International Journal of Scientific Research and Engineering Development-– Volume 2 Issue 6, Nov- Dec 2019

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Exploiting Semantics-Based Plagiarism Detection Methods

Sajin R Nair[1], Smita C Thomas[2]

[1]P G scholar, Dept. of CSE, Mount Zion College of Engineering, Kadammanitta, Kerala, India
Email:sajinrnair1995@gmail.com
[2]Research Scholar,Vels University

Email: smitabejoy@gmail.com

----------**********----------

Abstract:

Manycomplex networked systems exhibit natural divisions of network nodes. Each division, or community, is adversely connected subgroup. Such community structure not only helps comprehension but also finds wide applications in complex systems. Moreover, existing proposed applications of software community structure have not been directly compared or combined with existing software engineering practices. Comparison with baseline practices is needed to convince practitioners to adopt the proposed approaches. However today's vetting mechanisms are slow and less capable of catching new threats author have implemented a set of tools collectively called TOB-PD (TOB based Plagiarism Detection tool) by applying TOB to three existing representative dynamic birthmarks, including SCSSB (System Call Short Sequence Birthmark), DYKIS (DYnamic Key Instruction Sequence birthmark) and JB (an API based birthmark for Java). This experiments conducted on large number of binary programs show that proposedapproach exhibits strong resilience against state-of-the-art semantics-preserving code obfuscation techniques. Comparisons against the three existing tools SCSSB, DYKIS and JB show that the new framework is effective for plagiarism detection of multithreaded programs. The tools, the benchmarks and the experimental results are all publicly available.

Keywords **—Multithreaded programming, Android malware,Software plagiarism, Birthmark, SCSSB (System Call Short Sequence Birthmark), DYKIS (DYnamic Key Instruction Sequence birthmark), JB (an API based birthmark for JavTOB-PD (TOB based Plagiarism Detection tool).**

----------**********----------

I. INTRODUCTION

The growth of Android devices brings in a vibrant application ecosystem. Millions of applications (appforshort) have been installed by Android users around the world from various app markets. Prominent examples include Google Play, Amazon Appstore, Samsung Galaxy Apps,and tens of smaller third-party markets. Software plagiarism, ranging from open source code reusing to smartphone app repacking, severely a□ect both open source communities and software companies.Despite the tremendous progress in software birthmarking, For example, birthmarks extracted from multiple runs of the same multithreaded programs can bevery different due to the inherent non-determinism of thread scheduling. In this case software birthmarking fails to declare plagiarism even for simply duplicated multithreaded programs.Despite the tremendous progress in software birthmarking, the trend towards multithreaded programming greatly threatens its effectiveness, as the existing approaches remain optimized for sequential

programs. Birthmarks extracted from multiple runs of the same multithreaded programs can be very different due to the inherent non-determinism of thread scheduling.

The basic research problem for code similarity measurement techniques is to detect whether a component in one program is similar to a component in another program and quantitatively measure their similarity. A component can be a set of functions or a whole program. Existing methods include clone detection, binary similarity detection, and software plagiarism detection. While these approaches have been proven to be very useful, each of them has its shortcomings.software plagiarism detection technology, a new trend in software development greatly threatens its effectiveness. The trend towards multithreaded programs is creating a gap betauthoren the current software development practice and the software plagiarism detection technology, as the existing dynamic approaches remain optimized for sequential programs. Due to the perturbation caused by non-deterministic thread scheduling, existing birthmark generation and comparison are no longer applicable to modern software with multiple threads.

In this paper, compare different technologies or methods that can be used to find out the plagiarized softwares. And also find out best one from the comparison.

II. LITERATURE SURVEY

K. Chen, et al.[5] propose a new designs of vetting techniques have recently been proposed by there search community for capturing new apps associated with known suspicious behavior, such as dynamic loading of binary code from a remote untrusted authorbsite operations related to component hijacking Intent injection , etc.
All these approaches involve a heavyauthoright information-flow analysis and require a set of heuristics that characterize the known threats. Theyoftenneedadynamicanalysisinadditiontothestati cin section performedonappcode and further human interventions to annotate the code or even participate in the analysis. Moreover, emulators that most dynamic analysis tools employ can be detected and evaded by malware. Mass vetting at scale. Based on this simple idea, developed a novel, highly-scalable vetting mechanism for detecting repackaged Android malware on one market or cross markets.

Z. Tian, T. Liu, Q. Zheng, M. Fan, E. Zhuang, and Z. Yang, [4] Despite the tremendous progress in software birthmarking, the trend towards multi-threaded programming greatly threatens its e□ectiveness, as the existing approaches remain optimized for sequential programs. For example, birthmarks extracted from multiple runs of the same multithreaded programs can be very different due to the inherent non-determinism of thread scheduling. In this case software birthmarking fails to declare plagiarism even for simply duplicated multithreaded programs. In this paper, author introduce a thread-aware dynamic birthmark called TreSB (Thread-related System call Birthmark) that can e□ectively detect plagiarism of multithreaded programs. Being extracted by mining behavior characteristics from thread related system calls, TreSB is less susceptible to thread scheduling as these system calls are sources that impose thread scheduling rather than being a□ected. In addition, unlike many approachesour approach operates on binary executables rather than source code. The latter is usually unavailable when birthmarking is used to obtain initial evidence of software plagiarism. Author have implemented a prototype based on the PIN (Luk et al., 2005) instrumentation framework, and conducted extensive experiments on an publicly available benchmark1 consisting of 234 versions of 35 di□erent multithreaded programs. Our empirical study shows that TreSB and its comparison algorithms are credible in diferentiating independently developedprograms,andresilienttomoststate-of-the-artsemanticspreserving obfuscation techniques implemented in the best commercial and academic tools such as SandMark and DashO . In addition, a comparison of our method against two recently proposed thread-aware birthmarks show that TreSB outperforms both of them with respect to any of the three performance metrics URC, F-Measure and MCC.

The work in [2] introduces a thread-aware dynamic birthmark called TreSB (Thread-related

System call Birthmark) that can effectively detect plagiarism of multithreaded programs. Unlike many approaches ,TreSB operates on binary executables rather than source code. The latter is usually unavailable when birthmarking is used to obtain initial evidence of plagiarism. The extensive experiments conducted on a publicly available benchmark consisting of 234 versions of 35 multithreaded programs show good resilience and credibility of TreSB. A comparison of TreSB against two recently proposed thread-aware birthmarks shows that TreSB outperforms both of them. System calls recorded during program execution are a favorable basis for extracting high quality birthmarks. The hypothesis is that modifications of system calls usually leads to incorrect program behavior, and therefore a birthmark generated from system calls can be used to identify stolen programs even after they have been modified. Also previous empirical study shows that the system call based birthmarks are resilient against various obfuscation techniques. Yet as illustrated in , these birthmarks become no longer efficient for multithread programs due to the nondeterminism of thread scheduling. Based on similar principle, author also generate birthmarks from system calls. Yet rather than using all the calls, author extract TreSB just from the thread-related system calls, which are essential to the semantics and correct executions of a multithreaded program. A random or deliberate modification to the calls can result in very subtle errors and therefore they are the least possible code to be changed. More importantly, these calls are the source to impose thread interleaving rather than being affected by the nondeterminism, thus a birthmark extracted from these calls is less susceptible to thread scheduling. Specifically, author treat system calls that accomplish tasks including thread and process management (such as creation, join and termination, capability setting and getting), thread synchronization, signal manipulating, as authorll as thread and process priority setting, as thread-related.

L. Luo, J. Ming, D. Wu, P. Liu, and S. Zhu[1]binary-oriented, obfuscationresilient method named CoP. CoP is based on a new concept, longest common subsequence of semantically equivalent basic blocks, which combines rigorous program semantics with longest common subsequence based fuzzy matching. Specifically, author model program semantics at three different levels: basic block, path, and whole program. To model the semantics of a basic block, author adopt the symbolic execution technique to obtain a set of symbolic formulas that represent the input-output relations of the basic block in consideration.Then calculate the percentage of the output variables of the plaintiff block that have a semantically equivalent counterpart in the suspicious block. Author set a threshold for this percentage to allow some noises to be injected into the suspicious block. At the path level, author utilize the Longest Common Subsequence (LCS) algorithm to compare the semantic similarity of two paths, one from the plaintiff and the other from the suspicious ,constructedbased on the LCS dynamic programming algorithm, with basic blocks as the sequence elements. By trying more than one path, author use the path similarity scores from LCS collectively to model program semantic similarity. Notethat LCS is different from the longest common substring. Because LCS allows skipping non-matching nodes, it naturally tolerates noises inserted by obfuscation techniques. This novel combination of rigorous program semantics with longest common subsequence based fuzzy matching results in strong resiliency to obfuscation. Author have developed a prototype of CoP using the above method. Author evaluated CoP with several different experiments to measure its obfuscation resiliency, precision, and scalability.Benchmark programs, ranging from small to large real-world production software, authorre applied with different code obfuscation techniques and semantics-preserving transformations, including different compilers and compiler optimization levels. Author also compared our results with four state-of-the-art detection systems, MOSS, JPLagBdiff and DarunGrim2 where MOSS and JPLag are source code based, and Bdiff and DarunGrim2 are binary code based. Our experimental results show that CoP has stronger resiliency to the latest code obfuscation techniques as authorll as other semantics-preserving transformations; it can be applied to software

plagiarism and algorithm detection, and is effective and practical to analyze real-world software.

III. PROPOSED SYSTEM

In this paper propose TOB (Thread-Obliviousdynamic Birthmark), a framework that can revive existing dynamic birthmarks such as SCSSB(System Call Short Sequence Birthmark),DYKIS(DYnamic Key Instruction Sequence birthmark), JB(an API based birthmark for Java) to handle multithreaded programs. The program is a test case in the AUTHORT project with slight modifications. Author apply two typical plagiarism detection algorithms on multiple runs of this program. If the existing approaches fail to claimplagiarism on different runs of the same small program, even without any code modifications, the capability of such approaches is in doubt. Dynamic birthmarks usually give quantitative measurement betauthoren 0 and 1 to indicate the similarity betauthoren two runs. A value of 1 indicates identicalness and 0 refers to complete difference. The measurement is given by applying metrics, such as Cosine distance, Jaccard index, Dice coefficient and Containment on the birthmarks obtained by a pair of executions. Tables 1a and 1b show the experimental results of SCSSB and DYKIS, where the column and row headings indicate the number of threads and the evaluation metrics, respectively. When there is only one thread, the program becomes sequential. Without non-deterministic thread scheduling, the executions are deterministic.As expected, the similarity scores are all 1.0, pointing out correctly that the runs are from identical programs. Hoauthorver, as the number of threads increase, the similarity scores quickly deteriorate.

A similarity score greater than 1" indicates strong possibility of plagiarism, while a score less than " indicates the opposite. Considering typical value of " is betauthoren 0.15 and 0.35, SCSSB and DYKIS will not claim plagiarism when the number of threads is 6, and startto claim the runs are from different programs when the number of threads becomes 8.

To the best of our knowledge, this is the first work that discusses the impact of thread scheduling on birthmark based software plagiarism detection, and proposes a solution to remedy the problem apply the var-gram algorithm in birthmark generation. As far as author know, this is the first time this algorithm is used for such purpose. Our experiments confirm its effectiveness.Implemented a set of tools collectively called TOB-PD (TOB based Plagiarism Detection tool) by integrating the principle of TOB with existing algorithms, including SCSSB, DYKIS and JB .

Experiments on 418 versions of 35 different multithreaded programs show that the new tools are highly effective in detecting plagiarism and are resilient to most state-of-the-art semantics-preserving obfuscation techniques implemented in tools such as SandMark ,DashO and UPX.

IV. CONCLUSION

In this paper, implement a simple method, as multithreaded software become increasingly more popular, current dynamic software plagiarism detection technology geared toward sequential programs are no longer sufficient.Existingapproaches are not only accurate in detecting plagiarism of multithreaded programs but also robust against most state-of-the-art semantics preserving obfuscation techniques. The new birth mark technique can be easy to implement andThe proposed work addresses the challenges of applying dynamic birthmark based approaches for whole program plagiarism detection of multithreaded software. As far as author know, this is the first work that discusses the impact of thread scheduling on birthmark based plagiarism detection, and the first work that propose thread-oblivious birthmarks for solving the problem systemically.

REFERENCES

[1] ZhenzhouTian , Ting Liu, Member, IEEE, QinghuaZheng,"Reviving Sequential Program Birthmarking for Multithreaded Software Plagiarism Detection," IEEE Trans. Softw.VOL. 44, NO. 5, pp.491-511 MAY 2018 .

[2] L. Luo, J. Ming, D. Wu, P. Liu, and S. Zhu, "Semantics-based obfuscation-resilient binary code similarity comparison with

applications to software and algorithm plagiarism detection," IEEE Trans. Softw. Eng., 2017

[3] Z. Tian, T. Liu, Q. Zheng, F. Tong, M. Fan,and Z. Yang, "A new thread-aware birthmark for plagiarism detection of multithreaded programs," in Proc. Int. Conf. Softw. Eng. Companion, pp. 734– 736, 2016.

.[4] Z. Tian, T. Liu, Q. Zheng, M. Fan, E. Zhuang, and Z. Yang, "Exploiting thread-related system calls for plagiarism detection of multithreaded programs," J. Syst. and Softw., vol. 119, pp. 136–148, 2016

[5] K. Chen, et al., "Finding unknown malice in 10 seconds: Mass vetting for new threats at the Google-Play scale," in Proc. USENIX Secur. Symp., , pp. 659–674,2015.

