深圳大学实验报告

课程名称：计算机系统(2)

实验项目名称：逆向工程实验

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：刘 刚

报告人：林宪亮 学号：2022150130 班级：国际班

实验时间：2024 年 5 月 14 日～ 2024 年 5 月 17 日

实验报告提交时间：2024 年 5 月 16 日

教务处制

一、实验目的：

1. 理解程序（控制语句、函数、返回值、堆栈结构）是如何运行的
2. 掌握 GDB 调试工具和 objdump 反汇编工具

二、实验环境：

1. 计算机（Intel CPU）
2. Linux64 位操作系统
3. GDB 调试工具
4. objdump 反汇编工具

三、实验内容

本实验设计为一个黑客拆解二进制炸弹的游戏。我们仅给黑客（同学）提供一个二进制可执行文件 bomb 和主函数所在的源程序 bomb.c，不提供每个关卡的源代码。程序运行中有 6 个关卡（6 个 phase），每个关卡需要用户输入正确的字符串或数字才能通关，否则会引爆炸弹（打印出一条错误信息，并导致评分下降）！

要求同学运用 **GDB 调试工具和 objdump 反汇编工具**，通过分析汇编代码，找到在每个 phase 程序段中，引导程序跳转到“explode_bomb”程序段的地方，并分析其成功跳转的条件，以此为突破口寻找应该在命令行输入何种字符串来通关。

本实验要求解决 Phase_1、Phase_2、Phase_3、Phase_4、Phase_5、Phase_6。通过截图把结果写在实验报告上。

四、实验步骤和结果

1. 对二进制文件 bomb_64 进行反汇编，将结果保存到 1.txt。然后打开 1.txt 文件查看反汇编之后的汇编指令，并找到函数 phase1-phase6 所在的位置。图 1 为反汇编指令。

```
lxl@lxl-virtual-machine:~/下载$ objdump -d bomb_64 > 1.txt
```

图 1 反汇编代码

第一关:

下面是第一关对应的汇编代码。

```
0000000000400e70 <phase_1>:
  400e70:    48 83 ec 08          sub    $0x8,%rsp
  400e74:    be f8 1a 40 00       mov    $0x401af8,%esi
  400e79:    e8 bf 03 00 00       call   40123d <strings_not_equal>
  400e7e:    85 c0                test   %eax,%eax
  400e80:    74 05                je     400e87 <phase_1+0x17>
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
400e82:    e8 b6 07 00 00          call    40163d <explode_bomb>
400e87:    48 83 c4 08             add     $0x8,%rsp
400e8b:    c3                      ret
```

我们先逐条语句进行解释分析：

• sub $0x8,%rsp：这条指令将栈指针 %rsp 减去 8 字节，为局部变量或临时数据分配空间。

• mov $0x401af8,%esi：将立即数 0x401af8（一个地址）移动到 %esi 寄存器中。

• call 40123d <strings_not_equal>：调用 strings_not_equal 函数，调用前将当前指令的下一条指令的地址（0x400e7e）压入栈，以便函数返回后继续执行下一条指令。

•test %eax,%eax：将 %eax 寄存器与自身进行与操作，并设置标志位，检查 %eax 是否为零。

•je 400e87 <phase_1+0x17>：如果前一条指令设置的零标志位为真（即 %eax 为零），则跳转到地址 0x400e87 处继续执行，否则执行下一条指令。

call 40163d <explode_bomb>：如果 test 指令的结果不为零，即调用 strings_not_equal 函数返回值不为零，就会调用 explode_bomb 函数。

• add $0x8,%rsp：恢复栈指针，释放之前分配的空间。

• ret：返回函数调用的地方，使用之前压入栈的返回地址。

经过上述分析，可以推断出，破解此关的重点就是地址 0x401af8 存储的值，我们只需要输入一个和地址 0x401af8 存储的值相同的值即可让程序跳转到函数<explode_bomb>从而通过此关。

查询地址 0x401af8 存储的值：

```
(gdb) p(char *)0x401af8
$1 = 0x401af8 "Science isn't about why, it's about why not?"
```

图 2 查询值

即我们只需要输入 Science isn't abut why，it‘s about why not？ 即可通过此关。运行程序并输入 Science isn't abut why，it‘s about why not？，如下图所示

```
(gdb) r
Starting program: /home/lxl/下载/bomb_64
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!


Science isn't about why, it's about why not?
Phase 1 defused. How about the next one?
```

图 3 关 1 通过

如图所示，我们成功通过了第一关。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

第二关：

下面是第二关对应的汇编代码：

```
0000000000400e8c <phase_2>:
   400e8c:      48 89 5c 24 e0              mov      %rbx,-0x20(%rsp)
   400e91:      48 89 6c 24 e8              mov      %rbp,-0x18(%rsp)
   400e96:      4c 89 64 24 f0              mov      %r12,-0x10(%rsp)
   400e9b:      4c 89 6c 24 f8              mov      %r13,-0x8(%rsp)
   400ea0:      48 83 ec 48                 sub      $0x48,%rsp
   400ea4:      48 89 e6                    mov      %rsp,%rsi
   400ea7:      e8 97 08 00 00              call     401743 <read_six_numbers>
   400eac:      48 89 e5                    mov      %rsp,%rbp
   400eaf:4c 8d 6c 24 0c            lea      0xc(%rsp),%r13
   400eb4:      41 bc 00 00 00 00           mov      $0x0,%r12d
   400eba:      48 89 eb                    mov      %rbp,%rbx
   400ebd:      8b 45 0c                    mov      0xc(%rbp),%eax
   400ec0:      39 45 00                       cmp      %eax,0x0(%rbp)
   400ec3:      74 05                          je       400eca <phase_2+0x3e>
   400ec5:      e8 73 07 00 00              call     40163d <explode_bomb>
   400eca:      44 03 23                       add      (%rbx),%r12d
   400ecd:      48 83 c5 04                 add      $0x4,%rbp
   400ed1:      4c 39 ed                    cmp      %r13,%rbp
   400ed4:      75 e4                          jne      400eba <phase_2+0x2e>
   400ed6:      45 85 e4                    test     %r12d,%r12d
   400ed9:      75 05                          jne      400ee0 <phase_2+0x54>
   400edb:      e8 5d 07 00 00              call     40163d <explode_bomb>
   400ee0:      48 8b 5c 24 28              mov      0x28(%rsp),%rbx
   400ee5:      48 8b 6c 24 30              mov      0x30(%rsp),%rbp
   400eea:      4c 8b 64 24 38              mov      0x38(%rsp),%r12
   400eef:4c 8b 6c 24 40            mov      0x40(%rsp),%r13
   400ef4:      48 83 c4 48                 add      $0x48,%rsp
   400ef8:      c3                             ret
```

我们先逐条语句进行解释分析：

- `mov %rbx,-0x20(%rsp)`：将 `%rbx` 寄存器的值移动到栈上偏移为 `-0x20` 的位置，即将 `%rbx` 的值保存在栈中。
- `mov %rbp,-0x18(%rsp)`：将 `%rbp` 寄存器的值移动到栈上偏移为 `-0x18` 的位置，即将 `%rbp` 的值保存在栈中。
- `mov %r12,-0x10(%rsp)`：将 `%r12` 寄存器的值移动到栈上偏移为 `-0x10` 的位置，即将 `%r12` 的值保存在栈中。
- `mov %r13,-0x8(%rsp)`：将 `%r13` 寄存器的值移动到栈上偏移为 `-0x8` 的位置，即将 `%r13` 的值保存在栈中。
- `sub $0x48,%rsp`：从栈顶减去 72 字节（0x48），为局部变量或临时数据分配空间。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

• mov %rsp,%rsi：将栈指针 %rsp 的值移动到 %rsi 寄存器中，用作参数传递给后面要调用的函数。

• call 401743 <read_six_numbers>：调用函数 read_six_numbers，并将当前栈指针 %rsp 的值作为参数传递给它。这个函数用于读取输入的六个数字。

• mov %rsp,%rbp：将当前栈指针 %rsp 的值移动到 %rbp 寄存器中，保存当前的栈帧基址。

• lea 0xc(%rsp),%r13：计算并将栈上偏移为 0xc 的地址（即栈上第一个参数的位置）移动到 %r13 寄存器中。

• mov $0x0,%r12d：将立即数 0 移动到 %r12d 寄存器中，用作循环计数器。

• mov %rbp,%rbx：将栈帧基址 %rbp 的值移动到 %rbx 寄存器中，作为循环过程中的指针。

• mov 0xc(%rbp),%eax：从 %rbp 偏移 0xc 处读取一个双字（DWORD）数据，将其值移动到 %eax 寄存器中。

• cmp %eax,0x0(%rbp)：将 %eax 寄存器的值与 %rbp 偏移 0 处的双字数据进行比较。

• je 400eca <phase_2+0x3e>：如果比较结果相等（zero equal），则跳转到地址 0x400eca 处执行，否则继续执行下一条指令。

• call 40163d <explode_bomb>：调用函数 explode_bomb。

• add (%rbx),%r12d：将 %rbx 指向的双字数据加到 %r12d 寄存器中。

• add $0x4,%rbp： %rbp 增加 4，指向下一个双字数据。

• cmp %r13,%rbp：将 %r13 寄存器的值与 %rbp 寄存器的值进行比较。

• jne 400eba <phase_2+0x2e>：如果比较结果不相等（not equal），则跳转到地址 0x400eba 处执行，否则继续执行下一条指令。

• test %r12d,%r12d：对 %r12d 寄存器的值进行与操作，设置相应的标志位。

• jne 400ee0 <phase_2+0x54>：如果 %r12d 不为零，则跳转到地址 0x400ee0 处执行，否则继续执行下一条指令。

• mov 0x28(%rsp),%rbx：从栈顶偏移 0x28 处读取一个双字数据，将其值移动到 %rbx 寄存器中。

• mov 0x30(%rsp),%rbp：从栈顶偏移 0x30 处读取一个双字数据，将其值移动到 %rbp 寄存器中。

• mov 0x38(%rsp),%r12：从栈顶偏移 0x38 处读取一个双字数据，将其值移动到 %r12 寄存器中。

• mov 0x40(%rsp),%r13：从栈顶偏移 0x40 处读取一个双字数据，将其值移动到 %r13 寄存器中。

• add $0x48,%rsp：栈指针增加 72 字节（0x48），恢复栈的原始状态。

• ret：函数返回。

从上面对程序的分析以及重点关注加粗的指令，可以推断出，这个程序是读入六个 4 字节的值，之后开始循环比较当前的值是否与它后面间隔为 3 的值相等，并且会进行累加操作。具体来说就是：

输入六个数 a,b,c,d,e,f 需要满足以下条件：

（1）a==d,b==e,c==f。

（2）a+b!=0,a+b+c!=0,a+b+c+d!=0,a+b+c+d+e!=0,a+b+c+d+e+f!=0。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

那么答案就有很多了，比如输入1 2 3 1 2 3即可通过此关，如下图所示。

```
1 2 3 1 2 3
That's number 2.  Keep going!
```

图4 关2通过。

第三关：

下面是第二关对应的汇编代码：

```
0000000000400ef9 <phase_3>:
  400ef9:   48 83 ec 18             sub    $0x18,%rsp
  400efd:   48 8d 4c 24 08          lea    0x8(%rsp),%rcx
  400f02:   48 8d 54 24 0c          lea    0xc(%rsp),%rdx
  400f07:   be be 1e 40 00          mov    $0x401ebe,%esi
  400f0c:   b8 00 00 00 00          mov    $0x0,%eax
  400f11:   e8 9a fb ff ff          call   400ab0 <__isoc99_sscanf@plt>
  400f16:   83 f8 01                cmp    $0x1,%eax
  400f19:   7f 05                   jg     400f20 <phase_3+0x27>
  400f1b:   e8 1d 07 00 00          call   40163d <explode_bomb>
  400f20:   83 7c 24 0c 07          cmpl   $0x7,0xc(%rsp)
  400f25:   77 3c                   ja     400f63 <phase_3+0x6a>
  400f27:   8b 44 24 0c             mov    0xc(%rsp),%eax
  400f2b:   ff 24 c5 60 1b 40 00    jmp    *0x401b60(,%rax,8)
  400f32:   b8 17 02 00 00          mov    $0x217,%eax
  400f37:   eb 3b                   jmp    400f74 <phase_3+0x7b>
  400f39:   b8 d6 00 00 00          mov    $0xd6,%eax
  400f3e:   eb 34                   jmp    400f74 <phase_3+0x7b>
  400f40:   b8 53 01 00 00          mov    $0x153,%eax
  400f45:   eb 2d                   jmp    400f74 <phase_3+0x7b>
  400f47:   b8 77 00 00 00          mov    $0x77,%eax
  400f4c:   eb 26                   jmp    400f74 <phase_3+0x7b>
  400f4e:   b8 60 01 00 00          mov    $0x160,%eax
  400f53:   eb 1f                   jmp    400f74 <phase_3+0x7b>
  400f55:   b8 97 03 00 00          mov    $0x397,%eax
  400f5a:   eb 18                   jmp    400f74 <phase_3+0x7b>
  400f5c:   b8 9c 01 00 00          mov    $0x19c,%eax
  400f61:   eb 11                   jmp    400f74 <phase_3+0x7b>
  400f63:   e8 d5 06 00 00          call   40163d <explode_bomb>
  400f68:   b8 00 00 00 00          mov    $0x0,%eax
  400f6d:   eb 05                   jmp    400f74 <phase_3+0x7b>
  400f6f:   b8 9e 03 00 00          mov    $0x39e,%eax
  400f74:   3b 44 24 08             cmp    0x8(%rsp),%eax
  400f78:   74 05                   je     400f7f <phase_3+0x86>
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。

```
400f7a:   e8 be 06 00 00          call   40163d <explode_bomb>
400f7f:   48 83 c4 18             add    $0x18,%rsp
400f83:   c3                      ret
```

我们先逐条语句进行解释分析：
sub $0x18,%rsp: 将栈指针 %rsp 减去 0x18，即 24 字节，为局部变量或参数预留空间。
lea 0x8(%rsp),%rcx: 将 %rsp + 8 处的地址加载到 %rcx 寄存器中，可能是为了访问局部变量或参数。
lea 0xc(%rsp),%rdx: 将 %rsp + 12 处的地址加载到 %rdx 寄存器中，同样可能是为了访问局部变量或参数。
mov $0x401ebe,%esi: 将常数 0x401ebe 移动到 %esi 寄存器中，这可能是一个地址或值。
mov $0x0,%eax: 将常数 0x0 移动到 %eax 寄存器中。
call 400ab0 <__isoc99_sscanf@plt>: 调用函数 __isoc99_sscanf，这是用于处理格式化输入的函数。
cmp $0x1,%eax: 将 %eax 和常数 1 进行比较。
jg 400f20 <phase_3+0x27>: 如果 %eax 大于 1，则跳转到地址 400f20，否则继续执行下一条指令。
call 40163d <explode_bomb>: 如果 jg 的条件不满足,即 %eax <= 1,则调用 explode_bomb 函数。
cmpl $0x7,0xc(%rsp): 将 0xc(%rsp) 处的值和常数 0x7 进行比较。
ja 400f63 <phase_3+0x6a>: 如果 0xc(%rsp) 大于 0x7，则跳转到地址 400f63，继续执行后续指令。
mov 0xc(%rsp),%eax: 将 0xc(%rsp) 处的值移动到 %eax 寄存器中。
jmp *0x401b60(,%rax,8): 根据 %rax 的值跳转到不同的地址，这里是一个跳转表，根据 %eax 的值进行跳转。
后续的 mov 和 jmp 指令是基于 %eax 不同的值来选择不同的跳转路径。
最后的 add 指令将栈指针恢复，然后通过 ret 返回。

通过对重点指令的分析可以得出，程序就是先输入两个数，根据第一个数进行 switch 跳转，范围是 0-7，超过 7 炸弹就会爆炸，跳转之后会将第二个数和对应的数进行比较，如果不相等也会爆炸。
那么我们就可以使用 gdb 查询跳转表了：

```
(gdb) p/x *(int*)0x401b60
$7 = 0x400f32
(gdb) p/x *(int*)(0x401b68)
$8 = 0x400f6f
(gdb) p/x *(int*)(0x401b60+2*8)
$9 = 0x400f39
(gdb) p/x *(int*)(0x401b60+3*8)
$10 = 0x400f40
(gdb) p/x *(int*)(0x401b60+4*8)
$11 = 0x400f47
(gdb) p/x *(int*)(0x401b60+5*8)
$12 = 0x400f4e
(gdb) p/x *(int*)(0x401b60+6*8)
$13 = 0x400f55
(gdb) p/x *(int*)(0x401b60+7*8)
$14 = 0x400f5c
```

图 5 跳转表

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

如图，可以看到输入的第一个数对应的跳转位置，从而就可以找出所需要输入的第二个数。0-535，1-926，2-214，3-339，4-119，5-352，6-919，7-412。

之后进行测试：

```
7 412
Halfway there!
```

图 6 通过关 3

如图 6，可以看到我们输入 7 和 412 后通过了第三关。

第四关：

下面是第四关对应的汇编代码：

```
0000000000400f84 <func4>:
  400f84:    48 89 5c 24 f0          mov     %rbx,-0x10(%rsp)
  400f89:    48 89 6c 24 f8          mov     %rbp,-0x8(%rsp)
  400f8e:    48 83 ec 18             sub     $0x18,%rsp
  400f92:    89 fb                   mov     %edi,%ebx
  400f94:    b8 01 00 00 00          mov     $0x1,%eax
  400f99:    83 ff 01                cmp     $0x1,%edi
  400f9c:    7e 14                   jle     400fb2 <func4+0x2e>
  400f9e:    8d 7b ff                lea     -0x1(%rbx),%edi
  400fa1:    e8 de ff ff ff          call    400f84 <func4>
  400fa6:    89 c5                   mov     %eax,%ebp
  400fa8:    8d 7b fe                lea     -0x2(%rbx),%edi
  400fab:    e8 d4 ff ff ff          call    400f84 <func4>
  400fb0:    01 e8                   add     %ebp,%eax
  400fb2:    48 8b 5c 24 08          mov     0x8(%rsp),%rbx
  400fb7:    48 8b 6c 24 10          mov     0x10(%rsp),%rbp
  400fbc:    48 83 c4 18             add     $0x18,%rsp
  400fc0:    c3                      ret

0000000000400fc1 <phase_4>:
  400fc1:    48 83 ec 18             sub     $0x18,%rsp
  400fc5:    48 8d 54 24 0c          lea     0xc(%rsp),%rdx
  400fca:be c1 1e 40 00              mov     $0x401ec1,%esi
  400fcf: b8 00 00 00 00             mov     $0x0,%eax
  400fd4:    e8 d7 fa ff ff          call    400ab0 <__isoc99_sscanf@plt>
  400fd9:    83 f8 01                cmp     $0x1,%eax
  400fdc:    75 07                   jne     400fe5 <phase_4+0x24>
  400fde:    83 7c 24 0c 00          cmpl    $0x0,0xc(%rsp)
  400fe3:    7f 05                   jg      400fea <phase_4+0x29>
  400fe5:    e8 53 06 00 00          call    40163d <explode_bomb>
  400fea:8b 7c 24 0c                 mov     0xc(%rsp),%edi
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
400fee:e8 91 ff ff ff        call   400f84 <func4>
400ff3:83 f8 37              cmp    $0x37,%eax
400ff6:74 05                 je     400ffd <phase_4+0x3c>
400ff8:e8 40 06 00 00        call   40163d <explode_bomb>
400ffd:48 83 c4 18           add    $0x18,%rsp
401001:      c3              ret
```

逐条分析语句：

Fun 4：

mov %rbx,-0x10(%rsp): 将 %rbx 寄存器中的值存储到栈上偏移为 -0x10 的位置。

mov %rbp,-0x8(%rsp): 将 %rbp 寄存器中的值存储到栈上偏移为 -0x8 的位置。

sub $0x18,%rsp: 在栈上分配 24 字节的空间。

mov %edi,%ebx: 将 %edi 寄存器中的值移动到 %ebx 寄存器中。

mov $0x1,%eax: 将常数 1 移动到 %eax 寄存器中。

cmp $0x1,%edi: 将 %edi 寄存器中的值与常数 1 进行比较。

jle 400fb2 <func4+0x2e>: 如果 %edi 的值小于等于 1，则跳转到 400fb2 处执行；否则继续执行下一条指令。

lea -0x1(%rbx),%edi: 将 %rbx 寄存器中的值减去 1 后的地址加载到 %edi 寄存器中。

call 400f84 <func4>: 递归调用 func4 函数。

mov %eax,%ebp: 将函数返回值（存储在 %eax 中）移动到 %ebp 寄存器中。

lea -0x2(%rbx),%edi: 将 %rbx 寄存器中的值减去 2 后的地址加载到 %edi 寄存器中。

call 400f84 <func4>: 递归调用 func4 函数。

add %ebp,%eax: 将 %ebp 寄存器中的值加到 %eax 寄存器中。

mov 0x8(%rsp),%rbx: 恢复 %rbx 寄存器的值。

mov 0x10(%rsp),%rbp: 恢复 %rbp 寄存器的值。

add $0x18,%rsp: 释放栈空间。

ret: 返回函数调用。

phase_4：

sub $0x18,%rsp: 在栈上分配 24 字节的空间。

lea 0xc(%rsp),%rdx: 将栈上偏移为 0xc 的地址加载到 %rdx 寄存器中。

mov $0x401ec1,%esi: 将常数 0x401ec1 移动到 %esi 寄存器中。

mov $0x0,%eax: 将常数 0 移动到 %eax 寄存器中。

call 400ab0 <__isoc99_sscanf@plt>: 调用 __isoc99_sscanf 函数。

cmp $0x1,%eax: 将 %eax 寄存器中的值与常数 1 进行比较。

jne 400fe5 <phase_4+0x24>: 如果 %eax 的值不等于 1，则跳转到 400fe5 处执行；否则继续执行下一条指令。

cmpl $0x0,0xc(%rsp): 将栈上偏移为 0xc 的值与常数 0 进行比较。

jg 400fea <phase_4+0x29>: 如果栈上偏移为 0xc 的值大于 0，则跳转到 400fea 处执行；否则继续执行下一条指令。

mov 0xc(%rsp),%edi: 将栈上偏移为 0xc 的值移动到 %edi 寄存器中。

call 400f84 <func4>: 调用 func4 函数。

cmp $0x37,%eax: 将 %eax 寄存器中的值与常数 0x37 进行比较。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

je 400ffd <phase_4+0x3c>: 如果 %eax 的值等于 0x37，则跳转到 400ffd 处执行；否则继续执行下一条指令。

call 40163d <explode_bomb>: 调用 explode_bomb 函数。

add $0x18,%rsp: 释放栈空间。

ret: 返回函数调用。

对上诉指令的分析以及对加粗指令的重点分析，可以得出：

程序读入一个值，如果这个值小于等于 0 则直接让炸弹爆炸，大于 0 则条用 fun 4。

在 fun 4 中，如果传入的参数小于等于 1 就会开始返回，如果不是则会让参数减一然后调用自身，在调用的函数返回后，它还会用当前值加上返回的值。也就是说当我们传入 3，那么它会调用 fun 4（2），然后再返回 3+fun（2），返回条件是小于等于 1，这就是对斐波那契数列的计算。

后续又会与 55 进行比较，也就是说我们需要找出斐波那契数列的第几项等于 55，也就是第九项，所以只要输入 9 就可以通过。

测试：

```
9
So you got that one.  Try this one.
```

图 7 通过关卡 4

如图 7，我们通过了关卡 4。

第五关：

下面是第五关对应的汇编代码：

```
0000000000401002 <phase_5>:
  401002:    48 83 ec 18          sub    $0x18,%rsp
  401006:    48 8d 4c 24 08       lea    0x8(%rsp),%rcx
  40100b:    48 8d 54 24 0c       lea    0xc(%rsp),%rdx
  401010:    be be 1e 40 00       mov    $0x401ebe,%esi
  401015:    b8 00 00 00 00       mov    $0x0,%eax
  40101a:    e8 91 fa ff ff       call   400ab0 <__isoc99_sscanf@plt>
  40101f:    83 f8 01             cmp    $0x1,%eax
  401022:    7f 05                jg     401029 <phase_5+0x27>
  401024:    e8 14 06 00 00       call   40163d <explode_bomb>
  401029:    8b 44 24 0c          mov    0xc(%rsp),%eax
  40102d:    83 e0 0f             and    $0xf,%eax
  401030:    89 44 24 0c          mov    %eax,0xc(%rsp)
  401034:    83 f8 0f             cmp    $0xf,%eax
  401037:    74 2c                je     401065 <phase_5+0x63>
  401039:    b9 00 00 00 00       mov    $0x0,%ecx
  40103e:    ba 00 00 00 00       mov    $0x0,%edx
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
401043:   83 c2 01          add    $0x1,%edx
401046:   48 98             cltq
401048:   8b 04 85 a0 1b 40 00   mov    0x401ba0(,%rax,4),%eax
40104f:   01 c1             add    %eax,%ecx
401051:   83 f8 0f          cmp    $0xf,%eax
401054:   75 ed             jne    401043 <phase_5+0x41>
401056:   89 44 24 0c       mov    %eax,0xc(%rsp)
40105a:   83 fa 0c          cmp    $0xc,%edx
40105d:   75 06             jne    401065 <phase_5+0x63>
40105f:   3b 4c 24 08       cmp    0x8(%rsp),%ecx
401063:   74 05             je     40106a <phase_5+0x68>
401065:   e8 d3 05 00 00    call   40163d <explode_bomb>
40106a:   48 83 c4 18       add    $0x18,%rsp
40106e:   c3                ret
```

对汇编指令逐条分析：

sub $0x18,%rsp：从栈中减去 24 个字节的空间。

lea 0x8(%rsp),%rcx：将%rsp + 0x8 的地址加载到%rcx 寄存器。

lea 0xc(%rsp),%rdx：将%rsp + 0xc 的地址加载到%rdx 寄存器。

mov $0x401ebe,%esi：将地址$0x401ebe 移动到%esi 寄存器。

mov $0x0,%eax：将值 0 移动到%eax 寄存器。

call 400ab0 <__isoc99_sscanf@plt>：调用函数__isoc99_sscanf。

cmp $0x1,%eax：将%eax 与 1 进行比较。

jg 401029 <phase_5+0x27>：如果%eax 大于 1，跳转到 401029 处。

mov 0xc(%rsp),%eax：将%rsp + 0xc 处的值移动到%eax 寄存器。

and $0xf,%eax：对%eax 寄存器的值与 0xf 进行按位与操作。

mov %eax,0xc(%rsp)：将结果移动回%rsp + 0xc 处。

cmp $0xf,%eax：将%eax 与 0xf 进行比较。

je 401065 <phase_5+0x63>：如果相等，跳转到 401065 处。

mov $0x0,%ecx：将立即数 0x0（十进制为 0）移动到 %ecx 寄存器中。

mov $0x0,%edx：将立即数 0x0 移动到 %edx 寄存器中。

add $0x1,%edx：将 1 加到 %edx 寄存器的值上。

cltq：将 %eax 的 32 位值符号扩展到 %rax 的 64 位中。这意味着它将 %eax 中的 32 位（有符号整数）拓展到 %rax 的高 32 位，并保持符号性质不变。

mov 0x401ba0(,%rax,4),%eax：将从内存地址 0x401ba0 + (%rax * 4) 计算出来的 32 位值移动到 %eax 寄存器中。这里涉及到根据 %rax 的值计算有效地址。

add %eax,%ecx：将 %eax 中的值加到 %ecx 中的值上。

cmp $0xf,%eax：比较 %eax 中的值和 0xf（十进制为 15）。

jne 401043：如果比较结果不相等，则跳转到地址 401043，即 phase_5+0x41 处。

mov %eax,0xc(%rsp)：将 %eax 寄存器的值移动到栈顶偏移量为 0xc 的位置。

cmp $0xc,%edx：将 %edx 中的值和 0xc（十进制为 12）进行比较。

jne 401065：如果比较结果不相等，则跳转到地址 401065，即 phase_5+0x63 处。

cmp 0x8(%rsp),%ecx：将栈顶偏移量为 0x8 的位置的值与 %ecx 寄存器中的值进行比较。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

je 40106a：如果比较结果相等，则跳转到地址 40106a，即 phase_5+0x68 处。
call 40163d：调用函数，跳转到地址 40163d，即 explode_bomb 函数。
add $0x18,%rsp：将栈指针 %rsp 上移 0x18（十进制为 24），恢复栈空间。
ret：返回。

对指令进行分析并且重点关注加粗的指令，可以推断出，这个程序输入两个数，如果第一个数的后四位是 1111，那么就会让炸弹直接爆炸。
后面的程序就是循环遍历一个数组，使用 **mov 0x401ba0(,%rax,4),%eax** 取出下一个值，然后进行累加操作，直到取出的数是 15 则结束。
从后续的判断条件可以知道，这个程序需要循环 12 次，不然炸弹就会爆炸。
那么我们输入的两个值，一个就是数组的起始下标，一个就是累加和的值。
先查看数组内容：

```
(gdb) p*0x401ba0@16
$16 = {10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6, 5}
```

由于会循环 12 次，并且之后一个值一定是 15，那么我们可以逆序找出那 12 个数，
（1）15
（2）6
（3）14
（4）2
（5）1
（6）10
（7）0
（8）8
（9）4
（10）9
（11）13
（12）11

累加和为 93，并且 11 对应的下标为 7，因此输入 7 93。

```
7 93
Congratulations! You've (mostly) defused the bomb!
Hit Control-C to escape phase 6 (for free!), but if you want to
try phase 6 for extra credit, you can continue.  Just beware!
```

图 8 通过第五关

如图 8，我成功通过第五关。

第六关：

下面是第六关对应的汇编代码：
对汇编代码进行解释分析：
000000000040106f <fun6>:

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
  40106f:    4c 8b 47 08              mov     0x8(%rdi),%r8
  401073:    48 c7 47 08 00 00 00     movq    $0x0,0x8(%rdi)
  40107a:    00
  40107b:    48 89 f8                 mov     %rdi,%rax
  40107e:    48 89 f9                 mov     %rdi,%rcx
  401081:    4d 85 c0                 test    %r8,%r8
  401084:    75 40                    jne     4010c6 <fun6+0x57>
  401086:    48 89 f8                 mov     %rdi,%rax
  401089:    c3                       ret
  40108a:    48 89 d1                 mov     %rdx,%rcx
  40108d:    48 8b 51 08              mov     0x8(%rcx),%rdx
  401091:    48 85 d2                 test    %rdx,%rdx
  401094:    74 09                    je      40109f <fun6+0x30>
  401096:    39 32                    cmp     %esi,(%rdx)
  401098:    7f f0                    jg      40108a <fun6+0x1b>
  40109a:    48 89 cf                 mov     %rcx,%rdi
  40109d:    eb 03                    jmp     4010a2 <fun6+0x33>
  40109f:    48 89 cf                 mov     %rcx,%rdi
  4010a2:    48 39 d7                 cmp     %rdx,%rdi
  4010a5:    74 06                    je      4010ad <fun6+0x3e>
  4010a7:    4c 89 47 08              mov     %r8,0x8(%rdi)
  4010ab:    eb 03                    jmp     4010b0 <fun6+0x41>
  4010ad:    4c 89 c0                 mov     %r8,%rax
  4010b0:    49 8b 48 08              mov     0x8(%r8),%rcx
  4010b4:    49 89 50 08              mov     %rdx,0x8(%r8)
  4010b8:    48 85 c9                 test    %rcx,%rcx
  4010bb:    74 1a                    je      4010d7 <fun6+0x68>
  4010bd:    49 89 c8                 mov     %rcx,%r8
  4010c0:    48 89 c1                 mov     %rax,%rcx
  4010c3:    48 89 c7                 mov     %rax,%rdi
  4010c6:    48 89 ca                 mov     %rcx,%rdx
  4010c9:    48 85 c9                 test    %rcx,%rcx
  4010cc:    74 d4                    je      4010a2 <fun6+0x33>
  4010ce:    41 8b 30                 mov     (%r8),%esi
  4010d1:    39 31                    cmp     %esi,(%rcx)
  4010d3:    7f b8                    jg      40108d <fun6+0x1e>
  4010d5:    eb cb                    jmp     4010a2 <fun6+0x33>
  4010d7:    f3 c3                    repz ret

00000000004010d9 <phase_6>:
  4010d9:sub      $0x8,%rsp  从栈上减去 8 个字节的空间。
  4010dd:mov      $0xa,%edx  将立即数 0xa（十进制为 10）移动到 %edx 寄存器中。
  4010e2:mov      $0x0,%esi  将立即数 0x0 移动到 %esi 寄存器中。
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

4010e7:call 400b80 <strtol@plt> 调用 strtol 函数。

4010ec:mov %eax,0x20168e(%rip) # 602780 <node0>将 strtol 的返回值移动到内存地址 0x20168e(%rip) 处。

4010f2:mov $0x602780,%edi 将地址 0x602780 移动到 %edi 寄存器中。

4010f7:call 40106f <fun6> 调用 fun6 函数。

4010fc:mov 0x8(%rax),%rax 将 %rax 寄存器加上偏移量 0x8 处的值移动到 %rax 中。

401100:mov 0x8(%rax),%rax 再次将 %rax 寄存器加上偏移量 0x8 处的值移动到 %rax 中，相当于 %rax 指向的地址中的内容为二级指针。

401104:mov 0x8(%rax),%rax 再次将 %rax 寄存器加上偏移量 0x8 处的值移动到 %rax 中，相当于 %rax 指向的地址中的内容为三级指针。

401108:mov 0x201672(%rip),%edx # 602780 <node0> 将内存地址 0x201672(%rip) 处的值移动到 %edx 寄存器中。

40110e:cmp %edx,(%rax) 将 %edx 寄存器中的值与 %rax 寄存器指向的地址的值进行比较。

401110:je 401117 <phase_6+0x3e> 如果比较结果相等，则跳转到地址 401117，即 phase_6+0x3e 处。

401112:call 40163d <explode_bomb> 调用 explode_bomb 函数。

401117:add $0x8,%rsp 恢复栈空间，增加 8 个字节。

40111b:ret 返回。

由于 fun 6 的逻辑实在是太复杂了，尝试写 fun 6 对应的 c 语言代码，分支实在是太多了。但这题可以使用别的方法解决，分析 phase_6 可以知道，当寄存器 edx 中的值和寄存器 rax 指向的值相等时，炸弹就不会爆炸。

```
Breakpoint 1, 0x000000000040110e in phase_6 ()
(gdb) p $edx
$17 = 1231
(gdb) p *$rax
$18 = 673
```

图 9 断点查看

在比较前设置一个断点，然后运行查看寄存器 edx 中的值以及寄存器 rax 指向的值，我们可以知道寄存器 rax 指向的值是 673，而寄存器 edx 的值是我们输入的数的值。也就是说我们输入 673 就有机会通过这关。

```
673
Congratulations! You've defused the bomb! Again!
```

图 10 验证答案

如图，当输入 673 既可以通过次关卡。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

五、实验总结：

本次实验，我成功通过了六个关卡，难度从简单到难，让我更熟悉了循环结构，分支结构，switch 结构，递归结构的汇编指令实现。也让我对如 mov，lea，sub，and，add，test，jump 这些常用的汇编指令更加的熟悉。

同时，我也学会了使用命令行对可执行文件进行反汇编：objdump -d 源文件 -> 目标文件。学会了使用 gdb 对程序进行一些查看和调试，如使用 p *地址 查看对应地址中的数据，使用 b *地址 在想要的位置设置断点，方便对程序中的变量进行查看，如使用 p $edx 查看寄存器的值。

总体上，本次的实验是很有收获的，不仅熟悉了第三章的汇编指令，学会了对着汇编指令写一些比较容易看懂的伪代码。也学会了运用 GDB 调试工具和 objdump 反汇编工具。实验完成！

指导教师批阅意见：

成绩评定：

指导教师签字：刘刚

2024 年 5 月　　日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。