 都加 1，到%ebx=5 时跳出循环。

在每次循环内，都将%rbx 给%edx，此时%edx 为上面%ebx 的值，到后面 add -0x30(%rbp,%rdx,4),%ecx 和 cmp %ecx,-0x30(%rbp,%rax,4)可知后面被被比较的数都是%rbp，且%rbp=%rbp+%rdx，这五个数分别为 0+1=1,1+2=3,3+3=6,6+4=10,10+5=15，则六个数为 0,1,3,6,10,15。

将初始值设为 1 也满足，即为 1 2 4 7 11 16。





3.3 阶段 3 的破解与分析

密码如下：5 -709

破解过程：

先查看输入的值是什么类型，查看 0x40334f 地址的内容，发现为”%d %d”,则确定输入为两个整型数据

```
401468:      48 8d 55 fc            lea    -0x4(%rbp),%rdx
401471:      be 4f 33 40 00         mov    $0x40334f,%esi
```

```
yt1180300829@ubuntu: ~/hitics/bomb
yt1180300829@ubuntu:~/hitics/bomb$ gdb bomb
GNU gdb (Ubuntu 8.2.91.20190405-0ubuntu3) 8.2.91.20190405-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
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) x/s 0x40334f
0x40334f:       "%d %d"
```

直接观察__isoc99_sscanf@plt 后的内容，即为输入函数后的内容

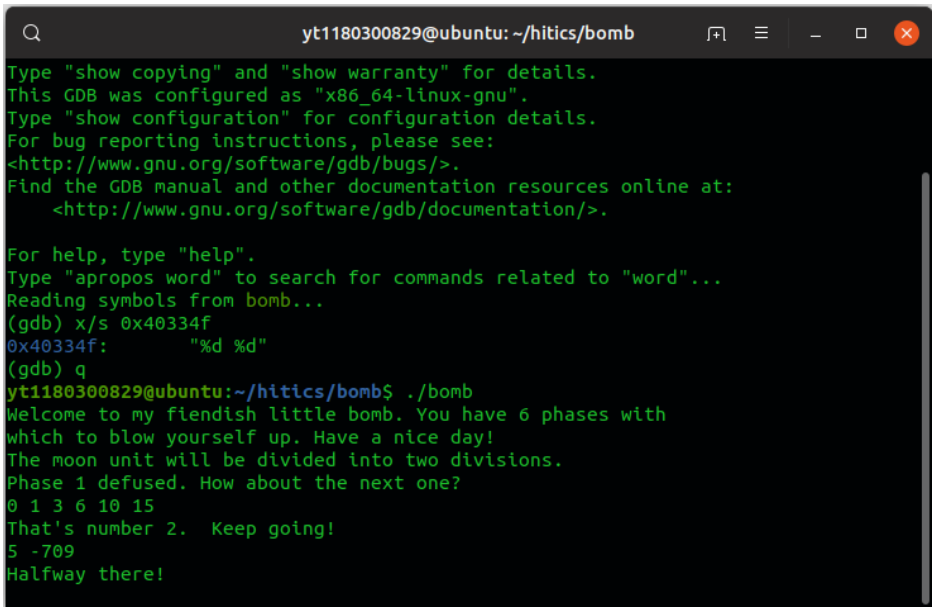
首先有语句 cmpl　$0x7,-0x4(%rbp)和 ja　401505 <phase_3+0xa4>，说明输入的第一个数为小于等于 7 的无符号数，则可确定第一个数的范围为 0~7。

然后有语句 cmpl　$0x5,-0x4(%rbp)和 jg　4014d4 <phase_3+0x73>，说明输入的第一个数为小于等于 5 的无符号数，则可确定第一个数有六种取值，分别为 0,1,2,3,4,5

```
40147b:   e8 a0 fc ff ff      callq  401120 <__isoc99_sscanf@plt>
401480:   83 f8 01            cmp    $0x1,%eax
401483:   7e 10               jle    401495 <phase_3+0x34>
401485:   83 7d fc 07         cmpl   $0x7,-0x4(%rbp)
401489:   77 7a               ja     401505 <phase_3+0xa4>
40148b:   8b 45 fc            mov    -0x4(%rbp),%eax
40148e:   ff 24 c5 c0 31 40 00 jmpq   *0x4031c0(,%rax,8)
401495:   e8 a4 04 00 00      callq  40193e <explode_bomb>
40149a:   eb e9               jmp    401485 <phase_3+0x24>
40149c:   b8 00 00 00 00      mov    $0x0,%eax0x4(%rbp),%eax
4014a1:   eb 05               jmp    4014a8 <phase_3+0x47>
4014a3:   b8 a5 00 00 00      mov    $0xa5,%eax
4014a8:   83 e8 67            sub    $0x67,%eax
4014ab:   05 06 02 00 00      add    $0x206,%eax
4014b0:   2d c5 02 00 00      sub    $0x2c5,%eax
4014b5:   05 c5 02 00 00      add    $0x2c5,%eax
4014ba:   2d c5 02 00 00      sub    $0x2c5,%eax
4014bf:   05 c5 02 00 00      add    $0x2c5,%eax
4014c4:   2d c5 02 00 00      sub    $0x2c5,%eax
4014c9:   83 7d fc 05         cmpl   $0x5,-0x4(%rbp)
4014cd:   7f 05               jg     4014d4 <phase_3+0x73>
4014cf:   39 45 f8            cmp    %eax,-0x8(%rbp)
4014d2:   74 05               je     4014d9 <phase_3+0x78>
4014d4:   e8 65 04 00 00      callq  40193e <explode_bomb>
4014d9:   c9                  leaveq
4014da:   c3                  retq
4014db:   b8 00 00 00 00      mov    $0x0,%eax      case0
4014e0:   eb c9               jmp    4014ab <phase_3+0x4a>
4014e2:   b8 00 00 00 00      mov    $0x0,%eax      case1
4014e7:   eb c7               jmp    4014b0 <phase_3+0x4f>
4014e9:   b8 00 00 00 00      mov    $0x0,%eax      case2
4014ee:   eb c5               jmp    4014b5 <phase_3+0x54>
4014f0:   b8 00 00 00 00      mov    $0x0,%eax      case3
4014f5:   eb c3               jmp    4014ba <phase_3+0x59>
4014f7:   b8 00 00 00 00      mov    $0x0,%eax      case4
4014fc:   eb c1               jmp    4014bf <phase_3+0x5e>
4014fe:   b8 00 00 00 00      mov    $0x0,%eax      case5
401503:   eb bf               jmp    4014c4 <phase_3+0x63>
401505:   e8 34 04 00 00      callq  40193e <explode_bomb>
40150a:   b8 00 00 00 00      mov    $0x0,%eax
40150f:   eb b8               jmp    4014c9 <phase_3+0x68>
```

然后对其中一种情况进行分析，若第一个数为 5，即为 case5 的情况，将跳转到 4014c4，执行 sub　　$0x2c5,%eax 得到第二个数值为-709。

验证得：



3.4 阶段 4 的破解与分析

密码如下：176 2

破解过程：

```
000000000040155b <phase_4>:
  40155b:       55                      push   %rbp
  40155c:       48 89 e5                mov    %rsp,%rbp
  40155f:       48 83 ec 10             sub    $0x10,%rsp
  401563:       48 8d 4d fc             lea    -0x4(%rbp),%rcx
  401567:       48 8d 55 f8             lea    -0x8(%rbp),%rdx
  40156b:       be 4f 33 40 00          mov    $0x40334f,%esi
  401570:       b8 00 00 00 00          mov    $0x0,%eax
  401575:       e8 a6 fb ff ff          callq  401120 <__isoc99_sscanf@plt>
  40157a:       83 f8 02                cmp    $0x2,%eax
  40157d:       75 0d                   jne    40158c <phase_4+0x31>
  40157f:       8b 45 fc                mov    -0x4(%rbp),%eax
  401582:       83 f8 01                cmp    $0x1,%eax
  401585:       7e 05                   jle    40158c <phase_4+0x31>
  401587:       83 f8 04                cmp    $0x4,%eax
  40158a:       7e 05                   jle    401591 <phase_4+0x36>
  40158c:       e8 ad 03 00 00          callq  40193e <explode_bomb>
  401591:       8b 75 fc                mov    -0x4(%rbp),%esi
  401594:       bf 09 00 00 00          mov    $0x9,%edi
  401599:       e8 73 ff ff ff          callq  401511 <func4>
  40159e:       39 45 f8                cmp    %eax,-0x8(%rbp)
  4015a1:       75 02                   jne    4015a5 <phase_4+0x4a>
  4015a3:       c9                      leaveq
  4015a4:       c3                      retq
  4015a5:       e8 94 03 00 00          callq  40193e <explode_bomb>
  4015aa:       eb f7                   jmp    4015a3 <phase_4+0x48>
```

首先查看 0x40334f 中的内容，为“%d %d”即输入为两个整型变量

```
(gdb) x/s 0x40334f
0x40334f:       "%d %d"
(gdb)
```

cmp $0x2,%eax 和 jne 40158c <phase_4+0x31>可以进一步判断为两个变量。

lea -0x4(%rbp),%rcx 和 lea -0x8(%rbp),%rdx 说明两个参数为%rcx 和%rdx。

先在 phase_4 中判断第一个参数的范围，cmp $0x1,%eax 和 jle 40158c <phase_4+0x31>以及 cmp $0x4,%eax 和 jle 401591 <phase_4+0x36>确定了第一个参数的范围为大于 1 且小于等于 4，我们取为 2。

mov -0x4(%rbp),%esi，mov $0x9,%edi 确定了 fun4 的函数参数有有第一个数和 9。

cmp %eax,-0x8(%rbp)是把 fun4 的返回值与第二个参数进行比较，则可说明 fun4 的返回值即为第二个参数的值。

然后分析 fun4 函数：

```
0000000000401511 <func4>:
  401511:       85 ff                   test   %edi,%edi
  401513:       7e 3d                   jle    401552 <func4+0x41>
  401515:       83 ff 01                cmp    $0x1,%edi
  401518:       74 3e                   je     401558 <func4+0x47>
  40151a:       55                      push   %rbp
  40151b:       48 89 e5                mov    %rsp,%rbp
  40151e:       41 55                   push   %r13
  401520:       41 54                   push   %r12
  401522:       53                      push   %rbx
  401523:       48 83 ec 08             sub    $0x8,%rsp
  401527:       89 f3                   mov    %esi,%ebx
  401529:       41 89 fc                mov    %edi,%r12d
  40152c:       8d 7f ff                lea    -0x1(%rdi),%edi
  40152f:       e8 dd ff ff ff          callq  401511 <func4>
  401534:       44 8d 2c 18             lea    (%rax,%rbx,1),%r13d
  401538:       41 8d 7c 24 fe          lea    -0x2(%r12),%edi
  40153d:       89 de                   mov    %ebx,%esi
  40153f:       e8 cd ff ff ff          callq  401511 <func4>
  401544:       44 01 e8                add    %r13d,%eax
  401547:       48 83 c4 08             add    $0x8,%rsp
  40154b:       5b                      pop    %rbx
  40154c:       41 5c                   pop    %r12
  40154e:       41 5d                   pop    %r13
  401550:       5d                      pop    %rbp
  401551:       c3                      retq
  401552:       b8 00 00 00 00          mov    $0x0,%eax
  401557:       c3                      retq
  401558:       89 f0                   mov    %esi,%eax
  40155a:       c3                      retq
```

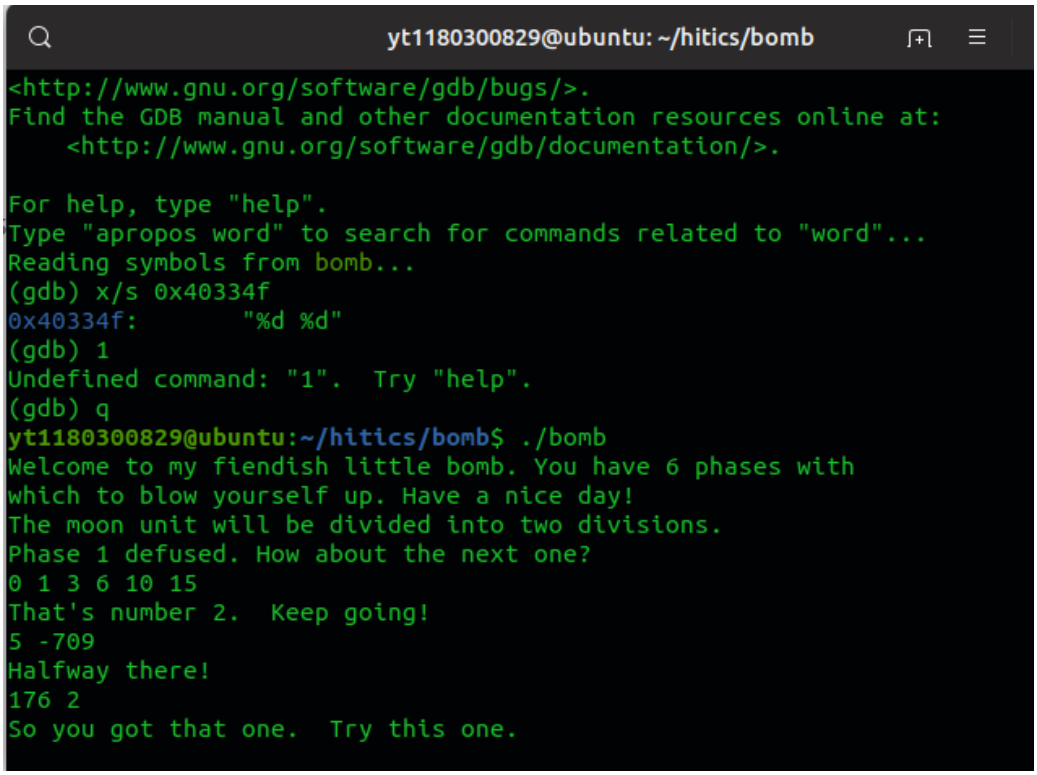
经过分析可得：fun4 函数为

```
int fun4(int n,int m)
{
    if(n==0)
    {
        return 0;
    }
    if(n==1)
    {
        return m;
    }
    if(n>1)
    {
        return fun4(n-1)+fun4(n-2)+m;
    }
}
```

其中 n 为 9，m 为 2，运算可得 fun(4)=176。

则两个值分别为 176 和 2

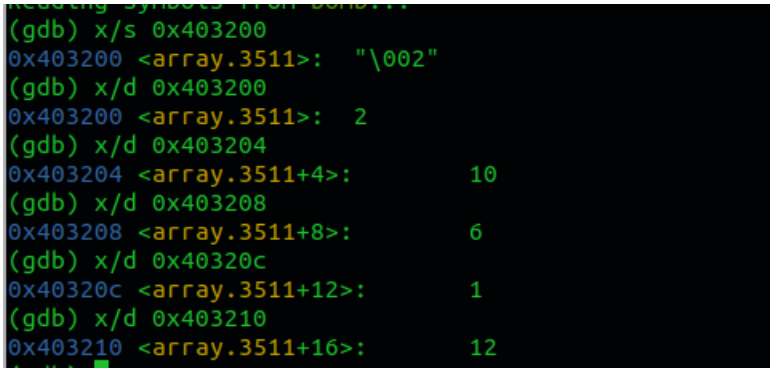
验证：



3.5 阶段 5 的破解与分析

密码如下：444401

破解过程：



```
0000000004015ac <phase_5>:
  4015ac:       55                      push   %rbp
  4015ad:       48 89 e5                mov    %rsp,%rbp
  4015b0:       53                      push   %rbx
  4015b1:       48 83 ec 08             sub    $0x8,%rsp
  4015b5:       48 89 fb                mov    %rdi,%rbx
  4015b8:       e8 6e 02 00 00          callq  40182b <string_length>
  4015bd:       83 f8 06                cmp    $0x6,%eax
  4015c0:       75 25                   jne    4015e7 <phase_5+0x3b>
  4015c2:       b9 00 00 00 00          mov    $0x0,%ecx
  4015c7:       b8 00 00 00 00          mov    $0x0,%eax
  4015cc:       83 f8 05                cmp    $0x5,%eax
  4015cf:       7f 1d                   jg     4015ee <phase_5+0x42>
  4015d1:       48 63 d0                movslq %eax,%rdx
  4015d4:       0f b6 14 13             movzbl (%rbx,%rdx,1),%edx
  4015d8:       83 e2 0f                and    $0xf,%edx
  4015db:       03 0c 95 00 32 40 00    add    0x403200(,%rdx,4),%ecx
  4015e2:       83 c0 01                add    $0x1,%eax
  4015e5:       eb e5                   jmp    4015cc <phase_5+0x20>
  4015e7:       e8 52 03 00 00          callq  40193e <explode_bomb>
  4015ec:       eb d4                   jmp    4015c2 <phase_5+0x16>
  4015ee:       83 f9 3c                cmp    $0x3c,%ecx
  4015f1:       75 07                   jne    4015fa <phase_5+0x4e>
  4015f3:       48 83 c4 08             add    $0x8,%rsp
  4015f7:       5b                      pop    %rbx
  4015f8:       5d                      pop    %rbp
  4015f9:       c3                      retq
  4015fa:       e8 3f 03 00 00          callq  40193e <explode_bomb>
  4015ff:       eb f2                   jmp    4015f3 <phase_5+0x47>
```

由 cmp $0x6,%eax 和 jne 4015e7 <phase_5+0x3b>可以判断输入为六个字符。

cmp $0x5,%eax 和 jg 4015ee <phase_5+0x42>未循环判断条件，即为数组中的每个元素，最多到第六个元素结束循环。

movzbl (%rbx,%rdx,1),%edx 和 and $0xf,%edx 是将输入字符拓展为 32 为赋值给%edx 再取%edx 的后四位

add 0x403200(,%rdx,4),%ecx 中是加上地址为 0x403200，偏移量为 4%rdx 的内容，于是查看里面的各字节的内容

```
(gdb) x/s 0x403200
0x403200 <array.3511>:  "\002"
(gdb) x/d 0x403200
0x403200 <array.3511>:  2
(gdb) x/d 0x403204
0x403204 <array.3511+4>:        10
(gdb) x/d 0x403208
0x403208 <array.3511+8>:        6
(gdb) x/d 0x40320c
0x40320c <array.3511+12>:       1
(gdb) x/d 0x403210
0x403210 <array.3511+16>:       12
```

add　　$0x1,%eax 是每加一次后，%eax+1。

cmp　　$0x3c,%ecx 为最后的判断条件，即当经过六次循环之后，%ecx 的值要等于 0x3c，即十进制的 60 才能不爆炸。

进行分析，为了六个数总和等于 60，我们可以取六个值为 12,12,12,12,2,10，即此时%rdx 为对应的 444401，即为破解密码。

验证：

```
yt1180300829@ubuntu:~/hitics/bomb$ ./bomb ans
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
So you got that one.  Try this one.
444401
Good work!  On to the next...
```

3.6 阶段 6 的破解与分析

密码如下：4 3 5 6 2 1

破解过程：

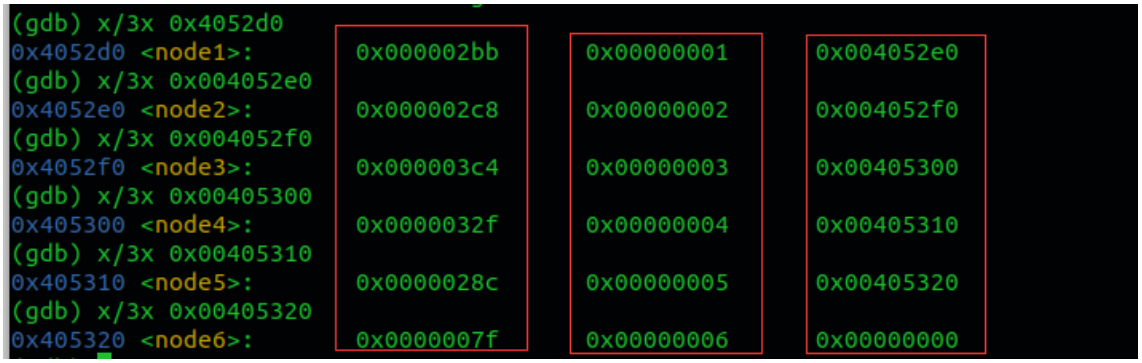
首先通过 read_six_numbers 函数可以得到输入了 6 个数，然后 mov $0x0,%r12d，cmp　　$0x5,%ebx 和 jg　　401667 <phase_6+0x66>可以判断这六个数大于等于 1，小于等于 6，即存在于 1,2,3,4,5,6 中

```
401612:       e8 49 03 00 00          callq  401960 <read_six_numbers>
401617:       41 bc 00 00 00 00       mov    $0x0,%r12d
40161d:       eb 29                   jmp    401648 <phase_6+0x47>
40161f:       e8 1a 03 00 00          callq  40193e <explode_bomb>
401624:       eb 37                   jmp    40165d <phase_6+0x5c>
401626:       83 c3 01                add    $0x1,%ebx
401629:       83 fb 05                cmp    $0x5,%ebx
40162c:       7f 17                   jg     401645 <phase_6+0x44>
40162e:       49 63 c4                movslq %r12d,%rax
401631:       48 63 d3                movslq %ebx,%rdx
401634:       8b 7c 95 c0             mov    -0x40(%rbp,%rdx,4),%edi
401638:       39 7c 85 c0             cmp    %edi,-0x40(%rbp,%rax,4)
40163c:       75 e8                   jne    401626 <phase_6+0x25>
40163e:       e8 fb 02 00 00          callq  40193e <explode_bomb>
401643:       eb e1                   jmp    401626 <phase_6+0x25>
401645:       45 89 ec                mov    %r13d,%r12d
401648:       41 83 fc 05             cmp    $0x5,%r12d
40164c:       7f 19                   jg     401667 <phase_6+0x66>
40164e:       49 63 c4                movslq %r12d,%rax
401651:       8b 44 85 c0             mov    -0x40(%rbp,%rax,4),%eax
401655:       83 e8 01                sub    $0x1,%eax
401658:       83 f8 05                cmp    $0x5,%eax
40165b:       77 c2                   ja     40161f <phase_6+0x1e>
40165d:       45 8d 6c 24 01          lea    0x1(%r12),%r13d
401662:       44 89 eb                mov    %r13d,%ebx
401665:       eb c2                   jmp    401629 <phase_6+0x28>
401667:       b8 00 00 00 00          mov    $0x0,%eax
40166c:       eb 13                   jmp    401681 <phase_6+0x80>
```

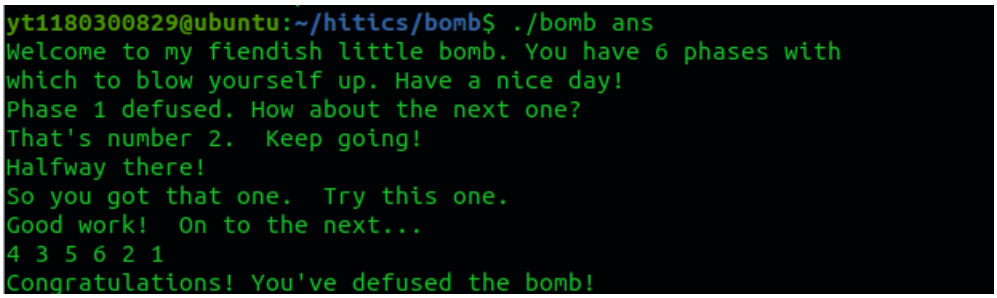
又由后面比较两个数如果相等就爆炸可以得到六个数为 1,2,3,4,5,6 中的不重复的排列。

分析后面的汇编代码可以得到其实这是对六个节点的降序排列，运用 gdb 查看第一个节点中储存的内容，可以发现节点中储存了用于比较大小的数值和下一个节点的地址，则运用 gdb 查看每一个节点地址里的内容可以对储存的数值进行降序排列，得到 3,4,2,1,5,6

然后每一个都与 7 进行了比较，所以为 7-n：4,3,5,6,2,1



验证：



3.7 阶段 7 的破解与分析(隐藏阶段)

密码如下：

破解过程：

第 4 章 总结

4.1 请总结本次实验的收获

本次实验学会了 codeblocks 的反汇编调试方法，学会了 edb 的调试方法，更加熟练运用了 gdb 调试，学会了分析反汇编代码来寻找其中储存的某些关键信息进而破解密码，对反汇编语句运用更加熟练，明白了汇编语言中各个函数之间的调用关系，对堆栈有了更深的理解，学会了对教材知识熟练运用

4.2 请给出对本次实验内容的建议

由于网上的知识可能与教材不一致或者表达方法不一样，希望能够在 ppt 中增加一些比较权威的相关知识的网址，进而能够让我们对知识进行更为权威的掌握。

希望增加以下方法的点拨或者参考。

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] 谌颖. 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.