**《软件安全》实验报告**

姓名：禹相祐 学号：2312900 班级： 计科

**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。实验代码如下所示：

#include <stdio.h>

 int main(int argc, char \*argv[])

 {

 char str[200];

 fgets(str,200,stdin);

 printf(str);

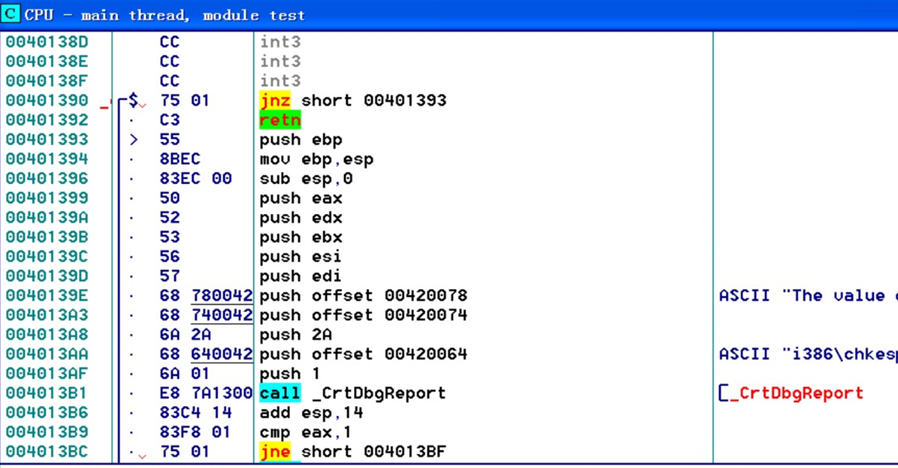
 return 0;

 }

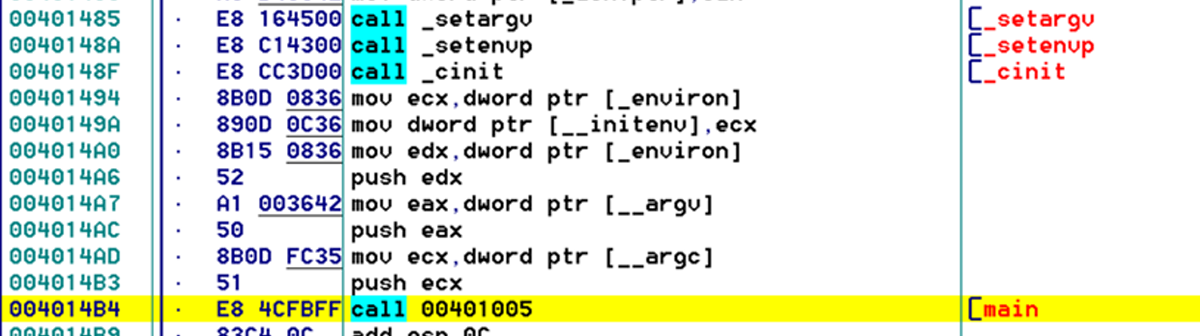
**实验过程：**

一.DEBUG模式

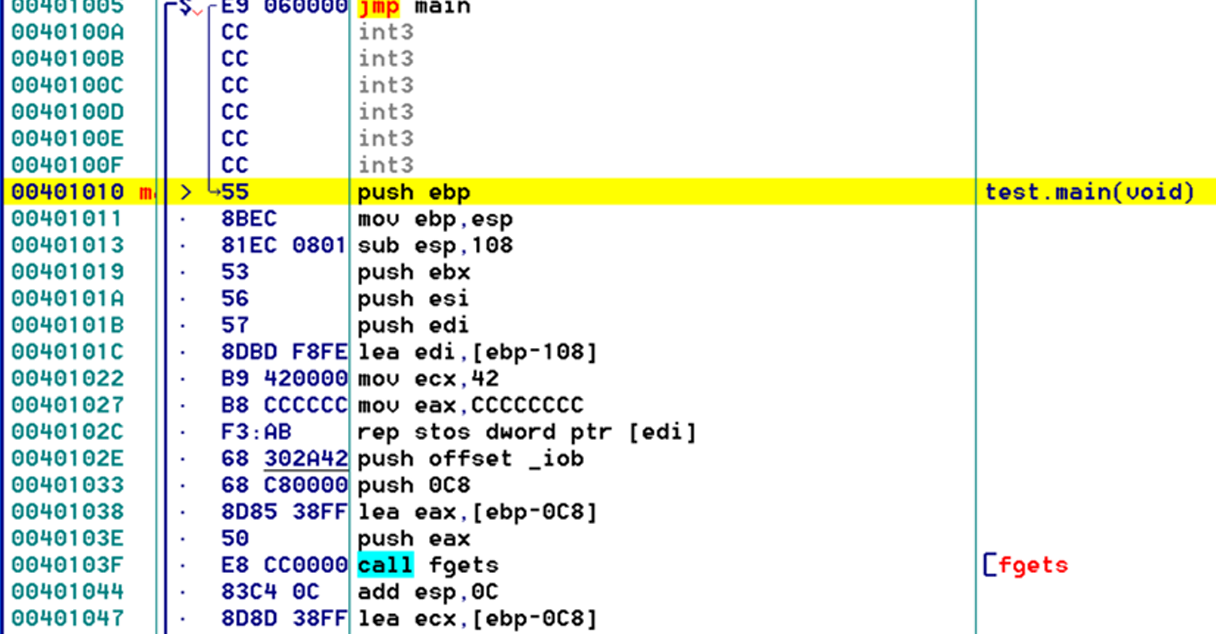
1. 首先将该段代码输入进VC6.0,进行debug的调试，并生成可执行文件exe，再将该exe拖入进ollydbg进行调试，如图：

****

1. 我们进行查找，直至找到三个mov指令，再往下找到call语句:call 00401005,如图：

****

然后再按下F7进入main函数，如图：

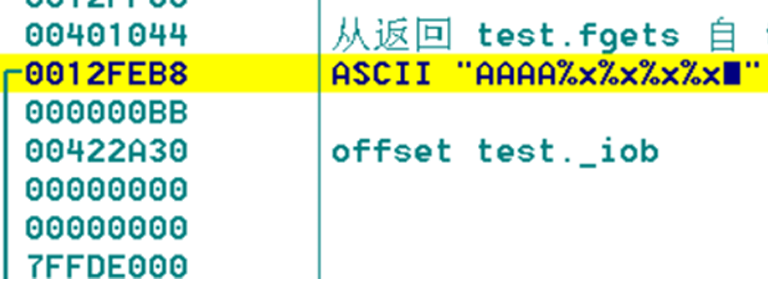


由图可知：从push ebp到push edi这六句其实就实现了将ebp压入，并且更新ebp 和 esp，并且留下了108的空间，紧接着压入了三个寄存器:ebx,esi,edi。

后面的四句就是将开辟的空间全部赋值为CCCCCCCCh。

1. 接下来就是实现fgets（）的调用：

我们直接输入AAAA%x%x%x%x：

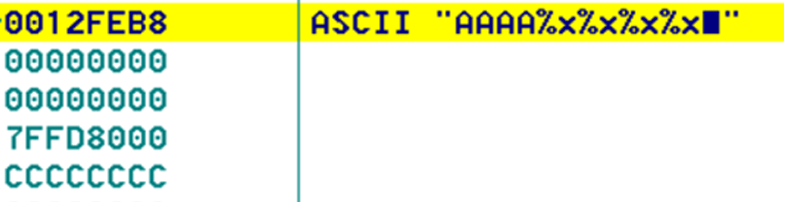


1. 接下来执行print（）：

得到输出:AAAA007ffd8000cccccccc

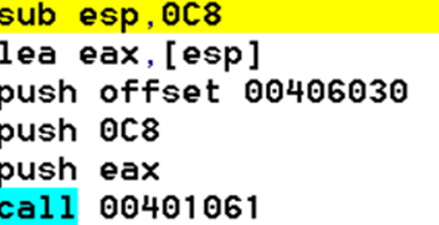
至此完成，接下来分析为什么会出现这个结果：

由栈中内容可知：



这四个%x分别对应的就是后面的00000000、00000000、7FFD8000、CCCCCCCC，与输出对应，分析结束。

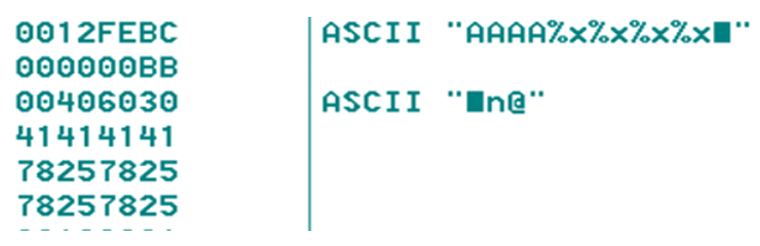
1. RELEASE 模式
2. 在VC6执行时从debug切换为release，然后再次将生成的可执行文件导入ollydbg中，通过三个push定位，然后按着F7进入主函数，依然得到如下：



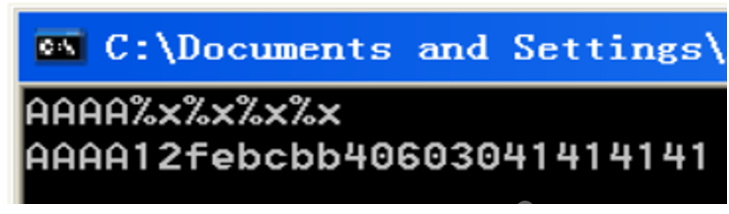
1. 对上面的指令进行分析：

Sub esp,0C8 就是为输入的str留下了200的空间，后面通过三个push将fegts()的参数入栈。

1. 接着执行fgets()并输入AAAA%x%x%x%x ：

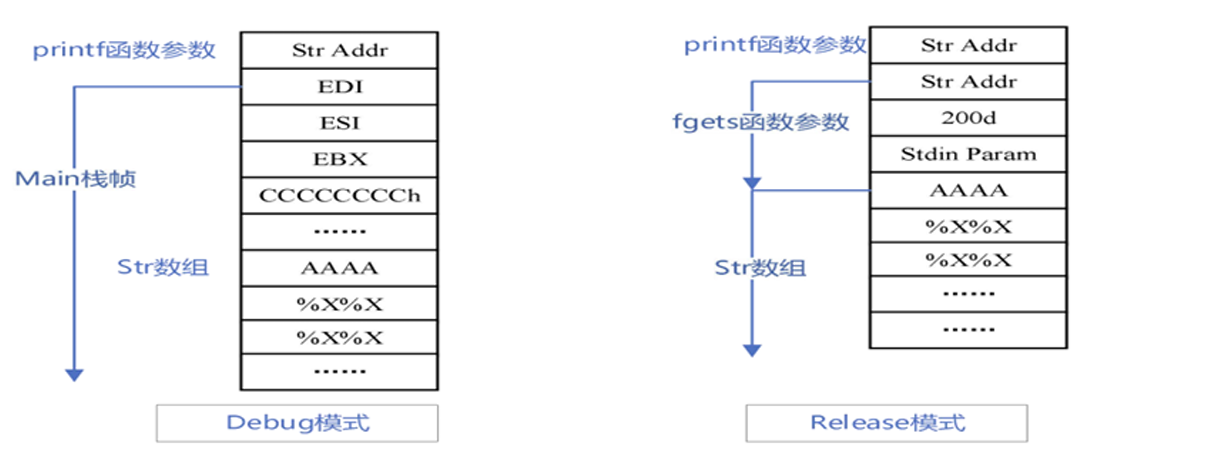


1. 接着执行print():



输出为此原因同上。

1. RELEASE 和 DEBUG 的比较：



可以从上图看出：

对于DEBUG 模式，其一开始会分配更大的空间（108），导致如果想读到输入字符串str的地址，需要更多的“%x”;

而对于RELEASE 模式，其不会有像DEBUG 模式那样的push ebp,mov ebp,esp，会在程序最后完成栈帧的改变。

**心得体会：**

通过此次实验，我通过动手更加深了我对于格式化字符溢出这个漏洞的理解，且也开始对Release 和 Debug 模式的差别有了一定的认识。

另外，上课老师所讲到的像是SQL Injection 这样的漏洞，实现不是特别困难，但引起的问题却非常的大。而且课上还有很多别的格式化字符一样能引起这样的漏洞，以后编程时需要对字符串的输入作一定规范，尽量减少这样情况的发生。