Java技术之所以在今天得到了如此广阔的应用，其中它的安全性是不能不提的。不同于其它技术（例如 Microsoft的ActiveX）中安全性作为附加设计和补丁，Java从设计之初便考虑到了安全性。因此Java的安全性是在语言层次实现的。 Java的安全性由下列三个方面保证： 1、语言特性（包括数组的边界检查、类型转换、取消指针型变量）。 2、资源访问控制（包括本地文件系统访问、Socket连接访问）。 3、代码数字签名（通过数字签名来确认代码源以及代码是否完整）。 本文主要讨论结合后两种技术来实现超越Applet的安全限制。我们先来看一下这三个方面的具体实现。我们知道Java的原代码是先编译成为一种字节码的中间代码，存放这种代码的文件就是.class的文件。真正执行的时候是将class文件装载到JVM（虚拟机）中，然后由JVM解释执行的。所以数组的上下界检查及合法的类型转换是通过JVM得到保证的。 Java通过一个类装载器类(ClassLoader)将虚拟机代码文件（即 class文件）装载到JVM中，当完成装载后，一个被称做安全管理器（SecurityManager）的类开始运行，这就是上面描述的第二个方面的实现。例如当一个Applet的class文件被缺省的类装载器

装载到JVM中后，JVM会立即为它装载一个SecurityManager的子类 AppletSecurity，由这个管理器来验证操作。代码的所有动作（例如文件读写）都要先经过验证，只有被该安全管理器接受的动作才能完成，否则就会抛出SecurityException异常。那么安全管理器类是怎么判断代码的权限的呢？这就是利用Policy文件。 对于 JDK1.0，权限被笼统的划分为两大块。一是拥有所有的权限，一个是仅拥有"沙箱"（sandBox）权限，这也是普通的Applet所拥有的权限。这时本地文件读写或是与源主机（Orignal Server）以外的主机连接都是被禁止的。这种划分的最大问题就是缺乏灵活性。例如我们希望一个Applet在用户信任的情况下能够对本地文件系统的某个目录进行读写，但并不要通过Socket与其它主机连接。这是JDK1.0的权限划分就不能达到要求。JDK1.1后改进了权限的划分，引入了权限集（PermissionSet）的概念。它细划了权限的放放面面，你可以有选择性的组合你需要的权限来达到特殊的要求。 