JAVA语言安全行研究－－Java的反编译 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/462/2021_2022_JAVA_E8_AF_AD_E8_A8_80_c104_462412.htm Java诞生于1995年，是一门较年轻的语言。它以平台无关性，安全性，面向对象，分布式，键壮性等特点赢得了众多程序员的青睐。特别是它简洁的面向对象的语言风格，更让许多人对它爱不释手。但人们在使用Java的过程中，会发现它有几个致命的弱点：运行速度慢，用户使用不便，源代码保护机制不够安全。特别是在保护源代码方面，Java是基于解释一种叫Java字节码的中间代码来运行其程序的，而且Jvm比计算机的微处理器要简单的多，文档也很齐全，结果造成其目标程序很容易被反编译，而且所得代码和其原始代码十分相似，甚至可以一模一样，可读性相当好。这就给Java的代码保护带来了不利。但要实现Java程序的保护，也不是不可能的，经研究和总结，至少有三种实现方式：1.混淆器；2.网络加载重要类；3加密重要类。 1 混淆器 目前，开发人员使用的比较多的保护代码的方法是用混淆器。混淆器是采用一些方法将类，变量，方法，包的名字改为无意义的字符串；使用非法的字符代替符号；贴加一些代码使反编译软件崩溃；贴加一些无关的指令或永远执行不到的指令等使反编译无法成功或所得的代码可读性很差。这样就实现了反反编译的目的。我们来做个演示。原始代码如下：import Java.io.*； import Java.security.*； public class sKey_kb{ public static void main（String args[]） throws Exception{ FileInputStream f=new FileInputStream（"key1.dat"）； ObJectInputStream b=new ObJectInputStream（f）； Key k=

（Key）b.readObJect（）；byte[] kb=k.getEncoded（）；FileOutputStream f2=new FileOutputStream（"keykb1.dat"）；f2.write（kb）；for（int i=0；i System.out.print（kb[i] "，"）；}}} 使用混淆器后，再用Jad反编译得代码如下：import Java.io.*；import Java.security.Key；public class sKey_kb{ public skey（）{} public static void main（String args[]）{ FileInputStream fileinputstream=new FileInputStream（ma）；ObJectInputStream obJectinputstream=new ObJectInputStream（filein putstream）；Key key=（Key）b.readObJect（）；byte abyte0[]=key.getEncoded（）；FileOutputStream fileoutputstream=new FileOutputStream（na）；fileoutputstream.write（abyte0）；for（int i=0；i System.out.print（abyte0[i] oa）；} private static String a（String s）{ int i=s.length（）；char ac[]=new char[i]；for（int J=0；J return new String（ac）；} private static String ma="u5AA1u5AAFu5AF3u5AFBu5AE4u5AAEu5AABu5ABE"；private static String na="u5AA1u5AAFu5AB3u5AA1u5AA8u5AFBu5AE4u5AAEu5AABu5ABE"；private static String oa="u5AE6"；public static{ ma=a（ma）；na=a（ma） oa=a（oa）；}} 混淆后，再反编译所仍然能得到源代码，但显然，所得代码与原始代码比，变得难以读懂，代码中多了其他的方法，文件名等信息也被打乱了。并且，把以上代码写进sKey_kb.Java中，无法通过编译。但是，如果在编写软件时，在软件中写入某些注册信息，或一些简单的算法，通过反编译，还是有可能得到这些信息的，从而未能达到保护软件的目的。反编译器与混淆器之间的

斗争是永无止尽的。所以从其他角度去保护Java的源代码是很有必要。 2 网络加载重要类 在Java中提供了一个ClassLoader类，这个类可以让我们使用类加载器将所需要的Java字节码文件加载到Jvm中。我们通过重写这个类，可以实现从网络通过url加载Java字节码文件。这样，我们就可以把一些重要的，隐秘的class放在网络服务器上，通过口令去检验是否有权限下载该类。从而实现Java代码保护的目的。其次在Java中正好提供了URLClassLoader这个类，通过此类，正好可以实现我们的目的。URLClassLoader类的基本使用方法是通过一个URL类型的数组告诉URLClassLoader类的对象是从什么地方加载类，然后使用loadclass（）方法，从给定的URL中加载字节码文件，获得它的方法，然后再执行。 具体步骤如下： 1.创建URLURL url[]={ new URL（"file:///c：/classloader/web"），new URL（"http://www.asp.zJc.zJut.edu.cn/Javaclass/"）}； 2.创建URLClassLoader对象 URLClassLoader cl=new URLClassLoader（url）； 3.使用URLClassLoader对象加载字节码文件 Class class=cl.loadClass（"class1"）； 4.执行静态方法Class getarg[]={（new String [1]）。getClass（）}； Method m=class.getMethod（"main"，getarg）； String[] myl={"arg1 passed"，"arg2 passed"）； ObJect myarg[]={myl}； m.invole（null，myarg）； 3 加密重要类 使用网络加载重要类的方法固然有一定的用处，但是，在遇到无网络的情况时，还是无法解决我们的问题。对于这种情况，我们只能把所有文件放在本地计算机上。那么，对此我们该怎么做才能保护好Java代码呢？ 