International Journal of IT & Knowledge Management (ISSN: 0973-4414)
Volume 12 • Issue 2 pp. 162-165 Jan 2019 - June 2019 www.csjournals.com

Comparative Analysis of WHLK and Dynamic Software Watermarking

Nishant Gupta, Shubhnandan S. Jamwal
Department of Computer Science and IT, University of Jammu

Introduction: Software piracy originates from the unauthorized copying of a copyrighted computer program. Looking at late 50s and 60s, the antecedents to software piracy were more focussed on selling hardware bundled with software. This had been an accepted practice for computer scientists to expand their programming knowledge by sharing programs. Currently, software piracy emerged as an issue with the increasing importance of the software industry and the introduction of personal computers, which has grown into a pervasive and global phenomenon. Recent studies (BSA, 2018) shows that 37% of the software programs installed in 2018 are pirated.This challenge has compelled severalpreventive and deterrent control measures which need to be employed in protecting the intellectual property.

In order to prevent software from piracy and unauthorized modification many techniques have been developed and among one of them is by inserting software watermarks, which is used to identify the intellectual property owner of a piece software through embedding some secret information into software as an identifier of the ownership of copyright. This watermark technique which isused to copy control of digital contents, for copyright protection and rights management has been studied and became active since mid-1990s and is also used in data integrity and data confidentiality. Watermarks need to be robust and invisible unlike data integrity and confidentiality applications. In recent years, number of watermarking techniques have been researched and applied to protect the copyrights of software codes but most of the work in this specific area is not published and kept as trade secrets.

Software watermarking

Software watermarks can be classified in different ways by their functions and properties.Software watermarks can be broadly divided into two categories: static and dynamic. Static watermarks are embedded in the code and/or data of a computer program, a trivial example is embedding a copyright notice in a string.Static watermarks are stored in parts of the program that are not generated during execution. This includes the executable section that contains instructions (code section) or executable that contains data like headers, strings, and debugging information.

Dynamic watermarking techniques store a watermark in a program's execution state, rather than in the program code itself, therefore they are resilient to semantics preserving transformations.The watermark is embedded in the running state of the program. The algorithm starts by mapping the watermark, which is a natural number, to a special data structure called Planted Plane Cubic Tree (PPCT). PPCT can be considered as a modified version of binary trees. The pointers between different nodes in the tree are chosen carefully to represent the watermark value. An offline code is used to build up the PPCT tree in the source code. By special addition of some fields, this code is linked to a base class in the program, and when the program is executed the PPCT will be constructed.

In this paper we have compared the Dynamic Software Watermarking method using Exception handling and WHLK model for the effectiveness.

Literature Review

E. N. d. L. F. Jorge and others [1] proposed a framework for embedding a software watermark using obfuscation and tamper-proofing techniques, called SensorWatermark in order to inhibit piracy of embedded software in sensor nodes in a WSN. The watermark scheme proposed is characterized as an

ordered sequence of obfuscations and guards implemented in the software, where such sequence identifies the software author.

H. Jiang, H. He and X. Wang [2] proposed the software watermarking algorithm based on Chinese remainder theorem, and introduces the authentication center to certify the watermark, to solve the problem that equation re-sorting algorithm are easily attacked by the random re-sequencing technology. Their experiment results shows that the algorithm is robust, as well as the code length and speed of program will not be affected, and the capability of new software watermarking algorithm is better than those of algorithms as equation reordering.

Jianhua Cheng and Yan Song [3], discussed dynamic map watermarking is a relatively new technique and proposed tamper resistant watermarking algorithm, based on PPCT hybrid coding structure under multiple constants. Their algorithm uses the internal structure of the watermark graph constants for encoding data to achieve watermark anti-tampering by creating one or more sub-procedures' functional dependency on the host program, on the real play of the tamper-resistant watermark effect to increase the difficulty of attackers, which can effectively protect the watermark.

Z. Tang and D. Fang [4] used a modified PPCT structure, a tamper-proof software watermark solution with code-based encryption. Changes to the source and object code are made to embed the watermark, and according to certain policies some parts of object code are encrypted with an en/decryption key which is highly coupled with object code to increase robustness and tamper-proof capability.

Some schemes statically embed watermark into program's text by, e.g., relocating registers or modifying program's abstract semantics [5,6]. Others focus on dynamic approaches in which the watermark is embedded in program's runtime behavior [7, 8, 9,10,11, 12, 13, 14]. It is very important for the software experts to make the techniques for the detection of the pirated copy because according to a report issued by the BSA, from 2015 to 2017 the worldwide unlicensed software rate was 37 percent, and the commercial value of unlicensed software was $46.3 billion globally [15].

WHLK versus Dynamic Software Watermarking

Yilong et.al[16] in their study proposed a dynamic software watermarking algorithm based on program exception handling. This algorithm includes watermark generation, embedding, and extraction processes. The algorithm encodes the binary watermark as an exception type sequence, which is embedded into the program source code by building the trigger condition and exception handling code.

The watermark is embedded by inserting the exception handling sequence in the source code of target program with the embedding key (several secret inputs of the program).Watermark extraction was directly performed during the dynamic execution of the program. The researchers have executed the program with the extraction key, monitor the exceptions triggered by the program, and obtained the embedded exception type sequence. Then, watermark was restored with watermark generation key.

Dynamic software watermarking algorithm is implemented by using various tools, such as code compression and encryption to verify the capability of the watermark to resist semantic transformation. Algorithm encodes the watermark as exception handing, involved in the program execution directly, and improves the robustness of watermark. The size and running time of the program increase linearly with the length of the watermark at the same time. Therefore, in order to ensure the imperceptibility of watermark, it was difficult to embed mass watermark in the program with smaller file and shorter running time. Also, the selection of embarking position has effect on the robustness of watermark, which requires watermark embarking one has enough understanding on the original program. Finally, it was important for embedding identification in the embedding exception handling to extract that watermark correctly. So, getting more reasonable and effective protection method of the identification was required in this research.

WHLK [17] is based on prevention of software from illicit copying forcing the software to be only used by an authorized running environment. We have proposed to embed a unique customer identifier in each copy of the software distributed which allows the software company to identify the individual that pirated the software.

Firstly, the user is required to submit his identification details which are being applied through algorithms to generate random unique key (string) and this key is embedded as a watermark in the program. Then, an automated program fetches the uniquely identifying characteristics of running environment such as the physical sequence numbers of the CPU, the main board, processor and other hardware information. These hardware characteristics and the License key are integrated to be used as parameters. All these parameters are encrypted using a pre-defined code generating a new random unique number called as Registration Code which is acknowledged to the user. Our model provides an integration of the privacy of users, security of information and license key, together for the control of the piracy. It enjoys a modular design and can be implemented by any machine with flexible configurations and windows X operating systems. Furthermore, the proposal allows flexible registration information definition. This Model not only makes it harder to create an additional available copy based on diversity, but also prevents illegal uses on the copy.

RESULTS AND OBSERVATIONS:

- We have observed a few observationson the basis of comparison between these two models that even though Yilong et.al[16]has embedded exception handling in the program to extract the watermark but no identification method has been incorporated for proper protection of the watermark. Whereas the same method has been implemented in the first phase of our research on WHLK. The identification details have been fetchedby an automated program and used as parameters to encrypt the code.
- Yilong et.al[16]mentioned that it is not possible to embed the largesize watermark in the program in a short time which, in comparison to WHLK, does not have any such constraint or restrictions. WHLK is flexible with the size of watermark.
- It has been observed that performance of software is not impacted by the variant size of watermark using WHLK model whereas Yilong's in his research has reported that when the size of the watermark is increased, the performance of the software in terms of program size and execution time reduces.In WHLK if size of watermarks changes, it will not impact the performance of the software.

Conclusion

It is visible from the comparison that WHLKseems to be much more effective in the protection of the software and the performance of software is not affected by the WHLK. WHLK seems to be a better option and more innovative which uses a no. of algorithms for implementing the protection of the software as compared to the dynamic watermarking using exception handling.

References

[1] E. N. d. L. F. Jorge, L. Pirmez, C. M. de Farias, R. d. O. Costa, D. R. Boccardo and L. F. R. d. C. Carmo, "SensorWatermark: a scheme of software watermark using code obfuscation and tamperproofing for WSN," 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), Dubrovnik, Croatia, 2015, pp. 916-922, doi: 10.1109/IWCMC.2015.7289205.

[2] H. Jiang, H. He and X. Wang, "Software watermark algorithm based on Chinese remainder theorem," IEEE Conference Anthology, China, 2013, pp. 1-4, doi: 10.1109/ANTHOLOGY.2013.6784841.

[3] Jianhua Cheng and Yan Song, "Dynamic map based on PPCT structure software watermark protection," World Automation Congress 2012, Puerto Vallarta, Mexico, 2012, pp. 133-136.

[4] Z. Tang and D. Fang, "A tamper-proof software watermark using code encryption," Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics, Beijing, China, 2011, pp. 156-160, doi: 10.1109/ISI.2011.5983991.

[5] P. Cousot and R. Cousot. An abstract interpretation-based framework for software watermarking. In Proceedings of the 31th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), pages 173–185, 2004.

[6] G. Myles and C. Collberg. Software watermarking through register allocation: Implementation, analysis, and attacks. In Proceedings of the 6th International Conference of Information Security and Cryptology (ICISC), pages 274–293, 2003

[7] C. Collberg, E. Carter, S. Debray, A. Huntwork, J. Kececioglu, C. Linn, and M. Stepp. Dynamic path-based software watermarking. In Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation (PLDI), pages 107–118, 2004.

[8] C. Collberg, C. Thomborson, and G. M. Townsend. Dynamic graph-based software watermarking. Technical Report TR04-08, Department of Computer Science, The University of Arizona, 2004.

[9] G. Myles and C. Collberg. Software watermarking via opaque predicates: Implementation, analysis, and attacks. Electronic Commerce Research, 6(2):155–171, 2006.

[10] G. Myles and H. Jin. Self-validating branch-based software watermarking. In Proceedings of the 7th International Workshop of Information Hiding (IH), pages 342–356, 2005.

[11] J. Nagra and C. Thomborson. Threading software watermarks. In Proceedings of the 6th International Workshop of Information Hiding (IH), pages 208–223, 2004.

[12] J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang. Experience with software watermarking. In Proceedings of the 16th Annual Conference of Computer Security Applications (ACSAC), pages 308–316, 2000.

[13] C. Ren, K. Chen, and P. Liu. Droidmarking: Resilient software watermarking for impeding android application repackaging. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (ASE), pages 635–646, 2014.

[14] R. Venkatesan, V. Vazirani, and S.Sinha. A graph theoretic approach to software watermarking. In Proceedings of the 4th International Workshop of Information Hiding (IH), pages 157–168, 2001.

[15] Unlicensed Software Use Still High Globally. Despite Costly Cybersecurity Threats. Accessed: May 2016. [Online]. Available: http://globals

[16] Y. Wang, D. Gong, B. Lu, F. Xiang and F. Liu, "Exception Handling-Based Dynamic Software Watermarking," in IEEE Access, vol. 6, pp. 8882-8889, 2018, doi: 10.1109/ACCESS.2018.2810058.

[17] Gupta, Nishant and others, WHLK: Framework for Software Authentication and Protection, Vol 7. No. 1, March 2014, page 69-80, IEEE African Journal of Computing & ICTs,ISSN 2006-1781.