

Literature Survey of Watermark Security for Android Apps

Rosy Babu¹, Neethu MT² and Annie Chacko³

¹ IVth year BTech Student, Department of Computer Science And Engineering, MBC CET ,
Peermade, Kerala, India
rosybabu8@gmail.com

² IVth year BTech Student, Department of Computer Science And Engineering, MBC CET ,
Peermade, Kerala, India
neethumt999@gmail.com

³ Assistant Professor, Department of Computer Science And Engineering, MBC CET ,
Peermade, Kerala, India
annievargh@gmail.com

Abstract

Smart Phones play an important role in our day to day life. There are many applications for Smart phones. But problem is that attacks on these apps are increasing widely. There are repackaged apps which share similar functionalities with the original apps, which makes them easily spread. To avoid this kind of attacks, we embed different kind of watermarks into these apps. So the embedding watermark should be errorless. We develop a new kind of watermark is called picture based watermarks into android apps. This provide the significant characteristics of pictures.

Keywords: Picture based watermarks, Collusion attack, Recognizer, Software watermarking.

1. Introduction

Smart Phones play an important role in our day to day life. There are many applications for Smart phones. But problem is that attacks on these apps are increasing widely. There are repackaged apps which share similar functionalities with the original apps, which makes them easily spread. To avoid this kind of attacks, we embed different kind of watermarks into these apps. So the embedding watermark should be errorless [1]. We develop

a new kind of watermark is called picture based watermarks into android apps. This provide the significant characteristics of pictures. Recent research shows that the android apps available in the market are repackaged ones.

Watermarking techniques are used mainly in the entertainment industry to identify multimedia files such as audio video files [2]. Watermark can be of two types-Static or Dynamic. Static watermarks are embedded in the code or data of computer program. Whereas dynamic watermark is stored in a program's execution state[1]. Static watermarking techniques embed the watermark in the target application executable, such as the initialized data section, the text section, and the symbol information. Moskowitz et al. [3] proposed to watermark a program by watermarking a media object and storing it in the code of the program. However, since these two algorithms are both based on reordering, they are easy to defeat: attackers simply have to reorder every "reorderable" list of items in the program themselves. Venkatesan et al. [8] described a graph theoretic approach to software watermarking, which marks a program by the addition of code for which the topology of the control-flow graph encodes a watermark. Then, implemented and evaluated the algorithm proposed

by, and showed that its fundamental dependence on static block marking leaves watermarked programs vulnerable to distortive attacks. A robust spread spectrum based software watermarking algorithm, which embeds a watermark by changing the statistical properties of a program. The algorithm extracts a frequency vector of instruction groups from the original program and then modifies the vector to embed the watermark. However, the transformations used to modify the frequency vector in the three known implementations of this algorithm.

In this paper, we design a picture-based watermark. It uses ASCII characters to represent pictures. To the best of our knowledge, this is the first picture-based watermark that represented by ASCII characters on smartphone apps[1].

2. Existing System

The following pages consist of the literature survey of the project. The overall purpose of this literature review is to demonstrate a knowledge of the existing body of research in a particular topic area.

In the paper, Software Watermarking Through Obfuscated Interpretation: Implementation and Analysis by Ying Zeng[15], a robust static software watermarking scheme, which can not only resist against semantics preserving transformations but also collusion attacks, is proposed based on obfuscated interpretation. A frequency vector of instruction groups extracted from the original program is modified to embed the watermark by translation and code insertion. Translation rules defined in obfuscated interpretation provide an alternative simple way to find enough substitution patterns for vector instruction groups, and thus enhance the efficiency of the scheme. Since the software program is protected by obfuscated interpretation, the embedded watermark is robust to various semantics-

preserving transformation attacks that modify the program code. Furthermore, as instructions are translated using randomly selected translation rules, attackers cannot recognize the location of a watermark by comparing differently watermarked versions of the same program.

In the paper, A Survey of Static Software Watermarking by James Hamilton and Sebastian Danicic[16], presented a survey of static software watermarking schemes from the basic, early patents to the latest register allocation based techniques. Previous studies have shown that static techniques are highly susceptible to semantics preserving transformation attacks and are therefore easily removed by an adversary. Software watermarking can be supplemented with other forms of protection, such as obfuscations or tamperproofing techniques, in order to better protect a program from copyright infringement and decompilation. Further research should focus on dynamic software watermarking algorithms as static watermarking schemes are not robust enough for intellectual property protection.

In the paper, Digital Watermarking Using Matlab by aseem Saxena, amit Kumar Sinha, shashank Chakrawarti, surabhi Charu[17], described three basic models along with their programming codes using Matlab Programming. These models have designed in such a way to enhance the properties and characteristics of Watermarking. There are basically two techniques or models that have developed for the encryption and decryption of digital Watermark. These watermarking without side information and Watermarking with side-information.

In the paper, Dynamic Path-Based Software Watermarking by C. Collberg E. Carter[18], described a new approach to software watermarking, path-based watermarking, which embeds the watermark in the runtime branching structure of the program. The idea is based on

the intuition that the (forward) branches executed by a program are an essential aspect of its computation and part of what makes the program unique. However, an obvious apparent drawback with using the branch structure of a program to encode information is that, in principle, the branch structure of a program can be modified quite extensively without affecting program semantics, using well-known transformations such as basic block reordering, branch chaining (where the target of a branch instruction is itself a branch to some other location), loop unrolling, etc. In order for a path based watermark to be resilient against such transformations, we must be able to either cause any such transformation to change the program semantics, or devise embedding techniques such that the watermark can survive such transformations. As we will see, path based watermarking lends itself well to error-correction and tamper-proofing significantly more so than most watermarking schemes proposed in the literature. Finally, since branches are ubiquitous in real programs, path-based watermarks are less likely to be susceptible to statistical attacks.

In the paper, AppMark: A Picture-based Watermark for Android Apps by Yingjun Zhang[19], described a picture-based watermark. It uses ASCII characters to represent pictures. To the best of our knowledge, this is the first picture-based watermark that represented by ASCII characters on smartphone apps. Here they implemented a prototype called AppMark that can automatically embed a picture-based watermark into Android apps. They made several evaluations on AppMark. The results show that the picture-based watermark is both effective and efficient.

3. Proposed System

It can be used to identify the original owner of the apps. There are two kinds of watermarks: static watermarks and dynamic watermarks. Static watermark embeds data or variables into apps. However, semantic-preserving transformation could break them. Dynamic watermark uses runtime information as the watermark such as running paths and memory status. Embedding watermarks in Android apps could potentially solve this problem. The embedded watermark can be used to identify the owner of apps. By extracting watermarks from apps and comparing them, we can identify the repackaged pairs. There are two kinds of watermarks: static watermarks and dynamic watermarks. Static watermark embeds data or variables into apps. However, semantic preserving transformation could break them. Once the watermarks are broken, we cannot extract the right one from the target apps. Dynamic watermark uses runtime information as the watermark such as running paths and memory status. AppInk embeds a graph-based dynamic watermark into Android apps. By constructing a special list of objects as the watermark and using the distance between the objects in the list to represent a number, AppInk could embed any number as the watermark into an app. However, the code for constructing the graph is easy to be identified.

Static watermarking techniques embed the watermark in the target application executable, such as the initialized data section, the text section, and the symbol information. Static watermarking techniques embed the watermark in the target application executable, such as the initialized data section, the text section, and the symbol information. From the existing literatures on software watermarking, know that although dynamic watermarking schemes are more robust than static ones, they have one main drawback, namely, the risk that the recognizer will be

affected by small changes to the execution environment such as packet delivery times in the network and the initialization of random number generators. Moreover, it is more difficult to implement the dynamic watermarking schemes, and both the watermark embedding and extraction need to execute the program. Hence despite the fact that current researches on software watermarking have made a great progress, the robustness of watermarks and the efficiency of watermarking schemes still need to be improved. In a nutshell, after a developer provides a picture as a watermark, we transform the picture into ASCII characters. Then we embed some pieces of code into the app to generate the ASCII characters dynamically in memory.

Once a watermark is generated, it is ready for extracting. To extract the watermark, we run the app and locate the watermark using a mark. The mark is a random and secret number which is set up previously in the process of embedding the watermark. Attackers do not know anything about the mark. In this way, even if an app is repackaged (and obfuscated), we could still extract the watermark from the repackaged apps. To combine watermarks with original apps, we design a picture-based watermark. Pictures have one special characteristic. That is, even if part of a picture is tampered, the picture could still be identified with high possibility.

Note that pictures are different from graphs. Once an edge or a node in a graph is tampered, the graph will no longer be the original one. However, pictures are different. Even if several pixels in a picture are tampered, people could still identify the pictures. In this way, the picture-based watermark will be more robust. Thus, we try to embed a picture into apps as the watermark. Different from other works, our watermark is a picture which is initially neither a sequence of numbers nor English letters. Since

we do not want to use the original file formats of pictures, we need to transform the pictures. We first generate an ASCII sketch of a picture. To make the watermark more robust, we divide the sketch into several parts. Different parts share redundant information. In this way, even if some parts of the sketch (watermark) are tampered, we can still know whether the watermark in an app is the same one as the one that original author supplies. Secondly, we automatically generate code that could draw the pictures in memory dynamically. Each part of the sketch is corresponding to a piece of code.

4. Conclusions

To mitigate this threat of repackaging, embedded a new kind watermark, called picture-based watermark, into Android apps. By making the inherent characteristics of pictures, it is resilient to obfuscation. The watermark code is highly combined with the original code, which is resilient to dependence analysis. Implemented a prototype call AppMark and evaluated its effectiveness and performance overhead. Results show that this approach is both effective and efficient. This paper has successfully discussed various techniques for the enhancement of various watermarking properties such as effectiveness, fidelity, robust, security etc. we have also presented through this paper various encryption and decryption models for the watermarking.

References

- [1] B. Markgraf, "U.s. smartphone users now number 129 million," <http://madmobilenews.com/u-s-smartphone-users-now-number-129million-823/>, 2013.
- [2] C. Gibler, R. Stevens, J. Crussell, H. Chen, H. Zang, and H. Choi, "Adrob: Examining the landscape and impact of android application plagiarism," in *MobiSys*, 2013.

- [3] J. Crussell, C. Gibler, and H. Chen, "Attack of the clones: Detecting cloned applications on android markets," *Computer Security–ESORICS 2012*, pp. 37–54, 2012.
- [4] S. Hanna, L. Huang, E. Wu, S. Li, C. Chen, and D. Song, "Juxtapp: A scalable system for detecting code reuse among android applications," in *DIMVA*, 2012.
- [5] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *CODASPY. ACM*, 2012, pp. 317–326.
- [6] E. Lafortune, "Proguard," <http://proguard.sourceforge.net/>, 2013.
- [7] S. Inc, "A specialized optimizer and obfuscator for android," <http://www.saikoa.com/dexguard>, 2013.
- [7] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation-tools for software protection," *Software Engineering, IEEE Transactions on*, vol. 28, no. 8, pp. 735–746, 2002.
- [8] S. A. Moskowitz and M. Cooperman, "Method for stega-cipher protection of computer code," Apr. 28 1998, uS Patent 5,745,569.
- [9] R. Venkatesan, V. Vazirani, and S. Sinha, "A graph theoretic approach to software watermarking," in *IH*, 2001.
- [10] C. Collberg and C. Thomborson, "Software watermarking: Models and dynamic embeddings," in *POPL*, 1999.
- [11] W. Zhou, X. Zhang, and X. Jiang, "Appink: watermarking android apps for repackaging deterrence," in *ASIA CCS. ACM*, 2013, pp. 1–12.
- [12] Mosaizer, "Textaizer pro, text mosaic creation at its best," <http://mosaizer.com/Textaizer/>, 2014.
- [13] B. S. Baker, "On finding duplication and near-duplication in large software systems," in *Reverse Engineering, 1995. IEEE*, 1995, pp. 86–95.
- [14] *Software Watermarking Through Obfuscated Interpretation: Implementation and Analysis* by Ying Zeng.
- [15] *A Survey of Static Software Watermarking* by James Hamilton and Sebastian Danicic.
- [16] *Digital Watermarking Using Matlab* by aseem Saxena, amit Kumar Sinha,shashank Chakrawarti, surabhi Charu.
- [17] *Dynamic Path-Based Software Watermarking* by C. Collberg E. Carter.
- [18] *AppMark: A Picture-based Watermark for Android Apps* by Yingjun Zhang.