

Computer Architecture

Bomb Lab Report 2016 Spring

Table of Content

1	introduction	2
	1.1 Overview	
	1.2 Preparation	2
2	phase_1	3
3	phase_2	5
4	phase_3	7
5	phase_4	12
6	phase_5	15
7	phase_6	18
8	secret_phase8.1 find the secret phase8.2 solve the secret phase	
9	conclusion	27

1 introduction

1.1 Overview

A "binary bombis a program provided to students as an object code file. When run, it prompts the user to type in 6 different strings. If any of these is incorrect, the bomb "explodes," printing an error message and logging the event on a grading server. Students must "defuse" their own unique bomb by disassembling and reverse engineering the program to determine what the 6 strings should be. The lab teaches students to understand assembly language, and also forces them to learn how to use a debugger. It's also great fun. A legendary lab among the CMU undergrads.

1.2 Preparation

在拆除炸弹之前,首先进行一些前期准备:

- 通过 putty 登陆服务器,发现我本次 Lab 的实验文件:bomb51.tar
- 执行 tar xvf bomb51.tar 解压文件,得到四个文件: bomb bomb.c ID README
- 其中 ID README 分别是学生的编号和这次的说明文档。

于是查看 bomb.c, 发现其头文件声明如下:

```
#include <stdio.h>
#include "support.h"
#include "phases.h"
```

phase.h 包含了这次炸弹的全部关卡,而它并没有在 Lab 中给出。因此我们只能从可执行文件 bomb 下手

- 执行 objdump -d bomb > bomb.txt , 反编译可执行文件 , 将汇编代码输出到 bomb.txt
- 反编译之后的代码非常长,不过在仔细研究之后,发现其中有六个函数 < phase_1..6 > ,分别对应六个 关卡。因此,这次 Lab 的关键就是破解这六个关卡对应的汇编代码,分析这六个函数的功能。

• 第一关的汇编代码如下:

```
1 08048c10 <phase_1>:
     8048c10:
                                                             $0 \times 1c, \% esp
                       83 ec 1c
                                                     sub
2
     8048c13:
                       c7 44 24 04 ec 99 04
                                                             $0 \times 80499 \text{ ec}, 0 \times 4(\% \text{ esp})
                                                     movl
3
    8048c1a:
                       08
                       8b 44 24 20
    8048c1b:
                                                             0 \times 20(\% esp), \% eax
                                                    mov
5
    8048c1f:
                       89 04 24
                                                             %eax,(%esp)
                                                    mov
     8048c22:
                       e8 1d 04 00 00
                                                             8049044 <strings_not_equal>
                                                     call
7
     8048c27:
                                                             %eax,%eax
                       85 c0
                                                     test
8
    8048c29:
                       74 05
                                                             8048c30 < phase_1 + 0x20 >
9
                                                    jе
    8048c2b:
                       e8 a3 09 00 00
                                                     call
                                                             80495d3 <explode_bomb>
10
    8048c30:
                       83 c4 1c
                                                             $0 \times 1c, \% esp
                                                     add
     8048c33:
12
                       с3
                                                     ret
```

• 这个函数较短,分析发现,它调用了一个名为 strings_not_equal 函数。

分析 strings_not_equal 这个函数,发现它实现这样一个功能:

比较两个字符串,相等返回0,不相等返回1

- 分析 call 之后的程序,发现如果返回值(即 %eax)不为 0 的话,会调用 explode_bomb 函数,炸弹会被引爆。因此,我们输入的字符串应该与某个字符串相同。
- 这样,破解 phase_1 的方法就很清晰了:

输入存在地址 0x80499ec 中的字符串

• 于是现在的问题是找出存放在地址 0x80499ec 中的字符串。

从调用 strings_not_equal 的语句向上看:

```
1 8048c13: c7 44 24 04 ec 99 04 movl $0x80499ec,0x4(%esp)
2 8048c1a: 08
3 8048c1b: 8b 44 24 20 mov 0x20(%esp),%eax
4 8048c1f: 89 04 24 mov %eax,(%esp)
```

这表示,这个函数的两个参数,一个是程序自身地址 0x80499ec,另一个 0x20(%esp) 就是我们的输入参数。

● 我们首先需要知道 0x80499ec 地址里存放的数据。于是使用 gdb , 输入命令 p (char *) 0x80499ec, 查看它存放的数据:

```
(gdb) p (char *) 0x80499ec
$1 = 0x80499ec "I turned the moon into something I like to call a Death Star."
```

● 第一关的答案水落石出:输入 I turned the moon into something I like to call a Death Star. , 顺利过 关。

Welcome to my fiendish little bomb. You have 6 phases with which to blow yourself up. Have a nice day!
I turned the moon into something I like to call a Death Star.
Phase 1 defused. How about the next one?

主要复习知识点:常量的存储、函数的参数传递

• 第二关的汇编代码如下:

```
1 08048c34 <phase_2>:
    8048c34:
                       53
                                                            %ebx
                                                    push
2
                                                            $0x38,%esp
    8048c35:
                       83 ec 38
                                                    sub
3
    8048c38:
                       8d 44 24 18
                                                            0 \times 18(\% esp), \% eax
4
                                                    lea
    8048c3c:
                       89 44 24 04
                                                            \%eax, 0 \times 4(\%esp)
                                                    mov
5
                                                            0\times40(\%esp),\%eax
    8048c40:
                       8b 44 24 40
                                                    mov
    8048c44:
                       89 04 24
                                                            %eax,(%esp)
                                                    mov
7
                       e8 8c 0a 00 00
                                                            80496d8 < read_six_numbers >
    8048c47:
                                                    call
8
    8048c4c:
                       83 7c 24 18 01
                                                    cmpl
                                                            $0 \times 1, 0 \times 18(\% esp)
9
    8048c51:
                       74 05
                                                    jе
                                                            8048c58 <phase_2+0x24>
10
                                                            80495d3 <explode bomb>
    8048c53:
                       e8 7b 09 00 00
11
                                                    call
    8048c58:
                       bb 01 00 00 00
                                                            $0\times1,\% ebx
12
                                                    mov
    8048c5d:
                                                            %ebx,%eax
                       89 d8
                                                    mov
13
    8048c5f:
                       83 c3 01
                                                    add
                                                            $0\times1.\% ebx
14
                                                            %ebx.%edx
15
    8048c62:
                       89 da
                                                    mov
                                                            0 \times 14(\% esp, \% eax, 4), \% edx
    8048c64:
                       Of af 54 84 14
16
                                                    imul
    8048c69:
                       39 54 84 18
                                                            \%edx, 0 \times 18(\%esp, \%eax, 4)
                                                    cmp
17
    8048c6d:
                                                            8048c74 <phase_2+0x40>
                       74 05
                                                    jе
18
19
    8048c6f:
                       e8 5f 09 00 00
                                                    call
                                                            80495d3 <explode bomb>
    8048c74:
                       83 fb 06
                                                            $0\times6,\% ebx
20
                                                    cmp
    8048c77:
                                                            8048c5d < phase_2 + 0x29 >
                       75 e4
21
                                                    jne
    8048c79:
                       83 c4 38
                                                    add
                                                            $0x38,%esp
22
    8048c7c:
                                                            %ebx
23
                       5b
                                                    pop
    8048c7d:
                       с3
24
                                                    ret
```

- 我们首先发现,它调用了一个 read_six_numbers 函数。经过分析,这个函数从输入中读入六个整数,并按地址从低到高存放在 0x18(%esp) (0x30%esp) 这 24 个字节中。也就是说,a[0] 在 0x18(%esp),a[1] 在 0x1c(%esp),以此类推。
- 然后观察下面一段代码:

```
      1
      8048 c4c:
      83 7c 24 18 01
      cmpl $0x1,0x18(%esp)

      2
      8048 c51:
      74 05
      je 8048 c58 < phase_2+0x24>

      3
      8048 c53:
      e8 7b 09 00 00
      call 80495d3 < explode_bomb>
```

- 这段代码将 0x18(%esp) 即 a[0] 与 1 进行比较,如果不等则炸弹爆炸。这一段说明,我们输入的第一个数必须为 1。
- 接下来是一个循环:

```
bb 01 00 00 00
                                                             $0 \times 1, \% ebx
   8048c58:
1
                                                    mov
                                                            %ebx.%eax
                       89 d8
    8048c5d:
                                                    mov
2
    8048c5f:
                       83 c3 01
                                                             $0\times1,\% ebx
                                                    add
3
                                                            \%ebx,\%edx
    8048c62:
                       89 da
                                                    mov
4
                                                             0 \times 14(\% esp,\% eax,4),\% edx
    8048c64:
                       Of af 54 84 14
5
                                                    imul
    8048c69:
                       39 54 84 18
                                                            \%edx, 0 \times 18(\%esp, \%eax, 4)
6
                                                    cmp
                                                             8048c74 <phase_2+0x40>
    8048c6d:
                       74 05
                                                    įе
7
    8048c6f:
                       e8 5f 09 00 00
                                                    call
                                                             80495d3 <explode_bomb>
8
    8048c74:
                                                             $0\times6,\% ebx
                       83 fb 06
9
                                                    cmp
                                                             8048c5d <phase_2+0x29>
    8048c77:
                       75 e4
                                                    jne
10
```

这个循环的循环变量是 %ebx,从 2 循环到 5。同时, %eax 始终为 %ebx-1 中间的判断语句,可以发现它是将 0x14(%esp, %eax, 4) * %edx 相乘,并与 0x18(%esp, %eax, 4) 比较。由于 a 数组的起始位置为 0x18(%esp),这两个地址就是 a[_eax - 1] 和 a[_eax]。

因此,其对应的 c 语言代码如下:

```
void phase_2() {
1
            if(num[0] != 1) explode_bomb();
2
            int b = 1;
3
            while (b \le 5) {
4
                     a = b + 1;
5
                     if(num[b-1] * a != num[b])
                              explode_bomb();
7
                     b++;
8
9
            }
10
```

- 这个函数表明,我们输入的数组 a 要满足以下条件:
 - 数组长度为 6
 - a[0] = 1
 - a[i] = a[i 1] * (i + 1)
- 所以, 第二关的答案就水落石出了:12624120720

```
I turned the moon into something I like to call a Death Star.
Phase 1 defused. How about the next one?
1 2 6 24 120 720
That's number 2. Keep going!
```

考察知识点:数组的存储

• 第三关的汇编代码如下:

```
1 08048c7e <phase_3>:
     8048c7e:
                       83 ec 2c
                                                      sub
                                                              $0 \times 2c, \% esp
2
     8048c81:
                       8d 44 24 1c
                                                      lea
                                                              0 \times 1 c(\% esp), \% eax
3
     8048c85:
                       89 44 24 0c
                                                              \%eax, 0 \times c(\%esp)
4
                                                      mov
     8048c89:
                       8d 44 24 18
                                                      lea
                                                              0 \times 18(\% esp), \% eax
5
     8048c8d:
                       89 44 24 08
                                                              \%eax, 0 \times 8(\%esp)
                                                      mov
     8048c91:
                       c7 44 24 04 c6 9d 04
                                                              $0 \times 8049 dc6, 0 \times 4(\% esp)
                                                      movl
7
     8048c98:
                       80
8
     8048c99:
                       8b 44 24 30
                                                      mov
                                                              0\times30(\%esp),\%eax
9
     8048c9d:
                       89 04 24
                                                              %eax,(%esp)
10
                                                      mov
                                                              8048910 < \_isoc99\_sscanf@plt>
     8048ca0:
                       e8 6b fc ff ff
11
                                                      call
     8048ca5:
                       83 f8 01
                                                              $0\times1,\% eax
12
                                                     cmp
     8048ca8:
                       7f 05
                                                              8048caf <phase_3+0x31>
13
                                                      jg
     8048 caa:
                       e8 24 09 00 00
                                                      call
                                                              80495d3 <explode_bomb>
14
15
     8048 caf:
                       83 7c 24 18 07
                                                      cmpl
                                                              $0 \times 7, 0 \times 18(\% esp)
     8048cb4:
                                                              8048d1a <phase_3+0x9c>
                       77 64
16
                                                      ja
     8048cb6:
                       8b 44 24 18
                                                              0 \times 18(\% esp), \% eax
17
                                                      mov
     8048 cba:
                        ff 24 85 5c 9a 04 08
                                                              *0 \times 8049 a 5 c (, \% eax, 4)
                                                     jmp
18
19
     8048cc1:
                       b8 00 00 00 00
                                                      mov
                                                              $0 \times 0, \% eax
     8048cc6:
                       eb 05
                                                              8048 ccd < phase_3 + 0x4f >
20
                                                     jmp
                                                              $0x18e,%eax
     8048cc8:
                       b8 8e 01 00 00
21
                                                     mov
     8048 ccd:
                                                              $0×166.%eax
                       2d 66 01 00 00
                                                      sub
     8048cd2:
                       eb 05
                                                              8048cd9 <phase_3+0x5b>
23
                                                     jmp
     8048cd4:
                       b8 00 00 00 00
                                                      mov
                                                              $0 \times 0, \% eax
24
     8048cd9:
                       05 92 00 00 00
                                                              $0x92,%eax
25
                                                      add
                                                              8048ce5 <phase_3+0x67>
     8048 cde:
26
                       eb 05
                                                     jmp
     8048ce0:
                                                              $0 \times 0.\% eax
                       Ь8 00 00 00 00
                                                      mov
27
                                                              $0 \times e9, \% eax
     8048ce5:
                       2d e9 00 00 00
                                                      sub
28
                                                              8048cf1 <phase_3+0x73>
29
     8048 cea:
                       eb 05
                                                     jmp
     8048 cec:
                       b8 00 00 00 00
                                                              $0 \times 0.\% eax
                                                     mov
30
                                                              $0x128,%eax
     8048 cf1:
                       05 28 01 00 00
                                                      add
31
     8048 cf6:
                                                              8048cfd <phase_3+0x7f>
                       eb 05
                                                     imp
32
     8048 cf8:
                       b8 00 00 00 00
                                                              $0\times0,\% eax
33
                                                     mov
     8048 cfd:
                       2d df 01 00 00
                                                      sub
                                                              $0 \times 1 df, \% eax
34
                                                              8048d09 <phase_3+0x8b>
     8048d02:
                       eb 05
                                                     jmp
35
                                                              $0\times0,\%eax
     8048d04:
                       b8 00 00 00 00
                                                     mov
36
                                                              $0 \times 1 df, \% eax
37
     8048d09:
                       05 df 01 00 00
                                                      add
```

```
8048d15 <phase_3+0x97>
    8048d0e:
                      eb 05
38
                                                  jmp
                                                          $0 \times 0.\% eax
    8048d10:
                      Ь8 00 00 00 00
                                                  mov
39
    8048d15:
                      83 c0 80
                                                          $0xffffff80,%eax
                                                  add
40
    8048d18:
                      eb 0a
                                                          8048d24 <phase_3+0xa6>
41
                                                  jmp
                                                          80495d3 <explode bomb>
    8048d1a:
                      e8 b4 08 00 00
                                                  call
42
    8048 d1f:
                      b8 00 00 00 00
                                                          $0 \times 0, \% eax
43
                                                  mov
    8048d24:
                      83 7c 24 18 05
                                                          $0x5,0x18(\%esp)
                                                  cmpl
                                                          8048d31 <phase_3+0xb3>
    8048d29:
                      7f 06
45
                                                  jg
                      3b 44 24 1c
    8048d2b:
                                                  cmp
                                                          0 \times 1 c(\% esp), \% eax
46
                                                          8048d36 <phase_3+0xb8>
    8048d2f:
                      74 05
47
                                                  jе
    8048d31:
                      e8 9d 08 00 00
                                                          80495d3 <explode_bomb>
                                                  call
48
    8048d36:
                      83 c4 2c
                                                          $0 \times 2c, \% esp
49
                                                  add
    8048d39:
                      с3
50
                                                  ret
```

• 首先看到它调用了 sscanf 函数,参数存放在 0x8049dc6 地址中。

于是首先用 gdb 查看 0x8049dc6 地址中的参数:

```
(gdb) p (char *) 0x8049dc6
$3 = 0x8049dc6 "%d %d"
```

"%d%d"代表读入了两个整数。调用 sscanf 结束后,将 sscanf 的返回值与 1 比较,若不大于 1 则炸弹爆炸。

由此可以得到我们这关的目标是输入两个整数 a b。

• 输入的两个整数 a b 存放在 0x18(%esp) 和 0x1c(%esp) 中。接着往下看程序:

```
1 8048 caf: 83 7c 24 18 07 cmpl $0x7,0x18(%esp)
2 8048 cb4: 77 64 ja 8048 d1a <phase_3+0x9c>
```

这里可以发现,程序将 0x18(%esp),即输入的第一个整数 a 与 7 进行比较,如果大于 7 则炸弹爆炸。同时,由于使用的是 ja 命令,即 a 为无符号数,a 应该不小于 0

这样我们可以得出第一个限制条件:输入的第一个整数 $0 \le a \le 7$

继续往下看,注意到这一段:

```
1 8048cb6: 8b 44 24 18 mov 0x18(%esp),%eax
2 8048cba: ff 24 85 5c 9a 04 08 jmp *0x8049a5c(,%eax,4)
```

• 这段具有很明显的 switch 语句的特征,而跳转表存放在 0x8049a5c 中。

于是调用 gdb , 先查看对应的跳转表:

```
(gdb) p /x *0x8049a5c
$4 = 0x8048cc8
(gdb) p /x *(0x8049a5c + 4)
$5 = 0x8048cc1
(gdb) p /x *(0x8049a5c + 8)
$6 = 0x8048cd4
(gdb) p /x *(0x8049a5c + 12)
$7 = 0x8048ce0
(gdb) p /x *(0x8049a5c + 16)
$8 = 0x8048cec
(gdb) p /x *(0x8049a5c + 20)
$9 = 0x8048cf8
(gdb) p /x *(0x8049a5c + 24)
$10 = 0x8048d04
(gdb) p /x *(0x8049a5c + 28)
$11 = 0x8048d10
```

这样可以将接下来的代码划分模块:

```
1 // case 1:
2 8048cc1:
                      b8 00 00 00 00
                                                          $0 \times 0, \% eax
                                                 mov
                                                         8048 ccd < phase_3 + 0 \times 4f >
3 8048cc6:
                      eb 05
                                                 jmp
4 // case 0:
5 8048cc8:
                      b8 8e 01 00 00
                                                 mov
                                                         $0x18e,%eax
 6 8048 ccd :
                      2d 66 01 00 00
                                                         $0 \times 166, \% eax
                                                 sub
7 8048 cd2:
                                                          8048cd9 < phase 3+0x5b >
                      eb 05
                                                 imp
8 // case 2:
9 8048cd4:
                      b8 00 00 00 00
                                                         $0\times0,\%eax
                                                 mov
10 8048 cd9:
                      05 92 00 00 00
                                                 add
                                                         $0x92,%eax
11 8048 cde:
                      eb 05
                                                         8048ce5 < phase_3 + 0x67 >
                                                 jmp
12 // case 3:
13 8048 ce 0 :
                      b8 00 00 00 00
                                                          $0 \times 0, \% eax
                                                 mov
                                                          $0 \times e9, \% eax
14 8048 ce5:
                      2d e9 00 00 00
                                                 sub
15 8048 cea:
                      eb 05
                                                         8048cf1 <phase_3+0x73>
                                                 jmp
16 // case 4:
17 8048 cec:
                      b8 00 00 00 00
                                                          $0 \times 0, \% eax
                                                 mov
18 8048 cf1:
                      05 28 01 00 00
                                                          $0×128.%eax
                                                 add
19 8048 cf6:
                      eb 05
                                                          8048 cfd < phase_3 + 0x7f >
                                                 jmp
20 // case 5:
21 8048 cf8:
                      b8 00 00 00 00
                                                          $0\times0,\% eax
                                                 mov
                                                         0x1df,\%eax
22 8048 cfd:
                      2d df 01 00 00
                                                 sub
```

```
23 8048d02:
                     eb 05
                                                       8048d09 <phase_3+0x8b>
                                               jmp
24 // case 6:
25 8048d04:
                     b8 00 00 00 00
                                                       $0 \times 0, \% eax
                                               mov
                     05 df 01 00 00
                                                       $0x1df,%eax
26 8048d09:
                                               add
                                                       8048d15 <phase_3+0x97>
27 8048 d0e:
                     eb 05
                                               imp
  // case 7:
  8048d10:
                                                       $0\times0,\%eax
                     b8 00 00 00 00
                                               mov
30 8048d15:
                     83 c0 80
                                                add
                                                       $0xffffff80,%eax
```

• 于是可以得出对应的 c 语言代码:

```
1
   int fun(int a) {
              int ret = 0;
2
              switch (a) {
                        case 1: ret = 0;
4
                        case 0:
5
                                   if (a != 1) ret = 0 \times 18e;
6
                                   ret -= 0 \times 166;
7
                        case 2: ret += 0 \times 92;
8
                        case 3: ret -= 0 \times e9;
9
                        case 4: ret += 0 \times 128;
10
                        case 5: ret -= 0 \times 1 df;
11
12
                        case 6: ret += 0 \times 1 df;
                        case 7: ret += 0 xffffff80;
13
14
              }
15
              return ret;
16
```

经过整理后的代码如下:

```
int fun(int a) {
           int ret = 0;
2
           switch(a) {
3
                    case 0: ret = 121; break;
                    case 1 : ret = -277; break;
5
                    case 2 : ret = 81; break;
                    case 3 : ret = -65; break;
7
                    case 4 : ret = 168; break;
8
                    case 5 : ret = -128; break;
9
                    case 6: ret = 351; break;
10
                    case 7 : ret = -128; break;
11
12
           return ret;
13
```

14 }

• 在 switch 语句段后,有这么一段:

```
8048 d1f:
                    b8 00 00 00 00
                                                        $0 \times 0, \% eax
                                               mov
1
   8048d24:
                    83 7c 24 18 05
                                                        $0x5,0x18(\%esp)
                                               cmpl
2
  8048d29:
                    7f 06
                                                        8048d31 <phase_3+0xb3>
3
                                               jg
  8048d2b:
                    3b 44 24 1c
                                               cmp
                                                       0 \times 1 c(\% esp), \% eax
  8048 d2f:
                    74 05
                                                        8048d36 <phase_3+0xb8>
                                                jе
   8048d31:
                    e8 9d 08 00 00
                                                       80495d3 <explode_bomb>
                                                call
```

这里给出了第二个限制条件:输入的第一个整数 $a \le 5$

最后,程序将输入的第二个整数 b 与 switch 的结果进行比较,不同则炸弹爆炸。

• 因此,这关的答案就水落石出了:

输入一个整数 $a(0 \le a \le 5)$, 和 a 经过 switch 之后的结果 b

选用 a = 4, b = 168, 顺利过关

```
I turned the moon into something I like to call a Death Star.

Phase 1 defused. How about the next one?

1 2 6 24 120 720

That's number 2. Keep going!

4 168

Halfway there!
```

• 第四关的汇编代码如下:

```
1
   08048d64 <phase_4>:
2
     8048d64:
                                                             $0 \times 2c, \% esp
                       83 ec 2c
                                                    sub
3
                                                             0 \times 1 c(\% esp), \% eax
     8048d67:
                       8d 44 24 1c
                                                    lea
4
                                                            \%eax, 0 \times 8(\%esp)
     8048d6b:
                       89 44 24 08
5
                                                    mov
                                                             $0 \times 8049 dc9, 0 \times 4(\% esp)
     8048d6f:
                       c7 44 24 04 c9 9d 04
                                                    movl
     8048d76:
7
     8048d77:
                       8b 44 24 30
                                                            0 \times 30(\% esp), \% eax
                                                    mov
8
     8048d7b:
                       89 04 24
                                                    mov
                                                            %eax,(%esp)
     8048d7e:
                       e8 8d fb ff ff
                                                             8048910 < \_isoc99\_sscanf@plt>
                                                    call
10
     8048d83:
                       83 f8 01
                                                             $0×1,%eax
11
                                                    cmp
                                                            8048d8f <phase_4+0x2b>
     8048d86:
                       75 07
12
                                                    jne
     8048d88:
                       83 7c 24 1c 00
                                                             $0\times0,0\times1c(\%esp)
                                                    cmpl
13
                       7f 05
                                                             8048d94 < phase 4+0x30>
14
     8048d8d:
                                                    jg
     8048d8f:
                       e8 3f 08 00 00
                                                             80495d3 <explode_bomb>
15
                                                    call
     8048d94:
                       8b 44 24 1c
                                                            0 \times 1 c(\% esp), \% eax
16
                                                    mov
                                                            %eax,(%esp)
     8048d98:
                       89 04 24
17
                                                    mov
     8048d9b:
                       e8 9a ff ff ff
                                                             8048d3a <func4>
                                                    call
18
19
     8048 da0:
                       3d a7 41 00 00
                                                    cmp
                                                             $0x41a7,%eax
                                                            8048dac <phase_4+0x48>
     8048 da5:
                       74 05
20
                                                    jе
                                                            80495d3 <explode_bomb>
     8048da7:
                       e8 27 08 00 00
21
                                                    call
     8048 dac:
                                                             $0 \times 2c, \% esp
                       83 c4 2c
                                                    add
     8048 daf:
23
                       с3
                                                    ret
```

首先,它和 phase_3 一样,调用了 sscanf 函数。和上一个函数的处理方法一样,运用 gdb,发现 sscanf 的参数是"%d",即读入一个整数 n,存放在 0x1c(%esp)。

```
1 8048d88: 83 7c 24 1c 00 cmpl $0x0,0x1c(%esp)
2 8048d8d: 7f 05 jg 8048d94 <phase_4+0x30>
3 8048d8f: e8 3f 08 00 00 call 80495d3 <explode_bomb>
```

读入后,将n与0比较,如果 $n \le 0$ 炸弹就会爆炸。

因此,我们得到了第一个限制条件:n>0

```
1 8048d94: 8b 44 24 1c mov 0x1c(%esp),%eax
2 8048d98: 89 04 24 mov %eax,(%esp)
3 8048d9b: e8 9a ff ff ff call 8048d3a <func4>
```

随后,将 n 作为参数传入函数 func4 中。

1 8048da0: 3d a7 41 00 00 cmp \$0x41a7,%eax 2 8048da5: 74 05 je 8048dac <phase_4+0x48> 3 8048da7: e8 27 08 00 00 call 80495d3 <explode_bomb>

这段代码表示,如果函数的返回值不为 0x41a7,则炸弹爆炸。

这样就知道了第四关的要求:找出使 func4 的结果为 0x41a7 的输入值。

• 于是我们开始研究 func4 函数。

```
1
   08048d3a <func4>:
    8048d3a:
                    83 ec 1c
                                                         $0 \times 1c, \% esp
                                                 sub
3
    8048d3d:
                     8b 54 24 20
                                                        0\times20(\%esp),\%edx
4
                                                 mov
    8048d41:
                     b8 01 00 00 00
                                                         $0x1,%eax
5
                                                 mov
    8048d46:
                     85 d2
                                                        %edx.%edx
                                                 test
6
                                                         8048d60 < func4+0x26>
    8048d48:
                     7e 16
                                                 jle
                                                         $0 \times 1, \% edx
8
    8048d4a:
                     83 ea 01
                                                 sub
    8048d4d:
                     89 14 24
                                                        %edx,(%esp)
                                                 mov
9
                                                         8048d3a <func4>
    8048d50:
                     e8 e5 ff ff ff
                                                 call
10
    8048d55:
                                                        0\times0 (, % eax, 8), % edx
11
                     8d 14 c5 00 00 00 00
                                                 lea
                                                        %eax,%edx
    8048d5c:
                     29 c2
                                                 sub
                                                        %edx,%eax
    8048d5e:
                     89 d0
                                                 mov
13
    8048d60:
                     83 c4 1c
                                                         $0 \times 1c, \% esp
                                                 add
14
15
    8048d63:
                     с3
                                                 ret
```

分析这一个函数,可以得出它的作用如下:

- 首先,函数将返回值置为1。
- 如果传入参数 $x \le 0$, 则直接返回。
- 递归调用 fun4
- 将递归的结果乘 7 后作为返回值

对应的 C 语言代码如下:

```
int fun4(int x) {
    int ret = 1;
    if (x <= 0)
        return ret;
    return ret * 7;
}</pre>
```

因此,我们的问题变成: $log_7(0x41a7) = ?$

经过计算得出, 7⁵ = 16807 = 0x41a7

输入 5,成功过关。

```
I turned the moon into something I like to call a Death Star.
Phase 1 defused. How about the next one?
1 2 6 24 120 720
That's number 2. Keep going!
4 168
Halfway there!
5 austinpowers
So you got that one. Try this one.
```

• 第五关的汇编代码如下:

```
1
   08048db0 <phase_5>:
2
                                                             %ebx
     8048db0:
                       53
                                                     push
3
                                                             $0 \times 28, \% esp
     8048db1:
                       83 ec 28
                                                     sub
4
                                                             0 \times 30(\% esp), \% ebx
     8048db4:
                       8b 5c 24 30
                                                     mov
5
     8048db8:
                                                             %gs:0\times14, %eax
                       65 a1 14 00 00 00
                                                     mov
                                                             \%eax, 0 \times 1c(\%esp)
     8048 dbe:
                       89 44 24 1c
                                                     mov
7
     8048dc2:
                       31 c0
                                                             %eax,%eax
8
                                                     xor
     8048dc4:
                       89 1c 24
                                                             %ebx,(%esp)
9
                                                     mov
     8048dc7:
                       e8 5f 02 00 00
                                                             804902b <string_length>
                                                     call
10
     8048 dcc:
                                                             $0\times6,\%eax
                       83 f8 06
11
                                                     cmp
                                                             8048dd6 <phase_5+0x26>
     8048 dcf:
                       74 05
12
                                                     jе
     8048dd1:
                       e8 fd 07 00 00
                                                             80495d3 <explode_bomb>
                                                     call
13
     8048dd6:
                       b8 00 00 00 00
                                                     mov
                                                             $0 \times 0.\% eax
14
     8048ddb:
                       Of be 14 03
                                                     movsbl (\%ebx,\%eax,1),\%edx
15
     8048 ddf:
                       83 e2 Of
                                                             0xf,\%edx
                                                     and
16
                                                     movzbl 0 \times 8049 a7 c(\% ed \times), \% ed \times
     8048 de 2 :
                       0f b6 92 7c 9a 04 08
17
                                                             \%dI, 0 \times 15(\%esp, \%eax, 1)
     8048 de 9 :
                       88 54 04 15
                                                     mov
18
                                                             $0×1,%eax
19
     8048 ded:
                       83 c0 01
                                                     add
                       83 f8 06
     8048 df0:
                                                             $0\times6,\%eax
20
                                                     cmp
                                                             8048ddb <phase_5+0x2b>
     8048 df3:
                       75 e6
                                                     jne
21
     8048 df5:
                                                             $0 \times 0, 0 \times 1b(\% esp)
                       c6 44 24 1b 00
                                                     movb
     8048 dfa:
                       c7 44 24 04 52 9a 04
                                                             $0 \times 8049a52, 0 \times 4(\% esp)
23
                                                     movl
     8048e01:
                       08
24
     8048e02:
                       8d 44 24 15
                                                     lea
                                                             0 \times 15(\% esp), \% eax
25
     8048e06:
                                                             %eax,(%esp)
                       89 04 24
26
                                                     mov
     8048e09:
                       e8 36 02 00 00
                                                             8049044 <strings_not_equal>
                                                     call
                                                             %eax,%eax
     8048e0e:
                       85 c0
                                                     test
28
                                                             8048e17 <phase_5+0x67>
29
     8048e10:
                       74 05
                                                     je
     8048e12:
                       e8 bc 07 00 00
                                                             80495d3 <explode_bomb>
                                                     call
30
     8048e17:
                       8b 44 24 1c
                                                             0 \times 1 c(\% esp), \% eax
31
                                                     mov
     8048e1b:
                       65 33 05 14 00 00 00
                                                             %gs:0\times14, %eax
32
                                                     xor
     8048e22:
                       74 05
                                                             8048e29 <phase_5+0x79>
33
                                                     jе
                                                             8048850 <__stack_chk_fail@plt>
     8048e24:
                       e8 27 fa ff ff
                                                     call
34
     8048e29:
                       83 c4 28
                                                     add
                                                             $0 \times 28, \% esp
35
                                                             %ebx
     8048e2c:
                       5 b
                                                     pop
36
37
     8048e2d:
                       с3
                                                     ret
```

```
8048dc7:
                    e8 5f 02 00 00
                                                       804902b <string_length>
1
   8048 dcc:
                    83 f8 06
                                                       $0\times6,\%eax
2
                                               cmp
                                                       8048dd6 <phase_5+0x26>
   8048 dcf:
                    74 05
                                               je
   8048dd1:
                    e8 fd 07 00 00
                                                       80495d3 <explode_bomb>
                                               call
```

首先,我们发现它调用了 string_length 这个函数,并且将返回值与 6 进行比较,不相等的话炸弹会爆炸。查看 string_length 的代码发现,它的功能为返回一个字符串的长度。

因此,我们得到了第一个限制条件:需要输入一个长度为6的字符串。

```
8048dd6:
                       b8 00 00 00 00
                                                               $0 \times 0, \% eax
                                                      mov
1
    8048ddb:
                       Of be 14 03
                                                      movsbl (%ebx,%eax,1),%edx
    8048 ddf:
                       83 e2 0f
                                                      and
                                                               0xf,\%edx
3
4
    8048 de 2 :
                       Of b6 92 7c 9a 04 08
                                                      movzbl 0 \times 8049 a7 c(\% edx), \% edx
    8048 de 9 :
                       88 54 04 15
                                                              \%dI, 0 \times 15(\%esp, \%eax, 1)
5
                                                      mov
    8048 ded:
                                                               $0 \times 1.\% eax
                       83 c0 01
                                                      add
6
    8048 df0:
                                                               $0\times6,\%eax
                       83 f8 06
                                                      cmp
                                                               8048ddb <phase_5+0x2b>
    8048 df3:
                       75 e6
                                                      jne
8
```

这里是一个循环,循环变量是%eax,范围从0到5。

```
1 8048ddb: 0f be 14 03 movsbl (%ebx,%eax,1),%edx
2 8048ddf: 83 e2 0f and $0xf,%edx
```

movsbl 这个语句的意思为,将位于 Mem[%ebx + %eax] 处地址的数据最低位取出,进行符号位扩展后送入 %edx 中,结合后面一句 **and** \$0xf, %edx,以及 %ebx 即为输入字符串的首地址,可以发现这句话的意思是取出输入字符串第 %eax 位 ASCII 码的后四位。

```
1 8048 ddf: 83 e2 0f and $0xf,%edx
2 8048 de2: 0f b6 92 7c 9a 04 08 movzbl 0x8049a7c(%edx),%edx
3 8048 de9: 88 54 04 15 mov %dl,0x15(%esp,%eax,1)
```

这个语句将刚刚取出的值作为下标,在从 0x8049a7c 开始的数组中进行寻值,并将结果构成新的字符串压入栈中。于是,我们调用 gdb 命令,查看 0x8049a7c 地址对应的值。

```
(gdb) p (char *) 0x8049a7c
$1 = 0x8049a7c "isrveawhobpnutfgS
```

这里相当于是一个密码加密的过程,即将每一个原先的字母用一个——对应的字母替换。

```
1 8048 df5: c6 44 24 1b 00 movb $0×0,0×1b(%esp)
2 8048 dfa: c7 44 24 04 52 9a 04 movl $0×8049a52,0×4(%esp)
```

```
8048e01:
                   80
   8048e02:
                   8d 44 24 15
                                                   0 \times 15(\% esp), \% eax
                                            lea
   8048e06:
                   89 04 24
                                                   %eax,(%esp)
                                            mov
5
                   e8 36 02 00 00
                                                   8049044 <strings_not_equal>
   8048e09:
                                            call
  8048e0e:
                   85 c0
                                            test
                                                   %eax,%eax
7
  8048e10:
                   74 05
                                                   8048e17 <phase_5+0x67>
                                            jе
  8048e12:
                   e8 bc 07 00 00
                                            call
                                                   80495d3 <explode bomb>
```

这里又向栈中压入了一个 0x8049a52 地址中的数据,并且调用 strings_not_equal 函数。根据之前的分析,如果传入的两个字符串不等,炸弹就会爆炸。因此,我们生成的字符串应该与 0x8049a52 地址中的相同。

• 于是调用 gdb, 查看 0x8049a52 地址中的内容:

```
(gdb) p (char *) 0x8049a52
$2 = 0x8049a52 "saints"
```

● 因此,我们只需要生成"saints"这个字符串即可。根据索引表,"saints"这六个字母对应的索引值分别为 1 , 5 , 0 , 11 , 13 , 1。

我们只需要输入六个 ASCII 码低 4 位为这六个值的字符即可。于是选用 a, e, 0, k, m, a。

输入 ae0kma, 顺利过关。

```
I turned the moon into something I like to call a Death Star.

Phase 1 defused. How about the next one?

1 2 6 24 120 720

That's number 2. Keep going!

4 168

Halfway there!

5 austinpowers

So you got that one. Try this one.

ae0kma

Good work! On to the next...
```

• 第六关的汇编代码如下:

```
1
   08048e91 <phase_6>:
2
     8048e91:
                                                             $0 \times 1c, \% esp
                       83 ec 1c
                                                    sub
3
                                                             $0xa,0x8(\%esp)
     8048e94:
                       c7 44 24 08 0a 00 00
4
                                                    movl
     8048e9b:
                       00
5
     8048e9c:
                       c7 44 24 04 00 00 00
                                                             $0 \times 0, 0 \times 4(\% esp)
                                                     movl
     8048ea3:
                       00
7
     8048ea4:
                                                             0\times20(\%esp),\%eax
                       8b 44 24 20
                                                    mov
8
     8048ea8:
                       89 04 24
                                                             %eax,(%esp)
9
                                                    mov
     8048 eab:
                       e8 f0 fa ff ff
                                                             80489a0 < strtol@plt >
10
                                                     call
     8048eb0:
                       a3 74 c1 04 08
                                                             %eax,0x804c174
11
                                                    mov
     8048eb5:
                       c7 04 24 74 c1 04 08
                                                             $0 \times 804 c174, (\% esp)
12
                                                     movl
     8048 ebc:
                       e8 6d ff ff ff
                                                             8048e2e <fun6>
                                                     call
13
     8048ec1:
                       8b 40 08
                                                             0\times8(\%eax),\%eax
14
                                                    mov
     8048ec4:
                                                             0\times8(\%eax),\%eax
15
                       8b 40 08
                                                    mov
     8048ec7:
                                                             0\times8(\%eax),\%eax
                       8b 40 08
16
                                                    mov
                                                             0\times8(\%eax),\%eax
     8048 eca:
                       8b 40 08
                                                    mov
17
     8048 ecd:
                                                             0\times8(\%eax),\%eax
                       8b 40 08
18
                                                    mov
                                                             0\times8(\%eax),\%eax
19
     8048ed0:
                       8b 40 08
                                                    mov
     8048ed3:
                       8b 15 74 c1 04 08
                                                             0\times804c174, \% edx
20
                                                    mov
     8048ed9:
                                                             %edx,(%eax)
                       39 10
21
                                                    cmp
                                                             8048ee2 <phase_6+0x51>
     8048 edb:
                       74 05
22
                                                    jе
     8048 edd:
                       e8 f1 06 00 00
                                                             80495d3 <explode_bomb>
                                                     call
23
     8048ee2:
                       83 c4 1c
                                                     add
                                                             $0 \times 1c, \% esp
24
     8048ee5:
25
                       с3
                                                     ret
```

首先发现它调用了 strtol 这个函数。观察这个函数的代码发现,它需要传入三个参数,这里分别为 %eax ,
 0 和 0xa。 strtol 的主要功能可以用以下 c 语言代码表示:

```
1 long strtol(const char *str, char **endptr, int base) {
2     long ret = 0;
3     for (; *str != endptr; str++)
4         ret = ret * base + *str - '0';
5     return ret;
6 }
```

这段代码的作用就是将输入的字符串转换为对应的 base 进制数表示。

18

```
8048eb0:
                       a3 74 c1 04 08
                                                              %eax, 0 \times 804c174
1
                                                     mov
     8048eb5:
                       c7 04 24 74 c1 04 08
                                                              $0 \times 804 c174, (\% esp)
2
                                                     movl
     8048 ebc:
                       e8 6d ff ff ff
                                                              8048e2e <fun6>
                                                     call
3
                                                              0\times8(\%eax),\%eax
     8048ec1:
                       8b 40 08
4
                                                     mov
    8048ec4:
                                                              0\times8(\%eax),\%eax
                       8b 40 08
5
                                                     mov
     8048ec7:
                       8b 40 08
                                                              0\times8(\%eax),\%eax
6
                                                     mov
     8048 eca:
                                                              0\times8(\%eax),\%eax
                       8b 40 08
                                                     mov
                                                              0\times8(\%eax),\%eax
     8048 ecd:
                       8b 40 08
                                                     mov
8
     8048ed0:
                                                              0\times8(\%eax),\%eax
                       8b 40 08
9
                                                     mov
                                                              0 \times 804 c174.\% edx
    8048ed3:
                       8b 15 74 c1 04 08
                                                     mov
10
    8048ed9:
                       39 10
                                                              %edx,(%eax)
11
                                                     cmp
```

这里调用了函数 fun6。由于传入的是一个地址 0x804c174,因此 fun6 的返回值是固定的,因此,经过一系列跳转后,在 cmp %edx, (%eax)中,(%eax)的值也是一定的。运用 gdb 设置断点,查看得知此时(%eax)的值为 268。

• 由于 %edx 的值就是我们输入的字符串,答案就唾手可得了。

输入 268, 顺利过关。

```
I turned the moon into something I like to call a Death Star.

Phase 1 defused. How about the next one?

1 2 6 24 120 720

That's number 2. Keep going!

4 168

Halfway there!

5 austinpowers

So you got that one. Try this one.

ae0kma

Good work! On to the next...

268

Curses, you've found the secret phase!

But finding it and solving it are quite different...
```

8 secret_phase

此时的炸弹还没有拆完。根据前人的经验,这里还有一个隐藏关等着破解。

8.1 find the secret phase

• 首先考虑如何进入隐藏关。由于每次拆弹成功都需要调用 phase_defused 函数,想进入新的关卡,突破口肯定在这个函数。函数的代码如下:

```
1
   08049728 <phase_defused >:
2
                                                               $0 \times 7c, \% esp
     8049728:
                       83 ec 7c
                                                      sub
3
                        65 a1 14 00 00 00
                                                              %gs:0 \times 14, %eax
     804972b:
4
                                                      mov
     8049731:
                       89 44 24 6c
                                                              \%eax, 0 \times 6c(\%esp)
                                                      mov
5
     8049735:
                       31 c0
                                                      xor
                                                              %eax,%eax
6
     8049737:
                                                               $0\times1,(\% esp)
7
                       c7 04 24 01 00 00 00
                                                      movl
                        e8 cb fb ff ff
                                                               804930e <send_msg>
     804973e:
                                                      call
8
9
     8049743:
                       83 3d 8c c7 04 08 06
                                                      cmpl
                                                               $0 \times 6, 0 \times 804c78c
     804974a:
                       75 76
                                                               80497c2 < phase_defused + 0x9a >
                                                      jne
10
11
     804974c:
                       8d 44 24 1c
                                                      lea
                                                              0 \times 1 c(\% esp), \% eax
     8049750:
                       89 44 24 0c
                                                              \%eax, 0 \times c(\%esp)
12
                                                      mov
                                                              0 \times 18(\% esp), \% eax
     8049754:
                       8d 44 24 18
                                                      lea
13
                                                              \%eax, 0 \times 8(\%esp)
     8049758:
                       89 44 24 08
14
                                                      mov
     804975c:
                        c7 44 24 04 cc 9d 04
                                                               $0 \times 8049 dcc, 0 \times 4(\% esp)
                                                      movl
15
16
     8049763:
                       08
     8049764:
                        c7 04 24 90 c8 04 08
                                                               $0 \times 804 c890, (\% esp)
17
                                                      movl
                        e8 a0 f1 ff ff
                                                               8048910 < \_isoc99\_sscanf@plt>
     804976b:
                                                      call
18
                                                               $0\times2.\%eax
     8049770:
                       83 f8 02
                                                      cmp
19
     8049773:
                        75 35
                                                      jne
                                                               80497aa < phase_defused + 0x82 >
20
21
     8049775:
                        c7 44 24 04 d2 9d 04
                                                      movl
                                                               $0 \times 8049 dd2, 0 \times 4(\% esp)
     804977c:
                        80
22
     804977d:
                       8d 44 24 1c
                                                              0 \times 1 c(\% esp), \% eax
                                                      lea
23
     8049781:
                       89 04 24
                                                              %eax,(%esp)
                                                      mov
                                                              8049044 < strings_not_equal>
     8049784:
                        e8 bb f8 ff ff
25
                                                      call
                                                              %eax,%eax
26
     8049789:
                        85 c0
                                                      test
     804978b:
                                                               80497aa <phase_defused+0x82>
27
                        75 1d
                                                      ine
     804978d:
                        c7 04 24 e8 9a 04 08
                                                               $0 \times 8049 \text{ ae8}, (\% \text{ esp})
                                                      movl
28
     8049794:
                        e8 07 f1 ff ff
                                                               80488a0 <puts@plt>
                                                      call
29
     8049799:
                        c7 04 24 10 9b 04 08
                                                               $0 \times 8049 \text{ b} 10, (\% \text{ esp})
30
                                                      movl
     80497a0:
                        e8 fb f0 ff ff
                                                               80488a0 <puts@plt>
                                                      call
31
     80497a5:
                        e8 8d f7 ff ff
                                                      call
                                                               8048f37 <secret_phase>
32
```

```
80497aa:
                       c7 04 24 48 9b 04 08
                                                              $0 \times 8049 \text{ b} 48, (\% \text{ esp})
33
                                                     movl
                                                              80488a0 <puts@plt>
     80497b1:
                        e8 ea f0 ff ff
                                                     call
34
     80497b6:
                       c7 04 24 74 9b 04 08
                                                              $0 \times 8049 b74, (\% esp)
35
                                                     movl
     80497bd:
                       e8 de f0 ff ff
                                                     call
                                                              80488a0 <puts@plt>
36
     80497c2:
                       8b 44 24 6c
                                                              0 \times 6 c(\% esp), \% eax
37
                                                     mov
     80497c6:
                       65 33 05 14 00 00 00
                                                              %gs:0\times14, %eax
38
                                                     xor
     80497cd:
                                                              80497d4 < phase defused + 0xac >
                       74 05
                                                     iе
     80497 cf:
                       e8 7c f0 ff ff
                                                              8048850 <__stack_chk_fail@plt>
                                                     call
40
     80497d4:
                       83 c4 7c
                                                     add
                                                              $0 \times 7c, \% esp
41
     80497d7:
                        с3
42
                                                     ret
```

• 观察这个函数,发现这里调用了 secret_phase 函数。

```
8049764:
                      c7 04 24 90 c8 04 08
                                                           $0 \times 804 c890, (\% esp)
1
                                                   movl
    804976b:
                      e8 a0 f1 ff ff
                                                           8048910 < \_\_isoc99\_sscanf@plt >
                                                   call
2
    8049770:
                      83 f8 02
                                                           $0\times2,\% eax
3
                                                   cmp
    8049773:
                      75 35
                                                           80497aa <phase_defused+0x82>
4
                                                   jne
    8049775:
                      c7 44 24 04 d2 9d 04
                                                           $0 \times 8049 dd2, 0 \times 4(\% esp)
                                                   movl
5
    804977c:
                      80
    804977d:
                      8d 44 24 1c
                                                           0 \times 1 c(\% esp), \% eax
                                                   lea
7
    8049781:
                      89 04 24
                                                           %eax,(%esp)
                                                   mov
8
    8049784:
                      e8 bb f8 ff ff
                                                   call
                                                           8049044 <strings_not_equal>
9
10
    8049789:
                      85 c0
                                                   test
                                                           %eax,%eax
                                                           80497aa < phase_defused + 0x82 >
    804978b:
                      75 1d
                                                   jne
    804978d:
                      c7 04 24 e8 9a 04 08
                                                           $0 \times 8049 \text{ ae8}, (\% \text{ esp})
12
                                                   movl
    8049794:
                                                           80488a0 <puts@plt>
                      e8 07 f1 ff ff
13
                                                   call
    8049799:
                      c7 04 24 10 9b 04 08
                                                   movl
                                                           0\times8049b10,(\% esp)
14
15
    80497a0:
                      e8 fb f0 ff ff
                                                           80488a0 <puts@plt>
                                                   call
                      e8 8d f7 ff ff
    80497a5:
                                                           8048f37 <secret_phase>
16
                                                   call
```

这段语句是在第四关拆炸弹成功后执行,若要跳转到 secret_phase 语句,需要以下两个条件:

- 由 cmp \$0x2, %eax 看出,需要输入两个参数
- 这里调用了 strings_not_equal 函数,输入的第二个参数要与 0x8049dd2 中的值一样。
- 于是调用 gdb , 查看 0x8049dd2 中的值:

```
(gdb) p (char *) 0x8049dd2
$3 = 0x8049dd2 "austinpowers"
```

这样进入隐藏关的方法就很清楚了:在解锁第四关时输入 5 austinpowers, 进入隐藏关。

8.2 solve the secret phase

• 隐藏关的代码如下:

```
1
   08048f37 <secret_phase >:
2
3
    8048 f 37 :
                      53
                                                   push
                                                           %ebx
    8048f38:
                                                            $0×18,%esp
                      83 ec 18
                                                   sub
4
                                                            8049612 < read line >
    8048f3b:
                      e8 d2 06 00 00
                                                   call
5
    8048 f40:
                      c7 44 24 08 0a 00 00
                                                            $0xa,0x8(\%esp)
                                                   movl
6
    8048 f47:
                      00
7
    8048f48:
                      c7 44 24 04 00 00 00
                                                            $0 \times 0, 0 \times 4(\% esp)
8
                                                   movl
    8048 f4f:
                      00
9
                                                           %eax,(%esp)
    8048f50:
                      89 04 24
                                                   mov
10
    8048f53:
                      e8 48 fa ff ff
                                                   call
                                                            80489a0 < strtol@plt >
11
                                                           %eax,%ebx
    8048f58:
                      89 c3
                                                   mov
12
                                                           -0\times1(\%eax),\%eax
    8048f5a:
                      8d 40 ff
                                                   lea
13
    8048f5d:
                      3d e8 03 00 00
                                                   cmp
                                                            $0 \times 3 = 8, \% eax
14
    8048 f 62 :
                      76 05
                                                   jbe
                                                            8048f69 <secret_phase+0x32>
15
    8048 f 64:
                      e8 6a 06 00 00
                                                   call
                                                            80495d3 <explode bomb>
16
    8048 f 69:
                      89 5c 24 04
                                                   mov
                                                           \%ebx, 0 \times 4(\%esp)
17
                                                           $0x804c0c0,(%esp)
    8048f6d:
                      c7 04 24 c0 c0 04 08
18
                                                   movl
    8048 f74:
                      e8 6d ff ff ff
                                                            8048ee6 <fun7>
                                                   call
19
    8048 f 79 :
                      83 f8 07
                                                            $0 \times 7, \% eax
                                                   cmp
20
    8048 f7c:
                      74 05
                                                            8048f83 <secret_phase+0x4c>
21
                                                   jе
    8048 f7e:
                      e8 50 06 00 00
                                                            80495d3 <explode_bomb>
22
                                                   call
    8048 f83:
                      c7 04 24 2c 9a 04 08
                                                            $0 \times 8049 a 2c, (% esp)
23
                                                   movl
    8048f8a:
                       e8 11 f9 ff ff
                                                            80488a0 <puts@plt>
                                                   call
24
    8048f8f:
                       e8 94 07 00 00
                                                            8049728 <phase_defused>
                                                   call
25
    8048 f 94:
                      83 c4 18
                                                   add
                                                            $0 \times 18, \% esp
26
    8048 f 97:
                                                           %ebx
                      5b
27
                                                   pop
28
    8048 f 98:
                       с3
```

• 这里也调用了 strtol 函数,用处也相同:将输入的字符串转换为 10 进制整数。

```
8048f40:
                     c7 44 24 08 0a 00 00
                                                          $0xa,0x8(\%esp)
1
                                                  movl
    8048 f47:
                     00
2
    8048 f 48:
                     c7 44 24 04 00 00 00
                                                          $0\times0,0\times4(\%esp)
                                                  movl
3
    8048 f4f:
                     00
4
    8048f50:
                     89 04 24
                                                          %eax,(%esp)
5
                                                  mov
    8048f53:
                     e8 48 fa ff ff
                                                  call
                                                          80489a0 < strtol@plt >
6
```

•

```
8048f58:
                                                           %eax,%ebx
                      89 c3
                                                   mov
1
                                                           -0\times1(\%eax),\%eax
                      8d 40 ff
2
    8048f5a:
                                                   lea
    8048f5d:
                      3d e8 03 00 00
                                                           $0 \times 3 = 8, \% eax
                                                  cmp
3
    8048 f 62:
                      76 05
                                                   jbe
                                                           8048f69 <secret_phase+0x32>
4
                      e8 6a 06 00 00
                                                           80495d3 <explode bomb>
    8048 f 64:
                                                   call
5
```

这里表示,我们输入的整数必须要不大于 0x3e9。

```
8048 f 69:
                      89 5c 24 04
                                                    mov
                                                            \%ebx, 0 \times 4(\%esp)
1
    8048 f6d:
                      c7 04 24 c0 c0 04 08
                                                            $0 \times 804 c0 c0, (\% esp)
                                                    movl
2
                                                            8048ee6 <fun7>
    8048 f74 :
                      e8 6d ff ff ff
                                                    call
3
    8048 f 79 :
                                                            $0 \times 7, \% eax
4
                      83 f8 07
                                                    cmp
    8048 f7c:
                                                            8048f83 <secret_phase+0x4c>
                      74 05
                                                    jе
5
                                                            80495d3 <explode bomb>
    8048 f7e:
                      e8 50 06 00 00
                                                    call
```

这一段表示,程序将存在 0x804c0c0 处的数据作为参数,传入 fun7 函数中。如果 fun7 的返回值不为 7 ,炸弹爆炸。

- 因此,我们的目标就是:找出使 fun7 的返回值为 7 的输入参数。
- fun7 的代码如下:

```
1
    08048ee6 <fun7>:
     8048ee6:
                       53
                                                             %ebx
                                                     push
 3
     8048ee7:
                                                              $0×18,%esp
                       83 ec 18
                                                     sub
 4
     8048 eea:
                       8b 54 24 20
                                                             0 \times 20(\% esp), \% edx
 5
                                                     mov
     8048 eee :
                       8b 4c 24 24
                                                             0 \times 24(\% esp), \% ecx
 6
                                                     mov
                                                             %edx,%edx
     8048 ef2:
                       85 d2
 7
                                                     test
     8048 ef4:
                                                              8048 f2d < fun7 + 0x47 >
                       74 37
 8
                                                     įе
     8048 ef6:
                       8b 1a
                                                             (\%edx),\%ebx
                                                     mov
 9
     8048 ef8:
                       39 cb
                                                     cmp
                                                             %ecx,%ebx
10
     8048 efa:
                                                             8048 f0f < fun7 + 0x29 >
                       7e 13
                                                     ile
11
                       89 4c 24 04
                                                             \%ecx, 0 \times 4(\%esp)
     8048 efc:
12
                                                     mov
                                                             0\times4(\%edx),\%eax
     8048 f00:
                       8b 42 04
                                                     mov
13
                                                             %eax,(%esp)
14
     8048 f 03:
                       89 04 24
                                                     mov
                                                             8048ee6 <fun7>
15
     8048 f 06:
                       e8 db ff ff ff
                                                     call
     8048f0b:
                       01 c0
                                                             %eax.%eax
                                                     add
16
                                                              8048f32 < fun7 + 0x4c >
     8048 f0d:
                       eb 23
                                                     jmp
17
     8048 f 0 f:
                       b8 00 00 00 00
                                                              $0\times0,\% eax
                                                     mov
18
     8048 f14:
                                                             %ecx,%ebx
                       39 cb
                                                     cmp
19
     8048f16:
                       74 1a
                                                              8048f32 < fun7 + 0x4c >
20
                                                     jе
     8048 f18:
                       89 4c 24 04
                                                             \%ecx, 0 \times 4(\%esp)
21
                                                     mov
                                                             0\times8(\%edx),\%eax
     8048f1c:
                       8b 42 08
22
                                                     mov
```

```
89 04 24
                                                          %eax,(%esp)
    8048 f1f:
                                                  mov
23
                      e8 bf ff ff ff
                                                          8048ee6 <fun7>
    8048 f22:
24
                                                  call
                                                          0\times1(\%ea\times,\%ea\times,1),\%ea\times
    8048 f27:
                      8d 44 00 01
                                                  lea
25
    8048 f2b:
                      eb 05
                                                          8048f32 < fun7+0x4c>
                                                  jmp
26
    8048f2d:
                      b8 ff ff ff
                                                          $0xfffffffff,%eax
                                                  mov
27
                      83 c4 18
                                                          $0 \times 18, \% esp
   8048 f32 :
                                                  add
28
    8048f35:
                                                          %ebx
                      5 b
                                                  pop
```

这段代码的等价 c 语言表示如下:

```
int fun7(int *a, int b) {
            if (a = NULL)
2
                   return -1;
3
           int ret;
4
            if (*a > b) {
5
                    ret = fun7(*(a + 4), b);
6
                    ret = ret * 2;
7
           }
8
            else
9
            if (*a == b) {
10
                   ret = 0;
11
           }
12
13
            else {
                    int ret = fun7(*(a + 8), b);
14
                    ret = ret * 2 + 1;
15
            }
16
17
            return ret;
18
```

这个可以理解为一个在一个二叉查找树中查找值的过程,从根开始,如果向右走,当前位就是 1 , 向左走就是 0。由于输出需要为 7 , 因此需要向右走 3 次。

• 首先通过 gdb 得出这一条路上的所有节点:

```
(gdb) p /x *0x804c0c0

$4 = 0x24

(gdb) p /x *(0x804c0c0 + 8)

$5 = 0x804c0d8

(gdb) p /x *0x804c0d8

$6 = 0x32

(gdb) p /x *(0x804c0d8 + 8)

$7 = 0x804c108

(gdb) p /x *0x804c108

$8 = 0x6b

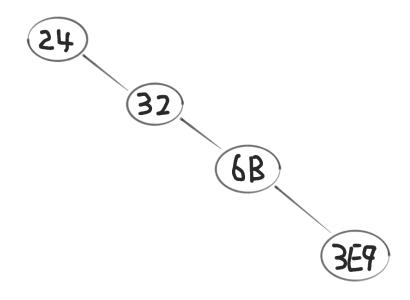
(gdb) p /x *(0x804c108 + 8)

$9 = 0x804c168

(gdb) p /x *0x804c168

$10 = 0x3e9
```

• 于是得出对应的二叉查找树:



• 所以我们输入的值就是 0x3e9, 转换为十进制就是 1001。

输入 1001, 问题解决。

```
I turned the moon into something I like to call a Death Star.
Phase 1 defused. How about the next one?
1 2 6 24 120 720
That's number 2. Keep going!
4 168
Halfway there!
5 austinpowers
So you got that one. Try this one.
ae0kma
Good work! On to the next...
268
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

9 conclusion

这次的 Bomb Lab 总计花了我 7 个小时,从晚上 6 点拆到凌晨 1 点,才算大功告成。不可否认,这个 lab 对我对于汇编的理解起到了很强的巩固作用。从一开始面对冗长汇编代码的手足无措,到一步步将炸弹抽丝剥茧,慢慢拆弹,到最后将隐藏关解决,这其中的每一关对我都是新的挑战。

这次 lab 的覆盖范围非常广泛:第一关的函数调用,第二关的数组的存储,第三关的 switch 跳转表的实现。。 几乎每一关的知识点都不重合,设计者的用心良苦可见一斑。

同时,通过这次 lab,我也系统学习了 gdb 的使用。在平常的编程环境中,我们常常只需要在 IDE 中用鼠标设置断点,点开 watch 窗口,就可以轻松享用 IDE 的调试功能,却很少想过这些调试功能都是怎么实现的。通过这次一步步的手动输入 gdb 命令,我对于 IDE 的调试功能有了一个粗浅的了解。当然,对于主流 IDE 调试功能的更深层次的实现,是我接下来需要探索的。

陆游说过:"纸上得来终觉浅,绝知此事要躬行。"对于 CSAPP 这门课来说,更是如此。通过这次 lab,通过大量的读代码,gdb 调试,我对于代码的底层实现有了新的认识。我想,这就是老师布置这次 lab 的用意之一吧。