软件安全实验4

姓名：何叶 学号：2313487 班级：范玲玲班

一、实验名称

格式化字符串漏洞

二、实验内容

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。

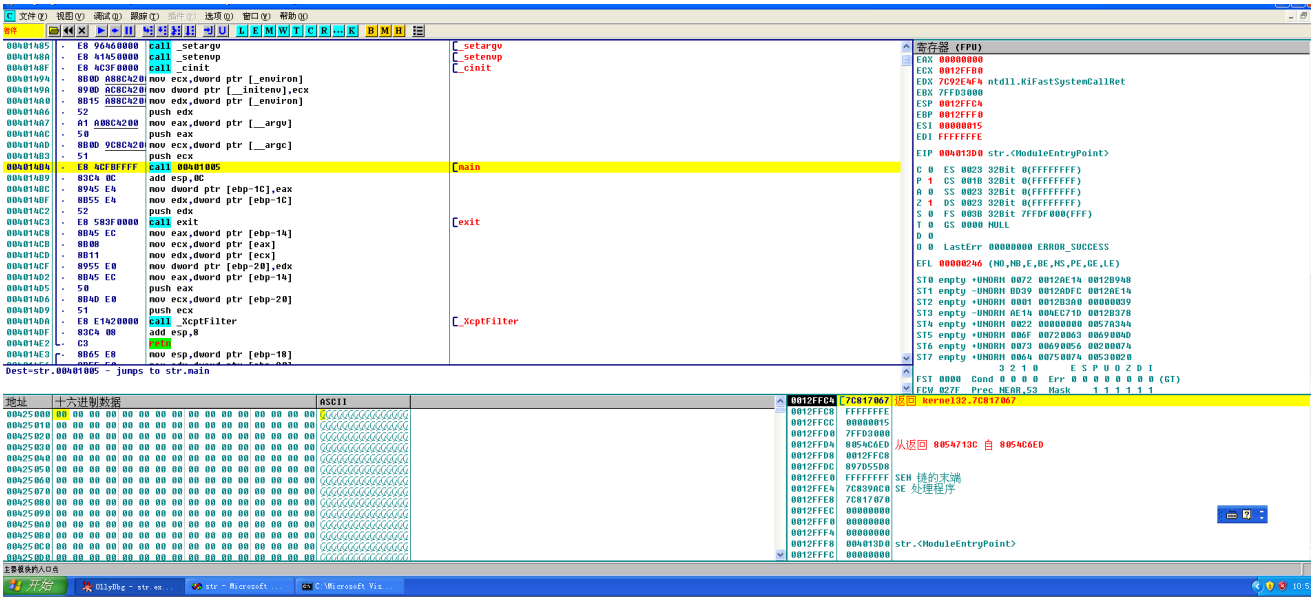
三、实验过程

1.实验源码：

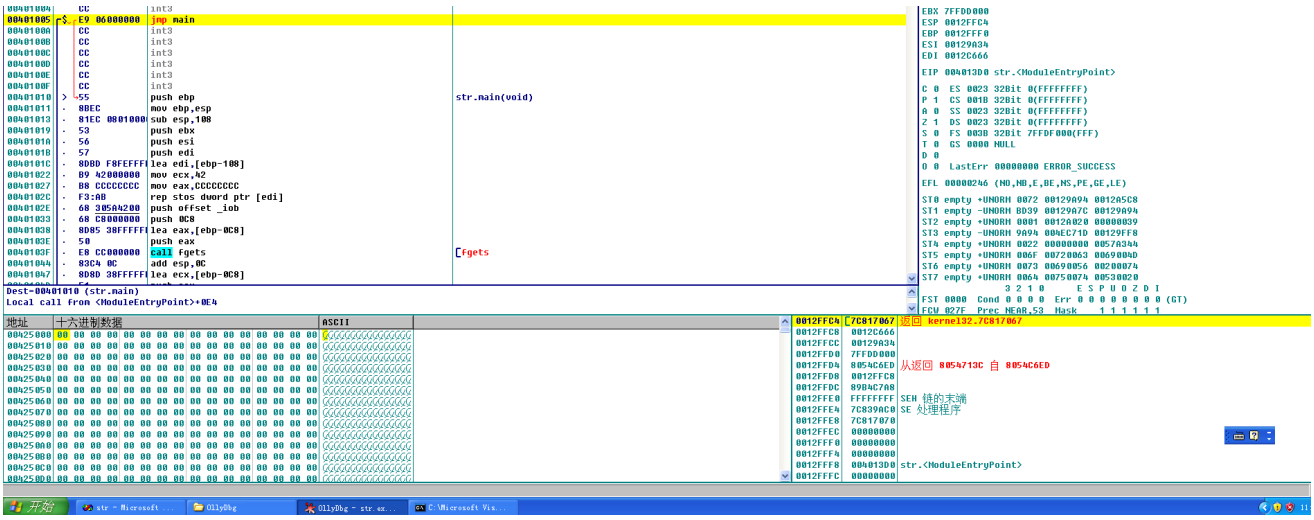
```
#include <stdio.h>
int main(int argc, char *argv[])
{
    char str[200];
    fgets(str,200,stdin);
    printf(str);
    return 0;
}
```

2.debug模式进入

2.1从ollydbg进入



2.2从main函数进入调试



```
0040100E    CC              int3
0040100F    CC              int3
00401010  > 55              push ebp                        str.main(void)
00401011  . 8BEC            mov ebp,esp
00401013  . 81EC 0801000(   sub esp,108
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]
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
00401044  . 83C4 0C         add esp,0C
00401047  . 8D8D 38FFFFFf   lea ecx,[ebp-0C8]
0040104D  . 51              push ecx
0040104E  . E8 3D000000     call printf                     [printf
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 0801000(   add esp,108
00401061  . 3BEC            cmp ebp,esp
00401063  . E8 28030000     call _chkesp
```

2.3理解汇编语句作用

push ebp//将ebp压入栈

mov ebp,esp//实现栈顶与栈底的转换

sub esp ,108//给了108的空间

push ebx//压入寄存器

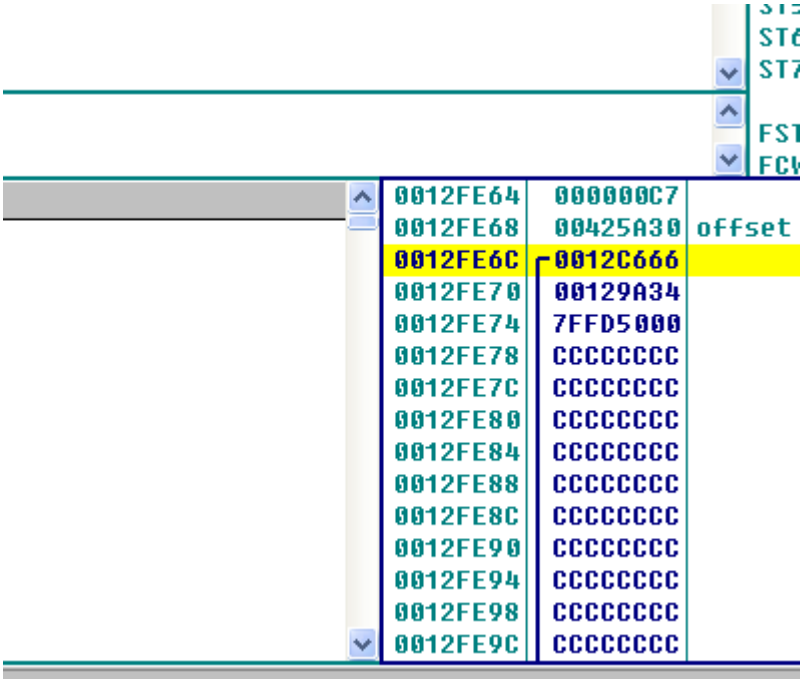
push esi//压入寄存器

push edi//压入寄存器

lea edi,[ebp-108]//取地址

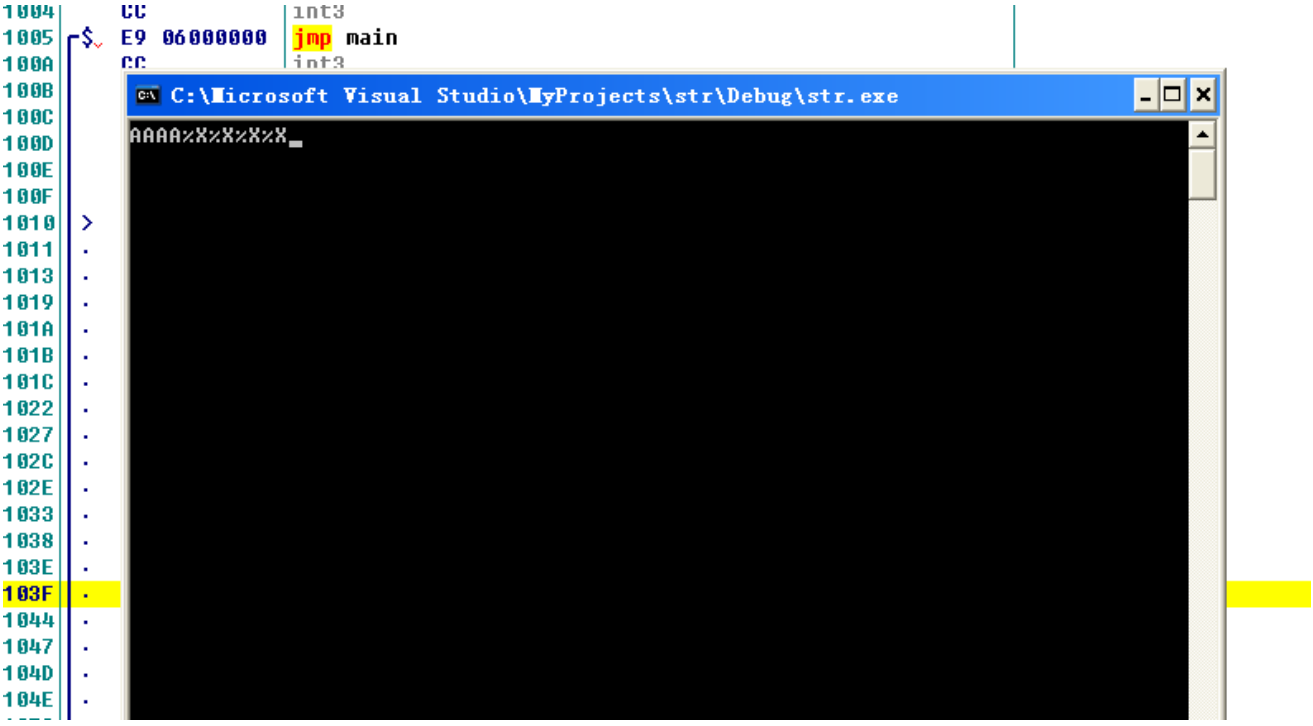
mov eax,42//

mov eax,cccccccc//将给的空间中全部赋值为CCCCCCCC

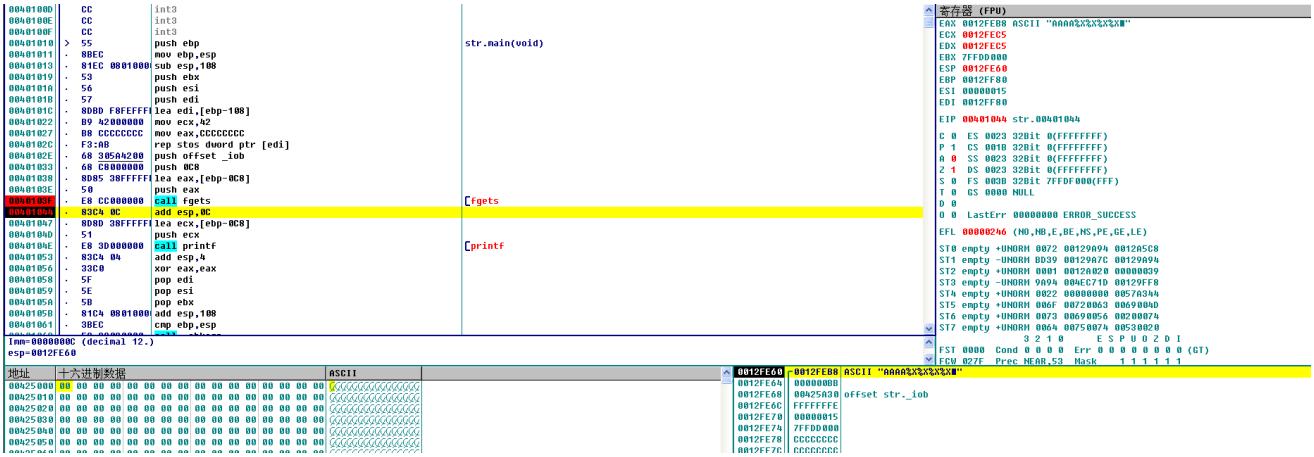


观察到空间全部赋值为CCCCCCCC

call fget//调用fget函数



输入AAAA%X%X%X%X:



可以看到已经将AAAA%X%X%X%X压入

add esp,0c//清除fget的栈帧

lea exc,[ebp-0c8]//将str的起始地址保存在ecx

call printf//执行printf函数

返回：AAAABB425A30FFFFFFFE15

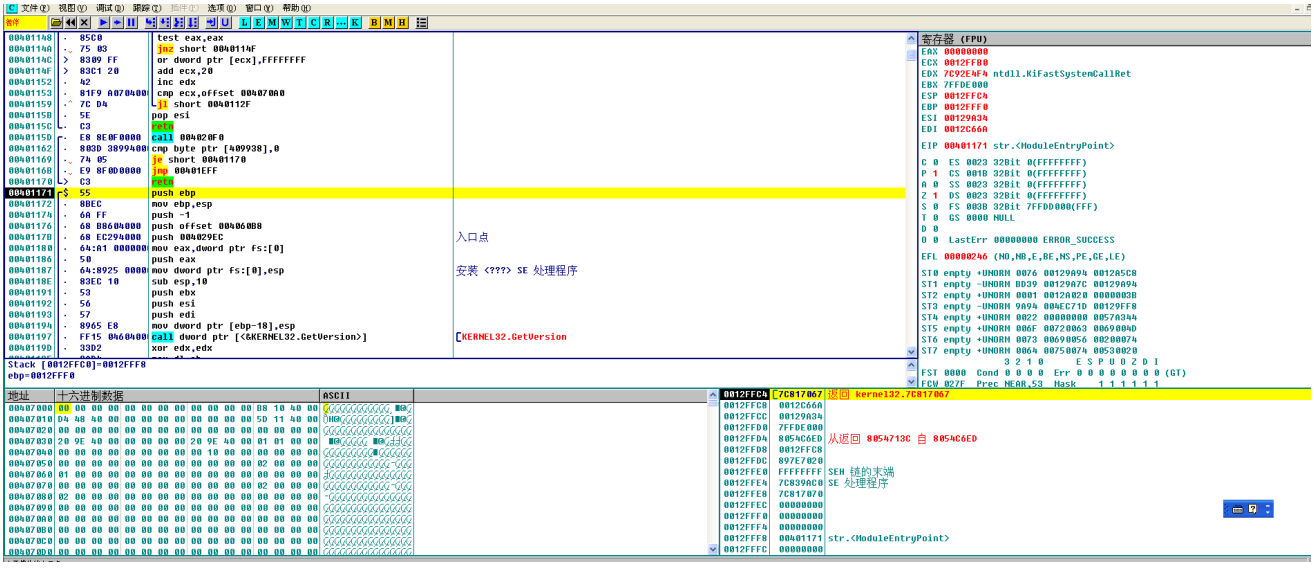
解释：

printf函数正常输出AAAA，遇到%X时，读取并输出后面的地址，即000000BB,00425A30,FFFFFFFE,00000015;

输出为：AAAABB425A30FFFFFFFE15

3.从release模式进入

3.1从ollydbg进入



观察到分配紧凑，代码紧靠

3.2找到main函数入口

```
0040120E   .  A3 24994000   mov dword ptr [409924],eax
00401213   .  50            push eax                          ; Arg3 => [409920] = 380BA8
00401214   .  FF35 1899400  push dword ptr [409918]           ; Arg2 = 380B40
0040121A   .  FF35 1499400  push dword ptr [409914]           ; Arg1 = 1
00401220   .  E8 DBFDFFFF   call 00401000                     ; str.00401000
00401225   .  83C4 0C       add esp,0C
```

3.3进入主函数

```
00401000  /$  81EC C800000  sub esp,0C8                       ; str.00401000(guessed Arg1,Arg2,Arg3)
00401006  |.  8D4424 00     lea eax,[esp]
0040100A  |.  68 30704000   push offset 00407030              ; ASCII " ■@"
0040100F  |.  68 C8000000   push 0C8
00401014  |.  50            push eax
00401015  |.  E8 47000000   call 00401061
0040101A  |.  8D4C24 0C     lea ecx,[esp+0C]
0040101E  |.  51            push ecx
0040101F  |.  E8 0C000000   call 00401030
00401024  |.  33C0          xor eax,eax
00401026  |.  81C4 D800000  add esp,0D8
0040102C  \.  C3            retn
0040102D      90            nop
0040102E      90            nop
0040102F      90            nop
00401030  /$  53            push ebx
00401031  |.  56            push esi
00401032  |.  BE 50704000   mov esi,offset 00407050
00401037  |.  57            push edi
```

观察到没有ebp入栈

sub esp,0C8//抬高200字节，仅仅给局部变量分配了空间

没有debug中一堆push寄存器的值

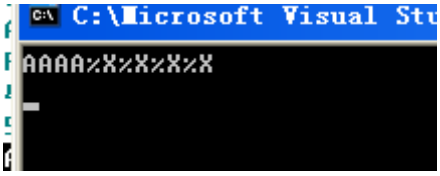
lea eax,[esp]//esp的地址给eax

push offset 00407030//参数入栈调用

push 0c8//参数入栈调用

push eax//参数入栈调用

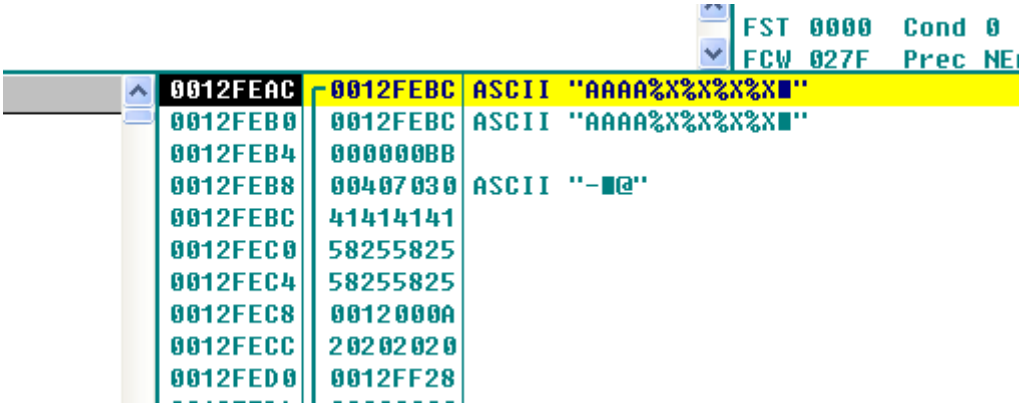
3.4进入fgets函数



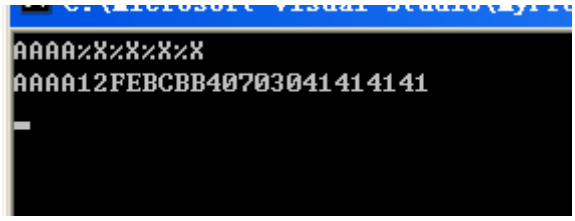
输入AAAA%X%X%X%X

```
00401014   .  50            push eax
00401015   .  E8 47000000   call 00401061
0040101A   .  8D4C24 0C     lea ecx,[esp+0C]
0040101E   .  51            push ecx                          ; ASCII "AAAA%X%X%X%X■"
0040101F   .  E8 0C000000   call 00401030
00401024   .  33C0          xor eax,eax
00401026   .  81C4 D800000  add esp,0D8
0040102C   .  C3            retn
```

观察到参数入栈



3.5printf函数



输出结果为AAAA12FEBCBB40703041414141

先输出AAAA，四个%X输出后四行的地址值，即0012FEBC,000000BB,00407030,41414141

4.debug模式与release模式的区别

4.1debug模式

main函数分配更大的栈空间，从EBP附近位置分配空间

4.2release模式

代码更加紧凑，简洁，效率更高

进入main函数时，没有严格栈帧转换，不初始化栈空间，没有 push eax来保存寄存器的值，程序最后add esp ,0D8来恢复栈帧

4.3总结

Debug模式主要用于开发和调试，包含调试信息，支持断点调试，代码优化少，运行速度慢，资源占用多，生成的代码可读性强，便于查找和修复问题。Release模式用于发布最终产品，不包含调试信息，不支持断点调试，代码优化程度高，运行速度快，资源占用少，生成的代码复杂度高，适合实际运行环境。

维度	Debug 模式	Release 模式
优化	无优化，便于调试	高度优化，提升性能
调试信息	包含完整调试信息	通常不包含或少量调试信息
性能	性能较低	性能最佳
文件大小	文件较大	文件较小
断言	启用断言和检查功能	禁用断言，提升性能
用途	开发和调试阶段	最终发布给用户

四、心得体会

从汇编语言的角度分析debug模式和release模式的区别，更加深入了解两种模式不同的应用场景和原因

%X攻击，格式化字符完成任意地址的数据获取，说明我们需要考虑程序的安全，需要给输入加一些限制条件来避免发生这种情况