首先将bomb代码反汇编

objdump –d bomb > bomb.txt

phase_1

gdb bomb

disas phase_1 获得它的汇编代码

b phase_1 设置断点

run 运行

随便输入一个字符串

stepi 接着 disas phase_1 发现程序执行了一步

再stepi接着disas phase_1 发现程序执行到了 callq

此时 info register 查看寄存器

x/s $esi

./bomb sol将答案放在文本文件里

phase_2

disas phase_2

```
Dump of assembler code for function phase_2:
   0x0000000000001210 <+0>:     push   %rbp
   0x0000000000001211 <+1>:     push   %rbx
   0x0000000000001212 <+2>:     sub    $0x28,%rsp
   0x0000000000001216 <+6>:     mov    %rsp,%rsi
   0x0000000000001219 <+9>:     callq  0x18ec <read_six_numbers>
   0x000000000000121e <+14>:    cmpl   $0x0,(%rsp)
   0x0000000000001222 <+18>:    jne    0x122b <phase_2+27>
   0x0000000000001224 <+20>:    cmpl   $0x1,0x4(%rsp)
   0x0000000000001229 <+25>:    je     0x1230 <phase_2+32>
   0x000000000000122b <+27>:    callq  0x18b0 <explode_bomb>
   0x0000000000001230 <+32>:    mov    %rsp,%rbx
   0x0000000000001233 <+35>:    lea    0x10(%rsp),%rbp
   0x0000000000001238 <+40>:    jmp    0x1243 <phase_2+51>
   0x000000000000123a <+42>:    add    $0x4,%rbx
   0x000000000000123e <+46>:    cmp    %rbp,%rbx
   0x0000000000001241 <+49>:    je     0x1254 <phase_2+68>
   0x0000000000001243 <+51>:    mov    0x4(%rbx),%eax
   0x0000000000001246 <+54>:    add    (%rbx),%eax
   0x0000000000001248 <+56>:    cmp    %eax,0x8(%rbx)
   0x000000000000124b <+59>:    je     0x123a <phase_2+42>
   0x000000000000124d <+61>:    callq  0x18b0 <explode_bomb>
   0x0000000000001252 <+66>:    jmp    0x123a <phase_2+42>
---Type <return> to continue, or q <return> to quit---return
```

%rsp 是输入的数字的栈

%rsi 得到了%rsp栈

前两位必须是0，1，成功跳到32

然后把输入的数字栈移动到了%rbx

给%rbq取地址0x10（%rsp），直接跳到51步

给%eax赋值 %rbx的第二位 即%rsp的第二位 即 1

然后 %eax = 1 + 0 与 %rbx第三位比较 如果相等就到42步 所以第三位是1

42步给%rbx后移了4位，然后和%rbp比较，相等就跳出，不相等则将%rbx的第三位 即1赋值给%eax

然后 %eax = 1 + 1 与 %rbx第三位比较 所以第三位是 2

%rbx又后移四位，%eax先被赋值为 2

斐波那契数列！0 1 1 2 3 5

lea即取地址，&s→d

phase_3

disas phase_3

```
gdb) disas phase_3
ump of assembler code for function phase_3:
   0x000000000000125b <+0>:     sub    $0x18,%rsp
   0x000000000000125f <+4>:     lea    0x8(%rsp),%rcx
   0x0000000000001264 <+9>:     lea    0xc(%rsp),%rdx
   0x0000000000001269 <+14>:    lea    0x1865(%rip),%rsi        # 0x2ad5
   0x0000000000001270 <+21>:    mov    $0x0,%eax
   0x0000000000001275 <+26>:    callq  0xec0 <__isoc99_sscanf@plt>
   0x000000000000127a <+31>:    cmp    $0x1,%eax
   0x000000000000127d <+34>:    jle    0x129e <phase_3+67>
   0x000000000000127f <+36>:    cmpl   $0x7,0xc(%rsp)
   0x0000000000001284 <+41>:    ja     0x1315 <phase_3+186>
   0x000000000000128a <+47>:    mov    0xc(%rsp),%eax
   0x000000000000128e <+51>:    lea    0x15fb(%rip),%rdx        # 0x2890
   0x0000000000001295 <+58>:    movslq (%rdx,%rax,4),%rax
   0x0000000000001299 <+62>:    add    %rdx,%rax
   0x000000000000129c <+65>:    jmpq   *%rax
   0x000000000000129e <+67>:    callq  0x18b0 <explode_bomb>
   0x00000000000012a3 <+72>:    jmp    0x127f <phase_3+36>
   0x00000000000012a5 <+74>:    mov    $0x263,%eax
   0x00000000000012aa <+79>:    jmp    0x12b1 <phase_3+86>
   0x00000000000012ac <+81>:    mov    $0x0,%eax
   0x00000000000012b1 <+86>:    sub    $0x241,%eax
   0x00000000000012b6 <+91>:    add    $0x3bf,%eax
--Type <return> to continue, or q <return> to quit---return
   0x00000000000012bb <+96>:    sub    $0xb0,%eax
   0x00000000000012c0 <+101>:   add    $0xb0,%eax
   0x00000000000012c5 <+106>:   sub    $0xb0,%eax
   0x00000000000012ca <+111>:   add    $0xb0,%eax
```

```
  0x00000000000012cf <+116>:    sub    $0xb0,%eax
  0x00000000000012d4 <+121>:    cmpl   $0x5,0xc(%rsp)
  0x00000000000012d9 <+126>:    jg     0x12e1 <phase_3+134>
  0x00000000000012db <+128>:    cmp    0x8(%rsp),%eax
  0x00000000000012df <+132>:    je     0x12e6 <phase_3+139>
  0x00000000000012e1 <+134>:    callq  0x18b0 <explode_bomb>
  0x00000000000012e6 <+139>:    add    $0x18,%rsp
  0x00000000000012ea <+143>:    retq
  0x00000000000012eb <+144>:    mov    $0x0,%eax
  0x00000000000012f0 <+149>:    jmp    0x12b6 <phase_3+91>
  0x00000000000012f2 <+151>:    mov    $0x0,%eax
  0x00000000000012f7 <+156>:    jmp    0x12bb <phase_3+96>
  0x00000000000012f9 <+158>:    mov    $0x0,%eax
  0x00000000000012fe <+163>:    jmp    0x12c0 <phase_3+101>
  0x0000000000001300 <+165>:    mov    $0x0,%eax
  0x0000000000001305 <+170>:    jmp    0x12c5 <phase_3+106>
  0x0000000000001307 <+172>:    mov    $0x0,%eax
  0x000000000000130c <+177>:    jmp    0x12ca <phase_3+111>
  0x000000000000130e <+179>:    mov    $0x0,%eax
--Type <return> to continue, or q <return> to quit---return
  0x0000000000001313 <+184>:    jmp    0x12cf <phase_3+116>
  0x0000000000001315 <+186>:    callq  0x18b0 <explode_bomb>
  0x000000000000131a <+191>:    mov    $0x0,%eax
  0x000000000000131f <+196>:    jmp    0x12d4 <phase_3+121>
```

首先 b_phase3设置断点

然后run，stepi到 +21 info register

看到%rsi，

```
(gdb) x/s 0x555555556ad5
0x555555556ad5: "%d %d"
```

发现需要输入两个整数

第一个数字不能等于一，输入个2，一直ni到这一步

```
 0x00005555555552db <+128>:    cmp    0x8(%rsp),%eax
```

```
(gdb) info register
rax            0x30f    783
```

查看寄存器，获得%eax的地址

```
(gdb) print 0x30f
$4 = 783
```

查看内容，获得第二个数字

phase_4

```
   0x0000555555555397 <+55>:    callq  0x555555555321 <func4>
=> 0x000055555555539c <+60>:    cmp    $0x4,%eax
```

可以随便输入，ni到这一步 使用 **p** %eax查看返回值，显而易见需要的返回值是4，第一个输入是2

```
 0x00005555555553b4 <+84>:    cmpl   $0x4,0x8(%rsp)
```

又知道 需要的第二个输入是4

所以答案 2 4

本题中：0xc是第一位输入，0x8是第二位输入

phase_5

```
 0x00005555555553c5 <+8>:     callq  0x555555555618 <string_length>
 0x00005555555553ca <+13>:    cmp    $0x6,%eax
```

由这句知道必须是6位的字符串

运行到比较的这一步

```
  0x0000555555555403 <+70>:    lea    0x147c(%rip),%rsi        # 0x555555556886
```

看看内存

```
(gdb) x/s 0x555555556886
0x555555556886: "devils"
(gdb) x/s ($rsp+0x10)
0x7fffffffddf0: ""
(gdb) x/s ($rsp+0x9)
0x7fffffffdde9: "aduier"
```

它和输入的不一样，看一下

```
  0x00005555555553d4 <+23>:    mov    $0x0,%eax
  0x00005555555553d9 <+28>:    lea    0x14d0(%rip),%rcx        # 0x5555555568b0
<array.3096>
  0x00005555555553e0 <+35>:    movzbl (%rbx,%rax,1),%edx
  0x00005555555553e4 <+39>:    and    $0xf,%edx
  0x00005555555553e7 <+42>:    movzbl (%rcx,%rdx,1),%edx
  0x00005555555553eb <+46>:    mov    %dl,0x9(%rsp,%rax,1)
  0x00005555555553ef <+50>:    add    $0x1,%rax
  0x00005555555553f3 <+54>:    cmp    $0x6,%rax
  0x00005555555553f7 <+58>:    jne    0x5555555553e0 <phase_5+35>
```

看看+28

```
(gdb) x/s 0x5555555568b0
0x5555555568b0 <array.3096>:    "maduiersnfotvbylSo you think you can stop the b
omb with ctrl-c, do you?"
```

它是以低4位为索引的 &（0xf)

且是从 0x5555555568b0 取索引 低位需要符合 2 5 12 4 15 7

man ascii

答案25<4?7

phase_6

```
0x0000555555555420 <+0>:     push   %r13
0x0000555555555422 <+2>:     push   %r12
0x0000555555555424 <+4>:     push   %rbp
0x0000555555555425 <+5>:     push   %rbx
0x0000555555555426 <+6>:     sub    $0x58,%rsp
0x000055555555542a <+10>:    lea    0x30(%rsp),%r12
0x000055555555542f <+15>:    mov    %r12,%rsi
0x0000555555555432 <+18>:    callq  0x5555555558ec <read_six_numbers>
0x0000555555555437 <+23>:    mov    $0x0,%r13d
0x000055555555543d <+29>:    jmp    0x555555555464 <phase_6+68>
0x000055555555543f <+31>:    add    $0x1,%r13d
0x0000555555555443 <+35>:    cmp    $0x6,%r13d
0x0000555555555447 <+39>:    je     0x555555555481 <phase_6+97>
0x0000555555555449 <+41>:    mov    %r13d,%ebx
0x000055555555544c <+44>:    movslq %ebx,%rax
0x000055555555544f <+47>:    mov    0x30(%rsp,%rax,4),%eax
0x0000555555555453 <+51>:    cmp    %eax,0x0(%rbp)
0x0000555555555456 <+54>:    je     0x55555555547a <phase_6+90>
0x0000555555555458 <+56>:    add    $0x1,%ebx
0x000055555555545b <+59>:    cmp    $0x5,%ebx
0x000055555555545e <+62>:    jle    0x55555555544c <phase_6+44>
0x0000555555555460 <+64>:    add    $0x4,%r12
0x0000555555555464 <+68>:    mov    %r12,%rbp
0x0000555555555467 <+71>:    mov    (%r12),%eax
0x000055555555546b <+75>:    sub    $0x1,%eax
0x000055555555546e <+78>:    cmp    $0x5,%eax
0x0000555555555471 <+81>:    jbe    0x55555555543f <phase_6+31>
0x0000555555555473 <+83>:    callq  0x5555555558b0 <explode_bomb>
```

```
   0x0000555555555478 <+88>:    jmp    0x55555555543f <phase_6+31>
   0x000055555555547a <+90>:    callq  0x5555555558b0 <explode_bomb>
   0x000055555555547f <+95>:    jmp    0x555555555458 <phase_6+56>
---Type <return> to continue, or q <return> to quit---r
   0x0000555555555481 <+97>:    mov    $0x0,%esi
   0x0000555555555486 <+102>:   mov    0x30(%rsp,%rsi,1),%ecx
   0x000055555555548a <+106>:   mov    $0x1,%eax
   0x000055555555548f <+111>:   lea    0x202e8a(%rip),%rdx        # 0x5555557583
20 <node1>
   0x0000555555555496 <+118>:   cmp    $0x1,%ecx
   0x0000555555555499 <+121>:   jle    0x5555555554a6 <phase_6+134>
   0x000055555555549b <+123>:   mov    0x8(%rdx),%rdx
   0x000055555555549f <+127>:   add    $0x1,%eax
   0x00005555555554a2 <+130>:   cmp    %ecx,%eax
   0x00005555555554a4 <+132>:   jne    0x55555555549b <phase_6+123>
   0x00005555555554a6 <+134>:   mov    %rdx,(%rsp,%rsi,2)
   0x00005555555554aa <+138>:   add    $0x4,%rsi
   0x00005555555554ae <+142>:   cmp    $0x18,%rsi
   0x00005555555554b2 <+146>:   jne    0x555555555486 <phase_6+102>
   0x00005555555554b4 <+148>:   mov    (%rsp),%rbx
   0x00005555555554b8 <+152>:   mov    %rsp,%rax
   0x00005555555554bb <+155>:   lea    0x28(%rsp),%rsi
   0x00005555555554c0 <+160>:   mov    %rbx,%rcx
   0x00005555555554c3 <+163>:   mov    0x8(%rax),%rdx
   0x00005555555554c7 <+167>:   mov    %rdx,0x8(%rcx)
   0x00005555555554cb <+171>:   add    $0x8,%rax
   0x00005555555554cf <+175>:   mov    %rdx,%rcx
   0x00005555555554d2 <+178>:   cmp    %rax,%rsi
   0x00005555555554d5 <+181>:   jne    0x5555555554c3 <phase_6+163>
   0x00005555555554d7 <+183>:   movq   $0x0,0x8(%rdx)
   0x00005555555554df <+191>:   mov    $0x5,%ebp
```

```
0x00005555555554e4 <+196>:    jmp    0x5555555554ef <phase_6+207>
0x00005555555554e6 <+198>:    mov    0x8(%rbx),%rbx
0x00005555555554ea <+202>:    sub    $0x1,%ebp
0x00005555555554ed <+205>:    je     0x555555555500 <phase_6+224>
0x00005555555554ef <+207>:    mov    0x8(%rbx),%rax
```

%rsp+0x20　%rsp+0x28　%rsp+0x30　%rsp+0x38　%rsp+0x40　%rsp+0x48

参数越靠后，地址越高/大，栈顶是最小的