《软件安全》实验报告

姓名：曹瑜 学号：2212794 班级：密码科学与技术

**实验名称：**

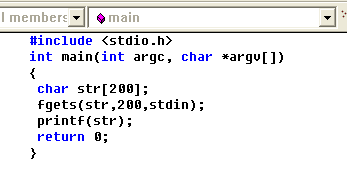
格式化字符串漏洞

**实验要求：**

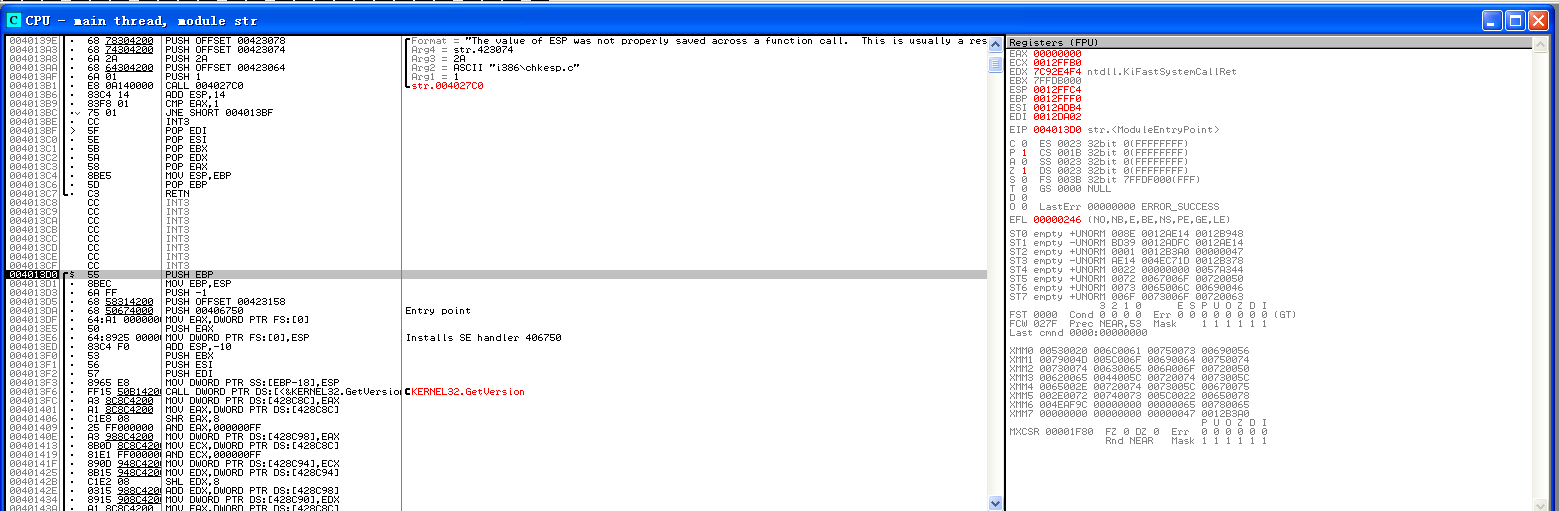
以第四章示例4-7代码，完成任意地址的数据获取，观察Release模式和Debug模式的差异，并进行总结

**实验过程：**

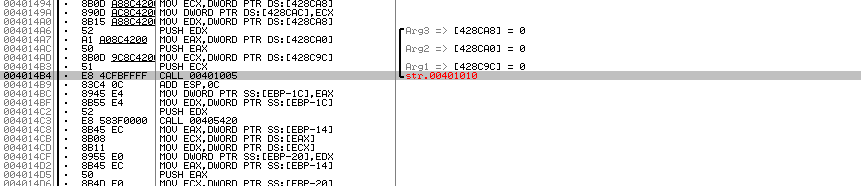
1. 进入VC反汇编



1. DEBUG模式下编译后到ollyDBG中打开：



找到Main函数执行区：

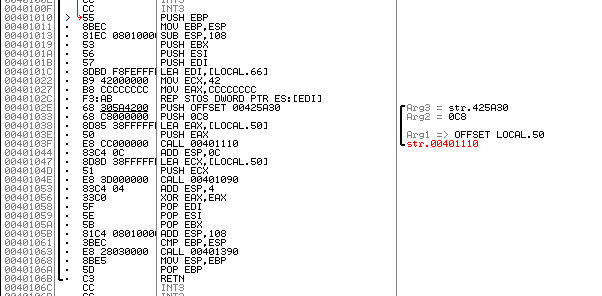


跳转到执行区：观察到操作为：抬高栈帧，留出大量区域；

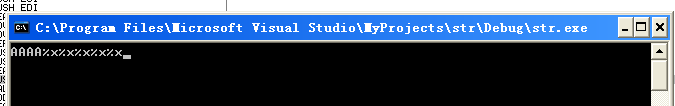
3个push操作在区域顶部保存了调用函数前函数中的一些值；

再初始化大小为108的空间为

cccccccc；



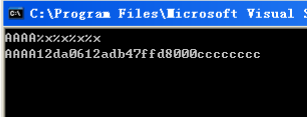
输入字符串：查看寄存器模块可见字符串存储在0x0012FB8中

当前模式下，每发生3个push就对应一个add操作，即调用add后会返回栈帧状态；

程序最终执行结果为：

输入：AAAA%x%x%x%x

输出：AAAA12da0612adb47ffd8000cccccccc



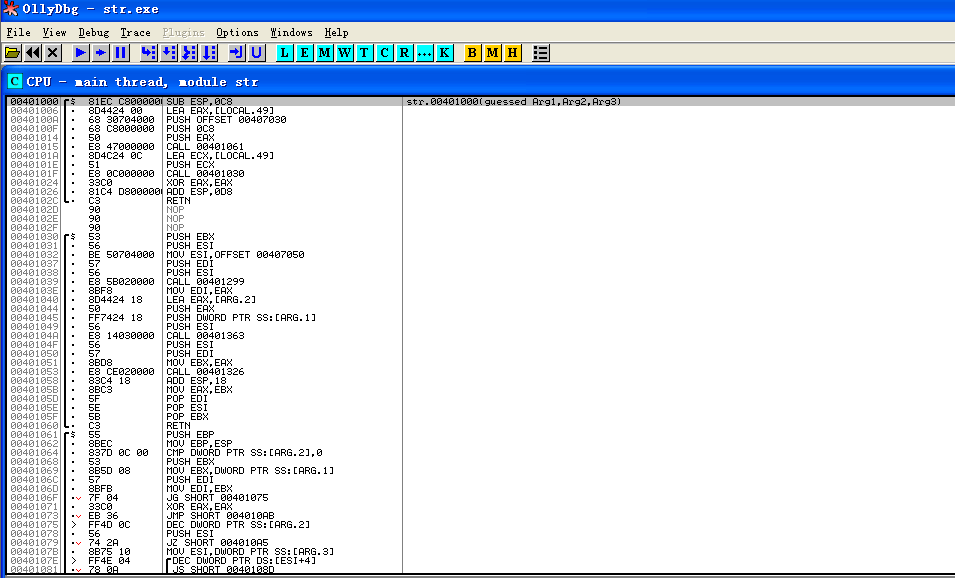
可见此时发生了内存泄漏；

可知在debug模式下，想读取str的地址需要很多的格式化字符；

3、 release模式：

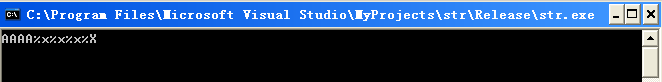
该模式中调试信息比debug模式少；

主函数中并没有发现ebp入栈，而是发现了一个sub抬高，此外，也没有push和寄存器的值，比较简洁，效率也更高



没有ebp入栈，只有栈顶抬高；

输入字符串：查看寄存器模块可见字符串存储在0x0012FB8中

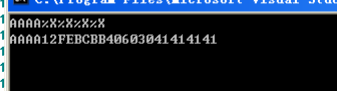


最终结果：

输入：AAAA%x%x%x%x

输出：AAAAAAAA18FE84BB40603041414141

在输出结果中0x41为A的ASCII值



**心得体会：**

通过本次实验，掌握了debug与release模式的差异：在Debug 模式下，开辟了足够大的栈帧并初始化，使得char str[200]从靠近 EBP 的地址分配空间，要读到 str 的地址就需要很多的格式化字符；在Relase 模式下，没有严格按照制式的栈帧分配，执行到 printf(str)的时候，栈区自顶到底部有分存内容，更加简洁。