Chapter 1
Unmasking Threat Vectors in Large Language Models:
A Deep Analysis of Adversarial Exploitation

Neeharika Joshi
Marwadi University, Rajkot, India

Sonukumar Pandit
Marwadi University, Rajkot, India

R. N. Ravikumar
http://orcid.org/0009-0009-3705-1681
Marwadi University, Rajkot, India

Shubhamkumar Pandit
Marwadi University, Rajkot, India

Satwik Kishore
http://orcid.org/0009-0002-2014-7897
Marwadi University, Rajkot, India

S. Aarthi
http://orcid.org/0009-0006-9064-2091
Marwadi University, Rajkot, India

ABSTRACT

AI-driven defect detection is increasingly essential as software systems grow in complexity, multilinguality, and scale. This chapter presents a comparative evaluation of traditional machine learning models and large language models for detecting structural and semantic defects across multiple programming languages. A multilingual dataset is used to assess Random Forest, SVM, XGBoost, and a prompt-based CodeT5 simulation, revealing the limitations of feature-engineered approaches and the superior semantic reasoning of LLMs. Practical integration scenarios in CI/CD, IDEs, and DevSecOps workflows are examined, followed by key challenges and emerging research directions for scalable, explainable, and automated defect detection.

DOI: 10.4018/979-8-3373-8252-4.ch001

INTRODUCTION

As the modern software systems grow in size and scale with increased level of sophistication, the eminent and effective defect discovery becomes all the more vital. The software may contain bugs small logical errors but also security holes so severe that a system crash, a data loss, or expensive payoffs are the consequences. Traditional software Quality Assurance (QA) processes such as manualized code reviews, static code analysis and test based on rules may be helpful in spite of being incapable of usually covering all errors in the large-scale and multi-lingual code of applications. These methods are usually limited; they possess set rules of notions, are limited by scaled issues, and do not cover meaning-dense mapping of any codes. On the other hand, AI (Artificial Intelligence) and its sub-categories; Machine Learning (ML) and Large Language Models (LLM) has shown a broad possibility of revolutionizing the processes of identifying software flaws and their interpretation (Chhabra & Chadha, 2024). Learning models supervised are a category of ML meta-techniques which it is possible to teach on past defect data to find out the strategy and tendencies that draw attention to buggy codes. Armed with well-engineered parameters such as code complexity measures, control flow information or Abstract Syntax Tree (AST) representations, such models have the ability to generalize well over data they did not previously see and automatically code data as either weighed down with flaws or pristine.

Machine Learning models also have their weaknesses, especially in comprehending code semantics as well as logical reasoning across programming paradigms. LLMs come in here since they are pretrained with huge amounts of code written by other programmers in open-source repositories. Such models as CodeT5, CodeBERT, and GPT-4 are capable of working with code in a form that closely reflects its comprehension by humans reading it, reading between the lines and even predicting the purpose of that code. Using prompt-based learning techniques, LLMs are able to detect bugs, propose solutions, and clarify how and why codes work, in a context and language-freeway. Combined use of ML and the approaches based on LLMs is a strong potential to develop stable, scalable, and smart defect detection systems. The typical multilanguage nature of business logic, where the same teams may write parts in Python, Java, C++, JavaScript, and Go, requires models that can cut across languages. A static analyzer is likely to not succeed in these cases because of the language-specific limitations, whereas the LLMs provide generalization and any ML model can be trained on shared structural features in all languages.

The purpose of this chapter is to fill the gap between the theoretical and practical face of the AI-based software defect detection. It presents a practical case study wherein it constructs an end-to-end multilingual defect prediction model comprising a simulated LLM model as well as an ML-based classifier. The ML model uses a