# HomeWork5

**SA23011083 吴承泽**

## (1)

经过测试，将参数1、2、3在gdb中调试输入，可以看到

- argc为1的时候，运行`f00()`

```
(gdb) r
Starting program: /home/mospic/Downloads/homework08

Program received signal SIGSEGV, Segmentation fault.
0x49484746 in ?? ()
```

- argc为2的时候，运行`f01()`

```
(gdb) r 2
Starting program: /home/mospic/Downloads/homework08 2

Program received signal SIGSEGV, Segmentation fault.
0x42415a59 in ?? ()
```

- argc为3的时候，运行`f02()`

```
(gdb) r 1 2
Starting program: /home/mospic/Downloads/homework08 1 2
Done.
The program exited normally.
[Inferior 1 (process 2027) exited normally]
```

因此`f00()`、`f01()`会导致段错误，`f02()`不会导致段错误。

## (2)

### f00()

`f00()`函数的汇编码指令如下所示：

```
Breakpoint 1, 0x080484da in f00 ()
(gdb) disas f00
Dump of assembler code for function f00:
   0x080484d1 <+0>:     push   %ebp
   0x080484d2 <+1>:     mov    %esp,%ebp
   0x080484d4 <+3>:     sub    $0x88,%esp
=> 0x080484da <+9>:     sub    $0x8,%esp
   0x080484dd <+12>:    push   $0x804a060
   0x080484e2 <+17>:    lea    -0x83(%ebp),%eax
   0x080484e8 <+23>:    push   %eax
   0x080484e9 <+24>:    call   0x8048320 <strcpy@plt>
   0x080484ee <+29>:    add    $0x10,%esp
   0x080484f1 <+32>:    nop
   0x080484f2 <+33>:    leave
   0x080484f3 <+34>:    ret
```

给`f00()`函数打上断点，执行到入口后进行调试，在函数入口处栈中保存的地址为**0xbfffef3c**为调用函数的返回地址，Lbuffer的首地址为**0x0804a060**，在strcpy的入口处，Lbuffer的地址保存在了**0xbfffeeb5**处，则偏移**offset = 0xbfffef3c - 0xbfffeeb5 = 0x87** 。

```
(gdb) b *0x080484d1
Breakpoint 1 at 0x80484d1
(gdb) b* 0x080484e9
Breakpoint 2 at 0x80484e9
(gdb) b* 0x080484f3
Breakpoint 3 at 0x80484f3
(gdb) r
Starting program: /home/mospic/Downloads/homework08

Breakpoint 1, 0x080484d1 in f00 ()
(gdb) x/x $esp
0xbfffef3c:     0x080485a8
(gdb) c
Continuing.

Breakpoint 2, 0x080484e9 in f00 ()
(gdb) x/x $esp
0xbfffeea0:     0xbfffeeb5
(gdb)
0xbfffeea4:     0x0804a060
(gdb) x/x 0x0804a060
0x804a060 <Lbuffer>:    0x44434241
```

最后执行到`f00()`末尾，溢出后函数返回地址为**0x49484746**。

```
(gdb) c
Continuing.

Breakpoint 3, 0x080484f3 in f00 ()
(gdb) x/x $esp
0xbfffef3c:     0x49484746
```

## f01()

`f01()`函数的汇编码指令如下所示：

```
(gdb) disas f01
Dump of assembler code for function f01:
   0x080484f4 <+0>:     push   %ebp
   0x080484f5 <+1>:     mov    %esp,%ebp
   0x080484f7 <+3>:     sub    $0x508,%esp
   0x080484fd <+9>:     sub    $0x8,%esp
   0x08048500 <+12>:    push   $0x400
   0x08048505 <+17>:    lea    -0x4fe(%ebp),%eax
   0x0804850b <+23>:    push   %eax
   0x0804850c <+24>:    call   0x804846b <init_buf>
   0x08048511 <+29>:    add    $0x10,%esp
   0x08048514 <+32>:    sub    $0x8,%esp
   0x08048517 <+35>:    lea    -0x4fe(%ebp),%eax
   0x0804851d <+41>:    push   %eax
   0x0804851e <+42>:    lea    -0xfe(%ebp),%eax
   0x08048524 <+48>:    push   %eax
   0x08048525 <+49>:    call   0x8048320 <strcpy@plt>
   0x0804852a <+54>:    add    $0x10,%esp
   0x0804852d <+57>:    nop
   0x0804852e <+58>:    leave
   0x0804852f <+59>:    ret
End of assembler dump.
```

给`f01()`函数打上断点，执行到入口后进行调试，得到**0xbfffef3c**为调用函数的返回地址，在strcpy之前，所传入的地址来自栈上，可以看到传入地址来自**0xbfffee3a**，因此偏移**offset = 0xbfffef3c - 0xbfffee3a = 0x102**。

```
(gdb) b *0x080484f4
Breakpoint 1 at 0x80484f4
(gdb) b *0x08048525
Breakpoint 2 at 0x8048525
(gdb) b *0x0804852f
Breakpoint 3 at 0x804852f
(gdb) r 1
Starting program: /home/mospic/Downloads/homework08 1

Breakpoint 1, 0x080484f4 in f01 ()
(gdb) x/x $esp
0xbfffef3c:     0x080485af
(gdb) c
Continuing.

Breakpoint 2, 0x08048525 in f01 ()
(gdb) x/x $esp
0xbfffea20:     0xbfffee3a
(gdb)
0xbfffea24:     0xbfffea3a
(gdb)
0xbfffea28:     0x00000000
```

最后执行到`f01()`末尾，溢出后函数的返回地址为**0x42415a59**。

```
(gdb) c
Continuing.

Breakpoint 3, 0x0804852f in f01 ()
(gdb) x/x $esp
0xbfffef3c:     0x42415a59
(gdb) si
0x42415a59 in ?? ()
(gdb) x/s $eip
0x42415a59:     <error: Cannot access memory at address 0x42415a59>
(gdb)
```