实验三 栈缓冲区溢出

# 实验目的

掌握栈缓冲区溢出原理；

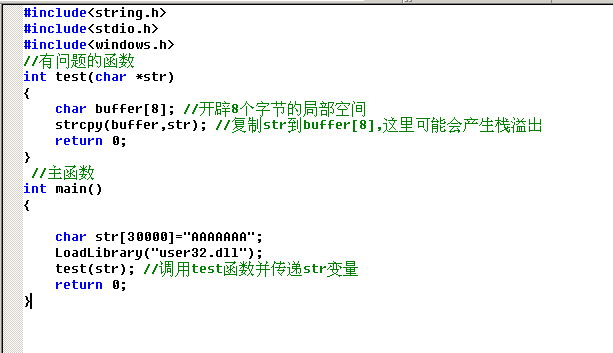
# 实验环境

吾爱破解虚拟机 WinXP\_52Pojie\_2.0、Microsoft Visual C++6.0（在虚拟机里安装）、Ollydbg

# 实验内容

注：由于实验环境、代码编写的不同，使用Ollydbg反汇编出来的指令的实际地址可能与本指导的地址有所差异，请按实际情况填写，并给出必要的截图。

1. 使用VC++6.0编写一段C程序，源代码如下：



其中，strcpy函数是一个不安全函数，无字符串长度检查，执行该函数可能会产生栈溢出。

1. 编译程序，生成Debug版本的EXE文件，使用Ollydbg打开EXE，程序断在入口点，首先寻找call main指令，向下查找汇编代码，在401344处找到一个call指令（实际情况可能不同，如图1），跟踪步入，到达401070处（如图2），从该地址开始main的执行。

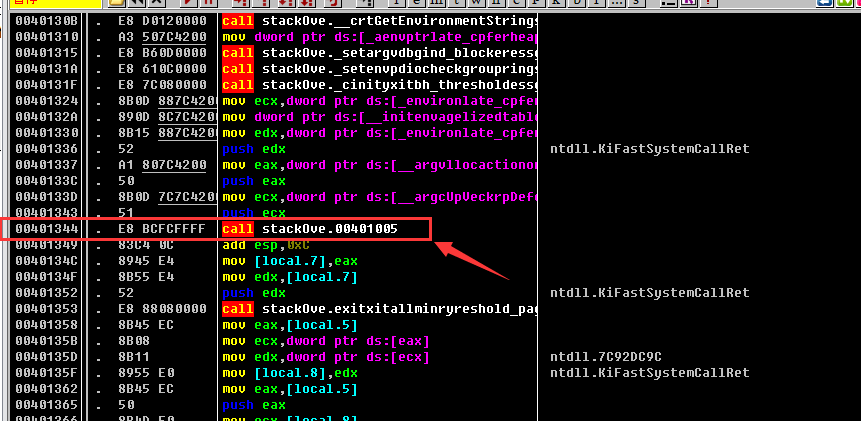


图1 call main

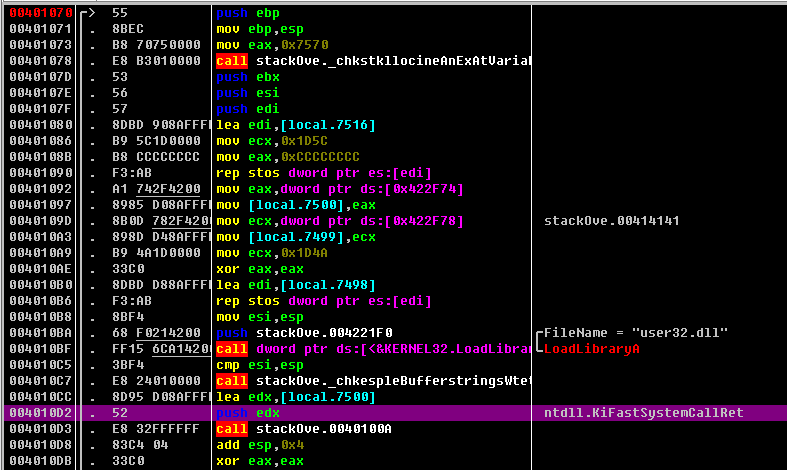


图2 main开始

(3)4010D2处令edx入栈，4010D3使用call指令调用test函数，因此edx寄存器的是test函数的参数，即

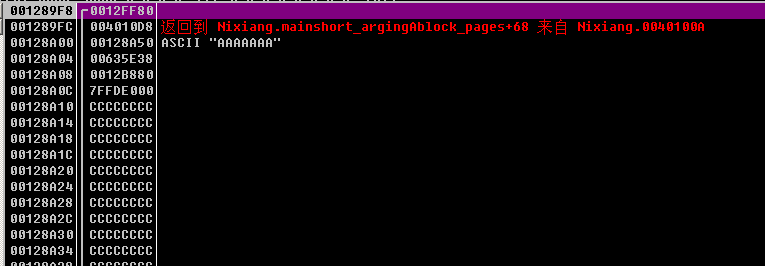
$edx=0x00128A50，

该寄存器代表变量名 str 的地址，变量值为 AAAAAAA。

(4)跟踪步入call 0040100A指令，进入test函数，给出test开始执行时的截图（从push ebp指令开始）。

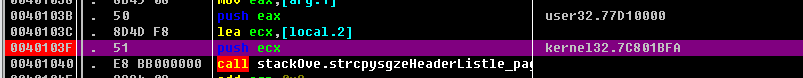


(5)子程序首先执行push ebp;mov ebp,esp;sub esp,0x48，为变量buffer分配一定的内存空间，此时函数栈帧结构已经完成，此时右击寄存器窗口的EBP寄存器，点击“堆栈窗口中跟随”，在右下角堆栈窗口中可以观察到栈底的情况，给出堆栈窗口截图：



根据截图得到test函数的返回地址为\_0x004010D8\_\_\_\_\_\_\_，此地址为(4)步入的call指令的下一条指令的地址。

(6)观察0040103B处的指令如下：



00401040处的call指令调用strcpy将变量str的字符串拷贝到变量buffer中，分析这四条指令，其中，

ecx=\_001289F0，代表的是变量\_buffer\_\_\_\_\_\_\_，

eax=\_00128A50，代表的是变量\_\_str\_\_\_\_\_\_。

(7)经过以上分析，如果str的字符串过长，那么在执行test函数调用strcpy会使字符串超过为buffer分配的空间，将可能覆盖(5)堆栈的返回地址，修改C程序的代码char str[30000]=”AAAAAAAAAAAAAAAAAAAA”，令程序执行完(6)的四条指令（可在401045处下断点令程序执行到此处），观察堆栈情况，发现返回地址变成\_0x004010D7\_\_\_\_\_\_\_，由此，test函数的返回地址被过长的字符串覆盖。由(6)得知buffer的地址为\_001289F0\_\_\_\_\_\_\_，而堆栈中存储返回地址的位置为\_001289FC\_\_\_\_\_\_\_，二者相差12字节，因此如果src的字符串长度超过12字节，如果字符串足够长，src的第13~16个字符会覆盖函数的返回地址。

