УДК 004.492

МЕТОДИКА ПОВЫШЕНИЯ ЗАЩИЩЕННОСТИ LLM-СЕРВИСОВ С ПРИМЕНЕНИЕМ LLM SECURITY PROXY

Евглевская Наталья Валерьевна, канд. техн. наук, доцент университета ИТМО, n.evglevskaya@gmail.com, Россия, Санкт-Петербург, ИТМО

Казанцев Александр Артурович, аспирант, akazancev@itmo.ru, Россия, Санкт-Петербург, Университет ИТМО

Введение. В прикладных системах, использующих большие языковые модели для поддержки пользователей, анализа документов и автоматизации внутренних операций, языковая модель, как правило, применяется не изолированно, а в составе программного сервиса, обеспечивающего обработку пользовательских запросов и взаимодействие с внешними компонентами информационной системы. В рамках настоящей работы такой программный комплекс, включающий языковую модель и обеспечивающую ее функционирование сервисную логику, далее обозначается как LLM-сервис. В архитектуре LLM-сервиса пользовательский запрос, входное текстовое сообщение, передается в языковую модель, а сформированный ответ может использоваться для выполнения операций во внешних компонентах, например, обращений к базе данных, вызовов прикладных API и программных инструментов.

В настоящей работе рассматриваются атаки типа Prompt Injection [1], при которых злоумышленник внедряет в текст запроса инструкции, изменяющие поведение LLM-сервиса и приводящие к обходу установленных ограничений. В условиях интеграции с корпоративными данными и инструментами успешная реализация Prompt Injection может приводить к раскрытию конфиденциальной информации из контекста[2], а также к формированию и инициированию несанкционированных действий во внешних компонентах, например, выполнению запросов к базе данных или вызовов внутренних API с некорректными параметрами.

Существующие подходы к противодействию Prompt Injection, как правило, фокусируются на отдельных мерах — фильтрации входных запросов или контроле выходных данных [3]. При этом для LLM-сервиса, взаимодействующего с внешними системами, требуется единый инфраструктурный механизм, обеспечивающий согласованное применение проверок и ограничений на всех этапах обработки запроса и выполнения инструментальных действий.

Основная часть. Предлагается архитектурное решение LLM Security Proxy — промежуточный защитный компонент, размещаемый между пользователем и LLM-сервисом. Прокси работает по принципу «контроль на входе — управление контекстом — контроль действий — контроль на выходе»: входящий запрос проверяется до передачи в модель, контекст взаимодействия формируется в изолированном виде, инструментальные действия выполняются только в рамках разрешенных политик, а выходной ответ проверяется перед выдачей пользователю или передачей во внешнюю систему.

Режим функционирования прокси выбирается на основе оценки риска Prompt Injection и определяет набор применяемых ограничений: требования к валидации входного запроса; правила формирования и очистки контекста; доступность инструментов и допустимые параметры их вызовов; строгость проверки выходного ответа.

LLM Security Proxy включает следующие модули:

Semantic Guard — анализирует входной текст запроса и выявляет признаки Prompt Injection — попытки навязать сервису новые правила, обойти системные инструкции или принудить к выполнению запрещенных действий. Модуль формирует количественную оценку риска, необходимую для выбора режима защиты.

Policy Engine — принимает оценку риска и выбирает режим работы normal, suspicious или safe, тем самым задавая уровень ограничений для последующих модулей. Назначение модуля — обеспечить согласованное применение политик безопасности при обработке запроса и при обращении к внешним компонентам.

Context Manager — управляет контекстом взаимодействия с моделью и обеспечивает его изоляцию: разделяет системные инструкции, описания инструментов, историю диалога и текущий ввод пользователя, исключая влияние пользовательского текста на защищаемые части контекста. Назначение модуля — снизить вероятность подмены инструкций и «перепрошивки» поведения сервиса через Prompt Injection.

Tool / Action Manager — контролирует инструментальные действия LLM-сервиса — вызовы API, обращения к базе данных, операции с файлами — и разрешает их только при соответствии политике и выбранному режиму. Назначение модуля — предотвратить выполнение несанкционированных операций во внешних системах.

Output Guard — проверяет сформированный ответ перед выдачей пользователю или передачей во внешнюю систему и блокирует/редактирует выходные данные при обнаружении нарушений, например, раскрытия конфиденциальной информации из контекста или генерации опасных команд. Назначение модуля — предотвратить утечки и распространение нежелательных инструкций, возникших вследствие Prompt Injection.

Обработка запроса в прокси выполняется последовательно: Semantic Guard оценивает риск Prompt Injection, Policy Engine выбирает режим, Context Manager формирует контролируемый контекст для модели, Tool / Action Manager проверяет и ограничивает инструментальные действия, а Output Guard контролирует итоговый ответ.

Выводы, практическое использование и внедрение. Предложенный подход обеспечивает инфраструктурную защиту LLM-сервиса от Prompt Injection без изменения внутренней архитектуры языковой модели и может применяться как единый шлюз безопасности для сервисов, интегрированных с корпоративными данными и инструментами. Практическое внедрение возможно поэтапно: развертывание прокси в контуре сервиса, настройка политик и режимов для критичных сценариев, проведение испытаний на типовых векторах Prompt Injection и последующее расширение правил контроля контекста, действий и выходных данных.

Список использованных источников

1. Мударова Р.М. Противодействие атакам типа инъекция подсказок на большие языковые модели // Современные информационные технологии и ИТ-образование. 2024. Т. 20. № 2. С. 112–119.
2. Prompt Injection: проблема лингвистических уязвимостей больших языковых моделей на современном этапе // Кибербезопасность и цифровые технологии. 2024. № 3. С. 22–30.
3. Евглевская Н.В., Казанцев А.А. Обеспечение безопасности сложных систем с интеграцией больших языковых моделей: анализ угроз и методов защиты // Вестник информационной безопасности. 2024. № 2. С. 18–27.