DBPD: A Dynamic Birthmark-based Software Plagiarism Detection Tool

Zhenzhou Tian, Qinghua Zheng, Ming Fan, Eryue Zhuang, Haijun Wang, Ting Liu*
Ministry of Education Key Lab For Intelligent Networks and Network Security
Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, 710049, China
{zztian, fanming.911025, hjwang.china}@stu.xjtu.edu.cn; {qhzheng, tingliu}@mail.xjtu.edu.cn; zhuang8225@126.com

Abstract: *With the burst of open source software, software plagiarism has been a serious threat to the software industry. In this paper, we present the demo tool DBPD: **D**ynamic **B**irthmark-based Software **P**lagiarism **D**etection. Major features of DBPD could be summarized as: 1) dynamic birthmark. The execution process of software is captured to generate the birthmark reflecting intrinsic properties of software; 2) high availability. It is available for cross-platform and binary executable's plagiarism detection; 3) customizable. The birthmarks, similarity calculation metrics and detection criteria are configurable. The DBPD is implemented using C++ and Java, and currently can work under both Windows and Linux system. Three dynamic birthmarks are implemented in DBPD to identify the software according to its instruction, stack operation and system call.*

Keywords: software plagiarism detection, dynamic birthmark

I. INTRODUCTION

Free and open source software projects allow users to use, change and distribute software under certain types of license such as the well-known GPL. However, driven by the huge commercial interests, some individuals and companies incorporate third party software or libraries into their own products without respecting the licensing terms. Recent incidents include the lawsuit against Verizon by Free Software Foundation for distributing Busybox in its FIOS wireless routers [1], and the crisis of Skype's VOIP service for the violation of licensing terms of Joltid. The unavailability of source code and the existence of powerful automated semantic-preserving code transformation tools, make the plagiarism an easy to implement but difficult to detect thing.

Software birthmark, a set of characteristics extracted from a program that reflect the program's intrinsic properties and that can be used to uniquely identify the program, is a promising way for solving the plagiarism detection problem. However, despite the tremendous progress of birthmark based plagiarism detection approaches, seldom tools are publically available. The rare few tools as far as we find are SandMark [2], Stigmata [3] and Birthmarking [4]. The former two are static birthmark based which are believed to be fragile faced with semantic-preserving code obfuscation techniques, and the last one is dynamic birthmark based which is believed to have better performance than the previous two static birthmarks, yet they all suffer the problem of language dependence, since they're only valid for java programs. Also, there are some mature tools such as the JPlag [5] that target at source code which is not always available, since plagiarists are not likely to provide their source code before certain evidences are collected. Thus more powerful and practical tools are in urgent needs to fill the gap of birthmark based plagiarism detection research and practice.

It is a generally accepted fact that dynamic birthmarks being abstractions of runtime behaviors are believed to be more accurate reflections of program semantics than static birthmarks. Therefore, we implement a demo tool DBPD for plagiarism detection using dynamic birthmark techniques. Three dynamic birthmarks are implemented in DBPD to identify the software, including DKISB (dynamic key instruction sequence birthmark) [6], SODB (call stack operation dynamic birthmark) [7] and SCSSB (system call short sequence birthmark) [8]. Since all of them can work directly on binary executables, DBPD can analyze various programing languages.

*Ting Liu is the corresponding author. The research was supported in part by National Science Foundation of China (91118005, 91218301, 61221063, 61203174), 863 Program (2012AA011003), The Ministry of Education Innovation Research Team (IRT13035), Key Projects in the National Science and Technology Pillar Program of China (2012BAH16F02).

II. TOOL OVERVIEW AND IMPLEMENTATION

A. Tool Overview

Fig.1 shows the overview of the DBPD. It consists of three main modules: the dynamic analysis module, the birthmark generator, the similarity calculator and decision maker. The modular architecture qualifies it with good scalability of easily introducing new kinds of dynamic birthmark methods.

Given two binary executables the plaintiff (original program), the defendant (suspicious program) and a set of inputs, DBPD executes both programs with the same input one by one. Meantime, the dynamic analysis module monitors the executions, performs dynamic analysis and collects execution profiles containing three kinds of events: key instructions, stack operations, and system calls. After sequences of both plaintiff and defendant programs are available, they are fed into the birthmark generator where noises are filtered, valid

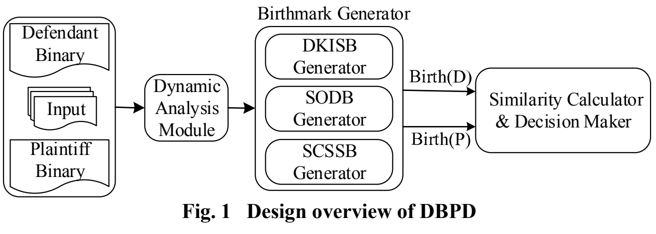

Fig. 1 Design overview of DBPD

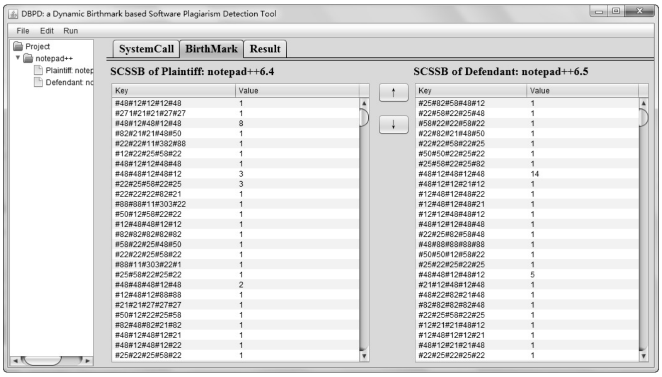

Fig. 2 DBPD: the dynamic birthmark based plagiarism detection tool

execution traces are constituted and either of the three kind birthmarks are extracted according to user configuration. Next, similarity scores are computed between the birthmarks of plaintiff and defendant in the similarity calculator. Finally, the decision maker judges whether the defendant is innocent or guilty according to the scores calculated and a given threshold.

B. Implementation

The dynamic analysis module is implemented based on the dynamic instrumentation framework Pin [10]. It consists of three sub-modules: DKISExtractor, StackTracer and SysTracer. They are responsible for the monitoring, analysis and collection of key instructions, stack operations and system calls respectively.

Three sub-generators implemented in the birthmark generator are in charge of the extraction of the three kind dynamic birthmarks accordingly. The DKISB [6] is extracted using k-gram algorithm from dynamic key instructions which refer to instructions that are both value updating and input-correlated. Since the underlying object that DKISB operates on is each assembly instruction, thus endows DBPD the ability of cross-platform plagiarism (for example, plagiarizing programs originally in linux platform to windows) detection. The SODB [7] is generated by analyzing the behavior of call stack during program executions. It utilizes the law of push and pop operation of call stack to uniquely identify a program, and believes that the laws of homologous programs are also the same. The SCSSB [8] is extracted from system call sequences which were originally widely used for intrusion detection to detect irregularities in the behavior of a program. Despite the high detection ability of DKISB, it suffers the scalability problem. For the other two birthmarks, only method calls need to be monitored, thus they have much lower overhead and better scalability. Therefore, for relatively larger programs, the other two birthmarks are preferred. This allows users to have more choices according to their requirements.

Four different similarity metrics including Cosine Distance, Jaccard Index, Dice Coefficient and Containment are supported in the similarity calculator. There's a default metric for each kind of birthmarks, but allows users to specify other metrics as the case may be. C++ is used for the implementation of the dynamic analysis module. The user interface and all the other modules are implemented in Java. Benefit from the support of pin for both the Windows and Linux systems, as well as the platform independence of Java, DBPD is able to work under both systems. Fig. 2 shows the interface of our DBPD.

TABLE I. Detection ability of three birthmarks adopting the default detection threshold of 0.25

	Precision	Recall	F-Measure
DKISB	1.00	0.96	0.98
SODB	1.00	0.98	0.99
SCSSB	1.00	0.83	0.91

III. Evaluation Setup

We evaluated the detection performance of DBPD under both Windows and Linux with plenty of experimental objects. Specially, all the three birthmark techniques implemented are assessed for ability of recognizing plagiarism utilizing various semantic-preserving code transformation techniques including adopting different compilers (llvm and gcc), using powerful tools such as the shelling tools (like UPX, ASProtect etc.) and the obfuscation tools (like SandMark, Allatori etc.), and the ability of distinguishing independently developed programs using plentifully programs that have similar and different functionalities. Totally 186 different versions generated from 38 software are used for the evaluation. As summarized in Table I the precision, recall and F-Measure values of the three birthmark techniques, we can see that all the birthmark methods show high detection ability with rather low false classification rates. The usability of DBPD is also confirmed by eight intern students with no prior experience of plagiarism detection.

IV. Conclusion and Future Work

In this demo, we present DBPD, a tool for software plagiarism detection using dynamic birthmarks. To the best of our knowledge, it's and will be the first publically available plagiarism detection tool that can handle binary executables directly, and the first tool that support cross-platform plagiarism detection. In the future work, more static and dynamic birthmark techniques such as the thread-aware birthmarks [9] will be integrated to improve detection accuracy, and more programming languages will be supported in DBPD.

References

[1] http://sourceauditor.com/blog/tag/lawsuits-on-open-source/.

[2] SandMark. http://sandmark.cs.arizona.edu/.

[3] Stigmata. http://stigmata.sourceforge.jp/implementation.html.

[4] Birthmarking. https://www.st.cs.uni-saarland.de/birthmarking/.

[5] JPlag. http://jplag.ipd.kit.edu/.

[6] Tian Z, Zheng Q, Liu T, et al. DKISB: Dynamic Key Instruction Sequence Birthmark for Software Plagiarism Detection[C]. In: HPCC'13.Zhang Jia Jie, Hu Nan: IEEE, 2013.

[7] Fan M, Tian Z, Liu T, et al. SODB: A Novel method for Software Plagiarism Detection based on Stack Operation Dynamic Birthmark[C]. In: CTCIS'14. Hu Bei, 2014. Unpublished.

[8] Wang X, Jhi Y, Zhu S, et al. Detecting Software Theft via System Call Based Birthmarks[C]. In:ACSAC'09.Washington DC, USA: IEEE, 2009.

[9] Tian Z, Zheng Q, Liu T, et al. Plagiarism Detection for Multithread Software Based on ThreadAware Software Birthmarks[C]. In: ICPC'14. Hyderabad, India,: ACM, 2014.

[10] Pin. https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool.