УДК 004.89

ИНТЕЛЛЕКТУАЛЬНОЕ ОБНАРУЖЕНИЕ ОШИБОК: ОБЗОР ИНТЕГРАЦИИ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ В СТАТИЧЕСКИЙ АНАЛИЗ КОДА

Юров М.А.[1]
Научный руководитель – кандидат технических наук, доцент Дергачев А.М.[1]
[1]Университет ИТМО

Введение

Обеспечение надежности программного обеспечения осложняется ростом его масштабов и зависимостью от сторонних компонентов. Традиционные методы статического анализа, основанные на формальных правилах, сохраняют свою значимость, однако их эффективность снижается при работе с динамически изменяющимся кодом и множественными зависимостями. Их ключевыми недостатками остаются высокий уровень ложных обнаружений уязвимостей и необходимость трудоемкого ручного описания спецификаций для внешних библиотек и фреймворков. В то же время большие языковые модели (LLM), обладающие способностью к семантическому пониманию кода и выведению намерений разработчика, предоставляют возможности для автоматизации и уточнения результатов анализа [1]. Цель работы – повышение эффективности выявления ложных обнаружений уязвимостей методом интеграции LLM в процессы статического анализа.

Основная часть

Определены и проанализированы три ключевых архитектурных подхода к интеграции LLM:

1. Применение LLM в фреймворках для вывода спецификаций и декомпозиции задач после статического анализа и выполнения структурной проверки детерминированными алгоритмами. В частности подход, применяемый в фреймворке IRIS, динамически определяет роли API, что позволяет значительно расширить покрытие анализа, распространяя его на сторонние библиотеки [2].
2. Генеративный синтез правил, реализуемый в инструментах, таких как KNighter, используют LLM для автоматического написания правил анализа на QL или C++ на основе истории изменений программного кода. Это решает проблему масштабируемости и позволяет обнаруживать ошибки в крупных проектах, таких как ядро Linux [3].
3. Интеллектуальная сортировка. Системы, подобные ZeroFalse, применяют LLM для постобработки результатов статического анализа, используя обогащенный контекст для исключения ложно обнаруженных уязвимостей из списка передаваемых конечному пользователю [4].

Выводы

Внедрение LLM в процессы статического анализа позволяет повысить эффективность выявления ложных обнаружений уязвимостей. Так применение гибридного подхода при анализе кода на примере набора данных OWASP Java Benchmark показало точность выявления ложных обнаружений уязвимостей на уровне 95% [4]. Также можно отметить, что применение LLM для вывода спецификаций способно повысить качество обнаружения уязвимостей. В частности, на наборе данных CWE-Bench-Java, средняя полнота увеличивается с 22,50% до 45,83%, а метрика F1-score возрастает с 0,076 до 0,177 по сравнению с базовыми инструментами статического анализа [2].

Литература

1. Jelodar H., Meymani M., Razavi-Far R. Large Language Models (LLMs) for Source Code Analysis: applications, models and datasets // arXiv preprint arXiv:2503.17502. – 2025.
2. Li Z., Dutta S., Naik M. IRIS: LLM-Assisted Static Analysis for Detecting Security Vulnerabilities // arXiv preprint arXiv:2405.17238. – 2024.
3. Yang C. et al. KNighter: Transforming Static Analysis with LLM-Synthesized Checkers // arXiv preprint arXiv:2503.09002. – 2025.
4. Iranmanesh M. et al. ZeroFalse: Improving Precision in Static Analysis with LLMs // arXiv preprint arXiv:2510.02534. – 2025.