L. Schmid, S. Hahner, and T. Sağlam, "JPlag: Detecting Obfuscated Software Plagiarism using Token Normalization Graphs", in Software Engineering (SE), GI, 2025, accepted, to appear.

JPlag: Detecting Obfuscated Software Plagiarism using Token Normalization Graphs

Larissa Schmid [1], Sebastian Hahner [1], and Timur Sağlam [1]

Abstract: While software plagiarism detectors have been used for decades, the assumption that evading detection requires programming proficiency is challenged by the emergence of automated plagiarism generators. These generators enable effortless obfuscation attacks, exploiting vulnerabilities in existing detectors by inserting statements to disrupt the matching. We present a language-independent defense mechanism that leverages program dependence graphs, rendering such attacks infeasible. We evaluate our approach with multiple real-world datasets and show that it defeats plagiarism generators by offering resilience against automated obfuscation while maintaining a low rate of false positives.

Keywords: Software Plagiarism Detection, Plagiarism Obfuscation, Obfuscation Attacks, JPlag

Overview

Plagiarism is widespread in computer science education [CJ08; Mur10], mainly due to the ease of copying digital assignments. Despite understanding it as misconduct, some students continue to engage in plagiarism, often attempting to obfuscate it by renaming, reordering, or inserting code [Kar16; NJK19; Sağ+22; Sağ+23b; Sağ+24b]. This issue is especially prominent in large, mandatory courses where manual inspection is impractical [Cam+17], making automated plagiarism detection essential [Ott76]. Software detectors like MOSS and JPlag are commonly used to address this problem, assuming that successful obfuscation requires skills already taught. However, plagiarism generators, such as MOSSAD [DB20], challenge this assumption by automating obfuscation without requiring expertise. MOSSAD breaks token-based detectors by inserting entropy or reordering statements to evade detection.

In response, we introduce a defense mechanism called *token sequence normalization* [Sağ+24a], which combines the scalability of token-based methods with the robustness of graph-based approaches. By leveraging program dependence graphs (PDGs), we normalize token sequences to counteract obfuscation attacks while ensuring the original, unaltered code remains traceable. We implement this mechanism within JPlag [PMP+02] and address the following research questions: Can our mechanism detect MOSSAD-style plagiarism? (**RQ1**) Can it resist insertion- and reordering-based obfuscation? (**RQ2**) How does it affect false positives? (**RQ3**) What is the performance impact? (**RQ4**)

1 Karlsruhe Institute of Technology (KIT), Institute of Information Security and Dependability, Germany,
larissa.schmid@kit.edu, https://orcid.org/0000-0002-3600-6899;
sebastian.hahner@kit.edu, https://orcid.org/0000-0003-3450-0508;
timur.saglam@kit.edu, https://orcid.org/0000-0001-5983-4032

In our evaluation using public and course datasets, our mechanism renders obfuscation attacks ineffective, increasing similarity scores to 95% while maintaining false positives under 1%. The overhead is minimal, making it practical for large-scale use. Our approach combines the scalability of token-based methods with the robustness of graph-based techniques, achieving over 98% similarity detection for individual obfuscation attacks and over 95% for combined ones. Future work will incorporate more semantic information and address emerging threats, such as AI-based attacks and those involving refactoring or minor behavior changes. The results of this work have already been incorporated in the open-source tool JPlag and are also publicly available in our data set [Sağ+23a].

References

[Cam+17] T. Camp et al. "Generation CS: The Growth of Computer Science". In: *ACM Inroads* (2017), pp. 44–50. DOI: 10.1145/3084362.

[CJ08] G. Cosma and M. Joy. "Towards a Definition of Source-Code Plagiarism". In: *IEEE Transactions on Education* (2008), pp. 195–200. DOI: 10.1109/te.2007.906776.

[DB20] B. Devore-McDonald and E. D. Berger. "Mossad: Defeating Software Plagiarism Detection". In: *PACMPL* (2020), pp. 1–28. DOI: 10.1145/3428206.

[Kar16] O. Karnalim. "Detecting source code plagiarism on introductory programming course assignments using a bytecode approach". In: *ICTS '16*. 2016, pp. 63–68. DOI: 10.1109/icts.2016.7910274.

[Mur10] W. Murray. "Cheating in Computer Science". In: *Ubiquity* (2010), p. 2. DOI: 10.1145/1865907.1865908.

[NJK19] M. Novak, M. Joy, and D. Kermek. "Source-Code Similarity Detection and Detection Tools Used in Academia: A Systematic Review". In: *TOCE* (2019), pp. 1–37. DOI: 10.1145/3313290.

[Ott76] K. J. Ottenstein. "An Algorithmic Approach to the Detection and Prevention of Plagiarism". In: *ACM SIGCSE Bulletin* (1976), pp. 30–41. DOI: 10.1145/382222.382462.

[PMP+02] L. Prechelt, G. Malpohl, M. Philippsen, et al. "Finding plagiarisms among a set of programs with JPlag." In: *Journal of Universal Computer Science* (2002), p. 1016.

[Sağ+22] T. Sağlam, S. Hahner, J. W. Wittler, and T. Kühn. "Token-Based Plagiarism Detection for Metamodels". In: *MODELS-C*. 2022, pp. 138–141. DOI: 10.1145/3550356.3556508.

[Sağ+23a] T. Sağlam et al. *Supplementary Material for Detecting Automatic Software Plagiarism via Token Sequence Normalization*. Zenodo, 2023. DOI: 10.5281/zenodo.10430321.

[Sağ+23b] T. Sağlam, L. Schmid, S. Hahner, and E. Burger. "How Students Plagiarize Modeling Assignments". In: *MODELS-C*. 2023, pp. 98–101. DOI: 10.1109/MODELS-C59198.2023.00032.

[Sağ+24a] T. Sağlam, M. Brödel, L. Schmid, and S. Hahner. "Detecting Automatic Software Plagiarism via Token Sequence Normalization". In: *IEEE/ACM 46th International Conference on Software Engineering*. 2024, pp. 1–13. DOI: 10.1145/3597503.3639192.

[Sağ+24b] T. Sağlam, S. Hahner, L. Schmid, and E. Burger. "Automated Detection of AI-Obfuscated Plagiarism in Modeling Assignments". In: *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*. 2024. DOI: 10.1145/3639474.3640084.