第一关：
汇编代码：

```
endbr64
sub    $0x8,%rsp
lea    0x1b9a(%rip),%rsi
callq  1aab <strings_not_equal>
test   %eax,%eax
jne    15c4 <phase_1+0x1d>
add    $0x8,%rsp
retq
callq  1cb2 <explode_bomb>
jmp    15bf <phase_1+0x18>
```

从代码中可见第一关是将两个字符串相比，不相等就爆炸，相等则通过。利用其中栈中存的是你输入的字符串，%rsi 中存的一个固定的字符串，利用 gdb 工具可知，该固定字符串为：“When I get angry, Mr. Biggleworth gets upset.”这就是第一关的通关密码。

第二关：
汇编代码：

```
00000000000015cb <phase_2>:
    15cb:       f3 0f 1e fa             endbr64
    15cf:       55                      push   %rbp
    15d0:       53                      push   %rbx
    15d1:       48 83 ec 28             sub    $0x28,%rsp
    15d5:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    15dc:       00 00
    15de:       48 89 44 24 18          mov    %rax,0x18(%rsp)
    15e3:       31 c0                   xor    %eax,%eax
    15e5:       48 89 e6                mov    %rsp,%rsi
    15e8:       e8 f1 06 00 00          callq  1cde <read_six_numbers>
    15ed:       83 3c 24 00             cmpl   $0x0,(%rsp)
    15f1:       78 0a                   js     15fd <phase_2+0x32>
    15f3:       48 89 e5                mov    %rsp,%rbp
    15f6:       bb 01 00 00 00          mov    $0x1,%ebx
    15fb:       eb 18                   jmp    1615 <phase_2+0x4a>
    15fd:       e8 b0 06 00 00          callq  1cb2 <explode_bomb>
    1602:       eb ef                   jmp    15f3 <phase_2+0x28>
    1604:       e8 a9 06 00 00          callq  1cb2 <explode_bomb>
    1609:       83 c3 01                add    $0x1,%ebx
    160c:       48 83 c5 04             add    $0x4,%rbp
    1610:       83 fb 06                cmp    $0x6,%ebx
    1613:       74 0c                   je     1621 <phase_2+0x56>
    1615:       89 d8                   mov    %ebx,%eax
    1617:       03 45 00                add    0x0(%rbp),%eax
    161a:       39 45 04                cmp    %eax,0x4(%rbp)
    161d:       74 ea                   je     1609 <phase_2+0x3e>
    161f:       eb e3                   jmp    1604 <phase_2+0x39>
    1621:       48 8b 44 24 18          mov    0x18(%rsp),%rax
    1626:       64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    162d:       00 00
    162f:       75 07                   jne    1638 <phase_2+0x6d>
    1631:       48 83 c4 28             add    $0x28,%rsp
    1635:       5b                      pop    %rbx
    1636:       5d                      pop    %rbp
    1637:       c3                      retq
    1638:       e8 e3 fb ff ff          callq  1220 <__stack_chk_fail@plt>
```

由代码可知你需要输入六个数，分析代码可知它从第一个数开始，将数的序号（1、2、3、4、5、6）存在%ebx 中，每分析一个数加一，与 6 相等则结束程序，而对于每一个数，它加上它的序号都必须等于它的下一个数。则假定第一个数为 x，那么这六个数就是：x、x+1、x+3、x+6、x+10、x+15，我将 x 设为 1，得到通关密码：“1 2 4 7 11 16”。

第三关：

汇编代码：

```
454 000000000000163d <phase_3>:
455     163d:       f3 0f 1e fa             endbr64
456     1641:       48 83 ec 18             sub    $0x18,%rsp
457     1645:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
458     164c:       00 00
459     164e:       48 89 44 24 08          mov    %rax,0x8(%rsp)
460     1653:       31 c0                   xor    %eax,%eax
461     1655:       48 8d 4c 24 04          lea    0x4(%rsp),%rcx
462     165a:       48 89 e2                mov    %rsp,%rdx
463     165d:       48 8d 35 a8 1c 00 00    lea    0x1ca8(%rip),%rsi        # 330c <array.3473+0x13c>
464     1664:       e8 57 fc ff ff          callq  12c0 <__isoc99_sscanf@plt>
465     1669:       83 f8 01                cmp    $0x1,%eax
466     166c:       7e 1a                   jle    1688 <phase_3+0x4b>
467     166e:       83 3c 24 07             cmpl   $0x7,(%rsp)
468     1672:       77 65                   ja     16d9 <phase_3+0x9c>
469     1674:       8b 04 24                mov    (%rsp),%eax
470     1677:       48 8d 15 32 1b 00 00    lea    0x1b32(%rip),%rdx        # 31b0 <_IO_stdin_used+0x1b0>
471     167e:       48 63 04 82             movslq (%rdx,%rax,4),%rax
472     1682:       48 01 d0                add    %rdx,%rax
473     1685:       3e ff e0                notrack jmpq *%rax
474     1688:       e8 25 06 00 00          callq  1cb2 <explode_bomb>
475     168d:       eb df                   jmp    166e <phase_3+0x31>
476     168f:       b8 b7 00 00 00          mov    $0xb7,%eax
477     1694:       39 44 24 04             cmp    %eax,0x4(%rsp)
478     1698:       75 52                   jne    16ec <phase_3+0xaf>
479     169a:       48 8b 44 24 08          mov    0x8(%rsp),%rax
480     169f:       64 48 33 04 25 28 00    xor    %fs:0x28,%rax
481     16a6:       00 00
482     16a8:       75 49                   jne    16f3 <phase_3+0xb6>
483     16aa:       48 83 c4 18             add    $0x18,%rsp
484     16ae:       c3                      retq
485     16af:       b8 6d 00 00 00          mov    $0x6d,%eax
486     16b4:       eb de                   jmp    1694 <phase_3+0x57>
487     16b6:       b8 29 03 00 00          mov    $0x329,%eax
488     16bb:       eb d7                   jmp    1694 <phase_3+0x57>
489     16bd:       b8 18 03 00 00          mov    $0x318,%eax
490     16c2:       eb d0                   jmp    1694 <phase_3+0x57>
491     16c4:       b8 3b 03 00 00          mov    $0x33b,%eax
492     16c9:       eb c9                   jmp    1694 <phase_3+0x57>
493     16cb:       b8 13 02 00 00          mov    $0x213,%eax
494     16d0:       eb c2                   jmp    1694 <phase_3+0x57>
495     16d2:       b8 b5 01 00 00          mov    $0x1b5,%eax
496     16d7:       eb bb                   jmp    1694 <phase_3+0x57>
497     16d9:       e8 d4 05 00 00          callq  1cb2 <explode_bomb>
498     16de:       b8 00 00 00 00          mov    $0x0,%eax
499     16e3:       eb af                   jmp    1694 <phase_3+0x57>
500     16e5:       b8 be 00 00 00          mov    $0xbe,%eax
501     16ea:       eb a8                   jmp    1694 <phase_3+0x57>
502     16ec:       e8 c1 05 00 00          callq  1cb2 <explode_bomb>
503     16f1:       eb a7                   jmp    169a <phase_3+0x5d>
504     16f3:       e8 28 fb ff ff          callq  1220 <__stack_chk_fail@plt>
```

分析代码得知需要输入两个数，第一个数大于 7 就爆炸，然后由 0x1685 行中的 jmp *%rax 可知此处为 switch 结构，根据输入的第一个数得到一个数，将这个数与输入的第二个数比较，不相等就爆炸。得到的第二个数存在了%eax 中，于是我输入 6 和随便一个数，由 gdb 工具得到 6 对应的数为 531，由此得到第三关的密码为“6 531”。

第四关：
汇编代码：

```
00000000000016f8 <func4>:
    16f8:       f3 0f 1e fa             endbr64
    16fc:       53                      push   %rbx
    16fd:       89 d0                   mov    %edx,%eax
    16ff:       29 f0                   sub    %esi,%eax
    1701:       89 c3                   mov    %eax,%ebx
    1703:       c1 eb 1f                shr    $0x1f,%ebx
    1706:       01 c3                   add    %eax,%ebx
    1708:       d1 fb                   sar    %ebx
    170a:       01 f3                   add    %esi,%ebx
    170c:       39 fb                   cmp    %edi,%ebx
    170e:       7f 06                   jg     1716 <func4+0x1e>
    1710:       7c 10                   jl     1722 <func4+0x2a>
    1712:       89 d8                   mov    %ebx,%eax
    1714:       5b                      pop    %rbx
    1715:       c3                      retq
    1716:       8d 53 ff                lea    -0x1(%rbx),%edx
    1719:       e8 da ff ff ff          callq  16f8 <func4>
    171e:       01 c3                   add    %eax,%ebx
    1720:       eb f0                   jmp    1712 <func4+0x1a>
    1722:       8d 73 01                lea    0x1(%rbx),%esi
    1725:       e8 ce ff ff ff          callq  16f8 <func4>
    172a:       01 c3                   add    %eax,%ebx
    172c:       eb e4                   jmp    1712 <func4+0x1a>
```

```
000000000000172e <phase_4>:
    172e:       f3 0f 1e fa             endbr64
    1732:       48 83 ec 18             sub    $0x18,%rsp
    1736:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    173d:       00 00
    173f:       48 89 44 24 08          mov    %rax,0x8(%rsp)
    1744:       31 c0                   xor    %eax,%eax
    1746:       48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    174b:       48 89 e2                mov    %rsp,%rdx
    174e:       48 8d 35 b7 1b 00 00    lea    0x1bb7(%rip),%rsi        # 330c <array.3473+0x13c>
    1755:       e8 66 fb ff ff          callq  12c0 <__isoc99_sscanf@plt>
    175a:       83 f8 02                cmp    $0x2,%eax
    175d:       75 06                   jne    1765 <phase_4+0x37>
    175f:       83 3c 24 0e             cmpl   $0xe,(%rsp)
    1763:       76 05                   jbe    176a <phase_4+0x3c>
    1765:       e8 48 05 00 00          callq  1cb2 <explode_bomb>
    176a:       ba 0e 00 00 00          mov    $0xe,%edx
    176f:       be 00 00 00 00          mov    $0x0,%esi
    1774:       8b 3c 24                mov    (%rsp),%edi
    1777:       e8 7c ff ff ff          callq  16f8 <func4>
    177c:       83 f8 07                cmp    $0x7,%eax
    177f:       75 07                   jne    1788 <phase_4+0x5a>
    1781:       83 7c 24 04 07          cmpl   $0x7,0x4(%rsp)
    1786:       74 05                   je     178d <phase_4+0x5f>
    1788:       e8 25 05 00 00          callq  1cb2 <explode_bomb>
    178d:       48 8b 44 24 08          mov    0x8(%rsp),%rax
    1792:       64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    1799:       00 00
    179b:       75 05                   jne    17a2 <phase_4+0x74>
    179d:       48 83 c4 18             add    $0x18,%rsp
    17a1:       c3                      retq
    17a2:       e8 79 fa ff ff          callq  1220 <__stack_chk_fail@plt>
```

分析代码知需要输入两个数，第一个数不能大于 14，然后将第一个数作为参数代入 fun4 函数，得到的结果如果不等于 7 就爆炸，然后比较第二个数，如果不等于 7 就爆炸。然后分析函数 fun7，发现是个递归函数，利用%edx 和%esi 中的值进行一系列的计算，得到的结果与输入的第一个数进行比较，如果相等结束递

归，返回计算结果，不相等继续递归。分析第一次递归时函数先计算（14>>31（逻辑）+14）>>1（算数），结果刚好为 7，那么如果输入的第一个数是 7 的话函数只递归一次，得到的结果刚好是 7，由此得到第四关的密码“7 7”。

第五关：
汇编代码：

```
00000000000017a7 <phase_5>:
    17a7:       f3 0f 1e fa             endbr64
    17ab:       53                      push   %rbx
    17ac:       48 83 ec 10             sub    $0x10,%rsp
    17b0:       48 89 fb                mov    %rdi,%rbx
    17b3:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    17ba:       00 00
    17bc:       48 89 44 24 08          mov    %rax,0x8(%rsp)
    17c1:       31 c0                   xor    %eax,%eax
    17c3:       e8 c2 02 00 00          callq  1a8a <string_length>
    17c8:       83 f8 06                cmp    $0x6,%eax
    17cb:       75 55                   jne    1822 <phase_5+0x7b>
    17cd:       b8 00 00 00 00          mov    $0x0,%eax
    17d2:       48 8d 0d f7 19 00 00    lea    0x19f7(%rip),%rcx        # 31d0 <array.3473>
    17d9:       0f b6 14 03             movzbl (%rbx,%rax,1),%edx
    17dd:       83 e2 0f                and    $0xf,%edx
    17e0:       0f b6 14 11             movzbl (%rcx,%rdx,1),%edx
    17e4:       88 54 04 01             mov    %dl,0x1(%rsp,%rax,1)
    17e8:       48 83 c0 01             add    $0x1,%rax
    17ec:       48 83 f8 06             cmp    $0x6,%rax
    17f0:       75 e7                   jne    17d9 <phase_5+0x32>
    17f2:       c6 44 24 07 00          movb   $0x0,0x7(%rsp)
    17f7:       48 8d 7c 24 01          lea    0x1(%rsp),%rdi
    17fc:       48 8d 35 a3 19 00 00    lea    0x19a3(%rip),%rsi        # 31a6 <_IO_stdin_used+0x1a6>
    1803:       e8 a3 02 00 00          callq  1aab <strings_not_equal>
    1808:       85 c0                   test   %eax,%eax
    180a:       75 1d                   jne    1829 <phase_5+0x82>
    180c:       48 8b 44 24 08          mov    0x8(%rsp),%rax
    1811:       64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    1818:       00 00
    181a:       75 14                   jne    1830 <phase_5+0x89>
    181c:       48 83 c4 10             add    $0x10,%rsp
    1820:       5b                      pop    %rbx
    1821:       c3                      retq
    1822:       e8 8b 04 00 00          callq  1cb2 <explode_bomb>
    1827:       eb a4                   jmp    17cd <phase_5+0x26>
    1829:       e8 84 04 00 00          callq  1cb2 <explode_bomb>
    182e:       eb dc                   jmp    180c <phase_5+0x65>
    1830:       e8 eb f9 ff ff          callq  1220 <__stack_chk_fail@plt>
```

分析代码可知需要输入长度为 6 的一个字符串，在 0x17d2 行读入了一个字符串存到%rcx 中，下面是一个循环，从输入字符串中依次提取出一个符号，该符号和 0xf 做与操作后得到一个数 n，然后提取出%rcx 存的字符串的第 n+1 个字符存到%rsp 中得到一个新的字符串，然后又读入一个字符串存到%rsi 中，比较这两个字符串是否相等，相等则过关。利用 gdb 得到%rcx 中的字符串为：“maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?”%rsi 中的字符串为“sabres”，将两个字符串一一对应得到输入字符串的六个字符对应的 n 值分别为：7、1、13、6、5、7 。先假设采用 16 进制，即输入 71d657，不通过，这时候利用 gdb 得到输入 71d657 时得到的新字符串为“saires”，发现 d 对应的 n 值为 4，即 d 在字母表中的位置，由此可利用字母表得到“gamfeg”，将该字符串输入，通过，第五关过关。

第六关：
汇编代码：

```
0000000000001835 <phase_6>:
    1835:       f3 0f 1e fa             endbr64
    1839:       41 56                   push   %r14
    183b:       41 55                   push   %r13
    183d:       41 54                   push   %r12
    183f:       55                      push   %rbp
    1840:       53                      push   %rbx
    1841:       48 83 ec 60             sub    $0x60,%rsp
    1845:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    184c:       00 00
    184e:       48 89 44 24 58          mov    %rax,0x58(%rsp)
    1853:       31 c0                   xor    %eax,%eax
    1855:       49 89 e5                mov    %rsp,%r13
    1858:       4c 89 ee                mov    %r13,%rsi
    185b:       e8 7e 04 00 00          callq  1cde <read_six_numbers>
    1860:       41 be 01 00 00 00       mov    $0x1,%r14d
    1866:       49 89 e4                mov    %rsp,%r12
    1869:       eb 28                   jmp    1893 <phase_6+0x5e>
    186b:       e8 42 04 00 00          callq  1cb2 <explode_bomb>
    1870:       eb 30                   jmp    18a2 <phase_6+0x6d>
    1872:       48 83 c3 01             add    $0x1,%rbx
    1876:       83 fb 05                cmp    $0x5,%ebx
    1879:       7f 10                   jg     188b <phase_6+0x56>
    187b:       41 8b 04 9c             mov    (%r12,%rbx,4),%eax
    187f:       39 45 00                cmp    %eax,0x0(%rbp)
    1882:       75 ee                   jne    1872 <phase_6+0x3d>
    1884:       e8 29 04 00 00          callq  1cb2 <explode_bomb>
    1889:       eb e7                   jmp    1872 <phase_6+0x3d>
    188b:       49 83 c6 01             add    $0x1,%r14
    188f:       49 83 c5 04             add    $0x4,%r13

1893:       4c 89 ed                mov    %r13,%rbp
1896:       41 8b 45 00             mov    0x0(%r13),%eax
189a:       83 e8 01                sub    $0x1,%eax
189d:       83 f8 05                cmp    $0x5,%eax
18a0:       77 c9                   ja     186b <phase_6+0x36>
18a2:       41 83 fe 05             cmp    $0x5,%r14d
18a6:       7f 05                   jg     18ad <phase_6+0x78>
18a8:       4c 89 f3                mov    %r14,%rbx
18ab:       eb ce                   jmp    187b <phase_6+0x46>
18ad:       be 00 00 00 00          mov    $0x0,%esi
18b2:       8b 0c b4                mov    (%rsp,%rsi,4),%ecx
18b5:       b8 01 00 00 00          mov    $0x1,%eax
18ba:       48 8d 15 5f 39 00 00    lea    0x395f(%rip),%rdx        # 5220 <node1>
18c1:       83 f9 01                cmp    $0x1,%ecx
18c4:       7e 0b                   jle    18d1 <phase_6+0x9c>
18c6:       48 8b 52 08             mov    0x8(%rdx),%rdx
18ca:       83 c0 01                add    $0x1,%eax
18cd:       39 c8                   cmp    %ecx,%eax
18cf:       75 f5                   jne    18c6 <phase_6+0x91>
18d1:       48 89 54 f4 20          mov    %rdx,0x20(%rsp,%rsi,8)
18d6:       48 83 c6 01             add    $0x1,%rsi
18da:       48 83 fe 06             cmp    $0x6,%rsi
18de:       75 d2                   jne    18b2 <phase_6+0x7d>
18e0:       48 8b 5c 24 20          mov    0x20(%rsp),%rbx
18e5:       48 8b 44 24 28          mov    0x28(%rsp),%rax
18ea:       48 89 43 08             mov    %rax,0x8(%rbx)
18ee:       48 8b 54 24 30          mov    0x30(%rsp),%rdx
18f3:       48 89 50 08             mov    %rdx,0x8(%rax)
18f7:       48 8b 44 24 38          mov    0x38(%rsp),%rax
18fc:       48 89 42 08             mov    %rax,0x8(%rdx)
```

```
1900:       48 8b 54 24 40          mov    0x40(%rsp),%rdx
1905:       48 89 50 08             mov    %rdx,0x8(%rax)
1909:       48 8b 44 24 48          mov    0x48(%rsp),%rax
190e:       48 89 42 08             mov    %rax,0x8(%rdx)
1912:       48 c7 40 08 00 00 00    movq   $0x0,0x8(%rax)
1919:       00
191a:       bd 05 00 00 00          mov    $0x5,%ebp
191f:       eb 09                   jmp    192a <phase_6+0xf5>
1921:       48 8b 5b 08             mov    0x8(%rbx),%rbx
1925:       83 ed 01                sub    $0x1,%ebp
1928:       74 11                   je     193b <phase_6+0x106>
192a:       48 8b 43 08             mov    0x8(%rbx),%rax
192e:       8b 00                   mov    (%rax),%eax
1930:       39 03                   cmp    %eax,(%rbx)
1932:       7d ed                   jge    1921 <phase_6+0xec>
1934:       e8 79 03 00 00          callq  1cb2 <explode_bomb>
1939:       eb e6                   jmp    1921 <phase_6+0xec>
193b:       48 8b 44 24 58          mov    0x58(%rsp),%rax
1940:       64 48 33 04 25 28 00    xor    %fs:0x28,%rax
1947:       00 00
1949:       75 0d                   jne    1958 <phase_6+0x123>
194b:       48 83 c4 60             add    $0x60,%rsp
194f:       5b                      pop    %rbx
1950:       5d                      pop    %rbp
1951:       41 5c                   pop    %r12
1953:       41 5d                   pop    %r13
1955:       41 5e                   pop    %r14
1957:       c3                      retq
1958:       e8 c3 f8 ff ff          callq  1220 <__stack_chk_fail@plt>
```

分析代码知需要输入 6 个数，这六个数都不能大于 6（无符号大于），而且这六个数互不相等，然后是一个循环，由 0x18ba 行可知，它先读入了一个新的数组设为 p，然后从输入的数组的第一个数开始，设该数为 n，将 p 的第 n 个提取出来形成新的数组设为 q，然后判断该数组是不是递减的，是则通过，不是则爆炸。此时先输入“6 5 4 3 2 1”，然后利用 gdb 得到数字 1-6 对应的数组 p 中的数分别为：0x103、0x1ad、0x0b0、0x3b6、0x204、0x274，将 1-6 按它们对应的值的从大到小顺序排序得到：4、6、5、2、1、3。得到通关密码：“4 6 5 2 1 3”。

秘密关卡：

1、寻找 secret_phase：

通过查找汇编代码，发现在函数 phase_defused 中有函数 secret_phase 的调用。

汇编代码：

```
0000000000001e6b <phase_defused>:
    1e6b:       f3 0f 1e fa             endbr64
    1e6f:       48 83 ec 78             sub    $0x78,%rsp
    1e73:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    1e7a:       00 00
    1e7c:       48 89 44 24 68          mov    %rax,0x68(%rsp)
    1e81:       31 c0                   xor    %eax,%eax
    1e83:       83 3d 22 38 00 00 06    cmpl   $0x6,0x3822(%rip)        # 56ac <num_input_strings>
    1e8a:       74 15                   je     1ea1 <phase_defused+0x36>
    1e8c:       48 8b 44 24 68          mov    0x68(%rsp),%rax
    1e91:       64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    1e98:       00 00
    1e9a:       75 73                   jne    1f0f <phase_defused+0xa4>
    1e9c:       48 83 c4 78             add    $0x78,%rsp
    1ea0:       c3                      retq
    1ea1:       48 8d 4c 24 0c          lea    0xc(%rsp),%rcx
    1ea6:       48 8d 54 24 08          lea    0x8(%rsp),%rdx
    1eab:       4c 8d 44 24 10          lea    0x10(%rsp),%r8
    1eb0:       48 8d 35 9f 14 00 00    lea    0x149f(%rip),%rsi        # 3356 <array.3473+0x186>
    1eb7:       48 8d 3d f2 38 00 00    lea    0x38f2(%rip),%rdi        # 57b0 <input_strings+0xf0>
    1ebe:       e8 fd f3 ff ff          callq  12c0 <__isoc99_sscanf@plt>
    1ec3:       83 f8 03                cmp    $0x3,%eax
    1ec6:       74 0e                   je     1ed6 <phase_defused+0x6b>
    1ec8:       48 8d 3d a9 13 00 00    lea    0x13a9(%rip),%rdi        # 3278 <array.3473+0xa8>
    1ecf:       e8 2c f3 ff ff          callq  1200 <puts@plt>
    1ed4:       eb b6                   jmp    1e8c <phase_defused+0x21>
    1ed6:       48 8d 7c 24 10          lea    0x10(%rsp),%rdi
    1edb:       48 8d 35 7d 14 00 00    lea    0x147d(%rip),%rsi        # 335f <array.3473+0x18f>
    1ee2:       e8 c4 fb ff ff          callq  1aab <strings_not_equal>
    1ee7:       85 c0                   test   %eax,%eax
    1ee9:       75 dd                   jne    1ec8 <phase_defused+0x5d>
    1eeb:       48 8d 3d 26 13 00 00    lea    0x1326(%rip),%rdi        # 3218 <array.3473+0x48>
    1ef2:       e8 09 f3 ff ff          callq  1200 <puts@plt>
    1ef7:       48 8d 3d 42 13 00 00    lea    0x1342(%rip),%rdi        # 3240 <array.3473+0x70>
    1efe:       e8 fd f2 ff ff          callq  1200 <puts@plt>
    1f03:       b8 00 00 00 00          mov    $0x0,%eax
    1f08:       e8 91 fa ff ff          callq  199e <secret_phase>
    1f0d:       eb b9                   jmp    1ec8 <phase_defused+0x5d>
    1f0f:       e8 0c f3 ff ff          callq  1220 <__stack_chk_fail@plt>
```

分析代码可知，代码首先判断拆除了多少个炸弹，如果拆除了 6 个炸弹进入下一步。然后代码读入了两个数据，用 gdb 工具可知%rdi 中存的是 phase_4 的通关密码，然后判断该密码的长度是不是 3，不是 3 结束程序，是 3 进入下一步。于是先将第四关的密码改为“7 7 d”，进入了下一步。发现代码将第四关密码的第三个字符串和另外一个字符串（存在%rsi）比较，相等则调用 secret_phase，不等则结束程序，利用 gdb 工具可知%rsi 中的字符串为“DrEvil”。所以如果第四关输入“7 7 DrEvil”就会进入秘密关卡。

2、破解 secret_phase。

汇编代码：

```
000000000000195d <fun7>:
    195d:       f3 0f 1e fa             endbr64
    1961:       48 85 ff                test   %rdi,%rdi
    1964:       74 32                   je     1998 <fun7+0x3b>
    1966:       48 83 ec 08             sub    $0x8,%rsp
    196a:       8b 17                   mov    (%rdi),%edx
    196c:       39 f2                   cmp    %esi,%edx
    196e:       7f 0c                   jg     197c <fun7+0x1f>
    1970:       b8 00 00 00 00          mov    $0x0,%eax
    1975:       75 12                   jne    1989 <fun7+0x2c>
    1977:       48 83 c4 08             add    $0x8,%rsp
    197b:       c3                      retq
    197c:       48 8b 7f 08             mov    0x8(%rdi),%rdi
    1980:       e8 d8 ff ff ff          callq  195d <fun7>
    1985:       01 c0                   add    %eax,%eax
    1987:       eb ee                   jmp    1977 <fun7+0x1a>
    1989:       48 8b 7f 10             mov    0x10(%rdi),%rdi
    198d:       e8 cb ff ff ff          callq  195d <fun7>
    1992:       8d 44 00 01             lea    0x1(%rax,%rax,1),%eax
    1996:       eb df                   jmp    1977 <fun7+0x1a>
    1998:       b8 ff ff ff ff          mov    $0xffffffff,%eax
    199d:       c3                      retq
```

```
000000000000199e <secret_phase>:
    199e:       f3 0f 1e fa             endbr64
    19a2:       53                      push   %rbx
    19a3:       e8 7b 03 00 00          callq  1d23 <read_line>
    19a8:       48 89 c7                mov    %rax,%rdi
    19ab:       ba 0a 00 00 00          mov    $0xa,%edx
    19b0:       be 00 00 00 00          mov    $0x0,%esi
    19b5:       e8 e6 f8 ff ff          callq  12a0 <strtol@plt>
    19ba:       48 89 c3                mov    %rax,%rbx
    19bd:       8d 40 ff                lea    -0x1(%rax),%eax
    19c0:       3d e8 03 00 00          cmp    $0x3e8,%eax
    19c5:       77 26                   ja     19ed <secret_phase+0x4f>
    19c7:       89 de                   mov    %ebx,%esi
    19c9:       48 8d 3d 70 37 00 00    lea    0x3770(%rip),%rdi        # 5140 <n1>
    19d0:       e8 88 ff ff ff          callq  195d <fun7>
    19d5:       83 f8 05                cmp    $0x5,%eax
    19d8:       75 1a                   jne    19f4 <secret_phase+0x56>
    19da:       48 8d 3d 9f 17 00 00    lea    0x179f(%rip),%rdi        # 3180 <_IO_stdin_used+0x180>
    19e1:       e8 1a f8 ff ff          callq  1200 <puts@plt>
    19e6:       e8 80 04 00 00          callq  1e6b <phase_defused>
    19eb:       5b                      pop    %rbx
    19ec:       c3                      retq
    19ed:       e8 c0 02 00 00          callq  1cb2 <explode_bomb>
    19f2:       eb d3                   jmp    19c7 <secret_phase+0x29>
    19f4:       e8 b9 02 00 00          callq  1cb2 <explode_bomb>
    19f9:       eb df                   jmp    19da <secret_phase+0x3c>
```

分析代码，发现我们需要输入一个数，该数首先要满足减去一后无符号不大于 1000，即该数小于等于 1001，然后代码读入了一个数，将这个数（设为 x 存在%rdi）和输入的数（设为 y 存在%rsi）作为参数调用函数 fun7。该函数是一个递归函数，如果 x>y，y 变为 8（%rdi），然后递归调用 fun7，将结果乘二返回；如果 x<y，y 变为 16（%rdi），然后递归调用 fun7，将结果乘二加一返回；如果 x=y ，返回 0。fun7 的函数结果如果等于 5 通关，否则不通关。显然 2*2+1=5，然后 1*2=2，然后 0*2+1=1。

一开始我以为 y 值的结构是链表，那么输入的值只需要大与第一个，小于第三个，大于第四个，等于第六个即可。可是我通过输入 1 来查看链表的值时发现第一个是 36，第三个是 6，不可能大于 36 又小于 6。我在这陷入了困境，后来通过输入不同的值发现输入的值不同，跳到同一个位置%rdi（y）中存的值不同，由此发现这不是简单的链表。然后由于第一个数 36 是固定的，于是我输入 37 得到第三个数为 50，第四个数为 45，然后再输入 47 得到第六个数为 47，由此得到了通关密码。后来在同学口中得知这是一个二叉树，所以走不同的路径得到的结果自然不同。