格式化字符串

原理：类printf函数的最大的特点就是，在函数定义的时候无法知道函数实参的数目和类型。

对于这种情况，可以使用省略号指定参数表。

带有省略号的函数定义中，参数表分为两部分，前半部分是确定个数、确定类型的参数，第二部分就是省略号，代表数目和类型都不确定的参数表，省略号参数表中参数的个数和参数的类型是事先的约定计算出来的，每个实参的地址（指针）是根据确定参数表中最后一个实参的地址算出来的。

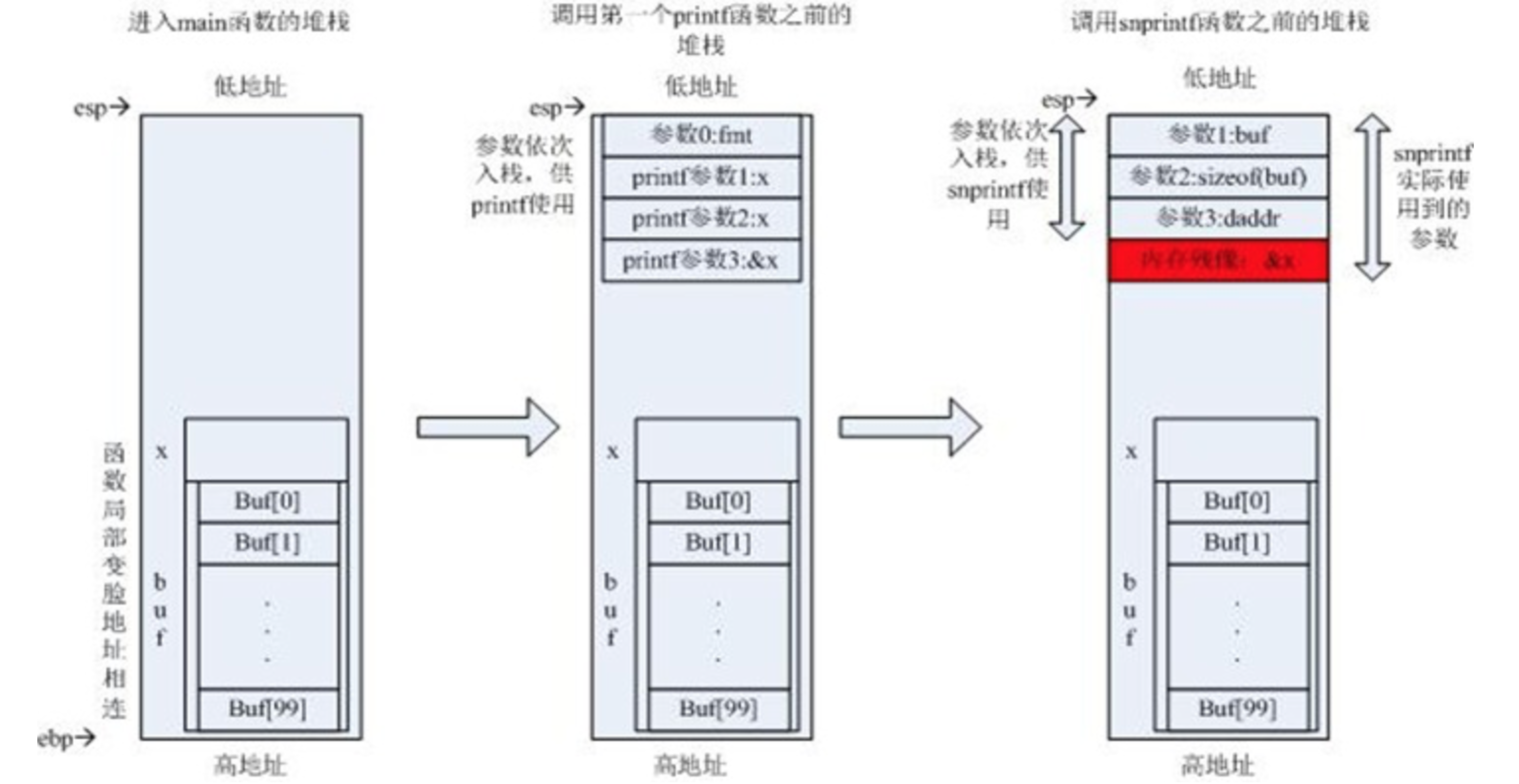
这里涉及到函数调用时的栈操作。函数栈的栈底是高地址，栈顶是底地址。在函数调用

时函数实参是从最后一个参数（最右边的参数）到第一个参数（最左边的参数）依次被压入栈顶方向。也就是说函数调用时，函数实参的地址是相连的，并且从左到右地址是依次增加的。

因为类printf函数中省略号参数表中参数的个数和类型都是由类printf函数中的那个格式化字符串来决定的，所以攻击者可以利用编程者的疏忽或漏洞，巧妙构造格式化字符串，达到攻击目的。

如果一个程序员的任务是：打印输出一个字符串或者把这个串拷贝到某缓冲区内。他可以写出如下的代码：printf("%s", str);但是为了节约时间和提高效率，并在源码中少输入6个字节，他会这样写：printf(str);

为什么程序员写的是错误的呢？他传入了一个他想要逐字打印的字符串。实际上该字符串被printf函数解释为一个格式化字符（formatstring），printf就会根据该字符串来决定printf函数中省略号参数表中参数的格式和类型，如果这个程序员想要打印的字符串中刚好有“%d”,“%x”之类的格式化字符，那么一个变量的参数值就从堆栈中取出。



漏洞程序：

int flag = 9;

int \*p = &flag;

char a[100];

len = recv(client\_sockfd,a,100,0);

printf(a);

黄色部分为格式化字符串漏洞

攻击程序：

%2000x%19$n

构造格式化字符串输入给漏洞程序，使的漏洞程序中变量值改变从而改变程序执行结果。