Fousseynou DIAKITE L0

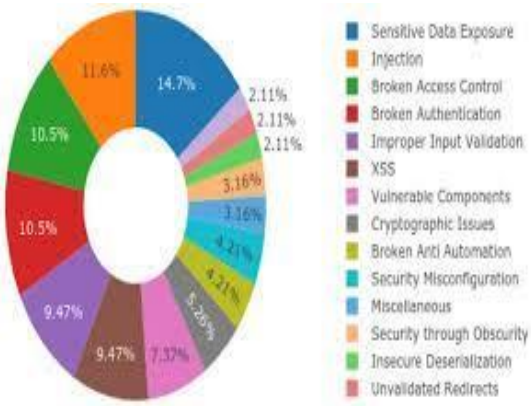
MASTER 2 MCI BRESIL

BIG DATA ANALYTIC: Data modeling and source code generation and ai

Article 3: CodeLMSec Benchmark: Systematically Evaluating and Finding Security Vulnerabilities in Black-Box Code Language Models.

Large language models represent a major advancement in current deep learning developments. With increasing size, their learning capacity allows them to be applied to a wide range of tasks, such as text translation [1], [2] and summarization [3], [2], chatbots like ChatGPT [4], and also for code generation and code understanding tasks [5], [6], [7], [8]. A prominent example is GitHub Copilot [9], an AI pair programmer based on OpenAI Codex [5], [10] that is already used by more than a million developers [11]. ChatGPT [4], Codex [5] and open models such as Code Llama [12], CodeGen [6] and InCoder [7] are trained on a large-scale corpus of natural language and code data and enable powerful and effortless code generation.

Security Vulnerability Issues of Code Generation Models Large language code generation models have been pre-trained using vast corpora of open-source code data [7], [5], [25]. These open-source codes can contain a variety of different security vulnerability issues, including memory safety violations [26], deprecated API and algorithms (e.g., MD5 hash algorithm [27], [15]), or SQL injection and cross-site scripting [28], [15] vulnerabilities.



There have been tremendous advances in large-scale language models for code generation, and state-of-the-art models are now used by millions of programmers every day. Unfortunately, we do not yet fully understand the shortcomings and limitations of such models, especially with respect to insecure code generated by different models. Most importantly, we have lacked a method for systematically identifying prompts that lead to code with security vulnerabilities. In this paper, we have presented an automated approach to address this challenge. We approximated the black-box inversion of the target models based on few-shot prompting, which allows us to automatically find different sets of targeted vulnerabilities of the black-box code generation models. We proposed three different few-shot prompting strategies and used static analysis methods to check the generated code for potential security vulnerabilities.