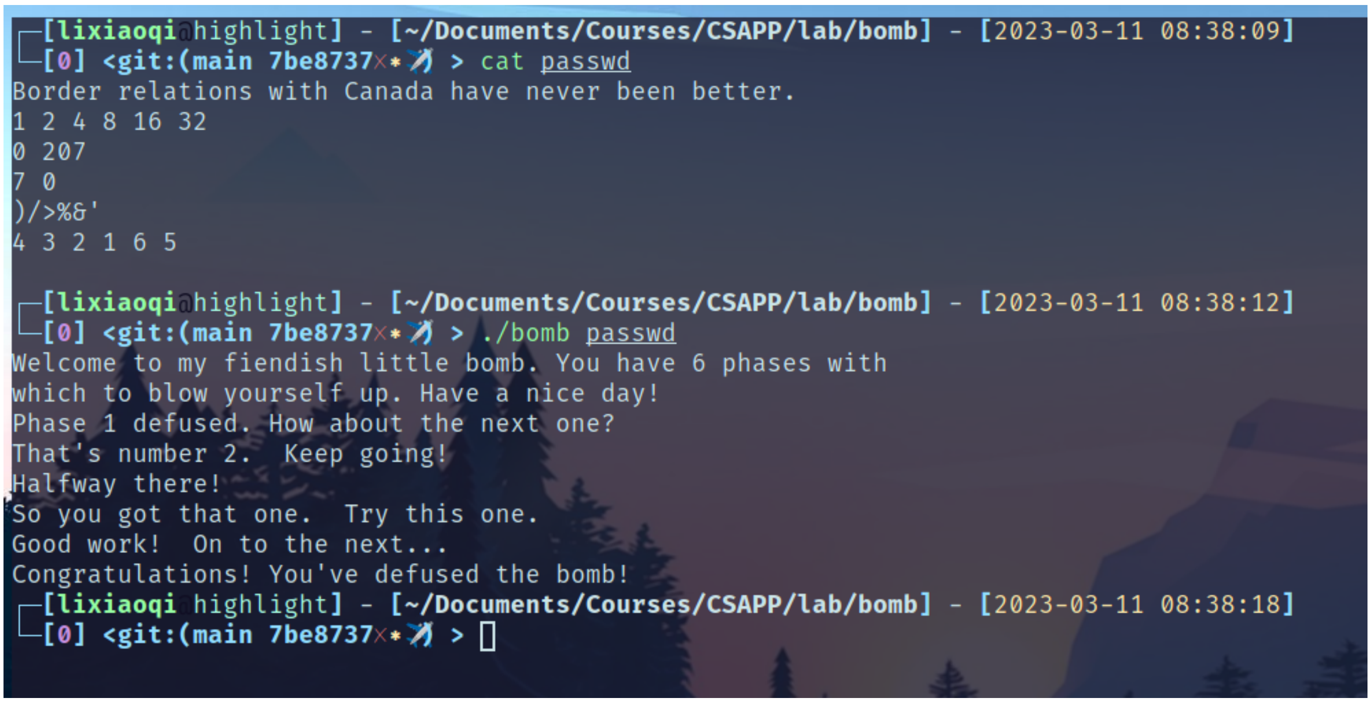
# bomblab报告

## 快览

这个实验需要我们根据给出的可执行文件+主函数源代码，确定6个字符串。

实验完成截图：



```
┌─[lixiaoqi@highlight] - [~/Documents/Courses/CSAPP/lab/bomb] - [2023-03-11 08:38:09]
└─[0] <git:(main 7be8737✗✱✈) > cat passwd
Border relations with Canada have never been better.
1 2 4 8 16 32
0 207
7 0
)/>%&'
4 3 2 1 6 5

┌─[lixiaoqi@highlight] - [~/Documents/Courses/CSAPP/lab/bomb] - [2023-03-11 08:38:12]
└─[0] <git:(main 7be8737✗✱✈) > ./bomb passwd
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...
Congratulations! You've defused the bomb!
┌─[lixiaoqi@highlight] - [~/Documents/Courses/CSAPP/lab/bomb] - [2023-03-11 08:38:18]
└─[0] <git:(main 7be8737✗✱✈) > ▯
```

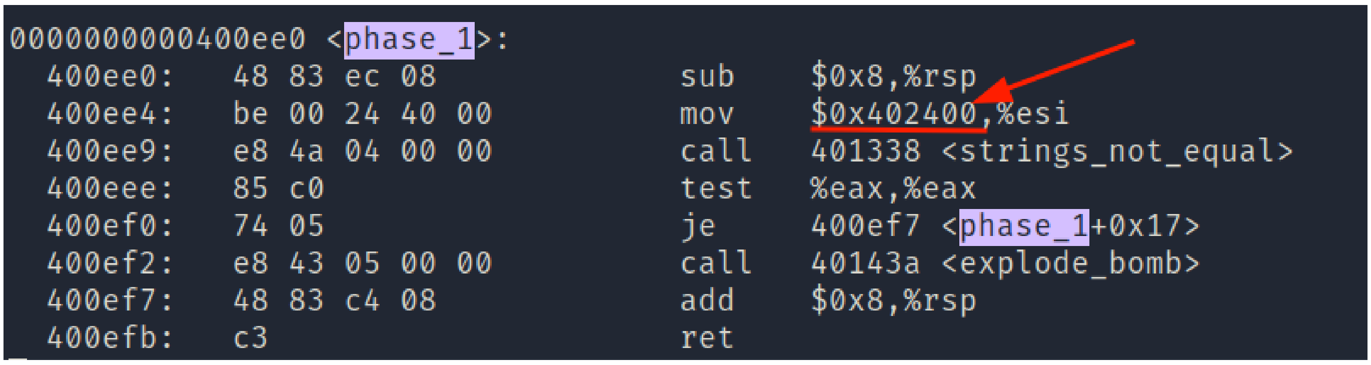
## 实验详解

使用工具：

- objdump，生成bomb的反汇编
- ghex，可以直接以十六进制读取文件，左边十六进制右边ascii，可以直接跳转到任意比特，非常方便。
- gdb逐指令调试

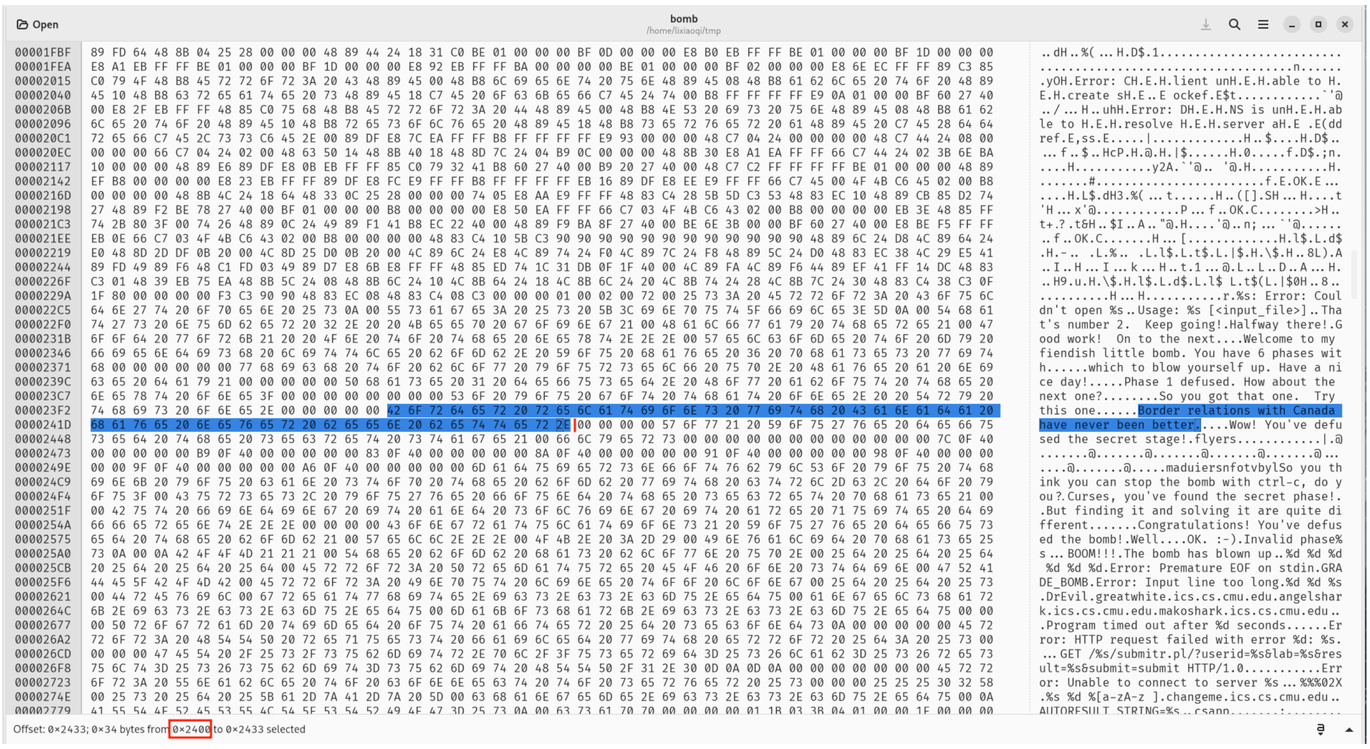
### phase1

红色箭头标识出，进行比较的另一个串首地址在`0x402400`，Linux 系统默认情况下将可执行文件的 .text 节映射到虚拟地址 `0x400000` 开始的位置，所以在文件中的实际偏移是`0x2400`



```
0000000000400ee0 <phase_1>:
  400ee0:    48 83 ec 08             sub    $0x8,%rsp
  400ee4:    be 00 24 40 00          mov    $0x402400,%esi
  400ee9:    e8 4a 04 00 00          call   401338 <strings_not_equal>
  400eee:    85 c0                   test   %eax,%eax
  400ef0:    74 05                   je     400ef7 <phase_1+0x17>
  400ef2:    e8 43 05 00 00          call   40143a <explode_bomb>
  400ef7:    48 83 c4 08             add    $0x8,%rsp
  400efb:    c3                      ret
```

在ghex中跳转至相应地址就可以找到对应字符串



```
00001FBF  89 FD 64 48 8B 04 25 28 00 00 00 48 89 44 24 18 31 C0 BE 01 00 00 00 BF 0D 00 00 00 E8 B0 EB FF FF BE 01 00 00 00 BF 1D 00 00 00  ..dH..%(...H.D$.1..........................
00001FEA  E8 A1 EB FF FF BE 01 00 00 00 BF 1D 00 00 00 E8 92 EB FF FF BA 00 00 00 00 BE 01 00 00 00 BF 02 00 00 00 E8 6E EC FF FF 89 C3 85  ..................................n........
00002015  C0 79 4F 48 B8 45 72 72 6F 72 3A 20 43 48 89 45 00 48 B8 6C 69 65 6E 74 20 75 6E 48 89 45 08 48 B8 61 62 6C 65 20 74 6F 20 48 89  .yOH.Error: CH.E.H.lient unH.E.H.able to H.
00002040  45 10 48 B8 63 72 65 61 74 65 20 73 48 89 45 18 C7 45 20 6F 63 6B 65 66 C7 45 24 74 00 B8 FF FF FF FF E9 0A 01 00 00 BF 60 27 40  E.H.create sH.E..E ockef.E$t...........`'@
0000206B  00 E8 2F EB FF FF 48 85 C0 75 68 48 B8 45 72 72 6F 72 3A 20 44 48 89 45 00 48 B8 4E 53 20 69 73 20 75 6E 48 89 45 08 48 B8 61 62  ../...H..uhH.Error: DH.E.H.NS is unH.E.H.ab
00002096  6C 65 20 74 6F 20 48 89 45 10 48 B8 72 65 73 6F 6C 76 65 20 48 89 45 18 48 B8 73 65 72 76 65 72 20 61 48 89 45 20 C7 45 28 64 64  le to H.E.H.resolve H.E.H.server aH.E .E(dd
000020C1  72 65 66 C7 45 2C 73 73 C6 45 2E 00 89 DF E8 7C EA FF FF B8 FF FF FF FF E9 93 00 00 00 48 C7 04 24 00 00 00 00 48 C7 44 24 08 00  ref.E,ss.E.....|.............H..$....H.D$..
000020EC  00 00 00 66 C7 04 24 02 00 48 63 50 14 48 8B 40 18 48 8D 7C 24 04 B9 0C 00 00 00 48 8B 30 E8 A1 EA FF FF 66 C7 44 24 02 3B 6E BA  ...f..$..HcP.H.@.H.|$......H.0.....f.D$.;n.
00002117  10 00 00 00 48 89 E6 89 DF E8 0B EB FF FF 85 C0 79 32 41 B8 60 27 40 00 B9 20 27 40 00 48 C7 C2 FF FF FF FF BE 01 00 00 00 48 89  ....H...........y2A.`'@.. '@.H..........H.
00002142  EF B8 00 00 00 00 E8 23 EB FF FF 89 DF E8 FC E9 FF FF B8 FF FF FF FF EB 16 89 DF E8 EE E9 FF FF 66 C7 45 00 4F 4B C6 45 02 00 B8  .......#.........................f.E.OK.E...
0000216D  00 00 00 00 48 8B 4C 24 18 64 48 33 0C 25 28 00 00 00 74 05 E8 AA E9 FF FF 48 83 C4 28 5B 5D C3 53 48 83 EC 10 48 89 CB 85 D2 74  ....H.L$.dH3.%(...t......H..([].SH...H....t
00002198  27 48 89 F2 BE 78 27 40 00 BF 01 00 00 00 B8 00 00 00 00 E8 50 EA FF FF 66 C7 03 4F 4B C6 43 02 00 B8 00 00 00 00 EB 3E 48 85 FF  'H...x'@............P...f..OK.C........>H..
000021C3  74 2B 80 3F 00 74 26 48 89 0C 24 49 89 F1 41 B8 EC 22 40 00 48 89 F9 BA 8F 27 40 00 BE 6E 3B 00 00 BF 60 27 40 00 E8 BE F5 FF FF  t+.?.t&H..$I..A.."@.H....'@..n;...`'@......
000021EE  EB 0E 66 C7 03 4F 4B C6 43 02 00 B8 00 00 00 00 48 83 C4 10 5B C3 90 90 90 90 90 90 90 90 90 90 90 90 48 89 6C 24 D8 4C 89 64 24  ..f..OK.C.......H...[.............H.l$.L.d$
00002219  E0 48 8D 2D DF 0B 20 00 4C 8D 25 D0 0B 20 00 4C 89 6C 24 E8 4C 89 74 24 F0 4C 89 7C 24 F8 48 89 5C 24 D0 48 83 EC 38 4C 29 E5 41  .H.-..  .L.%..  .L.l$.L.t$.L.|$.H.\$.H..8L).A
00002244  89 FD 49 89 F6 48 C1 FD 03 49 89 D7 E8 6B E8 FF FF 48 85 ED 74 1C 31 DB 0F 1F 40 00 4C 89 FA 4C 89 F6 44 89 EF 41 FF 14 DC 48 83  ..I..H...I...k...H..t.1...@.L..L..D..A...H.
0000226F  C3 01 48 39 EB 75 EA 48 8B 5C 24 08 48 8B 6C 24 10 4C 8B 64 24 18 4C 8B 6C 24 20 4C 8B 74 24 28 4C 8B 7C 24 30 48 83 C4 38 C3 0F  ..H9.u.H.\$.H.l$.L.d$.L.l$ L.t$(L.|$0H..8..
0000229A  1F 80 00 00 00 00 F3 C3 90 90 48 83 EC 08 48 83 C4 08 C3 00 00 00 01 00 02 00 72 00 25 73 3A 20 45 72 72 6F 72 3A 20 43 6F 75 6C  ..........H...H...........r.%s: Error: Coul
000022C5  64 6E 27 74 20 6F 70 65 6E 20 25 73 0A 00 55 73 61 67 65 3A 20 25 73 20 5B 3C 69 6E 70 75 74 5F 66 69 6C 65 3E 5D 0A 00 54 68 61  dn't open %s..Usage: %s [<input_file>]..Tha
000022F0  74 27 73 20 6E 75 6D 62 65 72 20 32 2E 20 20 4B 65 65 70 20 67 6F 69 6E 67 21 00 48 61 6C 66 77 61 79 20 74 68 65 72 65 21 00 47  t's number 2.  Keep going!.Halfway there!.G
0000231B  6F 6F 64 20 77 6F 72 6B 21 20 20 4F 6E 20 74 6F 20 74 68 65 20 6E 65 78 74 2E 2E 2E 00 57 65 6C 63 6F 6D 65 20 74 6F 20 6D 79 20  ood work!  On to the next....Welcome to my 
00002346  66 69 65 6E 64 69 73 68 20 6C 69 74 74 6C 65 20 62 6F 6D 62 2E 20 59 6F 75 20 68 61 76 65 20 36 20 70 68 61 73 65 73 20 77 69 74  fiendish little bomb. You have 6 phases wit
00002371  68 00 00 00 00 00 00 77 68 69 63 68 20 74 6F 20 62 6C 6F 77 20 79 6F 75 72 73 65 6C 66 20 75 70 2E 20 48 61 76 65 20 61 20 6E 69  h......which to blow yourself up. Have a ni
0000239C  63 65 20 64 61 79 21 00 00 00 00 00 50 68 61 73 65 20 31 20 64 65 66 75 73 65 64 2E 20 48 6F 77 20 61 62 6F 75 74 20 74 68 65 20  ce day!.....Phase 1 defused. How about the 
000023C7  6E 65 78 74 20 6F 6E 65 3F 00 00 00 00 00 00 00 00 53 6F 20 79 6F 75 20 67 6F 74 20 74 68 61 74 20 6F 6E 65 2E 20 20 54 72 79 20  next one?........So you got that one.  Try 
000023F2  74 68 69 73 20 6F 6E 65 2E 00 00 00 00 00 42 6F 72 64 65 72 20 72 65 6C 61 74 69 6F 6E 73 20 77 69 74 68 20 43 61 6E 61 64 61 20  this one......Border relations with Canada 
0000241D  68 61 76 65 20 6E 65 76 65 72 20 62 65 65 6E 20 62 65 74 74 65 72 2E 00 00 00 00 57 6F 77 21 20 59 6F 75 27 76 65 20 64 65 66 75  have never been better.....Wow! You've defu
00002448  73 65 64 20 74 68 65 20 73 65 63 72 65 74 20 73 74 61 67 65 21 00 66 6C 79 65 72 73 00 00 00 00 00 00 00 00 00 00 00 00 7C 0F 40  sed the secret stage!.flyers............|.@
00002473  00 00 00 00 00 B9 0F 40 00 00 00 00 00 83 0F 40 00 00 00 00 00 8A 0F 40 00 00 00 00 00 91 0F 40 00 00 00 00 00 98 0F 40 00 00 00  .......@.......@.......@.......@.......@...
0000249E  00 00 9F 0F 40 00 00 00 00 00 A6 0F 40 00 00 00 00 00 6D 61 64 75 69 65 72 73 6E 66 6F 74 76 62 79 6C 53 6F 20 79 6F 75 20 74 68  ....@.......@.....maduiersnfotvbylSo you th
000024C9  69 6E 6B 20 79 6F 75 20 63 61 6E 20 73 74 6F 70 20 74 68 65 20 62 6F 6D 62 20 77 69 74 68 20 63 74 72 6C 2D 63 2C 20 64 6F 20 79  ink you can stop the bomb with ctrl-c, do y
000024F4  6F 75 3F 00 43 75 72 73 65 73 2C 20 79 6F 75 27 76 65 20 66 6F 75 6E 64 20 74 68 65 20 73 65 63 72 65 74 20 70 68 61 73 65 21 00  ou?.Curses, you've found the secret phase!.
0000251F  00 42 75 74 20 66 69 6E 64 69 6E 67 20 69 74 20 61 6E 64 20 73 6F 6C 76 69 6E 67 20 69 74 20 61 72 65 20 71 75 69 74 65 20 64 69  .But finding it and solving it are quite di
0000254A  66 66 65 72 65 6E 74 2E 2E 2E 00 00 00 00 43 6F 6E 67 72 61 74 75 6C 61 74 69 6F 6E 73 21 20 59 6F 75 27 76 65 20 64 65 66 75 73  fferent.......Congratulations! You've defus
00002575  65 64 20 74 68 65 20 62 6F 6D 62 21 00 57 65 6C 6C 2E 2E 2E 00 4F 4B 2E 20 3A 2D 29 00 49 6E 76 61 6C 69 64 20 70 68 61 73 65 25  ed the bomb!.Well....OK. :-).Invalid phase%
000025A0  73 0A 00 0A 42 4F 4F 4D 21 21 21 00 54 68 65 20 62 6F 6D 62 20 68 61 73 20 62 6C 6F 77 6E 20 75 70 2E 00 25 64 20 25 64 20 25 64  s...BOOM!!!.The bomb has blown up..%d %d %d
000025CB  20 25 64 20 25 64 20 25 64 00 45 72 72 6F 72 3A 20 50 72 65 6D 61 74 75 72 65 20 45 4F 46 20 6F 6E 20 73 74 64 69 6E 00 47 52 41   %d %d %d.Error: Premature EOF on stdin.GRA
000025F6  44 45 5F 42 4F 4D 42 00 45 72 72 6F 72 3A 20 49 6E 70 75 74 20 6C 69 6E 65 20 74 6F 6F 20 6C 6F 6E 67 00 25 64 20 25 64 20 25 73  DE_BOMB.Error: Input line too long.%d %d %s
00002621  00 44 72 45 76 69 6C 00 67 72 65 61 74 77 68 69 74 65 2E 69 63 73 2E 63 73 2E 63 6D 75 2E 65 64 75 00 61 6E 67 65 6C 73 68 61 72  .DrEvil.greatwhite.ics.cs.cmu.edu.angelshar
0000264C  6B 2E 69 63 73 2E 63 73 2E 63 6D 75 2E 65 64 75 00 6D 61 6B 6F 73 68 61 72 6B 2E 69 63 73 2E 63 73 2E 63 6D 75 2E 65 64 75 00 00  k.ics.cs.cmu.edu.makoshark.ics.cs.cmu.edu..
00002677  00 50 72 6F 67 72 61 6D 20 74 69 6D 65 64 20 6F 75 74 20 61 66 74 65 72 20 25 64 20 73 65 63 6F 6E 64 73 0A 00 00 00 00 00 45 72  .Program timed out after %d seconds......Er
000026A2  72 6F 72 3A 20 48 54 54 50 20 72 65 71 75 65 73 74 20 66 61 69 6C 65 64 20 77 69 74 68 20 65 72 72 6F 72 20 25 64 3A 20 25 73 00  ror: HTTP request failed with error %d: %s.
000026CD  00 00 00 47 45 54 20 2F 25 73 2F 73 75 62 6D 69 74 72 2E 70 6C 2F 3F 75 73 65 72 69 64 3D 25 73 26 6C 61 62 3D 25 73 26 72 65 73  ...GET /%s/submitr.pl/?userid=%s&lab=%s&res
000026F8  75 6C 74 3D 25 73 26 73 75 62 6D 69 74 3D 73 75 62 6D 69 74 20 48 54 54 50 2F 31 2E 30 0D 0A 0D 0A 00 00 00 00 00 00 00 45 72 72  ult=%s&submit=submit HTTP/1.0...........Err
00002723  6F 72 3A 20 55 6E 61 62 6C 65 20 74 6F 20 63 6F 6E 6E 65 63 74 20 74 6F 20 73 65 72 76 65 72 20 25 73 00 00 00 25 25 25 30 32 58  or: Unable to connect to server %s...%%%02X
0000274E  00 25 73 20 25 64 20 25 5B 61 2D 7A 41 2D 7A 20 5D 00 63 68 61 6E 67 65 6D 65 2E 69 63 73 2E 63 73 2E 63 6D 75 2E 65 64 75 00 0A  .%s %d %[a-zA-z ].changeme.ics.cs.cmu.edu..
00002779  41 55 54 4F 52 45 53 55 4C 54 5F 53 54 52 49 4E 47 3D 25 73 0A 00 63 73 61 70 70 00 00 00 00 01 1B 03 3B 04 01 00 00 1F 00 00 00  AUTORESULT_STRING=%s..csapp.......;........
Offset: 0×2433; 0×34 bytes from 0×2400 to 0×2433 selected
```

## phase2

以下是 phase_2 的反汇编。

分析得到：

- 6个数字：400f05 处的函数调用 read_six_numbers
- 第一个数字是1：400f0a 处的比较
- 后一个数字是前一个数字2倍：400f17 到 400f2c 处的循环

而阅读函数 read_six_numbers 发现，字符到数字的转换是使用stl函数 sscanf 完成的，所以数字中间用空白字符(空格、tab等)隔开均可。

```
0000000000400efc <phase_2>:
  400efc:   55                   push   %rbp
  400efd:   53                   push   %rbx
  400efe:   48 83 ec 28          sub    $0x28,%rsp
  400f02:   48 89 e6             mov    %rsp,%rsi
  400f05:   e8 52 05 00 00       call   40145c <read_six_numbers>
  400f0a:   83 3c 24 01          cmpl   $0x1,(%rsp)
  400f0e:   74 20                je     400f30 <phase_2+0x34>
  400f10:   e8 25 05 00 00       call   40143a <explode_bomb>
  400f15:   eb 19                jmp    400f30 <phase_2+0x34>
  400f17:   8b 43 fc             mov    -0x4(%rbx),%eax
  400f1a:   01 c0                add    %eax,%eax
  400f1c:   39 03                cmp    %eax,(%rbx)
  400f1e:   74 05                je     400f25 <phase_2+0x29>
  400f20:   e8 15 05 00 00       call   40143a <explode_bomb>
  400f25:   48 83 c3 04          add    $0x4,%rbx
  400f29:   48 39 eb             cmp    %rbp,%rbx
  400f2c:   75 e9                jne    400f17 <phase_2+0x1b>
  400f2e:   eb 0c                jmp    400f3c <phase_2+0x40>
  400f30:   48 8d 5c 24 04       lea    0x4(%rsp),%rbx
  400f35:   48 8d 6c 24 18       lea    0x18(%rsp),%rbp
  400f3a:   eb db                jmp    400f17 <phase_2+0x1b>
  400f3c:   48 83 c4 28          add    $0x28,%rsp
  400f40:   5b                   pop    %rbx
  400f41:   5d                   pop    %rbp
  400f42:   c3                   ret
```

## phase3

```
0000000000400f43 <phase_3>:
  400f43:	48 83 ec 18          	sub    $0x18,%rsp
  400f47:	48 8d 4c 24 0c       	lea    0xc(%rsp),%rcx
  400f4c:	48 8d 54 24 08       	lea    0x8(%rsp),%rdx
  400f51:	be cf 25 40 00       	mov    $0x4025cf,%esi
  400f56:	b8 00 00 00 00       	mov    $0x0,%eax
  400f5b:	e8 90 fc ff ff       	call   400bf0 <__isoc99_sscanf@plt>
  400f60:	83 f8 01             	cmp    $0x1,%eax
  400f63:	7f 05                	jg     400f6a <phase_3+0x27>
  400f65:	e8 d0 04 00 00       	call   40143a <explode_bomb>
  400f6a:	83 7c 24 08 07       	cmpl   $0x7,0x8(%rsp)
  400f6f:	77 3c                	ja     400fad <phase_3+0x6a>
  400f71:	8b 44 24 08          	mov    0x8(%rsp),%eax
  400f75:	ff 24 c5 70 24 40 00 	jmp    *0x402470(,%rax,8)
  400f7c:	b8 cf 00 00 00       	mov    $0xcf,%eax
  400f81:	eb 3b                	jmp    400fbe <phase_3+0x7b>
  400f83:	b8 c3 02 00 00       	mov    $0x2c3,%eax
  400f88:	eb 34                	jmp    400fbe <phase_3+0x7b>
  400f8a:	b8 00 01 00 00       	mov    $0x100,%eax
  400f8f:	eb 2d                	jmp    400fbe <phase_3+0x7b>
  400f91:	b8 85 01 00 00       	mov    $0x185,%eax
  400f96:	eb 26                	jmp    400fbe <phase_3+0x7b>
  400f98:	b8 ce 00 00 00       	mov    $0xce,%eax
  400f9d:	eb 1f                	jmp    400fbe <phase_3+0x7b>
  400f9f:	b8 aa 02 00 00       	mov    $0x2aa,%eax
  400fa4:	eb 18                	jmp    400fbe <phase_3+0x7b>
  400fa6:	b8 47 01 00 00       	mov    $0x147,%eax
  400fab:	eb 11                	jmp    400fbe <phase_3+0x7b>
  400fad:	e8 88 04 00 00       	call   40143a <explode_bomb>
  400fb2:	b8 00 00 00 00       	mov    $0x0,%eax
  400fb7:	eb 05                	jmp    400fbe <phase_3+0x7b>
  400fb9:	b8 37 01 00 00       	mov    $0x137,%eax
  400fbe:	3b 44 24 0c          	cmp    0xc(%rsp),%eax
  400fc2:	74 05                	je     400fc9 <phase_3+0x86>
  400fc4:	e8 71 04 00 00       	call   40143a <explode_bomb>
  400fc9:	48 83 c4 18          	add    $0x18,%rsp
  400fcd:	c3                   	ret
```

还原出sscanf的函数调用：`sscanf(input, "%d "%d", 0x8(%rsp), 0xc(%rsp))`

第二个参数是因为内存`0x4025cf`保存的字符串是`"%d %d"`，所以输入字符串应该是两个数字，两个数字记为a、b。

- 根据位置`400f6a`的比较指令`cmpl $0x7,0x8(%rsp)`及之后的跳转：a<=7
- 根据`400f75`处的分支指令，程序会根据a的值为`%eax`赋值，最后将`$eax`和b比较，相等才能通过，由此a、b是一一对应的。

通过gdb调试，将所有8种情况列出：

| a | b |
|---|---|
| 0 | 207 |
| 1 | 311 |
| 2 | 707 |
| 3 | 256 |
| 4 | 389 |
| 5 | 206 |
| 6 | 682 |
| 7 | 327 |

## phase4

```
000000000040100c <phase_4>:
  40100c:	48 83 ec 18          	sub    $0x18,%rsp
  401010:	48 8d 4c 24 0c       	lea    0xc(%rsp),%rcx
  401015:	48 8d 54 24 08       	lea    0x8(%rsp),%rdx
  40101a:	be cf 25 40 00       	mov    $0x4025cf,%esi
  40101f:	b8 00 00 00 00       	mov    $0x0,%eax
  401024:	e8 c7 fb ff ff       	call   400bf0 <__isoc99_sscanf@plt>
  401029:	83 f8 02             	cmp    $0x2,%eax
  40102c:	75 07                	jne    401035 <phase_4+0x29>
  40102e:	83 7c 24 08 0e       	cmpl   $0xe,0x8(%rsp)
  401033:	76 05                	jbe    40103a <phase_4+0x2e>
  401035:	e8 00 04 00 00       	call   40143a <explode_bomb>
  40103a:	ba 0e 00 00 00       	mov    $0xe,%edx
  40103f:	be 00 00 00 00       	mov    $0x0,%esi
  401044:	8b 7c 24 08          	mov    0x8(%rsp),%edi
  401048:	e8 81 ff ff ff       	call   400fce <func4>
  40104d:	85 c0                	test   %eax,%eax
  40104f:	75 07                	jne    401058 <phase_4+0x4c>
  401051:	83 7c 24 0c 00       	cmpl   $0x0,0xc(%rsp)
  401056:	74 05                	je     40105d <phase_4+0x51>
  401058:	e8 dd 03 00 00       	call   40143a <explode_bomb>
  40105d:	48 83 c4 18          	add    $0x18,%rsp
  401061:	c3                   	ret
```

函数`phase_4`同样是从输入串中获取两个数字，分别记作a、b

- 结合 `401051` 处的比较指令和后续分支，判断b=0
- 结合 `40102e` 处的比较指令和后续分支，判断a<=14
- 调用func4： `func4(a,0,14)` ，需要满足其返回值为0( `40104d` 和 `40104f` 两条指令)，func4是一个递归函数，其伪代码如下：

  ```
  def f(a, b, c):
    if c < b:
      t = c - b + 1
    else:
      t = c - b
    t //= 2
    ecx = t + b
    if a <= ecx:
      t = 0
      if a < ecx:
        b = ecx + 1
        t = f(a, b, c)
        t = 2 * t + 1
    else:
      c = t + b - 1
      t = f(a, b, c)
      t *= 2
    return t
  ```

  看了很久也没看懂这个是干啥的，后来突然开窍，可以直接取特值。

  第一次调用，b=0，c=14，算出来t=ecx=7

  只需要a=7，就可以在if下边给t赋0，然后返回0。

## phase5

```
0000000000401062 <phase_5>:
  401062:	53                   	push   %rbx
  401063:	48 83 ec 20          	sub    $0x20,%rsp
  401067:	48 89 fb             	mov    %rdi,%rbx
  40106a:	64 48 8b 04 25 28 00 	mov    %fs:0x28,%rax
  401071:	00 00 
  401073:	48 89 44 24 18       	mov    %rax,0x18(%rsp)
  401078:	31 c0                	xor    %eax,%eax
  40107a:	e8 9c 02 00 00       	call   40131b <string_length>
  40107f:	83 f8 06             	cmp    $0x6,%eax
  401082:	74 4e                	je     4010d2 <phase_5+0x70>
  401084:	e8 b1 03 00 00       	call   40143a <explode_bomb>
  401089:	eb 47                	jmp    4010d2 <phase_5+0x70>
  40108b:	0f b6 0c 03          	movzbl (%rbx,%rax,1),%ecx
  40108f:	88 0c 24             	mov    %cl,(%rsp)
  401092:	48 8b 14 24          	mov    (%rsp),%rdx
  401096:	83 e2 0f             	and    $0xf,%edx
  401099:	0f b6 92 b0 24 40 00 	movzbl 0x4024b0(%rdx),%edx
  4010a0:	88 54 04 10          	mov    %dl,0x10(%rsp,%rax,1)
  4010a4:	48 83 c0 01          	add    $0x1,%rax
  4010a8:	48 83 f8 06          	cmp    $0x6,%rax
  4010ac:	75 dd                	jne    40108b <phase_5+0x29>
  4010ae:	c6 44 24 16 00       	movb   $0x0,0x16(%rsp)
  4010b3:	be 5e 24 40 00       	mov    $0x40245e,%esi
  4010b8:	48 8d 7c 24 10       	lea    0x10(%rsp),%rdi
  4010bd:	e8 76 02 00 00       	call   401338 <strings_not_equal>
  4010c2:	85 c0                	test   %eax,%eax
  4010c4:	74 13                	je     4010d9 <phase_5+0x77>
  4010c6:	e8 6f 03 00 00       	call   40143a <explode_bomb>
  4010cb:	0f 1f 44 00 00       	nopl   0x0(%rax,%rax,1)
  4010d0:	eb 07                	jmp    4010d9 <phase_5+0x77>
  4010d2:	b8 00 00 00 00       	mov    $0x0,%eax
  4010d7:	eb b2                	jmp    40108b <phase_5+0x29>
  4010d9:	48 8b 44 24 18       	mov    0x18(%rsp),%rax
  4010de:	64 48 33 04 25 28 00 	xor    %fs:0x28,%rax
  4010e5:	00 00 
  4010e7:	74 05                	je     4010ee <phase_5+0x8c>
  4010e9:	e8 42 fa ff ff       	call   400b30 <__stack_chk_fail@plt>
  4010ee:	48 83 c4 20          	add    $0x20,%rsp
  4010f2:	5b                   	pop    %rbx
  4010f3:	c3                   	ret
```

- 内存地址`0x40245e`处存储的字符串是`flyers`
- 内存地址`0x4024b0`开始的16个字符是： `maduiersnfotvbyl`

  flyers中6个字符对应的偏移量分别是：9, 15, 14, 5, 6, 7
- 根据汇编`40107f`处的指令，程序读入6个字符

- 根据 `40108b` ~ `4010ac` 处的指令，程序根据每个输入字符的低4位做索引，在上述内存地址 `0x4024b0` 处读取字符存入栈中，根据 `4010bd` 处的 `strings_not_equal` 函数调用，知道存入栈中的需要是字符串 `flyers`
- 综合以上，输入字符的高4位并无影响，低4位分别是：9, 15, 14, 5, 6, 7

```
  ) 9 I Y i y   © ¹ É Ù é
/ ? O _ o   ¯ ¿ Ï ß ï
. > N ^ n ~   ® ¾ Î Þ î
% 5 E U e u   ¥ µ Å Õ å
& 6 F V f v   ¦ ¶ Æ Ö æ
' 7 G W g w   § · Ç × ç
```

以上打印出了所有可能的结果，每一行的字符的低4位满足上述规则，为了满足测试，只需要在每一列中任选一个输入即可，如 `)/>%&'` 或者 `IONEFG` 等等

## phase6

```
00000000004010f4 <phase_6>:
  4010f4:   41 56                   push   %r14
  4010f6:   41 55                   push   %r13
  4010f8:   41 54                   push   %r12
  4010fa:   55                      push   %rbp
  4010fb:   53                      push   %rbx
  4010fc:   48 83 ec 50             sub    $0x50,%rsp
// read_six_numbers, 确定读进的是精确的6个数字
  401100:   49 89 e5                mov    %rsp,%r13
  401103:   48 89 e6                mov    %rsp,%rsi
  401106:   e8 51 03 00 00          call   40145c <read_six_numbers>
  40110b:   49 89 e6                mov    %rsp,%r14
  40110e:   41 bc 00 00 00 00       mov    $0x0,%r12d
// 两层循环：每个数字都<=0，6个数字各不相同
// 外层
  401114:   4c 89 ed                mov    %r13,%rbp
  401117:   41 8b 45 00             mov    0x0(%r13),%eax
  40111b:   83 e8 01                sub    $0x1,%eax
  40111e:   83 f8 05                cmp    $0x5,%eax
  401121:   76 05                   jbe    401128 <phase_6+0x34>
  401123:   e8 12 03 00 00          call   40143a <explode_bomb>
  401128:   41 83 c4 01             add    $0x1,%r12d
  40112c:   41 83 fc 06             cmp    $0x6,%r12d
  401130:   74 21                   je     401153 <phase_6+0x5f>
  401132:   44 89 e3                mov    %r12d,%ebx
// 内层
  401135:   48 63 c3                movslq %ebx,%rax
  401138:   8b 04 84                mov    (%rsp,%rax,4),%eax
  40113b:   39 45 00                cmp    %eax,0x0(%rbp)
  40113e:   75 05                   jne    401145 <phase_6+0x51> // 和栈顶不相等
  401140:   e8 f5 02 00 00          call   40143a <explode_bomb>
  401145:   83 c3 01                add    $0x1,%ebx
  401148:   83 fb 05                cmp    $0x5,%ebx
  40114b:   7e e8                   jle    401135 <phase_6+0x41>
  40114d:   49 83 c5 04             add    $0x4,%r13
  401151:   eb c1                   jmp    401114 <phase_6+0x20>
// ...
```

这个实在是太长了，看不完。但还是得硬着头皮看。

不过天无绝人之路，发现在`401114` ~ `401151`处是一个两层的循环，这两层循环保证了：

- 6个数字，每个都不能大于6
- 6个数字各不相同

还往下看吗？当然不！0~6共7个数，全排列的数量只有5040个，直接暴力求解(反正这个不扣分)。

用下边的脚本输出解：

```python
import subprocess

SUM = 6

EXEC = "./bomb"
ARGS = "passwd"

CMD = [EXEC, ARGS]

def check(lis: list):
    lis = [str(k) for k in lis]
    _in = " ".join(lis) + "\n"

    res = subprocess.run(CMD, input=_in.encode("utf8"), capture_output=True)
    if res.returncode == 0:
        print(_in)


def check_helper(lis):
    if len(lis) == SUM:
        #  print(lis)
        check(lis);
        return
    for i in range(SUM + 1):
        if i not in lis:
            lis.append(i)
            check_helper(lis)
            lis.pop()

if __name__ == '__main__':
    check_helper([])
```

结果输出：4 3 2 1 6 5