《软件安全》实验报告

姓名：刘星宇　学号：2212824　班级：信息安全法学双学位班

实验名称：

格式化字符串漏洞

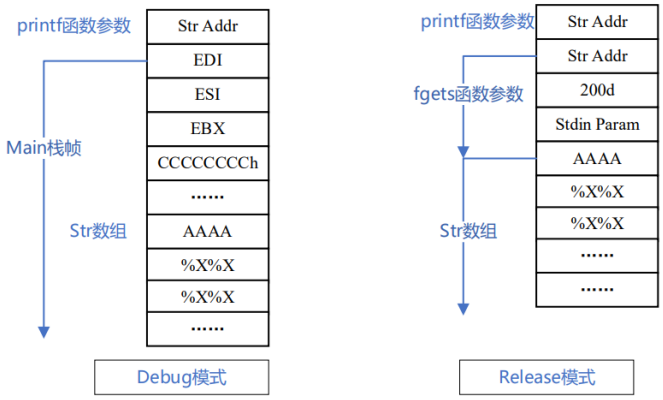
实验要求：

以第四章示例 4-7 代码，完成任意地址的数据获取，观察 Release 模式和 Debug 模式的差异，并进行总结。

实验过程：

1. 编译后运行（Release 模式）并输入：AAAA%x%x%x%x，可以读到 AAAA：AAAA18FE84BB40603041414141（0x41 就是 ASCII 的字母 A 的值）。

   考虑栈帧状态，参数入栈（字符串 str 的地址）后，通过%x 依次读参数下面的内存数据时，很快就读到了原来函数的局部变量 str 的数据了。
2. 执行 printf(str)语句的时候，对比 Debug 模式和 Relase 模式的栈帧结构，如下图所示。



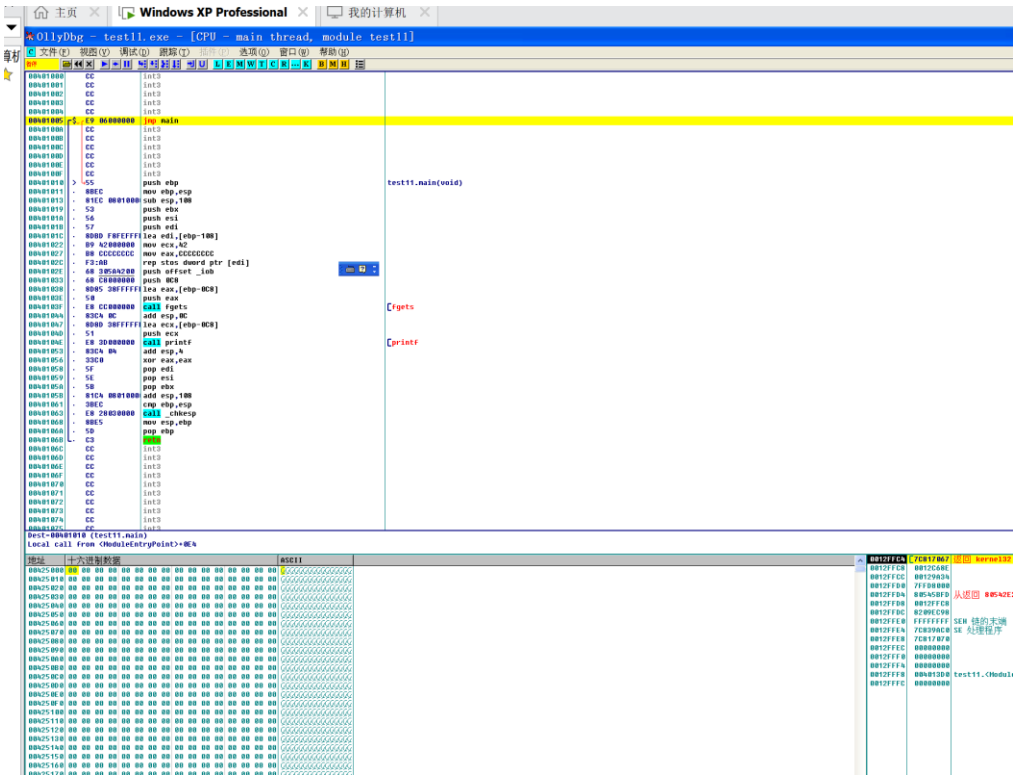
Debug 和 Relase 两种模式下的区别：

Debug 模式下，因为开辟了足够大的栈帧并初始化，char str[200]是从靠近 EBP 的地址分配空间，如果要读到 str 的地址，需要很多的格式化字符；

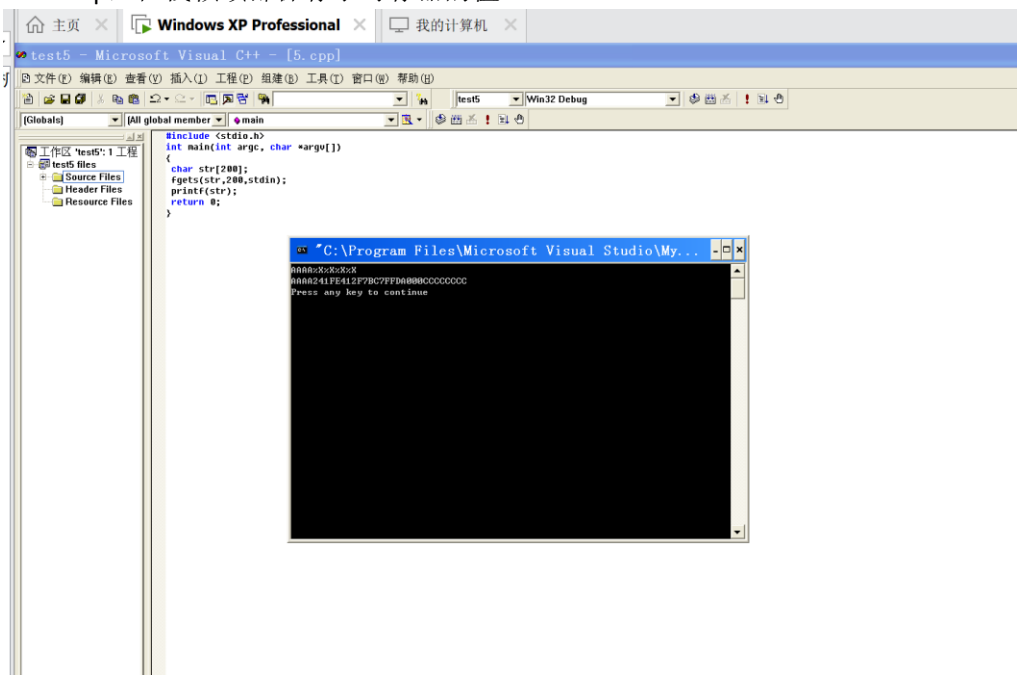
Relase 模式下，并没有严格按照制式的栈帧分配，而是考虑运行性能，在执行到 printf(str)的时候，栈区自顶到底部分为存着“printf 函数参数|fgets 函数参数|str 数组”的内容，在 Main 函数的 retn 语句前，才有一个 add esp XX 的处理。

可以在 Ollydbg 中进行观察：

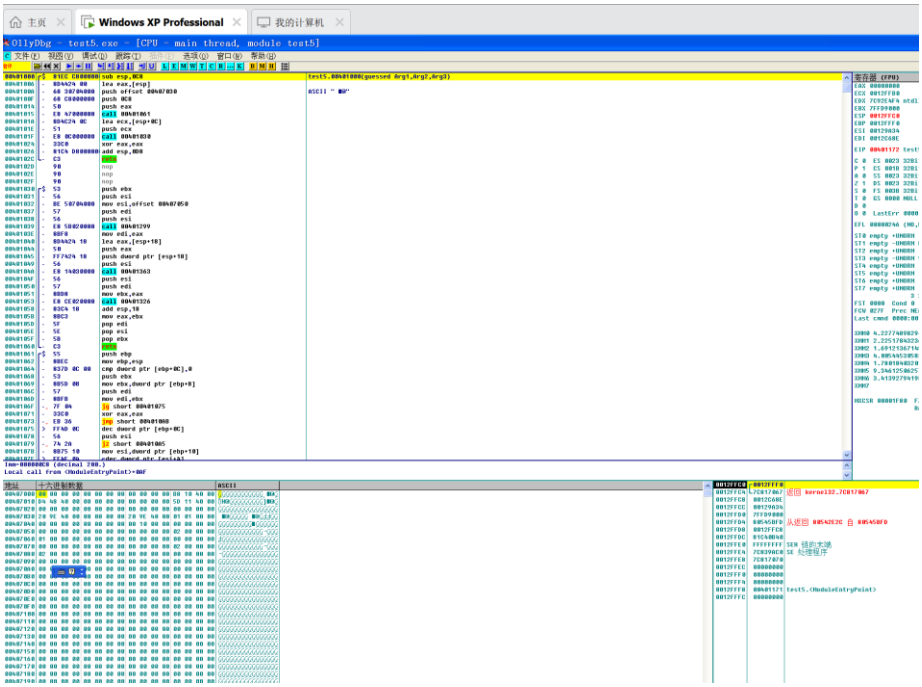
Debug 模式：



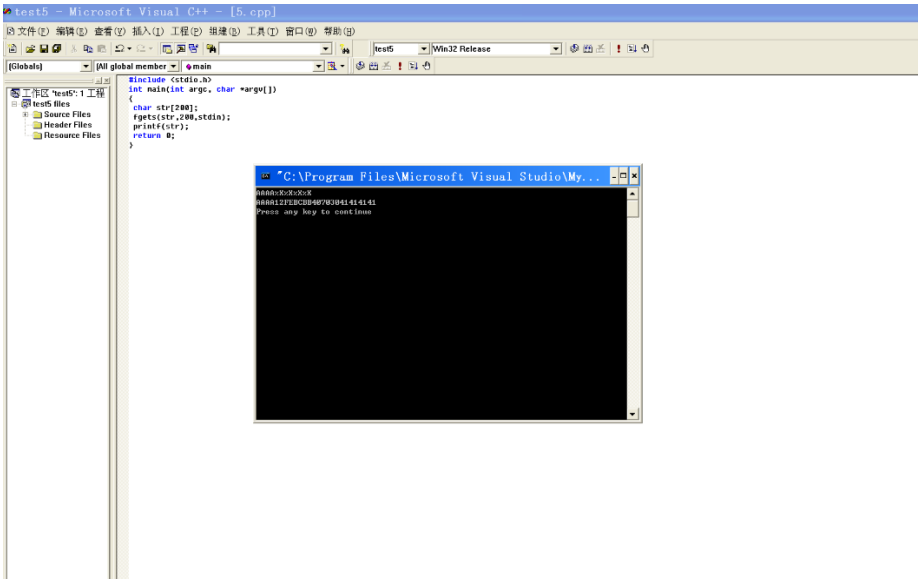
Push ebp，在栈帧顶部保存了寄存器的值。



Release 模式：



直接抬高了 0c8，没有 ebp 入栈，且抬高的较少，仅仅给局部变量声明了空间。



3. 如果将 AAAA 换成地址，第 4 个%x，换成%s 的读取参数指定的地址上的数据，就可以读取任意内存地址的数据。
输入：AAAA%x%x%x%s 这样就构造了去获取 0x41414141 地址上的数据的输入。

心得体会：

通过这次实验，我更加清楚的明白了 debug 和 release 两种不同的模式下的不同的栈帧结构。Debug 模式下，首先 push 寄存器，然后开辟了足够大的栈帧并初始化，如果要读到 str 的地址，需要很多的格式化字符；

Relase 模式下，并没有严格按照制式的栈帧分配，而是考虑运行性能，在执行到 printf(str)的时候，开辟的初始空间较少。
两种模式下%x 的输出是有区别的。