Software watermarking: contributions that transcend the theme

Lucila Bento[2], Davidson Boccardo[3], Raphael Machado[1,4], Felipe Simões[1], Vinícius Pereira de Sá[5]

[1]Institute of Computing – Fluminense Federal University (UFF)
Niterói – RJ – Brazil

[2]Institute of Mathematics and Statistics – State University of Rio de Janeiro (UERJ)
Rio de Janeiro – RJ – Brazil

[3]Clavis Information Security
Rio de Janeiro – RJ – Brazil

[4]National Institute of Metrology, Quality and Technology (INMETRO)
Duque de Caxias – RJ – Brazil

[5]Institute of Computing – Federal University of Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brazil

lucila.bento@ime.uerj.br, davidson@clavis.com.br, raphaelmachado@ic.uff.br
felipe_simoes@id.uff.br, vigusmao@dcc.ufrj.br

Abstract. *Software watermarking is the embedding of data in the graphs that represent the execution of a computer program. Such structures are important technology tools, but also bring interesting combinatorial problems. In the present paper, we describe the advances in software watermarking during the last two decades, highlighting the important contributions of Prof. Jayme Szwarcfiter.*

Resumo. *Marcas d'água em software são informações que podem ser codificadas por meio da alteração dos grafos que caracterizam a execução de um programa de computador. Tais estruturas são uma importante ferramenta tecnológica, mas também trazem à tona problemas combinatórios interessantes. No presente artigo, descrevemos os avanços realizados nas últimas duas décadas no tema das marcas dágua em software, destacando as importantes contribuições do Prof. Jayme Szwarcfiter.*

1. A practical problem

Despite the number of laws to protect intellectual property, software copyright infringement remains a problem to this date [Asongu 2021]. It is estimated that the financial impact of unlicensed software in 2022 will exceed US$19.8 billion [Goff 2022]. Techniques to protect against software piracy therefore emerge as a powerful ally.

For a long time watermarks have been used to establish authenticity, authorship, or ownership of objects to protect software intellectual property. They are included in the software to be protected, allowing the reveal which authentic copy it originated from in a future audit process.

Cadernos do IME - Série Informática
e-ISSN: 2317-2193 (online)
DOI: 10.12957/cadinf.2022.70579

Among the existent watermark schemes, the ones based on graphs stand out owing to their simplicity and resistance to attacks [Collberg and Thomborson 1999]. This type of software watermark comprises encoding/decoding algorithms (*codecs*) to translate an identification information (the *identifier*) into a graph, and embedding/extracting algorithms to insert/recognize existing watermarks in a software. A good watermark codec based on graphs should satisfy the following properties [Collberg and Nagra 2009]:

- stealthiness: the watermark graph must have the structure of a normal control flow graph (CFG) to avoid detection;
- resilience: the watermark graph must be resistant to alterations in vertices and edges;
- diversity: the encoding algorithm must be able to output multiple, distinct graphs for the same identifier;
- frugality: the difference in size between the original program and the one with a watermark must be negligible;
- efficiency: the codec must execute in polynomial time, so it does not pose any hindrance to the process of producing (legal) copies of the software or to auditing it in a timely fashion if that is ever required.

2. Advances in software watermarking

[Venkatesan et al. 2001] proposed the first graph-based watermarking scheme that embeds the watermark in the CFG. After them, other authors proposed different codecs using the most diverse classes of graphs. An attractive graph-based watermarking scheme was introduced by Collberg, Kobourov, Carter, and Thomborson [Collberg et al. 2003], and afterward developed and improved upon by Chroni and Nikolopoulos [Chroni and Nikolopoulos 2012]. These latter authors proposed a watermark graph belonging to a subclass of the reducible permutation graphs introduced by the former authors. Though the mechanics of encoding and decoding the proposed watermark are well described in [Chroni and Nikolopoulos 2012], such a special subclass of reducible permutation graphs had not been fully characterized. We provided such a characterization, based solely on the topology of the graph. They were referred to as *canonical reducible permutation graphs*. We have also formulated a robust polynomial-time algorithm that, given a watermark with an arbitrary number $k \geq 0$ of deleted edges, either retrieves the encoded identifier or proves that to be an impossible task. This result was published in Algorithmica in 2019 [Bento et al. 2019a].

Our proposed characterization gave rise to a linear-time algorithm that can always detect an attack and restore the original graph when $k \leq 2$ edges are missing. Furthermore, we proved that $k \leq 5$ general edge modifications (removals/insertions) can always be *detected* in polynomial time, and that both bounds are tight. Jayme presented such results in the 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'13) [Bento et al. 2013]. We also published them in Discrete Applied Mathematics in 2018 [Bento et al. 2018], where further details about the resilience of canonical reducible permutation graphs against this sort of attack are given.

Our results have shown that, while their codec can be implemented in linear time and has some resistance to edge deletion attacks, we can only embed a canonical reducible permutation graph into a software's CFG by introducing in its source code some

rather artificial instructions such as "GOTO" statements, which may give rise to suspicion and improve an attacker's odds of success. These observations led us to propose a new watermark codec altogether.

Our proposed codec differed from the previous codec in a number of aspects. It employs randomization to attain a high level of diversity, which is closely related to the resilience of the watermarks against some forms of attack. In short, the structure of the watermarks produced by our scheme is affected by random choices that are made during the execution of the encoding algorithm, which gives rise to a number of distinct graphs encoding the same identifier. This feature makes it less likely that a watermark can be spotted through brute force comparisons — undertaken by specialized diff tools. Furthermore, our watermark is smaller than the previous ones and both encoding and decoding algorithms can be implemented to run in linear time. Finally, we can customize at will the number of edge removals that our watermarks can withstand. That is accomplished by means of an edge-to-bit bijection, along with a decoding procedure that is immune to error propagation, making it possible that standard bit-level error-correction techniques are employed in the decoding algorithm. We presented this codec in the XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2014).

3. Something was missing... for a long time!

The new codec described in Section 2 satisfies the properties of a good watermark scheme. Among these properties, the most difficult to get is stealthiness. Indeed, we are interested in producing watermark graphs that have a structure equivalent to the structure of the programs to be protected. But what exactly is the class of graphs resulting from structured programs' CFGs?

To be able to propose a stealthy watermark scheme, we first need to understand those graphs. We therefore presented a characterization of the class of structured program graphs, which we called *Dijkstra graphs*, after Dijkstras's structured programming concepts presented fifty years ago [Dahl et al. 1972]. Our characterization led not only to a greedy linear-time recognition algorithm for the class of Dijkstra graphs, but also to an isomorphism algorithm. The latter was based on defining a convenient code for a Dijkstra graph comprising a string of integers. Such a code uniquely identifies the graph, and two Dijkstra graphs are isomorphic if and only if their codes coincide. The code itself has size $O(n)$, for a Dijkstra graph of n vertices, and the time complexity of the isomorphism algorithm is also $O(n)$. These results were published in Discrete Applied Mathematics in 2019 [Bento et al. 2019b] and presented by Jayme at the II Workshop Franco-Brésilien de Graphes et Optimization Combinatoire (2016), at the 19th Haifa Workshop on Interdisciplinary Applications of Graphs (2019), and last but not least in a online seminar from the series of Graphs, Algorithms and Combinatorics seminars hosted by COPPE Sistemas (UFRJ) in 2020.

The characterization of Dijkstra graphs allowed us to propose more stealthy watermarking schemes. In fact, we describe a codec for graph-based software watermarking, where the code corresponds to the Dijkstra Graphs [Bento et al. 2017]. Thus, when protecting structured software, the resulting watermarked CFG will always belong to the class of Dijkstra graphs. The proposed watermark differs from all existing graph-based watermarks we know of, where "GOTO" statements are inevitable for their embedding.

Encoding and decoding algorithms run in linear time. The encoding algorithm employs randomization to produce distinct watermarks for the same identifier upon different executions. The watermark is small. There is a one-to-one correspondence between the edges of the watermark and the bits of the encoded identifier. Hence distortive attacks (whereby the watermark is damaged, but not removed) can be detected after the graph-to-identifier decoding process, and the correction of any flipped bit — up to some predefined number — can be carried out by standard error-correction codes.

4. Ongoing work and future directions

We are now working on embedding/extracting algorithms to complete the watermarking scheme. The biggest challenge in this context is to identify the best place to insert the watermark in the source code, and to be able to keep the watermark belonging to the Dijkstra graphs class even after its insertion in the source code. We recently presented a preliminary version of the embedding/extracting algorithms at the 2022 IEEE International Workshop on Metrology for Industry 4.0 and IoT [Bento et al. 2022].

Our huge, heartfelt thanks to Professor Jayme Szwarcfiter, without whom none of this would even have started, let alone produced so many fruits.

References

Asongu, S. A. (2021). Global software piracy, technology and property rights institutions. *Journal of the Knowledge Economy*, 12(20/018):1036–1063.

Bento, L. M., Boccardo, D. R., Machado, R. C., de Sá], V. G. P., and Szwarcfiter, J. L. (2018). On the resilience of canonical reducible permutation graphs. *Discrete Applied Mathematics*, 234:32 – 46. Special Issue on the Ninth International Colloquium on Graphs and Optimization (GO IX), 2014.

Bento, L. M., Boccardo, D. R., Machado, R. C., de Sá], V. G. P., and Szwarcfiter, J. L. (2019a). Full characterization of a class of graphs tailored for software watermarking. *Algorithmica*, 81:2899 – 2916.

Bento, L. M., Boccardo, D. R., Machado, R. C., Miyazawa, F. K., de Sá], V. G. P., and Szwarcfiter, J. L. (2019b). Dijkstra graphs. *Discrete Applied Mathematics*, 261:52 – 62.

Bento, L. M. S., Boccardo, D., Machado, R. C. S., Pereira de Sá, V. G., and Szwarcfiter, J. L. (2013). Towards a provably resilient scheme for graph-based watermarking. In Brandstädt, A., Jansen, K., and Reischuk, R., editors, *Graph-Theoretic Concepts in Computer Science: 39th International Workshop, WG 2013, Lübeck, Germany, June 19-21, 2013, Revised Papers*, pages 50–63. Springer Berlin Heidelberg.

Bento, L. M. S., Boccardo, D. R., Machado, R., de Sá, V. G. P., and Szwarcfiter, J. L. (2017). Marca d'água estruturada. In *XVII Simpósio Brasileiro de Seguraça Informação e de Sistemas Computacionais (SBSEG'17). SBC, Brasília, Brazil*, pages 388–399.

Bento, L. M. S., Machado, R. C. S., and Simões, F. S. (2022). Software watermark scheme. In *2022 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*, pages 306–310.

Chroni, M. and Nikolopoulos, S. D. (2012). An efficient graph codec system for software watermarking. In *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, pages 595–600.

Collberg, C., Kobourov, S., Carter, E., and Thomborson, C. (2003). Error-correcting graphs for software watermarking. *Lecture Notes in Computer Science*, 2880:156–167.

Collberg, C. and Nagra, J. (2009). *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Addison-Wesley Professional, 1st edition.

Collberg, C. and Thomborson, C. (1999). Software watermarking: Models and dynamic embeddings. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '99, pages 311–324, New York, NY, USA. ACM.

Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R., editors (1972). *Structured Programming*. Academic Press Ltd., London, UK, UK.

Goff, M. (2022). Software piracy 2022 stat watch.

Venkatesan, R., Vazirani, V., and Sinha, S. (2001). A graph theoretic approach to software watermarking. In Moskowitz, I. S., editor, *Information Hiding*, pages 157–168, Berlin, Heidelberg. Springer Berlin Heidelberg.