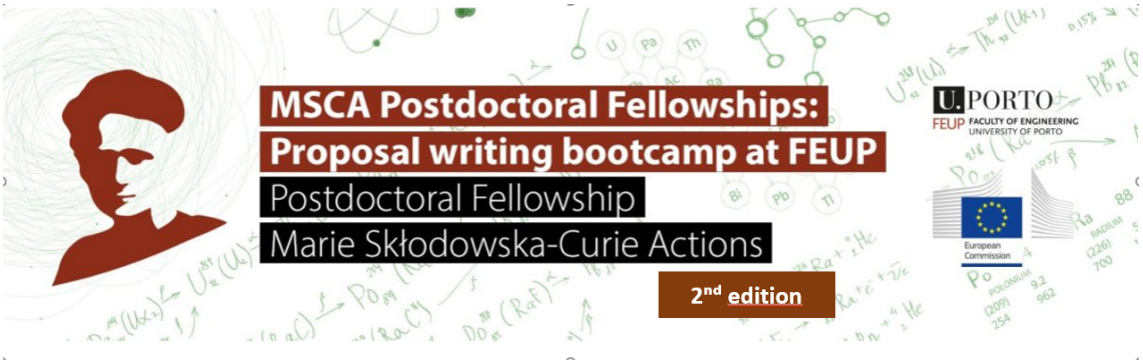
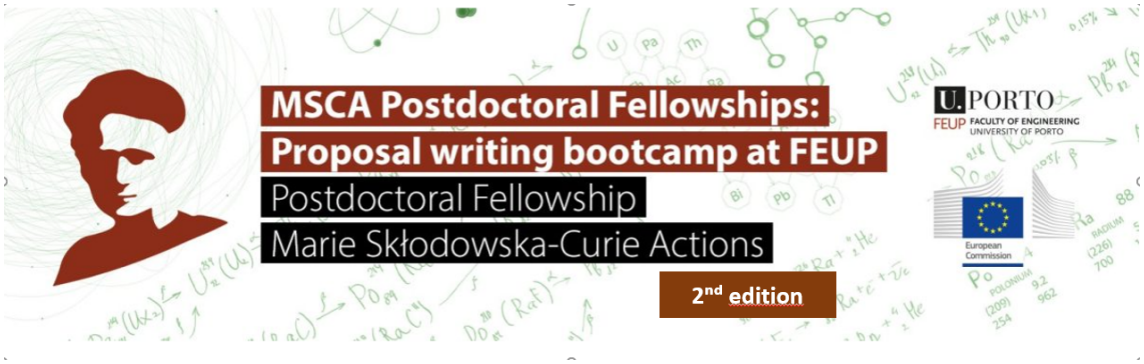


SUPERVISOR INFORMATION	
First and Last name	José Campos
URL of supervisor webpage	https://jose.github.io
Department	Department of Informatics Engineering
Field(s) of research	Software Security, Software Vulnerabilities, Empirical Software Engineering, Mining Software Repositories
PROJECT PROPOSAL	
Title (optional)	Inter-Procedural Vulnerabilities

Brief project description

In software security, organizations face significant challenges due to the increasing complexity and volume of vulnerabilities, mostly stemming from developers' lack of expertise in effectively addressing these issues. As an attempt to detect, analyze (e.g., the root cause and the impact), and repair software vulnerabilities, researchers have proposed several data-driven approaches (e.g., machine-learning-based and deep-learning- based approaches). Although such approaches achieve convincing performance in the laboratory environment, their performances drop dramatically in the real-world scenario [1]. A prior work [2] observed 20% to 71% of vulnerability labels are inaccurate and 17% to 99% of vulnerability data were duplicated in four state-of-the-art software vulnerability datasets [3, 4, 5, 6]. The security vendors, such as National Vulnerability Database (NVD) [7] and Snyk [8], are important sources for collecting real-world vulnerability data. It is common to extract the patch information from the disclosed vulnerability records (i.e., Common Vulnerabilities and Exposures, CVEs) to trace and identify real-world vulnerabilities and the corresponding fixes. However, it is challenging to improve the data quality.

When constructing a vulnerability dataset, the current practice [5] simply considers the original functions vulnerability dataset as the vulnerable functions. However, for an inter-procedural vulnerability---a vulnerability that have vulnerable code snippets scattered among functions or files---, a single vulnerable code snippet in one function is not necessarily meant to be vulnerable by itself. Moreover, the granularity of the current vulnerability datasets is usually at the function level and the function-level information is not enough for learning the pattern of inter-procedure vulnerabilities, which may introduce bias in model training. D2A [5] provided the bug trace information which can be used for inter-procedural vulnerabilities detection. However, D2A is constructed using the static tool, suffering from the inaccurate labeling problem [2] and the limitation of limited vulnerability types.



Given most vulnerabilities are inter-procedural [5, 9], it is crucial to provide a dataset that considers inter-procedural vulnerabilities and provides information that is more than a single function. That is, this project proposal aims to investigate and develop novel methodologies and approaches to first detect and then describe inter-procedural vulnerabilities.

[1] S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, "Deep learning based vulnerability detection: Are we there yet," IEEE Transactions on Software Engineering (TSE), 2021.

[2] R. Croft, M. A. Babar, and M. M. Kholoosi, "Data quality for software vulnerability datasets," in Proceedings of the 45th International Conference on Software Engineering (ICSE). IEEE, 2023, pp. 121–133.

[3] J. Fan, Y. Li, S. Wang, and T. N. Nguyen, "A c/c++ code vulnerability dataset with code changes and cve summaries," in Proceedings of the 17th International Conference on Mining Software Repositories (MSR), 2020, pp. 508–512.

[4] Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu, "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks," Advances in neural information processing systems (NIPS), vol. 32, 2019.

[5] Y. Zheng, S. Pujar, B. Lewis, L. Buratti, E. Epstein, B. Yang, J. Laredo, A. Morari, and Z. Su, "D2a: A dataset built for ai-based vulnerability detection methods using differential analysis," in Proceedings of 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 2021, pp. 111–120.

[6] T. Boland and P. E. Black, "Juliet 1.1 c/c++ and java test suite," IEEE Computer Architecture Letters, vol. 45, no. 10, pp. 88–90, 2012.

[7] "National vulnerability database." [Online]. Available: https://nvd.nist.gov/

[8] "Snyk." [Online]. Available: https://security.snyk.io/vuln

[9] W. Zheng, Y. Jiang, and X. Su, "Vu1spg: Vulnerability detection based on slice property graph representation learning," in Proceedings of the 32nd International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2021, pp. 457–467.