《漏洞利用及渗透测试基础》实验报告

姓名：申宗尚　学号：2213924　班级：信息安全

实验名称：

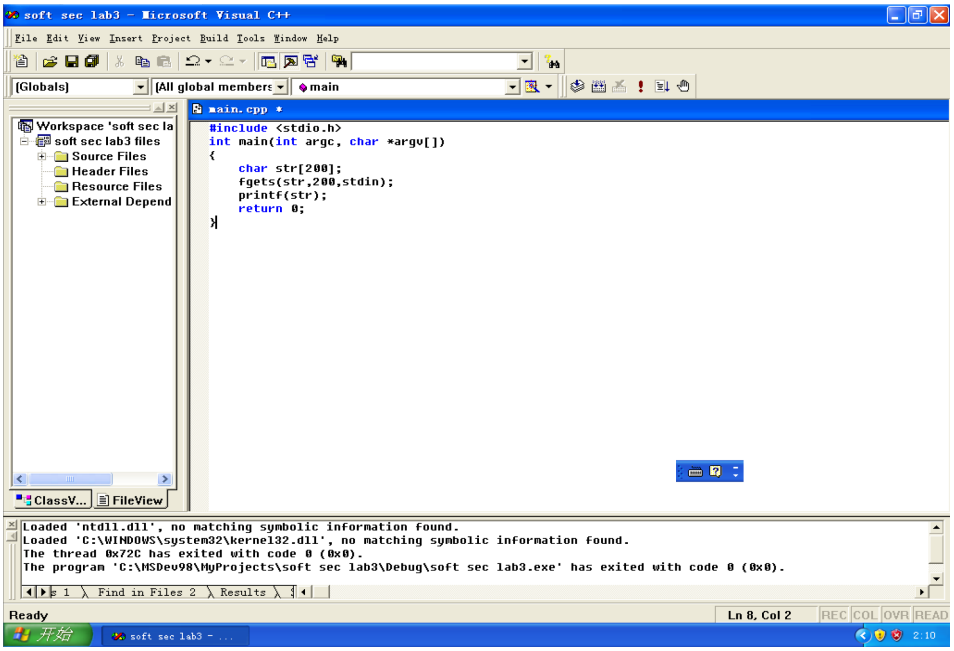
格式化字符串漏洞

实验要求：

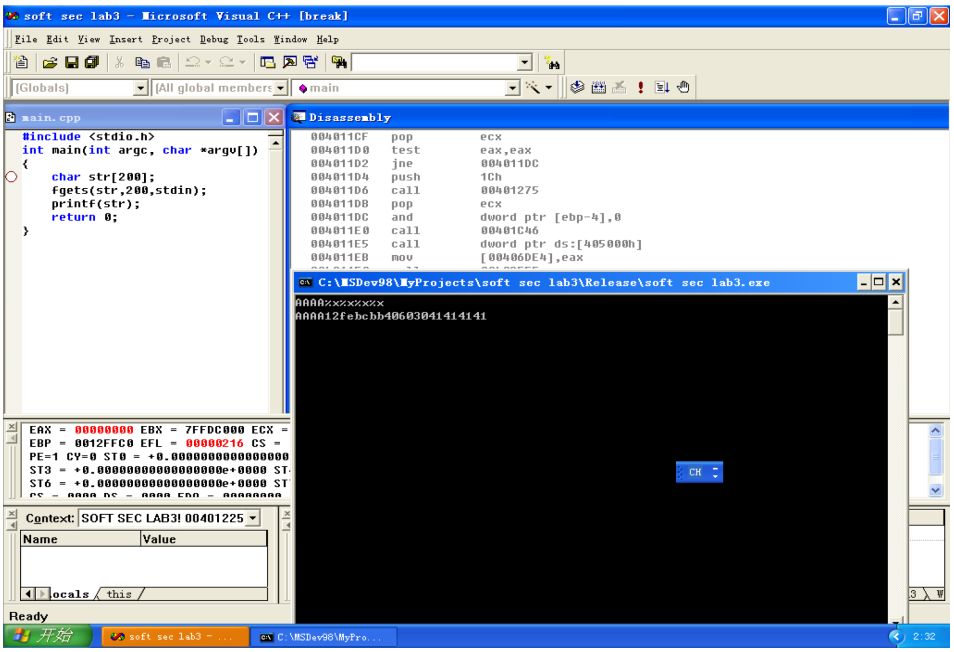
以第四章示例 4-7 代码，完成任意地址的数据获取，观察 Release 模式和 Debug 模式的差异，并进行总结

实验过程：

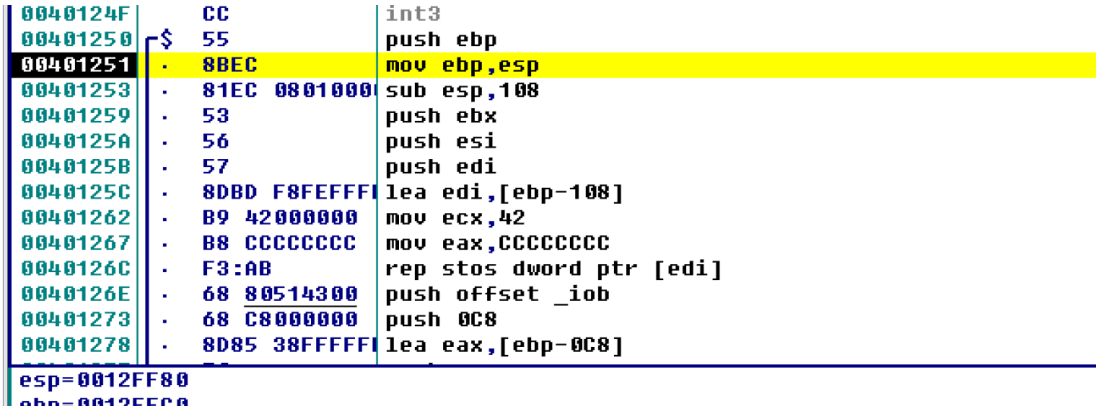
1. 进入 VC6，打开提供的 4-7 代码进行调试(debug 模式和 release 模式)。



2. 运行程序（以 release 模式为例），可以得到如下输出结果。

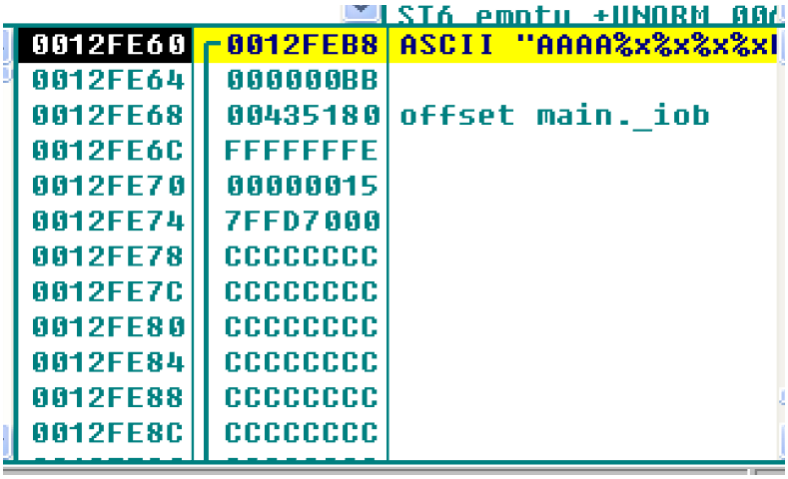


3.打开 ollydbg 对生成的.exe 文件（debug 模式）进行分析，通过 argc 和 argv 找到 main 函数的入口，按 F7 步进进入函数内，可以看到函数内汇编代码如下：

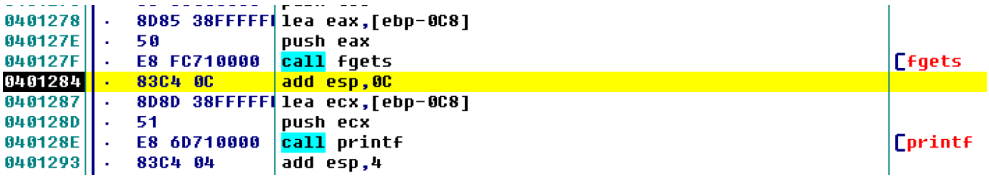


在一开始，首先 push ebp 存下旧栈帧，然后将栈帧移到 esp 位置，最后 sub esp，108，创建新的位置，可以发现，对于 char str[200]，只需要 200 字节，转换为 16 进制为 0xC8，但 debug 模式下却给出了 0x108 大小的空位，并通过后面的 rep stos 指令，将这些空位全部设初始值 0xCCCCCCCC

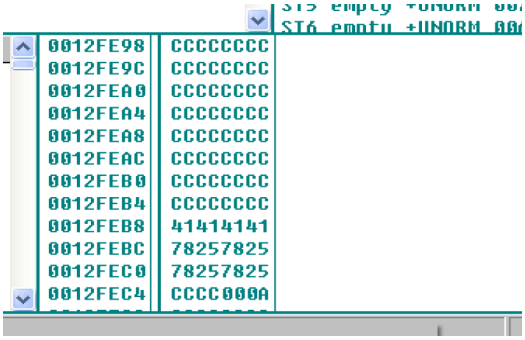
在栈中也可以看到，在栈帧下，全部为 0xCCCCCCCC



4. 继续调试，运行到 fgets，调用 fgets 函数获取输入：AAAA%x%x%x%x

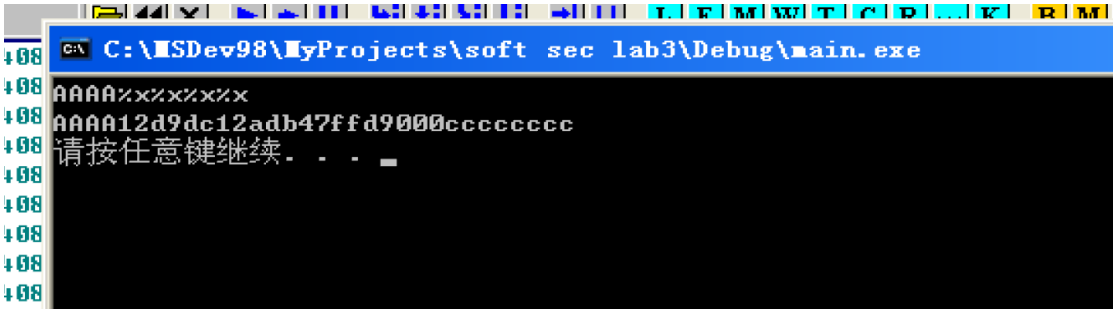


栈变化如下：



可以看到，在 CCCCCCCC 结束之后（即 0x108 大小后），存入了 str 的值，为 AAAA（A 的 ASCII 码 16 进制下为 41），即为 41414141

5. 随后继续调试，直到调用 printf 函数，获取程序输出如下：

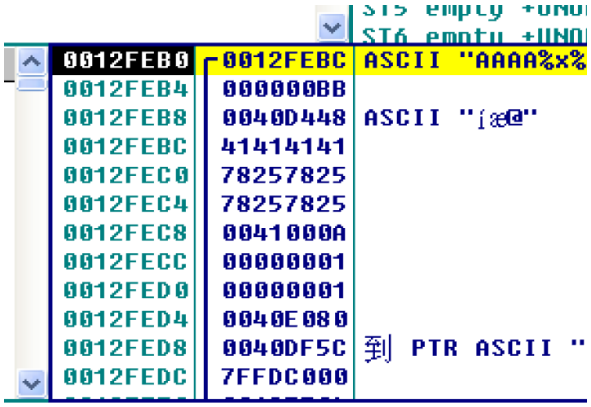


由于 printf 会输出字符串内容，从而 str 的 AAAA 被输出，而后面的四个%x 被看作格式化参数，由于格式化字符串漏洞的特性，在没有给出实际参数的情况下，会将格式化字符串后的栈内容取出作为参数

从而，对于前面的 3 个%x，分别对于后面的地址，以 16 进制输出了堆栈内容：0x0012d9dc，0x0012adb7，0x0ffd9000…，（省略前置 0）而最后到了初始化的“CCCCCCCC”部分，从而输出对应地址代表的 CCCCCCCC。

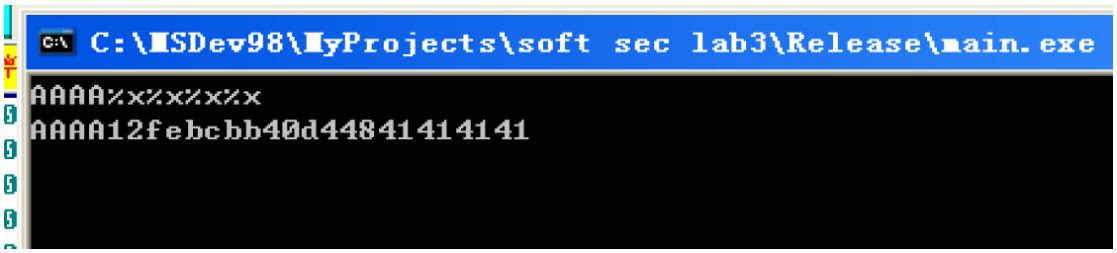
6. 类似地、通过 ollydbg 进行 release 模式下的.exe 文件分析，进入 main 函数：

可以发现，与 debug 模式不同，在一开始没有 push ebp 操作，只有 sub esp 0C8，即将栈指针减去 0C8，刚好预留出 str 需要的 200 字节大小，从而其堆栈内内容如下：



在栈顶的下面，即为 41414141，没有重复的 0x108 字节的 CCCCCCCC

7. 类似地，对其进行 fgets 函数和 printf 函数的调试，原理同上，获取输出如图所示：



分析如下：对于 AAAA：正常输出

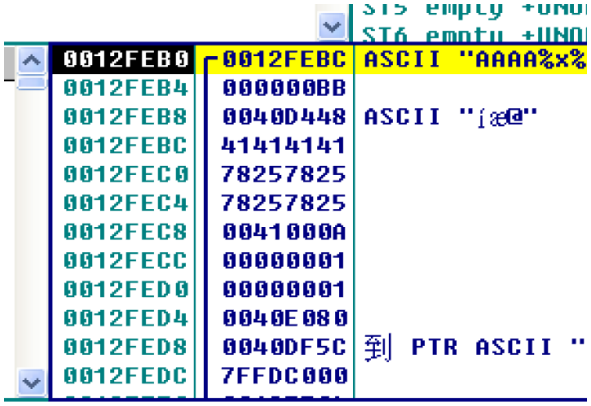
第一个%x:输出帧顶部的第一个地址处的 16 进制值:0x0012FEBC(省略前置 0)

第二个%x：输出下一个地址 16 进制值:0x000000BB

第三个%x：输出下一个地址 16 进制值:0x0040D448

第四个%x：输出下一个地址 16 进制值:0x41414141（AAAA）

从而，由于开辟 main 函数帧空间时对于空间申请大小的不同导致 str 的 AAAA 存储在不同的位置，在利用格式化字符串漏洞进行数据获取时候，会造成不同的输出。



心得体会：

通过实验，了解、学习了 release 模式和 debug 模式二者的异同，进行了一番课外学习。

同时，使用 VC6 对程序进行了几次调试，中间出现了一点问题（断点方面），进行了一些学习，对于 VC6 对调试界面有了更好的理解。

随后放弃 VC6，使用 ollydbg 进行调试，对于寻找 main 函数有些忘了，通过视频学习和知识复习，巩固了 ollydbg 的用法。

最后，亲身实践了课上学习的格式化字符串漏洞，了解了其利用方式和可能影响、实际操作，对于其有了更全面的理解。