IJETST Open access Journal International Journal of Emerging Trends in Science and Technology

Software Protection against Piracy and Reverse Engineering using Software Watermarking Technique

Authors

Dnyaneshwar Karale, Archana Tapase, Prof. Bhushan S. Chaudhari (Guide)

Department of Information Technology Engineering, Sandip Institute of Technology and Research Centre
Nashik, Maharashtra
E-mail: *vickykarale@gmail.com*

Abstract

The rise in use of Internet and byte code languages such as Java byte code and Microsoft's Common Intermediate language have made copying, decompiling and disassembling software easier with the rapid development of Internet and Software Industry. The issue of software piracy and security is of great concern. Software watermarking is a new technique appeared in recent years for software copyright protection. It embeds some secret information (watermark) into software as an identifier of the ownership of copyright for the software. We are going to implement a software watermarking algorithm that is effective for the copyright protection of java program, especially for java class files.

Index Terms - *software watermarking, byte code, software copyright, protection, java program, class files.*

Introduction

SOFTWARE WATERMARKING is a method of software protection by embedding the secret information into the text of software. We insert such secret information to show ownership of the software. This enables the copyright holders to establish the ownership of the software by extracting this secret message from an unauthorized copy of this software when an unauthorized use of this software occurs. Software watermarking can be regarded as a branch of digital watermarking, which started about 1954. Since the publication of a seminal work by Tanaka et al, in 1990 [4], digital watermarking has made considerable progress and become a popular technique for copyright protection of information.

Research on software watermarking started in the 1990s. The patent by Davidson and Myhrvold presented the first published software watermarking algorithm. The preliminary concepts of software watermarking also appeared in paper [5] and patents [6].

1.1 Overview

1.1.1 What is Watermark Software?

Software Watermarking does not prevent theft but instead discourages software thieves by providing a means to identify the owner of a piece of software. It can then be extracted by an extractor or verified by a recognizer to prove ownership of software. The former extracts the original watermark, while the latter merely confirm the presence of a watermark. Watermark recognition

Or extraction algorithm may also classify as blind, where the original program and watermark is unavailable, or informed.

1.1.2 What is Need of Watermark?

Software Theft:

Software theft, also known as software piracy, is the act of copying a legitimate application and

1.1.3 How to Choose a Watermark?

Watermark software should have the following characteristics:

1) Simple operation, clear steps, anyone can easily use it, not as complicated as Photoshop, because the watermark software's customers are not professionals.

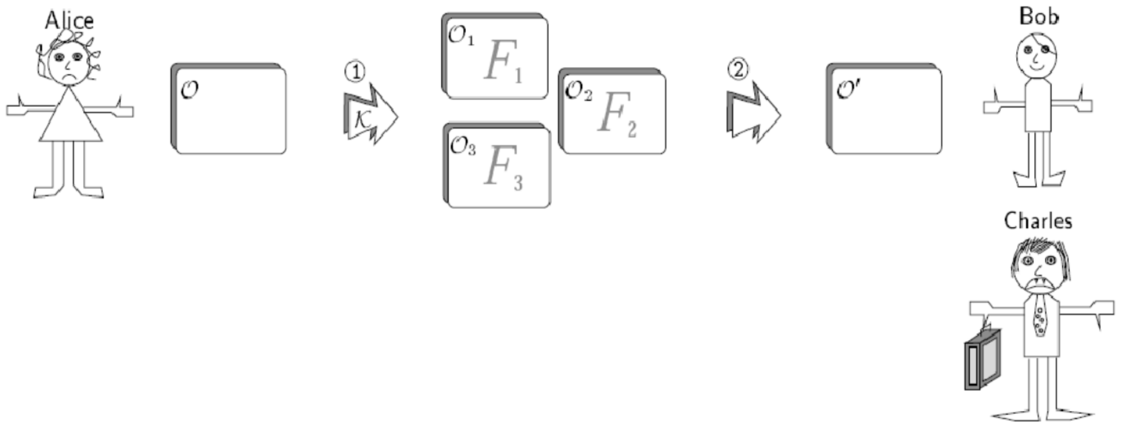

Fig.1. before Watermark

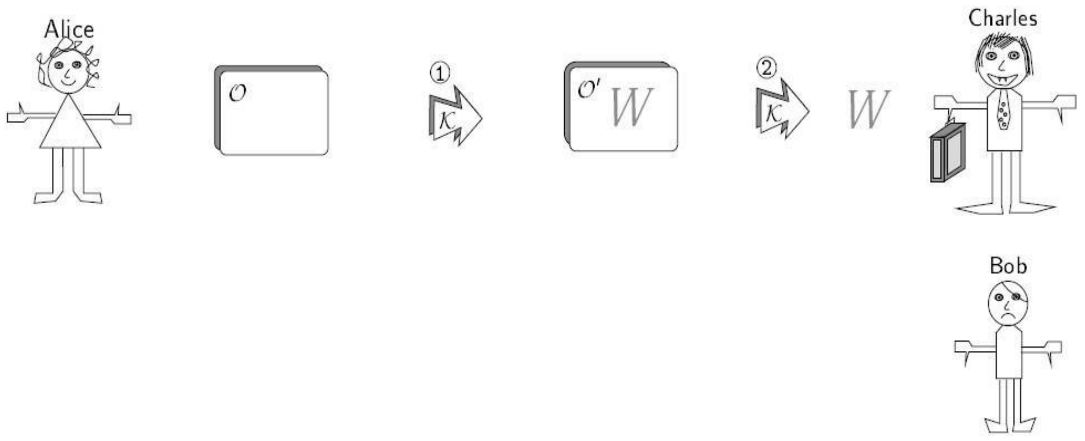

Fig.2. after Watermark

illegally distributing that software, either free or for profit. Legal methods to protect software producers such as copyright laws, patents and license agreements do not always dissuade people from stealing software, especially in emerging markets where the price of software is high and incomes are low.

2) The operation process distinguishes between watermarks, because there are different watermark, and different people have different needs.

3) Can save user-defined watermark mode for the next use, including the watermark itself, location, transparency and all the information, to ensure the next directly use when necessary.

1.1.4 Watermarking Algorithms

Add Expression:

Adds a bogus addition expression containing the watermark to a class file.

Add Initialization:

Adds bogus local variables to a method in a class file.

Add Method and Field:

splits a watermarking two - one half stored in the name of a bogus field, the other half store in the name of a bogus method.

1.1.5 Code Obfuscation

Code obfuscation increases the difficulty of reverse engineering. A scheme of zero-watermarking is one which embeds the watermarking information into programs without additional codes. The watermarking information is hidden in the program when the process of code obfuscation is executed. The Combination between software watermarking and code obfuscation would make the reverse engineering more difficult and the software products more secure [1]. Four principal types of software obfuscations are design obfuscation, data obfuscation, control obfuscation, and layout obfuscation.

1.1.6 System Architecture

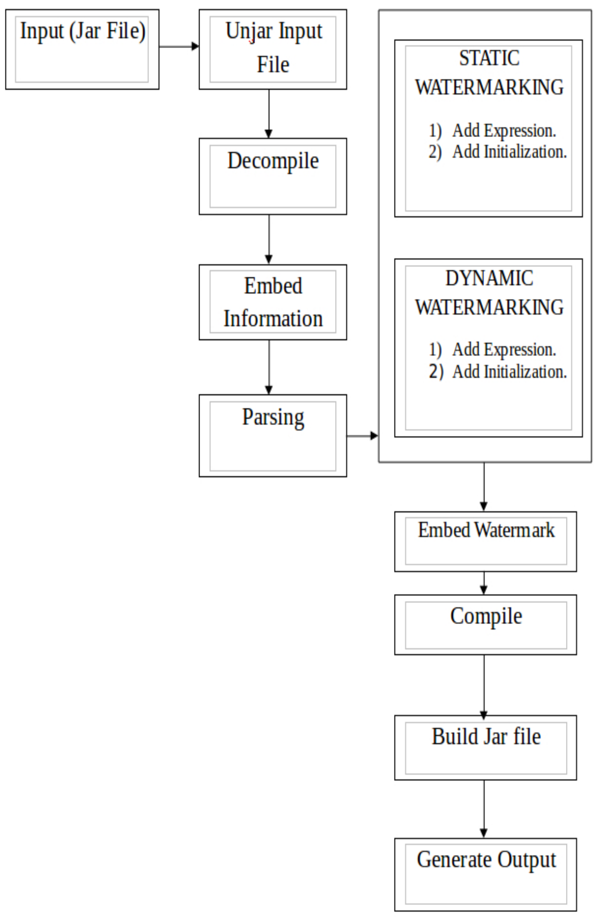

Fig.3. Embed Watermark

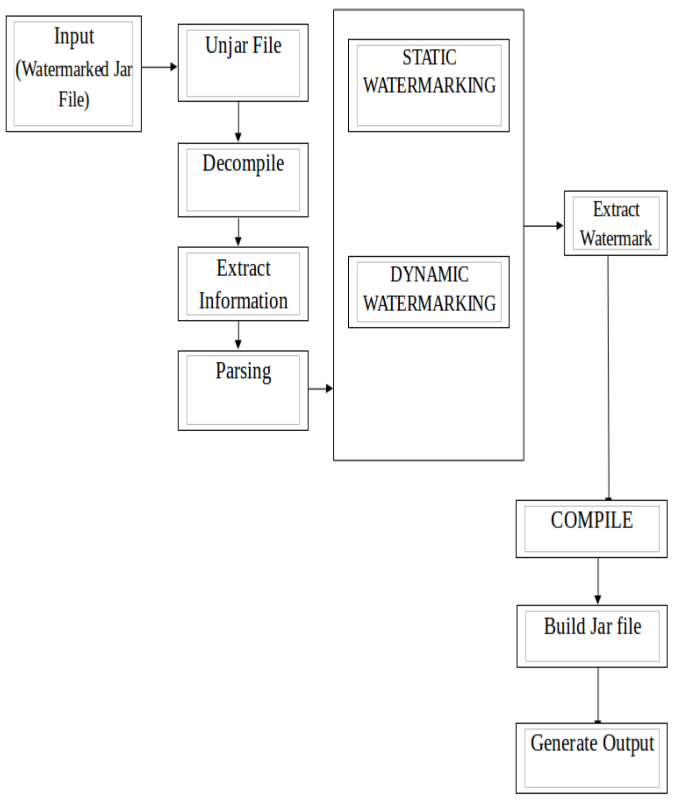

Fig.4. Extract Watermark

1.2 Taxonomy of Software Watermark

We can classify software watermarks in different ways by their functions or properties. The following are some classification schemes for publishing literature:

1.2.1 Classification by Purpose

Software watermarks can be classified by their functional goals [7].

Authorship Mark: identifying a software author, or authors. These watermarks are generally visible and robust.

Fingerprinting Mark: identifying the channel of distribution, i.e., the person who leaked the software, the watermarks are generally invisible, robust and consist of a unique identifier such as a customer reference number.

Validation Mark: to verify that software is genuine and unchanged, for example like digitally signed Java Applets. These watermarks must be visible to the end-user to allow validation and fragile to ensure the software is not tampered with.

1.2.2 Classification by Extracting Technique

Classification of software watermark by the extracting technique falls into two classes: [8].

Static Watermark: A static software watermark is one inserted in the data area or the text of codes. The extraction of such watermarks does not run the software. Generally, there are two types of static watermarks [8]: data watermarks and code watermarks. A data watermark is inserted directly into the data area of a program, while a code watermark is inserted into the code area of a program. A simple code watermark involves a permutation of the order of some instructions in a program.

Dynamic Watermark: A dynamic software watermark is one inserted in the execution state of a software object. More precisely, in dynamic software watermarking, what has been embedded is not the watermark itself but some codes which cause the watermark to be expressed, or extracted, when the software is run. An example is the dynamic data structure watermark proposed by Collberg and Thomborson [9]. Dynamic software watermarks come in three types: Dynamic Easter egg watermarks, dynamic execution trace watermarks, and dynamic data structure watermark.

1.2.3 Research Platforms

The following four systems research platforms for software watermarking:

JavaWiz [10], Hydan [11], UWStego [12], and Sand- Mark [13].

JavaWiz is software watermarking system developed at Purdue University. It can watermark Java source programs, and it is written entirely in Java. This system has implemented the CT algorithm.

Hydan is software watermarking system developed at Columbia University. It is used to watermark an executable.

UW Stego is software watermarking research tool developed at the University of Wisconsin for experimenting and testing various software watermarking techniques and has a toolset for developing new software watermarking algorithms.

Lastly, Sand Mark is a comprehensive research tool for software watermarking and obfuscation developed at the University of Arizona. It can be used to measure the effectiveness of software watermarking algorithms. Like Java Wiz, this platform is written in Java.

1.3 Project Scope

Due to rapid progress of computer and java byte-code language, we are becoming more and more capable of enhancing the degree of security. Current watermarking research depends on a secret key for security, hiding the watermark for protection. At the time of selling the software, thefts can try to make illegal copies of our software. So there are chances of confidential Information or being snooped. Our system works as follows:

Cover file + watermark key = target file.

1.4 Future Work

Further work will involve extending the evaluation to dynamic watermarks which, in theory, should be resilient to semantics preserving transformations. However, it has been shown that at least one dynamic algorithm is only minimally stronger than the static version. We intend to investigate this claim and extend the investigation to evaluate other dynamic watermarking algorithms and their advantages over static algorithms.

CONCLUSION

Software watermarking is the process of embedding a large number into a program. This is a challenging problem that, to the best of our knowledge, has not previously been addressed in the academic literature. The few published accounts of which we are aware (mostly software patents) describe schemes in which watermarks or fingerprints are embedded in the object code of a program. These static techniques are susceptible to attacks such as translation, optimization, or obfuscation. The Combination between software watermarking and code obfuscation would make the reverse engineering more difficult and the software products more secure

REFERENCES

1. Guangxing Xu and Guangli Xiang, “A Method of Software Watermarking”, 2012 International Conference on Systems and Informatics (ICSAI 2012).
2. James Hamilton and Sebastian Danicic, “An Evaluation of the Resilience of Static Java Bytecode Watermarks Against Distortive Attacks”, IAENG International Journal of Computer Science, 38:1, 2011.
3. Christian S. Collberg and Clark Thomborson “Watermarking, Tamper-Proofing, and Obfuscation Tools for Software Protection”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 8, AUGUST 2002.
4. William Feng Zhu “Concepts and Techniques in Software Watermarking and Obfuscation”, a thesis on Doctor of Philosophy in Computer Science, August 2007.
5. H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, “Design and evaluation of birthmarks for detecting theft of java programs”, in Proc. IASTED International Conference on Software Engineering (IASTED SE2004), Feb 2004, pp. 569- 57.
6. D. Grover “The Protection of Computer Software - Its Technology and Applications”, 2nd ed. Cambridge University Press, 1997.
7. J. Nagra, C. Thomborson, and C. Collberg “A functional taxonomy for software watermarking”, in Twenty-Fifth Australasian Computer Science Conference (ACSC2002), M. J. Oudshoorn, Ed. Melbourne, Australia: ACS, 2002.
8. Y. He Tamper proofing a software watermark by encoding constants, Master’s thesis, University of Auckland, Mar 2002.
9. C. Collberg and C. Thomborson, “Software watermarking: Models and dynamic embeddings”, in Proceedings of Symposium on Principles of Programming Languages, POPL 99, 1999, pp. 311324.
10. L. Zhang, Y. Yang, X. Niu, and S. Niu, “A survey on software watermarking”, Journal of Software, vol. 14, no. 2, pp. 268-277, 2003.
11. R. El-Khalil and A. Keromytis, “Hydan: Embedding secrets in program binaries”, Aug. 14 2004.

12. C. Collberg, S. Jha, D. Tomko, and H. Wang, "Uwstego: A general architecture for software watermarking", Technical Report TR04-11.
13. C. Collberg, G. Myles, and A. Huntwork, "Sandmark - a tool for software protection research", IEEE Security and Privacy, vol. 1, no. 4, pp. 40-49, 2003.
14. A. Majumdar and C. Thomborson, "On the use of opaque predicates in mobile agent code obfuscation," in ISI 2005, ser. LNCS, vol. 3495, May 2005, pp. 648–649.