编号：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |

武汉大学国家网络安全学院

课程实验(设计)报告

题 目： 实验三：格式化字符串

专业(班)： 信息安全

学 号：

姓 名：

课程名称：

任课教师：

2023年 12 月 12日

目录

[实验3 1](#_Toc155364352)

[3.1实验名称 1](#_Toc155364353)

[3.2实验原理 1](#_Toc155364354)

[3.3实验目步骤及内容 1](#_Toc155364355)

[3.4 实验结果 2](#_Toc155364356)

## 实验3

### 3.1实验名称

* 格式化字符串

### 3.2实验原理

格式化字符串的类型：

printf函数的格式化字符串常见的有 %d，%f，%c，%s，%x（输出16进制数，前面没有0x），%p（输出16进制数，前面带有0x）等等。

还有个不常见的格式化字符串 %n ，它的功能是将%n之前打印出来的字符个数，赋值给一个变量，我们格式化字符串溢出常用到这个格式符%n，用于修改某个内存中的内容

还有一个直接定位参数的用法：%< number>$x 是直接读取第number个位置的参数，同样可以用在%n，%d等等。

比如我们想要读取格式化字符串的第7个参数：%7$x

用一个简单的例子来演示：



可以看到，我们通过精心构造一个我们要printf的字符串，可以泄露内存地址，我们通过上述方式找到我们要写入的位置，然后通过%< number>$x，将我们要写入的东西，写入到我们找到的位置中

### 3.3实验目步骤及内容

我先分析一下vul.c

   char buff[36] = {0};

   int a = 0;

   int b = 0;

   FILE \*fp;

   char ch;

   if( (fp = fopen(argv[1], "r")) != NULL )

   fread(buff, 32, 1, fp);

首先为buff数组开辟了大小为36的字符空间，并且使用fread函数打开命令行参数传入的文件，从中读取32字节的内容，并存储到buff数组中

   printf("the addresses of a b buf are %x,%x,%x\n", &a, &b, buff);

然后使用printf函数打印a,b的地址，及buff所指向的字符串的值

   a = 0;

   printf(buff);

   printf("a= %d\n", a);

然后打印buff和a的值，这里，这里就可以发生格式化字符串漏洞，通过在buff中填充一些格式符，来实现内存泄露，甚至可以更改a的值

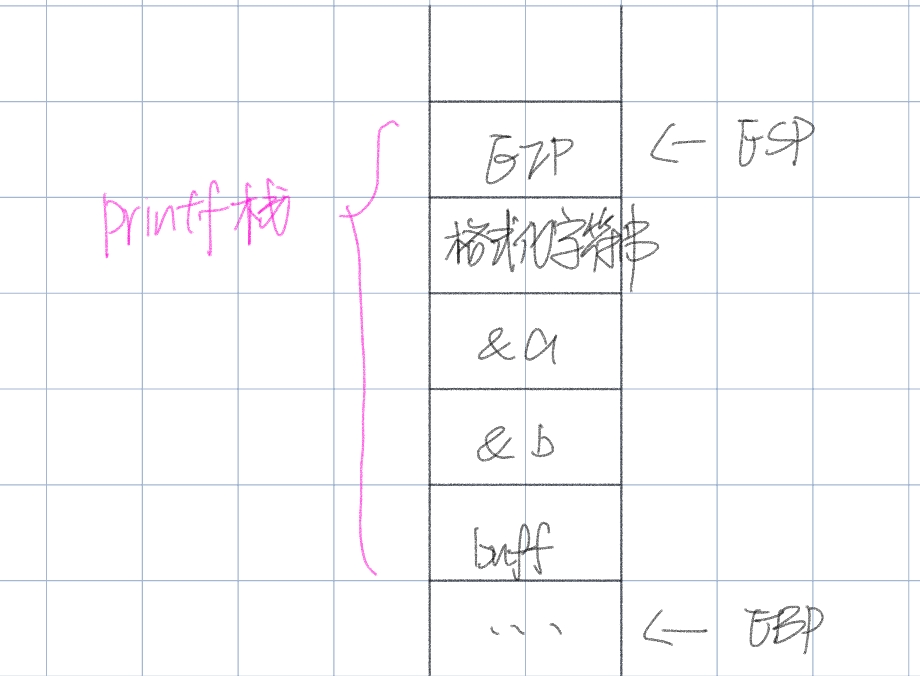
   if(fp)

     fclose(fp);

最后关闭文件

   printf("the addresses of a b buf are %x,%x,%x\n", &a, &b, buff);

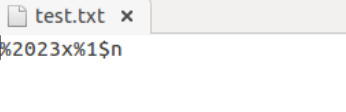
这句printf函数对应的栈空间如下所示



想要修改a的值，我们就要定位到a的地址，如上图所示，就是格式化字符串下面的位置，于是我们使用%1$n，定位到第一个格式化字符串的参数，即a所在的位置，我们可以将arg1前面输出字符串的个数写入到a中

在这里，我们想让a的值变成2023，那我就让前面的输出的字符串的位数变成2023，所以使用%2023x即可

因此，我们buff的内容就构造好了



### 3.4 实验结果

运行



可以看到a已经变成2023