第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)01—0093—02 文献标识码：A 中图分类号：TP311.1

改进的 Java 类文件保护方法

鲍福良，徐 洁，方志刚
(浙江大学城市学院信电分院，杭州 310015)

摘 要：编译后的 Java 类文件很容易被反编译，在 Java 应用上会产生源代码的安全问题。该文对传统的 Java 类文件保护方法进行分析，在类加载器(ClassLoader)加密技术的基础上，提出一种使用 JNI 技术调用本地解密接口的改进方法。解密接口在设计上保证了 JNI 技术调用本地解密接口时的通信安全，从而提升了 Java 类文件的安全性。实验结果表明，该方法是有效的。

关键词：类加载器；Java 本地接口；类文件；安全性

Improved Method for Protecting Java Class Files

BAO Fu-liang, XU Jie, FANG Zhi-gang
(School of Information & Electric, City College, Zhejiang University, Hangzhou 310015)

【Abstract】 Because the compiled Java class files are easily decompiled, the security for original code in Java application has been a problem. On the basis of ClassLoader encryption, an improved method for transferring the local decrypted interface with Java Native Interface(JNI) technology is proposed, after analyzing the traditional protecting way for Java class files. The decrypted interface garantees the communication security when JNI technology transfers the local decrypted interface, which promotes the security of Java class files. Experimental results show this method is effective.

【Key words】 ClassLoader; Java Native Interface(JNI); class file; security

1 概述

由于编译后的Java类文件不是真正的二进制文件，而是一种有格式的中间代码，因此给黑客反编译Java类文件提供了方便，反编译后的代码和源代码几乎没有差别。一些作者编写的源代码就会轻易地被黑客窃取，一些重要的算法也会泄漏出去[1]。因此，Java应用程序在源代码上会产生很大的安全问题。近年来，已经有许多公司和Java开发人员对Java类文件和虚拟机进行了深入分析，并在此基础上采取各种方法保护Java类文件，在一定程度上起到了保护Java类文件的作用。如本地编译技术、代码混淆技术、ClassLoader加密技术以及数字水印技术等，但是都有各自的局限性。

本文在 ClassLoader 加密技术的基础上，使用 JNI 技术调用本地解密接口，同时保证 JNI 技术调用本地解密接口时的通信安全，从而将 Java 类文件的安全性提升到本地应用程序的安全强度。

2 传统的 Java 类文件保护方法

对 Java 类文件进行反编译后得到的代码和源代码几乎没有差别，因此，如何保护 Java 类文件不被反编译或提高反编译的难度已成为 Java 领域的一个研究热点。目前针对 Java 类文件的保护方法主要有以下几种方法：本地编译技术，代码混淆技术，ClassLoader 加密技术以及数字水印技术。

2.1 本地编译技术

Java本地编译[2]是指将Java应用程序编译成本地应用程序，如Windows操作系统上扩展名为exe的应用程序。其步骤为：先编写Java源代码，然后通过Java编译器将Java源代码编译成Java类文件，最后将Java类文件编译成真正的本地应用程序。

用该技术生成的本地应用程序是二进制格式的可执行文件，与在虚拟机中执行的 Java 应用程序相比，执行速度更快，内存占用更小，而且其安全性能也等价于本地可执行应用程序的安全强度，这些对于当今许多应用都很关键。但该方法牺牲了 Java 的跨平台特性，且现有的本地编译器都不够成熟、类支持不够广泛等原因直接影响了该技术的应用。

2.2 代码混淆技术

代码混淆技术是目前比较成熟和流行的Java类文件保护方法[3]，其本质上是类文件模糊技术。它的原理是把类文件重新进行组织，使别人无法轻易地读懂反编译出来的代码，但是处理以后的类文件功能和处理以前的类文件功能在逻辑上是等同的，即运行后能够得到一样的输出结果。

有些专业代码混淆工具的效果已经非常出色，如果合理使用这些工具就可以对自己的产品起到很好的保护作用。但是代码混淆工具也不是万无一失的，事实上只要有足够的耐心，这些混淆了的代码还是可以被反编译出来并且能够读懂。特别是在重构技术非常成熟的今天，要替换这些变量名或者函数名还是非常容易的。

2.3 ClassLoader 加密技术

该技术是利用Java虚拟机调入Class到系统中进行执行的过程。由于虚拟机每次装入类文件时都需要使用一个ClassLoader对象，因此该对象负责把新的类装入到虚拟机中。虚拟机向ClassLoader对象提供一个包含待装入类名字的字符串，然后由ClassLoader负责找到类文件，并把它转换成一个Class对象[4]。利用上述机制可以重载ClassLoader对象，在装入原始数据后先进行解密，然后再转换成Class对象。由于把原始字节码转换成Class对象的过程完全由系统负责，因此只

基金项目：浙江省科技厅基金资助项目(2006C31006)
作者简介：鲍福良(1979－)，男，讲师、硕士，主研方向：软件工程；徐 洁，讲师、硕士；方志刚，教授、博士
收稿日期：2008-05-20 **E-mail：**baofl@zucc.edu.cn

需先获得原始数据，接着就可以进行包含解密在内的任何转换。

这种保护系统源代码的方法比其他方法更加安全，然而这种加密方法存在一个严重漏洞，即由于 ClassLoader 的类是用 Java 编写的，如果对 ClassLoader 类进行反编译，提取其中解密算法，就可解密所有被加密的其他类。

2.4 数字水印技术

Java类文件容易被反编译，使得对Java程序的未授权使用变得容易。所以，在需要证明程序是否非法使用时，数字水印就变得很重要。像在图片声音中嵌入水印一样，在Java类文件中也能嵌入透明的、安全的和鲁棒性的信息。使用水印技术并不能阻止类文件被反编译，但是可以在需要确认某些程序是否属于剽窃时提供有效的证据。它可以有效地保证开发者对该程序的版权。嵌入的水印对程序的使用者来说是透明的，而对程序的开发者来说，可以轻易地找出未经授权的非法程序使用[5]。

然而数字水印技术也存在一些不足，如需要插入额外的代码，需要仔细地编写哑函数及其调用，否则容易被有经验的反编译者识破，从而擦除水印。而且如果同时利用各种Java混淆器对带有水印的Java文件进行攻击，在有些情况下水印将失效[3]。

3 ClassLoader 加密技术的改进

上述各种传统方法对 Java 类文件的安全性能起到一定的保护作用，但是都有各自的局限性。本文针对这种情况，在上述 ClassLoader 加密技术的基础上，提出采用 JNI(Java Native Interface)技术解决 ClassLoader 加密技术的内在问题。

本改进方法的核心思想是解密其他类文件的解密算法无需使用 Java 语言实现，而使用其他高级语言(如 C 语言)来实现，实现后的解密算法是个二进制文件，黑客无法轻易破解。解密算法实现后，即可在 ClassLoader 类中使用 JNI 技术调用本地解密接口，如图 1 所示。

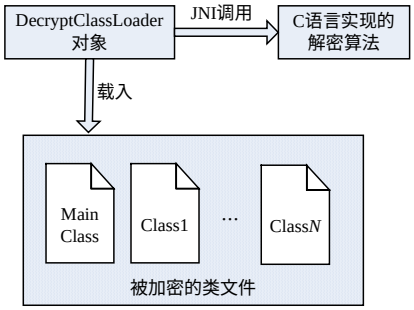

图 1 ClassLoader 类工作示意图

因为解密算法放在二进制文件中，即使 ClassLoader 类被反编译，也无法提取其中解密算法，所以就无法解密其他类文件，从而弥补了传统 ClassLoader 加密技术固有的缺陷，也将 Java 类文件的安全性提升到本地应用程序的安全强度。

3.1 ClassLoader 的定制

ClassLoader 是 Java 虚拟机的类文件加载器，专门用来载入虚拟机所需要的类。Java 虚拟机把要装入的类名告诉 ClassLoader，在默认情况下，虚拟机使用的是 System ClassLoader，System ClassLoader 根据 CLASSPATH 环境变量设置的值或者是在运行 Java 命令的时候通过-CLASSPATH 参数传入的值来查找类，并读入找到的类文件数据，然后把它转换成一个 Class 对象并交给 Java 虚拟机去运行。

当需要实现一些附加功能，如 Applet 从网络上获取类文件就需要使用定制的 NetworkClassLoader 从网络上载入所需要的类文件。本改进方法也需要实现一个定制的 DecryptClassLoader，在载入被加密的类文件时可自动解密。通过重载该类的 Class findClass(String className)函数，并在该函数使用 JNI 技术调用实现的本地解密接口即可完成类文件的载入。

3.2 解密接口的定义

在 Java 代码中，使用 JNI 技术调用本地接口需要声明一个 native 接口，接口的实现部分放在本地动态链接库中。对本改进方法使用的 Java 解密接口声明如下：

```
public native Class decryptClass(byte[] encrypedClass, String className);
```

该解密接口的参数有 2 个：encrypedClass 是加密后类文件的字节内容，className 是该类文件所代表的类名；返回值类型为 java.lang.Clsss(Java 类对象)，而不是 InputStream(类文件数据)。如果返回值类型为 InputStream，当虚拟机使用 JNI 技术调用本地解密接口时，返回值(即解密后的类文件数据)很有可能被第三方工具截留并保存下来，所以，在虚拟机和本地接口通信时存在一个安全隐患。如果返回值类型为 java.lang.Class 对象，该对象是运行时对象，即使被截留也只能看到该对象当前的一些状态，没有其他任何信息，因此，不存在虚拟机和本地接口通信时的安全问题。

3.3 解密接口的实现

在 Java 中声明了解密之后，需要用 javah 工具生成可供 C 语言开发的头文件，该头文件包括了解密接口的 C 语言定义形式：

```
JNIEXPORT jclass JNICALL DecryptClassLoader_decrypt Class (JNIEnv *, jobject, jbyteArray, jstring);
```

根据生成的头文件，即可用具体的加密算法对类文件进行解密，如 DES 算法，本文不作具体讨论。

4 结束语

采用改进的方法来保护 Java 类文件，如果黑客想窃取 Java 源代码，就不得不先反编译用 C 语言实现的的解密算法，其难度是非常大的。因此，加密后的 Java 应用程序的安全强度可以达到本地应用程序的安全强度。

参考文献

[1] 刘 劼. Java 反编译技术和代码安全[J]. 现代电子技术, 2004, 27(10): 22-24.

[2] 冀振燕. Java 编译程序技术与 Java 性能[J]. 软件学报, 2000, 11(2): 173-175.

[3] Wu Yang. Advanced Obfuscation Techniques for Java Bytecode[J]. Journal of Systems and Software Volume, 2004, 71(1): 1-3.

[4] Berghei H. Wartermarking Cyberspace[J]. Communications of the ACM, 1997, 40(11): 19-21.

[5] 张敦华, 刘 建. Java 动态类加载机制及其应用[J]. 计算机工程与设计, 2004, 25(3): 432-435.