5.4. Безопасность и комплаенс

Основные риски и уязвимости LLM-приложений

OWASP GenAI Top 10

OWASP опубликовал список из десяти наиболее критичных уязвимостей для приложений на основе больших языковых моделей. Этот список помогает оценить угрозы и встроить механизмы защиты в архитектуру систем:

#	Уязвимость	Разъяснение
LLM01	Prompt Injection	Подмена инструкций: злоумышленник вставляет команды, изменяющие поведение модели, что приводит к несанкционированному доступу, утечкам данных или генерации нежелательного контента[1].
LLM02	Insecure Output Handling	Отсутствие проверки и фильтрации выходных данных может привести к исполнению вредоносного кода, SQL-инъекциям и другим атакам, особенно когда вывод LLM передаётся в другие сервисы[1].
LLM03	Training Data Poisoning	Вредоносные данные в тренировочном наборе ухудшают точность модели и могут внедрить предвзятость или скрытые инструкции[1].
LLM04	Model Denial of Service	Массированные или ресурсоёмкие запросы перегружают модель, вызывая задержки, увеличение стоимости и отказ в обслуживании[1].
LLM05	Supply Chain Vulnerabilities	Компрометация сторонних плагинов, библиотек или моделей приводит к уязвимостям, утечкам данных и нарушениям доверия[1].
LLM06	Sensitive Information Disclosure	Ошибки в контроле доступа или генерации позволяют модели раскрывать конфиденциальные данные, что ведёт к юридическим рискам и утрате конкурентного преимущества[1].
LLM07	Insecure Plugin Design	Плагины, обрабатывающие непроверенный ввод без достаточной изоляции, могут привести к удалённому выполнению кода или эскалации привилегий[1].
LLM08	Excessive Agency	Предоставление модели неограниченного доступа к инструментам и ресурсам может вызвать непредсказуемое поведение и нарушить безопасность или приватность[1].
LLM09	Overreliance	Слепое доверие выводам модели и игнорирование проверок повышает риск ошибок и юридических последствий[1].
LLM10	Model Theft	Кража и несанкционированное копирование моделей приводит к утечке интеллектуальной собственности и повторному использованию с вредоносными целями[1].

Эти угрозы актуальны как для публичных чатов, так и для внутренних инженерных ассистентов. Обеспечение безопасности требует не только настроек модели, но и разработки безопасной архитектуры вокруг LLM.

Защита от prompt-injection и вредоносных данных

Prompt Shields (Microsoft Azure AI Content Safety)

Microsoft разработала механизм `Prompt Shields`, который анализирует входящие запросы и «прикреплённые» документы, чтобы определить попытки изменить системные правила. Этот механизм распознаёт `user prompt attacks` (попытки изменить инструкцию модели) и `document attacks` (заражение сторонних документов скрытыми командами). В документации Azure указан пример: в классическом `prompt`-атака пользователь пытается заме-

нить системную персону и потребовать выдачу ответов без ограничений. `Prompt Shields` классифицирует такие атаки и блокирует запросы[2]. Подкатегории атак включают:

- Изменение системных правил — просьба игнорировать инструкции или действовать без ограничений[2].
- Встроенный разговор — в запрос вставляют ложные «диалоги» для сбивания модели[2].
- Ролевое переопределение — просьба играть другого персонажа без правил[2].
- Кодирование и стили — использование шифров или необычного форматирования, чтобы обойти фильтры[2].

Также есть защита от `document attacks`, где вредоносные инструкции скрыты в документах. Такие атаки могут вызывать несанкционированные команды, утечку данных или кражу идентификационных данных[2]. Практический вывод: разработчики должны реализовать проверку входящих запросов и вложенных данных. Для внутренних ассистентов рекомендуется создавать белые списки допустимых команд и явно запрещать изменение системных правил. Использование встроенных механизмов (`Prompt Shields`, фильтры конфиденциальных данных) снижает риск эксплойтов.

Контент-фильтры и выходная валидация

- **Контент-фильтры** — сервис Azure Content Safety обеспечивает фильтрацию опасного входного и выходного контента. Разработчики могут настроить уровни чувствительности для категорий (насилие, ненавистнический язык и т. д.) и настроить логику прекращения генерации.
- **Валидация выходов:** Guardrails AI предлагает библиотеку для контроля структуры и содержания выходных данных. Документация демонстрирует, как с помощью валидаторов проверять отдельные поля структурированного ответа (например, имя и email) с помощью регулярных выражений и Pydantic. После определения схемы создаётся `Guard` и используется метод `validate` для проверки данных[3]. Это позволяет ловить «галлюцинации» и несоответствия до передачи результата в основной сервис.
- **Блок “Guardrails” в low-code-платформах:** в платформе Sim описан блок Guardrails, который проверяет контент по нескольким критериям. Он обеспечивает *качество данных*, предотвращает галлюцинации, обнаруживает персональные данные (PII) и проверяет формат перед тем, как данные пойдут по пайплайну. Доступны типы валидации: `JSON`-валидация, `regex`-валидация, и `PII`.

Выводы для инженерных ассистентов

- **Явно фиксируйте системные инструкции.** Используйте «system»-сообщения в API, которые не должны быть перезаписаны пользовательским вводом. Разделяйте роли (`system`, `user`, `assistant`) и обнуляйте историю при смене задач.
- **Проверяйте вход и выход.** Внедряйте фильтры на уровне платформы (Azure Content Safety, Prompt Shields) и валидаторы (Guardrails AI, Pydantic). Проверяйте, что JSON-ответы соответствуют ожидаемой схеме; применяйте регулярные выражения для полей (телефоны, `emails`). Автоматическая проверка помогает обнаружить инъекции и неверный формат.

- **Логи и аудиты.** Записывайте промпты и результаты, используйте системы аудита для отслеживания и расследования инцидентов.
- **Изоляция инструментов.** Когда LLM получает доступ к инструментам (агентный режим), разграничивайте привилегии: отдельный слой (`sandbox`) выполняет код и возвращает результат; LLM не должен иметь прямого доступа к критичным системам.

Управление данными и соответствие NIST AI RMF

NIST выпустил `AI Risk Management Framework (AI RMF 1.0)` и `Generative AI Profile (NIST AI 600-1)`. Эти документы предназначены для добровольного использования и помогают организациям строить доверительные и безопасные AI-системы. Основные рекомендации:

- **Включение требований правовых норм.** Организации должны понять и документировать юридические требования (конфиденциальность, авторские права) и интегрировать их в процессы разработки и эксплуатации моделей[4].
- **Прозрачность и происхождение данных.** Необходимо устанавливать процедуры, чтобы документировать происхождение тренировочных данных и генерируемого контента; обеспечивать прозрачность источников, сохраняя коммерческую тайну[4].
- **Оценка рисков и порогов.** Предлагается определять уровни риска и минимальные критерии качества перед развёртыванием модели; рассматривать факторы, влияющие на информационную целостность, вредоносные или ненадлежащие выходы, потенциальный ущерб для групп населения и возможность злонамеренного использования[4].
- **Политики по недопустимому контенту.** Организации должны разработать политики, запрещающие генерацию незаконного контента, включая вредоносные, насильственные или дискриминационные материалы[4].
- **Периодический мониторинг.** В документе указано, что процесс управления рисками требует постоянного мониторинга и обновления политик; необходимо чётко назначать ответственных лиц и определять частоту ревизий[4].

Генеративный профиль NIST подчёркивает, что многие риски усиливаются в случае LLM и требуют дополнительного внимания: безопасность данных, согласование с правами интеллектуальной собственности, предотвращение конфабуляций, предотвращение предвзятости и обеспечение прозрачности происхождения выводов.

Роль библиотек и структурированного промптинга

Чтобы уменьшить риски галлюцинаций и упростить комплаенс, разработчики всё чаще применяют *структурированные форматы вывода*. OpenAI в документации описывает, что схемы (`JSON Schema` или `Pydantic`) позволяют *надёжно обеспечить типовую безопасность и отказ для неверных запросов*; такие методы делают промпты проще и предотвращают случайные форматные ошибки[5]. Наборы валидаторов (`Guardrails AI`, `Pydantic`, `Regex-match`) обеспечивают контроль форматных и семантических требований. Также существуют библиотеки (например, `Outlines`, `SGLang` или встроенные инструменты `vLLM`) для *грамматически ограниченной генерации*. Вместо пост-обработки, модель генерирует текст, следуя заранее определённой грамматике (`JSON Schema`, `EBNF`, регулярные выражения). Такие библиотеки используют конечные автоматы для исключения некорректных токенов и снижения риска инъекций.

Практические рекомендации для внедрения

- **Следуйте принципу наименьших привилегий.** Организуйте архитектуру так, чтобы LLM имел доступ только к необходимым данным и инструментам. Используйте прокси-слой, проверяющий каждую команду.
- **Используйте многоуровневые фильтры.** Комбинируйте контент-фильтры, механизмы `prompt-shield` и валидаторы выходного формата. Это снижает риск как явных инъекций, так и скрытых инструкций в документах.
- **Стандартизируйте промпты и выводы.** Применяйте строгие схемы (JSON/XML) и библиотеку `Guardrails` для проверки; избегайте `free-form` вывода, особенно если результат используется в приложениях (генерация кода, SQL-запросы).
- **Учитывайте требования NIST и OWASP.** Разработайте внутренние политики, охватывающие юридические требования, управление данными, прозрачность, оценку рисков и мониторинг.
- **Проводите обучение команды и руководства.** Без вовлечения тимлида разработчики могут избегать использования LLM. Формируйте культуру безопасности и экспериментирования: демонстрируйте преимущества, обеспечивайте обучение и контроль качества.
- **Планируйте инцидент-репортинг.** Организация должна быть готова реагировать на нарушения — мониторить логи, создавать процессы для остановки систем при обнаружении – , проводить разбор инцидентов и корректировать политики.

Заключение

Безопасность и комплаенс — неотъемлемая часть внедрения LLM-ассистентов в разработку. OWASP Top 10 указывает основные классы уязвимостей, а Azure `Prompt Shields` и `Guardrails AI` демонстрируют практические меры защиты. Документы `NIST AI RMF` побуждают компании внедрять системное управление рисками, уделяя внимание происхождению данных, правовым требованиям и постоянному мониторингу. Комплексный подход — сочетание политик, фильтров, структурированных промптов и контролируемого доступа — позволяет использовать LLM в инженерных задачах, минимизируя риски для организации и её пользователей.

Источники

[1] Источник: owasp.org
[2] Источник: learn.microsoft.com
[3] Источник: guardrailsai.com
[4] Источник: nvlpubs.nist.gov
[5] Источник: platform.openai.com