北京科技大学实验报告

学院：计通学院　　专业：物联网工程　　班级：物联 201

姓名：赵方程　　学号：42024137　　日期：2021 年 10 月 28 日

实验名称：Bomblab 二进制炸弹实验

实验目的

通过拆解给定的二进制炸弹程序，熟悉 Linux 系统的使用，掌握程序反汇编和逆向工程的基本方法，理解汇编语言，学习使用调试器的方法。

实验环境

- 操作系统： Ubuntu 20.04.3 LTS
- GDB 版本：GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

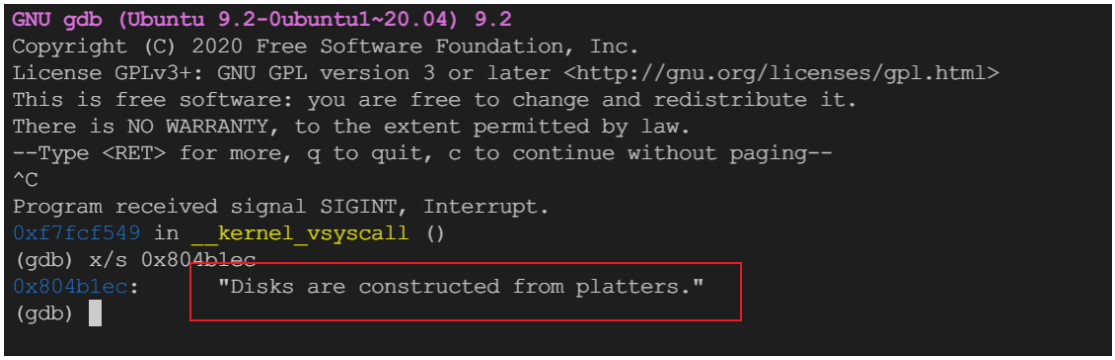
实验内容与步骤

Phase 0

```
08049587 <phase_0>:
 8049587:   f3 0f 1e fb             endbr32
 804958b:   55                      push   %ebp
 804958c:   89 e5                   mov    %esp,%ebp
 804958e:   83 ec 08                sub    $0x8,%esp
 8049591:   83 ec 08                sub    $0x8,%esp
 8049594:   68 ec b1 04 08          push   $0x804b1ec     //压入栈
 8049599:   ff 75 08                pushl  0x8(%ebp)      //压入栈
 804959c:   e8 f6 07 00 00          call   8049d97 <strings_not_equal>
 80495a1:   83 c4 10                add    $0x10,%esp
 80495a4:   85 c0                   test   %eax,%eax      //按位与
 80495a6:   74 0c                   je     80495b4 <phase_0+0x2d>//jump
 80495a8:   e8 6a 0a 00 00          call   804a017 <explode_bomb>
 80495ad:   b8 00 00 00 00          mov    $0x0,%eax
 80495b2:   eb 05                   jmp    80495b9 <phase_0+0x32>
 80495b4:   b8 01 00 00 00          mov    $0x1,%eax
 80495b9:   c9                      leave
 80495ba:   c3                      ret
```

读题可知，入栈的 0x8(%esp)应该与 0x804b1ec 处字符串相等。

使用 GDB 查看



可知答案

Phase 1

```
 080495bb <phase_1>:
80495bb:   f3 0f 1e fb             endbr32
80495bf:   55                      push   %ebp
80495c0:   89 e5                   mov    %esp,%ebp
80495c2:   83 ec 38                sub    $0x38,%esp
80495c5:   8b 45 08                mov    0x8(%ebp),%eax
80495c8:   89 45 d4                mov    %eax,-0x2c(%ebp)
80495cb:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
80495d1:   89 45 f4                mov    %eax,-0xc(%ebp)
80495d4:   31 c0                   xor    %eax,%eax
80495d6:   c7 45 f0 66 75 27 0b    movl   $0xb277566,-0x10(%ebp)
80495dd:   db 45 f0                fildl  -0x10(%ebp)
80495e0:   d9 5d e4                fstps  -0x1c(%ebp)
80495e3:   8d 45 ec                lea    -0x14(%ebp),%eax
80495e6:   50                      push   %eax
80495e7:   8d 45 e8                lea    -0x18(%ebp),%eax
80495ea:   50                      push   %eax
80495eb:   68 11 b2 04 08          push   $0x804b211       //"%d %d"
80495f0:   ff 75 d4                pushl  -0x2c(%ebp)
80495f3:   e8 f8 fb ff ff         call   80491f0 <__isoc99_sscanf@plt>
80495f8:   83 c4 10                add    $0x10,%esp
80495fb:   83 f8 02                cmp    $0x2,%eax        //输入两个数
80495fe:   74 0c                   je     804960c <phase_1+0x51>
8049600:   e8 12 0a 00 00          call   804a017 <explode_bomb>
8049605:   b8 00 00 00 00          mov    $0x0,%eax
804960a:   eb 34                   jmp    8049640 <phase_1+0x85>
804960c:   8d 45 e4                lea    -0x1c(%ebp),%eax
804960f:   0f b7 00                movzwl (%eax),%eax
8049612:   0f bf d0                movswl %ax,%edx
```

```
8049615:   8b 45 e8              mov    -0x18(%ebp),%eax
8049618:   39 c2                 cmp    %eax,%edx    //break 查看 reg
804961a:   75 13                 jne    804962f <phase_1+0x74>
804961c:   8d 45 e4              lea    -0x1c(%ebp),%eax
804961f:   83 c0 02              add    $0x2,%eax
8049622:   0f b7 00              movzwl (%eax),%eax
8049625:   0f bf d0              movswl %ax,%edx
8049628:   8b 45 ec              mov    -0x14(%ebp),%eax
804962b:   39 c2                 cmp    %eax,%edx    //break 查看 reg
804962d:   74 0c                 je     804963b <phase_1+0x80>
804962f:   e8 e3 09 00 00        call   804a017 <explode_bomb>
8049634:   b8 00 00 00 00        mov    $0x0,%eax
8049639:   eb 05                 jmp    8049640 <phase_1+0x85>
804963b:   b8 01 00 00 00        mov    $0x1,%eax
8049640:   8b 4d f4              mov    -0xc(%ebp),%ecx
8049643:   65 33 0d 14 00 00 00  xor    %gs:0x14,%ecx
804964a:   74 05                 je     8049651 <phase_1+0x96>
804964c:   e8 3f fb ff ff        call 8049190 <__stack_chk_fail@plt>
8049651:   c9                    leave
8049652:   c3                    ret
```

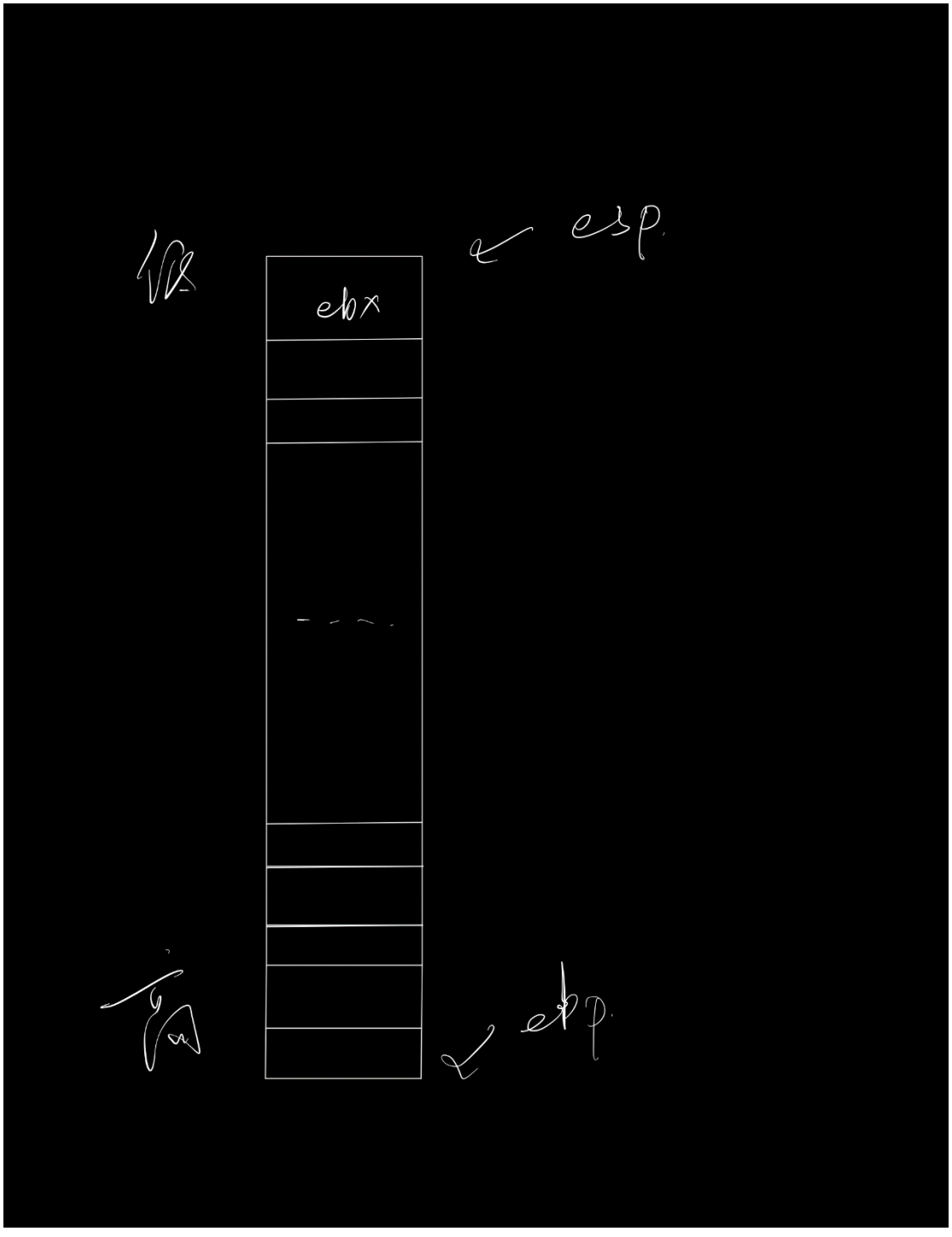
观察可知在 0x8049618 处比较 eax 与 edx 的值，不相等则跳转调用 explode_bomb

故 两次分别在观察 reg 可得

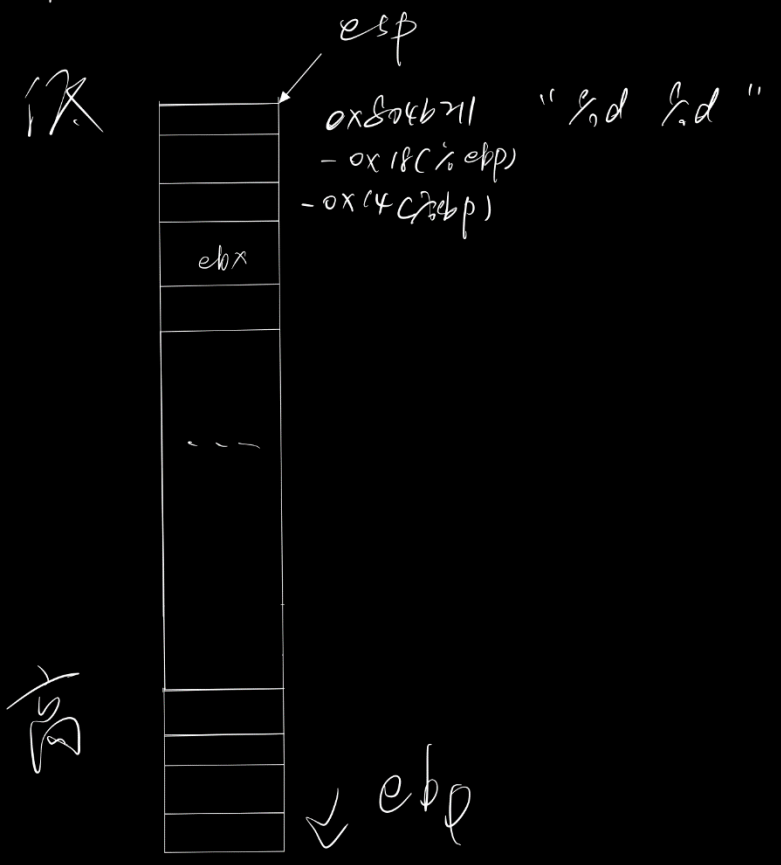
输入值为 30550 19762

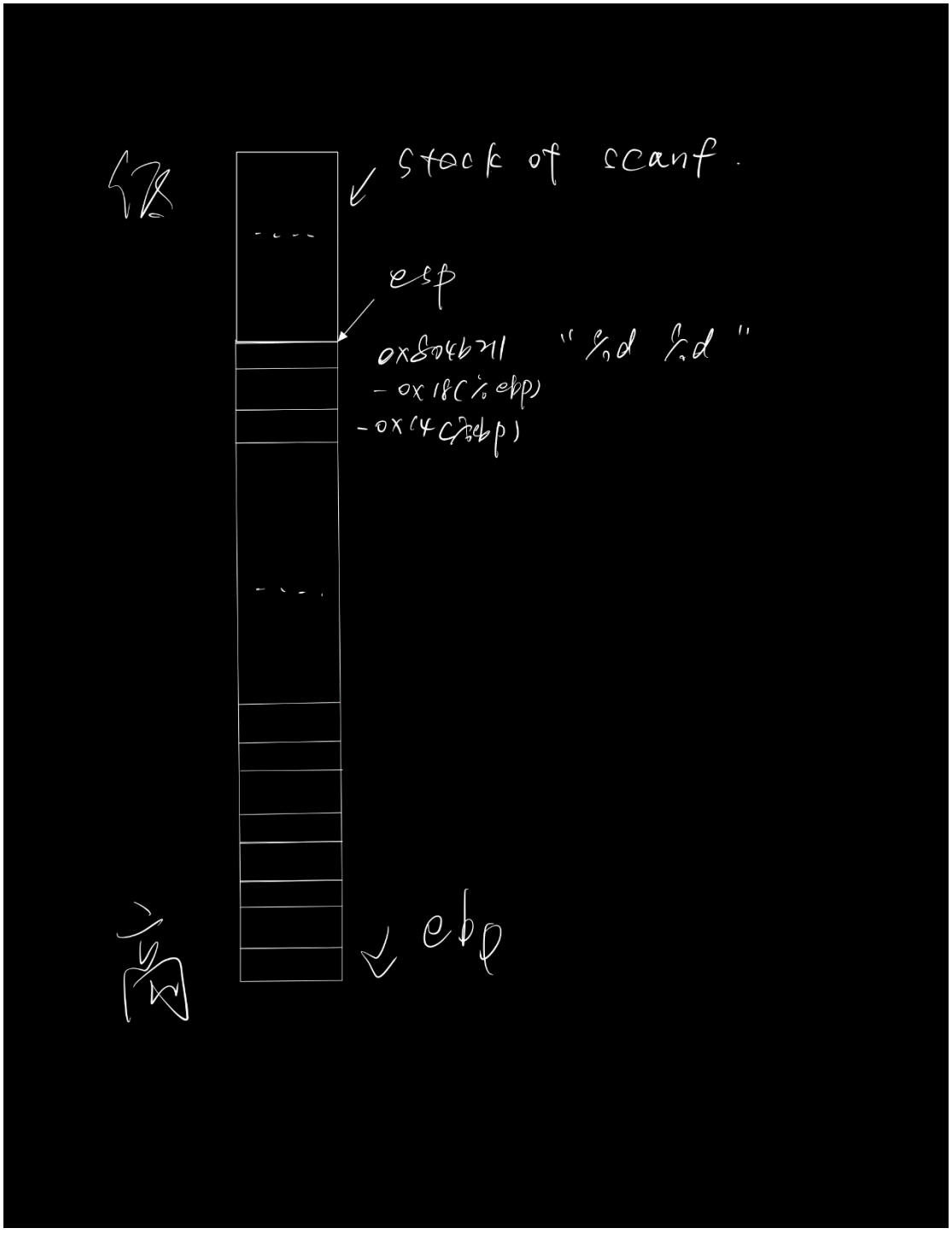
```
eax            0x1                 1
ecx            0x0                 0
edx            0x7756              30550
ebx            0xffffcfb0          -12368
esp            0xffffcf30          0xffffcf30
ebp            0xffffcf68          0xffffcf68
esi            0xf7fb6000          -134520832
edi            0xf7fb6000          -134520832
eip            0x8049618           0x8049618 <phase_1+93>
eflags         0x246               [ PF ZF IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
gs             0x63                99
```

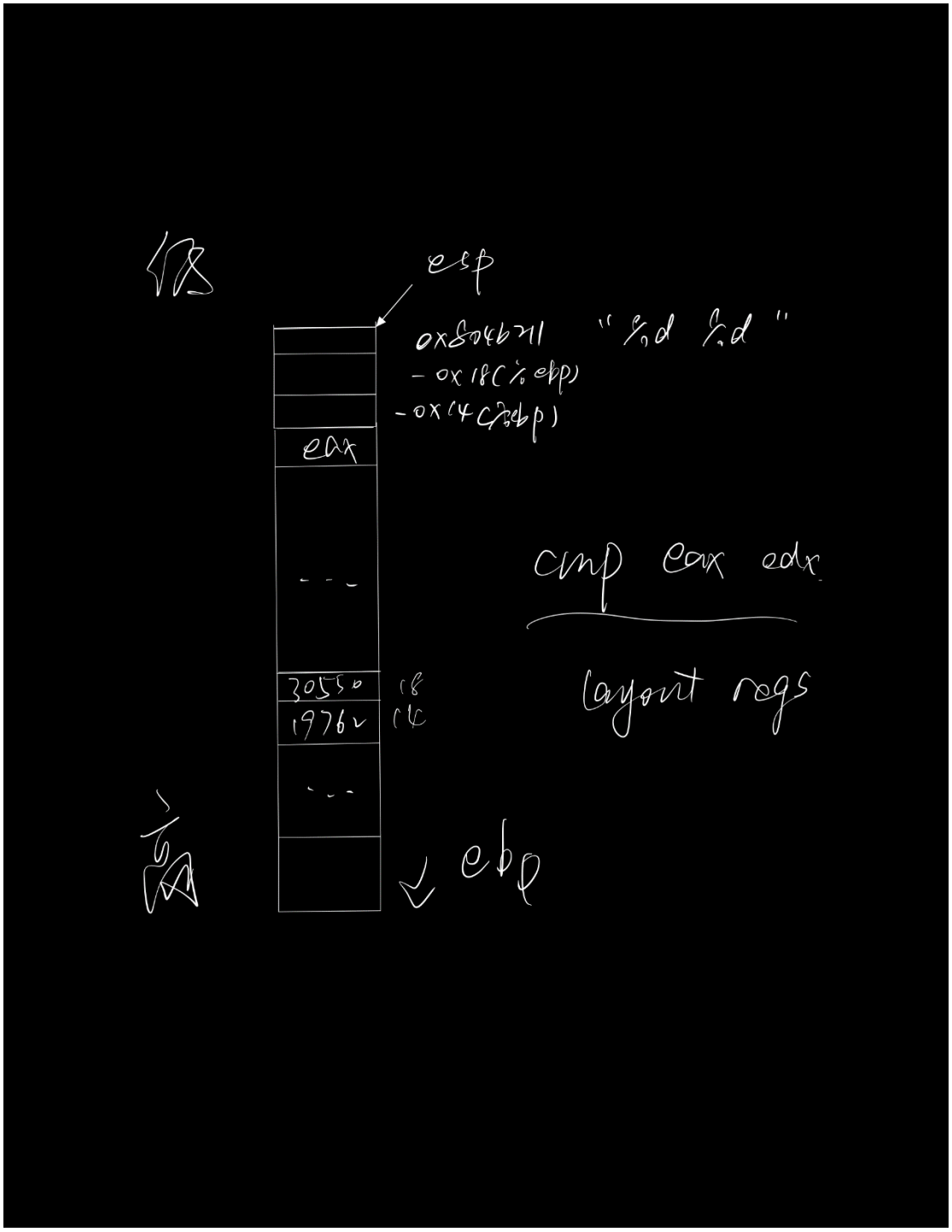
eax	0x4d32	19762
edx	0x4d32	19762
esp	0xffffcf30	0xffffcf30
esi	0xf7fb6000	-134520832
eip	0x804962b	0x804962b <phase_1+112>
cs	0x23	35
ds	0x2b	43
fs	0x0	0



```
lea   -0x14(%ebp), %eax
push  %eax
lea   -0x18(%ebp), %eax
push  %eax
push  $0x804b211
```




stack of scanf.
esp
0x8048611 "%d %d"
-0x18(%ebp)
-0x14(%ebp)
ebp



Phase_2

```
08049653 <phase_2>:
 8049653:  f3 0f 1e fb           endbr32
 8049657:  55                    push   %ebp
 8049658:  89 e5                 mov    %esp,%ebp
 804965a:  83 ec 48              sub    $0x48,%esp
 804965d:  8b 45 08              mov    0x8(%ebp),%eax
 8049660:  89 45 c4              mov    %eax,-0x3c(%ebp)
```

```
8049663:  65 a1 14 00 00 00     mov    %gs:0x14,%eax
8049669:  89 45 f4              mov    %eax,-0xc(%ebp)
804966c:  31 c0                 xor    %eax,%eax
804966e:  83 ec 04              sub    $0x4,%esp
8049671:  6a 09                 push   $0x9           //入栈   9
8049673:  8d 45 d0              lea    -0x30(%ebp),%eax
8049676:  50                    push   %eax
8049677:  ff 75 c4              pushl  -0x3c(%ebp)
804967a:  e8 56 06 00 00        call   8049cd5 <read_n_numbers>
804967f:  83 c4 10              add    $0x10,%esp
8049682:  85 c0                 test   %eax,%eax
8049684:  75 07                 jne    804968d <phase_2+0x3a>
8049686:  b8 00 00 00 00        mov    $0x0,%eax
804968b:  eb 59                 jmp    80496e6 <phase_2+0x93>
804968d:  8b 45 d0              mov    -0x30(%ebp),%eax
8049690:  3d 90 00 00 00        cmp    $0x90,%eax    //eax==144
8049695:  74 0c                 je     80496a3 <phase_2+0x50>
8049697:  e8 7b 09 00 00        call   804a017 <explode_bomb>
804969c:  b8 00 00 00 00        mov    $0x0,%eax
80496a1:  eb 43                 jmp    80496e6 <phase_2+0x93>
80496a3:  c7 45 cc 01 00 00 00  movl   $0x1,-0x34(%ebp)
80496aa:  eb 2f                 jmp    80496db <phase_2+0x88>
80496ac:  8b 45 cc              mov    -0x34(%ebp),%eax    //loop begin
80496af:  8b 44 85 d0           mov    -0x30(%ebp,%eax,4),%eax
80496b3:  8b 55 cc              mov    -0x34(%ebp),%edx
80496b6:  83 ea 01              sub    $0x1,%edx
80496b9:  8b 54 95 d0           mov    -0x30(%ebp,%edx,4),%edx
80496bd:  8b 4d cc              mov    -0x34(%ebp),%ecx
80496c0:  01 c9                 add    %ecx,%ecx
80496c2:  29 ca                 sub    %ecx,%edx
80496c4:  83 c2 01              add    $0x1,%edx
80496c7:  39 d0                 cmp    %edx,%eax
80496c9:  74 0c                 je    80496d7 <phase_2+0x84> //loop end
80496cb:  e8 47 09 00 00        call   804a017 <explode_bomb>
80496d0:  b8 00 00 00 00        mov    $0x0,%eax
80496d5:  eb 0f                 jmp    80496e6 <phase_2+0x93>
80496d7:  83 45 cc 01           addl   $0x1,-0x34(%ebp)
80496db:  83 7d cc 08           cmpl   $0x8,-0x34(%ebp)
80496df:  7e cb                 jle    80496ac <phase_2+0x59>
80496e1:  b8 01 00 00 00        mov    $0x1,%eax
80496e6:  8b 4d f4              mov    -0xc(%ebp),%ecx
80496e9:  65 33 0d 14 00 00 00  xor    %gs:0x14,%ecx
80496f0:  74 05                 je     80496f7 <phase_2+0xa4>
80496f2:  e8 99 fa ff ff        call   8049190 <__stack_chk_fail@plt>
```

```
 80496f7:  c9                    leave
 80496f8:  c3                    ret
```

观察可知 loop 中

Array[i+1]=Array[i]-1-2*i

Array[0] = 144

Nums = 9

所以有结果 144 143 140 135 128 119 108 95 80

Phase_3

```
080496f9 <phase_3>:
 80496f9:  f3 0f 1e fb           endbr32
 80496fd:  55                    push   %ebp
 80496fe:  89 e5                 mov    %esp,%ebp
 8049700:  83 ec 38              sub    $0x38,%esp
 8049703:  8b 45 08              mov    0x8(%ebp),%eax
 8049706:  89 45 d4              mov    %eax,-0x2c(%ebp)
 8049709:  65 a1 14 00 00 00     mov    %gs:0x14,%eax
 804970f:  89 45 f4              mov    %eax,-0xc(%ebp)
 8049712:  31 c0                 xor    %eax,%eax          //0
 8049714:  8d 45 e8              lea    -0x18(%ebp),%eax
 8049717:  50                    push   %eax
 8049718:  8d 45 e4              lea    -0x1c(%ebp),%eax
 804971b:  50                    push   %eax
 804971c:  68 11 b2 04 08        push   $0x804b211     //"%d %d"
 8049721:  ff 75 d4              pushl  -0x2c(%ebp)
 8049724:  e8 c7 fa ff ff        call   80491f0 <__isoc99_sscanf@plt>
 8049729:  83 c4 10              add    $0x10,%esp
 804972c:  89 45 f0              mov    %eax,-0x10(%ebp)
 804972f:  83 7d f0 01           cmpl   $0x1,-0x10(%ebp)  //返回值要大于
1
 8049733:  7f 0c                 jg     8049741 <phase_3+0x48>   //jump1
 8049735:  e8 dd 08 00 00        call   804a017 <explode_bomb>
 804973a:  b8 00 00 00 00        mov    $0x0,%eax
 804973f:  eb 7c                 jmp    80497bd <phase_3+0xc4>
 8049741:  c7 45 ec 00 00 00 00  movl   $0x0,-0x14(%ebp)         //jump1
here
 8049748:  8b 45 e4              mov    -0x1c(%ebp),%eax
 804974b:  83 e8 25              sub    $0x25,%eax                //eax-
```

```
=37
 804974e:  83 f8 08              cmp  $0x8,%eax  //if  8<eax,then=>eax<=8
//=>eax<=45
 8049751:  77 45                 ja              8049798   <phase_3+0x9f>
//jump2==explod
 8049753:  8b 04 85 18 b2 04 08  mov             0x804b218(,%eax,4),%eax
//0x804b218+4*%eax
 804975a:  3e ff e0              notrack            jmp             *%eax
//switch
 804975d:  83 45 ec 78           addl   $0x78,-0x14(%ebp)         //+120
 8049761:  81 45 ec 24 03 00 00  addl   $0x324,-0x14(%ebp)        //+804
 8049768:  83 6d ec 78           subl   $0x78,-0x14(%ebp)         //-120
 804976c:  83 45 ec 78           addl   $0x78,-0x14(%ebp)         //+120
 8049770:  81 45 ec 24 03 00 00  addl   $0x324,-0x14(%ebp)        //+804
 8049777:  83 6d ec 78           subl   $0x78,-0x14(%ebp)         //-120
 804977b:  81 45 ec 24 03 00 00  addl   $0x324,-0x14(%ebp)        //+804
 8049782:  81 6d ec 24 03 00 00  subl   $0x324,-0x14(%ebp)        //-804
 8049789:  83 45 ec 78           addl   $0x78,-0x14(%ebp)         //+120
 804978d:  90                    nop
 804978e:  8b 45 e4              mov      -0x1c(%ebp),%eax        //eax=-
0x1c(%ebp)
 8049791:  83 f8 2b              cmp                           $0x2b,%eax
//eax<=43
 8049794:  7f 16                 jg     80497ac <phase_3+0xb3>  //bomb
 8049796:  eb 0c                 jmp    80497a4 <phase_3+0xab>
 8049798:  e8 7a 08 00 00        call    804a017 <explode_bomb>  //jump2
here
 804979d:  b8 00 00 00 00        mov    $0x0,%eax
 80497a2:  eb 19                 jmp    80497bd <phase_3+0xc4>
 80497a4:  8b 45 e8              mov      -0x18(%ebp),%eax       //eax=-
0x18(%ebp)
 80497a7:  39 45 ec              cmp       %eax,-0x14(%ebp)  //eax  ==  -
0x14(%ebp)
 80497aa:  74 0c                 je     80497b8 <phase_3+0xbf>
 80497ac:  e8 66 08 00 00        call   804a017 <explode_bomb>
 80497b1:  b8 00 00 00 00        mov    $0x0,%eax
 80497b6:  eb 05                 jmp    80497bd <phase_3+0xc4>
 80497b8:  b8 01 00 00 00        mov    $0x1,%eax
 80497bd:  8b 55 f4              mov    -0xc(%ebp),%edx
 80497c0:  65 33 15 14 00 00 00  xor    %gs:0x14,%edx
 80497c7:  74 05                 je     80497ce <phase_3+0xd5>
 80497c9:  e8 c2 f9 ff ff        call   8049190 <__stack_chk_fail@plt>
 80497ce:  c9                    leave
 80497cf:  c3                    ret
```

首先 输入的数 小于等于 45

若输入 42 ，在 0x80497a7 处查看$ebp-0x14 可知第二个数为 0

43 …… 同理

Answer:

42 0

Phase_4

```
08049824 <phase_4>:
 8049824:   f3 0f 1e fb             endbr32
 8049828:   55                      push   %ebp
 8049829:   89 e5                   mov    %esp,%ebp
 804982b:   83 ec 38                sub    $0x38,%esp
 804982e:   8b 45 08                mov    0x8(%ebp),%eax
 8049831:   89 45 d4                mov    %eax,-0x2c(%ebp)
 8049834:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 804983a:   89 45 f4                mov    %eax,-0xc(%ebp)
 804983d:   31 c0                   xor    %eax,%eax
 804983f:   8d 45 e8                lea    -0x18(%ebp),%eax
 8049842:   50                      push   %eax
 8049843:   8d 45 e4                lea    -0x1c(%ebp),%eax
 8049846:   50                      push   %eax
 8049847:   68 11 b2 04 08          push   $0x804b211
 804984c:   ff 75 d4                pushl  -0x2c(%ebp)
 804984f:       e8   9c   f9   ff   ff                  call       80491f0
<__isoc99_sscanf@plt>
 8049854:   83 c4 10                add    $0x10,%esp
 8049857:   89 45 ec                mov    %eax,-0x14(%ebp)
 804985a:   83 7d ec 02             cmpl   $0x2,-0x14(%ebp)
 804985e:   75 08                   jne    8049868 <phase_4+0x44>
 8049860:   8b 45 e4                mov    -0x1c(%ebp),%eax
 8049863:      83   f8   07                          cmp       $0x7,%eax
//eax>7
 8049866:   7f 0c                   jg     8049874 <phase_4+0x50>
 8049868:   e8 aa 07 00 00          call   804a017 <explode_bomb>
 804986d:   b8 00 00 00 00          mov    $0x0,%eax
 8049872:   eb 2b                   jmp    804989f <phase_4+0x7b>
 8049874:   8b 45 e4                mov    -0x1c(%ebp),%eax
 8049877:   83 ec 0c                sub    $0xc,%esp
 804987a:   50                      push   %eax
 804987b:    e8 50 ff ff ff           call    80497d0 <func4>     //调用
func4
```

```
 8049880:   83 c4 10                add    $0x10,%esp
 8049883:   89 45 f0                mov    %eax,-0x10(%ebp)
 8049886:   8b 45 e8                mov    -0x18(%ebp),%eax
 8049889:   39 45 f0                cmp    %eax,-0x10(%ebp)
 804988c:   74 0c                   je     804989a <phase_4+0x76>
 804988e:   e8 84 07 00 00          call   804a017 <explode_bomb>
 8049893:   b8 00 00 00 00          mov    $0x0,%eax
 8049898:   eb 05                   jmp    804989f <phase_4+0x7b>
 804989a:   b8 01 00 00 00          mov    $0x1,%eax
 804989f:   8b 55 f4                mov    -0xc(%ebp),%edx
 80498a2:   65 33 15 14 00 00 00    xor    %gs:0x14,%edx
 80498a9:   74 05                   je     80498b0 <phase_4+0x8c>
 80498ab:      e8   e0   f8   ff   ff                  call      8049190
<__stack_chk_fail@plt>
 80498b0:   c9                      leave
 80498b1:   c3                      ret
```

观察可知

Phase_4 需要两个输入，并将这两个输入压栈递归调用 func4

这两个输入要求满足:

第一个输入大于 8 ，

第二个数等于 func(the first input)

Func4

```
080497d0 <func4>:
 80497d0:   f3 0f 1e fb             endbr32
 80497d4:   55                      push   %ebp
 80497d5:   89 e5                   mov    %esp,%ebp
 80497d7:   53                      push   %ebx
 80497d8:   83 ec 04                sub    $0x4,%esp
 80497db:   83 7d 08 00             cmpl   $0x0,0x8(%ebp)
 80497df:   7f 07                   jg     80497e8 <func4+0x18>
 80497e1:   b8 0c 00 00 00          mov    $0xc,%eax
 80497e6:   eb 37                   jmp    804981f <func4+0x4f>
 80497e8:   83 7d 08 01             cmpl   $0x1,0x8(%ebp)
 80497ec:   75 07                   jne    80497f5 <func4+0x25>
 80497ee:   b8 13 00 00 00          mov    $0x13,%eax
 80497f3:   eb 2a                   jmp    804981f <func4+0x4f>
 80497f5:   8b 45 08                mov    0x8(%ebp),%eax
 80497f8:   83 e8 01                sub    $0x1,%eax
 80497fb:   83 ec 0c                sub    $0xc,%esp
 80497fe:   50                      push   %eax
 80497ff:   e8 cc ff ff ff          call   80497d0 <func4>        //递归
```

```
 8049804:   83 c4 10              add    $0x10,%esp
 8049807:   89 c3                 mov    %eax,%ebx
 8049809:   8b 45 08              mov    0x8(%ebp),%eax
 804980c:   83 e8 02              sub    $0x2,%eax
 804980f:   83 ec 0c              sub    $0xc,%esp
 8049812:   50                    push   %eax
 8049813:   e8 b8 ff ff ff        call   80497d0 <func4>
 8049818:   83 c4 10              add    $0x10,%esp
 804981b:   d1 f8                 sar    %eax
 804981d:   01 d8                 add    %ebx,%eax
 804981f:   8b 5d fc              mov    -0x4(%ebp),%ebx
 8049822:   c9                    leave
 8049823:   c3                    ret
```

故 在 phase_4 中 0x8049880 处设置断点，

第一个数为 8 时 ，递归调用 func4 的结果为 0x8049880 处 eax 的值。

```
eax            0xa0               160
ecx            0x0                0
edx            0x0                0
ebx            0xffffcfb0         -12368
esp            0xffffcf20         0xffffcf20
ebp            0xffffcf68         0xffffcf68
esi            0xf7fb6000         -134520832
edi            0xf7fb6000         -134520832
eip            0x8049880          0x8049880 <phase_4+92>
eflags         0x216              [ PF AF IF ]
cs             0x23               35
```

即 第二个数为 160

第一个输入为 9…… 时同理。

Phase_5

```
080498b2 <phase_5>:
 80498b2:   f3 0f 1e fb           endbr32
 80498b6:   55                    push   %ebp
 80498b7:   89 e5                 mov    %esp,%ebp
 80498b9:   83 ec 38              sub    $0x38,%esp
 80498bc:   8b 45 08              mov    0x8(%ebp),%eax
 80498bf:   89 45 d4              mov    %eax,-0x2c(%ebp)
 80498c2:   65 a1 14 00 00 00     mov    %gs:0x14,%eax
 80498c8:   89 45 f4              mov    %eax,-0xc(%ebp)
 80498cb:   31 c0                 xor    %eax,%eax
 80498cd:   83 ec 0c              sub    $0xc,%esp
```

```
 80498d0:   ff 75 d4                pushl  -0x2c(%ebp)
 80498d3:   e8 8f 04 00 00          call   8049d67 <string_length>
 80498d8:   83 c4 10                add    $0x10,%esp
 80498db:   89 45 e8                mov    %eax,-0x18(%ebp)
 80498de:   83 7d e8 06             cmpl   $0x6,-0x18(%ebp)//字符串长度为
6
 80498e2:   74 0c                   je     80498f0 <phase_5+0x3e>
 80498e4:   e8 2e 07 00 00          call   804a017 <explode_bomb>
 80498e9:   b8 00 00 00 00          mov    $0x0,%eax
 80498ee:   eb 62                   jmp    8049952 <phase_5+0xa0>
 80498f0:   c7 45 e4 00 00 00 00    movl   $0x0,-0x1c(%ebp)
 80498f7:   eb 26                   jmp    804991f <phase_5+0x6d>
 80498f9:   8b 55 e4                mov    -0x1c(%ebp),%edx//loop begin
 80498fc:   8b 45 d4                mov    -0x2c(%ebp),%eax
 80498ff:   01 d0                   add    %edx,%eax
 8049901:   0f b6 00                movzbl (%eax),%eax
 8049904:   0f be c0                movsbl %al,%eax
 8049907:   83 e0 0f                  and    $0xf,%eax   //按位与（取低四
位）
 804990a:   0f b6 80 64 d2 04 08    movzbl 0x804d264(%eax),%eax//char 数
组
 8049911:   8d 4d ed                lea    -0x13(%ebp),%ecx
 8049914:   8b 55 e4                mov    -0x1c(%ebp),%edx
 8049917:   01 ca                   add    %ecx,%edx
 8049919:   88 02                   mov    %al,(%edx)
 804991b:   83 45 e4 01             addl   $0x1,-0x1c(%ebp)
 804991f:   83 7d e4 05             cmpl   $0x5,-0x1c(%ebp)
 8049923:   7e d4                   jle    80498f9 <phase_5+0x47>
 8049925:   c6 45 f3 00             movb   $0x0,-0xd(%ebp)
 8049929:   83 ec 08                sub    $0x8,%esp
 804992c:   68 3c b2 04 08          push   $0x804b23c
 8049931:   8d 45 ed                lea    -0x13(%ebp),%eax
 8049934:   50                      push   %eax
 8049935:   e8 5d 04 00 00          call   8049d97 <strings_not_equal>
 804993a:   83 c4 10                add    $0x10,%esp
 804993d:   85 c0                   test   %eax,%eax
 804993f:   74 0c                   je     804994d <phase_5+0x9b>
 8049941:   e8 d1 06 00 00          call   804a017 <explode_bomb>
 8049946:   b8 00 00 00 00          mov    $0x0,%eax
 804994b:   eb 05                   jmp    8049952 <phase_5+0xa0>
 804994d:   b8 01 00 00 00          mov    $0x1,%eax
 8049952:   8b 4d f4                mov    -0xc(%ebp),%ecx
 8049955:   65 33 0d 14 00 00 00    xor    %gs:0x14,%ecx
 804995c:   74 05                   je     8049963 <phase_5+0xb1>
```

```
 804995e:       e8   2d   f8   ff   ff                   call       8049190
<__stack_chk_fail@plt>
 8049963:   c9                        leave
 8049964:   c3                        ret
```

Decompile：

```
for ( i = 0; i <= 5; ++i )
{
      v3[i] = CharArrayAt_0x804d264 [(int) (i + a2) & 0xF];
}
```

Phase_5 是要求输入一个长度为 6 的字符串

对于其中每个字符而言，依次将其 ASCII 二进制低四位（转换为 10 进制后）作为在 0x804d264 处数组的索引，获取字符，最终调用 string_not_equal 判断获取的字符串与 0x804b23c 处的字符是否相等。

```
(gdb) x/s 0x804b23c
0x804b23c:      "daemlo"
```

```
(gdb) x/s 0x804d264
0x804d264 <array.2708>: "boefknpmiacjldhg"
```

ASCII可显示字符

二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形
0010 0000	32	20	(空格) (SP)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x
0011 1001	57	39	9	0101 1001	89	59	Y	0111 1001	121	79	y
0011 1010	58	3A	:	0101 1010	90	5A	Z	0111 1010	122	7A	z

查询 ASCII 表可知答案为 MIBGLQ

Phase_6

```
08049965 <phase_6>:
 8049965:  f3 0f 1e fb           endbr32
 8049969:  55                    push   %ebp
 804996a:  89 e5                 mov    %esp,%ebp
 804996c:  83 ec 68              sub    $0x68,%esp
 804996f:  8b 45 08              mov    0x8(%ebp),%eax
 8049972:  89 45 a4              mov    %eax,-0x5c(%ebp)
 8049975:  65 a1 14 00 00 00     mov    %gs:0x14,%eax
 804997b:  89 45 f4              mov    %eax,-0xc(%ebp)
 804997e:  31 c0                 xor    %eax,%eax
 8049980:  c7 45 b8 a4 d1 04 08  movl   $1,-0x48(%ebp)
 8049987:  83 ec 04              sub    $0x4,%esp
 804998a:  6a 07                 push   $0x7
 804998c:  8d 45 bc              lea    -0x44(%ebp),%eax
 804998f:  50                    push   %eax
 8049990:  ff 75 a4              pushl  -0x5c(%ebp)
 8049993:  e8 3d 03 00 00        call 8049cd5 <read_n_numbers>
 8049998:  83 c4 10              add    $0x10,%esp
 804999b:  85 c0                 test   %eax,%eax
 804999d:  75 0a                 jne    80499a9 <phase_6+0x44>
 804999f:  b8 00 00 00 00        mov    $0x0,%eax
 80499a4:  e9 37 01 00 00        jmp    8049ae0 <phase_6+0x17b>
 80499a9:  c7 45 b0 00 00 00 00  movl   $0x0,-0x50(%ebp)
 80499b0:  eb 60                 jmp    8049a12 <phase_6+0xad>
 80499b2:  8b 45 b0              mov    -0x50(%ebp),%eax
 80499b5:  8b 44 85 bc           mov    -0x44(%ebp,%eax,4),%eax
 80499b9:  85 c0                 test   %eax,%eax
 80499bb:  7e 0c                 jle    80499c9 <phase_6+0x64>
 80499bd:  8b 45 b0              mov    -0x50(%ebp),%eax
 80499c0:  8b 44 85 bc           mov    -0x44(%ebp,%eax,4),%eax
 80499c4:  83 f8 07              cmp    $0x7,%eax
 80499c7:  7e 0f                 jle    80499d8 <phase_6+0x73>
 80499c9:  e8 49 06 00 00        call   804a017 <explode_bomb>
```

```
80499ce:  b8 00 00 00 00        mov    $0x0,%eax
80499d3:  e9 08 01 00 00        jmp    8049ae0 <phase_6+0x17b>
80499d8:  8b 45 b0              mov    -0x50(%ebp),%eax
80499db:  83 c0 01              add    $0x1,%eax
80499de:  89 45 b4              mov    %eax,-0x4c(%ebp)
80499e1:  eb 25                 jmp    8049a08 <phase_6+0xa3>
80499e3:  8b 45 b0              mov    -0x50(%ebp),%eax
80499e6:  8b 54 85 bc           mov    -0x44(%ebp,%eax,4),%edx
80499ea:  8b 45 b4              mov    -0x4c(%ebp),%eax
80499ed:  8b 44 85 bc           mov    -0x44(%ebp,%eax,4),%eax
80499f1:  39 c2                 cmp    %eax,%edx
80499f3:  75 0f                 jne    8049a04 <phase_6+0x9f>
80499f5:  e8 1d 06 00 00        call   804a017 <explode_bomb>


80499fa:  b8 00 00 00 00        mov    $0x0,%eax
80499ff:  e9 dc 00 00 00        jmp    8049ae0 <phase_6+0x17b>


//end



8049a04:  83 45 b4 01           addl   $0x1,-0x4c(%ebp)
8049a08:  83 7d b4 06           cmpl   $0x6,-0x4c(%ebp)
8049a0c:  7e d5                 jle    80499e3 <phase_6+0x7e>
8049a0e:  83 45 b0 01           addl   $0x1,-0x50(%ebp)
8049a12:  83 7d b0 06           cmpl   $0x6,-0x50(%ebp)
8049a16:  7e 9a                 jle    80499b2 <phase_6+0x4d>
8049a18:  c7 45 b0 00 00 00 00  movl   $0x0,-0x50(%ebp)
8049a1f:  eb 36                 jmp    8049a57 <phase_6+0xf2>
8049a21:  8b 45 b8              mov    -0x48(%ebp),%eax
8049a24:  89 45 ac              mov    %eax,-0x54(%ebp)
8049a27:  c7 45 b4 01 00 00 00  movl   $0x1,-0x4c(%ebp)
8049a2e:  eb 0d                 jmp    8049a3d <phase_6+0xd8>
8049a30:  8b 45 ac              mov    -0x54(%ebp),%eax
8049a33:  8b 40 08              mov    0x8(%eax),%eax
8049a36:  89 45 ac              mov    %eax,-0x54(%ebp)
8049a39:  83 45 b4 01           addl   $0x1,-0x4c(%ebp)
8049a3d:  8b 45 b0              mov    -0x50(%ebp),%eax
8049a40:  8b 44 85 bc           mov    -0x44(%ebp,%eax,4),%eax
8049a44:  39 45 b4              cmp    %eax,-0x4c(%ebp)
```

```
8049a47:  7c e7                jl     8049a30 <phase_6+0xcb>
8049a49:  8b 45 b0             mov    -0x50(%ebp),%eax
8049a4c:  8b 55 ac             mov    -0x54(%ebp),%edx
8049a4f:  89 54 85 d8          mov    %edx,-0x28(%ebp,%eax,4)
8049a53:  83 45 b0 01          addl   $0x1,-0x50(%ebp)
8049a57:  83 7d b0 06          cmpl   $0x6,-0x50(%ebp)
8049a5b:  7e c4                jle    8049a21 <phase_6+0xbc>
8049a5d:  8b 45 d8             mov    -0x28(%ebp),%eax
8049a60:  89 45 b8             mov    %eax,-0x48(%ebp)
8049a63:  8b 45 b8             mov    -0x48(%ebp),%eax
8049a66:  89 45 ac             mov    %eax,-0x54(%ebp)
8049a69:  c7 45 b0 01 00 00 00 movl   $0x1,-0x50(%ebp)
8049a70:  eb 1a                jmp    8049a8c <phase_6+0x127>
8049a72:  8b 45 b0             mov    -0x50(%ebp),%eax
8049a75:  8b 54 85 d8          mov    -0x28(%ebp,%eax,4),%edx
8049a79:  8b 45 ac             mov    -0x54(%ebp),%eax
8049a7c:  89 50 08             mov    %edx,0x8(%eax)
8049a7f:  8b 45 ac             mov    -0x54(%ebp),%eax
8049a82:  8b 40 08             mov    0x8(%eax),%eax
8049a85:  89 45 ac             mov    %eax,-0x54(%ebp)
8049a88:  83 45 b0 01          addl   $0x1,-0x50(%ebp)
8049a8c:  83 7d b0 06          cmpl   $0x6,-0x50(%ebp)
8049a90:  7e e0                jle    8049a72 <phase_6+0x10d>
8049a92:  8b 45 ac             mov    -0x54(%ebp),%eax
8049a95:  c7 40 08 00 00 00 00 movl   $0x0,0x8(%eax)
8049a9c:  8b 45 b8             mov    -0x48(%ebp),%eax
8049a9f:  89 45 ac             mov    %eax,-0x54(%ebp)
8049aa2:  c7 45 b0 00 00 00 00 movl   $0x0,-0x50(%ebp)
8049aa9:  eb 2a                jmp    8049ad5 <phase_6+0x170>
8049aab:  8b 45 ac             mov    -0x54(%ebp),%eax
8049aae:  8b 10                mov    (%eax),%edx
8049ab0:  8b 45 ac             mov    -0x54(%ebp),%eax
8049ab3:  8b 40 08             mov    0x8(%eax),%eax
8049ab6:  8b 00                mov    (%eax),%eax
8049ab8:  39 c2                cmp    %eax,%edx
8049aba:  7e 0c                jle    8049ac8 <phase_6+0x163>
8049abc:  e8 56 05 00 00       call   804a017 <explode_bomb>


8049ac1:  b8 00 00 00 00       mov    $0x0,%eax
8049ac6:  eb 18                jmp    8049ae0 <phase_6+0x17b>
8049ac8:  8b 45 ac             mov    -0x54(%ebp),%eax
8049acb:  8b 40 08             mov    0x8(%eax),%eax
8049ace:  89 45 ac             mov    %eax,-0x54(%ebp)
```

```
8049ad1:  83 45 b0 01           addl   $0x1,-0x50(%ebp)
8049ad5:  83 7d b0 05           cmpl   $0x5,-0x50(%ebp)
8049ad9:  7e d0                 jle    8049aab <phase_6+0x146>
8049adb:  b8 01 00 00 00        mov    $0x1,%eax
8049ae0:  8b 4d f4              mov    -0xc(%ebp),%ecx
8049ae3:  65 33 0d 14 00 00 00  xor    %gs:0x14,%ecx
8049aea:  74 05                 je     8049af1 <phase_6+0x18c>
8049aec:  e8 9f f6 ff ff        call   8049190 <__stack_chk_fail@plt>
8049af1:  c9                    leave
8049af2:  c3                    ret
```

Phase_6 要求输入 7 个互不相等的数字，通过这串数字调整链表的顺序

最终要使得链表为升序（大于等于）

通过 gdb 查看初始链表

7 1 9 6 4 3 2

要求所得为

9 6 7 4 3 2 1

```
for ( k = 0; k <= 6; ++k )
{
  v2 = (int)&node1;
  for ( l = 1; l < *(&v15 + k - 17); ++l )
    v2 = *(_DWORD *)(v2 + 8);
  *(&v15 + k - 10) = v2;
}
v11 = v13;
v3 = v13;
for ( m = 1; m <= 6; ++m )
{
  *(_DWORD *)(v3 + 8) = *(&v15 + m - 10);
  v3 = *(_DWORD *)(v3 + 8);
}
*(_DWORD *)(v3 + 8) = 0;
v4 = v11;
for ( n = 0; n <= 5; ++n )
{
  if ( *(_DWORD *)v4 > **(_DWORD **)(v4 + 8) )
  {
    explode_bomb();
    break;
  }
  v4 = *(_DWORD *)(v4 + 8);
}
```

输入 2 7 6 5 4 1 3 调整链表

Secret_phase

直接在 phase_defused 中查看明码 0x804b34b

得 vmGMwSLa

在 phase_4 后输入即可进入

```
08049b5a <secret_phase>:
 8049b5a:  f3 0f 1e fb           endbr32
 8049b5e:  55                    push   %ebp
 8049b5f:  89 e5                 mov    %esp,%ebp
 8049b61:  83 ec 18              sub    $0x18,%esp
 8049b64:  e8 67 03 00 00        call   8049ed0 <read_line>
 8049b69:  89 45 ec              mov    %eax,-0x14(%ebp)
 8049b6c:  83 ec 0c              sub    $0xc,%esp
 8049b6f:  ff 75 ec              pushl  -0x14(%ebp)
 8049b72:  e8 a9 f6 ff ff        call   8049220 <atoi@plt>
 8049b77:  83 c4 10              add    $0x10,%esp
 8049b7a:  89 45 f0              mov    %eax,-0x10(%ebp)
 8049b7d:  83 7d f0 00           cmpl   $0x0,-0x10(%ebp)
 8049b81:  7e 09                 jle    8049b8c <secret_phase+0x32>
 8049b83:  81 7d f0 e9 03 00 00  cmpl   $0x3e9,-0x10(%ebp)
 8049b8a:  7e 0c                 jle    8049b98 <secret_phase+0x3e>
 8049b8c:  e8 86 04 00 00        call   804a017 <explode_bomb>
 8049b91:  b8 00 00 00 00        mov    $0x0,%eax
 8049b96:  eb 42                 jmp    8049bda <secret_phase+0x80>
 8049b98:  83 ec 08              sub    $0x8,%esp
 8049b9b:  ff 75 f0              pushl  -0x10(%ebp)
 8049b9e:  68 58 d2 04 08        push   $0x804d258
 8049ba3:  e8 4b ff ff ff        call   8049af3 <fun7>
 8049ba8:  83 c4 10              add    $0x10,%esp
 8049bab:  89 45 f4              mov    %eax,-0xc(%ebp)
 8049bae:  83 7d f4 01           cmpl   $0x1,-0xc(%ebp)
 8049bb2:  74 0c                 je     8049bc0 <secret_phase+0x66>
 8049bb4:  e8 5e 04 00 00        call   804a017 <explode_bomb>
 8049bb9:  b8 00 00 00 00        mov    $0x0,%eax
 8049bbe:  eb 1a                 jmp    8049bda <secret_phase+0x80>
```

```
 8049bc0:  83 ec 0c              sub    $0xc,%esp
 8049bc3:  68 44 b2 04 08        push   $0x804b244
 8049bc8:  e8 e3 f5 ff ff        call   80491b0 <puts@plt>
 8049bcd:  83 c4 10              add    $0x10,%esp
 8049bd0:  e8 6f 04 00 00        call   804a044 <phase_defused>
 8049bd5:  b8 01 00 00 00        mov    $0x1,%eax
 8049bda:  c9                    leave
 8049bdb:  c3                    ret
```

显然 secret_phase 调用了 func7

观察 func7

```
08049af3 <fun7>:
 8049af3:  f3 0f 1e fb          endbr32
 8049af7:  55                    push   %ebp
 8049af8:  89 e5                 mov    %esp,%ebp
 8049afa:  83 ec 08              sub    $0x8,%esp
 8049afd:  83 7d 08 00           cmpl   $0x0,0x8(%ebp)
 8049b01:  75 07                 jne    8049b0a <fun7+0x17>
 8049b03:  b8 ff ff ff ff        mov    $0xffffffff,%eax
 8049b08:  eb 4e                 jmp    8049b58 <fun7+0x65>
 8049b0a:  8b 45 08              mov    0x8(%ebp),%eax
 8049b0d:  8b 00                 mov    (%eax),%eax
 8049b0f:  39 45 0c              cmp    %eax,0xc(%ebp)
 8049b12:  7d 19                 jge    8049b2d <fun7+0x3a>
 8049b14:  8b 45 08              mov    0x8(%ebp),%eax
 8049b17:  8b 40 04              mov    0x4(%eax),%eax
 8049b1a:  83 ec 08              sub    $0x8,%esp
 8049b1d:  ff 75 0c              pushl  0xc(%ebp)
 8049b20:  50                    push   %eax
 8049b21:  e8 cd ff ff ff        call   8049af3 <fun7>
 8049b26:  83 c4 10              add    $0x10,%esp
 8049b29:  01 c0                 add    %eax,%eax
 8049b2b:  eb 2b                 jmp    8049b58 <fun7+0x65>
 8049b2d:  8b 45 08              mov    0x8(%ebp),%eax
 8049b30:  8b 00                 mov    (%eax),%eax
 8049b32:  39 45 0c              cmp    %eax,0xc(%ebp)
 8049b35:  75 07                 jne    8049b3e <fun7+0x4b>
 8049b37:  b8 00 00 00 00        mov    $0x0,%eax
 8049b3c:  eb 1a                 jmp    8049b58 <fun7+0x65>
 8049b3e:  8b 45 08              mov    0x8(%ebp),%eax
 8049b41:  8b 40 08              mov    0x8(%eax),%eax
 8049b44:  83 ec 08              sub    $0x8,%esp
 8049b47:  ff 75 0c              pushl  0xc(%ebp)
 8049b4a:  50                    push   %eax
```

```
8049b4b:  e8 a3 ff ff ff        call   8049af3 <fun7>
8049b50:  83 c4 10              add    $0x10,%esp
8049b53:  01 c0                 add    %eax,%eax
8049b55:  83 c0 01              add    $0x1,%eax
8049b58:  c9                    leave
8049b59:  c3                    ret
```

Fun7 中存在递归

Decompile:

Secret_phase

```
Key code:
int *arr;
if ( input > 0 && input <= 1001 )
  {
    if ( fun7(arr, input) == 1 )
    {
       printf("Wow! You've defused the secret stage!");
       //…… phase_defuased
       result = 1;
    }
    else
    {
       explode_bomb();
       result = 0;
    }
  }
  else
  {
       explode_bomb();
       result = 0;
  }

ruturn result;
```

func7

```
Key Code
 if ( a1 )
  {
    if ( a2 >= *(_DWORD *)a1 )
    {
```

```
    if ( a2 == *(_DWORD *)a1 )
      result = 0;
    else
      result = 2 * fun7(*(_DWORD *)(a1 + 8), a2) + 1;
  }
  else
  {
    result = 2 * fun7(*(_DWORD *)(a1 + 4), a2);
  }
}
else
{
  result = -1;
}
return result;
```

注意到要求 result 为 1

只要进入 a2>*(_DWORD *)a1 的状态后 a2==*(_DWORD *)(a1 + 8)即可。

即 40

总结

学习了一部分汇编、了解了 IDA 反汇编，了解了数据结构，复习了 Linux 指令，了解了 GDB 调试，了解了计算机体系结构