一种Java软件水印植入虚方法的永假式构造

李 奎 陈建平 施 佺 李桂森
(南通大学计算机科学与技术学院 南通 226019)

摘 要 软件水印是近年来出现的软件版权保护技术,它通过在软件作品中嵌入版权信息(水印)来达到版权保护的目的。针对基于字节码的Java软件水印算法,提出一种永假式的构造设计方法,用于水印算法中虚方法的植入。利用Java语言的反射机制动态随机生成一个0、1字符串,对该字符串进行正反码编码和解码,得到一个始终全为0的字符串,以此作为永假式的条件,确保虚方法不被执行。永假式的构造具有良好的隐蔽性,能抵抗多种常见的水印攻击。

关键词 软件水印,虚方法,永假式,Java反射机制,正反码
中图法分类号 TP309.2 **文献标识码** A

Design of Contradiction Structure for Dummy Method Insertion in Java Software Watermarking

LI Kui CHEN Jian-ping SHI Quan LI Gui-sen
(School of Computer Science and Technology, Nantong University, Nantong 226019, China)

Abstract Software watermarking is a software copyright protection technology appeared in recent years. It achieves the purpose of copyright protection by embedding copyright information (watermark) into a software product. This paper proposed a design method of the contradiction structure for the dummy method insertion used in the bytecode based Java software watermark algorithm. The Java reflection mechanism is used to dynamically generate a random string of 0 and 1. The string is then encoded and decoded using the technique of the positive and inverse coding to obtain a string of all zeros. This string is used as the condition of the contradiction structure, which ensures that the dummy method will never be executed. The presented contradiction structure has good concealment and can resist various watermark attacks.

Keywords Software watermarking, Dummy method, Permanent false expression, Java reflection mechanism, Positive and inverse code

1 引言

软件产业近年来迅猛发展,已成为每年具有数千亿产值的重大产业。与此同时,软件产品的非法复制和盗用问题也变得非常严重,受到业内外广泛关注。如何保护软件产品的知识产权,防止软件被非法复制和盗用,是信息安全领域的一个重要研究课题。软件水印是近年来出现的保护软件作品知识产权的技术[1-3],它在不影响程序功能的前提下在软件作品中嵌入版权保护信息或身份认证信息(即水印),当程序遭到非法复制或盗用时,可以通过提取这些信息来证明软件作品的所有权。

目前,在各种软件产品中,用Java语言开发的占有很大的比重。Java语言具有跨平台的可移植性,带来使用上的便利,同时也带来程序容易被复制和盗用的问题。一些开发者可以在自己开发的程序中未经授权地使用他人开发的类文件,也可以通过反编译,从类文件获得源文件,学习他人解决某个问题的方法。因此,研究和开发Java语言软件的版权保护技术具有十分重要的现实意义和应用价值。

关于Java软件水印技术,已有人提出了一些方案和算法[4-6]。其中,Monden和Iida等人[7]提出了一种基于字节码的Java软件水印算法。其基本思想是,在需要保护的Java程序中植入永不执行的方法(称为虚方法),然后在这个虚方法对应类文件的字节码中,对指令的操作数进行修改或者对指令的操作码进行编码,以此来嵌入水印信息。该算法具有较好的实用性和可实现性,王春红、陈建平等人[8]在Monden等提出的算法的基础上进行了进一步的研究,并对算法进行了实现。到目前为止,对该算法研究的重点是如何在虚方法中嵌入和提取水印信息。对虚方法如何植入,如何构造有效的使得虚方法不被执行的永假结构尚未涉及。而这个问题不解决,虚方法就不能得到很好的隐藏,容易受到攻击,水印算法也就无法得到真正应用。本文针对这一问题提出解决方案。

2 基于字节码的Java软件水印算法

本节对基于字节码的Java软件水印算法做一简单介绍,

到稿日期:2013-07-10 返修日期:2013-09-03 本文受国家自然科学基金面上项目(61171132),江苏省自然科学基金项目(BK2010280),南通市应用研究计划项目(BK2011026)资助。

李 奎(1988—),男,硕士生,主要研究方向为软件水印;陈建平(1960—),男,教授,主要研究方向为信息安全,E-mail:chen.jp@ntu.edu.cn(通信作者);施 佺(1973—),男,教授,主要研究方向为社会网络;李桂森(1989—),男,硕士生,主要研究方向为文本水印。

说明永假式构造的重要性。水印算法的流程如图1所示。首先将一个虚方法植入到需要保护的Java源程序中。所谓虚方法是指这个方法实际不会被执行。对包含虚方法的Java源程序进行编译,得到由字节码构成的Java类文件。在类文件中找到虚方法的代码所对应的指令,通过一定的方式修改这些指令的操作数或编码这些指令的操作码,来嵌入水印信息。水印的提取则是通过检测类文件中的虚方法的指令,提取相关操作数或操作码来实现。具体如何嵌入和提取水印,不是本文的重点,此处不再详细介绍,可参见文献[7,8]。

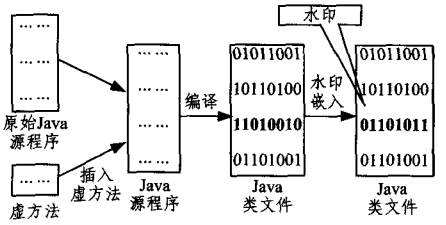

图1 基于字节码的Java软件水印算法流程

如上所述,要嵌入水印信息,需要在Java源程序中植入一个永不执行的虚方法。为了让虚方法看起来不虚,需要增加一个虚方法的调用,其一般结构如下所示:

```
If (condition){
  dummy();
}
```

其中,condition是一个永假的表达式,即该表达式的值永远不会为true或1。dummy()即为欲植入的虚方法。由于condition永远不成立,因此方法dummy()永远不会被执行。为了使虚方法不被轻易识破,构建的永假式不能太简单和直接。否则,攻击者通过一些简单的调试工具便可以发现这个永不执行的方法,将其删除或破坏,使得嵌入的水印信息丢失或无法正确提取。因此,永假式的构造既要有一定复杂性和合理性,又要保证该表达式在任何时候均不会成立。

3 永假式构造的基本方法

本文结合Java语言的技术特点,提出一种永假式的设计构造方法。设计的基本思想是,利用Java语言的反射机制,动态随机产生一个由0、1组成的字符串,运用通信原理中的正反码编码技术,对字符串进行编码和解码,最终得到一个始终全为0的字符串,以此作为永假式的判断条件,使得条件永远不会成立,从而虚方法不被执行。

Java语言的反射机制是指,在程序运行状态中,对任何一个类,都可以获取这个类的所有方法和属性。使用Java反射机制在实际Java程序中很常见,如读取XML配置文件等,不会引起怀疑。一个实际Java程序的类中通常都会含有几个以上的方法,在程序执行期间,可以利用Java反射机制获得类中的这些方法名,针对每个方法名,用随机函数产生一个0或1,形成一个0、1字符串。该0、1串是在程序执行时产生的,其位数不固定,取决于被保护程序中的方法的个数,每一位的值是随机的,可能是0,也可能是1,具有很好的不确定性和隐蔽性。下一步需要设法将这个随机0、1串在任何情况下都变成一个确定值。这里,我们采用通信技术中正反码编码的方法。

正反码是通信技术中一种简单的用于纠正错码的编码[9],由信息位和监督位两部分组成。其中,监督位根据信息位产生,其值是信息位的简单重复或者是信息位的逐位取反,具体由信息位中"1"的个数决定。当信息位中有奇数个"1"时,监督位是信息位的重复;当信息位有偶数个"1"时,监督位是信息位的反码。例如,若信息位为11001,则监督位为11001,编码码组为1100111001;若信息位为10001,则监督位为01110,码组为1000101110。

正反码接收端解码的方法为,将接收码组中的前一半(即信息位)和后一半(即监督位)按模2相加,得到一个合成码组。由此合成码组产生一个校验码组。若接收码组中有偶数个"1",则校验码组与合成码组相同;若接收码组中有奇数个"1",则取合成码组的反码作为校验码组。最终得到的校验码组的各位全为0。

由上述正反码的编码和解码过程可知,不管原始信息位的个数有多少及各位0、1如何构成,只要编码和解码步骤正确,最终产生的校验码组的各位一定全为0。因此,可以将上面随机产生的0、1串作为正反码的信息位,通过正反码的编码和解码,将最终产生的校验码组作为永假式的条件。

4 永假式构造的具体实现

由于Java程序通常以类文件的形式发布和使用,下面以Java程序的类为保护对象进行设计,即在需要进行版权保护的某个类中植入虚方法,嵌入水印信息。为说明问题,每一步给出必要的核心代码。

第1步 利用Java反射机制解析出需要保护的类中的所有方法名,并把这些方法名存储在一个数组中。实现代码如下(其中数组取名为methods):

```
Class<?> class=Class. forName("所要保护的Java程序的类名");
  Method []methods=class. getDeclaredMethods();
```

第2步 迭代这个数组,针对每个数组元素,用随机函数生成1个"0"或"1",将产生的0、1拼接成一个0、1字符串。相关代码如下:

```
for(Method method:methods) {
  sb. append(new Random(). nextInt(2));
}
```

选择通过随机函数生成0、1码,而不是直接赋值"0"或"1",原因是通过随机函数赋值具有高度的通用性和隐蔽性,即使使用单步调试工具,也不易发现其中的赋值必然性,不会引起怀疑。

第3步 对0、1字符串进行正反码编码。根据前述编码原理,将上面得到的0、1串作为正反码的信息位,根据信息位产生监督位。最终得到编码后的0、1串。该编码过程用一专门的方法完成,与实现解码的方法一起放在另一个独立的类中,而不置于本类,以提高隐蔽性。实现编码和解码的具体代码没有什么特殊性,不再列出。

第4步 构建隐藏虚方法的条件表达式。为了进一步提高永假式构造的复杂性和虚方法的隐蔽性,我们设计了二重if语句的结构。第一个if语句用于对第3步得到的编码后的字符串进行解码处理。代码如下:

```
if(str. contains("0") || str. contains("1")){
  str=Code. decode(str);
}
```

其中，str 为编码后的字符串，decode()是对字符串进行解码操作的方法。将编码后的字符串是否含有“0”或“1”作为 if 语句的执行条件。此条件始终成立，因此执行解码过程。将解码后得到的结果(即校验码组)重新赋给原字符串，此时该字符串的各位必定全为 0。

第 5 步　构建第二个 if 语句，即永假式语句，虚方法隐藏于此 if 语句中。将上述解码得到的字符串是否包含“1”作为 if 语句的执行条件，语句结构如下：

```
if(str. contains("1")){
  dummy();
}else{
  func();
}
```

其中，dummy()是植入的虚方法，func()是类中原有的某个方法。由于解码后的字符串必定不含“1”，因此条件永远不成立，dummy()方法不会被执行，转而执行程序中原应执行的方法 func()。

至此，完成了永假式的构造和虚方法的植入。将上述整个过程用一个方法表示，如取名 insert_dummy()，则只需将原程序中对 func()方法的调用换成调用 insert_dummy()方法即可。

5　实验测试

实验中假定需要保护的 Java 程序的类名为 Protected_Class，其中含有 5 个方法，分别为 func1()－func5()，其具体功能在本测试中不重要，仅以打印语句表示，输出相应的方法名。同样，植入的虚方法中用于嵌入水印信息的代码也不是本文的重点，也以打印语句代替。可以选择这 5 个方法中的任何一个与虚方法进行整合，构成上节讨论的 insert_dummy()方法。此处选择 func5()，相应程序的主要代码如下：

```
package org. experiment;
public class Protected_Class{
    public int func1(){
      System. out. println("func1");
      return 1;
    }
      ⋮
       //func2()－func4()类似，此处省略
    public int func5(){
      System. out. println("func5");//输出方法名
      return 1;
    }
    public void dummy(){//虚方法
      System. out. println("dummy");
    }
    public void insert_dummy(){//植入虚方法
      List<String> lst=new ArrayList<String>();
      StringBuffer sb=new StringBuffer();
       Class<?> c = Class. forName("org. experiment. Protected_
                    Class");
    Method []methods=c. getDeclaredMethods();
    for(Method method:methods) {
      sb. append(new Random(). nextInt(2));
    }
    String str=Code. encode(sb. toString());
      if(str. contains("0") || str. contains("1")){
      str=Code. decode(str);
    }
    if(str. contains("1")){
      dummy(); //调用虚方法
    }else{
      func5();//调用 func5()
    }
  }
}
```

测试的主程序如下：

```
import org. experiment;
public class Test{
  public void main(String[]args) {
    Protected_Class. insert_dummy();
    Protected_Class. func5();
  }
}
```

程序运行结果如图 2 所示。图中，输出的第一个 func5 为调用植入虚方法的方法 insert_dummy()得到的结果，输出的第二个 func5 为直接调用 func5()方法得到的结果，两者相同，说明虚方法没有执行，实际执行的是 func5()，测试结果正确。在测试中，我们还特别输出了通过 Java 反射机制动态随机产生的原始字符串和经过正反码编码和解码后的字符串。前者结果为 1100100，串长 7 位(类中原有 5 个方法 func1()－func5()，加上添加的两个方法 insert_dummy()和 dummy()，共 7 个方法，故为 7 位)，各位由 0 和 1 组成。编解码后的字符串为 0000000，各位全为 0，结果正确。前面提到，通过 Java 反射机制产生的 0、1 串是随机生成的，每次程序运行都可能不一样。我们对上述程序反复运行多次，尽管产生的初始 0、1 串各不相同，但编解码后的结果均全为 0。程序每次运行，虚方法 dummy()均不执行，执行的是程序本应执行的方法 func5()，结果正确。

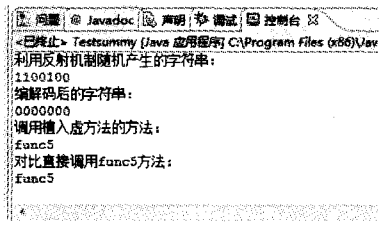

图 2　测试程序输出结果

6　性能分析

上述永假式构造由于采用 Java 反射机制、随机产生 0、1 串、正反码编解码、多重 if 语句结构等方法，具有很好的隐蔽性，不易被察觉和发现。同时可以抵抗多种常见的针对软件水印的攻击，包括语义变换攻击、统计攻击、代码增减攻击和单步跟踪调试攻击等，具体讨论如下。

设计是利用类中的方法名产生一个 0、1 串，对该 0、1 串

进行编码和解码，这一过程不与任何特定的代码形式或它们的执行顺序有关。因此，即使通过语义变换，改变代码的形式或顺序，只要功能不变，上述永假式的构造不会遭到破坏。同时，0、1 串是用随机函数产生的，每次运行都不一样，无法穷举，所以无法确切地统计相关信息，以找到规律发现虚方法的存在，因而能抵抗统计攻击和单步跟踪调试攻击。与语义变换攻击一样，在程序中增加其它代码对本方法不会产生任何影响。对于减少代码攻击，如果减去的不是与永假式构造有关的代码，则不会对本程序产生影响；如果减去的代码与 0、1 串编解码的过程有关，由于不能正确进行编解码，将会导致程序出错，无法正常运行，这对于攻击者而言也没有意义，因为攻击者的目的是要盗用程序，若盗取的程序无法完成预定的功能，攻击便失去意义和效果。

结束语 本文针对基于字节码的 Java 软件水印算法，提出了一种永假式的设计构造方法，用于植入虚方法进行水印信息的嵌入，有效地解决了虚方法的隐藏和抗攻击问题，使得该水印方案更加完备，可实际应用于 Java 程序的版权保护。根据本文提出的方法以及相关水印嵌入、提取技术，我们已设计开发出用于对实际 Java 应用程序进行版权信息嵌入与提取的软件系统。

参 考 文 献

[1] Collberg C, Thomborson C. Watermarking, tamper-proofing, and Obfuscation—Tools for Software Protection[J]. IEEE Transactions on Software Engineering, 2002, 28(8): 735-746

[2] 张立和，杨义先，钮心忻. 软件水印综述[J]. 软件学报，2003，14(2)：268-277

[3] Zhu W, Thomborson C, Wang F. A Survey of Software Watermarking[C] // IEEE International Conference on Intelligence and Security Informatics. 2005: 454-458

[4] Hamilton J, Danicic S. A survey of static software watermarking [C]//IEEE World Congress on Internet Security. 2011: 100-107

[5] 鲍福良，彭俊艳，方志刚. Java 类文件保护方法综述[J]. 计算机系统应用，2007，6：124-126

[6] 周正虎，陈丹，周光霞，等. 基于病毒多态性的 Java 软件水印技术[J]. 计算机与数字工程，2011，39(11)：97-100

[7] Monden A, Iida H, Matsumoto K, et al. A Practical Method for Watermarking Java Programs [C] // The 24th International Computer Software and Applications Conference. 2000: 191-197

[8] 王春红，陈建平，王杰华，等. 基于字节码的 Java 软件水印的研究与实现[J]. 微电子学与计算机，2009，26(9)：146-149

[9] 樊昌信，曹丽娜. 通信原理[M]. 北京：国防工业出版社，2010

(上接第 199 页)

网络应用下的安全性能，本文设计如下仿真实验进行验证。首先，构造 500 对验证和应答请求，其中包含了协议所定义的参数极限值。然后，用 3 种协议分别执行这些验证数据，统计在 3 种网络应用环境下的验证正确率，由仿真结果可知，本文协议的正确率与文献[7]相当，较文献[1]则安全很多。协议安全性能仿真结果如图 3 所示。

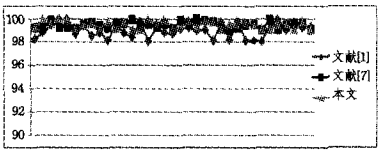

图 3 协议安全性能仿真结果

结束语 匿名身份认证对保护用户的隐私和信息安全发挥越来越重要的作用，本文提出了一个新的基于使用数字签名的零知识证明的匿名身份认证方案，通过引入中间第三方代理来完成服务器与用户之间的通信。该方案不需要知道所有用户的公钥，大大减少了通信量，同时很好地消除了安全漏洞，具有良好的匿名性、认证性和不可关联性。文中还对该协议的安全性和计算效率进行了实验仿真。经验证，该协议的安全性能优且较其他协议效率更高。

参 考 文 献

[1] Lee W B, Chang C C. The protocols guarantee user anonymity in distributed network user authentication and key distribution[J]. Computer Systems Science and Engineering, 1999, 15(4): 113-116

[2] Wu T S, Hsu C L. Efficient user identification scheme with key distribution preserving anonymity for distributed computer networks[J]. Computers and Security, 2004, 23(2): 120-125

[3] Viet D Q, Yamamura A, Tanaka H. Anonymous authenticated key exchange protocol based on password: Advances in Cryptology INDOCRYPT, 2005[C]//LNCS, Vol. 3797. Berlin: Spring-Verlag, 2005: 244-257

[4] Yang Jing, Zhang Zhen-feng. New anonymous password-based authenticated key exchange protocol[C]//Chowdhury D R, Rijmen V, Das A, eds. INDOCRYPT, 2008. LNCS 5365, 2008: 200-212

[5] Cui Hui, Cao Tian-jie. A new anonymous identity authentication and key exchange protocols[J]. Journal of Network, 2003, 4(10): 985-992

[6] Bo Z, Wan Z G, et al. Anonymous secure routing for mobile ad hoc networks[C] // 29th Annual IEEE International Conference, 2004. 2004: 102-108

[7] Chien H Y, Chen C H. Remote authentication mechanism of guarantee user anonymity[C]//Proceedings of the 19th International Conference on Advanced Information Networking and Applications-AINA, 2005. 2005: 245-248

[8] Durresi A. Anonymous communications in the Internet [J]. Cluster Computing, 2007, 10(1): 57-66

[9] 王尚平，王育民，王晓峰，等. DSA 数字签名的零知识证明[J]. 电子学报，2004，32(5)：878-880

[10] Cesena E, Löhr H, Ramunno G, et al. Anonymous authentication with TLS and DAA[C]//Proceedings of the Third International Conference on Trust and Trustworthy Computing (TRUST'10), 2008. LNCS, vol. 6101, 2008: 47-62

[11] Chen L, Page D, et al. On the design and implementation of an efficient DAA scheme[C]//9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application (CARDIS'10), 2010. LNCS, vol. 6035, 2010: 223-237