软件安全实验报告

姓名：郭子涵 学号：2312145 班级：信息安全、法学双学位班

1 实验名称：

格式化字符串漏洞-任意地址数据获取

2 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。

3 实验过程：

以下述代码为例，观察Release模式和Debug模式的差异:

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    char str[200];
    fgets(str,200,stdin);
    printf(str);
    return 0;
}
```

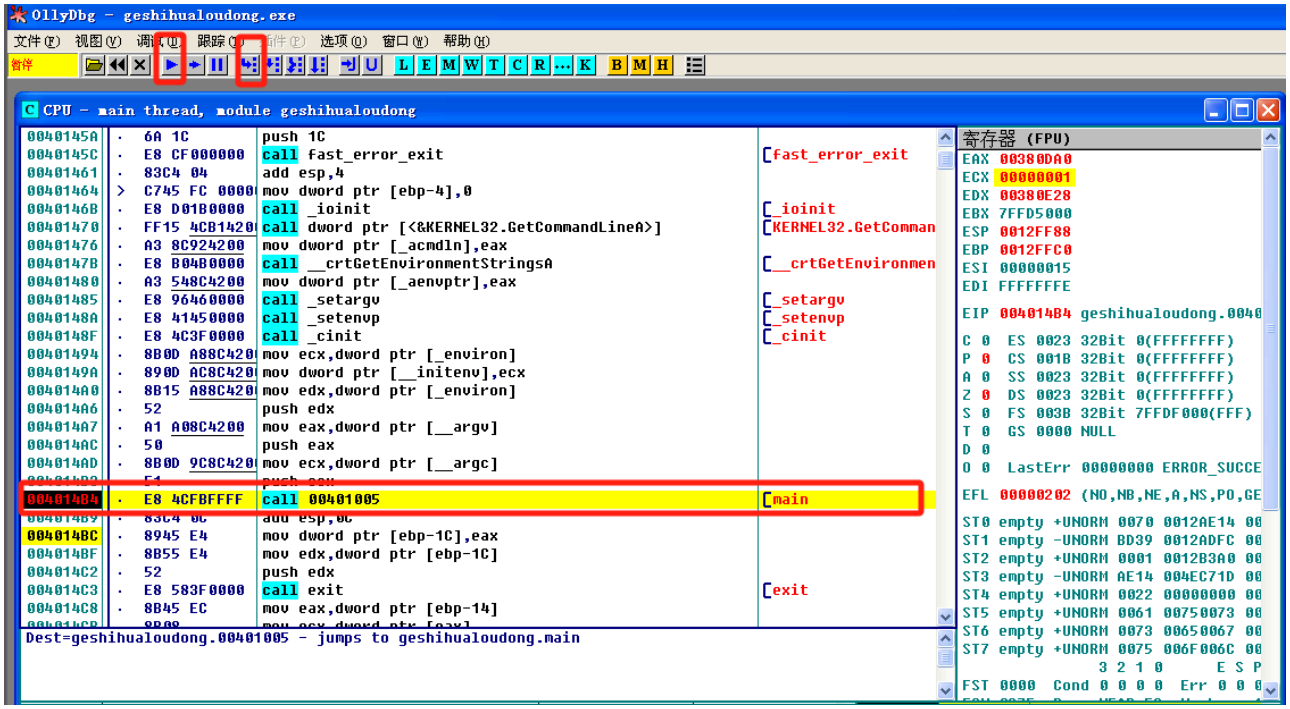
3.1 Debug模式：

在Vc6中输入源代码，进行Debug模式的调试，并将exe文件导入到ollydug中进行调试：

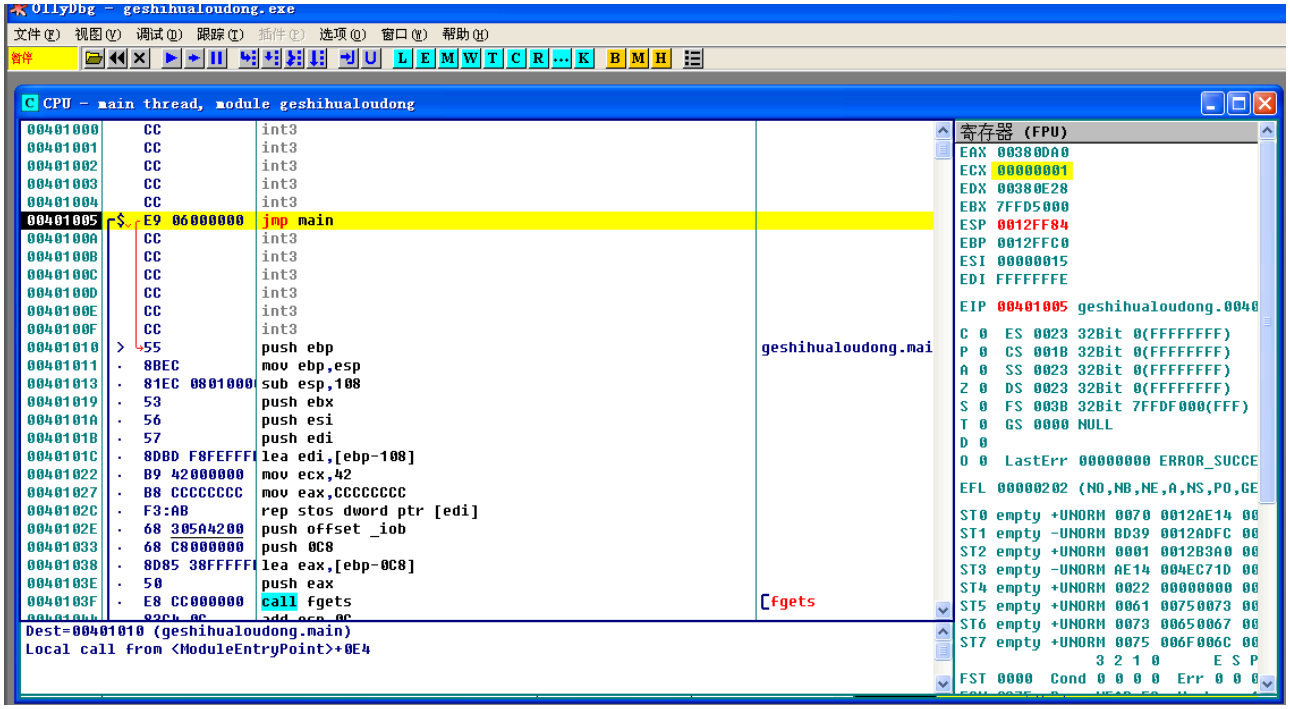
```
00401005  /$ /E9 06000000   jmp main
00401010  |> \55            push ebp                                        ;
geshihualoudong.main(void)
00401011  |.  8BEC          mov ebp,esp
00401013  |.  81EC 08010000 sub esp,108 ;调整栈帧，分配局部变量空间
00401019  |.  53            push ebx
0040101A  |.  56            push esi
0040101B  |.  57            push edi
0040101C  |.  8DBD F8FEFFFF lea edi,[ebp-108]
00401022  |.  B9 42000000   mov ecx,42
00401027  |.  B8 CCCCCCCC   mov eax,CCCCCCCC
0040102C  |.  F3:AB         rep stos dword ptr [edi] ;保存寄存器，防止被修改，初始化局部变
量区域，填充值是0xcccccccc，
0040102E  |.  68 305A4200   push offset _iob
00401033  |.  68 C8000000   push 0C8
00401038  |.  8D85 38FFFFFF lea eax,[ebp-0C8]
0040103E  |.  50            push eax
```

```
0040103F  |.  E8 CC000000   call fgets  ; [fgets: 调用fgets(buffer,0xc8,_iob),从标准输入读取最多200字节内容到缓冲区中
00401044  |.  83C4 0C       add esp,0C
00401047  |.  8D8D 38FFFFFF lea ecx,[ebp-0C8]
0040104D  |.  51            push ecx
0040104E  |.  E8 3D000000   call printf ;[printf;直接把用户输入的内容打印出来
00401053  |.  83C4 04       add esp,4
00401056  |.  33C0          xor eax,eax
00401058  |.  5F            pop edi
00401059  |.  5E            pop esi
0040105A  |.  5B            pop ebx
0040105B  |.  81C4 08010000 add esp,108
00401061  |.  3BEC          cmp ebp,esp
00401063  |.  E8 28030000   call _chkesp
00401068  |.  8BE5          mov esp,ebp
0040106A  |.  5D            pop ebp
0040106B  \.  C3            retn;把寄存器还原、释放栈空间、做栈完全性检查(_chkesp)最后ret返回
```

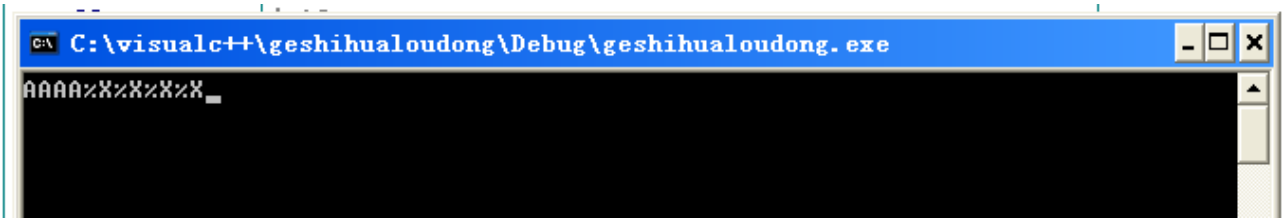
首先找到main函数，通过F2设置断点，点击运行到此处：

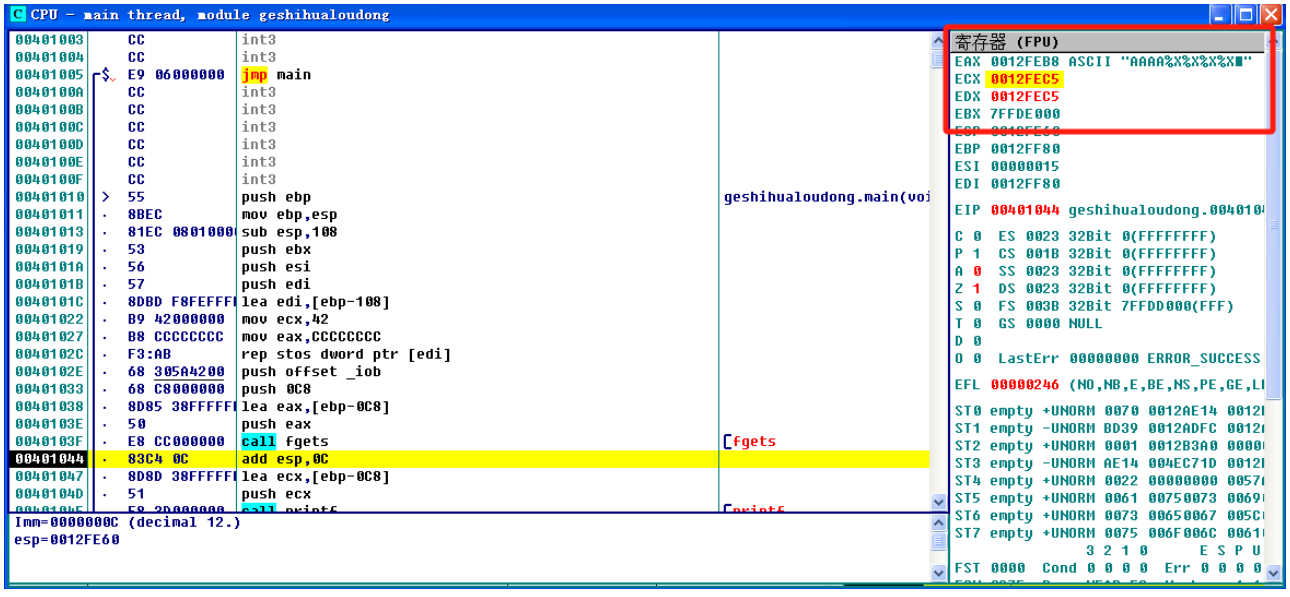


执行单步步入进入主函数：

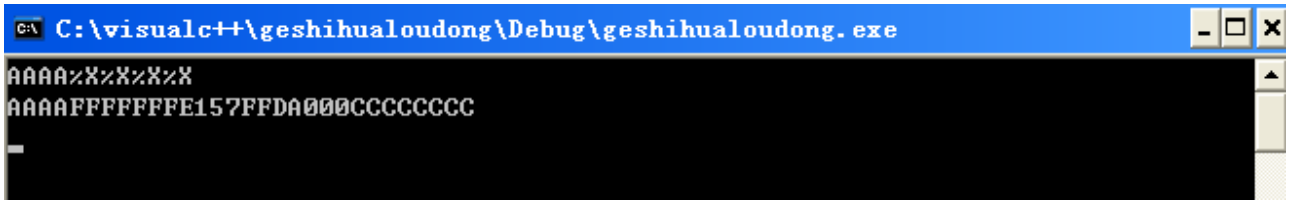


按F8至fgets函数，输入字符串AAAA%X%X%X%X：





执行至printf处，查看输出如下，四个%x会自动读取后面的地址作为输出：

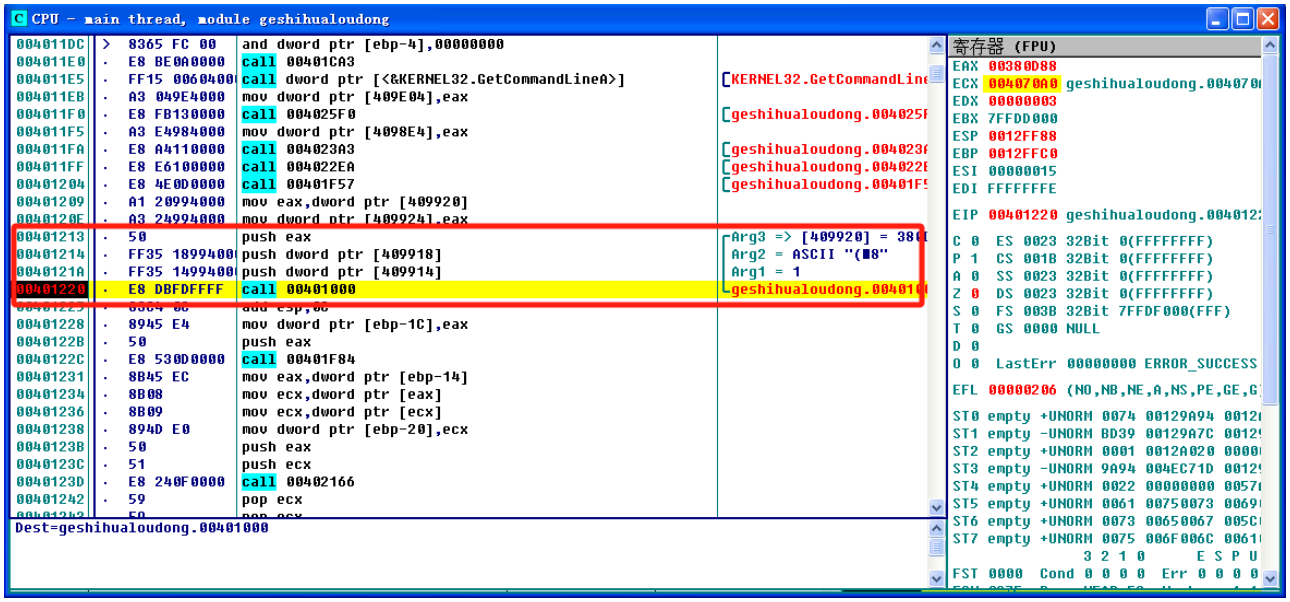


3.2 release模式：

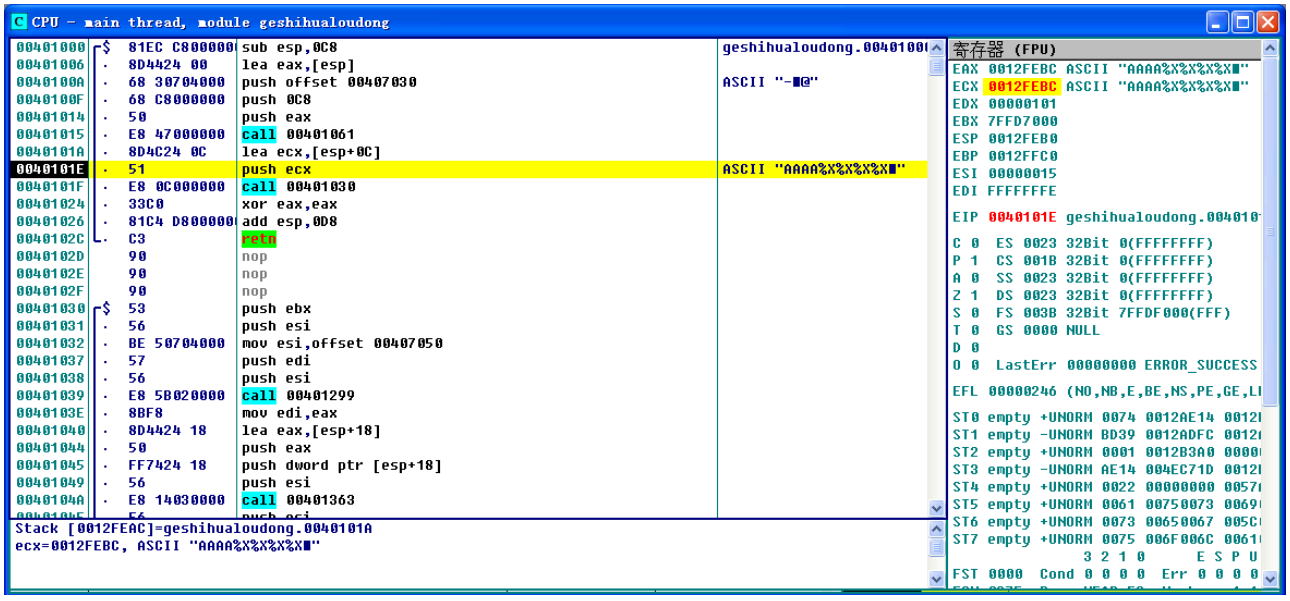
VC6中切换`Windows Release`模式，同样编译后导出exe文件导入ollydbg调试：

```
00401000  /$  81EC C8000000 sub esp,0C8        ;给栈分配200字节的局部变量空间；无过多的栈内存空间分配，无寄存器保存入栈操作
; geshihualoudong.00401000(guessed Arg1,Arg2,Arg3)
00401006  |.  8D4424 00     lea eax,[esp]     ;将当前栈顶地址放入eax中
0040100A  |.  68 30704000   push offset 00407030                              ; ASCII "-•@"
0040100F  |.  68 C8000000   push 0C8
00401014  |.  50            push eax;传入函数所需参数：buffer,0xc8,00407030
00401015  |.  E8 47000000   call 00401061;调用的输入函数
0040101A  |.  8D4C24 0C     lea ecx,[esp+0C]
0040101E  |.  51            push ecx
0040101F  |.  E8 0C000000   call 00401030;调用打印函数
00401024  |.  33C0          xor eax,eax
00401026  |.  81C4 D8000000 add esp,0D8
0040102C  \.  C3            retn;函数返回，释放栈空间，标准的清理和返回过程
```

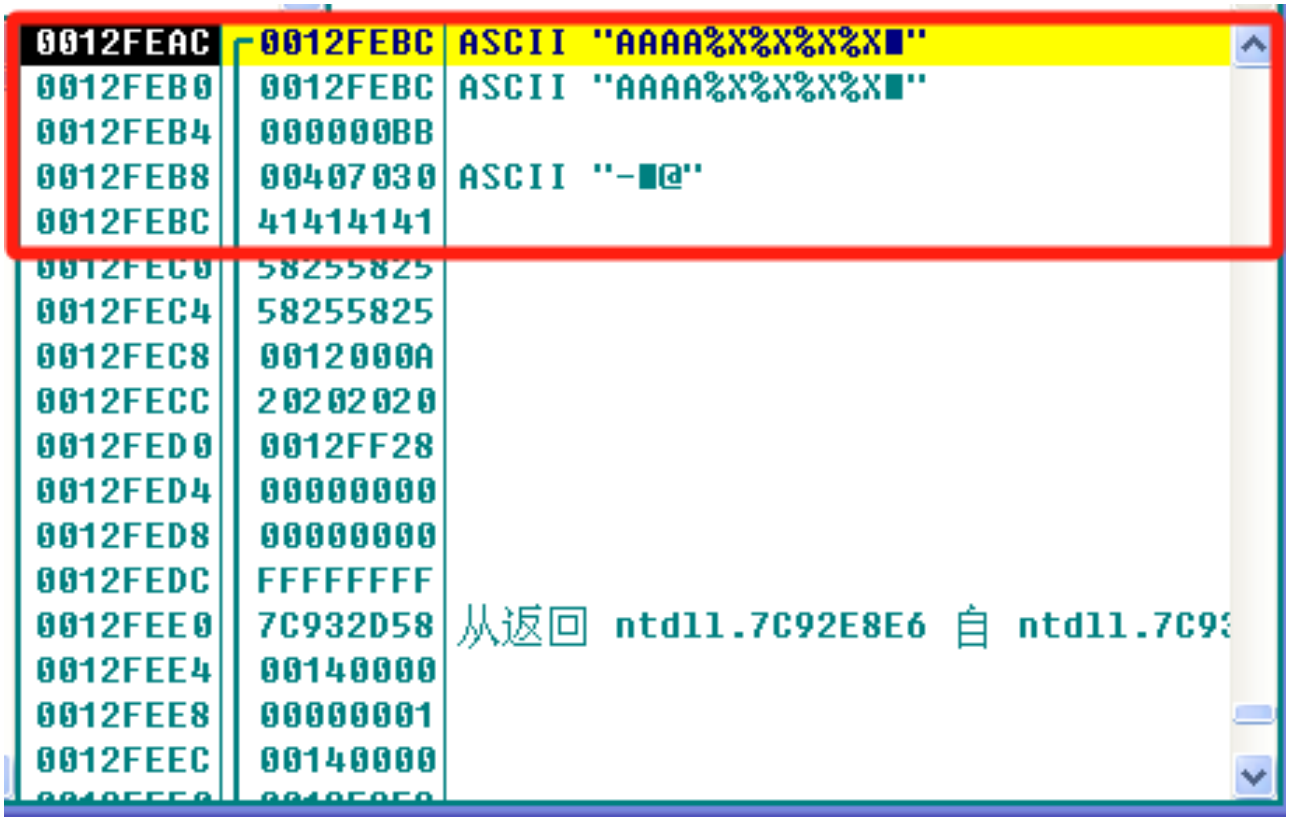
同样通过三个`push`找到主函数入口，按`F2`设置断点：

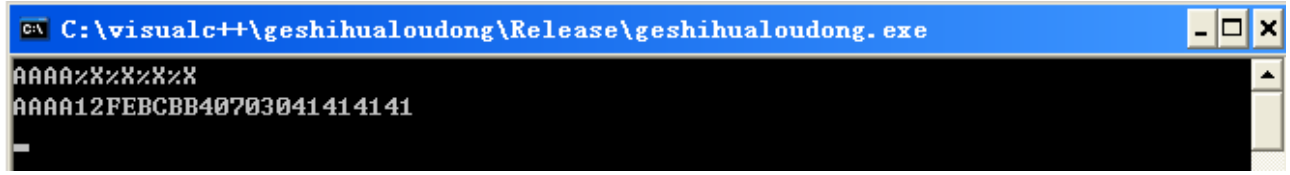


执行运行后，单步步入main函数：

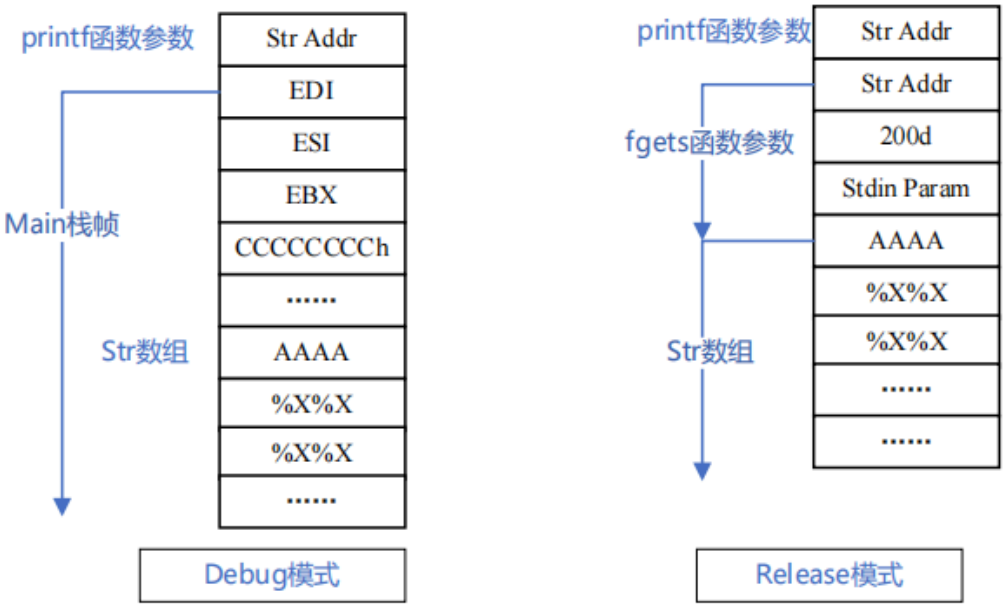


同样按F8执行输入函数，执行输出函数，四个%X格式化草索夫读取后四行的内容作为参数输出，结果如图所示实现任意地址数据读取：





4 Debug vs Release 模式对比分析



特性	Debug 模式	Release 模式
函数结构	明确可见 main() 函数结构，带调试标志	优化重排，可能内联或重命名，无符号
变量填充	明确清空局部变量（mov eax, 0xCCCCCCCC, rep stosd）	没有变量初始化，直接使用栈
栈空间	分配 108h 字节（264 字节），对齐+冗余	精准分配 0xC8（200 字节），无冗余
保护措施	有 _chkesp 调用、防止栈错乱	无多余保护逻辑
可读性	更易识别、反汇编能还原较多信息	结构紧凑，函数可能被合并或优化
调试指令	有 int3（0xCC）填充	无调试指令
函数调用参数	使用局部变量栈传参	直接用 esp、压栈调用
返回值处理	明确 xor eax, eax 设置返回值	相同处理，但更靠近尾部逻辑
清栈方式	调用函数后明确 add esp, N 清栈	同样处理，但更简洁

5 心得体会:

本次实验基于同一包含潜在安全隐患的 C 代码，分别在 Debug 和 Release 模式下进行编译与反汇编分析，并尝试触发其中的格式化字符串漏洞，通过对比两种模式下程序行为的异同，深入理解编译优化对安全性的影响。总结以下几点体会：

1. Release 模式为了执行效率大幅压缩代码、去除调试信息，但也失去了很多防御机制，因此发布版本更容易暴露漏洞，特别是在开发者未注意输入校验的情况下。Debug 模式中初始化变量、栈检查等机制虽拖慢运行速度，但在开发测试阶段极其重要，能帮助开发者尽早发现潜在问题。
2. 格式化字符串漏洞的破坏力显著：尤其在 Release 模式下，如果攻击者能控制输入，轻则数据泄露，重则远程代码执行，应始终避免未加格式符的 printf 风格函数调用。
3. 通过反汇编分析，我更加理解了编译器对函数调用、变量管理、栈布局等的处理机制，也体会到 Release 模式对程序控制流的高度优化对逆向与漏洞利用分析的重要影响。