Relevance of Existing Taxonomies to Current Software Watermarking Research

Mikhail Gouline
Department of Computer Science
University of Auckland
mgou027@aucklanduni.ac.nz

Abstract

Software watermarking has become a popular security research topic in the recent years. However, despite the growing number of researchers focusing on the applications of watermarking technology, there are significant variations in the terminology and models being used. The resulting vagueness in understanding what software watermarking is, can be fixed by defining a theoretical classification for such practices. Although they are rarely found in literature, this paper summarizes and interconnects the existing taxonomies and relates them to the current research topics, in order to evaluate the relevance of these taxonomies to practical applications.

1 Introduction

A software watermark, according to [3], is defined as small amount of information embedded in the bytecode of a piece of software for the purposes of identification. However, there is much confusion among the published literature as to what other properties are required from a watermark. Some of this confusion is due to unclear definitions, for example, the terminology used for different purpose watermarks. In other words, a question whether a generic term "watermark" should be used for any types of embedded identification information, or should there be distinct terms for different agendas behind watermarking. Other confusion arises from arguments between published papers about the properties that a watermark must and must not possess. For instance, papers such as [2] constantly refer to watermarks as having to be robust, however other applications, such as currency watermarking, call for watermarks to be fragile for validation purposes. Similarly, an argument over the visibility of watermarks exists.

We use a combined software watermarking taxonomy described in [3] and [5],

since the two are related and in fact share two authors, to identify these points of confusion and address them. This paper will start by summarizing the said taxonomy and defining terminology used throughout the rest of the paper in sections 2 and 3 and then testing it on examples of literature on watermarking to identify whether that theory covers the practical instances in the research community.
The selection process for the test papers is based around giving a broad overview of literature in terms of age and topics covered. Hence, the article [2] was selected as being the oldest one, published in 1998 and providing the widest look not only on the field of watermarking, but also on classical steganography. At the other end, articles [4] and [1] offer specific new techniques and approaches to the field marking, in their own respective ways, the departure from the standards adapted in the years when the taxonomy was developed. Such mixture of papers brings diversity to the analysis and hence makes the conclusions drawn from it more credible.

2 Background

This section defines the taxonomy that will be used in the paper, by presenting the classification of watermarks and discussing the confusion points described in [5], making clarifications on them.

2.1 Types of watermarks

In order to have separate terminology for different purpose watermarks, [5] defines four types discussed in the following subsections. This terminology will be applied in the analysis of the test papers.

2.1.1 Authorship mark

One of the more popular types of a watermark discussed in literature [3] [4] serves the purpose of identifying who the author of the software is. In other words, being able to run the software through a recognizer and hence prove that the author of the software is who is claiming to be that.

2.1.2 Fingerprinting mark

Although different purposes can be classified as being in this category, fingerprinting essentially refers to embedding copy-specific information that identifies the person who purchased the software, such as a serial number among others. This can be used to trace a copy of the software involved in a violation back to the licensee.

2.1.3 Validation mark

In addition to the interests expressed by the authors and the distributors, watermarks are also used to satisfy the customers' interests. Validation is needed to verify the integrity of the software received by the end-user. In other words, to assure the customer that the copy of the software that they have purchased had not been modified by a third party in any way and that their privacy has not been compromised.

2.1.4 Licensing mark

Finally, a watermark can carry licensing information attached to the software. This refers to the information controlling the way the software can be used by the licensee, which can prove what terms of use the author had attached to the software before the customer purchased a copy.

2.2 Properties of watermarks

In addition to classifying the watermarks by their purpose, confusion arises over the properties possessed by the watermarks. These are discussed in the following subsections.

2.2.1 Visibility

As [5] explains in references, certain papers argue that watermarks should be visible, meaning that anyone, including a potential attacker, can see the watermark but only the owner can modify or remove it. The motivation behind this is making the technology open in order for the parties to be able to ensure of its effectiveness. Others, on the other hand, are leaning towards having invisible watermarks. This point of view suggest that not being able to even detect the watermark decreases the attacker's chances of being able to damage it.

2.2.2 Robustness

Unlike visibility, robustness of a watermark is not a point of opinion but rather a property dictated by the purpose behind the watermark. If the goal of the system is preserving the embedded information in the software regardless of the modifications, then a robust watermark is desired that will not be susceptible to attacks. However, if the purpose is to verify the integrity of a distributed copy, the watermark should be fragile, so that if any modifications have been made, it becomes invalid indicating that the software is not in its original state.

2.2.3 Efficiency

Watermarking systems introduce an efficiency trade-off in two categories. First, between watermarks being easy to recognize and hard to damage – because the more sophisticated the technique is, the more computationally expensive it becomes to extract the information therefore making it even more difficult for the attackers. And second, between watermarking not interfering with the performance of the software and, again, embedded information being harder to modify. More sophisticated techniques increase computational costs to the software when used, which decreases the utility of such software to the end-user.
However, looking at efficiency from a different perspective, the second trade-off may not be valid for invisible watermarks, since the very fact that the performance of the software is noticeably affected by the embedding, can give away its existence to the attacker, which is most undesirable for such systems.

3 Attacks

Potentially, the purpose of watermarking is identifying unauthorized practices and therefore setting limitations to what users are allowed to do, which means that attempts will be made to bypass that by performing attacks on the watermark. Possible attack models are formalized in [3] and [5] as presented in the following subsections, grouped by the robustness property due to the differences in models caused by this distinction.

3.1 Robust watermarks

The goal in attacking a robust watermark is rendering it unverifiable. The taxonomy identifies four types of such attacks, three of which can be applied to both, watermarking and fingerprinting systems while the fourth one – collusive attack model – is exclusive to the fingerprinting, as it it relies on the embedded information being different from one copy to another.

3.1.1 Rewrite attack

The hardest but most effective approach for removing a watermark is completely rewriting the program and simply leaving the watermark out. In this case, any attempts to verify the origin of the software based on the embedded watermark will yield no results, since the attacker did not add the watermark while rewriting the program and hence it does not exist in that copy anymore. Although it is the most effective attack model, it is also the most difficult to practically implement

since it involves a complete rewrite of the code, which may not be feasible or even possible in certain settings.

3.1.2 Additive attack

Apart from completely removing the watermark, an attacker may choose to modify the software in order to confuse the possible attempts to identify the origin of the software. Such an attack involves the adversary embedding their own watermark in the program code to falsely claim ownership. In this case, not only can the attacker counter-claim for plagiarism if the situation ends up in court, since their watermark is now provably present in the software, but the original mark could have been damaged to the point where it became unverifiable as a result of inserting a new one and the interference that caused.

3.1.3 Distortive attack

Instead of attempting to completely remove the watermark, which can be too difficult, as mentioned before, the attacker can try to modify it to a state where the author's verification system will not be able to recognize it. Distortive attacks can be performed using semantic-preserving transformations to the program code, such as code obfuscation. Modifications like these are designed to damage the watermark but retain the full functionality of the original software by changing names, ordering and formatting of the bytecode leaving the overall structure intact, expecting that the watermarking technique used relies on those factors.

3.1.4 Collusive attack

Finally, another type of attack can be used on fingerprinted software, where the attacker obtains multiple copies of the software, then by comparing them, discovers where the fingerprint is located and removes it. By definition, fingerprinted software has a property that every copy is essentially different, since the embedded fingerprint is different. This property can be used by the attacker to identify the parts that do not match, concluding that they contain the embedding. To prevent collusive attacks, the author can apply different types of modifications to each distributed copy of the software to render the matching technique useless.

3.2 Fragile watermarks

Unlike a robust watermark, fragile watermarks should become invalid if any modification has been made to the software, so the goal of the attacker is to keep the

watermark valid while performing these modifications. The three possible scenarios of how that can be achieved, as defined in the taxonomy, are presented below.

3.2.1 Bypassing watermarks

One possibility is to leave the watermark intact while modifying the software. In other words, identify the actions likely to break the watermark and only apply transformations that perform them, so that the recognizer cannot detect the forgery.

3.2.2 Reinserting watermarks

If the watermark has been damaged as a result of performing some modifications, and the attacker has a way of acquiring the author's private key and the digest algorithm, they can create a new watermark and embed it back into the program. That way, it would appear to the recognizer that the software has not been changed, since the watermark is as valid as the original one, because they were generated in the same way.

3.2.3 Crafting recognizer

Finally, if the watermark has been damaged and it is obvious to the attacker that the recognizer will be able to identify the violation, they can create a crafted recognizer that verifies the damaged watermark as being genuine. In this case, it is largely dependent on the ability to pass the crafted recognizer for the real one in the setting involved.

4 Evaluation

In this section, the taxonomy defined above is tested on various watermarking papers. In particular, the aim is to establish whether the taxonomy covers each case and how well it does that, based on how straightforward and definite the process of applying it is.

4.1 On The Limits of Steganography

Although the article [2] focuses on the broader area of steganography, which does include software watermarking but does not set it as the main emphasis point, it can be viewed as the early attempt to set up a standardized framework for understanding steganography. The reason behind testing the taxonomy on this paper is identifying whether it lines up with the similar work done prior to its creation that has a wider scope inclusive of our topic.

4.1.1 Overview

The authors essentially concentrate on the Prisoners' Problem, where two prisoners pass messages to each other, by means of the warden forwarding them, and the goal is to get the secret message across to the addressee without the warden noticing that the secret message – the stego object – is even there. This is defined as the *passive* warden model in the article, since the warden makes no attempts to modify the message and instead only tries to identify the existence of that secret message. However, in the software watermarking area *active* "wardens" are more common, such as pirates, who modify the message in hope of damaging or removing the hidden message. This model is then related to the way watermarks of general and unique types are passed as hidden messages in the software copies.

4.1.2 Analysis

Throughout the article, the authors make it clear that watermarks are defined to be invisible. The Prisoners' Problem described above suggests that regardless of the type of wardens present – passive or active – they do not know that the secret messages are being passed and how they are being passed, in fact this is the very point of steganography, according to the authors – keeping the process of passing secret messages a secret from others not involved in the conversation. From that argument, it is clear that the article talks about robust watermarks, where the goal is to withstand an active warden and preserve the information embedded in the software.
Interestingly, the paper makes the same distinction between "watermarks" and "fingerprints", defined in the taxonomy as authorship and fingerprinting marks respectively, where the former refers to a general mark placed in all copies of the software in order to identify the author and the latter refers to a copy-specific mark that uniquely identifies each copy.
A notion presented in the article that does not completely line up with the taxonomy is the situation where the attacker knows where and how the watermark is embedded and uses that knowledge to remove it. Since the motivation is still rendering the mark unverifiable, it could be categorized as the distortive attack model. However, the distortive attack model suggests it is unknown where the mark is and the attacker uses generic techniques for damaging it, that have been shown to work on similar systems, but are not guaranteed to work on the current one. Alternatively, it is similar to the collusive model, where comparing multiple fingerprinted copies gives away the location of the watermark, which is then used to remove it. The difference is that the article illustrates a situation where it does not have to be neither a fingerprinted piece of software, nor does the location of the mark have to

be discovered by comparing multiple copies – in fact, it can be any watermarked software, and the inside information could have even been obtained from any non-technological sources, such as direct human-to-human interaction (for example, phishing). So in this respect, the taxonomy does not fully cover the point raised in the article.
Efficiency, although mostly being left out of the analysis, does appear in multiple sections of the article, confirming the notion defined in the taxonomy that the embedded information should not impede the performance of the original product, which in this case is software.

4.1.3 Overall fitness

Although most of the points raised by the authors of the article are covered by the taxonomy, certain blanks can be identified on both sides. First of all, the article does not discuss other dimensions of points defined in the taxonomy, such as visibility. On the other hand, the taxonomy does not cover the situation where inside information on the location of the watermark is used to remove it, as it falls between the distortive and collusive attack models, not fully satisfying requirements for either of them.

4.2 Dynamic K-gram based Software Birthmark

The article [4] introduces a novel approach to watermarking bytecode by considering the dynamic states that the program goes through on a given input. The technique proposed for that is k-gram slicing of the program op-code.

4.2.1 Overview

Building on an earlier development in [6] presenting a technique to use k-grams of static program code for watermarking, the article offers a way to consider the states of the program when run on a given input in addition to the static code. According to experimental evidence presented, this increases the effectiveness of the watermark and lessens the chances of an attacker damaging it by using obfuscation techniques. To achieve that, the article provides a way to slice the op-code generated from the program byte code into k-grams and as a result generate the watermark, relative to the program and the input fed into it.

4.2.2 Analysis

The authors clearly state the intention they had for their watermarking technique – being able to sustain obfuscation attacks without damaging the watermark. This

description is classified as being a robust watermark in the taxonomy. Furthermore, obfuscation attacks are regarded as the distortive attack model in the class of robust watermarks. Other attack models are purposefully left out of the paper to focus on the most prevalent one for the given class of watermarks.
The purpose of the information embedded in the watermark is not clearly stated, although since no information is being embedded inside the code, and the code is being analyzed to create the watermark instead, the suggested purpose, prevalent throughout the examples used in the experimentation, is to confirm authorship. Hence, under the taxonomy, this qualifies as an authorship mark technique.
Finally, the visibility is not discussed explicitly but since, again, no information is being added to the code and the watermark generation process is detailed in the article, the technique qualifies for the visible mark. Unlike visibility, however, efficiency is the point that is not obvious from reading the paper since no computational complexity analysis has been included and considering the nature of the technique being focused on the verification process, this can be important in practical applications of the technology.

4.2.3 Overall fitness

Overall, the taxonomy covers all the points discussed in the article either explicitly or by conclusions drawn from the absence of alternatives, such as the attack model. However, the authors fail to discuss some important points, such as efficiency, that are relevant to the nature of the presented technology. Furthermore, they do not use any formal definitions when defining the purpose and type of the watermark that they developed, leaving the classification and standardization of terminology to the reader. It is also noteworthy that no conflicts of definitions and classifications have been identified between those given in the taxonomy and those mentioned in the article.

4.3 Protecting Mobile Agent's Computation Results with Reversible Watermarking and Digital Signature

The article [1] targets the security issues involved in the Mobile Agent (MA) technology. Namely, protecting the results of the agent's computation upon its return to the home system. This is done with a reversible watermark and digital signature.

4.3.1 Overview

The authors proposed a novel way of ensuring that the MA has not been intercepted and modified while collecting data and computing results on the network, such

that the verification can be performed upon its return to the home platform. The proposed method involves first watermarking the collected results, then computing the hash of the watermarked data and finally signing it with private key of the remote host. In the end, we obtain a digital signature, in this case generated by RSA. Upon its arrival at the home system, the hashes are re-computed and verified against the transferred values and digital signatures are checked for authenticity using RSA.

4.3.2 Analysis

Applying the taxonomy to the example illustrated in the article, it is clear that this is a fragile validation mark. The purpose of it is to become invalid in case the MA has been modified before its return to the home platform, hence the usage of the hash to identify inconsistencies. The authors do not mention the visibility status that their system requires but the fact that the process description along with the algorithm have been published freely in a scientific article rather than in a proprietary closed-source medium, suggests that the watermark can be classified as visible, unless the information about which of the open techniques is used on a particular MA system is kept secret. But this is not stated.
Although the article does not discuss possible attack situations, all three attack models on this class of watermarks, defined in the taxonomy, apply to this example namely, bypassing the protection, reinserting the watermark and crafting a recognizer. In particular, the first two are very much relevant while the third one is less likely, since the verification process takes place on the trusted home platform and placing a third-party recognizer there would be difficult, but nevertheless possible, especially if a remote distributed verification system is used.

4.3.3 Overall fitness

The article demonstrated a complete compatibility with the defined taxonomy, since all discussed points were covered. The initial classification of the type of watermarking referred to and one of the properties matched the descriptions offered in the taxonomy. In addition, it introduced points, such as attack models, that the authors did not explicitly state their stance on, which leaves room for more research to be done in the area of MA security.

5 Conclusion

The testing performed on the watermarking articles showed that the defined taxonomy covered the points discussed by the authors to a satisfactory level, i.e. with no

serious conflicts. What is prevalent throughout the analysis is that, as mentioned in the beginning of this paper, authors are not consistent with the terminology they use, generalizing or even misinterpreting the definitions, disregarding the properties that their approaches may or may not have and the reasons behind that. Finally, one of the articles completely ignored the much needed analysis of possible attack models that the adversaries can use to compromise their systems.
Some blanks in the taxonomy were identified as well, by the earlier paper focusing on the broader field of steganography, where an attack model could not be accurately classified, instead falling in between the two defined models not entirely satisfying either of them.
A conclusion to be drawn from the analysis is that taxonomies help formalize each new approach and line it up with the previous work done on the subject. The authors not following this model end up publishing material that has to be first carefully classified by the reader before relating to the background developments and hence evaluating the utility of the new offering.

References

[1] X. Niu A. Khan and Z. Yong. Protecting mobile agent's computation results with reversible watermarking and digital signature. *Third International Conference on International Information Hiding and Multimedia Signal Processing*, 2007.

[2] R. Anderson and F. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications*, 1998.

[3] C. Collberg and J. Nagra. *Surreptitious Software. Obfuscation, Watermarking, Tamperproofing for Software Protection*. Addison-Wesley, 2009.

[4] Y. Bai et al. Dynamic k-gram based software birthmark. *19th Australian Conference on Software Engineering*, 2008.

[5] C. Thomborson J. Nagra and C. Collberg. A functional taxonomy for software watermarking. *Twenty-Fifth Australasian Computer Science Conference*, 2002.

[6] G. Myles and C. Collberg. K-gram based software birthmarks. *ACM Symposium on Applied Computing*, 2005.