EXPERIENCE WITH SOFTWARE WATERMARKING

K.K.Savitha M.C.A., M.Phil., Lecturer, K.S.R. College of Engineering, Tiruchengode.

Abstract

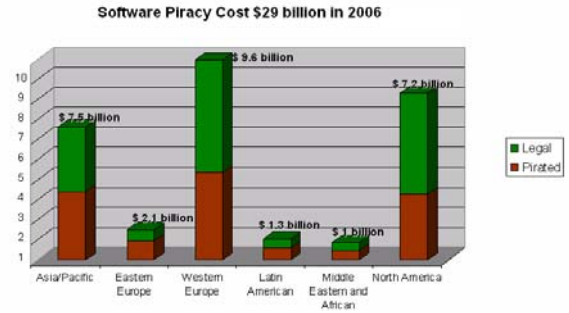
Software piracy is a major concern for software providers, despite the many defense mechanisms that have been proposed to prevent it. Watermarking embeds a secret message into a cover message. In media watermarking the secret is usually a copyright notice and the cover a digital image. Watermarking an object discourages intellectual property theft, or when such theft has occurred, allows us to prove ownership.

Introduction

Cryptography and steganography have been used throughout history as means to add secrecy to communications during times of war and peace [2]. Some of the early methods to hide information include text written on wax-covered tablets, using invisible ink and shaving the head of a messenger and tattooing the message on the scalp.

Software piracy became a major problem with the fast and vast growth in the use of the Internet. Moreover, the new computer technologies such as the ability to copy CDs fast and easily aided in increasing software piracy. In fact, many people consider software too expensive to buy and they lack the respect and enforcement of intellectual properties laws. Specialists believe that there is no technique that can prevent all kinds of software piracy though the effort on preventing software piracy is continuing, and hence, the goal is to raise the cost for software pirates.

According to the Business Software Alliance organization (BSA), software piracy caused a loss of nearly $29 billion in year 2005 only, which is 36% of software installed on computers worldwide in the same year.



The Need to Prove Software Ownership

Suppose Alice has built some software and now wants to sell it for profit. She copyrights the software, but Bob still manages to make a pirate copy. Bob may be interested in the software for several reasons, including 1) private use, 2) industrial espionage, and 3) further selling for his own profit. In the first case, Alice's profits may suffer; in the second case, some of her algorithmic techniques may be found out; and in the third case, she may risk that a competitor sells her own software, perhaps somewhat modified. This leaves the question:

Question: How does Alice protect her copyright?

Answer: She must be able to prove ownership, possibly in a court of law, of a given copy of the software.

If it is known that Alice is capable of proving ownership of her software, then this capability may help deter theft.

What is Software Watermarking?

A technique used to aid in the prevention of software piracy. The idea is to embed a message w (the \watermark") into a program P, such that w uniquely identi_es the owner of P (w is a copyright notice) or the purchaser of P (w is a fingerprint).

Watermarking has been proposed as a promising approach to controlling piracy. *Digital watermarking* is the process of embedding a small amount of identifying information in images,audio, video, software and other media [3] without significant deterioration in the user perceived content of media. *Software watermarking*, in contrast to *media watermarking*, faces a distinct challenge because of a program's executable nature. The embedding of a watermark must maintain a program's behavioural (dynamic) semantics in addition to its syntactic (static) correctness.

A program is watermarked using semantic-preserving transformations which embed a watermark into the code or data sections of a program. [3] This watermark may encode information such as the identity of the author, the program's date of publication, a copyright notice, or a *fingerprint*. A fingerprint is a unique watermark in each copy of the program. It makes individual copies of a program distinguishable and can help trace the distribution of a pirated copy to its source.

Watermarking Classifications

According to [4], we can classify different software watermarking functions as follows:

1- $embed(P;w;key) \Longrightarrow Pw$

By using a secret *key* and embedding a watermark *w*, we can transform a program *P* into *Pw*.

2- $extract(Pw;key) \Longrightarrow w$

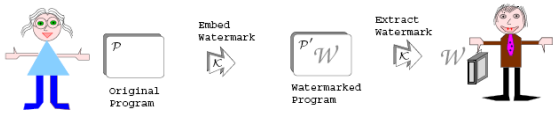
With the function *extract*, we can get the watermark *w* from *Pw*.

3- *recognize*(*Pw*;*key*;*w*) ⟹ [0:0;1:0]

If we use the *recognize* function, the returned value could reflect the probability of the existence of the watermark *w* in *P*.

4- *attack*(*Pw*) ⟹ *P'w*

The purpose of *attack* function is to alter the program *Pw* such that *w* can no longer be extracted.



Attacks on Software Watermarking

To evaluate the quality of a watermarking scheme we must also know how well it stands up to different types of attacks[5]. Software watermarking techniques are subject to several attacks such as:

Subtractive attacks: by locating and eliminating the existence of the watermark; like in removing static dead code or code protected by opaque predicates. Preventing subtractive attacks can be done by making the program execution dependent on the watermark existence such that without the watermark the program will not be usable.

Distortive attacks: by applying code transformations to make extracting the watermark more difficult; like running obfuscation and optimization techniques. However, those techniques can be considered, sometimes, as supportive to hide the watermark. Distortive attacks can be prevented by considering only part of the code or program data to extract the watermark. Yet, this does not guarantee that Distortive attacks became ineffective.

Additive attacks: by adding a new watermark to trim the value of the original watermark. To prevent such attacks, the original uniquely watermarked program can be compared with the attacked one.

Collusive attacks: by comparing different versions of the program where each is having different fingerprint. This can be done by monitoring program execution or statically analyze two or more versions of the programs. This attack can be prevented by allowing embedding more than one watermark. Also, one common watermark can be embedded in all the versions. In addition, more than one transformation can be applied to different versions to complicate the comparison process [4]. A main threat to watermarking schemes is the meaning-preserving transformation attack. This threat does not really count in media watermarking schemes [4]. A basic attack to protections approaches that use tokens will try to locate and remove the code that checks for the token existence by using a debugging tool. Also, the code that checks for license violation can be searched and removed. The target of software vendors using such approaches is to increase the cost of doing reverse engineering to their product [3].

Why use software watermarking?

- Discourages illegal copying and redistribution.
- If we embed copyright notice software watermarking can be used to provide proof of ownership.
- If we embed a fingerprint it can be used to trace the source of the illegal redistribution.
- Does not prevent illegal copying and redistribution.

Categories

- ***Static***: the watermark is stored directly in the data or code sections of a native executable or class file.
- ***Dynamic***: the watermark is stored in the run-time structures of the program.

Static Watermarking

Static watermarks are stored in any section of the Java class file. Two static watermarking types are basically: **code watermarks**, in the executable instructions, and **data watermarks**, such as headers, and string sections [2] [3].

Data watermarks are easy to embed and extract, and hence, considered to be common. However, it is very vulnerable to distortive attacks such as obfuscation. For example, by splitting all strings into scattered substrings or by converting static data into a program distortion will happen [2] [3].

Code watermarks contain redundant information, similar to media watermarks having embedding in redundant bits. There are many simple distortive de-watermarking attacks and code obfuscation techniques that attack code watermarks. For example, flow-of-control can be destroyed by inserting predicated branches to break the basic block order [2] [3]. The watermark is stored in the program itself as data or code. It can then be extracted directly from the program without executing it [4].

```
char V;
switch e{
case 1 : V = 'G '
case 2 : V = ' I '
case 3 : V = 'N'
case 4 : V = 'G '
case 5 : V = 'E '
case 6 : V = 'R '
}
```

Make use of the features of an application that are available at compile-time. Java application: constant pool table, method names, instruction sequence.

Advantages:
There are a variety of locations to embed a watermark. Fairly simple to modify these features and still maintain the semantics of the application.

Dynamic Watermark

In dynamic watermarks, the watermark is stored in the execution (behavior) of the program and not in the program itself. Therefore, dynamic watermarks have fewer threats of obfuscation transformations [2] [3].

Basically, there are three kinds of dynamic watermarks: Data Structure Watermark, Execution Trace Watermark, and Easter Egg Watermark. Those three methods differ in the way of storing and extracting the watermark. However, in all of the three methods, the application runs with a predetermined input sequence to enter the watermark state [2] [3].

Three common dynamic watermarks are:

- Dynamic Easter Egg Watermarks
- Dynamic Execution Trace Watermarks
- Dynamic Data Structure Watermarks.

Dynamic Easter Egg Watermarks is trivial and easily noticeable by the user since it basically displays the watermarking message or image after entering a certain input.

A piece of code that gets activated for a highly unusual input to the application. The watermark is generally immediately perceptible by the user. Typically the watermark displays a copyright message or an unexpected image on the screen.

If the watermark is extracted by tracing the addresses or instructions while executing the program, with a special input, then it is called **Dynamic Execution Trace Watermarks**, which embeds the watermark within the trace of the application as it is executed with a special input sequence. This differs from the data structure watermark in that the watermark is embedded in the application's instructions or address instead of the application's state.

Finally, **Dynamic Data Structure Watermarks** can be extracted by checking the values of particular program's variables with a particular input sequence, simply by using a debugger tool [4]. The watermark is stored in the execution state of the program. Embeds the watermark in the state of a program as the program is executed with a particular input sequence. e.g. global, heap, and stack data. Far more stealthy than easter egg watermark since no output is produced.

Classification by Extracting Technique

Classification of software watermark by the extracting technique falls into two classes:static or dynamic. A static software watermark is one inserted in the data area or the text of codes. The extraction of such watermarks needs not run the software.

Generally, there are two types of static watermarks [3]: 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 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.

Classification by Purpose

Software watermarks can be classified by their functional goals. Each watermark has a single goal in this taxonomy.

1. *Prevention marks*: Watermarks to prevent unauthorized uses of software.
2. *Assertion marks:* Watermarks to make a public claim to ownership of software.
3. *Permission marks:* Watermarks to allow a (limited) change or copy operation to the software.
4. *Affirmation marks:* Watermarks to ensure an end-user of the software's authenticity.

Protection of Software Watermark

The main types of protection techniques for software watermark are obfuscation and tamperproofing [5]. In obfuscation, there occurs a semantics-preserving translation of a program into another program which is hard for an adversary to understand and so it is hard for them to attack it. Obfuscation of a program also makes it hard for an adversary to locate the watermark.

Tamperproofing is, likewise, semantics preserving translation of a program into another program. It differs from obfuscation in that it is hard for an adversary to modify the new program without changing its behaviors. Thus, even if adversaries locate the watermark inserted into a tamperproved program, it may be hard to remove it without affecting the program's usability.

Watermark Evaluation Properties

Credibility: The watermark should be readily detectable for proof of authorship while minimizing the probability of coincidence.

Data-rate: Maximize the length of message that can be embedded.

Perceptual Invisibility (Stealth): A watermark should exhibit the same properties as the code around it so as to make detection difficult.

Part Protection: A good watermark should be distributed throughout the software in order to protect all parts of it.

Resilience: A watermark should withstand a variety of attacks

SandMark

A research tool for studying software protection techniques for Java bytecode[1]. Software watermarking, code obfuscation, and tamper-proofing. Includes a variety of tools to study the strength of a watermarking algorithm.

Conclusion:

Software watermarking is the process of embedding a large number into a program such that (a) the number can be reliably retrieved after the program has been subjected to sementics-preserving transformations, (b) the embedding is imperceptible to an adversary, and (c) the embedding does not degrade the performance of the program.

References

1.Sandmark:A Tool for Software Protection Research Collberg, Myles, Huntwork, IEEE Security and Privacy,July-August 2003 (Vol. 1, No. 4).

2. J. Palsberg and S. Krishnaswami, Kwon, M., Ma, D., Shao, Q. & Zhang, Y. (2001), Experience with software watermarking, in Proc. 16th Ann. Comp. Security Applications Conf. (AC-SAC'00)', IEEE Computer Society, pp. 308-316.

3. C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," in Tech. Report, No.148, Dept. of Computer Sciences, Univ. of Auckland, 1997.

4.William Feng Zhu, " Concepts and Techiques in Software Watermarking and Obfuscation" , Aug 2007

5.Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 8, AUGUST 2002