《软件安全》实验报告

姓名：王昱　　学号：2212046　　班级：信息安全班

一、实验名称：

格式化字符串漏洞

二、实验要求：

以第四章示例 4-7 代码为准，完成任意地址的数据获取，观察 Release 和 Debug 模式的差异。

三、实验原理：

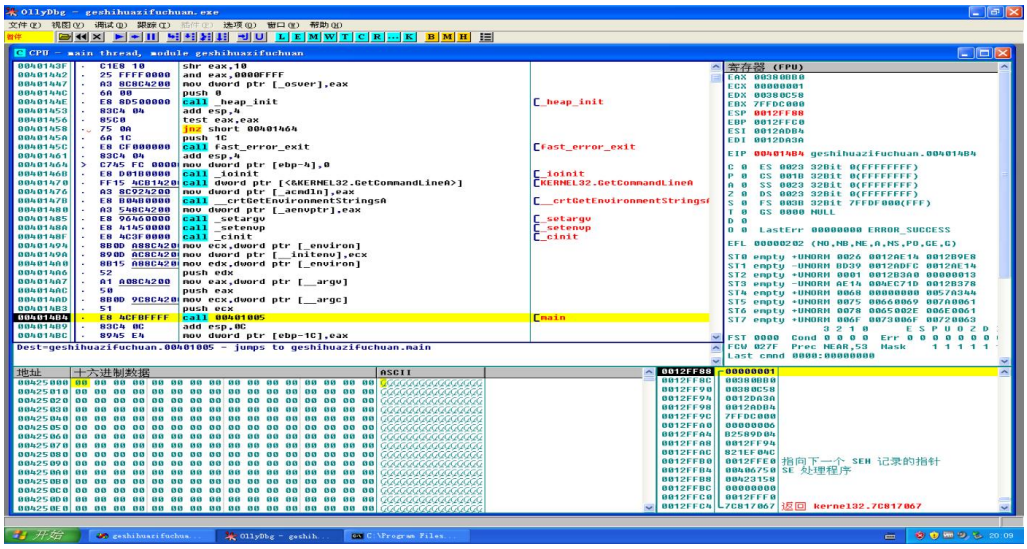
C 语言中的格式化函数允许可变参数，它根据传入的格式化字符串获知可变参数的个数和类型，并依据格式化符号进行参数的输出。如果调用这些函数时，给出了格式化符号串，但没有提供实际对应参数时，这些函数会将格式化字符串后面的多个栈中的内容取出作为参数，并根据格式化符号将其输出。调用时如果传入”%x%x…%x”，则 printf 会打印出堆栈中的内容，不断增加%x 的个数会逐渐显示堆栈中高地址的数据，从而导致堆栈中的数据泄漏。而如果输入%s，则会输出对应地址所指向的字符串。

四、实验过程：

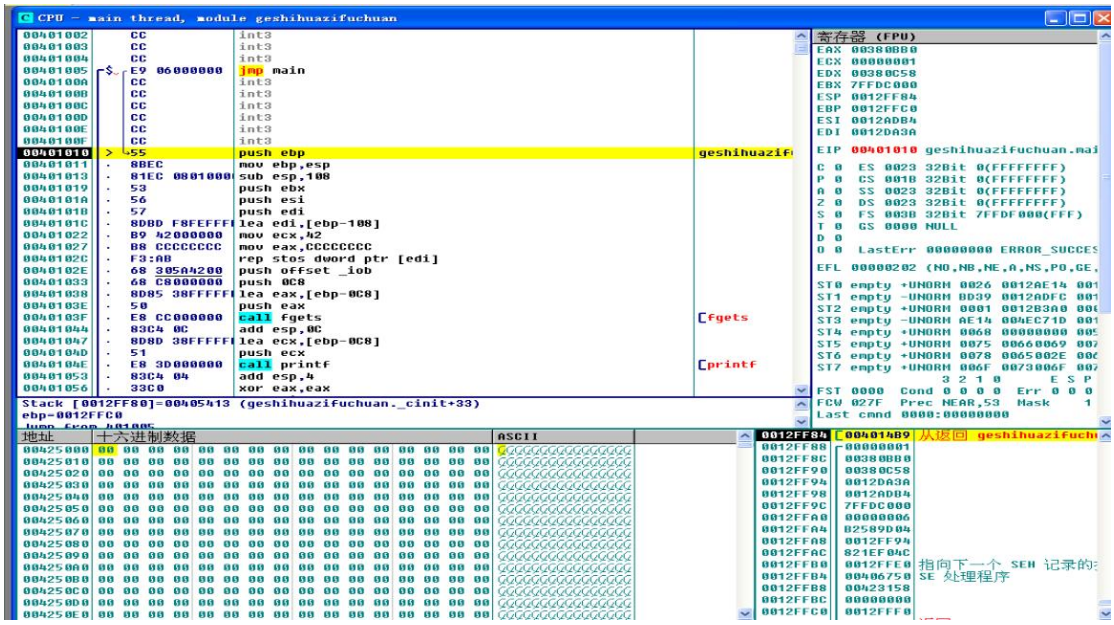
Debug 模式：

（1）输入代码，并用 Debug 模式构建程序，用 Ollydbg 进行反汇编。

Ollydbg 打开程序



通过 F8 单步执行，F7 步入主函数代码



（2）分析主函数的汇编代码

```
55          push ebp
8BEC        mov ebp,esp
81EC 0801000( sub esp,108
53          push ebx
56          push esi
57          push edi
8DBD F8FEFFF( lea edi,[ebp-108]
B9 42000000 mov ecx,42
B8 CCCCCCCC mov eax,CCCCCCCC
F3:AB       rep stos dword ptr [edi]
```

这部分进行函数的栈帧调整，首先保存 ebp 的值，将其入栈，然后调整函数栈帧为 main 函数的栈帧，将 esp 减去 108，栈向上增长开辟新的空间，为函数开辟栈帧。随后将要用到的寄存器入栈，再将新开辟的栈空间初始化为 CCCCCCCC。

0012FE6C	0012DA3A
0012FE70	0012AD84
0012FE74	7FFDC000
0012FE78	CCCCCCCC
0012FE7C	CCCCCCCC
0012FE80	CCCCCCCC
0012FE84	CCCCCCCC
0012FE88	CCCCCCCC
0012FE8C	CCCCCCCC
0012FE90	CCCCCCCC
0012FE94	CCCCCCCC
0012FE98	CCCCCCCC
0012FE9C	CCCCCCCC
0012FEA0	CCCCCCCC
0012FEA4	CCCCCCCC
0012FEA8	CCCCCCCC

如图所示为调整后的栈。从上往下依次是 edi,esi,ebx。

```
push offset _iob
push 0C8
lea eax,[ebp-0C8]
push eax
```

以上汇编代码定义了局部变量 char str[200],参数从右到左依次入栈，0C8 十六进制，对应 200 十进制，保存了数组的大小。eax 保存了数组的首地址[ebp-0C8]。然后将数组的大小入栈。

此时栈的情况：从上到下分别是数组的首地址，数组的大小。

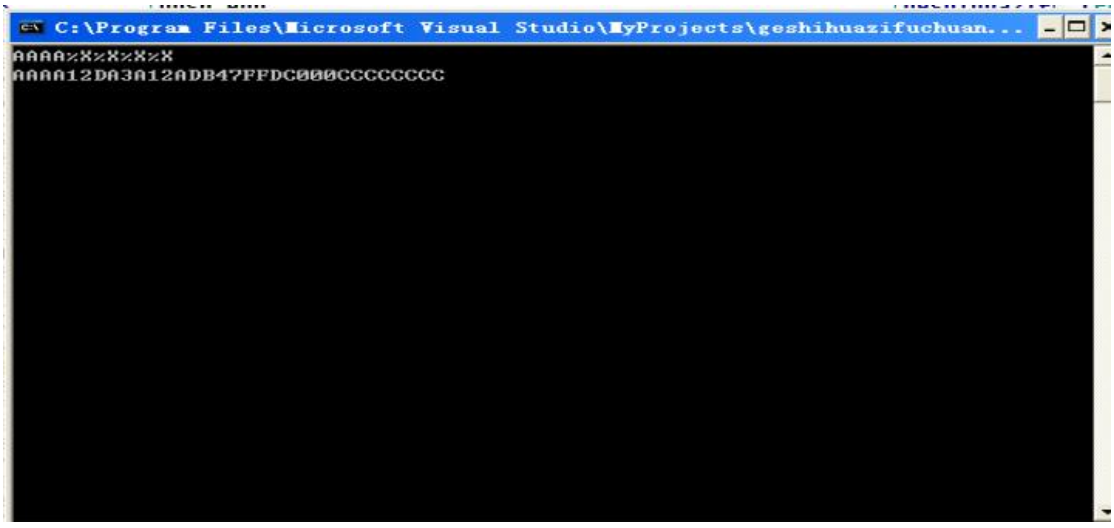
```
call fgets
add esp,0C
```

调用 fgets 函数，此时在命令行输入 AAAA%x%x%x%x，那么数组的首地址保存的内容就是字符串的内容 AAAA%x%x%x%x。之后调用 add esp,0C 还原主函数栈帧。

如图所示，0012FEB8 保存了字符串。

```
8D8D 38FFFFFF lea ecx,[ebp-0C8]
51            push ecx
E8 3D000000   call printf
83C4 04       add esp,4
```

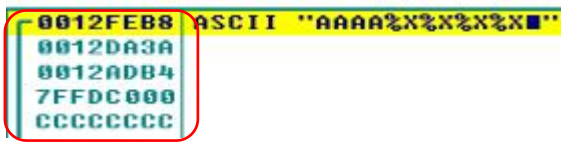
首先，前两行代码将 printf 函数的参数——数组的首地址压入栈中。调用 printf 函数后函数输出的结果如下：



随后将函数的栈帧恢复。

（3）输出结果分析：

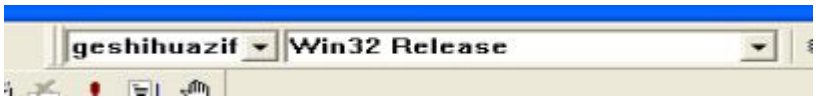
函数输出 AAAA，随后碰到%x,则会逐渐显示堆栈中高地址的数据。从首地址开始往下找下一个参数,但是由于只传了 ECX 一个参数,所以将输出 0012DA3A,0012ADB4,7FFDC000,CCCCCCCC,分别对应 main 函数开始时 push 的 edi、esi、ebx 以及栈帧初始化的内容 CCCCCCCC。输出内容如下图的栈中所示：



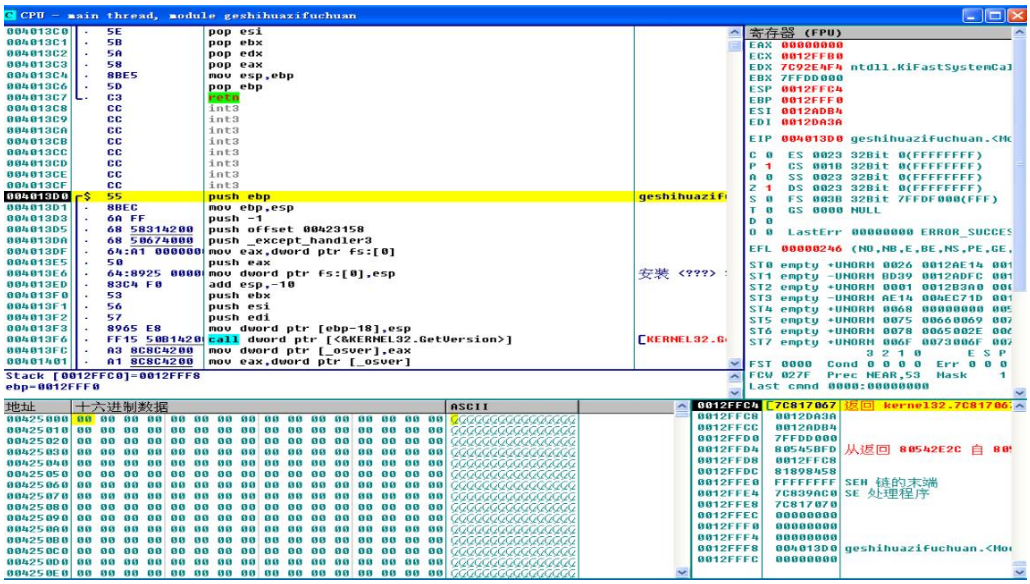
我们的目的是读取任意地址的内容，但是由于 debug 模式会开辟一大段空间并且进行初始化为 CCCCCCCC，所以很难读取到我们所构造的地址。因为开辟了足够大的栈帧并初始化，char str[200]是从靠近 EBP 的地址分配空间，如果要读到 str 的地址，需要很多的格式化字符。

Release 模式：

（1）输入代码，并用 Release 模式构建程序，用 Ollydbg 进行反汇编。



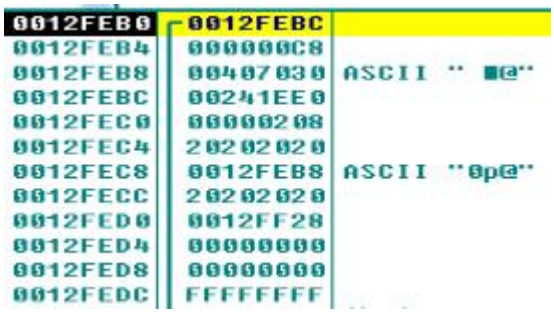
转换为 Release 模式



单步执行到 call 00401000，按 F7 步入主函数

(2)主函数汇编代码分析

直接将 esp-0c8，将栈向上增长开辟空间，而空间大小为 200，恰好为数组所需要的空间。随后将 esp 地址的值赋给 eax。此时 eax 保存的值就是数组的首地址。然后将 fgets 函数参数依次入栈。



如图为 fgets 传参后栈的内容

```
call 00401061
lea ecx,[esp+0C]
```

执行 fgets 函数，命令行输入 AAAA%x%x%x%x。【esp+0C】指的就是数组的首地址

如图为执行 fgets 函数后栈的内容

```
lea ecx,[esp+0C]
push ecx
call 00401030
```

将【esp+0C】，即 str 的地址保存到 ecx 之中，然后把 ecx 压入栈中。调用 printf 函数。

如图为执行 printf 函数后栈的内容。（从上到下依次是：ecx、fgets 的三个参数、字符串的 ASCII 值）

（3）分析程序输出

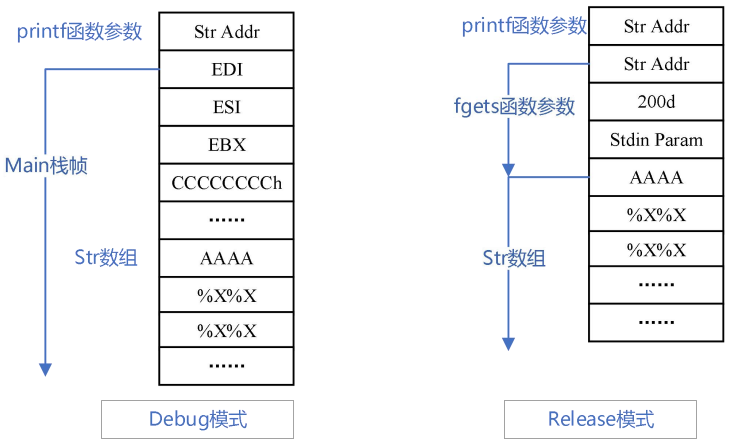
执行 printf 函数时，我们仅仅传入了 AAAA，所以先输出 AAAA，随后遇到了%x，会自动向下寻找参数，所以会依次输出栈中的内容，分别是 fgets 的三个参数：0012FEBC,000000BB，00407030,字符串的内容 41414141（0x41 是 A 的 ASCLL 值）。因此，在 Release 模式下，没有多余的空间去初始化为 CCCC，所以我们可以通过控制过格式化字符串漏洞打印任意构造地址（str 的 ASCLL 值）的数据。

要注意此时要将最后一个%x 改为%s，这样才能输出对应的数据。

Debug 模式和 Release 模式的区别：

（1）Debug 是调试版本，包含调试信息 ，所以容量比 Release 大很多，并且不进行任何优化（优化会使调试复杂化，因为源代码和生成的指令间关系会更复杂），便于程序员调试。，程序局部变量的内存空间会被扩大，并且全部初始化为 0xCC，并且会在栈中保存一些寄存器的信息，如 EDI,ESI,EDX 等。具体在本次实验中，因为开辟了足够大的栈帧并初始化，char str[200]是从靠近 EBP 的地址分配空间，如果要读到 str 的地址，需要很多的格式化字符。

（2）Release:发布版本，不对源代码进行调试 ，编译时对应用程序的速度进行 优化 ，使得程序在代码大小和运行速度上都是最优的，以便用户很好地使用。Relase 模式下，可以看到，并没有严格按照制式的栈帧分配，而是考虑运行性能，往往会进行优化，删除调试信息，以达到代码最小和速度最优的目的，此时栈中不再保留不必要的内容，并且函数局部变量的空间也不会超过所需要的大小。



（执行 printf(str)语句的时候，对比 Debug 模式和 Relase 模式的栈帧结构）

四、心得体会：

1.对 Ollydbg 的使用更加熟悉，明白了如何通过单步调试和步入调试进入主函数。

2.了解了格式字符串漏洞的原理以及如何利用格式化字符串漏洞实现读取任意内存地址的数据。

3. 明白了 Debug 模式和 Release 模式的差异，各自的用途以及优缺点。