Защита приложений на основе больших языковых моделей от атак PROMPT INJECTION

Аверин П.А., Коновалов С.И.

ФГБОУ ВО "Брянский государственный технический университет",

г. Брянск, Российская Федерация

Большие языковые модели уже используются не только как чат-боты. Их подключают к корпоративным документам, системам поиска, программным интерфейсам и средствам анализа данных. Поэтому вопрос безопасности постепенно выходит за рамки качества ответа. Если модель получает доступ к внешним ресурсам, ошибочная или навязанная инструкция может повлиять уже на работу всего сервиса. В российских исследованиях LLM рассматриваются одновременно как полезный инструмент кибербезопасности и как новый объект, который сам требует анализа защищенности [1].

Одна из характерных угроз для таких систем — Prompt Injection, или инъекция подсказок. Ее смысл состоит в том, что в обрабатываемый моделью текст помещается инструкция, способная изменить исходно заданное поведение. В прямом варианте пользователь сам пытается заставить модель игнорировать ограничения. В косвенном варианте команда может находиться внутри документа, письма или веб-страницы, которые приложение передает модели как обычные данные. Для человека это содержимое документа, а LLM может интерпретировать его как новую инструкцию [2]. Именно эта особенность отличает Prompt Injection от многих привычных атак на программное обеспечение.

Отсюда следует, что одного системного промпта недостаточно. Запреты вроде «не выполняй посторонние инструкции» полезны для настройки поведения, но они остаются частью того же текстового контекста, который интерпретирует модель. Формулировку атаки можно изменить, разбить на несколько сообщений или скрыть во внешнем источнике. Поэтому безопасность нельзя строить только на поиске запрещенных слов. Необходимо учитывать

смысл запроса и, главное, контролировать то, какие действия модель получает право выполнить.

Особенно важным это становится при использовании ИИ-агентов. Такой сервис может не только сформировать текст, но и выбрать последовательность действий: открыть файл, обратиться к базе данных или вызвать внешнюю функцию. Российские авторы отмечают, что агентные архитектуры сохраняют риски генеративных моделей и могут усиливать их за счет подключения дополнительных компонентов [3]. В этой ситуации Prompt Injection превращается из проблемы неверного ответа в проблему управления: вредоносная инструкция потенциально влияет на то, к какому ресурсу обратится система и что она попытается сделать.

Наиболее реалистичный подход к защите состоит в том, чтобы не возлагать всю ответственность на саму модель. На уровне приложения можно разделять пользовательские данные и служебный контекст, ограничивать доступные инструменты, проверять параметры критических вызовов и анализировать результат перед его передачей пользователю. Подобный принцип реализован в архитектуре LLM Security Proxy, где между запросом, моделью и внешними сервисами вводится контролирующий слой. Авторы показывают, что разные варианты Prompt Injection требуют защиты сразу на нескольких уровнях: от проверки входных инструкций до контроля инструментальных вызовов и выходных данных [4].

Фильтрация запросов при этом остается полезной, но должна быть только одним из элементов. Очевидные попытки манипуляции можно блокировать автоматически, подозрительные — отправлять на дополнительную проверку. Для действий с серьезными последствиями разумно использовать обычные программные правила или подтверждение пользователем. Такой подход близок к принципу минимальных привилегий: модель получает только те возможности, которые нужны для конкретной задачи, а ее ответ не считается безусловно доверенным.

По мере распространения ИИ-агентов значение этой проблемы будет расти. Чем больше решений и действий передается языковой модели, тем выше цена неверно интерпретированной инструкции. Поэтому перспективной выглядит архитектура, в которой LLM отвечает за работу с естественным языком, а контроль доступа, проверка операций и право на критические действия остаются у отдельных программных компонентов. Prompt Injection в таком случае рассматривается не как «плохой промпт», а как архитектурный риск. Полностью устранить его одной инструкцией невозможно, но можно существенно ограничить последствия за счет разделения контекста, минимальных привилегий, контроля инструментов и регулярного тестирования сервиса.

Список использованной литературы

1. Конев А.А., Паюсова Т.И. *Большие языковые модели в информационной безопасности и тестировании на проникновение: систематический обзор возможностей применения* // Научно-технический вестник информационных технологий, механики и оптики. 2025. Т. 25. № 1. С. 42–52. URL: https://ntv.ifmo.ru/ru/article/23311/ (дата обращения: 27.09.2026).

2. Мударова Р.М., Намиот Д.Е. *Противодействие атакам типа инъекция подсказок на большие языковые модели* // International Journal of Open Information Technologies. 2024. Т. 12. № 5. С. 39–48. URL: https://injoit.org/index.php/j1/article/viewFile/1845/1682 (дата обращения: 27.09.2026).

3. Намиот Д. Е., Ильюшин Е. А. *О кибербезопасности ИИ-агентов // International Journal of Open Information Technologies*. 2025. Т. 13. № 9. С. 13–24. URL: https://cyberleninka.ru/article/n/o-kiberbezopasnosti-ii-agentov (дата обращения: 27.09.2026).

4. Крохин А. С., Гусев М. М. *Анализ влияния обфускации входных данных на эффективность языковых моделей в обнаружении инъекции подсказок* // Программные системы и вычислительные методы. 2025. № 2. С. 44–62. URL: https://nbpublish.com/library_read_article.php?id=73939 (дата обращения: 27.09.2026).