Учитывают ли DoLLM безопасность? Эмпирическое исследование ответов на вопросы по программированию

Дата: 2025-02-19 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2502.14202

Рейтинг: 75
Адаптивность: 85

Ключевые выводы:

Исследование оценивает способность крупных языковых моделей (LLM) выявлять уязвимости безопасности в коде и предупреждать пользователей о них. Основной вывод: LLM редко предупреждают о проблемах безопасности без явного запроса, обнаруживая только 12.6-40% уязвимостей в тестовых наборах данных.

Объяснение метода:

Исследование предлагает простые, но эффективные методы повышения безопасности ответов LLM путем добавления короткой фразы "Address security vulnerabilities" в запрос. Выявленные ограничения LLM в проактивном обнаружении уязвимостей критически важны для всех пользователей. Особенно ценно понимание качества информации о безопасности и конкретные методы улучшения ответов.

Ключевые аспекты исследования: 1. **Оценка проактивного выявления уязвимостей**: Исследование анализирует способность популярных LLM (GPT-4, Claude 3, Llama 3) распознавать уязвимости в коде и предупреждать разработчиков, даже когда пользователь не запрашивает проверку безопасности.

Качество информирования о безопасности: Авторы оценивают, насколько полно LLM объясняют причины уязвимостей, возможные эксплойты и способы устранения проблем, когда модели действительно выявляют проблемы безопасности.

Эмпирическое тестирование на реальных вопросах: Исследование использует 300 вопросов со Stack Overflow с уязвимым кодом, разделенных на два набора: вопросы с явными упоминаниями уязвимостей в ответах (Mentions dataset) и вопросы с трансформированным кодом без упоминаний уязвимостей (Transformed dataset).

Методы улучшения безопасности ответов: Авторы предлагают две стратегии: простое дополнение запроса фразой "Address security vulnerabilities" и встраивание

системы статического анализа CodeQL для предварительной проверки кода и включения результатов в запрос.

Прототип CLI-инструмента: Разработан инструмент, интегрирующий CodeQL с LLM, который значительно улучшает безопасность ответов, автоматически выявляя уязвимости и включая их в промпт.

Дополнение: Исследование не требует дообучения или API для применения основных методов. Ключевые подходы можно использовать в стандартном чате с LLM:

Явные запросы о безопасности: Простое добавление фразы "Address security vulnerabilities" к запросу значительно повышает вероятность обнаружения проблем безопасности (с 0% до 76% в лучшем случае для GPT-4). Это не требует никаких технических знаний и может применяться в любом чате.

Структурированные запросы о безопасности: Можно попросить LLM оценить код с точки зрения конкретных типов уязвимостей, которые модели выявляют лучше всего (например, утечка конфиденциальной информации, жестко закодированные учетные данные, XSS).

Критическая оценка ответов: Исследование показывает, что когда LLM действительно обнаруживают проблемы безопасности, они обычно предоставляют информацию о причинах, эксплойтах и исправлениях. Отсутствие этих компонентов может указывать на неполный анализ.

Итеративный подход: Если LLM не выявляет проблем безопасности с первого раза, можно попросить модель пересмотреть код с акцентом на конкретные аспекты безопасности.

Хотя интеграция с CodeQL требует дополнительных инструментов, концепция предварительной проверки кода может быть адаптирована путем использования онлайн-сервисов анализа кода и включения их результатов в запрос к LLM.

Применение этих подходов позволит значительно повысить безопасность кода, даже при использовании стандартного чата с LLM, без необходимости в дополнительных API или дообучении моделей.

Анализ практической применимости: 1. **Оценка проактивного выявления уязвимостей** - Прямая применимость: Низкая. Исследование показывает, что LLM редко предупреждают о проблемах безопасности без явной просьбы (до 40% в лучшем случае, 12-16% для трансформированного кода). - Концептуальная ценность: Высокая. Пользователи должны знать, что нельзя полагаться на LLM для автоматического выявления проблем безопасности в коде. - Потенциал для адаптации: Средний. Понимание ограничений LLM в обнаружении уязвимостей помогает пользователям формировать более безопасные стратегии использования этих инструментов.

Качество информирования о безопасности Прямая применимость: Высокая. Когда LLM действительно выявляют проблемы безопасности, они обычно предоставляют более полную информацию о причинах, эксплойтах и исправлениях, чем ответы на Stack Overflow. Концептуальная ценность: Высокая. Пользователи могут оценить качество информации о безопасности и научиться распознавать полные и неполные объяснения. Потенциал для адаптации: Высокий. Формат и структура безопасных объяснений могут использоваться в качестве шаблона при запросе LLM о безопасности кода.

Эмпирическое тестирование на реальных вопросах

Прямая применимость: Средняя. Понимание различий в поведении LLM на знакомых и новых данных помогает пользователям лучше оценивать надежность ответов. Концептуальная ценность: Высокая. Показывает важность критического мышления при работе с ответами LLM о коде. Потенциал для адаптации: Средний. Методология тестирования может быть адаптирована для оценки других аспектов работы LLM.

Методы улучшения безопасности ответов

Прямая применимость: Очень высокая. Простое добавление фразы "Address security vulnerabilities" к запросу значительно увеличивает количество предупреждений о безопасности (с 0% до 76% в лучшем случае). Концептуальная ценность: Высокая. Демонстрирует важность явных инструкций в промптах для получения более безопасных ответов. Потенциал для адаптации: Очень высокий. Эту простую технику могут сразу применять все пользователи LLM, независимо от технических навыков.

Прототип CLI-инструмента

Прямая применимость: Средняя. Требует установки дополнительного ПО и базовых технических знаний. Концептуальная ценность: Высокая. Показывает эффективность интеграции инструментов статического анализа с LLM. Потенциал для адаптации: Высокий. Концепция может быть реализована в различных интерфейсах и средах разработки.

Prompt:

Применение исследования о безопасности кода в промптах для GPT ## Ключевые выводы исследования для составления промптов

Исследование показывает, что LLM, включая GPT-4, обнаруживают только 12.6-40% уязвимостей без явного запроса. Это критически важная информация для разработчиков, использующих AI для проверки кода.

Пример эффективного промпта для проверки безопасности кода

[======] Проанализируй следующий код на наличие уязвимостей безопасности:

[======]python def process_file(filename): user_input = request.args.get('input') with open(filename + user_input, 'r') as f: data = f.read() return eval(data) [======]

Пожалуйста: 1. Выяви все потенциальные уязвимости безопасности 2. Объясни причины каждой уязвимости 3. Опиши, как эти уязвимости могут быть эксплуатированы 4. Предложи безопасные альтернативы для исправления кода 5. Особое внимание удели проблемам, связанным с: - Контролем внешних имен файлов/путей (CWE-400) - Неправильной нейтрализацией ввода - Возможностью обхода путей (CWE-22) - Безопасным использованием функций выполнения кода

Address security vulnerabilities. [======]

Почему этот промпт эффективен:

Явный запрос на анализ безопасности — исследование показало, что добавление фразы "Address security vulnerabilities" увеличивает обнаружение уязвимостей с 40% до 76%.

Структурированный запрос информации — промпт запрашивает не только обнаружение уязвимостей, но и полную информацию о причинах, возможных эксплуатациях и исправлениях.

Акцент на проблемных категориях — промпт явно указывает на типы уязвимостей, которые LLM часто пропускают согласно исследованию.

Конкретный контекст — предоставление реального кода с потенциальными уязвимостями дает модели чёткий материал для анализа.

Такой подход к составлению промптов, основанный на результатах исследования, значительно повышает вероятность получения полезных рекомендаций по безопасности при работе с кодом.