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 计算机科学与技术

学　　号 1183200123

班　　级 11803003

学　　生 祁天

指 导 教 师 史先俊

实 验 地 点 G712

实 验 日 期 2019 年 10 月 23 日

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 - 3 -

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 - 6 -

3.1 阶段 1 的破解与分析 - 6 -
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 - 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 - 14 -

4.1 请总结本次实验的收获 - 14 -
4.2 请给出对本次实验内容的建议 - 14 -

参考文献 - 15 -

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)、Og、O0、O1、O2、O3、Og，-m32/m64。再次查看生成的汇编语言与原来的区别。

堆栈访问[rbp+-n]或[rsp+n]。-fno-omit-frame-pointer。
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。

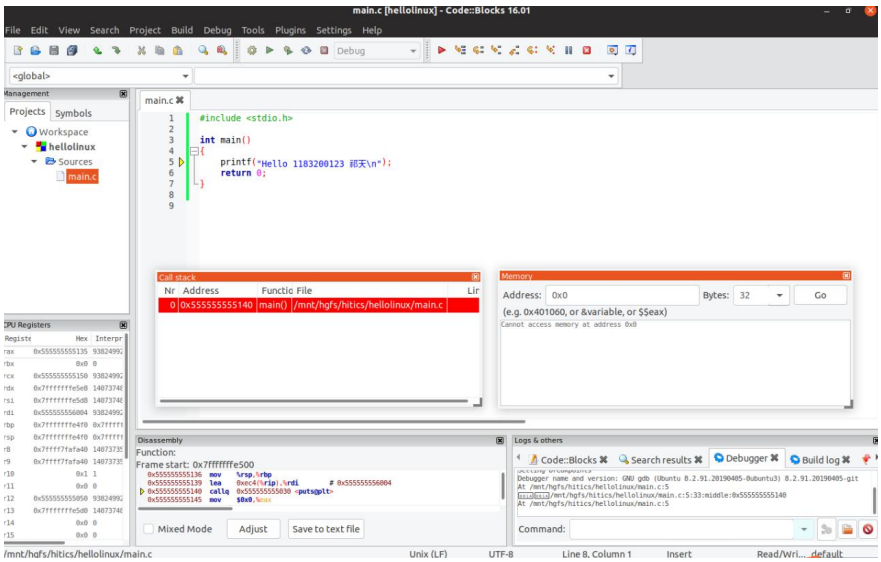


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

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

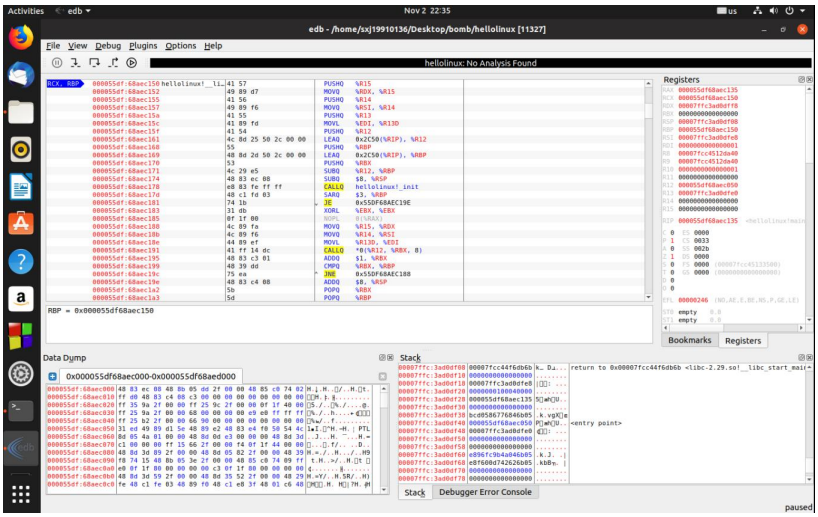


图 2-2 Ubuntu 下 EDB 截图

第 3 章 各阶段炸弹破解与分析（rsp 版本）

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：

运行到 phase_1 之前需调用 bomb!read_line 函数读取用户输入的字符串并将其首地址放至%rdi 中。

```
00000000:004012ef  e8 51 06 00 00   CALLQ   bomb!read_line
00000000:004012f4  48 89 c7         MOVQ    %RAX, %RDI
00000000:004012f7  e8 ec 00 00 00   CALLQ   bomb!phase_1
```

phase_1 需要两个参数：%rdi（用户输入的字符串地址）、%esi（密码的地址，我们根据这个地址到内存中查看相应内容，即为第 1 关密码），然后调用 bomb!string_not_equal 函数判断这两个是否相等，相等返回 0，不等返回 1，返回值在%eax 中。如果两字符串不相等（输入密码错误），则调用 bomb!explode_bomb 函数，即引爆炸弹；如果相等，则返回主函数，进入下一关。

```
00000000:004013e8 bomb!phase_1  48 83 ec 08         SUBQ    $8, %RSP
00000000:004013ec               be 50 31 40 00      MOVL    $0x403150, %ESI
00000000:004013f1               e8 09 04 00 00      CALLQ   bomb!strings_not_equal
00000000:004013f6               85 c0               TESTL   %EAX, %EAX
00000000:004013f8               75 05               JNE     0x4013FF
00000000:004013fa               48 83 c4 08         ADDQ    $8, %RSP
00000000:004013fe               c3                  RETQ
00000000:004013ff               e8 e0 04 00 00      CALLQ   bomb!explode_bomb
00000000:00401404               eb f4               JMP     0x4013FA

00000000:00403150  49 20 61 6d 20 6a 75 73 74 20 61 20 72 65 6e 65  I am just a rene
00000000:00403160  67 61 64 65 20 68 6f 63 6b 65 79 20 6d 6f 6d 2e  gade hockey mom.
```

3.2 阶段 2 的破解与分析

密码如下：1 2 4 8 16 32

破解过程：

首先判断是否读入 6 个数字，函数 bomb!read_six_numbers 实现了这个功能。当少于 6 个字符时会爆炸，而大于等于 6 个时不会爆炸。

```
00000000:00401906 bomb!read_six_numbers  48 83 ec 08        SUBQ   $8, %RSP
00000000:0040190a                        48 89 f2           MOVQ   %RSI, %RDX
00000000:0040190d                        48 8d 4e 04        LEAQ   4(%RSI), %RCX
00000000:00401911                        48 8d 46 14        LEAQ   0x14(%RSI), %RAX
00000000:00401915                        50                 PUSHQ  %RAX
00000000:00401916                        48 8d 46 10        LEAQ   0x10(%RSI), %RAX
00000000:0040191a                        50                 PUSHQ  %RAX
00000000:0040191b                        4c 8d 4e 0c        LEAQ   0xC(%RSI), %R9
00000000:0040191f                        4c 8d 46 08        LEAQ   8(%RSI), %R8
00000000:00401923                        be 03 33 40 00     MOVL   $0x403303, %ESI     ASCII "%d %d %d %d %d %d"
00000000:00401928                        b8 00 00 00 00     MOVL   $0, %EAX
00000000:0040192d                        e8 ee f7 ff ff     CALLQ  bomb!__isoc99_sscanf@plt
00000000:00401932                        48 83 c4 10        ADDQ   $0x10, %RSP
00000000:00401936                        83 f8 05           CMPL   $5, %EAX
00000000:00401939                        7e 05              JLE    0x401940
00000000:0040193b                        48 83 c4 08        ADDQ   $8, %RSP
00000000:0040193f                        c3                 RETQ
00000000:00401940                        e8 9f ff ff ff     CALLQ  bomb!explode_bomb
```

bomb!phase_2 想要表达意思是：当 i<=5 时，是否满足 a[i]==2 a[i-1]，若有一项不满足，则炸弹爆炸。其中 i 在 rdx 中，i-1 在 rax 中。

```
00000000:00401406 bomb!phase_2  53                 PUSHQ  %RBX
00000000:00401407               48 83 ec 20        SUBQ   $0x20, %RSP
00000000:0040140b               48 89 e6           MOVQ   %RSP, %RSI
00000000:0040140e               e8 f3 04 00 00     CALLQ  bomb!read_six_numbers
00000000:00401413               83 3c 24 01        CMPL   $1, 0(%RSP)
00000000:00401417               75 07              JNE    0x401420
00000000:00401419               bb 01 00 00 00     MOVL   $1, %EBX
00000000:0040141e               eb 0f              JMP    0x40142F
00000000:00401420               e8 bf 04 00 00     CALLQ  bomb!explode_bomb
00000000:00401425               eb f2              JMP    0x401419
00000000:00401427               e8 b8 04 00 00     CALLQ  bomb!explode_bomb
00000000:0040142c               83 c3 01           ADDL   $1, %EBX
00000000:0040142f               83 fb 05           CMPL   $5, %EBX
00000000:00401432               7f 14              JG     0x401448
00000000:00401434               48 63 d3           MOVSLQ %EBX, %RDX
00000000:00401437               8d 43 ff           LEAL   -1(%RBX), %EAX
00000000:0040143a               48 98              CLTQ
00000000:0040143c               8b 04 84           MOVL   0(%RSP, %RAX, 4), %EAX
00000000:0040143f               01 c0              ADDL   %EAX, %EAX
00000000:00401441               39 04 94           CMPL   %EAX, 0(%RSP, %RDX, 4)
00000000:00401444               74 e6              JE     0x40142C
00000000:00401446               eb df              JMP    0x401427
00000000:00401448               48 83 c4 20        ADDQ   $0x20, %RSP
00000000:0040144c               5b                 POPQ   %RBX
00000000:0040144d               c3                 RETQ
```

并且上面分析可知：bomb!phase_2 只关心我们输入的前 6 个数字是否满足要求，并不关心之后的数字。例如下图所示：

```
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
I am just a renegade hockey mom.
Phase 1 defused. How about the next one?
1 2 4 8 16 32 54
That's number 2.  Keep going!
```

3.3 阶段 3 的破解与分析

密码如下：0 240

破解过程：

由 bomb!phase_3 前面的汇编代码分析知，首先会检测是否输入了大于等于两个数字，如果少于两个数字则炸弹引爆。间接地，告诉我们了 phase_3 的密码是两个数字，为了叙述方便，我们记为 a、b。

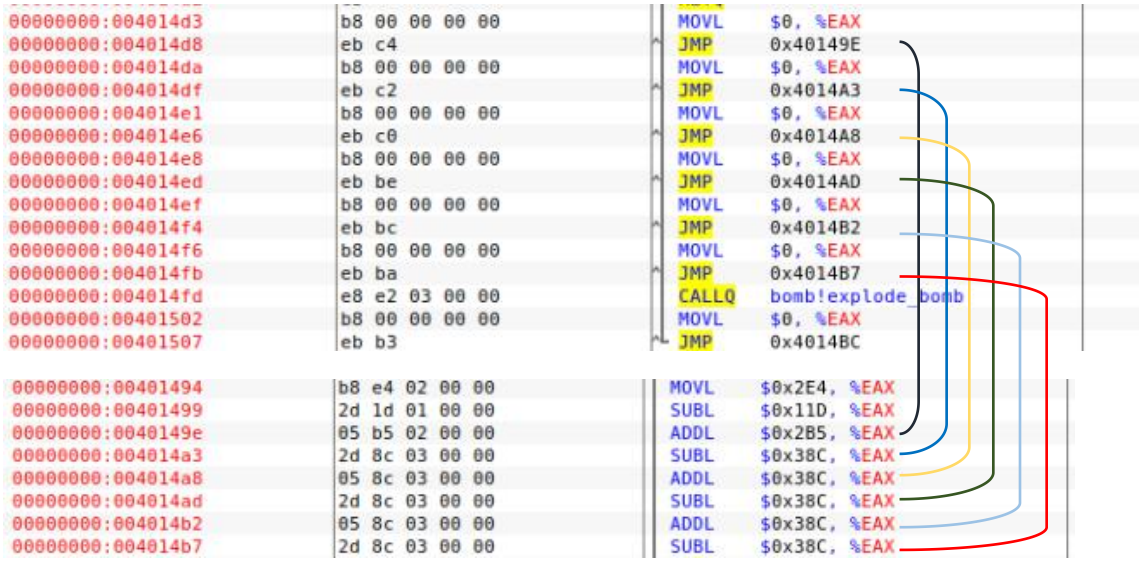
```
00000000:0040144e bomb!phase_3   48 83 ec 18      SUBQ   $0x18, %RSP
00000000:00401452                48 8d 4c 24 08   LEAQ   8(%RSP), %RCX
00000000:00401457                48 8d 54 24 0c   LEAQ   0xC(%RSP), %RDX
00000000:0040145c                be 0f 33 40 00   MOVL   $0x40330F, %ESI        ASCII "%d %d"
00000000:00401461                b8 00 00 00 00   MOVL   $0, %EAX
00000000:00401466                e8 b5 fc ff ff   CALLQ  bomb!__isoc99_sscanf@plt
00000000:0040146b                83 f8 01         CMPL   $1, %EAX
00000000:0040146e                7e 16            JLE    0x401486
```

我们分析下面这段代码。我们输入两个数字在栈内分别在 0xC(%rsp)和 8(%rsp)中。由分析知（见图中的 comments），a 是小于等于 5 的，我们可以输入 1~5 任意数字，当然第一个数字不同，对应的第二个数字也不同，不过我们可以把所有的组合都求解出来。

```
00000000:00401470   83 7c 24 0c 07      CMPL   $7, 0xC(%RSP)          0xC(%rsp)<=7
00000000:00401475   0f 87 82 00 00 00   JA     0x4014FD
00000000:0040147b   8b 44 24 0c         MOVL   0xC(%RSP), %EAX
00000000:0040147f   ff 24 c5 b0 31 40 00 JMPQ   *0x4031B0(, %RAX, 8)
00000000:00401486   e8 59 04 00 00      CALLQ  bomb!explode_bomb
00000000:0040148b   eb e3               JMP    0x401470               bomb
00000000:0040148d   b8 00 00 00 00      MOVL   $0, %EAX
00000000:00401492   eb 05               JMP    0x401499
00000000:00401494   b8 e4 02 00 00      MOVL   $0x2E4, %EAX
00000000:00401499   2d 1d 01 00 00      SUBL   $0x11D, %EAX
00000000:0040149e   05 b5 02 00 00      ADDL   $0x2B5, %EAX
00000000:004014a3   2d 8c 03 00 00      SUBL   $0x38C, %EAX
00000000:004014a8   05 8c 03 00 00      ADDL   $0x38C, %EAX
00000000:004014ad   2d 8c 03 00 00      SUBL   $0x38C, %EAX
00000000:004014b2   05 8c 03 00 00      ADDL   $0x38C, %EAX
00000000:004014b7   2d 8c 03 00 00      SUBL   $0x38C, %EAX
00000000:004014bc   83 7c 24 0c 05      CMPL   $5, 0xC(%RSP)          0xC(%rsp)<=5
00000000:004014c1   7f 06               JG     0x4014C9               bomb
00000000:004014c3   39 44 24 08         CMPL   %EAX, 8(%RSP)          %eax=%8(%rsp)
00000000:004014c7   74 05               JE     0x4014CE
00000000:004014c9   e8 16 04 00 00      CALLQ  bomb!explode_bomb
00000000:004014ce   48 83 c4 18         ADDQ   $0x18, %RSP
00000000:004014d2   c3                  RETQ
```

这两步是有趣的，我们输入不同的 a 的会得到不同的跳转地址，可以在内存中查看它们（这里给出了 8 个跳转地址，其实到这一步之前 a 是可以取 0~7 的，不过再看后面的代码发现当取 6 和 7 时会被炸弹炸到）。

```
00000000:004031b0 94 14 40 00 00 00 00 00 8d 14 40 00 00 00 00 00  ..@.......@.....
00000000:004031c0 d3 14 40 00 00 00 00 00 da 14 40 00 00 00 00 00  ┐.@.....─.@.....
00000000:004031d0 e1 14 40 00 00 00 00 00 e8 14 40 00 00 00 00 00  £.@.....|.@.....
00000000:004031e0 ef 14 40 00 00 00 00 00 f6 14 40 00 00 00 00 00  ∩.@..... .@.....
```



当 a = 0 时，b = 240;

当 a = 1 时，b = -500;

当 a = 2 时，b = -215;

当 a = 3 时，b = -908;

当 a = 4 时，b = 0;

当 a = 5 时，b = -908;

以上是 bomb!phase_3 的所有密码

3.4 阶段 4 的破解与分析

密码如下：108 2

破解过程：

bomb!phase_4 是 bomb!phase_3 的升级版，增加了一个递归的函数。我们将输入的两个数记为 a、b。由分析可知， b=2 或 3（分析方法与 bomb!phase_3 相同）。下面我们分析 bomb!phase_4 的重点——bomb!fun4 函数

```
00000000:00401509 bomb!func4      85 ff          TESTL   %EDI, %EDI
00000000:0040150b                 7e 2b          JLE     0x401538
00000000:0040150d                 83 ff 01       CMPL    $1, %EDI
00000000:00401510                 74 2c          JE      0x40153E
00000000:00401512                 41 54          PUSHQ   %R12
00000000:00401514                 55             PUSHQ   %RBP
00000000:00401515                 53             PUSHQ   %RBX
00000000:00401516                 89 f3          MOVL    %ESI, %EBX
00000000:00401518                 89 fd          MOVL    %EDI, %EBP
00000000:0040151a                 8d 7f ff       LEAL    -1(%RDI), %EDI
00000000:0040151d                 e8 e7 ff ff ff CALLQ   bomb!func4
00000000:00401522                 44 8d 24 18    LEAL    0(%RAX, %RBX), %R12D
00000000:00401526                 8d 7d fe       LEAL    -2(%RBP), %EDI
00000000:00401529                 89 de          MOVL    %EBX, %ESI
00000000:0040152b                 e8 d9 ff ff ff CALLQ   bomb!func4
00000000:00401530                 44 01 e0       ADDL    %R12D, %EAX
00000000:00401533                 5b             POPQ    %RBX
00000000:00401534                 5d             POPQ    %RBP
00000000:00401535                 41 5c          POPQ    %R12
00000000:00401537                 c3             RETQ
00000000:00401538                 b8 00 00 00 00 MOVL    $0, %EAX
00000000:0040153d                 c3             RETQ
00000000:0040153e                 89 f0          MOVL    %ESI, %EAX
00000000:00401540                 c3             RETQ
```

其实这个函数的 C 程序可以写成如下：

```
int bomb_fun4(int x, int y){
    if(x <= 0)
        return    0;
    if(x == 1)
        return    y;
    if(x >= 1)
        return    bomb_fun4 (x-1,y)+ bomb_fun4 (x-2,y)+y;
}
```

其中 x=8，y=b

当 b=2 时，a= bomb_fun4(8,2)=108；

当 b=3 时，a= bomb_fun4(8,3)=162；

故 bomb!phase_4 的密码为 108 2 和 162 3

3.5 阶段 5 的破解与分析

密码如下：beldog

破解过程：

我们现在已经很熟悉这种套路了，bomb!phase_5 需要输入一个长度为 6 的字符串（bomb!string_length 会检查字符串的长度）

```
00000000:00401596 bomb!phase_5   53              PUSHQ   %RBX
00000000:00401597                48 83 ec 10     SUBQ    $0x10, %RSP
00000000:0040159b                48 89 fb        MOVQ    %RDI, %RBX
00000000:0040159e                e8 48 02 00 00  CALLQ   bomb!string_length
00000000:004015a3                83 f8 06        CMPL    $6, %EAX                len=6
00000000:004015a6                75 24           JNE     0x4015CC                bomb
```

下面是一个循环操作。其含义是：取每个字符 ASCII 码的低位，加上 0x4031f0 得到内存地址并将其内容依次压栈。

```
00000000:004015a8  b8 00 00 00 00   MOVL    $0, %EAX
00000000:004015ad  83 f8 05         CMPL    $5, %EAX
00000000:004015b0  7f 21            JG      0x4015D3
00000000:004015b2  48 63 c8         MOVSLQ  %EAX, %RCX
00000000:004015b5  0f b6 14 0b      MOVZBL  0(%RBX, %RCX), %EDX
00000000:004015b9  83 e2 0f         ANDL    $0xF, %EDX
00000000:004015bc  0f b6 92 f0 31 40 00  MOVZBL  0x4031F0(%RDX), %EDX
00000000:004015c3  88 54 0c 09      MOVB    %DL, 9(%RSP, %RCX)
00000000:004015c7  83 c0 01         ADDL    $1, %EAX
00000000:004015ca  eb e1            JMP     0x4015AD
00000000:004015cc  e8 13 03 00 00   CALLQ   bomb!explode_bomb
00000000:004015d1  eb d5            JMP     0x4015A8
00000000:004015d3  c6 44 24 0f 00   MOVB    $0, 0xF(%RSP)
00000000:004015d8  be 9e 31 40 00   MOVL    $0x40319E, %ESI
00000000:004015dd  48 8d 7c 24 09   LEAQ    9(%RSP), %RDI
00000000:004015e2  e8 18 02 00 00   CALLQ   bomb!strings_not_equal
00000000:004015e7  85 c0            TESTL   %EAX, %EAX
00000000:004015e9  75 06            JNE     0x4015F1
00000000:004015eb  48 83 c4 10      ADDQ    $0x10, %RSP
00000000:004015ef  5b               POPQ    %RBX
00000000:004015f0  c3               RETQ
00000000:004015f1  e8 ee 02 00 00   CALLQ   bomb!explode_bomb
00000000:004015f6  eb f3            JMP     0x4015EB
```

我们得到的字符串要和存储在 0x40319e 的字符串进行比较，即下图：

```
00000000:0040319e  64 65 76 69 6c 73 00 00 00 00 00 00 00 00 00 00  devils..........
```

也就是对应 ASCII 码相同。我们再看 0x4031f0(%rdx)，当%rdx 为 0~f 时依次对应下图 ACSII 码，所以我们只需按 64 54 76 69 6c 73 的顺序依次找到它们即可。然后再查 ASCII 表，看低位为该情况时对应哪个字符即可。所以可以得到许多组合，这里不一一列举。

```
00000000:004031f0  6d 61 64 75 69 65 72 73 6e 66 6f 74 76 62 79 6c
```

3.6 阶段 6 的破解与分析

密码如下：1 5 4 6 3 2

破解过程：

bomb!read_six_numbers 函数我们在 phase_2 中已经遇到了，它要求我们输入大于等于 6 个字符的字符串。

```
00000000:004015f8 bomb!phase_6    41 54          PUSHQ   %R12
00000000:004015fa                 55             PUSHQ   %RBP
00000000:004015fb                 53             PUSHQ   %RBX
00000000:004015fc                 48 83 ec 50    SUBQ    $0x50, %RSP
00000000:00401600                 48 8d 74 24 30 LEAQ    0x30(%RSP), %RSI
00000000:00401605                 e8 fc 02 00 00 CALLQ   bomb!read_six_numbers
00000000:0040160a                 bd 00 00 00 00 MOVL    $0, %EBP
00000000:0040160f                 eb 29          JMP     0x40163A
00000000:00401611                 e8 ce 02 00 00 CALLQ   bomb!explode_bomb
00000000:00401616                 eb 36          JMP     0x40164E
00000000:00401618                 e8 c7 02 00 00 CALLQ   bomb!explode_bomb
```

下面部分的代码有两个功能：判断输入的 6 个数不能相等，且均需小于等于 6。

```
00000000:0040161d 83 c3 01       ADDL    $1, %EBX
00000000:00401620 83 fb 05       CMPL    $5, %EBX
00000000:00401623 7f 12          JG      0x401637
00000000:00401625 48 63 c5       MOVSLQ  %EBP, %RAX
00000000:00401628 48 63 d3       MOVSLQ  %EBX, %RDX
00000000:0040162b 8b 7c 94 30    MOVL    0x30(%RSP, %RDX, 4), %EDI
00000000:0040162f 39 7c 84 30    CMPL    %EDI, 0x30(%RSP, %RAX, 4)
00000000:00401633 75 e8          JNE     0x40161D
00000000:00401635 eb e1          JMP     0x401618                    bomb
00000000:00401637 44 89 e5       MOVL    %R12D, %EBP
00000000:0040163a 83 fd 05       CMPL    $5, %EBP
00000000:0040163d 7f 18          JG      0x401657
00000000:0040163f 48 63 c5       MOVSLQ  %EBP, %RAX
00000000:00401642 8b 44 84 30    MOVL    0x30(%RSP, %RAX, 4), %EAX
00000000:00401646 83 e8 01       SUBL    $1, %EAX
00000000:00401649 83 f8 05       CMPL    $5, %EAX
00000000:0040164c 77 c3          JA      0x401611                    bomb
00000000:0040164e 44 8d 65 01    LEAL    1(%RBP), %R12D
00000000:00401652 44 89 e3       MOVL    %R12D, %EBX
00000000:00401655 eb c9          JMP     0x401620
```

将 0x004052d0~0x00405320 按照我们输入的大小顺序压入栈中。

```
00000000:00401657 be 00 00 00 00 MOVL    $0, %ESI
00000000:0040165c eb 17          JMP     0x401675
00000000:0040165e 48 8b 52 08    MOVQ    8(%RDX), %RDX
00000000:00401662 83 c0 01       ADDL    $1, %EAX
00000000:00401665 48 63 ce       MOVSLQ  %ESI, %RCX
00000000:00401668 39 44 8c 30    CMPL    %EAX, 0x30(%RSP, %RCX, 4)
00000000:0040166c 7f f0          JG      0x40165E
00000000:0040166e 48 89 14 cc    MOVQ    %RDX, 0(%RSP, %RCX, 8)
00000000:00401672 83 c6 01       ADDL    $1, %ESI
00000000:00401675 83 fe 05       CMPL    $5, %ESI
00000000:00401678 7f 0c          JG      0x401686
00000000:0040167a b8 01 00 00 00 MOVL    $1, %EAX
00000000:0040167f ba d0 52 40 00 MOVL    $0x4052D0, %EDX
00000000:00401684 eb df          JMP     0x401665
```

这部分是将内存中存储内容按有小到大的进行比较。其顺序即为最终的密码。

```
00000000:004016b2  bd 00 00 00 00    MOVL   $0, %EBP
00000000:004016b7  eb 07             JMP    0x4016C0
00000000:004016b9  48 8b 5b 08       MOVQ   8(%RBX), %RBX
00000000:004016bd  83 c5 01          ADDL   $1, %EBP
00000000:004016c0  83 fd 04          CMPL   $4, %EBP
00000000:004016c3  7f 11             JG     0x4016D6
00000000:004016c5  48 8b 43 08       MOVQ   8(%RBX), %RAX
00000000:004016c9  8b 00             MOVL   0(%RAX), %EAX
00000000:004016cb  39 03             CMPL   %EAX, 0(%RBX)
00000000:004016cd  7e ea             JLE    0x4016B9
00000000:004016cf  e8 10 02 00 00    CALLQ  bomb!explode_bomb
00000000:004016d4  eb e3             JMP    0x4016B9
```

```
00000000:004052d8 e0 52 40 00 00 00 00 00
00000000:004052e0 a2 03 00 00 02 00 00 00
00000000:004052e8 f0 52 40 00 00 00 00 00
00000000:004052f0 32 03 00 00 03 00 00 00
00000000:004052f8 00 53 40 00 00 00 00 00
00000000:00405300 bb 02 00 00 04 00 00 00
00000000:00405308 10 53 40 00 00 00 00 00
00000000:00405310 00 02 00 00 05 00 00 00
00000000:00405318 20 53 40 00 00 00 00 00
00000000:00405320 c2 02 00 00 06 00 00 00
```

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

4.2 请给出对本次实验内容的建议

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.