

《格式化字符串漏洞》实验报告

姓名：田晋宇 学号：2212039 班级：物联网班

实验名称：

格式化字符串漏洞——任意地址的数据获取

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：

格式化字符串输出获取任意地址的数据是在调用格式化函数时，给出了格式化符号串，但没有提供对应参数时，这些函数会将格式化字符串后面的多个栈中的内容取出作为参数，并根据格式化符号将其输出。

Debug 模式：

1. 首先生成 Debug 模式下的可执行文件，在 OllyDbg 中打开，找到主函数的入口，进入。

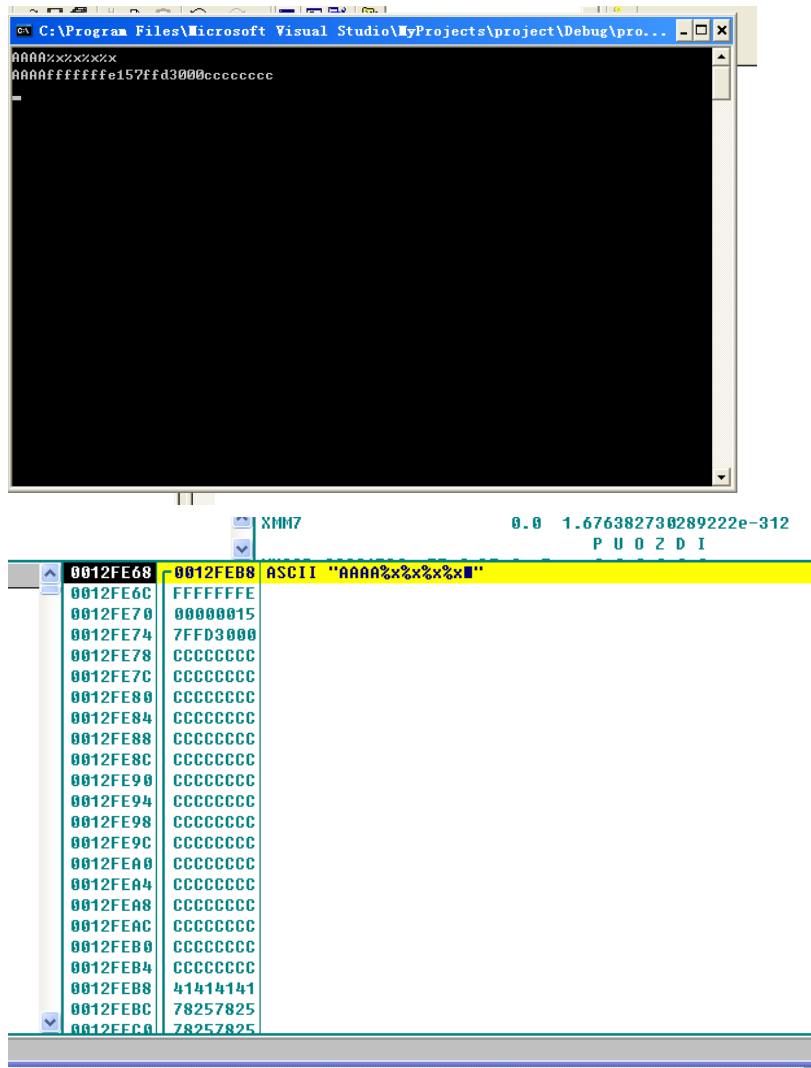
```
00401480 . A3 548C4200 mov dword ptr [_aenvptr],eax
00401485 . E8 96460000 call _setargv
0040148A . E8 41450000 call _setenvp
0040148F . E8 4C3F0000 call _cinit
00401494 . 8B0D A88C4200 mov ecx,dword ptr [_environ]
0040149A . 890D AC8C4200 mov dword ptr [__initenv],ecx
004014A0 . 8B15 A88C4200 mov edx,dword ptr [_environ]
004014A6 . 52 push edx
004014A7 . A1 A08C4200 mov eax,dword ptr [__argv]
004014AC . 50 push eax
004014AD . 8B0D 9C8C4200 mov ecx,dword ptr [__argc]
004014B3 . 51 push ecx
004014B4 . E8 4CFBFFFF call 00401005
004014B9 . 83C4 0C add esp,0C
004014BC . 8945 E4 mov dword ptr [ebp-1C],eax
004014BF . 8B55 E4 mov edx,dword ptr [ebp-1C]
004014C2 . 52 push edx
004014C3 . E8 583F0000 call exit
004014C8 . 8B45 EC mov eax,dword ptr [ebp-14]
004014CB . 8B08 mov ecx,dword ptr [eax]
004014CD . 8B11 mov edx,dword ptr [ecx]
004014CF . 8955 E0 mov dword ptr [ebp-20],edx
004014D2 . 8B45 EC mov eax,dword ptr [ebp-14]
```

Stack [0012FFC0]=0012FFF8
ebp=0012FFF0

2. 进入主函数内部首先是对主函数调用之前的一些寄存器进行保存，并对栈空间进行初始化。可以看到在初始化栈时将栈抬高了 108 的空间，远远超过了字符串初始化的空间。在调用 fgets 前，通过计算将输入字符串的起始地址存入 eax 寄存器中。

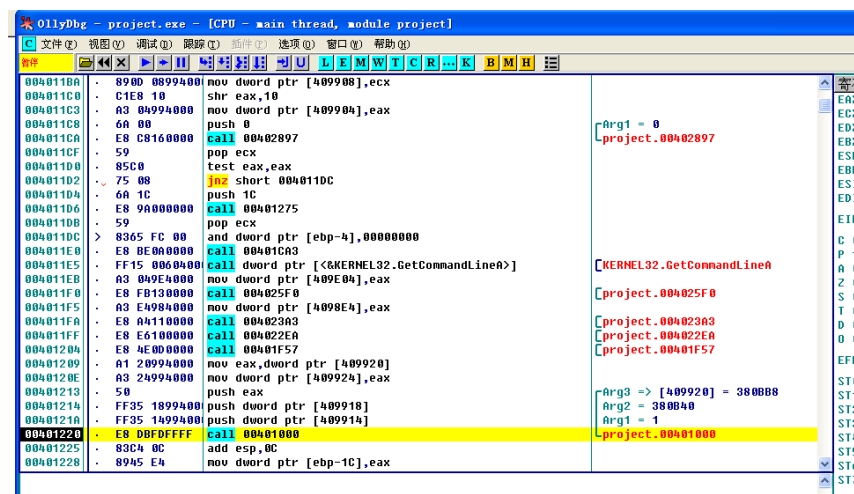
```
00401005 . E9 06000000 jmp main
0040100A . CC int3
0040100B . CC int3
0040100C . CC int3
0040100D . CC int3
0040100E . CC int3
0040100F . CC int3
00401010 . 55 push ebp
00401011 . 8BEC mov ebp,esp
00401013 . 81EC 00010000 sub esp,108
00401019 . 53 push ebx
0040101A . 56 push esi
0040101B . 57 push edi
0040101C . 8DBD F8FEFF 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 C0000000 push 0C8
00401038 . 8DB5 38FFFF lea eax,[ebp-0C8]
0040103E . 50 push eax
```

3. 接着在调试窗口输入格式化字符串“AAAA%x%x%x%x”，单步执行到 printf 函数的时候，调试窗口输出了输入的字符串以及一串地址信息。当“AAAA”输出完成后，函数继续在栈中找下一个参数，即接下来的连续四个地址，发生了内存的泄露。



Release 模式：

1. 生成 Release 模式下的可执行文件，在 OllyDbg 中打开，同样找到主函数的入口。

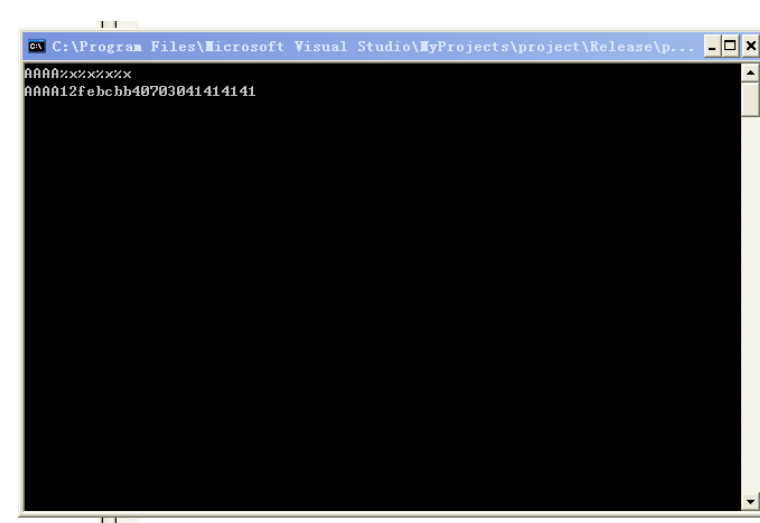


2. 进入主函数内部可以发现对栈空间进行初始化时只抬高了 0C8 的空间即局部变量的大小，也减少了寄存器的 push 指令，提高了程序的执行效率。

```

00401000 81EC C8000001 sub esp, 0C8
00401006 8D4424 00 lea eax, [esp]
0040100A 68 30704000 push offset 00407030
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 D8000000 add esp, 0D8
0040102C C3 ret
0040102D 90 nop
0040102E 90 nop
0040102F 90 nop
00401030 53 push ebx
00401031 56 push esi
00401032 0E 50704000 mov esi, offset 00407050
00401037 57 push edi
00401038 56 push esi
00401039 E8 5B020000 call 00401299
0040103E 8BF8 mov edi, eax
00401040 8D4424 18 lea eax, [esp+18]
00401044 50 push eax
00401045 FF7424 18 push dword ptr [esp+18]
00401049 56 push esi
0040104A E8 14030000 call 00401363
0040104F 56 push esi
  
```

3. 执行到 fgets 函数入口时，我们在调试窗口输入“AAAA%x%x%x%x”，继续往下执行到 printf 函数入口时，我们可以发现最后一位输出了 41414141，这是由于在 Release 模式下没有申请额外的栈空间，不会输出“CCCCCCCC”，而是输出了输入字符串的起始地址所保存的值。



0012FEAC	0012FEBC	ASCII "AAAA%x%x%x%x"
0012FEB0	0012FEB0	ASCII "AAAA%x%x%x%x"
0012FEB4	000000BB	
0012FEB8	00407030	ASCII "-0"
0012FEB0	41414141	
0012FEC0	78257825	
0012FEC4	78257825	
0012FEC8	0012000A	
0012FECC	20202020	
0012FED0	0012FF28	
0012FED4	00000000	
0012FED8	00000000	
0012FEDC	FFFFFFFF	
0012FEE0	7C932D58	从返回 ntd11.7C92E8E6 自 ntd11.7C932D

两种模式的差异：

再用两种模式对程序进行调试时，Debug 模式会首先会对栈初始化一个较大的空间，以及对寄存器的 push 操作，而且包含了调试信息，可以通过断点等操作快速找到错误信息，这样生成的代码更易于理解和调试，但由于没有对代码进行优化，运行速度较慢。

Release 模式的代码段比较紧凑，首先在初始化栈空间时只初始化了局部变量所需要的空间，并没有额外申请空间，对代码进行了优化，优点是代码体积更小，执行效率更高，缺点是不利于对代码的调试和跟踪，也容易导致内存泄漏等一系列问题。

心得体会：

理解了格式化字符串导致内存泄漏的基本原理，掌握了在 OllyDbg 中逐步调试的方法，同时通过对 Debug 模式和 Release 模式的比较，直观感受到了两种模式的差异。