《软件安全》实验报告

姓名：辛浩然　　学号：2112514　班级： 信息安全、法学

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读取。观察 Release 模式和 Debug 模式的差异，并进行总结。

实验过程：

1.写入代码

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

该代码写入一个最长 200 字节的字符串，然后调用 printf 函数打印该字符串。printf 函数属于格式化函数，其允许可变参数，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。调用时，如果给出了格式化符号串，但没有提供实际对应参数时，它会将格式化字符串后面的多个栈中的内容弹出作为参数，并根据格式化符号将其输出。

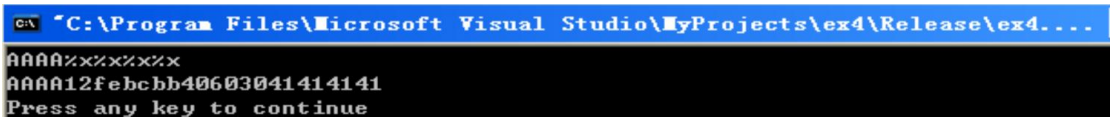
所以在本实验中，如果输入的 str 内容含有格式化符号串，并且不为它提供对应参数，就可以利用格式化字符串漏洞实现越界数据的访问。

2.编译并运行（**Release** 模式）

输入：AAAA%x%x%x%x

得到：AAAA12febcbb40603041414141

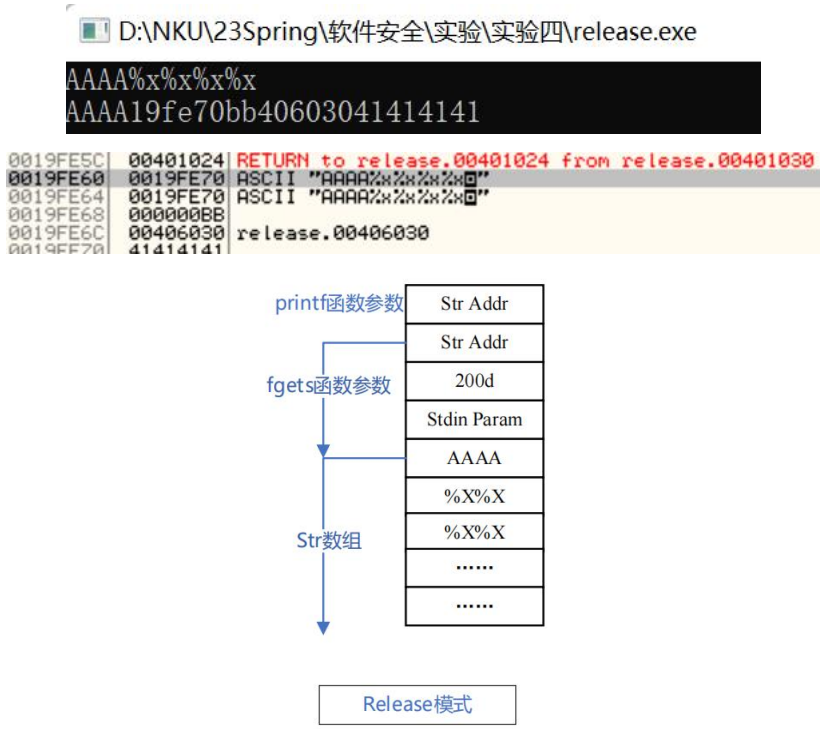
参数入栈（字符串 str 的地址）后，通过%x 依次读参数下面的内存数据时，很快就读到了原来函数的局部变量 str 的数据了。



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，在执行到 printf(str)的时候，栈区自顶到底部分为存着“printf 函数参数|fgets 函数参数|str 数组”的内容，在 Main 函数的 retn 语句前，才有一个 add esp XX 的处理。

在 ollydbg 中打开 exe 文件：

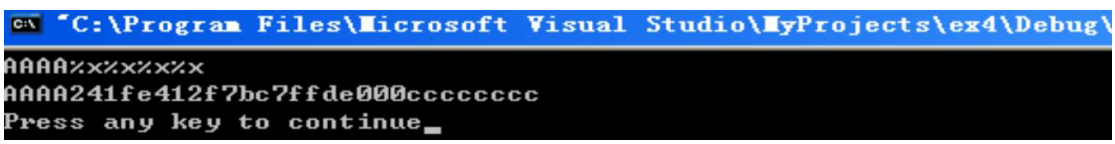
可以发现，参数入栈（字符串 str 的地址）后，通过%x 依次读参数下面的内存数据，以 16 进制的形式输出了 fgets 函数的三个参数，很快就读到了原来函数的局部变量 str 的数据。



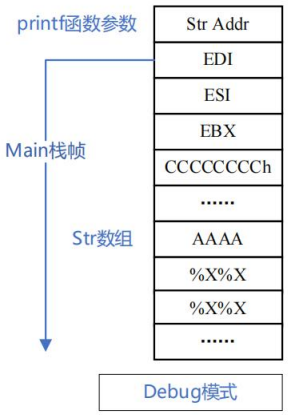
3.编译并运行（**Debug 模式**）

输入：AAAA%x%x%x%x

得到：AAAA241fe412f7bc7ffde000cccccccc



Debug 模式下，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；



在 ollydbg 中打开，可以发现开辟了足够大的栈帧，通过%x 依次读参数下面的内存数据，以 16 进制的形式输出了 edi、esi、ebx 和 CCCCCCCC.

4. 如果将 AAAA 换成地址，第 4 个%x，换成%s 的读取参数指定的地址上的数据，就可以读取任意内存地址的数据了。

心得体会：

1.通过实验，体会到了调用函数时 release 模式和 debug 模式栈帧结构的不同：debug 模式开辟足够大的栈帧并进行初始化，Relase 模式考虑运行性能，没有严格按照制式的栈帧分配。

2.利用格式化函数允许可变参数特性进行了越界数据的访问，对其理解更为深入。