Studying the Impact of Obfuscation on Source Code Plagiarism Detection

Deliverable 5: Context and bibliography

Albert Cabré Juan
20th July 2013

Index

1. Stakeholders 3
1.1 Developers 3
1.2 Project tutor 3
1.3 Plagiarism detectors developers 3
1.4 Users 3

2. State of the art 4
2.1 Plagiarism 4
2.2 Source code plagiarism 4
2.3 Source code plagiarism detection 5
2.4 Code obfuscation 5
2.4.1 Recreational purposes 6
2.4.2 Legit purposes 6
2.4.3 Unethical/Illegal purposes 6
2.5 Detecting obfuscated source code plagiarism 7

3. Bibliography 8

1. Stakeholders

This section aims to describe the context within which the project will be developed. Stakeholders are people or organizations who have an interest in your research project, or affect or are affected by its outcomes.

1.1 Developers

I am the one and only developer. My role is the carry out the project.

1.2 Project tutor

This project's tutor is Jens Krinke, his role is to supervise that the project fits the timetable and that the objectives are being accomplished. Additionally, he guides and helps the developer to carry out the project.

1.3 Plagiarism detectors developers

People or companies that developer plagiarism detectors will be very interested in whatever findings this project makes. Developing obfuscation resilient plagiarism detectors is one of the current unsolved goals for them and so any research in this field will be of high interest for them.

1.4 Users

This project's users are all the people that will benefit from the conclusions obtained from it. Because virtually everyone uses open source software, everyone could be considered an end user of this project.

2. State of the art

2.1 Plagiarism

Plagiarism is the "wrongful appropriation" and "purloining and publication" of another author's "language, thoughts, ideas, or expressions," and the representation of them as one's own original work [1].

Plagiarism doesn't always constitute a crime, but is always considered a serious ethical offense. However, copyright infringement is being done when the plagiarised work is under intellectual property.

Most of the world's jurisdictions treat copyright infringement under civil law, while others treat under criminal law [2]. What's important, however, is that copyright infringement is illegal in most of the world countries [3].

2.2 Source code plagiarism

Plagiarism is common in the computer science field, especially in education [4]. This fact explains why source code plagiarism wasn't considered a serious problem or a threat until recently [5]. However, the Oracle vs Google case changed the way we look at software plagiarism. Oracle filed a billion dollar lawsuit against Google, accusing the later of source code plagiarism. Although Google won the lawsuit and was declared innocent of all charges, some industry big companies like Microsoft still refuse to accept the verdict [6] and the case remains a matter of great controversy.

Source code plagiarism is still hard to define [7]. Because source code aims to solve a problem, different authors may come up with very similar or even identical solutions if the piece of code is small enough. For this reason, it's hard to define the line between what's coincidence and plagiarism.

2.3 Source code plagiarism detection

Source code plagiarism differs though from most other plagiarism forms and so requires completely different tools to detect it. Extensive research has been done around source code plagiarism detectors [8] and, to date, these are the types of plagiarism detectors techniques [9]:

- Text-based: These techniques try to find plagiarised code by searching for textual matches of text segments. Text-based plagiarism detectors are the fastest available but also the least powerful ones.
- Token-based: These techniques work by first doing some simple transformations of the code (removing whitespace, removing comments...) and then analyzing the result in a very similar way to what text-based techniques do.
- Tree-based: The goal here is to first build a parse tree for the source codes being analyzed and then analyzing and comparing the resulting trees. These techniques allow for higher level analysis, since simple constructs (conditional statements or loops) are revealed in the trees.
- PDG-based: Detectors that use PDG-based techniques work by building a Program Dependency Graph (this requires actually running the program) and then analyzing the resulting graph. Because dependencies graphs reveal all the inner structure and program flow of a source code, comparing these graphs allows for much better plagiarism detection.
- Metrics-based: These techniques aim to record specific metrics of the analyzed codes (number of function calls, number of if statements...) and then compare these metrics. The main problem with this approach is it's high sensitivity: two different programs might have perfectly equal metrics while being completely different. For this reason this techniques are usually used as a complement to other techniques.
- Hybrid: A combination of the previously mentioned techniques.

Two of the most used plagiarism detection engines are MOSS and JPlag. These two engines are, however, aimed at educational environments and so only implement the first techniques listed above. This is because in an educational environment codes that have the same behaviour are not only expected but actually required, so techniques that analyze the behaviour of the program (trees, pdgs and metrics) would result in a large number of false positives.

2.4 Code obfuscation

Code obfuscation is the act of refactoring a source code to produce an obfuscated version, that is, a version that is written differently while preserving the original code behaviour. This may be done for recreational, legit or illegal/unethical purposes.

2.4.1 Recreational purposes

One of the oldest purposes of code obfuscation is recreation. Because obfuscation is complex, properly obfuscating source code is considered an art by many. Obfuscation contests exist [10] and winning them is considered a great merit amongst the hacker culture.

2.4.2 Legit purposes

Many software companies ship their final products obfuscated to prevent people from tampering with it. This is especially true for paid software, where the parts of the software that deal with the licensing are obfuscated to prevent hackers from finding a way to crack it, that is, to use it without paying. Some development frameworks support and even encourage obfuscating the code before releasing it to the public. This is the case for the .NET framework [11], for example.

2.4.3 Unethical/Illegal purposes

2.4.3.1 Malware obfuscation

Malware has been the most benefited kind of software from obfuscation [12]. Obfuscating malware makes it harder for antivirus software to detect it and so black hat hackers have developed an extensive set of techniques to obfuscate code over the years. What's more, because antivirus software would quickly update it's databases to detected newly obfuscated version of the same malware, polymorphic malware was developed. That is, malware that rewrites itself by using automatic obfuscation techniques before replicating. It's then easy to understand why most of the automatic obfuscation techniques were developed in the early years of polymorphic malware. Note, however, that malware obfuscation is applied to machine code in most of the cases and so some of the techniques developed for this purpose are invalid for real source code.

2.4.3.2 Plagiarism

Because most of the plagiarism detection tools rely on analyzing the source code directly, obfuscation prevents most of this tools from being able to detect the plagiarised code. This is further explained in next section.

2.5 Detecting obfuscated source code plagiarism

Current source code plagiarism detection approaches are still premature. In fact, none of them is resilient to code obfuscation, and they all can be "defeated" by (in most cases rather simple) code-obfuscation-based counter-detection measures [5] [13]. Recent developments in code obfuscation have made such measures extremely easy and affordable to take, and indeed mature obfuscation tools have been freely available, making the situation even worse. Moreover, many existing schemes rely on analyzing the source code of a suspected software product, which often cannot be obtained until some strong evidences have been collected [13].

3. Bibliography

[1] Random House Compact Unabridged Dictionaryqtd. in Stepchyshyn, Vera; Robert S. Nelson (2007). Library plagiarism policies. Assoc of College & Resrch Libraries. p. 65. ISBN 0-8389-8416-9.

[2] Irina D. Manta Spring 2011 The Puzzle of Criminal Sanctions for Intellectual Property Infringement Harvard Journal of Law & Technology 24(2):469-518

[3] Correa, Carlos Maria; Li, Xuan (2009). Intellectual property enforcement: international perspectives. Edward Elgar Publishing. p. 211. ISBN 978-1-84844-663-2.

[4] M. Joy , G. Cosma , J. Y. Yau and J. Sinclair "Source code plagiarism—A student perspective", *IEEE Trans. Educ.*, vol. 54, no. 1, pp.125 -132 2011

[5] Student Projects Ideas Dr Jens Krinke, University College of London

[6] Brief for amici curiae Microsoft corporation, Emc corporation, and Netapp, inc. in support of appellant, Microsoft Corporation

[7] G. Cosma and M. Joy, “Towards a definition of source-code plagiarism,” IEEE Trans. Educ., vol. 51, pp. 195–200, May 2008.

[8] Plagiarism Prevention and Detection - On-line Resources on Source Code Plagiarism. Higher Education Academy, University of Ulster.

[9] Roy, Chanchal Kumar;Cordy, James R. (September 26, 2007). "A Survey on Software Clone Detection Research". School of Computing, Queen's University, Canada.

[10] The International Obfuscated C Code Contest. Leo Broukhis, Simon Cooper, Landon Curt Noll. The IOOCC.

[11] Techniques for Automating Obfuscation. .NET Library documentation. Microsoft Corporation.

[12] Obfuscation: Malware’s best friend. Jushua Cannell, Malwarebytes Corporation.

[13] SPLAD: Obfuscation Resilient Software Plagiarism Detection, Dinghao Wu, College of Information Sciences and Technology, Penn State University