数据。一种防止这种攻击的方法是设定栈页面为读/写权限，而不是执行权限。虽然大多数操作系统都一定不支持这个功能，但现代的“奔腾”CPU可以做到这一点。还有一种攻击在栈不能被执行的条件下也能奏效，这就是返回libc攻击（return to libc attack）。

假设一个缓冲区溢出攻击或格式化字符串攻击成功修改了当前函数的返回地址，但是无法执行栈中的攻击代码，那么还能否通过修改当前函数的返回地址到指定位置来实现攻击呢？答案是肯定的。几乎所有的C程序连接了libc库（通常该库为共享的），这个库包括了C程序几乎所有的关键函数，其中的一个就是strcpy。该函数将一个任意长度的字符串从任意地址复制到另一地址。这种攻击的本质是欺骗strcpy函数将恶意程序（通常是共享的）复制到数据段并在那里执行。

下面让我们观察这种攻击实现的具体细节。在图9-25a中，函数f在main函数中被调用，形成图中的栈。我们假设这个程序在超级用户的权限下运行（如，SETUID root），并且存在漏洞使得攻击者可以将自己的shellcode注入到内存中，如图9-25b所示。此时这段代码在栈顶，因此无法被执行。

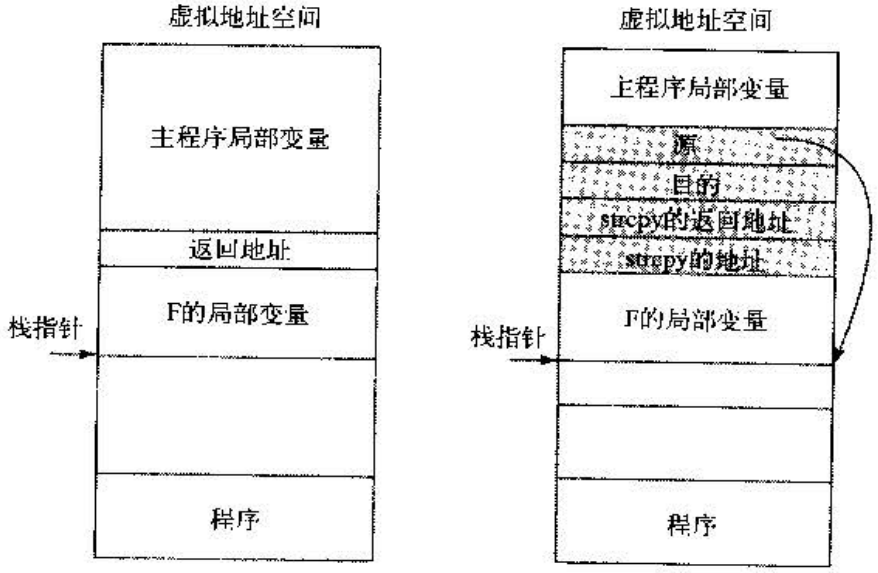

图9-25 a) 攻击之前的栈；b) 被重写之后的栈

除了将shellcode放到栈顶，攻击者还需要重写图9-25b中阴影部分的四个字。这四个字的最低地址之前保存了该函数的返回地址（返回到main），但现在它保存的是strcpy函数的地址，所以当f返回的时候，它实际上进入了strcpy。在strcpy中，栈指针将会指向一个伪造的返回地址，该函数完成后会利用该地址返回。而这个伪造的返回地址所指向的，就是攻击者注入shellcode的地址。在返回地址之上的两个字分别是strcpy函数执行复制操作的源地址和目的地址。当strcpy函数执行完毕，shellcode被复制到可执行的数据段，同时strcpy函数返回到shellcode处。shellcode此时具有被攻击程序所有的权限，它为攻击者创建一个shell，并开始监听一些IP端口，等待来自攻击者的命令。从此刻起，这台机器变成了僵尸机器（zombie），可以被用来发送垃圾邮件或者发起“拒绝服务攻击”（denial-of-service attack）。

9.6.4 整数溢出攻击

计算机进行定长整型数的运算，整型数的长度一般有8位、16位、32位和64位。如果相加或相乘的结果超过了整型数可以表示的最大值，就称溢出发生了。C程序并不会捕捉这个错误，而是会将错误的结果存储下来并继续使用。一种特别的情况是，当变量为有符号整数时，两个整数相加或想成的结果可能因为溢出而成为负数。如果变量是无符号整数，溢出的结果依然是整数，不过会围绕0和最大值进行循环（wrap around）。例如，两个16位无符号整数的值都是40 000，如果将其相乘的结果存入另一个一个16位的无符号整型变量中，其结果将会是4096。

由于这种溢出不会被检测，因此可能被用作攻击的手段。一种方式就是给程序传入两个合法的（但是非常大）的参数，它们的和或乘积将导致溢出。例如，一些图形程序要求通过命令行传入图像文件的高和宽，以便对输入的图像进行大小转换。如果传入的高度和宽度会导致面积的“溢出”，程序就会错