


Predicting Security-Vulnerable Developers Based on Their Techno-Behavioral Characteristics

M. D. J. S. Goonetillake, School of Computing, University of Colombo, Sri Lanka

 <https://orcid.org/0000-0002-3864-3699>

Rangana Jayashanka, School of Computing, University of Colombo, Sri Lanka

S. V. Rathnayaka, School of Computing, University of Colombo, Sri Lanka

ABSTRACT

Assigning developers for highly secured software projects requires identifying developers' tendency to contribute towards vulnerable software codes called developer-centric security vulnerability to mitigate issues on human resource management, financial, and project timelines. There are problems in assessing the previous codebases in evaluating the developer-centric security vulnerability level of each developer. Thus, this paper suggests a method to evaluate this through the techno-behavioral features of their previous projects. Consequently, the authors present results of an exploratory study of the developer-centric security vulnerability level prediction using a dataset of 1,827 developers by logically selecting 13 techno-behavioral features. The results depict that there is a correlation between techno-behavioral features and developer-centric security vulnerability with 89.46% accuracy. This model enables to predict developer-centric security vulnerability level of any developer if the required techno-behavioral features are available, avoiding the analysis of his/her previous codebases.

KEYWORDS

Decision Tree, Developer-Centric Security Vulnerability, Logic Regression, Machine, Naïve Bayes, Random Forest, Support Vector Vector Machine, Techno-Behavioral Features

INTRODUCTION

Computer software should satisfy two types of requirements in the application domain namely functional and non-functional requirements. Both are equally important to be satisfied regardless of their operational industry domain. Moreover, software quality has been described by many characteristic aspects. There are several metrics for software quality that can be used to evaluate the qualities of software such as scalability, security, reliability, and usability (Gorton, 2011). Among all these quality metrics, software security has been described as one of the most significant quality attribute (Stephenson et al., 1992) since, a security vulnerability can be a cause of a huge disaster which can lose billion dollars of assets (Willetts, 2014) to an organization or even lives (Csulak et

DOI: 10.4018/IJISP.2022010103

al., 2017). Generally, functional issues in a software which can be considered as causing *defects* in software are possible to be tested and validated by executing test scenarios of the business logic. However, non-functional issues which can be considered as causing vulnerability in the software are difficult to identify since it may not get exposed by the execution of predetermined test scenarios (Zimmermann et al., 2010; Krsul, 1998). It should be noted that the term '*software vulnerability*' is mainly referred to in the computing domain concerning a security flaw, glitch, or weakness found in software or in an operating system (OS) that can lead to security concerns.

Although many Computer-Aided Software Engineering (CASE) tools are available, humans still dominate as the core contributors of the software development process. To this end, it is an inevitably applicable scenario that software is vulnerable to functional and non-functional defects due to human mistakes. Software defects and vulnerabilities have many similarities since both are incurred due to human mistakes. However, vulnerabilities differ from defects since they are actively observed by the attackers with malicious and criminal intent while defects are exposed through the valid use cases of its normal usage (Krsul, 1998). The vulnerability of a software application could occur at any stage of the software development life cycle and may be introduced due to various reasons such as invalid requirement specification, weak architectural designs, weak and vulnerable implementation techniques, and algorithms and weak test scenarios executed. In this study, the focus is scoped on vulnerabilities that the developer has caused or contributed to source code of the software. Each software developer has a unique skill level, experience, capacity, technology interests, domain interests, and many other characteristics which can affect the overall quality of the software positively or negatively that he/she develops.

In large-scale software development projects, it is important, but a complex task to ensure that all developers are skilled and experienced enough to contribute and collaborate towards the success of the project avoiding any security vulnerabilities in software throughout the development lifecycle. Thus, it will be a challenging task to evaluate and identify a developer's tendency to contribute towards vulnerable software codes which are for convenience termed as a ***developer-centric security vulnerability*** in this paper from this point onwards. Prior knowledge of developer-centric security vulnerability is significant when there is a need for selecting developer teams for highly secured mission-critical software development projects. To this end, it would be better if developer-centric security vulnerability could be identified in advance and taken as a parameter into the developer selection criteria at the very initial stage of the software project. This is because developers with less or no vulnerability prediction may minimize the risk of causing security vulnerabilities in the final software product. Otherwise, it will be a time-consuming and tedious task to investigate the accumulating codebase of an ongoing software project for security vulnerability. Moreover, this investigation process must be done on each developer basis identifying individual code fragments in the codebase along with their vulnerabilities. If the developer code contribution has not been up to the security standards or practices it may cause practical issues in human resource management leading to substitutions or swaps in developer teams and their responsibilities. This is not a feasible task in the context of certain projects such as highly secured mission-critical software development projects with strict project timelines and many other technical and non-technical constraints and may also add a significant amount of extra cost/overhead to the software project. It should also be noted that detecting the security vulnerability of ongoing software projects requires special tools to have access to the codebase of the project and is more difficult than identifying a software defect that may occur due to a functional issue (Zimmermann et al., 2010; Krsul, 1998). Adding to this complexity, the accessibility of the codebase of highly secured mission-critical projects may very well subject to restrictions. Hence, it emphasizes the requirement to have a mechanism to identify the potential developer-centric vulnerability in advance thus mitigating the security vulnerability of ongoing software projects.

A probable scenario is to analyze the quality features of the previously developed software codebase to find out the responsible developers who have contributed security-wise weak code

24 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/article/predicting-security-vulnerable-developers-based-on-their-techno-behavioral-characteristics/284048

Related Content

Three New Directions for Time Banking Research: Information Management, Knowledge Management, and the Open Source Model

Lukas Valek (2018). *Multidisciplinary Perspectives on Human Capital and Information Technology Professionals* (pp. 324-340).

www.irma-international.org/chapter/three-new-directions-for-time-banking-research/198264

Association Rule Hiding in Privacy Preserving Data Mining

S. Vijayarani Mohanand Tamarasi Angamuthu (2018). *International Journal of Information Security and Privacy* (pp. 141-163).

www.irma-international.org/article/association-rule-hiding-in-privacy-preserving-data-mining/208130

Integrating Circular Economy Concerns Into the Industry 4.0 Roadmaps of Companies: A Literature Review

Sekkat Souhail, Ibtiham El Hassani and Anass Cherrafi (2024). *Enhancing Performance, Efficiency, and Security Through Complex Systems Control* (pp. 64-77).

www.irma-international.org/chapter/integrating-circular-economy-concerns-into-the-industry-40-roadmaps-of-companies/337452

Investigation of Credit Risk based on Indian Firm Performance

Manoj Kumar (2017). *International Journal of Risk and Contingency Management* (pp. 35-46).

www.irma-international.org/article/investigation-of-credit-risk-based-on-indian-firm-performance/177839

CFS-MHA: A Two-Stage Network Intrusion Detection Framework

Ritinder Kaur and Neha Gupta (2022). *International Journal of Information Security and Privacy* (pp. 1-27).

www.irma-international.org/article/cfs-mha/313663