SecureFalcon: Удалось ли нам достичь автоматического обнаружения уязвимостей в программном обеспечении с помощью LLM?

Дата: 2025-03-03 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2307.06616

Рейтинг: 70
Адаптивность: 75

Ключевые выводы:

Исследование направлено на создание эффективной модели для автоматического обнаружения уязвимостей в программном обеспечении с использованием больших языковых моделей (LLM). Основной результат - разработка SecureFalcon, компактной модели на основе Falcon-40B, которая достигает 94% точности в бинарной классификации и до 92% в мультиклассовой классификации уязвимостей, превосходя существующие модели при мгновенном времени вывода на CPU.

Объяснение метода:

Исследование демонстрирует эффективное применение LLM для обнаружения уязвимостей в коде с высокой точностью (94%). Предлагаемая архитектура SecureFalcon и методология имеют значительную ценность для разработчиков и могут быть интегрированы в инструменты разработки. Однако узкая специализация (только C/C++ код) и необходимость значительных ресурсов для воспроизведения ограничивают непосредственную применимость для широкой аудитории.

Ключевые аспекты исследования: 1. **Создание SecureFalcon** - компактная модель с 121 миллионом параметров, основанная на FalconLLM40B, специально настроенная для обнаружения уязвимостей в программном обеспечении. 2. **Использование двух наборов данных для обучения**: FormAI (синтетические данные, созданные с помощью GPT-3.5-turbo и проверенные ESBMC) и FalconVulnDB (агрегированный набор данных из нескольких публичных источников). 3. **Высокая точность обнаружения уязвимостей**: 94% в бинарной классификации (уязвимый/неуязвимый код) и 92% в многоклассовой классификации (определение конкретного типа уязвимости). 4. **Превосходство над традиционными ML-моделями и другими LLM**: SecureFalcon превосходит традиционные алгоритмы машинного обучения на 11% и существующие модели LLM, такие как BERT, RoBERTa и CodeBERT, на 4%. 5. **Быстрое время вывода**: модель обеспечивает время вывода, достаточное для интеграции в системы завершения кода в режиме

реального времени.

Дополнение: Для работы методов этого исследования действительно требуется дообучение модели, так как SecureFalcon представляет собой специально настроенную версию FalconLLM40B. Однако многие концепции и подходы могут быть адаптированы для использования в стандартном чате с LLM без необходимости в дообучении.

Концепции и подходы, которые можно применить в стандартном чате:

Структурированный анализ кода Можно формулировать промпты, которые просят LLM анализировать код по определенной структуре: сначала искать проблемы с управлением памятью, затем проблемы с вводом данных и т.д. Пример: "Проанализируй этот C-код шаг за шагом, сначала проверяя на утечки памяти, затем на переполнение буфера, затем на проблемы с указателями."

Использование примеров из наборов данных

В промпты можно включать примеры уязвимостей из известных наборов данных (например, CWE) для сравнения Пример: "Вот пример кода с уязвимостью CWE-119 (переполнение буфера): [пример]. Проверь, содержит ли мой код похожие уязвимости."

Многоэтапная проверка

Можно разбить анализ кода на несколько этапов, сначала запрашивая общий анализ, затем уточняя конкретные аспекты Пример: "Сначала укажи все подозрительные участки кода, затем для каждого участка определи тип возможной уязвимости."

Использование специализированной терминологии

Включение в запросы специфических терминов и концепций из CWE и других стандартов Пример: "Проверь этот код на наличие уязвимостей из категорий CWE-120, CWE-476 и CWE-190."

Контрпримеры и проверка

Можно просить LLM генерировать контрпримеры для проверки наличия уязвимостей Пример: "Если в этом коде есть уязвимость переполнения буфера, приведи конкретный пример входных данных, которые могут вызвать эту уязвимость." Потенциальные результаты от применения этих подходов: - Повышение точности обнаружения уязвимостей в коде по сравнению с простым запросом "найди ошибки в коде" - Более структурированный и систематический анализ кода - Лучшее понимание типов уязвимостей и их причин - Возможность обнаружения более сложных и неочевидных уязвимостей - Повышение осведомленности разработчиков о потенциальных проблемах безопасности

Хотя такой подход не достигнет точности специально обученной модели (94%), он может значительно улучшить результаты анализа кода в стандартном чате с LLM.

Анализ практической применимости: **1. Создание SecureFalcon** - **Прямая применимость**: Ограниченная. Обычные пользователи не могут напрямую воспроизвести процесс создания и обучения такой модели без значительных вычислительных ресурсов и экспертизы. - **Концептуальная ценность**: Высокая. Демонстрирует, что даже относительно компактные LLM (121M параметров) могут эффективно решать специализированные задачи при правильной настройке. - **Потенциал для адаптации**: Средний. Архитектурные решения и подход к уменьшению размера модели могут быть адаптированы для других задач анализа кода.

2. Использование двух наборов данных для обучения - **Прямая применимость**: Низкая. Пользователи не могут непосредственно использовать эти наборы данных в своих запросах к LLM. - **Концептуальная ценность**: Высокая. Демонстрирует важность комбинирования синтетических и реальных данных для обучения моделей, что может помочь пользователям понять ограничения и возможности чат-моделей. - **Потенциал для адаптации**: Средний. Понимание структуры данных, использованных для обучения, может помочь пользователям формулировать более эффективные запросы к моделям.

3. Высокая точность обнаружения уязвимостей - **Прямая применимость**: Высокая. Пользователи могут использовать модель для проверки своего кода на уязвимости, что особенно ценно для разработчиков. - **Концептуальная ценность**: Высокая. Показывает возможности LLM в задачах анализа кода и обнаружения уязвимостей. - **Потенциал для адаптации**: Высокий. Методы обнаружения уязвимостей могут быть применены к другим задачам статического анализа кода.

4. Превосходство над традиционными ML-моделями и другими LLM - **Прямая применимость**: Средняя. Пользователи могут выбрать наиболее подходящий инструмент для анализа кода. - **Концептуальная ценность**: Высокая. Демонстрирует преимущества специализированных LLM над традиционными подходами. - **Потенциал для адаптации**: Средний. Понимание сравнительных преимуществ разных подходов может помочь в выборе инструментов для других задач.

5. Быстрое время вывода - **Прямая применимость**: Высокая. Позволяет использовать модель в реальных сценариях разработки и в интерактивных средах. - **Концептуальная ценность**: Средняя. Показывает компромисс между размером модели, точностью и скоростью вывода. - **Потенциал для адаптации**: Высокий. Методы оптимизации для быстрого вывода могут быть применены к другим моделям и задачам.

Prompt:

Использование знаний из исследования SecureFalcon в промтах для GPT ## Ключевые знания из исследования

Исследование SecureFalcon демонстрирует высокую эффективность специализированных LLM в обнаружении уязвимостей в коде:

• 94% точность в бинарной классификации (уязвимый/безопасный код)

• 92% точность в мультиклассовой классификации (определение конкретных типов уязвимостей)

• Особенно высокая точность (близкая к 100%) для определенных типов уязвимостей:

• CWE-78 (OS Command Injection)

• CWE-121 (Stack-Based Buffer Overflow)

• CWE-122 (Heap-Based Buffer Overflow)

• CWE-762 (Mismatched Memory Management)

Пример промта для GPT

[======] Я хочу, чтобы ты выступил в роли эксперта по безопасности программного обеспечения, используя знания, аналогичные модели SecureFalcon.

Проанализируй следующий фрагмент кода на C/C++ и: 1. Определи, содержит ли код уязвимости (да/нет) 2. Если уязвимости присутствуют, классифицируй их по стандарту CWE 3. Особенно обрати внимание на: - OS Command Injection (CWE-78) - Stack-Based Buffer Overflow (CWE-121) - Heap-Based Buffer Overflow (CWE-122) - Mismatched Memory Management (CWE-762) 4. Предложи исправления для обнаруженных уязвимостей

Код для анализа: [======]c void process_user_input(char *input) { char command[100]; sprintf(command, "echo %s", input); system(command);

char *buffer = malloc(10); strcpy(buffer, input); // Обработка данных free(buffer); buffer[0] = '\0'; } [======]

Формат ответа: - Уязвимость обнаружена: [Да/Нет] - Идентифицированные CWE: [список] - Подробный анализ: [описание каждой уязвимости] - Рекомендуемые исправления: [код с исправлениями] [======]

Как работают знания из исследования в этом промте

Структура запроса: Промт опирается на способность моделей, подобных SecureFalcon, выполнять бинарную и мультиклассовую классификацию

уязвимостей.

Фокус на конкретных типах уязвимостей: Промт специально указывает на типы уязвимостей, которые модель SecureFalcon определяет с высокой точностью (близкой к 100%).

Комплексный анализ: Запрос требует не только обнаружения уязвимостей, но и их классификации по стандарту CWE, что соответствует возможностям SecureFalcon в мультиклассовой классификации.

Практическое применение: Промт отражает одно из практических применений SecureFalcon, упомянутых в исследовании — анализ кода на наличие уязвимостей в процессе разработки.

Такой подход позволяет эффективно использовать общие языковые модели для задач, в которых специализированные модели (как SecureFalcon) показывают высокие результаты.