

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ОБРАБОТКИ РЕЗУЛЬТАТОВ SAST-АНАЛИЗАТОРОВ

В.А. Рящиков¹, И.С. Поздняк¹

*¹Поволжский государственный университет телекоммуникации
и информатики, Самара
e-mail: ryashickov.vova@yandex.ru*

Аннотация

В работе представлена система верификации результатов статического анализа кода (SAST) с применением локальных больших языковых моделей (LLM). Для снижения числа ложных срабатываний предложен конвейер: исходный код проходит SAST-анализатор, затем обрабатывается модулем построения абстрактного синтаксического дерева (AST). Обогащенные данные передаются в LLM для вынесения итогового вердикта. Подход автоматизирует триаж уязвимостей и повышает эффективность процессов безопасной разработки.

Ключевые слова: SAST, LLM, информационная безопасность, анализ кода, триаж уязвимостей, автоматическое исправление.

Введение

Важнейшим вопросом обеспечения безопасности программного обеспечения на ранних этапах жизненного цикла разработки является своевременное выявление уязвимостей в исходном коде. Классическим инструментом для решения этой задачи выступает статический анализ кода (SAST). Однако применение традиционных SAST-анализаторов сопряжено с критической проблемой - генерацией большого количества ложных срабатываний (False Positives) [1]. Это связано с тем, что такие инструменты опираются на жесткие правила и регулярные выражения, не всегда учитывая сложный контекст выполнения программы. Необходимость ручного разбора отчетов (триажа) требует значительных временных затрат квалифицированных специалистов по информационной безопасности. Для автоматизации данного процесса и повышения точности анализа предлагается использовать механизмы интеллектуальной верификации на базе больших языковых моделей (LLM). Внедрение технологий машинного обучения в процессы статического анализа представляет собой переход к современным методам обеспечения безопасности кода [2].

1. Архитектура и принцип работы системы

Предлагаемый комплекс реализует гибридный подход к анализу кода, объединяя результаты проверки собственного исходного кода и сторонних компонентов в едином интерфейсе.

Базовый процесс обработки данных состоит из следующих этапов:

1. Исходный код поступает на вход классического SAST-анализатора, который формирует первичный отчет о потенциальных дефектах безопасности.

2. Код дополнительно обрабатывается специализированным модулем построения абстрактного синтаксического дерева (AST) связи с тем, что статические анализаторы зачастую не строят полный граф потока данных.

3. Обогащенные данные, включающие отчет SAST и контекст из AST, передаются на вход дообученной локальной большой языковой модели.

LLM проводит интеллектуальную верификацию: анализирует контекст, проверяет достижимость уязвимого участка и выносит итоговый вердикт о наличии реальной угрозы.

Пример архитектуры конвейера обработки данных представлен на рисунке 1.



Рисунок 1 – Архитектура интеллектуальной системы верификации уязвимостей

2. Интеллектуальная верификация и фильтрация ложных срабатываний

Применение локальных LLM позволяет добавить дополнительные факторы в процессы верификации уязвимостей. В процессе работы модель выступает в качестве экспертной системы, которая оценивает наличие механизмов очистки пользовательского ввода и сопоставляет найденный дефект с реальными векторами атак. Непрерывный анализ контекста с помощью нейросети позволяет динамически отсеивать информационный шум. Данный подход решает проблему контроля ложных срабатываний путем автоматической классификации инцидентов, оставляя для ручного анализа только подтвержденные угрозы. Использование именно локальных LLM в закрытом контуре обеспечивает целостность и конфиденциальность процесса верификации, предотвращая утечку исходного кода. Модуль, обеспечивающий подобный анализ, можно встраивать в другие экспертные системы [3].

3. Экспериментальное исследование и оценка эффективности.

Для подтверждения теоретических выводов и оценки эффективности предложенного подхода было проведено экспериментальное исследование работы модуля интеллектуальной верификации. В качестве базовой модели использовалась локальная LLM, дообученная на размеченном датасете, содержащем примеры реальных уязвимостей (True Positives) и ложных срабатываний (False Positives), сгенерированных классическими SAST-инструментами.

Тестовая выборка составила 3201 инцидент безопасности, из которых 2521 являлись подтвержденными уязвимостями (класс True), а 680 – ложными срабатываниями (класс False).

Анализ полученных метрик показывает высокую эффективность предложенного подхода:

1. Точность выявления ложных срабатываний: Модель успешно выявляет ложные срабатывания (класс False) с полнотой (Recall) 0.76. Это означает, что 76% всего информационного шума отсеивается автоматически, что подтверждает заявленное в теоретической части снижение нагрузки на специалистов.

2. Точность подтверждения уязвимостей: Для реальных уязвимостей (класс True) метрика Precision составила 0.93, а Recall - 0.88. Высокий показатель F1-score (0.91) для данного класса гарантирует минимальный риск пропуска критических дефектов (False Negatives).

3. Общая эффективность: Взвешенная точность (Weighted Avg F1-score) составила 0.86. Дополнительно зафиксированы высокие показатели площади под кривой ошибок: ROC-AUC = 0.914 и PR-AUC = 0.975, что свидетельствует об отличной разделяющей способности модели.

Оценка ожидаемого влияния внедрения системы на процессы безопасной разработки представлена в таблице 1.

Таблица 1 – Оценка эффективности

Критерий оценки	Описание результата
Точность анализа	Снижение уровня ложных срабатываний (False Positive Rate) на 80-90% за счет глубокого контекстного анализа нейросетью.
Временные затраты	Сокращение времени на обработку одного отчета безопасности с нескольких часов до 15-30 минут за счет автоматизации рутинного триажа.
Конфиденциальность	Обеспечение безопасности данных путем развертывания решения в закрытом контуре с использованием локальных LLM.
Масштабируемость	Возможность интеграции решения в существующие CI/CD конвейеры без существенной перестройки процессов разработки.

Как видно из результатов эксперимента и таблицы 1, внедрение предложенной архитектуры оказывает комплексное положительное влияние на процесс разработки. Снижение доли ложных срабатываний напрямую коррелирует с уменьшением временных затрат специалистов, что позволяет сфокусировать ресурсы на устранении действительно критичных уязвимостей. При этом сохраняется высокий уровень конфиденциальности анализируемых данных.

Заключение

Применение больших языковых моделей в системах статического анализа кода позволяет организовать автоматизированный процесс верификации уязвимостей, обеспечивает глубокий анализ контекста выполнения программы и реализует эффективную фильтрацию ложных срабатываний. Однако эффективность данного подхода во многом зависит от качества построения абстрактного синтаксического дерева и полноты обучающей выборки для LLM. В дальнейшем планируется разработка доказательства концепции на базе open-source SAST-инструментов. Кроме того, предполагается развитие функционала системы в части внедрения механизмов автоматического исправления уязвимостей. Использование моделей машинного обучения позволит генерировать синтаксически верные патчи и предлагать безопасные

версии библиотек, что обеспечит полный цикл автоматизированного реагирования на инциденты безопасности в исходном коде.

Список использованных источников

1. Информационная безопасность: анализ и оценка угроз, кибер/криптозащита организаций, разработка безопасного ПО / А. В. Кузнецов [и др.] // Москва: Лань, 2023. – 284 с.
2. Казарин, О. В. Надежность и безопасность программного обеспечения / О. В. Казарин, И. Б. Шубинский. – Москва: Издательство Юрайт, 2025. – 352 с.
3. Филиппов, Н. В. Подход к созданию экспертной системы оценки информационной безопасности телекоммуникационных систем / Н. В. Филиппов, Н. В. Киреева, И. С. Поздняк // Электросвязь. – 2022. – №2. – С. 61–66.

INTELLIGENT SYSTEM FOR PROCESSING RESULTS OF SAST ANALYZERS

V.A. Ryashickov ¹, I.S. Pozdnyak ¹

*¹Volga Region State University of Telecommunications and Informatics, Samara
e-mail: ryashickov.vova@yandex.ru*

Abstract

The paper presents a verification system for static application security testing (SAST) results using local large language models (LLMs). To reduce false positives, a pipeline is proposed: source code passes through a SAST analyzer and an abstract syntax tree (AST) module. The enriched data is fed to the LLM to make a final verdict. This approach automates vulnerability triage and improves the efficiency of secure development processes.

Keywords: SAST, LLM, information security, code analysis, vulnerability triage, false positives.