Static Software Watermarking Using Graph-Reckoning: Piracy Control for Information Systems

Sohail Sarwar[1], Muhammad Safyan[2], Zia Ul Qayyum[1], Muddassir Iqbal[3], Yasir[4], and Farrukh Latif[5]
[1] University of Gujrat, Pakistan
[2] GC University Lahore, Pakistan
[3] London South Bank University, England
[4] Iqra University
[5] Bahria University
Email: sohail.sarwar@seecs.edu.pk, m.safyan@gcul.edu.pk, ziaqayyum@uog.edu.pk, miqbal@lsbu.uk, yasir@iqraisb.edu.pk, flatif@gmail.com

***Abstract*—Information Systems, as intellectual property, ensures potential earning of businesses that have been affected badly by software piracy. The impact of software piracy can be languished through different techniques such as obfuscation, birthmarks and watermarks etc. used to counter pirated software. A watermarking approach has been presented using data mining techniques. The watermark generation exploits the program constructs (as properties and their relations) in flow graphs. The watermarks generated are embedded in methods for keeping track of actual program ownership. The major advantage of proposed technique is its improved degree for piracy detection (at method level granularity). Also, the technique presented is more resilient to major attacks such as additive, distortive and removal attacks when compared with prevalent watermarking techniques.**

***Index Terms*—piracy, watermarking, graph mining, resilience**

I. Introduction

In this era of technical advancements, all aspects of technology are observed to grow at a greater pace. Under this revolutionary progress in software industry, threat of software piracy is causing losses in billions of dollars every year [1], [2]. In order to counter such losses, software owners are striving hard for developing privacy control techniques such as software watermarking, software birthmarking, obfuscation, encryption and tamper-proofing [3], [4]. However, focus of this work is to devise a resilient software watermarking technique. Software watermarking is the phenomenon of embedding secret information called a “watermark” in the target software for discouraging ownership theft of intellectual property [3]. The watermark is extracted from the suspected/pirated software to prove the software ownership. Fingerprinting is specialized watermarking technique, in which information of software-user is embedded into the software [5]. It helps to identify those who propagate software piracy, in addition to proving software ownership.

Software watermarking loomed here exploits data mining techniques (especially graph mining) [6]. In order to calculate the watermarks of each method and class, to prove the ownership of that method or class, graph theory as well as network science concepts have been employed [7], [8]. Software watermarks are extracted from software properties and elements of program methods and then embedded into the software methods. Unlike using only the method names for watermark, method information is used to protect program methods through method elements based watermark. The proposed technique works by extracting the syntactic structure of program to compute property value for each element and relation among those elements. Graph theoretic properties such as clustering coefficient have been used with property values to compute watermarks based on method-elements. The data then is transformed into graphs to identify if a method/class has been copied through graph comparison. Coupled class/method relation in the program is used for the generation of program watermark. The watermarks of incertitude code snippets have been extracted in order to formulate the degree of resemblance among methods so that similarity/dissimilarity may be asserted for method/class.

The proposed technique is measured against major watermarking attacks. The experiment results show that proposed technique is resilient against additive, subtractive, distortive and collusive attacks. Further, evaluation of proposed technique is performed with parameters of overhead, data-rate, and resilience. Organization of paper is given below:

Section 2 provides a brief review of watermarking techniques followed by proposed approach in section 3. Section 4 furnishes conclusion and further potential directions.

Manuscript received May 6, 2019, revised September 26, 2019.

II. State of the Art Literature

The preliminary concepts in watermarking techniques as well as some effective watermarking techniques have been discussed. Further, advantages of watermarking over obfuscation are also discussed.

A. Software Watermarking

Software watermark is secret information that is embedded into the distributed software (process adding/extracting watermark is called as software watermarking) [3]. It is a defensive and preventive measure against software piracy [9]. The watermark in the software is secret i.e. it should not be revealed by anonymous person but can be extracted by the owner only. There are two types of watermarking i.e. Static and Dynamic watermarks.

B. Static Software Watermarking

Static watermark strategies embed the watermark *W = {w1, w2, w3,…. wn}* in the information/or code [10] of program *P = {P1, P2, P3 …. Pn}*. These types of watermarks are embedded secretly in the software, mostly as dead code. The watermark is extracted statically when it is required from the code. The programs are not executed to extract the watermark. The static watermarking is further classified into data watermarking and code watermarking [3].

C. Dynamic Software Watermarking

The watermark *W = {w1, w2, w3,…. wn}* is embedded in the state of the program *P = {P1, P2, P3 …. Pn}*. These types of watermarks require execution of program to recognize the watermark [11]. Special inputs I = {I1, I2, I3, …. In} are required so that the state of the watermark is executed. When special set of inputs is provided the program may produce special output, this type of dynamic watermarks is called Easter Egg watermarks [3]. Sometimes special output is not produced instead sequence of operation or data structure values, based on special input is treated as dynamic watermarks.

D. Flow of Software Code

Flow-graph *G = {G1, G2, G3, …. Gn}* of the software program *P = {P1, P2, P3 …. Pn}* have frequently been used as preventive measure against piracy control. The flow-graphs have been used in both static as well as dynamic software watermarking. In static watermarking methodology the flow-graph is used to embed watermark [12], while in dynamic watermarking the control flow has been used to compute the watermark [13]. In other than watermarking the graph based techniques [14] have also been frequently used for software birthmarking as well.

III. Proposed Approach

Software programs are composed of various program methods and each method is composed of different elements. The proposed technique based on static watermarking uses Graph Reckoning; that identifies intrinsic properties from program method (such as variables, iterations, decisions and data) and relation among the elements. The relation among the method elements is hard to change and affects the output in addition to effecting program performance. The relation among the elements with their properties is transformed into graph, where node is an element and edges represents the relations.

Each of the nodes along with its properties alone is inadequate to proffer substantiation in method piracy control. Properties of all the methods in form of graph are used to compute the watermark *W = {w1, w2, w3,…. wn}* for program *P = {P1, P2, P3 …. Pn}*. These set of watermarks are hint software origin. The elements and their relations are identified in the form of nodes from the method code. Two types of relations have been used in the generation and hence computation of watermark for method. Essential-relation represented with edge may be in-relation to one node and out-relation with other node based on the properties.

For example for a graph G with set of nodes N and edges E is defined below.

$$G = \{N, E\};\ N = \{n1, n2, n3, \ldots nk\};$$

$$E = \{e1, e2, e3, \ldots em\}$$

For any node i and j, there must be one Essential-relation between them. ni → nj for i ≠ j

The looping constructs (or element) and relation among method elements is given in Table I.

TABLE I. Method Relations for Coupling the Elements

Element Name	Properties				
	Global variable	Local variable	Loop	Condition	Data
Global variable	assign	assign	used	used	assign
Local variable	assign	assign	used	used	assign
Loop	used	used	sibling	used	used
Condition	exit	used	used	Sibling	used
Data	assign	assign	used	used	assign

The computation and generation of watermark for 'Program *P*' has been illustrated in Fig. 1. Code purification and designation reads elements along with their properties that are assembled by collecting the information and generate individual method graph watermark.

If we have program

$$P = \{P1, P2, P3 \ldots Pn\}$$

Graph G = {N, E} is generated and

$$W = \{w1, w2, w3, \ldots wn\}$$

is computed from the G. The watermark W is embedded in the program P.

Usually watermarking detects designated software as a whole pirated but proposed technique for watermarking detects if some portion of the program us pirated. This is courtesy to generation and extraction of watermark (based on method properties and relations).

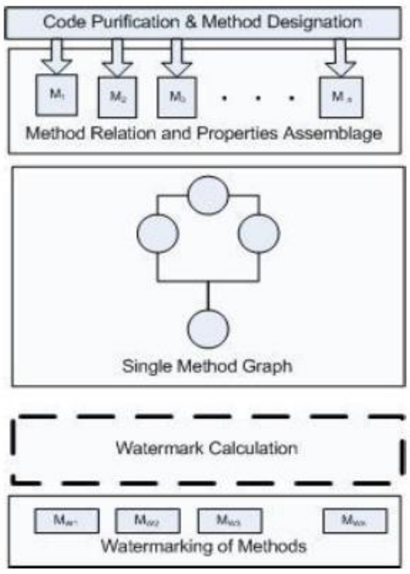

Figure 1. Watermark computation and embedding

IV. Results and Evaluation

The proposed watermarking technique was evaluated over three code snippets from sourceforge.net [15]. The proposed technique working on method code based watermarks performed better since it takes into account relation and properties of each of the node. The similarity percentage between different and same methods was calculated.

Accuracy computed for similar as well as non-similar objects in the methods is 0.80 to 0.93 as tabulated in Table II below.

TABLE II. Confusion Matrix Similarity Calculation for Modified Programs

Classes	Similar Objects'	Non-Similar Objects'	Total	% Accuracy
Detection of Similar Objects	959	91	1050	0.93
Detection of Non-Similar Objects	235	827	1297	0.88
Total	1521	362	1883	0.80

In order to measure the performance of a watermark, the resilience property is used that identifies the modified and transformed programs as similar. It becomes important when programs are modified to attack the existing watermarks. Zero percent 0% means code is dissimilar, while 100% indicates codes are perfect copy of each other.

Table III shows results for similarity classification with focus on credibility and reliability for selected programs. The similarity classification has been computed among different programs and same (self) program. The results show that self-programs has been classified as perfect copy where as other programs show 0% classification results. This detection suggests that watermarking technique is fully resilient for detection of program copy.

TABLE III. Similarity Classification for Credibility and Reliability

Java Software Packages	ATM	Library System	Point of Sale	Hospital System	K-means Algo.
ATM	100	0	0	0	0
Library System	0	100	0	0	0
Point of Sale	0	0	100	0	0
Hospital System	0	0	0	100	0
K-means Algo.	0	0	0	0	100

V. Conclusion and Future Work

A graph-based static software watermarking technique has been proposed in this paper. It operates at method level program constructs for identifying program element properties and relations between those elements. These element and relations are modeled in the graphs. The resultant graphs are embedded in the code for detecting if codes are original or modified. The detection accuracy through watermark has shown an acceptable level with resilience of watermark. We look forward to devise a dynamic software watermarking technique with larger repository of code snippets.

Acknowledgment

We hereby acknowledge the funds support under the project PrivSoft sponsored by Higher Education Commission under the grant number: 112116-Eg043. Also, we acknowledge the contributions of Prof. Dr Muddesar from LSBU England in coordinating the experiments and write-ups among involved groups.

References

[1] B. S. Alliance (BSA), "9th annual global piracy study," 2013.

[2] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation-tools for software protection," *Softw. Eng. IEEE Trans.*, vol. 28, no. 8, pp. 735–746, 2002.

[3] B. Fu, G. Richard III, and Y. Chen, "Some new approaches for preventing software tampering," in *Proc. the 44th Annual Southeast Regional Conference*, 2006, pp. 655–660.

[4] P. E. Solutions, "Dotfuscator, technical white paper, version 2," 2004.

[5] S. K. Udupa, S. K. Debray, and M. Madou, "Obfuscation: Reverse engineering obfuscated code," in *Proc. 12th Working Conference on Reverse Engineering*, 2005, p. 10.

[6] J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang, "Experience with software watermarking," in *Proc. 16th Conference Computer Security Applications*, 2000, pp. 308-316.

[7] J. Nagra and C. Collberg, "Surreptitious software: Obfuscation, watermarking, and tamper proofing for software protection," *Pearson Education*, 2009.

[8] C. S. Collberg, C. Thomborson, and G. M. Townsend, "Dynamic graph-based software fingerprinting," *ACM Transaction on. Programing Language System*, vol. 29, no. 6, pp. 35-51, 2007.

[9] Y. Bai, X. Sun, G. Sun, X. Deng, and X. Zhou, "Dynamic k-gram based software birthmark," in *Proc. 19th Australian Conference on Software Engineering*, 2008, pp. 644–649.

[10] D. Schuler, V. Dallmeier, and C. Lindig, "A dynamic birthmark for java," in *Proc. the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, 2007, pp. 274–283.

[11] J. Chen, K. Li, W. Wen, W. Chen, and C. Yan, "Software watermarking for Java program based on method name encoding," in *Proc. International Conference on Advanced Intelligent Systems and Informatics*, 2017, pp. 865–874.

[12] M. D. Preda and M. Pasqua, "Software watermarking: A Semantics-based approach," *Electronic Notes on Theoretical Comput. Sci.*, vol. 331, pp. 71–85, 2017.

[13] K. Kumar, V. Kehar, and P. Kaur, "An evaluation of dynamic Java bytecode software watermarking algorithms," *Watermark*, vol. 10, no. 7, 2016.

[14] Z. Chen, C. Jia, and D. Xu, "Hidden path: Dynamic software watermarking based on control flow obfuscation," in *Proc. IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC)*, 2017, vol. 2, pp. 443–450.

[15] Sourceforge. (Mar. 2018). [Online]. Available: Sourceforge.net

Dr. Sohail Sarwar received the PhD degree in Computer Science from University of Gujrat, Pakistan. His research interests include machine learning techniques, piracy control, vehicular technologies, semantic technologies and knowledge engineering techniques in different applications.

Dr. Muhammad Safyan is Assistant Professor in Government College University (GCU) Lahore. He received PhD degree from National University of Sciences and Technology in 2018. His area of interest is ontology alignment, e-learning and semantic activity recognition.

Prof. Zia Ul Qayyum is currently a Professor at University of Gujrat Pakistan. He received his Ph.D. degree in Computer Science from Leeds University UK in 2005. His research interests include Artificial Intelligence, Knowledge Engineering, Vehicular technologies, Data mining, Semantic web and e-learning.

Yasir is MS in Computer Science in Computer Science 2008. His research interests include semantic technologies, ontology engineering, and ontology-based data integration.

Dr. Muddassir is professor at University of Gujrat. He did PhD from Kingston University UK. His research interests include CBT, V2X technologies, Disaster Management and IoT.

Farrukh Latif is MS scholar in Bahria University. He has more than 10 years of experience in domain of software development. His area of interest is software ethics, piracy control and Stagnography.