**《软件安全》实验报告**

姓名：胡博浩 学号：2212998 班级：信息安全

**一、实验名称：**

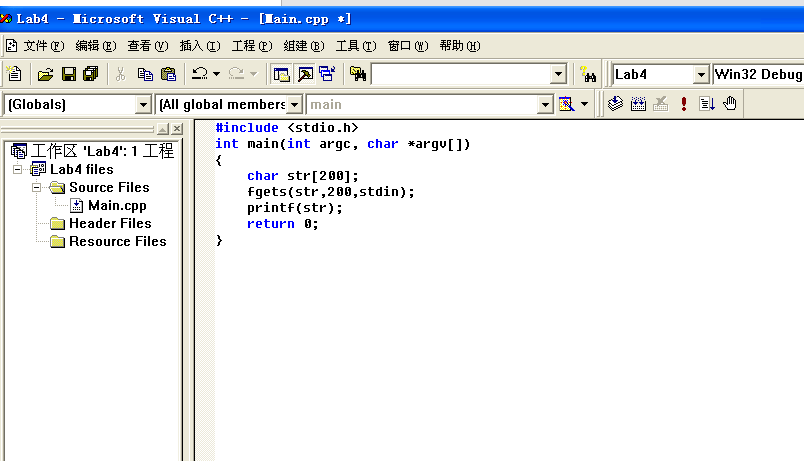
格式化字符串漏洞

**二、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。

**三、实验过程：**

**1.创建程序**

****

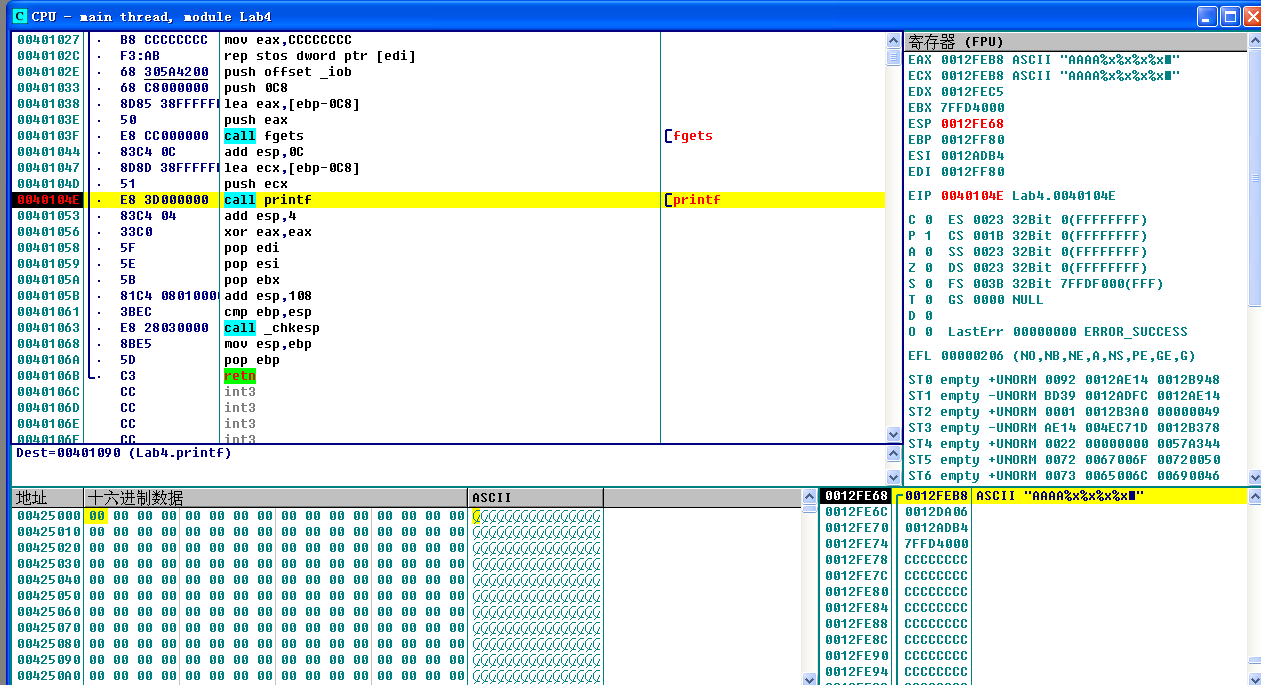
**2.代码分析**

首先创建了一个大小为200字节的str数组，然后从命令行获取一个字符串，之后将这个字符串进行格式化输出。

由于我们要输入的字符串含有格式化字符，但并未给提供对应的参数，因此会将格式化字符串后面的多个栈中的内容取出作为参数，并根据格式化符号将其输出。调用时如果传入“%x%x...%x”,则printf会打印出堆栈中的内容,不断增加%x的个数会逐渐显示堆栈中高地址的数据,从而导致堆栈中的数据泄漏。

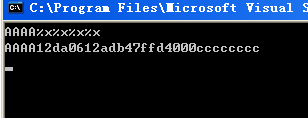
**3.实验过程**

1. 测试在Debug模式下输入“AAAA%x%x%x%x”

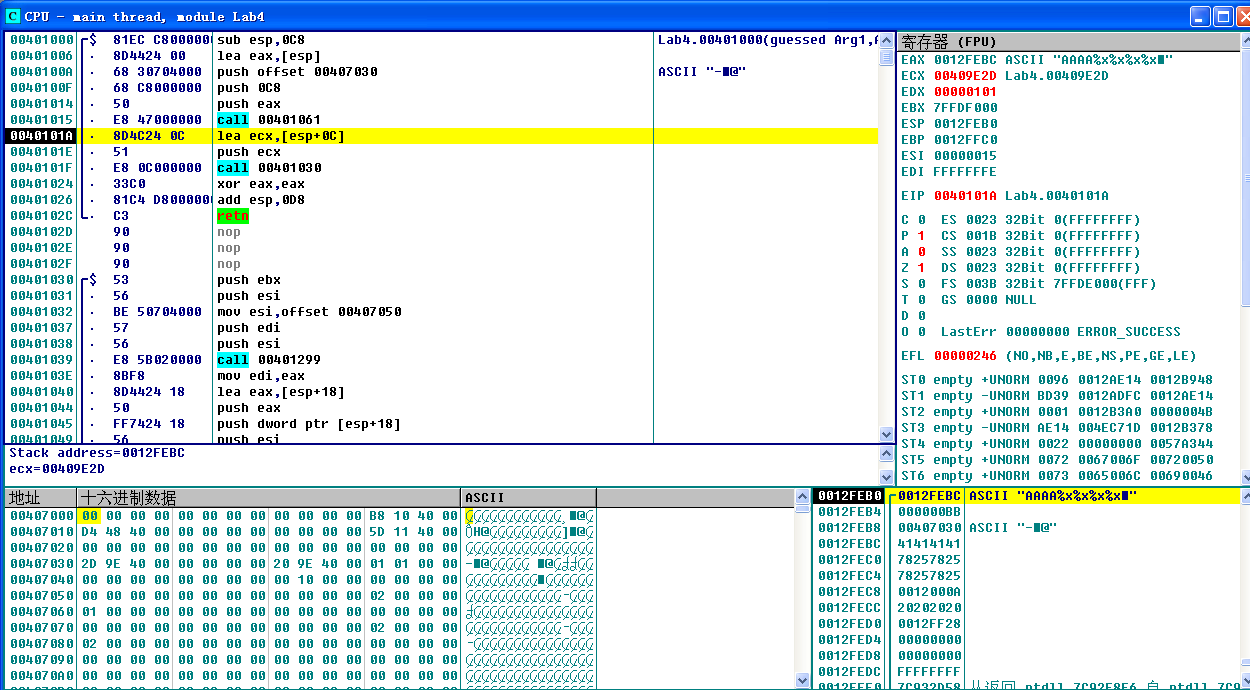


将断点设置在printf函数之前，执行到断点，通过ESP寄存器确定当前的栈顶的地址，然后观察当前栈中的数据情况，如上图所示。

栈中的第一个位置是输入的字符串的首地址，其后跟随的是主函数栈帧中保存的EDI，ESI和EBX。当我们调用格式化输出函数的时候，由于输入的字符串中包含格式化字符，因此函数将会按照格式化字符去寻找参数，即首先会输出AAAA，然后去寻找第一个参数，由于我们并没有提供对应的参数，函数将会在栈中寻找，拿到第一个数据0012DA06，输出为12da06，然后找第二个数据0012ADB4，输出为12adb4，然后寻找第三个数据7FFD4000，输出为7ffd4000，最后寻找第四个数据CCCCCCCC，按照16进制输出为cccccccc。得到最终的输出结果，如下图所示。



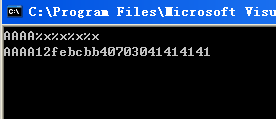
1. 测试在Release模式下输入“AAAA%x%x%x%x”



在Release模式下，在调用printf函数时，栈中的情况如上图所示，那么同理，推测打印的内容应该为

AAAA12febcbb40703041414141

实验结果也正验证了我们的推测。

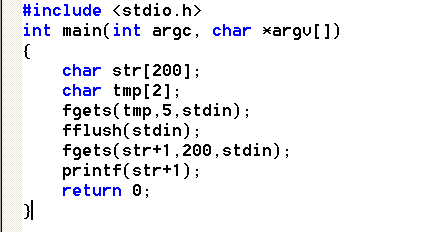


1. 测试在Debug模式下输入“AAAA%x%x%x%s”

在Debug模式下输入AAAA%x%x%x%s并不能得到我们预期的结果，因为对于地址0xCCCCCCCC的访问是非法的，而且这个地址我们很难去修改，但是在Release模式下是可以通过操作实现的。

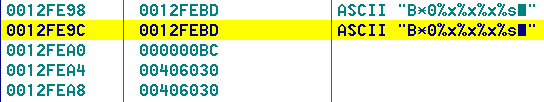
1. 测试在Release模式下输入“AAAA%x%x%x%s”

首先要先修改一下源代码

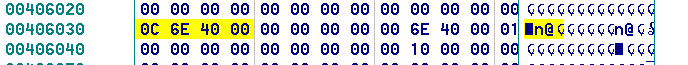


加入tmp变量的目的是为了能够通过字符串的溢出，将str第一位置成0.这样的话我们就可以构造出一个合理的0x00xxxxxx的地址进行访问。选择如下地址0x00422A30进行实验，这意味着需要从str+1处输入B\*0%x%x%x%s。

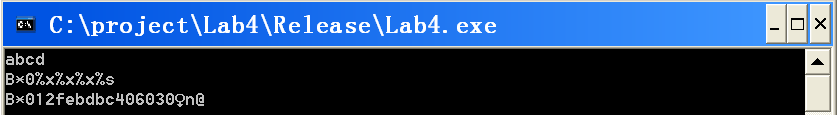
分别输入abcd和AAA%x%x%x%s后，可以观察到在内存中两个字符串存储的内容



而printf函数会将0x00406030作为一个字符串的地址将其输出



可以看到最终的输出结果为



这和预期是完全一致的。

**4.Release和Debug模式的差异**

（1）Debug模式：主要关注代码的调试和可维护性。编译器会保留大量调试信息，程序局部变量的内存空间会被扩大，并且全部初始化为0xCC，并且会在栈中保存一些寄存器的信息，如EDI,ESI,EDX等。生成的代码通常包含额外的调试支持，例如断点和调试符号，以支持开发人员在调试器中进行代码调试和跟踪。

（2）Release模式：主要关注代码的运行效率和性能优化。通常会对代码和数据进行更高程度的优化，删除调试信息，以达到代码最小和速度最优的目的，此时栈中不再保留不必要的内容，并且函数局部变量的空间也不会超过所需要的大小。生成的代码更接近最终产品的执行形式。

**四、心得体会：**

通过本次实验，我深入了解了格式化字符串漏洞的原理和内部机制。使用OllyDbg对printf函数调用前的栈进行观察分析，以及分析输入的带有格式化字符的字符串的输出结果，帮助我理解了漏洞是如何利用格式化字符串的特性来进行内存读取或写入的。这种漏洞发生时，可以由于格式化字符串中的未经检查的用户输入而导致程序执行不可预料的行为，从而引发安全问题。

在尝试利用格式化输出漏洞输出指定地址处的内容时，我意识到可以通过构造特定的格式化字符串，将要读取的地址传递给printf函数。但同时我也意识到需要小心处理，确保读取的地址合法且不会导致程序崩溃或其他意外行为。这一过程提高了我的安全意识和对程序漏洞的认识。

另外，我也学会了区分DEBUG和RELEASE模式下编译出的可执行文件之间的差异，认识到了在软件开发过程中不同编译模式的重要性。