ATLANTIS PRESS
Advances in Engineering Research (AER), volume 61
International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2017)

Static Software Birthmark based on Multiple Attributes

Guangye Shi
Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
1191479746@qq.com

Keywords: Software birthmark, Multiple attributes, Software theft.

Abstract. Software birthmarks have been proposed as a method for enabling the detection of programs that may have been stolen by measuring the similarity between the two programs. A birthmark is created from each program by extracting its native characteristics. The birthmarks of the programs can then be compared. However, the existing software birthmark is mainly software birthmark single attribute, they will play a good effect in the specific scene, but poor robustness in general single attribute software. For a single attribute of software birthmark has poor robustness, resistance to attack ability is not strong, multi attribute extraction software birthmark from point of view, this paper firstly proposed Java software related attribute analysis of hierarchical structure, on the basis of a multi attribute static software birthmark.

1. Introduction

Software theft causes major damage to the software industry. For example, cases of GPL-licensed software installation resulting in unintentional open-sourcing have been reported. To combat software theft, software birthmarks have been proposed as a theft-detection method [1, 2]. A software birthmark is a set of characteristic elements unique to a program. It is extracted from a program's binary code and used to evaluate the similarity between one program and another. Various types of birthmarks have been proposed, each focusing on different program characteristics. Different extraction methods and comparison methods have also been defined for each type of birthmark and have been evaluated according to those definitions

2. Related Work

Birthmarks are a concept that was proposed by Tamada et al. as a method for detecting theft [1, 2]. Features unique to a program that are contained in the program's binary code are extracted as birthmark information and used to measure similarity. Unlike software watermarks, there is no need for prior information embedding; features unique to the program are taken from the compiled binary and defined as the birthmark. Several different types of birthmark that focus on a different program features have been proposed [3-8].

Birthmarks ordinarily extract elements by examining features in the program. The extracted elements are maintained in a structure such as a set, sequence or graph, and the similarity computation method is defined in accordance with the structure used. For example, the Jaccard index [7], Dice's coefficient [4] or cosine similarity [6] are often used for sets, and the longest common subsequence [5, 8] is used for sequences. Methods for graphs are more complex; however, methods using graph isomorphism have been proposed [3, 9].

3. Multi Attribute Birthmark Construction

The existing software birthmark is a single attribute of Java to a certain level or its deformation based on various types of attributes can describe the program from different angles, to a certain extent, software birthmark is extracted to reflect the characteristics of the program. Extraction of specific semantic software birthmark is often difficult to get an accurate representation of a program based on

Copyright © 2017, the Authors. Published by Atlantis Press.
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

a single attribute, different attributes of different program semantics preserving transformation effect of different degrees, separately from the Java instruction level and method level were analyzed and extracted the relevant directive attributes this two kinds of software birthmark for different deformation resistance degree of confusion is not the same. Considering the attributes at different levels of program impact and characterization of different types of code obfuscation algorithm on attribute, we propose a multi attribute Java static software birthmark based on comprehensive.

According to the needs of the application, these attributes can be used to describe the characteristics of the program. We can also generate new deformation attribute information on its basis, to build a software birthmark. Selecting the suitable measure to describe the software birthmark attribute, is the key to achieve good classification results. Software birthmark based on credibility and robustness, combined with the existing literature to describe the birthmark, from different levels of Java program related attribute selection and analysis.

(1) Application level attributes

Inheritance is one of the two characteristics of object-oriented Java software, and the class file inheritance graph can be used to construct the application class. Class file inheritance graph reflects the dependencies within a Java application class files, embodies the semantic program overall, having a high degree of abstraction. For most of the Java method and instruction level obfuscation it is robust. So the statistics application class number F1 class inheritance graph depth F2, edge, the number of F3 F4, the inheritance graph with maximum degree and out degree of information f5f6, as described in class inheritance graph information, denoted as A1= (F1, F2, F3, F4, F5, F6).

(2) Class file attribute

Many method calls are for a single class file, and the method call relations and the number represents the logical relations of the whole class level. The relationship is difficult to carry on the change, therefore calling the method of single class times as a class attribute is used to describe the level of F7, denoted by A2= (F7).

(3) Method level attributes

Method level attributes can be analyzed and described from the following perspectives. The Java compiler usually is not optimized to call library function. It is difficult without changing the program behavior under the premise of replacing the library function call, the statistical library function call frequency F8. Java method call usually use many types such as Vector, Stack, List, Set and other packaging containers. The container encapsulates some data structures in the program, and complete the corresponding function. These containers is difficult to remove by means of code obfuscation. Statistical interference graph nodes and edges are determined by loop or branch, the method of loop cycle frequency and conditional branch statement frequency f10f11 as the method of internal logic node an attribute structure, denoted as A3= (F8 F9, F10, F11).

(4) Instruction level attributes

Java virtual machine to meet the specifications of has a total of 204 instructions. Different types of instructions in the Java program to complete the different function. Because the Java language is designed according to the object-oriented idea, the program has a large number of instructions, such as the creation of the object, the access to the object variables and the call object method, as shown in table 1. The core property represents a Java program with these operating instructions, and reflect the characteristics of the Java program from the command level. At the same time, the compiler and the code obfuscation is not to replace these instructions, instructions related to statistical object frequency, which is denoted as A4= (F12, f15,... , F22).

The comprehensive analysis shows that the program from the different level of the hierarchy attribute A1, A2, A3, A4, relevant information on these properties were quantified to define a 22 dimensional VBirthmark=<f1,... , f22>, which is suspected pirated software to determine the premise.

Table 1. Operation instruction

Object instruction name	Object instruction function description
New	create new object
Putfield getfield putstatic getstatic	Member variable access
Invokevirtual,invokestatic, invokeinterface, invokespecial	Method call
Checkcast,instanceof	Type check

4. Decision Model Based On Support Vector Machine

Piracy and non-piracy problem between two software can be converted into a two classification problems, the structural risk minimization principle of small sample statistical learning theory proposed by SVM Vapnik [10], the basic idea is to find an optimal separating hyperplane, the two types of samples are separated, the minimum error probability Classification. To obtain the best generalization ability we maximize the classification margin. It has special advantage in solving small sample learning, nonlinear and high dimensional pattern recognition.

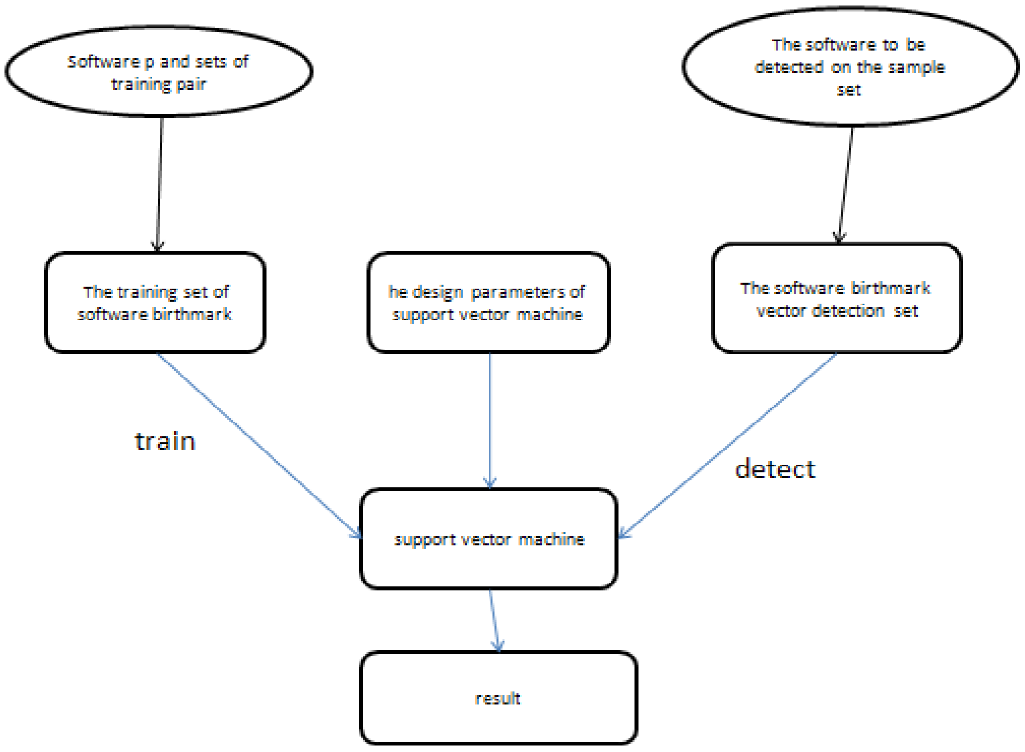
Software p and sets of training pair
The software to be detected on the sample set
The training set of software birthmark
he design parameters of support vector machine
The software birthmark vector detection set
train
detect
support vector machine
result

Fig.1 Pirate detection process based on support vector machine

Specific process as shown in Figure 1, the process based on the detection process is divided into two main parts of the training and prediction, through support vector machine. The first extraction software and its software birthmark vector piracy and non-pirated links, extract the birthmark vector from multi-attribute comprehensive perspective, and consistency vector dimension of the birthmark. The pirated software of pirated software on the birthmark and non-vector combination as training data set for training the excellent performance of support vector machine. Through the data set of training support vector machine parameters. Finally, the optimal support vector machine (SVM) with the best generalization performance is obtained, and the relationship between the program P and the suspected pirate program is determined.

Due to the diversity of the software itself and the complexity of the composition of the software, the SVM nonlinear decision making is generally used to classify it. The use of software birthmark determine software whether this kind of nonlinear classification of piracy, first through the nonlinear mapping in X, the input data from the original feature space is mapped to a high dimensional feature space, and then use the linear support vector machine theory to establish the optimal classification face the classification discrimination.

Select the Java executable program Notepad.jar as the benchmark is denoted by Pl, and were used to confuse the tool Sandmark, Smokescreen, ZKM, Jarg optimization tools to optimize the confusion and distortion, generating 52 Notepad deformation process, denoted as P2, P3.., P53, after collecting 16 programs that have no relationship with Notepad.jar program as the training and prediction program set, denoted as B1-B16. The use of SVM for classification and identification of a certain number of training samples, but because of the deformation of the Java program tools are limited, the following training and prediction of the sample set is used.

The training set: P1-P30 with pair combination, and extract the corresponding software birthmark vector V, denoted by (Vp1, Vp2), (Vp1, Vp3},..., (Vp30, Vp1) a total of 900 sets of vectors as training data set for suspected piracy. Combine P1-P30 of extraction and B1-B16, software birthmark vector, denoted by (Vp 1, Vb 1),(Vp1, Vb2)... , (Vp30, Vb16) a total of 480 sets of vectors as non-pirated training data sets.

Test set: P1-P30 and P31-P53 combination, and extract the corresponding software birthmark vector V, denoted by (Vp 1, VP31), (Vp1, Vp32),..., (Vp30, Vp53) a total of 690 sets of vectors as suspected pirated test data set. P31 -P53 and B1-B16 combination, and extract the software birthmark vector, denoted by (Vp31, Vb1), (Vp2, Vb2),..., (Vp53, Vb16) a total of 368 sets of vectors as the test data set of non-pirated.

After the training and test data sets are constructed, the data sets are processed using LibSVM related tools. First a simple zoom operation using the svmscale.exe normalized all the data of the training set and test set, its purpose is: 1) to avoid some characteristic value range is too large while some characteristic value range is too small; 2) to avoid numerical difficulties in training in order to compute the kernel function and the computation of inner product. Data is usually scaled to [-1, 1] or [0, 1]. And the use of svmtrain.exe to train the support vector machine and related kernel functions and corresponding parameters, with the aim of training the optimal classification model, to predict the test set data using svmpredict.exe the next step. This paper selects 900+480=1380 software birthmark vector group as training data set for training support vector machine, and a total of 1058 groups of test data are analyzed and predicted by support vector machines after the completion of the training. Since SVM requires the kernel function K (x, y) to implement the mapping. In this paper, based on the above data set, we use four kinds of kernel functions in the LibSVM toolkit to test the default parameters of the software.

Table 2. kernel function test results

kernel function	accuracy
$K(x,y)=x^T *y$	61.1209%
$K(x,y)=(r*xy+b)^d,r>0,d=1,2,\dots$	87.1256%
$K(x,y)=\exp(-r*\lvert x-y\rvert^2),r>0$	58.2589%
K(x,y)=tanh(r*(xT*y)+b)	63.8903%

From the table data shows a linear kernel function, radial basis function (RBF) kernel function, Sigmoid kernel function of such classification is poor, polynomial kernel function in higher classification accuracy, so we choose the polynomial kernel function for further experiment. According to the principle of choosing the important parameter r and the penalty factor C, the classification performance of the polynomial kernel is tested under different parameters. The experimental results shown in the table, when the misclassification penalty factor C was 2, can be found in table R from any value, the correct rate will reach a relatively high level; when R was 0.015, C was found that the correct rate will reach a higher the level. These can be found when C is 2, R is 0.015, and the overall classification performance of SVM reached the highest value 88.94%.

Table 3. Svm performance test results

Accuracy						
	0.0195	0.039	0.078	0.015	0.031	0.062
2	88.19%	85.73%	84.31%	88.94%	87.81%	85.63%
4	86.11%	82.23%	83.12%	79.97%	82.21%	72.08%
8	81.63%	80.97%	82.25%	77.12%	70.76%	66.56%
16	70.24%	67.78%	69.38%	65.53%	62.21%	60.56%
32	61.58%	58.57%	56.74%	55.66%	52.31%	50.91%

5. Conclusion

This paper combines software birthmark and Similarity judgment unification, based on the existing literature of various birthmark of the premise, from different levels and analyzed the related properties of program, proposed a multi attribute comprehensive static software birthmark. SVM training based on birthmark extraction, and the use of training model of SVM to distinguish the classification of pirated software. The experimental results show that there are some differences between the results of SVM prediction for different SVM kernel functions and their intrinsic parameters. The use of SVM model classification can overcome the artificial threshold setting and the selection of similarity description model, which may lead to the inconsistency of final decision results. Experiments show that the method is effective and can be further improved in the following research work.

References

[1] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, “Design and evaluation of birthmarks for detecting theft of Java programs,” in Proc. IASTED International Conference on Software Engineering (IASTED SE 2004), February 2004, pp. 569–575, (Innsbruck, Austria).

[2] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, “Java birthmarks —detecting the software theft —,” IEICE Transactions on Information and System, vol. E88-D, no. 9, pp. 2148–2158, September 2005.

[3] P. Chan, L. Hui, and S. Yiu, “Heap graph based software theft detection,” IEEE Transactions on Information Forensics and Security, vol. 8, pp. 101–110, October 2012.

[4] S. Choi, H. Park, H. il Lim, and T. Han, “A static API birthmark for windows binary executables,” Journal of Systems and Software, vol. 82, no. 5, pp. 862–873, May 2009.

[5] Y.-C. Jhi, X. Wang, X. Jia, S. Zhu, P. Liu, and D. Wu, “Value-based program characterization and its application to software plagiarism detection,” in Proc. the 33rd International Conference on Software Engineering (ICSE 2011), 2011, pp. 756–765.

[6] C. McMillan, M. Grechanik, and D. Poshyvanyk, “Detecting similar software applications,” in Proc. the 34th International Conference on Software Engineering (ICSE 2012), 2012, pp. 364–374.

[7] D. Schuler, V. Dallmeier, and C. Lindig, “A dynamic birthmark for Java,” in Proc. of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), 2007, pp. 274–283.

[8] H. Park., H. il Lim, S. Choi, and T. Han, “A static Java birthmark based on operand stack behaviors,” in International Conference on Information Security and Assurance (ISA 2008), 2008, pp. 133–136.

[9] H. il Lim, H. Park, S. Choi, and T. Han, “Detecting theft of Java applications via a static birthmark based on weighted stack patterns,” IEICE Transactions on Information and System, vol. E91-D, no. 9, pp. 2323–2332, September 2008

[10] Nello Cristianini, John Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods [M]. Cambridge University Press, 2000.