

# 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 计算机类

学    号 1161800218

班    级 1636101

学    生 陈翔

指 导 教 师 史先俊

实 验 地 点 G709

实 验 日 期 2017.10.

计算机科学与技术学院

# 目 录

<b>第 1 章 实验基本信息</b>	<b>- 3 -</b>
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 -
<b>第 2 章 实验环境建立</b>	<b>- 5 -</b>
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 -
<b>第 3 章 各阶段炸弹破解与分析</b>	<b>- 7 -</b>
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	- 10 -
3.5 阶段 5 的破解与分析	- 11 -
3.6 阶段 6 的破解与分析	- 14 -
3.7 阶段 7 的破解与分析(隐藏阶段)	- 17 -
<b>第 4 章 总结</b>	<b>- 20 -</b>
4.1 请总结本次实验的收获	- 20 -
4.2 请给出对本次实验内容的建议	- 20 -
<b>参考文献</b>	<b>- 21 -</b>

## 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

Visual Studio 2010 64 位以上; GDB/OBJDUMP; 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。

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

## 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。

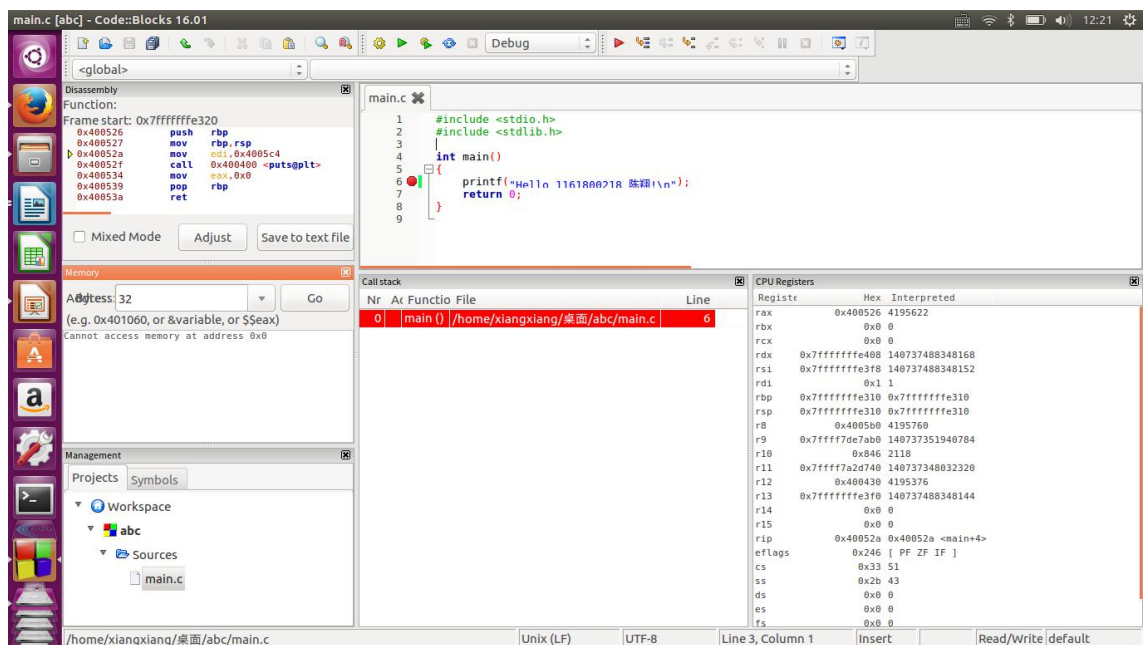


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

## 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

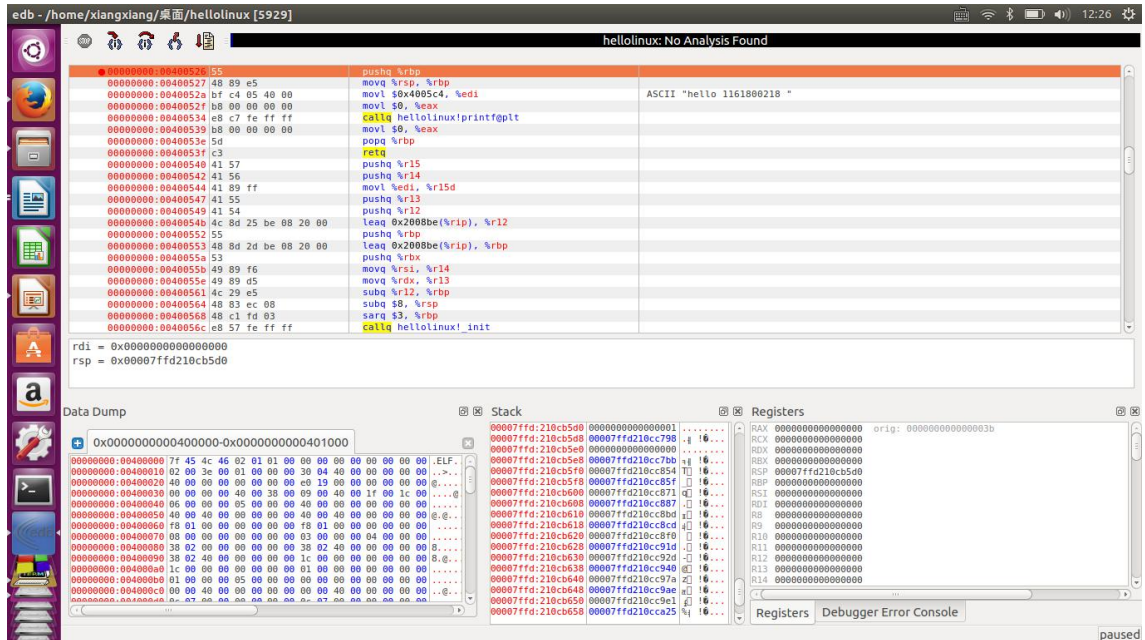


图 2-2 Ubuntu 下 EDB 截图

## 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 for medical liability at the federal level.

破解过程：放入栈中两个字符串，然后比较它们是否相等

比较原有字符串和输入的字符串是否一样

而栈中存放的是 I am for medical liability at the federal level.

如果输入的字符串和它相等炸弹就不会爆炸；否则 call strings\_not\_equal 这个函数，炸弹就炸了

subq \$8, %rsp	
movl \$0x402410, %esi	ASCII "I am for medical liability at the federal level."
callq bomb!strings_not_equal	
testl %eax, %eax	
je 0x400ea4	
callq bomb!explode_bomb	
addq \$8, %rsp	
retq	

### 3.2 阶段 2 的破解与分析

密码如下：1 2 4 7 11 16

破解过程：一个循环语句，共作了 5 次，可以看到址就差 0x4，也就是一个 32 位整数的地址，所以相邻两个数的一个比较。

ebx=1;

eax=ebx=1;

eax+=[rbp]=1+1=2;

ebx+=1;

eax=ebx=2;

eax+=[rbp]=2+2=4;

```

ebx+=1;

eax=ebx=3;

eax+=[rbp]=4+3=7;

ebx+=1;

eax=ebx=4;

eax+=[rbp]=7+4=11;

ebx=1;

eax=ebx=5;

eax+=[rbp]=11+5=16;

ebx+=1;

```

所以答案就是 1\_2\_4\_7\_11\_16

```

pushq %rbp
pushq %rbx
subq $0x28, %rsp
movq %fs:0x28, %rax
movq %rax, 0x18(%rsp)
xorl %eax, %eax
movq %rsp, %rsi
callq bomb!read_six_numbers
cmpl $0, (%rsp)
jns 0x400ed2
callq bomb!explode_bomb
movq %rsp, %rbp
movl $1, %ebx
movl %ebx, %eax
addl (%rbp), %eax
cmpl %eax, 4(%rbp)
je 0x400ee9
callq bomb!explode_bomb
addl $1, %ebx
addq $4, %rbp
cmpl $6, %ebx
jne 0x400eda
movq 0x18(%rsp), %rax
xorq %fs:0x28, %rax
je 0x400f0a
callq bomb!__stack_chk_fail@plt
addq $0x28, %rsp
popq %rbx
popq %rbp
retq

```

### 3.3 阶段 3 的破解与分析

密码如下：0 d 204

破解过程：switch 语句



读取 3 个参数，

第一个参数是 0 时

第三个参数 0xcc=204

eax=0x64=100

al=100= 'd' (ASCII 码为 100)

所以第二个参数是 d

答案为:0 d 204 (答案不唯一，同理可以找到另外几个)

subq \$0x28, %rsp	
movq %fs:0x28, %rax	
movq %rax, 0x18(%rsp)	
xorl %eax, %eax	
leaq 0x14(%rsp), %r8	
leaq 0xf(%rsp), %rcx	
leaq 0x10(%rsp), %rdx	
movl \$0x40246e, %esi	ASCII "%d %c %d"
callq bomb!_isoc99_sscanf@plt	
cmpl \$2, %eax	
jg 0x400f48	
callq bomb!explode_bomb	
cmpl \$7, 0x10(%rsp)	
ja 0x401048	
movl 0x10(%rsp), %eax	
jmpq *0x402480(, %rax, 8)	
movl \$0x64, %eax	
cmpl \$0xcc, 0x14(%rsp)	
je 0x401052	
callq bomb!explode_bomb	
movl \$0x64, %eax	
jmp 0x401052	
movl \$0x64, %eax	
cmpl \$0x35e, 0x14(%rsp)	
je 0x401052	
callq bomb!explode_bomb	
movl \$0x64, %eax	
jmp 0x401052	
movl \$0x76, %eax	
cmpl \$0x318, 0x14(%rsp)	
je 0x401052	
callq bomb!explode_bomb	
movl \$0x76, %eax	
jmp 0x401052	
movl \$0x62, %eax	
cmpl \$0xa8, 0x14(%rsp)	
je 0x401052	
callq bomb!explode_bomb	
movl \$0x62, %eax	
jmp 0x401052	
movl \$0x73, %eax	
cmpl \$0x3d, 0x14(%rsp)	

```

v je 0x401052
  callq bomb!explode_bomb
  movl $0x73, %eax
v jmp 0x401052
  movl $0x6a, %eax
  cmpl $0x19e, 0x14(%rsp)
v je 0x401052
  callq bomb!explode_bomb
  movl $0x6a, %eax
v jmp 0x401052
  movl $0x64, %eax
  cmpl $0x38b, 0x14(%rsp)
v je 0x401052
  callq bomb!explode_bomb
  movl $0x64, %eax
v jmp 0x401052
  movl $0x64, %eax
  cmpl $0xc5, 0x14(%rsp)
v je 0x401052
  callq bomb!explode_bomb
  movl $0x64, %eax
v jmp 0x401052
  callq bomb!explode_bomb
  movl $0x61, %eax
  cmpb 0xf(%rsp), %al
v je 0x40105d
  callq bomb!explode_bomb
  movq 0x18(%rsp), %rax
  xorq %fs:0x28, %rax
v je 0x401072
  callq bomb!__stack_chk_fail@plt
  addq $0x28, %rsp
  retq

```

### 3.4 阶段 4 的破解与分析

密码如下：13 31 DrEvil（第七关隐藏关卡的字符串）

破解过程：递归

eax==31

func4(-0x4(%rsp))==31

fun(11)=31

所以答案是：13 31

subq \$0x18, %rsp	
movq %fs:0x28, %rax	
movq %rax, 8(%rsp)	
xorl %eax, %eax	
leaq 4(%rsp), %rcx	
movq %rsp, %rdx	
movl \$0x40260f, %esi	ASCII "%d %d"
callq bomb!__isoc99_sscanf@plt	
cmpl \$2, %eax	
jne 0x4010db	
cmpl \$0xe, (%rsp)	
jbe 0x4010e0	
callq bomb!explode_bomb	
movl \$0xe, %edx	
movl \$0, %esi	
movl (%rsp), %edi	
callq bomb!func4	
cmpl \$0x1f, %eax	
jne 0x4010fe	
cmpl \$0x1f, 4(%rsp)	
je 0x401103	
callq bomb!explode_bomb	
movq 8(%rsp), %rax	
xorq %fs:0x28, %rax	
je 0x401118	
callq bomb!__stack_chk_fail@plt	
addq \$0x18, %rsp	
retq	

func4()

pushq %rbx
movl %edx, %eax
subl %esi, %eax
movl %eax, %ebx
shrl \$0x1f, %ebx
addl %ebx, %eax
sarl \$1, %eax
leal (%rax, %rsi), %ebx
cmpl %edi, %ebx
jle 0x401098
leal -1(%rbx), %edx
callq bomb!func4
addl %ebx, %eax
jmp 0x4010a8
movl %ebx, %eax
cmpl %edi, %ebx
jge 0x4010a8
leal 1(%rbx), %esi
callq bomb!func4
addl %ebx, %eax
popq %rbx
retq

### 3.5 阶段 5 的破解与分析

密码如下：2233NN

破解过程:

```
pushq %rbx
movq %rdi, %rbx
callq bomb!string_length
cmpl $6, %eax
v je 0x401130
callq bomb!explode_bomb
movq %rbx, %rax
leaq 6(%rbx), %rdi
movl $0, %ecx
movzbl (%rax), %edx
andl $0xf, %edx
addl 0x4024c0(, %rdx, 4), %ecx
addq $1, %rax
cmpq %rdi, %rax
^ jne 0x40113c
cmpl $0x2c, %ecx
v je 0x40115c
callq bomb!explode_bomb
popq %rbx
retq
```

先调用<string\_length>函数获得字符串的长度，如果等于 6 就继续往下做，如果不等于 6 就直接调用<explode\_bomb>，炸了。所以字符串的长度应当等于 6 然后是一个做了 6 次的循环，每次循环对应一个字符。变量 a 就是这 6 次循环算出的结果，那每次循环具体做了什么呢？单步完成这条代码后，eax 内容刚好是对应字符的 ASCII 码，然后下一步是将 eax 的内容和 0xf 按位相与。这两行的意思其实就是先获得字符的 ASCII 码，然后%16。再往下看，将一个静态数组中的某个值放入%edx，再累加到变量 a 中，数组中的索引和上一步%16 的余数相对应。先找到 0x4024c0，再将这个静态数组打印出来：

```

(gdb) x 0x4024c0
0x4024c0 <array.3600>: 0x00000002
(gdb) x
0x4024c4 <array.3600+4>: 0x0000000a
(gdb)
0x4024c8 <array.3600+8>: 0x00000006
(gdb)
0x4024cc <array.3600+12>: 0x00000001
(gdb)
0x4024d0 <array.3600+16>: 0x0000000c
(gdb)
0x4024d4 <array.3600+20>: 0x00000010
(gdb)
0x4024d8 <array.3600+24>: 0x00000009
(gdb)
0x4024dc <array.3600+28>: 0x00000003
(gdb)
0x4024e0 <array.3600+32>: 0x00000004
(gdb)
0x4024e4 <array.3600+36>: 0x00000007
(gdb)
0x4024e8 <array.3600+40>: 0x0000000e
(gdb)
0x4024ec <array.3600+44>: 0x00000005
(gdb)
0x4024f0 <array.3600+48>: 0x0000000b
(gdb)
0x4024f4 <array.3600+52>: 0x00000008
(gdb)
0x4024f8 <array.3600+56>: 0x0000000f
(gdb)
0x4024fc <array.3600+60>: 0x0000000d

```

%16 余数	数组变量	值
0	<array.0>	0x2
1	<array.0+4>	0xa
2	<array.0+8>	0x6
3	<array.0+12>	0x1
4	<array.0+16>	0xc
5	<array.0+20>	0x10
6	<array.0+24>	0x9
7	<array.0+28>	0x3

8	<array.0+32>	0x4
9	<array.0+36>	0x7
10	<array.0+40>	0xe
11	<array.0+44>	0x5
12	<array.0+48>	0xb
13	<array.0+52>	0x8
14	<array.0+56>	0xf
15	<array.0+60>	0xd

我们需要从中找到 6 个数的和刚好是  $0x2c=44$

可以找到  $1+1+f+f+6+6=44$

所以答案是 2233NN（不唯一）

### 3.6 阶段 6 的破解与分析

密码如下：3 1 6 5 2 4

破解过程：

这段代码要求输入 6 个数字，然后经过处理得到一个 6 个元素的链表，最后进行比较。

第一个循环，保证 6 个数字都大于 0、小于等于 6、互不相等。这就是说 6 个数字分别是 1、2、3、4、5、6，我们要做的就是确定它们的顺序。

每个元素是一个 struct（称之为“Node”），每个 Node 有三个成员，我们依次称之为 int a、int b、Node\*next。变量 b 是我们输入的数字，变量 a 是根据 b 算出的数值，指针 next 指向下一个元素，Node 的结构如下代码所示。整个链表包括 6 个元素，逻辑顺序为 node1->node2->node3->node4->node5->node6。

<node 1> 34a 1 ②

<node 1> 34a 1 ⑤

<node 1> 34a 1 ①

<node 1> 34a 1 ⑥



<node 1> 34a 1 ④

<node 1> 34a 1 ③

所以排序得到 node3->node1->node6->node5->node2->node4

于是答案就是 3 1 6 5 2 4

```
(gdb) x 0x6032f0
0x6032f0 <node1>:      0x0000034a
(gdb)
0x6032f4 <node1+4>:    0x00000001
(gdb)
Firefox 网络浏览器 >: 0x00603300
(gdb)
0x6032fc <node1+12>:   0x00000000
(gdb)
0x603300 <node2>:      0x000000d0
(gdb)
0x603304 <node2+4>:    0x00000002
(gdb)
0x603308 <node2+8>:    0x00603310
(gdb)
0x60330c <node2+12>:   0x00000000
(gdb)
0x603310 <node3>:      0x000003b7
(gdb)
0x603314 <node3+4>:    0x00000003
(gdb)
0x603318 <node3+8>:    0x00603320
(gdb)
0x60331c <node3+12>:   0x00000000
(gdb)
0x603320 <node4>:      0x0000009a
(gdb)
0x603324 <node4+4>:    0x00000004
(gdb)
0x603328 <node4+8>:    0x00603330
(gdb)
0x60332c <node4+12>:   0x00000000
(gdb)
0x603330 <node5>:      0x000001f8
(gdb)
0x603334 <node5+4>:    0x00000005
(gdb)
0x603338 <node5+8>:    0x00603340
(gdb)
0x60333c <node5+12>:   0x00000000
(gdb)
0x603340 <node6>:      0x000002a1
```

```
pushq %r13
pushq %r12
pushq %rbp
pushq %rbx
subq $0x68, %rsp
movq %fs:0x28, %rax
movq %rax, 0x58(%rsp)
xorl %eax, %eax
movq %rsp, %rsi
callq bomb!read_six_numbers
movq %rsp, %r12
movl $0, %r13d
movq %r12, %rbp
movl (%r12), %eax
subl $1, %eax
cmpl $5, %eax
jbe 0x40119d
callq bomb!explode_bomb
addl $1, %r13d
cmpl $6, %r13d
je 0x4011e4
movl %r13d, %ebx
movslq %ebx, %rax
movl (%rsp, %rax, 4), %eax
cmpl %eax, (%rbp)
jne 0x4011ba
callq bomb!explode_bomb
addl $1, %ebx
cmpl $5, %ebx
jle 0x4011aa
addq $4, %r12
jmp 0x401189
movq 8(%rdx), %rdx
addl $1, %eax
cmpl %ecx, %eax
jne 0x4011c8
movq %rdx, 0x20(%rsp, %rsi, 2)
addq $4, %rsi
cmpq $0x18, %rsi
jne 0x4011e9
jmp 0x4011fd
movl $0, %esi
```



```

movl $0, %esi
movl (%rsp, %rsi), %ecx
movl $1, %eax
movl $0x6032f0, %edx
cmpl $1, %ecx
\ jg 0x4011c8
\ jmp 0x4011d3
movq 0x20(%rsp), %rbx
leaq 0x20(%rsp), %rax
leaq 0x48(%rsp), %rsi
movq %rbx, %rcx
movq 8(%rax), %rdx
movq %rdx, 8(%rcx)
addq $8, %rax
movq %rdx, %rcx
cmpq %rsi, %rax
\ jne 0x40120f
movq $0, 8(%rdx)
movl $5, %ebp
movq 8(%rbx), %rax
movl (%rax), %eax
cmpl %eax, (%rbx)
\ jge 0x40123f
callq bomb!explode_bomb
movq 8(%rbx), %rbx
subl $1, %ebp
\ jne 0x401230
movq 0x58(%rsp), %rax
xorq %fs:0x28, %rax
\ je 0x40125d
callq bomb!__stack_chk_fail@plt
addq $0x68, %rsp
popq %rbx
popq %rbp
popq %r12
popq %r13
retq

```

### 3.7 阶段 7 的破解与分析(隐藏阶段)

密码如下：35

破解过程：

首先要发现隐藏关卡，那你就得通读代码，所以把代码打在纸上能更清晰些。其次，如何调出关卡就得分析 `secret_phase` 在哪里被调用了，可以发现其被调用的地方只有一处，在 `phase_defused` 的代码中，而 `phase_defused` 是每关拆完之后会被调用的。于是拿来 `phase_defused` 分析一下：

subq \$0x78, %rsp	
movq %fs:0x28, %rax	
movq %rax, 0x68(%rsp)	
xorl %eax, %eax	
cmpl \$6, 0x20215b(%rip)	
jne 0x401691	
leaq 0x10(%rsp), %r8	
leaq 0xc(%rsp), %rcx	
leaq 8(%rsp), %rdx	
movl \$0x402659, %esi	ASCII "%d %d %s"
movl \$0x603890, %edi	
callq bomb!_isoc99_sscanf@plt	
cmpl \$3, %eax	
jne 0x401687	
movl \$0x402662, %esi	ASCII "DrEvil"
leaq 0x10(%rsp), %rdi	
callq bomb!strings_not_equal	
testl %eax, %eax	
jne 0x401687	
movl \$0x402538, %edi	ASCII "Curses, you've found the secret phase!"
callq bomb!puts@plt	
movl \$0x402560, %edi	ASCII "But finding it and solving it are quite different..."
callq bomb!puts@plt	
movl \$0, %eax	
callq bomb!secret_phase	
movl \$0x402598, %edi	ASCII "Congratulations! You've defused the bomb!"
callq bomb!puts@plt	
movq 0x68(%rsp), %rax	
xorq %fs:0x28, %rax	
je 0x4016a6	
callq bomb!_stack_chk_fail@plt	
addq \$0x78, %rsp	
retq	

可以看到，必须要过了第 6 关隐藏关卡才能被调用

在第四关答案后边加字符串 “DrEvil”

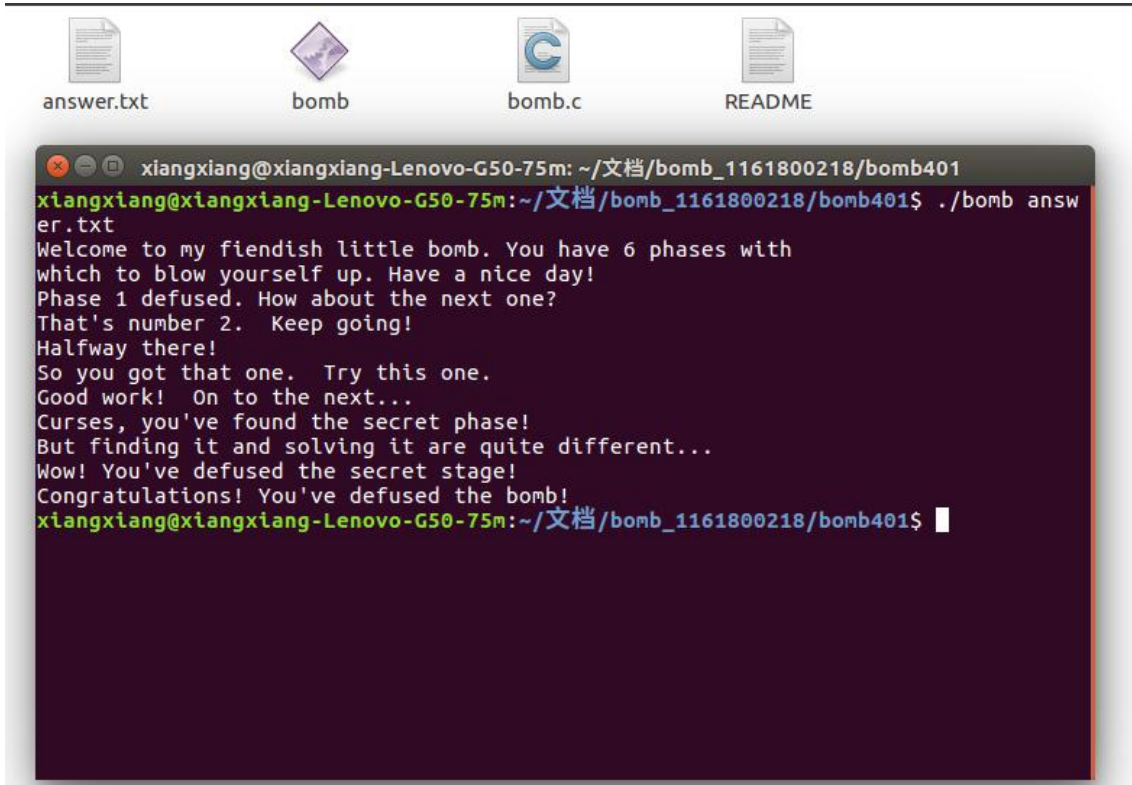
查看 fun7 的代码

subq \$8, %rsp	
testq %rdi, %rdi	
je 0x40129c	
movl (%rdi), %edx	
cmpl %esi, %edx	
jle 0x401284	
movq 8(%rdi), %rdi	
callq bomb!fun7	
addl %eax, %eax	
jmp 0x4012a1	
movl \$0, %eax	
cmpl %esi, %edx	
je 0x4012a1	
movq 0x10(%rdi), %rdi	
callq bomb!fun7	
leal 1(%rax, %rax), %eax	
jmp 0x4012a1	
movl \$0xffffffff, %eax	
addq \$8, %rsp	
retq	
pushq %rbx	
callq bomb!read_line	
movl \$0xa, %edx	
movl \$0, %esi	
movq %rax, %rdi	
callq bomb!strtoul@plt	
movq %rax, %rbx	
leal -1(%rax), %eax	
cmpl \$0x3e8, %eax	
jbe 0x4012d0	
callq bomb!explode_bomb	
movl %ebx, %esi	
movl \$0x603110, %edi	
callq bomb!fun7	
cmpl \$6, %eax	
je 0x4012e6	
callq bomb!explode_bomb	
movl \$0x402448, %edi	ASCII "Wow! You've defused the secret stage!"
callq bomb!puts@plt	
callq bomb!phase_defused	
popq %rbx	
retq	

这是一个加强版的递归汇编，其实原理和 phase\_4 的递归是一样的，只不过稍微分支多了点。若返回值==6，则成功 defused.

穷举得到答案为 35

### 结果展示:



The image shows a file manager interface with four files: `answer.txt`, `bomb`, `bomb.c`, and `README`. Below the file manager is a terminal window with the following output:

```
xiangxiang@xiangxiang-Lenovo-G50-75m: ~/文档/bomb_1161800218/bomb401
xiangxiang@xiangxiang-Lenovo-G50-75m:~/文档/bomb_1161800218/bomb401$ ./bomb answer.txt
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.
Good work! On to the next...
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
xiangxiang@xiangxiang-Lenovo-G50-75m:~/文档/bomb_1161800218/bomb401$
```

## 第 4 章 总结

### 4.1 请总结本次实验的收获

一个二进制文件（可执行文件），共 6 个关卡，每关要输入一个密码才能过关，就像解谜游戏一样，还是很有意思的，同时对于程序（函数，返回值，堆栈的组织，循环，递归，链表，结构体等等）如何运行的有更深入的理解。破解唯一可用的线索就只有这个二进制文件了，对于反汇编能有更深入练习，并且还能熟悉 `gdb`, `objdump` 这类调试工具和反汇编工具。

感觉我的汇编语言也有了一些提高，真正的在“做中学”，收获颇深。

### 4.2 请给出对本次实验内容的建议

很好的，每个人炸弹都不同，反抄袭。

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>.