《软件安全》实验报告

姓名： 冯思程 学号： 2112213 班级： 1120（张健）

**实验名称：**

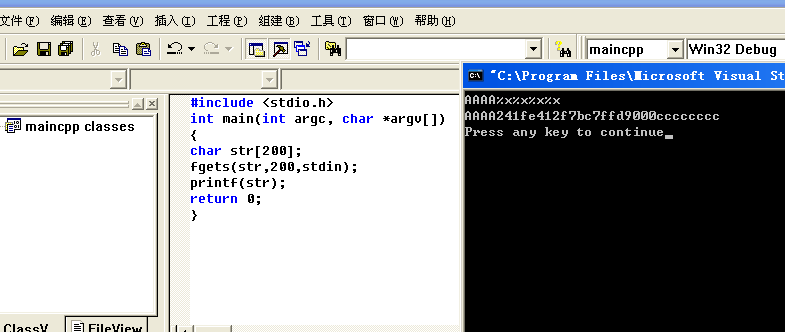
格式化字符串漏洞

**实验要求：**

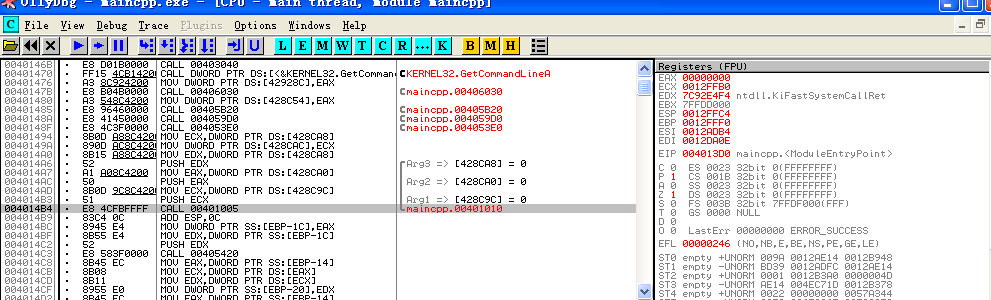
参照第四章示例 4-7 代码，完成任意地址的数据获取。观察 Release 模式和 Debug 模式

的差异，并进行总结。

**实验过程：**

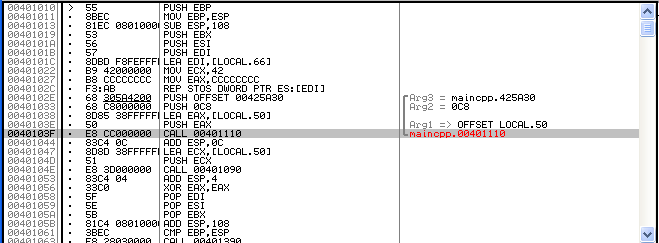
1. Debug模式下：首先，打开VMware虚拟机中的VC6然后将示例4-7的代码输入，然后先在Debug模式下运行，在打开的对话框中输入AAAA%x%x%x%x，发现输出如下：

然后在ollybug中打开Debug模式运行的文件，找到主函数所在位置（右侧有argc注释）如下图：

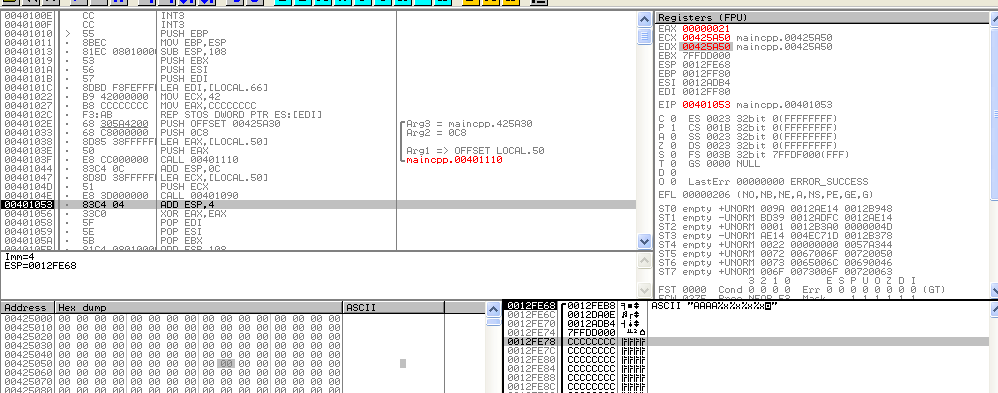


然后F7进入内部，观察代码，发现在调用fgets函数之前，实现了对栈帧的一些寄存器值的保存和对栈帧为函数开辟空间的初始化，如下图：

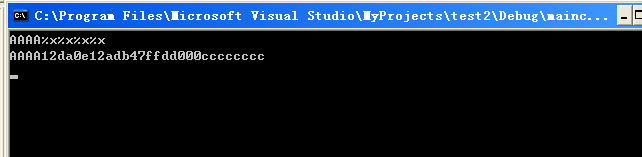
（其中push ebp和mov ebp，esp是栈帧的初始化，sub esp，108是为栈帧开辟空间（开辟了0x108字节空间），然后注意到mov eax，CCCCCCCC和rep stos dword ptr ES:[EDI]这两语句是将开辟出来的栈帧空余空间初始化成0xCCCCCCCCh



不难发现，程序在输出了用户输入的 AAAA（我们知道输入字符串在地址是0012FEB8）之后并未将后面的%x 当作字符串来处理， 而是对每个%x 都输出一个地址的内容，如图中的栈帧向高地址延展在地址存在内容 0012DA0E，0012ADB4，7FFDD000和cccccccc （直接输出）。这样一来，本来不属于字符串内容的部分也被输出出来，发生了内存的泄露。在窗口进行验证发现上述说法是正确的。（内容前面的0被省略）



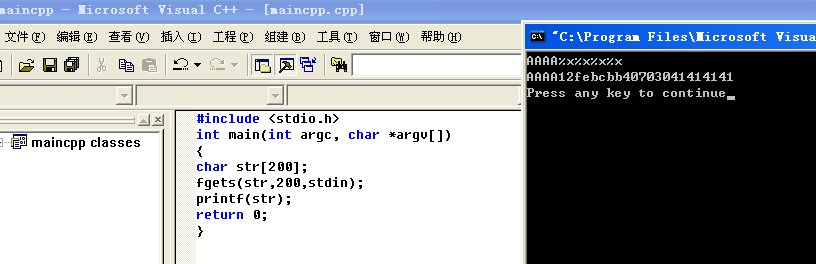




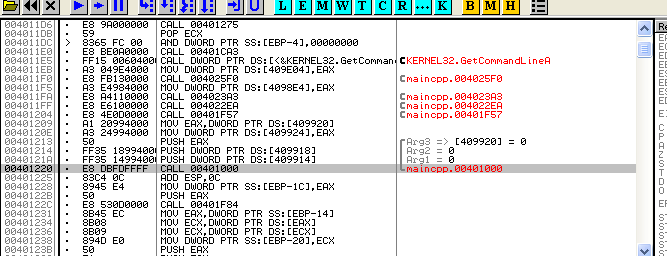
补充：在存输入字符串的地址向高延展的三个地址存储的内容正是EDI,ESI和EBX的内容，因为在前面这三个在开辟栈帧需要用到的寄存器已经存入了。

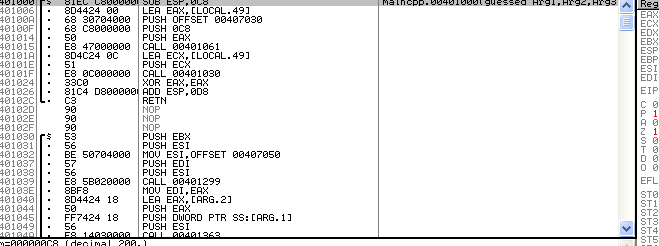
补充：如果我们将 AAAA 所代表的地址修改，并把%x%x%x%x 改为%x%x%x%s，就可以获取任意地址的数据了。

1. release模式下：在VC6中切换到release模式下，运行，结果如下：



然后在ollybug中打开release文件夹中文件，找到主函数（观察到右侧注释有argc标志）F7进入：



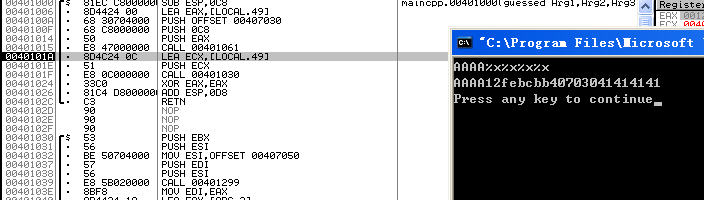


和DEBUG模式下反汇编代码对比可见，release模式没有push ebp,mov ebp,esp的语句，同时sub esp,0xc8正好为str数组的大小，没有空间的多余，而DEBUG模式下为sub esp,0x108，申请了大量的栈空间。

同时，release模式下也没有对寄存器状态的保存(与之对应，DEBUG模式下有push ebx, push esi, push edi)，也没有利用rep stos语句来对栈区进行统一赋值。之后3个push和一个call对应了调用函数fgets(str,200,stdin)。

我们输入AAAA%X%X%X%X，紧接着将调用printf(str)，在call printf语句前栈区的状态

如下图：





在release模式下，此时栈内第一行为printf()的参数str，第二、三、四行为fgets()的三个参数，第五、六、七行存储了输入(AAAA%X%X%X%X)，因此，逻辑上跟debug模式一样，即printf此时会往栈内高地址继续读取四个“参数”，依次输出，但栈的内容却不一样，因此最终输出的结果有所区别：AAAA12FEBCBB406030414141，这个结果也与上图中栈区对应内容一致。

最后执行了add esp,0xD8，这也是release模式和debug的一大区别，即debug在每次call后均会add esp，而release将堆栈平衡进行了统一处理。

1. debug模式和release模式的对比总结部分：

Release 模式下的代码少了很多注释和不必要的代码，一些无用（但有助于我们阅读和修改程序）的代码也被删除，对计算机来说运行负担更小，适合封装为客户端发行等。

Release模式下main函数不执行严格的栈帧转换(即push ebp,mov ebp,esp)，也不对栈空间进行统一初始化(即rep stos指令)，也不通过push保存寄存器原来的值。

Debug模式main函数一开始sub esp会分配更大的栈空间，char str[200]是从靠近 EBP 的地址分配空间，因此在DEBUG模式下如果要读到 str 的地址，需要很多的格式化字符。

**心得体会：**

通过本实验，了解了计算机的格式化字符漏洞为何存在和利用方法。观察到了Debug 模式和 Release 模式下代码各自的特性和适用情况。

同时也进一步熟练了ollydbg的调试方法，更加精通逆向分析的原理和操作。