A review of “Robust Object Watermarking: Application to Code”

Tapas R. Sahoo

The proliferation of digital document in recent times is creating a need for the protecting copyright ownership. Different directions are being explored to dissuade an user from copying and distributing his own illegitimate copies. One drawback of conventional cyrptography is that once the secret key is broken there is no way to prevent the copyright violater from reproduction and redistribution. A digital watermark is intended to trace the ownership the document by embedding information in the data. This paper focusses on a robust technique to embed watermarks in exceutable code.

The text documents, source codes and machine codes create difficulties in watermarking due to there inherent structure which allows a low level of modification when compared to the audio, video or images. One primary reason for such tighter constraints in modification in the former case is that both the structure as well as the functionality needs to be preserved. The difficulty to modify data can be taken advantage of to create a robust way to embed information. But it should also be noted that for an application to provide freedom for stealthly modifying it also gives the attacker equal or infact more oppurtunity to fiddle with and defeat the entire watermarking scheme.

This paper introduces a new approach to watermarking. Instead of applying the watermarking scheme to the raw code directly, a new vector representation of the code is created. This approach strongly limits the range of modification that can be applied to the marked object. The conventional approach of taking advantage of the structure to create efficient watermark limits the effectiveness only to larger target programs where one can utilise the structure to its fullest potential. The basic idea revolving around this scheme is that instead of considering the overall structure of the code and its control flow, the code is viewed as a statistical object. The frequencies of group of instructions in the entire code are taken into consideration in creating a new vector representation of the data and then using spread spectrum techniques over it, a robust watermarking scheme can be obtained. Spreading the watermark throughout the target code ensures a large measure of security against intentional or unintentional attacks.

The vector extraction paradigm A piece of data can be represented by many different vectors in different spaces. So contrary to the idea of treating the entire data as a single vector, a particular vector is first extracted from the data and the watermarking scheme is then applied to it. This extraction step happens to be the most crucial to the stealth and robustness of the overall scheme. The extraction step then aids in outlining the degree to which the different attacks can be resisted. On the otherhand, recovery of the watermark or simply checking its presence or absense depends on the spread spectrum technique implemented over it. So each one of them, the intial representation as well as the embedding step through spread spectrum technique play an equal role in the effectiveness of the watermark. Additionally, in the extraction step, a distance d is defined in the vector space. For the watermark to be robust, the different *possible* attacks should only allowed to modify the initial vector within a particular limit. When it exceeds this threshold, the code is rendered useless. In other words, we can define this limit as boundary within which the watermaker can successfully retrieve his embedded watermark.

Approach
The foremost thing that comes to attention before any watermarking scheme is the imperceptibility or stealth of the watermark. This holds different intepretation for different kinds of data to be watermarked. For code, the imperceptibilty should span across the not just the structure of the code but also its functionality ie. the watermarked code should behave *similarly* to the original code. One major motivation behind this vector extraction step is that the nature of code transformations that an attacker normally adopts to defeat the marking are mainly localized modifications and functional reordering. This is actually a serious limitation for the attacker since applying random global modification has a high chance of rendering the code useless unless implemented with caution. The vector extraction step is completely immune to function reordering while survives the local modifications to an appreciable extent.
Vector Extraction: Define n as a security parameter. Define a set S of n ordered groups of machine language instructions. For each group i in S, compute the frequency c_i of the group in the code, and form the vector $c = (c_1, \ldots, c_n)$. This c is the extracted vector. Then the Euclidean norm calculates to the distance d on the vector space.

Algorithm
Initialisation: Set a detection threshold $\sigma (0 < \sigma < 1)$.
Watermark Insertion:

- Apply the vector extraction step to obtain a vector c (of length n).
- Choose a n coordinates vector to $w = (w_1, \ldots, w_n)$ whose coefficients are randomly distributed following a normal law with standard deviation α.
- Modify the code in such a way that the new extracted vector $\overline{c}$ is $c + w$.

Watermark Testing:

- Apply the extraction step to obtain a vector d.

- Compute a simliarity measure Q between $d - \overline{c}$ and w.
- If Q is higher than σ then the algorithm outputs $marked$, else it outputs $unmarked$.

The actual implementation of modifying the frequency of the code is done in an iterative manner. A random modification is done to the code while preserving its correctness and the new frequency vector computed for the new frequencies of the same block of instructions which was used to calculate c initially. If the new vector is closer to $\overline{c}$, the process is continued, else the modification is refused. Ofcourse such modifications to the code does not have arbitrary degrees of freedom. It shall depend on the specificity of the code language. As a simple example, its not acceptable to swap a pair of instructions if a variable $defined$ in the first instruction has its use in the later one.

Analysis of Attacks As mentioned earlier, the scope of the attacks can be summarised to quite an appreciable extent because of the inflexiblity of the machine codes. Let us now enumerate some of the possible attacks and to what extent the above scheme can withstand it.
The *code decompiling* attack ie. decompiling and recompiling the code, possess the most serious threat since this can completely distort the frequencies of the vector. Since decompiling a small section of the code will not affect the vector frequency, the attacker needs to decompile the entire code which becomes extremely difficult and tedious for large programs. Similarly, the *code addition* attack could distort the vector frequency appreciably, but the added code size needs to be very large to produce the effect. This is solely because the attacker has no idea of the precise deviation α that was used in creating the watermark. So random code frequency modification may decrease the code frequency in one group and increase it in another, thus nullifying each others effect and the vector stays intact. *Code compression* does not possess serious attack since compressing the code renders it nonexecutable. Compressed code can be easily detected and uncompressed to retrieve the watermark.

Implementation: The implementation was done using x86 assembly language as the target code for watermarking. The basic approach was outlining the groups of instructions in the code whose frequencies are to be modified ie. the watermarking is to be done across these group of instructions. The next step was to built a *code book* of equivalent group of instructions. This is basically a mapping from the previous instruction to the restructured instrution. When the modifications (local or global) are done to the code, the functionality of the code should be predictable (ie. it should not render the code non-functional) or it should be done in a conservative manner, verifying the code at each small steps of frequency modification. Different types of modifications can be implemented, for example reordering pair of instructions withing a group, or even swapping two group of instructions as long as any dependencies are not violated. Other modifications include substituting a groups of instructions by an functionally equivalent set of instructions.

Conclusion The vector extraction paradigm and its application to machine code produces a robust watermarking technique for machine codes. It mainly takes advantage of the distribution of the watermark over the code to maintain its stealth. It is also shown that data with lower level of tolerance for modification are easier to withstand tampering.

Bibliography

1. Julien P. Stern, Gael Hachez, Francois Koeune, and Jean-Jacques Quisquater. Robust Object Watermarking: Application to Code. In A. Pfitzmann, editor, Information Hiding '99, volume 1768 of Lectures Notes in Computer Science (LNCS), pages 368–378, Dresden, Germany, 2000. Springer-Verlag. http://citeseer.nj.nec.com/stern00robust.html

2. I. Cox, J. Kilian, T. Leighton and T. Shamoon, "A secure, robust watermark for multimedia", in Proc. Workshop on Information Hiding, Univ. of Cambridge, U.K., May 30 - June 1, 1996, pp. 175-190 http://citeseer.nj.nec.com/article/cox96secure.html

3. Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In Aucsmith, 3 pages 218–238. ISBN 3-540-65386-4. http://citeseer.nj.nec.com/petitcolas98attacks.html