实验报告

phase_1

反汇编代码

```
0000000000002439 <phase_1>:
2439:  48 83 ec 08            sub    $0x8,%rsp
243d:  48 8d 35 04 1d 00 00   lea    0x1d04(%rip),%rsi        # 4148
<_IO_stdin_used+0x148>
2444:  e8 e0 05 00 00         callq  2a29 <strings_not_equal>
2449:  85 c0                  test   %eax,%eax
244b:  75 05                  jne    2452 <phase_1+0x19>
244d:  48 83 c4 08            add    $0x8,%rsp
2451:  c3                     retq
2452:  e8 ae 08 00 00         callq  2d05 <explode_bomb>
2457:  eb f4                  jmp    244d <phase_1+0x14>
```

解题步骤

首先函数没有改变%rdi中的值，并作为参数传入strings_not_equal函数中，显然当两个字符串不相等时%eax中的值为0，此时炸弹爆炸。所以我们需要读取数据段0x1d04(%rip)中的字符串，作为参数转给phase_1。

密码

Ques te anim? Ques sej mentalle? Ques io fuocca? Ques qui Diavola?

phase_2

反汇编代码

```
0000000000002459 <phase_2>:
2459:  55                     push   %rbp
245a:  53                     push   %rbx
245b:  48 83 ec 28            sub    $0x28,%rsp
245f:  64 48 8b 04 25 28 00   mov    %fs:0x28,%rax
2466:  00 00
2468:  48 89 44 24 18         mov    %rax,0x18(%rsp)
246d:  31 c0                  xor    %eax,%eax
246f:  48 89 e6               mov    %rsp,%rsi
2472:  e8 4e 09 00 00         callq  2dc5 <read_six_numbers>
2477:  83 3c 24 09            cmpl   $0x9,(%rsp)
247b:  75 0a                  jne    2487 <phase_2+0x2e>
247d:  48 89 e3               mov    %rsp,%rbx
2480:  48 8d 6c 24 14         lea    0x14(%rsp),%rbp
2485:  eb 10                  jmp    2497 <phase_2+0x3e>
2487:  e8 79 08 00 00         callq  2d05 <explode_bomb>
248c:  eb ef                  jmp    247d <phase_2+0x24>
248e:  48 83 c3 04            add    $0x4,%rbx
2492:  48 39 eb               cmp    %rbp,%rbx
2495:  74 15                  je     24ac <phase_2+0x53>
2497:  8b 05 87 3c 00 00      mov    0x3c87(%rip),%eax        # 6124 <mul.2>
249d:  0f af 03               imul   (%rbx),%eax
24a0:  39 43 04               cmp    %eax,0x4(%rbx)
24a3:  74 e9                  je     248e <phase_2+0x35>
24a5:  e8 5b 08 00 00         callq  2d05 <explode_bomb>
```

```
24aa:  eb e2                  jmp    248e <phase_2+0x35>
24ac:  48 8b 44 24 18         mov    0x18(%rsp),%rax
24b1:  64 48 2b 04 25 28 00   sub    %fs:0x28,%rax
24b8:  00 00
24ba:  75 07                  jne    24c3 <phase_2+0x6a>
24bc:  48 83 c4 28            add    $0x28,%rsp
24c0:  5b                     pop    %rbx
24c1:  5d                     pop    %rbp
24c2:  c3                     retq
24c3:  e8 d8 fb ff ff         callq  20a0 <__stack_chk_fail@plt>
```

解题步骤

首先函数调用read_six_numbers函数，显然是依次读取六个数放入栈中。首先将第一个数与0x9做对比，若不相等则爆炸。之后将最后一个数的地址传入%rbp中，每次将栈指针+4。当栈指针与%rbp的值相等时退出循环。循环体内每次比较前一个数的两倍是否与后一个数相等，若不等则爆炸。所以答案为以9为首项，2位公比，长度为6的等比数列。

密码

9 18 36 72 144 288

phase_3

反汇编代码

```
00000000000024c8 <phase_3>:
24c8:  48 83 ec 28            sub    $0x28,%rsp
24cc:  64 48 8b 04 25 28 00   mov    %fs:0x28,%rax
24d3:  00 00
24d5:  48 89 44 24 18         mov    %rax,0x18(%rsp)
24da:  31 c0                  xor    %eax,%eax
24dc:  48 8d 4c 24 0f         lea    0xf(%rsp),%rcx
24e1:  48 8d 54 24 10         lea    0x10(%rsp),%rdx
24e6:  4c 8d 44 24 14         lea    0x14(%rsp),%r8
24eb:  48 8d 35 c4 1c 00 00   lea    0x1cc4(%rip),%rsi        # 41b6
<_IO_stdin_used+0x1b6>
24f2:  e8 59 fc ff ff         callq  2150 <__isoc99_sscanf@plt>
24f7:  83 f8 02               cmp    $0x2,%eax
24fa:  7e 29                  jle    2525 <phase_3+0x5d>
24fc:  8b 05 1e 3c 00 00      mov    0x3c1e(%rip),%eax        # 6120 <mask.1>
2502:  30 44 24 0f            xor    %al,0xf(%rsp)
2506:  83 7c 24 10 07         cmpl   $0x7,0x10(%rsp)
250b:  0f 87 05 01 00 00      ja     2616 <phase_3+0x14e>
2511:  8b 44 24 10            mov    0x10(%rsp),%eax
2515:  48 8d 15 b4 1c 00 00   lea    0x1cb4(%rip),%rdx        # 41d0
<_IO_stdin_used+0x1d0>
251c:  48 63 04 82            movslq (%rdx,%rax,4),%rax
2520:  48 01 d0               add    %rdx,%rax
2523:  ff e0                  jmpq   *%rax
2525:  e8 db 07 00 00         callq  2d05 <explode_bomb>
252a:  eb d0                  jmp    24fc <phase_3+0x34>
252c:  b8 61 00 00 00         mov    $0x61,%eax
2531:  81 7c 24 14 7b 03 00   cmpl   $0x37b,0x14(%rsp)
```

```
2538:  00
2539:  0f 84 e1 00 00 00      je     2620 <phase_3+0x158>
253f:  e8 c1 07 00 00         callq  2d05 <explode_bomb>
2544:  b8 61 00 00 00         mov    $0x61,%eax
2549:  e9 d2 00 00 00         jmpq   2620 <phase_3+0x158>
254e:  b8 66 00 00 00         mov    $0x66,%eax
2553:  81 7c 24 14 de 03 00   cmpl   $0x3de,0x14(%rsp)
255a:  00
255b:  0f 84 bf 00 00 00      je     2620 <phase_3+0x158>
2561:  e8 9f 07 00 00         callq  2d05 <explode_bomb>
2566:  b8 66 00 00 00         mov    $0x66,%eax
256b:  e9 b0 00 00 00         jmpq   2620 <phase_3+0x158>
2570:  b8 76 00 00 00         mov    $0x76,%eax
2575:  81 7c 24 14 4e 01 00   cmpl   $0x14e,0x14(%rsp)
257c:  00
257d:  0f 84 9d 00 00 00      je     2620 <phase_3+0x158>
2583:  e8 7d 07 00 00         callq  2d05 <explode_bomb>
2588:  b8 76 00 00 00         mov    $0x76,%eax
258d:  e9 8e 00 00 00         jmpq   2620 <phase_3+0x158>
2592:  b8 7a 00 00 00         mov    $0x7a,%eax
2597:  81 7c 24 14 b7 02 00   cmpl   $0x2b7,0x14(%rsp)
259e:  00
259f:  74 7f                  je     2620 <phase_3+0x158>
25a1:  e8 5f 07 00 00         callq  2d05 <explode_bomb>
25a6:  b8 7a 00 00 00         mov    $0x7a,%eax
25ab:  eb 73                  jmp    2620 <phase_3+0x158>
25ad:  b8 68 00 00 00         mov    $0x68,%eax
25b2:  83 7c 24 14 4a         cmpl   $0x4a,0x14(%rsp)
25b7:  74 67                  je     2620 <phase_3+0x158>
25b9:  e8 47 07 00 00         callq  2d05 <explode_bomb>
25be:  b8 68 00 00 00         mov    $0x68,%eax
25c3:  eb 5b                  jmp    2620 <phase_3+0x158>
25c5:  b8 6a 00 00 00         mov    $0x6a,%eax
25ca:  81 7c 24 14 c9 02 00   cmpl   $0x2c9,0x14(%rsp)
25d1:  00
25d2:  74 4c                  je     2620 <phase_3+0x158>
25d4:  e8 2c 07 00 00         callq  2d05 <explode_bomb>
25d9:  b8 6a 00 00 00         mov    $0x6a,%eax
25de:  eb 40                  jmp    2620 <phase_3+0x158>
25e0:  b8 6d 00 00 00         mov    $0x6d,%eax
25e5:  81 7c 24 14 af 01 00   cmpl   $0x1af,0x14(%rsp)
25ec:  00
25ed:  74 31                  je     2620 <phase_3+0x158>
25ef:  e8 11 07 00 00         callq  2d05 <explode_bomb>
25f4:  b8 6d 00 00 00         mov    $0x6d,%eax
25f9:  eb 25                  jmp    2620 <phase_3+0x158>
25fb:  b8 73 00 00 00         mov    $0x73,%eax
2600:  81 7c 24 14 f3 02 00   cmpl   $0x2f3,0x14(%rsp)
2607:  00
2608:  74 16                  je     2620 <phase_3+0x158>
260a:  e8 f6 06 00 00         callq  2d05 <explode_bomb>
260f:  b8 73 00 00 00         mov    $0x73,%eax
2614:  eb 0a                  jmp    2620 <phase_3+0x158>
2616:  e8 ea 06 00 00         callq  2d05 <explode_bomb>
261b:  b8 6b 00 00 00         mov    $0x6b,%eax
```

```
2620:  38 44 24 0f             cmp    %al,0xf(%rsp)
2624:  75 15                   jne    263b <phase_3+0x173>
2626:  48 8b 44 24 18          mov    0x18(%rsp),%rax
262b:  64 48 2b 04 25 28 00    sub    %fs:0x28,%rax
2632:  00 00
2634:  75 0c                   jne    2642 <phase_3+0x17a>
2636:  48 83 c4 28             add    $0x28,%rsp
263a:  c3                      retq
263b:  e8 c5 06 00 00          callq  2d05 <explode_bomb>
2640:  eb e4                   jmp    2626 <phase_3+0x15e>
2642:  e8 59 fa ff ff          callq  20a0 <__stack_chk_fail@plt>
```

解题步骤

首先观察scanf的输入格式，为"%d %c %d"，则为两个整数与一个字符。其次函数主体为一个switch跳转表，同时它限制第一个输入的数字小于等于7，则推断第一个数字控制跳转表的分支。在每一个分支中要确保第三个数字等于某一个常数，第二个字符的ASCII码值异或32后等于某一个常数，否则爆炸。以此确定第2、3个参数的值。

密码

其中某一分支：0 A 891

phase_4

反汇编代码

phase_4主体代码

```
0000000000002683 <phase_4>:
2683:  48 83 ec 18             sub    $0x18,%rsp
2687:  64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
268e:  00 00
2690:  48 89 44 24 08          mov    %rax,0x8(%rsp)
2695:  31 c0                   xor    %eax,%eax
2697:  48 89 e1                mov    %rsp,%rcx
269a:  48 8d 54 24 04          lea    0x4(%rsp),%rdx
269f:  48 8d 35 3b 1f 00 00    lea    0x1f3b(%rip),%rsi        # 45e1
<array.0+0x3f1>
26a6:  e8 a5 fa ff ff          callq  2150 <__isoc99_sscanf@plt>
26ab:  83 f8 02                cmp    $0x2,%eax
26ae:  75 0b                   jne    26bb <phase_4+0x38>
26b0:  8b 04 24                mov    (%rsp),%eax
26b3:  83 e8 02                sub    $0x2,%eax
26b6:  83 f8 02                cmp    $0x2,%eax
26b9:  76 05                   jbe    26c0 <phase_4+0x3d>
26bb:  e8 45 06 00 00          callq  2d05 <explode_bomb>
26c0:  8b 34 24                mov    (%rsp),%esi
26c3:  bf 07 00 00 00          mov    $0x7,%edi
26c8:  e8 7a ff ff ff          callq  2647 <func4>
26cd:  39 44 24 04             cmp    %eax,0x4(%rsp)
26d1:  75 15                   jne    26e8 <phase_4+0x65>
26d3:  48 8b 44 24 08          mov    0x8(%rsp),%rax
26d8:  64 48 2b 04 25 28 00    sub    %fs:0x28,%rax
```

```
26df:  00 00
26e1:  75 0c                  jne    26ef <phase_4+0x6c>
26e3:  48 83 c4 18            add    $0x18,%rsp
26e7:  c3                     retq
26e8:  e8 18 06 00 00         callq  2d05 <explode_bomb>
26ed:  eb e4                  jmp    26d3 <phase_4+0x50>
26ef:  e8 ac f9 ff ff         callq  20a0 <__stack_chk_fail@plt>
```

fun4函数代码

```
0000000000002647 <func4>:
2647:  b8 00 00 00 00         mov    $0x0,%eax
264c:  85 ff                  test   %edi,%edi
264e:  7e 32                  jle    2682 <func4+0x3b>
2650:  41 54                  push   %r12
2652:  55                     push   %rbp
2653:  53                     push   %rbx
2654:  89 fb                  mov    %edi,%ebx
2656:  89 f5                  mov    %esi,%ebp
2658:  89 f0                  mov    %esi,%eax
265a:  83 ff 01               cmp    $0x1,%edi
265d:  74 1e                  je     267d <func4+0x36>
265f:  44 8d 66 01            lea    0x1(%rsi),%r12d
2663:  8d 7f ff               lea    -0x1(%rdi),%edi
2666:  44 89 e6               mov    %r12d,%esi
2669:  e8 d9 ff ff ff         callq  2647 <func4>
266e:  01 c5                  add    %eax,%ebp
2670:  8d 7b fe               lea    -0x2(%rbx),%edi
2673:  44 89 e6               mov    %r12d,%esi
2676:  e8 cc ff ff ff         callq  2647 <func4>
267b:  01 e8                  add    %ebp,%eax
267d:  5b                     pop    %rbx
267e:  5d                     pop    %rbp
267f:  41 5c                  pop    %r12
2681:  c3                     retq
2682:  c3                     retq
```

解题步骤

观察phase_4函数入口，发现是输入两个整数，第二个整数作为fun4函数的参数，第一个整数用于与fun4返回值做对比。若正确则拆弹成功。对于fun4函数，它是一个递归函数，共有两个参数，为phase_4中传入的立即数$7与我们输入的第二个参数。当第一个参数的值为0或1时作为递归终点返回，在整个函数体内一共两次递归调用fun4。因为fun4没有调用explore_bomb函数，且有递归终点，所以每一个参数输入都会有一个正确的值返回，那么只需要任意指定一个大于4的参数就可以得到对应的返回值，作为答案输入即可。

密码

其中一个密码为：246 4

phase_5

反汇编代码

```
00000000000026f4 <phase_5>:
26f4:  53                    push   %rbx
26f5:  48 83 ec 10           sub    $0x10,%rsp
26f9:  48 89 fb              mov    %rdi,%rbx
26fc:  64 48 8b 04 25 28 00  mov    %fs:0x28,%rax
2703:  00 00
2705:  48 89 44 24 08        mov    %rax,0x8(%rsp)
270a:  31 c0                 xor    %eax,%eax
270c:  e8 fb 02 00 00        callq  2a0c <string_length>
2711:  83 f8 06              cmp    $0x6,%eax
2714:  75 58                 jne    276e <phase_5+0x7a>
2716:  ba 00 00 00 00        mov    $0x0,%edx
271b:  48 8d 0d ce 1a 00 00  lea    0x1ace(%rip),%rcx        # 41f0 <array.0>
2722:  0f be 04 13           movsbl (%rbx,%rdx,1),%eax
2726:  83 c0 0f              add    $0xf,%eax
2729:  83 e0 0f              and    $0xf,%eax
272c:  0f b6 04 01           movzbl (%rcx,%rax,1),%eax
2730:  88 44 14 01           mov    %al,0x1(%rsp,%rdx,1)
2734:  48 83 c2 01           add    $0x1,%rdx
2738:  48 83 fa 06           cmp    $0x6,%rdx
273c:  75 e4                 jne    2722 <phase_5+0x2e>
273e:  c6 44 24 07 00        movb   $0x0,0x7(%rsp)
2743:  48 8d 7c 24 01        lea    0x1(%rsp),%rdi
2748:  48 8d 35 70 1a 00 00  lea    0x1a70(%rip),%rsi        # 41bf
<_IO_stdin_used+0x1bf>
274f:  e8 d5 02 00 00        callq  2a29 <strings_not_equal>
2754:  85 c0                 test   %eax,%eax
2756:  75 1d                 jne    2775 <phase_5+0x81>
2758:  48 8b 44 24 08        mov    0x8(%rsp),%rax
275d:  64 48 2b 04 25 28 00  sub    %fs:0x28,%rax
2764:  00 00
2766:  75 14                 jne    277c <phase_5+0x88>
2768:  48 83 c4 10           add    $0x10,%rsp
276c:  5b                    pop    %rbx
276d:  c3                    retq
276e:  e8 92 05 00 00        callq  2d05 <explode_bomb>
2773:  eb a1                 jmp    2716 <phase_5+0x22>
2775:  e8 8b 05 00 00        callq  2d05 <explode_bomb>
277a:  eb dc                 jmp    2758 <phase_5+0x64>
277c:  e8 1f f9 ff ff        callq  20a0 <__stack_chk_fail@plt>
```

解题思路

首先观察读入，发现函数开头调用了string_length函数且当字符串长度不等于6时炸弹爆炸，则代表需要读入长度为6的字符串。之后对于字符串的每一位循环进行处理。每次将字符的ASCII码取出加上0xf并取其低四位作为之后字符串寻址的偏移量，从%rip+0x1ace开始寻找与之后0x1a70 +%rip的字符串相等的字符串。那么我们只需要原字符串开始，找到偏移量，再以此退出我们需要输出的字符串每一位的ASCII码。若两字符串相等，则拆弹成功。

密码

:0?678

phase_6

part 1

反汇编代码

```
0000000000002781 <phase_6>:
2781:  41 57                 push   %r15
2783:  41 56                 push   %r14
2785:  41 55                 push   %r13
2787:  41 54                 push   %r12
2789:  55                    push   %rbp
278a:  53                    push   %rbx
278b:  48 83 ec 78           sub    $0x78,%rsp
278f:  64 48 8b 04 25 28 00  mov    %fs:0x28,%rax
2796:  00 00
2798:  48 89 44 24 68        mov    %rax,0x68(%rsp)
279d:  31 c0                 xor    %eax,%eax
279f:  4c 8d 74 24 10        lea    0x10(%rsp),%r14
27a4:  4c 89 74 24 08        mov    %r14,0x8(%rsp)
27a9:  4c 89 f6              mov    %r14,%rsi
27ac:  e8 14 06 00 00        callq  2dc5 <read_six_numbers>
27b1:  4d 89 f4              mov    %r14,%r12
27b4:  41 bf 01 00 00 00     mov    $0x1,%r15d
27ba:  4d 89 f5              mov    %r14,%r13
27bd:  e9 c6 00 00 00        jmpq   2888 <phase_6+0x107>


27cc:   48 83 c3 01            add    $0x1,%rbx
27d0:   83 fb 05               cmp    $0x5,%ebx
27d3:   0f 8f a7 00 00 00      jg     2880 <phase_6+0xff>
27d9:   41 8b 44 9d 00         mov    0x0(%r13,%rbx,4),%eax
27de:   39 45 00               cmp    %eax,0x0(%rbp)
27e1:   75 e9                  jne    27cc <phase_6+0x4b>
27e3:   e8 1d 05 00 00         callq  2d05 <explode_bomb>
27e8:   eb e2                  jmp    27cc <phase_6+0x4b>


2888:   4c 89 f5               mov    %r14,%rbp
288b:   41 8b 06               mov    (%r14),%eax
288e:   83 e8 01               sub    $0x1,%eax
2891:   83 f8 05               cmp    $0x5,%eax
2894:   0f 87 28 ff ff ff      ja     27c2 <phase_6+0x41>
289a:   41 83 ff 05            cmp    $0x5,%r15d
289e:   0f 8f 46 ff ff ff      jg     27ea <phase_6+0x69>
28a4:   4c 89 fb               mov    %r15,%rbx
28a7:   e9 2d ff ff ff         jmpq   27d9 <phase_6+0x58>
```

解题思路

在phase_6开头可以看到调用了read_six_numbers函数，说明此题时需要我们读入6个整数。接下来有两重循环，可以看到两个循环结束条件分别为%r15 > 5 与 %ebx > 5, 且在以%ebx为循环变量的循环中，每次%ebx中%r15开始每次判断栈中相邻两个元素的是否相等。因此可以推断出这两重循环是嵌套的关系，作用为判断输入的6个这整数互不相等且都小于等于6， 这六个整数为1-6的一个排列。

part 2

反汇编代码

```
27ea:   48 8b 54 24 08          mov    0x8(%rsp),%rdx
27ef:  48 83 c2 18              add    $0x18,%rdx
27f3:  b9 07 00 00 00           mov    $0x7,%ecx
27f8:  89 c8                    mov    %ecx,%eax
27fa:  41 2b 04 24              sub    (%r12),%eax
27fe:  41 89 04 24              mov    %eax,(%r12)
2802:  49 83 c4 04              add    $0x4,%r12
2806:  4c 39 e2                 cmp    %r12,%rdx
2809:  75 ed                    jne    27f8 <phase_6+0x77>
280b:  be 00 00 00 00           mov    $0x0,%esi
2810:  8b 4c b4 10              mov    0x10(%rsp,%rsi,4),%ecx
2814:  b8 01 00 00 00           mov    $0x1,%eax
2819:  48 8d 15 40 db 00 00     lea    0xdb40(%rip),%rdx        # 10360 <node1>
2820:  83 f9 01                 cmp    $0x1,%ecx
2823:  7e 0b                    jle    2830 <phase_6+0xaf>
2825:  48 8b 52 08              mov    0x8(%rdx),%rdx
2829:  83 c0 01                 add    $0x1,%eax
282c:  39 c8                    cmp    %ecx,%eax
282e:  75 f5                    jne    2825 <phase_6+0xa4>
2830:  48 89 54 f4 30           mov    %rdx,0x30(%rsp,%rsi,8)
2835:  48 83 c6 01              add    $0x1,%rsi
2839:  48 83 fe 06              cmp    $0x6,%rsi
283d:  75 d1                    jne    2810 <phase_6+0x8f>
283f:  48 8b 5c 24 30           mov    0x30(%rsp),%rbx
2844:  48 8b 44 24 38           mov    0x38(%rsp),%rax
2849:  48 89 43 08              mov    %rax,0x8(%rbx)
284d:  48 8b 54 24 40           mov    0x40(%rsp),%rdx
2852:  48 89 50 08              mov    %rdx,0x8(%rax)
2856:  48 8b 44 24 48           mov    0x48(%rsp),%rax
285b:  48 89 42 08              mov    %rax,0x8(%rdx)
285f:  48 8b 54 24 50           mov    0x50(%rsp),%rdx
2864:  48 89 50 08              mov    %rdx,0x8(%rax)
2868:  48 8b 44 24 58           mov    0x58(%rsp),%rax
286d:  48 89 42 08              mov    %rax,0x8(%rdx)
2871:  48 c7 40 08 00 00 00     movq   $0x0,0x8(%rax)
2878:  00
```

解题思路

之后判断为1-6的全排列后，进入此代码段。通过分析代码可以知道0xdb40+%rip的位置存了一个全局数组。之后通过两重循环，以我们输入的全排列与7取补后的排列为参照，将数组中每个元素的地址以此存入栈中。

part 3

反汇编代码

```
28ac:   48 8b 5b 08             mov    0x8(%rbx),%rbx
28b0:   83 ed 01                sub    $0x1,%ebp
28b3:   74 11                   je     28c6 <phase_6+0x145>
28b5:   48 8b 43 08             mov    0x8(%rbx),%rax
28b9:   8b 00                   mov    (%rax),%eax
28bb:   39 03                   cmp    %eax,(%rbx)
28bd:   7d ed                   jge    28ac <phase_6+0x12b>
28bf:   e8 41 04 00 00          callq  2d05 <explode_bomb>
28c4:   eb e6                   jmp    28ac <phase_6+0x12b>
```

解题思路

之后依次从栈中读取之前存入的地址，每次枚举相邻两个元素，判断大妈的大小关系。若前一个数小于后一个数，则爆炸。则需要保证排序后的数组为一个非降序列。所以此题为一个数组的冒泡排序，而数据是通过链表的方式存储的，同时在代码有排列与7取补，所以密码为数组下标非降排序后与7取补的结果。

密码

6 3 1 5 4 2

secret_phase

进入方式

在反汇编代码中寻找secret_phase调用的位置，发现在每一次phase_defused调用的时候。同时发现，每一次决定是否调用与%rip的值有关，因为隐藏关卡的触发提示是在phase_6完成之后，并且发现需要成功进入隐藏关卡的条件是输入两个整数后输入一个字符串作为钥匙。在全部的六关中，密码为两个整数为第四关，同时通过读取内存0x1346+%rip发现钥匙为Testify。

反汇编代码

secret_phase

```
000000000000294e <secret_phase>:
294e:   48 83 ec 18             sub    $0x18,%rsp
2952:   c7 44 24 0c 1b 00 00    movl   $0x1b,0xc(%rsp)
2959:   00
295a:   e8 a7 04 00 00          callq  2e06 <read_line>
295f:   48 89 c6                mov    %rax,%rsi
2962:   48 8d 3d 77 51 00 00    lea    0x5177(%rip),%rdi        # 7ae0 <t0>
2969:   e8 7c ff ff ff          callq  28ea <fun7>
296e:   8b 54 24 0c             mov    0xc(%rsp),%edx
2972:   39 c2                   cmp    %eax,%edx
2974:   75 16                   jne    298c <secret_phase+0x3e>
2976:   48 8d 3d 13 18 00 00    lea    0x1813(%rip),%rdi        # 4190
<_IO_stdin_used+0x190>
297d:   e8 ee f6 ff ff          callq  2070 <puts@plt>
2982:   e8 b9 05 00 00          callq  2f40 <phase_defused>
```

```
2987:  48 83 c4 18           add    $0x18,%rsp
298b:  c3                    retq
298c:  e8 74 03 00 00        callq  2d05 <explode_bomb>
2991:  eb e3                 jmp    2976 <secret_phase+0x28>
```

fun7

```
00000000000028ea <fun7>:
28ea:  55                    push   %rbp
28eb:  53                    push   %rbx
28ec:  48 83 ec 08           sub    $0x8,%rsp
28f0:  48 89 fb              mov    %rdi,%rbx
28f3:  48 89 f5              mov    %rsi,%rbp
28f6:  48 85 ff              test   %rdi,%rdi
28f9:  74 2b                 je     2926 <fun7+0x3c>
28fb:  0f b6 55 00           movzbl 0x0(%rbp),%edx
28ff:  84 d2                 test   %dl,%dl
2901:  74 2a                 je     292d <fun7+0x43>
2903:  80 fa 61              cmp    $0x61,%dl
2906:  74 29                 je     2931 <fun7+0x47>
2908:  0f be d2              movsbl %dl,%edx
290b:  83 ea 61              sub    $0x61,%edx
290e:  b8 01 00 00 00        mov    $0x1,%eax
2913:  39 d0                 cmp    %edx,%eax
2915:  74 1f                 je     2936 <fun7+0x4c>
2917:  83 c0 01              add    $0x1,%eax
291a:  83 f8 1a              cmp    $0x1a,%eax
291d:  75 f4                 jne    2913 <fun7+0x29>
291f:  e8 e1 03 00 00        callq  2d05 <explode_bomb>
2924:  eb 21                 jmp    2947 <fun7+0x5d>
2926:  e8 da 03 00 00        callq  2d05 <explode_bomb>
292b:  eb ce                 jmp    28fb <fun7+0x11>
292d:  8b 03                 mov    (%rbx),%eax
292f:  eb 16                 jmp    2947 <fun7+0x5d>
2931:  ba 00 00 00 00        mov    $0x0,%edx
2936:  48 8d 75 01           lea    0x1(%rbp),%rsi
293a:  48 63 d2              movslq %edx,%rdx
293d:  48 8b 7c d3 08        mov    0x8(%rb,x%rdx,8),%rdi
2942:  e8 a3 ff ff ff        callq  28ea <fun7>
2947:  48 83 c4 08           add    $0x8,%rsp
294b:  5b                    pop    %rbx
294c:  5d                    pop    %rbp
294d:  c3                    retq
```

解题思路

首先观察secret_phase函数中调用read_line函数，则需要读取一行并作为参数传入fun7中。其次发现正确返回的条件是调用fun7的返回值是否与27相等。则解题的关键在输入参数使fun7返回27。通过观察fun7，发现输入合法字符为a-z，每次通过输入字符与a的偏移量改变%rdi总的值，跳转内存。地址当字符为空时返回以%rdi为内存地址中的数。所以该函数的作用为输入一个字符串为Trie树中的路径以找到节点的值为27的节点。通过读取入口地址的相关值，找到0x1b与合法路径作为输入。

密码

sheriruth