《漏洞利用及渗透测试基础》实验报告

姓名：申宗尚 学号：2213924 班级：信息安全

**实验名称：**

格式化字符串漏洞

**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：**

1. 进入VC6，打开提供的4-7代码进行调试(debug模式和release模式)。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

2. 运行程序（以release模式为例），可以得到如下输出结果。

图形用户界面, 应用程序

描述已自动生成

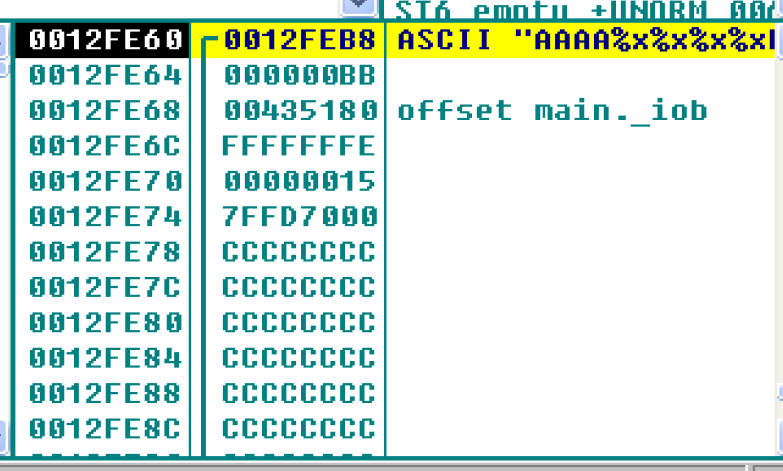
3.打开ollydbg对生成的.exe文件（debug模式）进行分析，通过argc和argv找到main函数的入口，按F7步进进入函数内，可以看到函数内汇编代码如下：

文本

描述已自动生成

在一开始，首先push ebp存下旧栈帧，然后将栈帧移到esp位置，最后sub esp，108，创建新的位置，可以发现，对于char str[200]，只需要200字节，转换为16进制为0xC8，但debug模式下却给出了0x108大小的空位，并通过后面的rep stos指令，将这些空位全部设初始值0xCCCCCCCC

在栈中也可以看到，在栈帧下，全部为0xCCCCCCCC



4. 继续调试，运行到fgets，调用fgets函数获取输入：AAAA%x%x%x%x

图片包含 图形用户界面

描述已自动生成

栈变化如下：

文本, 表格

描述已自动生成

可以看到，在CCCCCCCC结束之后（即0x108大小后），存入了str的值，为AAAA（A的ASCII码16进制下为41），即为41414141

5.随后继续调试，直到调用printf函数，获取程序输出如下：  
图形用户界面, 文本

描述已自动生成

由于printf会输出字符串内容，从而str的AAAA被输出，而后面的四个%x被看作格式化参数，由于格式化字符串漏洞的特性，在没有给出实际参数的情况下，会将格式化字符串后的栈内容取出作为参数

从而，对于前面的3个%x，分别对于后面的地址，以16进制输出了堆栈内容：0x0012d9dc，0x0012adb7，0x0ffd9000…，（省略前置0）而最后到了初始化的“CCCCCCCC”部分，从而输出对应地址代表的CCCCCCCC。

6. 类似地、通过ollydbg进行release模式下的.exe文件分析，进入main函数：

图形用户界面, 文本

描述已自动生成

可以发现，与debug模式不同，在一开始没有push ebp操作，只有sub esp 0C8，即将栈指针减去0C8，刚好预留出str需要的200字节大小，从而其堆栈内内容如下：

表格

描述已自动生成

在栈顶的下面，即为41414141，没有重复的0x108字节的CCCCCCCC

7. 类似地，对其进行fgets函数和printf函数的调试，原理同上，获取输出如图所示：

图形用户界面, 文本

描述已自动生成

分析如下：对于AAAA：正常输出

第一个%x：输出帧顶部的第一个地址处的16进制值:0x0012FEBC（省略前置0）

第二个%x：输出下一个地址16进制值:0x000000BB

第三个%x：输出下一个地址16进制值:0x0040D448

第四个%x：输出下一个地址16进制值:0x41414141（AAAA）

从而，由于开辟main函数帧空间时对于空间申请大小的不同导致str的AAAA存储在不同的位置，在利用格式化字符串漏洞进行数据获取时候，会造成不同的输出。

表格

描述已自动生成

**心得体会：**

通过实验，了解、学习了release模式和debug模式二者的异同，进行了一番课外学习。

同时，使用VC6对程序进行了几次调试，中间出现了一点问题（断点方面），进行了一些学习，对于VC6对调试界面有了更好的理解。

随后放弃VC6，使用ollydbg进行调试，对于寻找main函数有些忘了，通过视频学习和知识复习，巩固了ollydbg的用法。

最后，亲身实践了课上学习的格式化字符串漏洞，了解了其利用方式和可能影响、实际操作，对于其有了更全面的理解。