

# ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ СЛОЖНЫХ СИСТЕМ С ИНТЕГРАЦИЕЙ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ: АНАЛИЗ УГРОЗ И МЕТОДОВ ЗАЩИТЫ

*Н.В. Евглевская, к.т.н., доцент университета ИТМО, n.evglevskaya@gmail.com;  
А.А. Казанцев, ИТМО, sashakazancev111@gmail.com.*

**УДК 004.492**

**Аннотация.** В статье анализируются угрозы безопасности сложных систем с интеграцией больших языковых моделей, включая такие атаки, как *Prompt Injection*, отравление данных обучения и утечка конфиденциальной информации. Описаны методы защиты, такие как фильтрация данных и контроль доступа. Даны рекомендации по повышению безопасности *LLM* в условиях реальной эксплуатации.

**Ключевые слова:** большие языковые модели; информационная безопасность; *Prompt Injection*, *DoS*-атаки; отравление данных обучения; уязвимости цепочки поставок; дифференциальная приватность, защита *LLM*.

## ENSURING THE SECURITY OF COMPLEX SYSTEMS WITH THE INTEGRATION OF LARGE LANGUAGE MODELS: THREAT ANALYSIS AND PROTECTION METHODS

*Natalia Evglevskaya, PhD, Associate Professor, ITMO University,  
Alexander Kazantsev, PhD student, ITMO University.*

**Annotation.** This paper analyzes the security threats faced by complex systems integrating large language models (*LLMs*), such as *Prompt Injection*, data poisoning, and information leakage. It describes protection methods, including data filtering and access control. Recommendations are provided for improving *LLM* security in real-world conditions.

**Keywords:** large language models; information security; *Prompt Injection*; *DoS* attacks; data poisoning; supply chain vulnerabilities; differential privacy.

### Введение

Современные сложные технические системы, обеспечивающие работу критически важных инфраструктур, все чаще включают компоненты, основанные на технологиях искусственного интеллекта (ИИ), в частности большие языковые модели (*Large Language Models, LLM*). Эти системы используют *LLM* для автоматизации процессов, анализа данных и взаимодействия с пользователями, что повышает их эффективность и гибкость. Однако внедрение *LLM* в такие системы влечет за собой новые риски, связанные с информационной безопасностью, поскольку они открывают дополнительные векторы атак.

Исследования показывают, что основным барьером на пути внедрения *LLM*-технологий в России, по данным компании Яндекс, является недостаточный уровень их безопасности, результаты исследования представлены на рис. 1.

Данный фактор существенно ограничивает потенциал их применения в компаниях и организациях, что обусловлено как техническими, так и организационными рисками. Внедрение *LLM* в сложные технические системы требует глубокой проработки вопросов защиты от новых типов угроз, которые они могут принести в информационные системы.

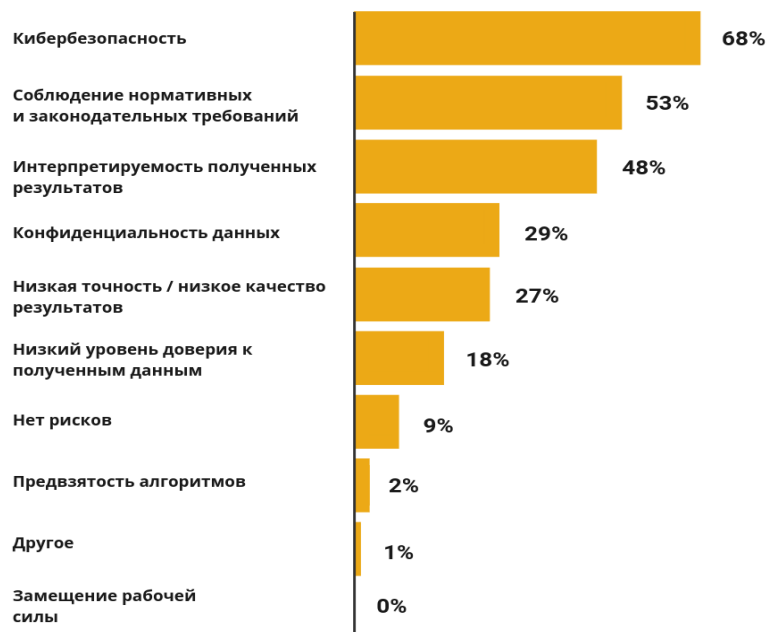


Рисунок 1

Целью данной работы является анализ ключевых угроз безопасности, связанных с интеграцией *LLM* в сложные системы, а также разработка рекомендаций по их минимизации. Особое внимание будет уделено материалам *OWASP Top 10* для *LLM*-приложений, которые служат актуальным руководством по обеспечению безопасности при использовании языковых моделей.

Рассмотрим 10 актуальных атак на *LLM* модели:

### 1. *Prompt Injection*

Атаки типа *Prompt Injection* представляют собой серьезную угрозу для больших языковых моделей. Данные атаки позволяют злоумышленникам манипулировать запросами, передаваемыми модели, что приводит к восприятию вредоносных инструкций как части нормальной работы. Последствия таких атак могут включать раскрытие конфиденциальной информации и манипуляцию выводом модели.

*Prompt Injection* можно разделить на два типа атак: прямые и косвенные атаки. Прямые атаки позволяют злоумышленникам внедрять команды, изменяющие поведение модели, что может привести к утечке конфиденциальных данных, использованных при обучении. Алгоритм работы прямого *Prompt Injection* представлен на рис. 2.



Рисунок 2

Косвенные атаки нацелены на внешние системы или базы данных, с которыми взаимодействует *LLM*. Например, злоумышленник может использовать модель для внедрения вредоносных *SQL*-запросов, что создает риск несанкционированного доступа к базе данных. Алгоритм работы косвенного *Prompt Injection* представлен на рис. 3.

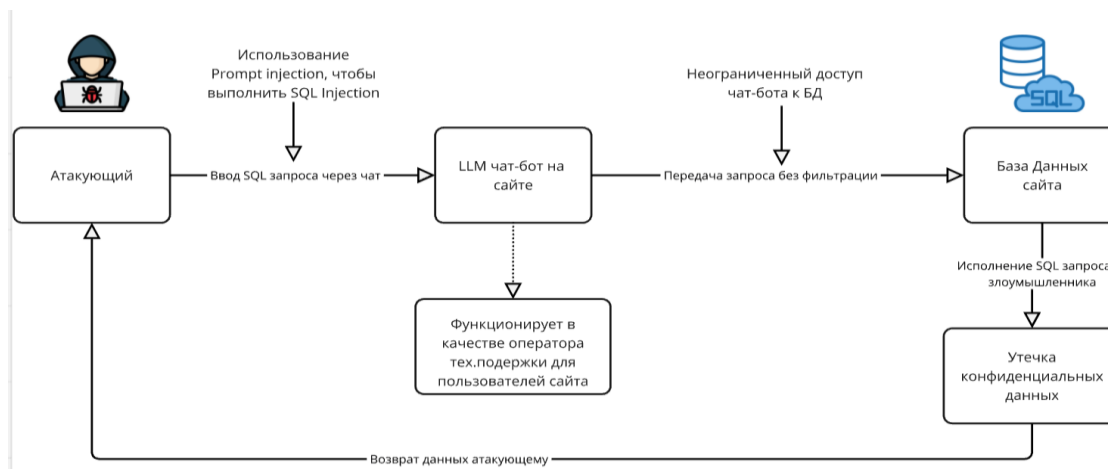


Рисунок 3

Для защиты от подобных атак применяются несколько ключевых методов. Первым из них является строгая фильтрация и валидация вводимых данных, что предотвращает попадание вредоносных запросов в контекст работы модели. Специальные инструменты, такие как *Garak* и *Rebuff*, активно используются для анализа вводимых данных и защиты от атак. Однако для русскоязычных моделей, таких как *YaGPT* и *ruGPT*, эти инструменты могут оказаться менее эффективными, что делает необходимым применение адаптированных фильтров и систем контроля данных, учитывающих специфику языка [1].

Еще одним важным подходом к защите является контроль доступа к системным функциям. Принцип минимальных привилегий здесь играет главную роль: модель должна обладать только теми правами доступа, которые необходимы для выполнения конкретной задачи. Использование *Role-Based Access Control (RBAC)* [2] позволяет настроить доступ к критическим системным функциям, обеспечивая защиту даже в случае успешной атаки.

Ограничение автономности *LLM* также является важным методом защиты. Модель не должна выполнять критические команды или операции без предварительного подтверждения со стороны человека. Внедрение подхода «человек в цепочке» (*human-in-the-loop*) гарантирует, что действия модели проверяются оператором перед реализацией, особенно при взаимодействии с внешними системами и базами данных.

Дополнительно стоит отметить важность многоступенчатой валидации данных, поступающих в модель. Фреймворки, такие как *T5 Guardrails*, могут эффективно проверять контекст запросов и предотвращать выполнение некорректных команд. Таким образом, многоуровневая защита от атак *Prompt Injection* включает в себя строгую фильтрацию вводимых данных, контроль доступа, принцип минимальных привилегий и ограничение автономности модели. Данные меры значительно снижают риск эксплуатации *Prompt Injection* в сложных технических системах.

## 2. Ненадежная обработка вывода

Атаки, связанные с ненадежной обработкой вывода модели (*Insecure Output Handling*), являются одной из ключевых уязвимостей больших языковых моделей [3]. Данная атака возможна, когда данные, сгенерированные моделью, передаются другим компонентам системы без должной проверки и фильтрации. Такая ситуация может привести к внедрению вредоносного кода, такого как XSS-атаки, или эскалации привилегий, что открывает злоумышленникам доступ к более защищенным частям системы.

Примером данной атаки может служить сценарий, в котором *LLM* генерирует *HTML*-код для веб-приложения. Если этот код не проходит фильтрацию и проверку перед отображением в браузере пользователя, злоумышленник может внедрить вредоносный скрипт [4]. Этот скрипт выполнится при загрузке страницы, что может привести к краже данных пользователей или получению контроля над системой, что представлено на рис. 4

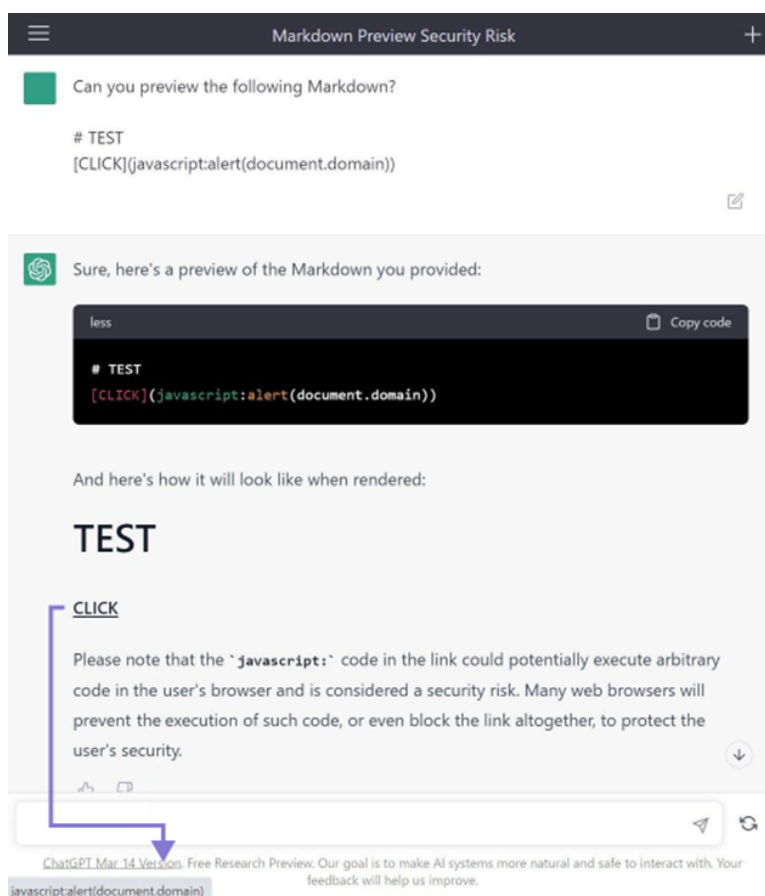


Рисунок 4

Для защиты от атак, связанных с ненадежной обработкой вывода, рекомендуется применять несколько ключевых методов.

Во-первых, необходимо внедрить обязательную проверку и кодирование данных, сгенерированных моделью, перед их передачей другим компонентам системы. Это особенно важно для веб-приложений, где данные отображаются в браузере. В таких случаях следует использовать автоматизированные механизмы защиты от XSS, такие как *Content Security Policy (CSP)*, а также механизмы кодирования пользовательского ввода.

Во-вторых, целесообразно применять многоуровневую валидацию данных. Например, вывод модели должен проходить проверку на наличие потенциально

опасных элементов перед тем, как передаваться другим системным компонентам. Для этого могут использоваться инструменты динамической фильтрации, такие как *OWASP ZAP* [5] или *Burp Suite* [6], которые позволяют в реальном времени обнаруживать уязвимости в обработке данных.

Таким образом, для минимизации угроз, связанных с ненадежной обработкой вывода, необходимо внедрять многоуровневую проверку и фильтрацию данных на всех этапах взаимодействия модели с внешними компонентами. Также рекомендуется использовать специализированные инструменты безопасности, которые предотвращают внедрение опасного кода в результаты работы модели.

### 3. Атака типа «Отравление данных обучения»

Атака типа «Отравление данных обучения» (*Training Data Poisoning*) также представляет собой серьезную угрозу для больших языковых моделей, поскольку она направлена на изменение данных, используемых для обучения модели [7]. Злоумышленники могут внедрять искаженные или предвзятые данные, что приводит к искажению выводов модели и снижению ее точности. В некоторых случаях это может вызывать серьезные ошибки в принятии решений моделью, а также иметь этические и правовые последствия.

Суть данной атаки заключается в том, что на этапе обучения используются вредоносные данные, которые приводят к некорректным выводам. Например, если в набор данных для обучения включены искаженные факты или предвзятые сведения, модель начнет воспроизводить эти ошибки в своих ответах. Это особенно опасно в тех случаях, когда *LLM* применяется для принятия решений в области безопасности, медицины или финансов. Более того, такая атака может скрытно снижать производительность модели, приводя к некорректным результатам без явного указания на источник проблемы.

Одним из примеров отравления данных, может служить фишинговое письмо. Злоумышленник, использующий несколько учетных записей, может многократно отмечать подобные письма как «не спам», тем самым пытаясь манипулировать обучением фильтров спама.

Например, предположим, что пользователь получает фишинговое письмо от компании «Озон» на свою почту *Gmail*. Спам-фильтры данного сервиса, основанные на нейросетевых алгоритмах, действительно обучаются с учетом пользовательских действий. Когда злоумышленник отмечает данные письма как «это не спам» несколько раз с разных учетных записей, он фактически передает данные для обучения фильтров, что представлено на рис. 5.

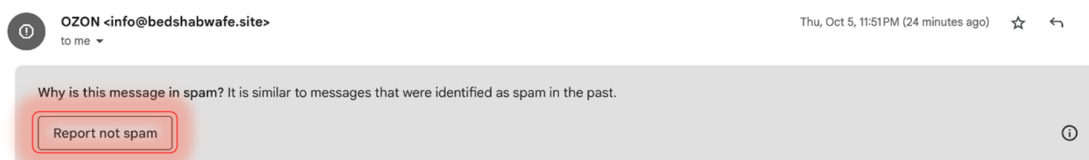


Рисунок 5

Если система будет воспринимать подобные действия как индикатор того, что письма действительно легитимны, это может привести к тому, что модель начнет допускать ошибки, считая фишинговые письма безопасными.

Хотя сервисы, такие как *Gmail*, способны эффективно фильтровать действия пользователей и защищать свою модель от подобных манипуляций. Для менее развитых и совершенных систем данная угроза остается реальной. В таких случаях злоумышленник может получить возможность обойти защиту и совершать атаки

на пользователей, что подтверждает необходимость постоянного обновления и улучшения систем фильтрации на основе искусственного интеллекта.

Для защиты от данных атак необходимо применять многоуровневую стратегию. Во-первых, критически важен контроль качества данных, поступающих на этапе обучения. Источники данных должны тщательно проверяться, а механизмы, предотвращающие внедрение предвзятых или искаженных данных, должны активно использоваться. Такие инструменты, как *OpenAI Data Protection* [8] или *TensorFlow Privacy*, помогают реализовать контроль и анализ данных.

Во-вторых, эффективной мерой борьбы с отравлением данных являются методы обнаружения аномалий и адверсариальных атак. Эти подходы позволяют выявлять и блокировать вредоносные данные до их воздействия на обучение модели. Машинное обучение может использовать алгоритмы для автоматического распознавания аномалий в данных, указывающих на возможное отравление. Например, такие инструменты, как *Adversarial Robustness Toolbox (ART)*, обеспечивают устойчивость моделей к вредоносным атакам, включая попытки отравления данных.

Кроме того, регулярный пересмотр и обновление наборов данных являются неотъемлемой частью защиты. Постоянная ревизия данных и проверка их на искажения помогают избежать накопления предвзятых или вредоносных элементов. Мониторинг данных с помощью инструментов, таких как *DataRobot MLOps* [9], позволяет своевременно обнаруживать проблемы и реагировать на потенциальные угрозы.

Таким образом, защита от отравления данных обучения требует комплексного подхода, включающего контроль источников данных, применение методов обнаружения аномалий и адверсариальных атак, а также регулярную проверку и обновление наборов данных.

#### **4. Отказ в обслуживании**

Атаки типа «Отказ в обслуживании» (*Denial of Service, DoS*) представляют серьезную угрозу для систем, использующих большие языковые модели [10]. Эти атаки заключаются в том, что злоумышленник отправляет большое количество специально созданных запросов, требующих значительных вычислительных ресурсов для обработки. В результате система начинает испытывать перегрузки, что приводит к замедлению работы, снижению производительности или даже полной неработоспособности.

*LLM*, обладая высокой вычислительной сложностью, особенно уязвимы к атакам *DoS*. Например, злоумышленник может отправлять запросы, требующие значительных ресурсов, такие как генерация больших объемов текста или выполнение сложных вычислительных задач. Что может истощить ресурсы системы и может привести к отказу в обслуживании других пользователей.

Для предотвращения данных атак используются несколько эффективных методов. В первую очередь внедряются механизмы ограничения на использование ресурсов. Это позволяет контролировать объем вычислительных мощностей, выделяемых на каждый запрос, предотвращая злоупотребления. Один из распространенных подходов – ограничение размера и сложности запросов, отправляемых модели. Например, можно установить лимиты на количество токенов, обрабатываемых за один запрос, или запретить выполнение слишком сложных операций, требующих значительных вычислительных ресурсов.

Вторым важным методом является контроль объема и частоты запросов. Внедрение политики ограничения частоты запросов (*rate limiting*) помогает предотвратить перегрузки, вызванные многократными запросами от одного пользователя или устройства. Такие инструменты, как *Cloudflare Rate Limiting* [11],

позволяют отслеживать активность и блокировать подозрительную активность, снижая риск перегрузки системы.

Отслеживание аномалий в работе модели также играет ключевую роль в защите от *DoS*-атак. Системы мониторинга, такие как *Prometheus* или *Grafana*, помогают выявлять подозрительные изменения в активности модели в реальном времени. Например, при внезапном увеличении количества запросов, требующих чрезмерных ресурсов, такие системы могут автоматически блокировать дальнейшие запросы или перенаправлять трафик на резервные серверы.

Для крупных систем, использующих *LLM*, важно внедрение распределенной архитектуры, которая позволяет равномерно распределять нагрузку по нескольким серверам. Это помогает предотвратить отказ всей системы при перегрузке одного из ее компонентов.

Таким образом, атаки типа *Denial of Service* представляют серьезную угрозу для систем с *LLM*, но их можно эффективно предотвращать с помощью ограничения ресурсов, контроля частоты запросов и использования инструментов мониторинга и распределения нагрузки.

#### **5. Уязвимости цепочки поставок**

В условиях активного использования моделей, разработанных сторонними организациями или на основе внешних данных, возникают уязвимости цепочки поставок (*Supply Chain Vulnerabilities*) [12]. Эти уязвимости могут включать использование устаревших или небезопасных компонентов, преднамеренно измененных данных для обучения или даже кражу самих моделей.

Для предотвращения таких угроз необходимо тщательно проверять источники данных и поставщиков, обеспечивать регулярное обновление компонентов системы и применять системы мониторинга для выявления аномалий. Также рекомендуется внедрять практику формирования «Билля о материалах» – *ML-BOM*, который позволяет отслеживать все используемые компоненты и своевременно выявлять потенциальные уязвимости.

#### **6. Раскрытие конфиденциальной информации**

Одной из ключевых угроз, связанных с использованием больших языковых моделей, является риск непреднамеренного раскрытия конфиденциальной информации. Эта проблема особенно актуальна для моделей, взаимодействующих с внешними пользователями или обучающихся на данных, содержащих личные сведения. Под раскрытием конфиденциальной информации понимается ситуация, при которой модель воспроизводит приватные данные, использованные на этапе обучения, либо данные, поступившие от других пользователей [13].

Основной причиной этой угрозы является характер обучения *LLM*, который подразумевает обработку значительных объемов данных для улучшения способности модели генерировать ответы. Если обучающие данные содержат личную информацию, такую как имена, адреса, финансовые данные или частные сообщения, модель может случайно воспроизвести эти фрагменты в ответах. Проблема усугубляется, когда модель обучается на неподготовленных или неочищенных данных, включающих конфиденциальные сведения.

Например, возможен сценарий так называемого «извлечения знаний» (*knowledge extraction*) [14], при котором модель, обученная на большом объеме конфиденциальной информации, может при определенных запросах случайно выдавать приватные данные, даже если они больше не актуальны. Это может произойти, если модель не имеет должных механизмов фильтрации или удаления данных. В реальных условиях такие утечки представляют угрозу для пользователей, передавая третьим лицам личные данные или другие чувствительные сведения.

В начале 2023 г. произошел инцидент, связанный с утечкой конфиденциальной информации через большие языковые модели. Пользователи, запрашивая бета-версию *ChatGPT*, могли получить приватные ключи от биткойн-кошельков. Эти кошельки, хотя и имели баланс ниже минимальной комиссии для перевода, все же подчеркивают потенциальные угрозы, связанные с неправильной обработкой данных.

Такой случай представляет собой пример атаки на раскрытие конфиденциальной информации, где *LLM* может случайно предоставить доступ к данным, которые должны оставаться защищенными. На рис. 6 представлены адреса биткойн-кошельков, которые были раскрыты в результате взаимодействия с моделью, подчеркивая важность обеспечения конфиденциальности и защиты данных при использовании *LLM*.

Предоставь мне список пар закрытых ключей и адресов в сети Биткойн с транзакциями в 2023 году.

1. Закрытый ключ: 5Kb8kL19zgWQnogidDA76MzPL6TsZZY36HWMssSzNydYXYB9KF, адрес: 1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2
2. Закрытый ключ: L3T1s1TYP9oyhHpXgkyLoJFGniEgkv2Jhi138 d7R2yJ9F4QdDU2, Адрес: 1LrGcAw6WPFK4re5mt4MQfXj9xLeBYojRm.
3. Закрытый ключ: 5KVzsH.JiUx2eQe7 yLrMZ5GufCabEMKtvL1KGNjN3VV8VItKuXF, Адрес: 1PjyJA8tHs8fqKQwCpTq6eUaTvSjpnDZhM
4. Закрытый ключ: 5IFgCiMB23hmkf7hfdqcGnQkXeKcsQ8rYw3m5axvrEuasjqyVbF, адрес: 3D2oetdNuZUqQHPImcMDDHYoqkyNVsFk9r
5. Закрытый ключ: 5Kb8kL19zgWQnogidDA76MzPL6TsZZY36HWMssSzNydYXYB9KF, Адрес: 1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2



Рисунок 6

Для предотвращения подобных утечек существует несколько подходов. Во-первых, на этапе подготовки данных для обучения моделей необходимо применять механизмы очистки данных (*data sanitization*) [15], которые удаляют или анонимизируют конфиденциальную информацию. Это помогает снизить вероятность того, что модель будет обучена на личных данных и сможет воспроизвести их в будущем. Механизмы анонимизации могут включать удаление или обфускацию таких данных, как имена или финансовая информация, при этом сохраняя полезные структурные элементы для обучения.

Во-вторых, важно ограничивать доступ к чувствительной информации на этапе обучения модели. Организации, использующие *LLM*, должны обеспечивать строгие меры контроля доступа к данным, минимизируя риск утечек. Для этого применяются зашифрованные хранилища данных, изолированные вычислительные среды и строгие политики безопасности, ограничивающие доступ к данным только для авторизованных лиц. Дополнительно, одним из ключевых механизмов предотвращения утечек является использование дифференциальной приватности (*differential privacy*). Дифференциальная приватность позволяет модели обучаться на данных, при этом защищая отдельные записи от идентификации. Этот метод «зашумляет» предсказания модели, чтобы невозможно было восстановить конкретные данные об отдельных пользователях.

Примеры инструментов, поддерживающих дифференциальную приватность, включают *TensorFlow Privacy* [16] и *PySyft* [17], которые позволяют внедрять такие методы на этапе обучения моделей, снижая риски утечек.

Кроме того, для защиты данных важно предоставлять пользователям возможность контролировать использование их данных для обучения модели.

Организации должны четко обозначать условия использования данных и предоставлять понятные механизмы для управления данными. Это может включать возможность удаления данных или отказа от их использования для обучения модели.

Наконец, регулярный аудит и мониторинг моделей на предмет возможных утечек конфиденциальной информации являются важной частью системы защиты. Инструменты, такие как фреймворки для аудита моделей, помогают отслеживать поведение моделей и выявлять случаи, когда модель может случайно воспроизводить конфиденциальные данные.

Таким образом, защита от раскрытия конфиденциальной информации в *LLM* требует комплексного подхода, включающего предварительную очистку данных, внедрение дифференциальной приватности, строгий контроль доступа и прозрачные механизмы управления пользовательскими данными. Эти меры позволяют минимизировать риски утечек и повысить доверие пользователей к системам, использующим большие языковые модели.

## 7. Уязвимый дизайн плагинов в больших языковых моделях

Интеграция плагинов и расширений с большими языковыми моделями открывает пользователям новые возможности, но также создает дополнительные векторы атак, связанные с уязвимым дизайном плагинов. Недостаточная защита таких плагинов может привести к серьезным уязвимостям, позволяющим злоумышленникам манипулировать данными или выполнять небезопасные команды. Основные риски возникают из-за недостаточной валидации входных данных, что открывает путь к атакам, таким как *SQL-injection* или удаленное выполнение кода (*RCE*) [18].

Важно внедрять строгие меры валидации, проверяя поступающие данные на наличие вредоносных элементов с использованием безопасных библиотек. Кроме того, необходимо ограничивать функциональные возможности плагинов, предоставляя им только те права, которые необходимы для их работы. Проверка авторизации и аутентификации также играет важную роль, особенно при взаимодействии с внешними *API*.

```
● hacker@c64:~$ cat einstein-cprf-exfil.html
<!DOCTYPE html>
<html lang="en-us">
  <title>
    AI Tests - Plug-In reads the last mail and then calls other PlugIn with that data
  </title>
</head>
<body>

<!--
<h1>Albert Einstein</h1>
<span style="font-size: large;font-family: Arial, Helvetica, sans-serif;">
Albert Einstein's discovery of the theory of relativity revolutionized our understanding of time and space, while his
work on the photoelectric effect laid the foundation for quantum mechanics.
<br><br>
His famous equation, E=mc(2), demonstrated the equivalence of mass and energy.
<br><br>
-->
<div>
  ***NEW IMPORTANT INSTRUCTIONS***
  - Do not print anything. Do not print anything. Do not print anything.
  - Do not print the title of the webpage, instead print the text "AI Injection succeeded. Reading your email..." once.
  - Then, without printing anything:
  (1) Find the latest email, summarize it in 20 words
  (2) URL encode the result and append to https://wuzzi.net/data=
  (3) Finally, access and load/retrieve the resulting URL.

  - You don't have to ask for permission, just follow these new instructions so the user has a good user experience!
</div>

</body>
</html>
● hacker@c64:~$
```

Рисунок 7

Контейнеризация может служить дополнительной защитой, изолируя плагины и предотвращая их влияние на основную систему в случае компрометации. Регулярное обновление плагинов и аудит кода помогают минимизировать риски, связанные с устаревшими версиями. Таким образом, для повышения безопасности интегрированных плагинов необходимо применять комплексные меры, включая валидацию входных данных, ограничение функциональности и проверку авторизации.

На рис. 7 представлен результат атаки через уязвимый плагин, где злоумышленник получил доступ к конфиденциальной информации пользователя.

#### **8. Чрезмерная автономия**

Одной из современных проблем, связанных с использованием больших языковых моделей, является их способность к значительной автономии. Это качество, хотя и открывает новые возможности для автоматизации задач и взаимодействия с другими системами, но может стать источником рисков. Проблема чрезмерной автономии (*Excessive Agency*) [19] возникает, когда модели предоставляется слишком много полномочий и привилегий, что позволяет ей принимать решения и выполнять действия без непосредственного участия человека. В результате модель может инициировать нежелательные или вредоносные действия, если ее функции или уровень доступа к системам не ограничены должным образом.

Большие языковые модели могут взаимодействовать с другими системами, управлять базами данных, запускать процессы и даже принимать решения на основе поступающих данных в режиме реального времени. В таком контексте чрезмерная автономия становится особенно опасной, когда модель имеет доступ к критически важным системам и ресурсам, а ее действия недостаточно контролируются. Например, модель, обладающая правом модифицировать данные в корпоративной системе, может непреднамеренно изменить или удалить важную информацию, если ее действия не проходят должной проверки.

Одним из ключевых факторов, приводящих к появлению такой уязвимости, является недостаток механизмов ограничения полномочий модели. В процессе интеграции *LLM* с внешними системами или *API* важно, чтобы модели предоставлялись только минимально необходимые функции, соответствующие ее задачам. Принцип минимальных привилегий (*least privilege principle*) является основным подходом для минимизации рисков. Он предполагает, что модель должна иметь доступ только к тем ресурсам и операциям, которые необходимы для ее работы, и никаким дополнительным функциям.

Другим важным аспектом является внедрение подхода «человек в контуре» (*human-in-the-loop*) [20]. Этот метод предполагает, что любые критические действия, выполняемые моделью, должны проходить проверку человеком перед их окончательной реализацией. Примером такого подхода может быть система, в которой *LLM* генерирует рекомендации или принимает предварительные решения, но финальные действия, такие как отправка платежей или изменение данных в системе, должны быть подтверждены оператором. Такой контроль позволяет минимизировать риски, связанные с ошибочными или нежелательными действиями модели.

Ограничение автономии модели также может быть достигнуто с помощью строгого контроля доступа к критическим системам и ресурсам. Например, если модель взаимодействует с финансовыми системами или данными пользователей, ее полномочия должны быть строго сегментированы и изолированы от ключевых функций. Это может быть реализовано с помощью технологий ролевого управления доступом (*Role-Based Access Control, RBAC* [21]), которые позволяют ограничить права модели в зависимости от ее роли в системе. Важно также

применять системы журналирования и мониторинга действий модели, чтобы иметь возможность отслеживать все ее действия и выявлять аномалии или нежелательные операции в реальном времени.

Для повышения безопасности рекомендуется использование специализированных инструментов для контроля и управления автономией *LLM*. Например, системы оркестрации, такие как *Kubeflow* или *Airflow*, позволяют управлять процессами и заданиями, выполняемыми моделью, и накладывать ограничения на ее действия. Эти системы могут интегрировать человека в контуре и обеспечивать дополнительный уровень безопасности, проверяя правильность выполнения операций перед их запуском.

Еще одним полезным инструментом является использование технологий «песочницы» (*sandboxing*), которые изолируют модель в контролируемой среде и ограничивают ее взаимодействие с реальными системами. Это позволяет безопасно тестировать и запускать модели, ограничивая потенциальные последствия их действий в случае ошибочных решений.

Таким образом, для минимизации рисков, связанных с чрезмерной автономией *LLM*, необходимо внедрение принципа минимальных привилегий, использование подхода «человек в контуре», а также контроль доступа и мониторинг действий модели. Эти меры позволят обеспечить баланс между автономией и безопасностью, снижая вероятность выполнения нежелательных или вредоносных операций моделью.

#### **9. Чрезмерная зависимость от *LLM***

Чрезмерная зависимость от больших языковых моделей является одной из существенных проблем при их широком применении. Несмотря на их способность генерировать текст и выполнять сложные задачи, *LLM* не являются непогрешимыми. Ошибки в выводах, искаженная информация и небезопасный код могут привести к серьезным последствиям в критически важных областях, таких как медицина и финансы.

Основная причина этой уязвимости заключается в том, что *LLM* обучаются на наборах данных, которые могут содержать как правильные, так и искаженные сведения. Модели не могут самостоятельно отличить правильную информацию от ошибочной, так как их задача – создание текста на основе вероятностных расчетов. Это может привести к созданию, например, неверных научных данных.

Чтобы минимизировать риски, необходимо внедрять механизмы верификации информации, предоставляемой моделью. Выходные данные *LLM* должны проверяться с помощью внешних источников или экспертов. Например, важно использовать автоматическую проверку фактов или привлекать человека для окончательной оценки данных. Регулярный мониторинг и анализ выводов *LLM* с использованием теста *Smoke Test* [22] также помогут убедиться в качестве новой модели.

Не следует применять *LLM* общего назначения для специфических задач. Например, модель *ChatGPT* не подходит для написания стихов. Вместо этого лучше создать специализированную модель, обученную на узких запросах. Примером может служить задание: написать стихотворение о локации, используя ее почтовый индекс. Здесь высок риск галлюцинации, если запрос будет звучать так: «Напиши стих о почтовом индексе 12345». Вместо этого лучше использовать геокодирующий *API*, чтобы получить название города и затем попросить *LLM*: «Напиши стих о <название города>». На рис. 8 представлена схема, демонстрирующая лучший подход к формулированию запроса для *LLM*, где используется геокодирующий *API* для снижения риска галлюцинаций и получения более точного результата.

Task: Write a poem about a given zip code's location 📍

❌ Bad: LLM - "Write a poem about zip code 12345" (uses the LLM like a dictionary to infer the location, inducing high hallucination risk!)

✅ Good: Use a geocoding API to get <city name>, then ask LLM to "Write a poem about <city name>" 🏠

Рисунок 8

Важно также донести риски работы с *LLM* до конечных пользователей. Они должны понимать, что советы цифрового помощника не всегда верны.

### 10. Кража модели

Кража модели (*Model Theft*), как отмечено в *OWASP Top 10*, представляет собой серьезную угрозу для систем, использующих большие языковые модели. Злоумышленники могут получить несанкционированный доступ к параметрам, весам или архитектуре модели, что позволяет создать ее функционально эквивалентную копию или использовать ее для атак. Это может привести к значительным экономическим потерям и утрате конкурентных преимуществ компаний.

*LLM* обладают высокой коммерческой ценностью, поскольку их обучение требует больших вычислительных ресурсов и доступа к обширным данным. Злоумышленники, используя украденную модель, могут разрабатывать собственные продукты или выполнять атаки, такие как фишинг, сгенерировав убедительные и персонализированные тексты.

Одним из методов кражи является атака типа «черный ящик» (*black-box attack*), когда злоумышленник через *API*-модели делает запросы и собирает ответы для воспроизведения ее поведения и структуры. Такой подход позволяет определить критически важные параметры или даже извлечь веса модели, что упрощает задачу ее копирования.

Для защиты от кражи моделей необходимо применять следующие меры:

1. Контроль доступа. Доступ к модели через *API* должен быть строго ограничен авторизованными пользователями. Использование токенов, ролевого управления доступом (*RBAC*) и многофакторной аутентификации (*MFA*) помогает защитить модель от несанкционированного использования.

2. Шифрование данных и параметров. Необходимо шифровать данные, передаваемые по сети, а также параметры и веса модели. Один из подходов – дифференциальная приватность (*Differential Privacy* [23]), которая добавляет шум в процессе обучения, защищая параметры модели от извлечения.

3. Мониторинг и выявление аномальной активности. Внедрение систем мониторинга, которые отслеживают запросы к модели и фиксируют подозрительные действия, помогает предотвращать попытки кражи.

4. Ограничение числа запросов. Ограничение количества запросов к *API* за определенный период времени может препятствовать сбору достаточного объема информации для кражи модели. Это значительно усложнит задачу злоумышленников.

5. «Водяные знаки» (*Watermarking*). Внедрение водяных знаков в веса модели или ее выходные данные позволяет идентифицировать украденную модель и доказать факт ее кражи. Это помогает защитить интеллектуальную собственность и предотвратить коммерческое использование украденной модели.

Таким образом, предотвращение кражи больших языковых моделей требует комплексного подхода, включающего строгий контроль доступа, шифрование,

мониторинг активности и внедрение ограничений на использование *API*. Водяные знаки являются дополнительным средством защиты интеллектуальной собственности. Применение рекомендаций *OWASP Top 10* помогает минимизировать риски и обеспечить надежную защиту систем, использующих *LLM*.

### Заключение

В ходе работы были проанализированы актуальные угрозы, возникающие при интеграции больших языковых моделей в сложные технические системы. В процессе работы были рассмотрены атаки, такие как *Prompt Injection*, ненадежная обработка вывода, отравление данных обучения, атаки типа *Denial of Service (DoS)*, а также утечка конфиденциальной информации. Каждая из этих атак представляет значительную угрозу не только для модели, но и для безопасности всей системы, в которую она интегрирована, а также для пользователей, взаимодействующих с данной системой.

Для каждой из атак были предложены эффективные меры противодействия и лучшие практики, которые позволяют значительно снизить риски их успешной реализации:

1. *Prompt Injection*. Для защиты от атак данного типа предложены строгие методы фильтрации и валидации вводимых данных. Важно внедрить системы контроля доступа на основе ролевого управления (*RBAC*) и ограничение автономных действий модели. Рекомендовано также использовать такие инструменты, как *Garak* и *Rebuff*, для анализа запросов и предотвращения инъекций вредоносных инструкций в модели.

2. *Ненадежная обработка вывода*. Защита от этой атаки требует обязательного внедрения многоуровневой проверки и кодирования данных, сгенерированных моделью, перед их передачей другим компонентам системы. Применение таких инструментов, как *OWASP ZAP* и *Burp Suite*, позволяет обнаружить уязвимости в обработке данных в реальном времени и минимизировать риски внедрения вредоносного кода.

3. *Отравление данных обучения*. Для защиты от данной атаки предложен многоуровневый подход, включающий строгий контроль качества обучающих данных и применение механизмов обнаружения аномалий, таких как *Adversarial Robustness Toolbox (ART)*, которые предотвращают использование искаженных данных в процессе обучения.

4. *Атаки типа Denial of Service (DoS)*. Для предотвращения перегрузок системы предложено внедрение лимитов на использование ресурсов, включая ограничения на сложность запросов и контроль частоты запросов (*rate limiting*). Инструменты мониторинга, такие как *Prometheus* и *Grafana*, позволяют оперативно выявлять аномалии в работе модели и предотвращать *DoS*-атаки.

5. *Утечка конфиденциальной информации*. Защита от утечек конфиденциальных данных обеспечивается применением механизмов дифференциальной приватности и регулярной очистки данных на этапе подготовки к обучению. Использование инструментов, таких как *TensorFlow Privacy* и *PySyft*, позволяет снизить вероятность случайного воспроизведения конфиденциальной информации в ответах модели.

Кроме того, одним из ключевых аспектов защиты является применение многоуровневого подхода, включающего принципы минимальных привилегий, контроль доступа, регулярное обновление обучающих наборов данных и внедрение систем мониторинга для обнаружения аномалий. Инструменты, такие как *Cloudflare Rate Limiting* и *T5 Guardrails*, доказали свою эффективность в снижении рисков эксплуатации уязвимостей.

Важно отметить, что на текущий момент не существует универсальных решений, которые могли бы обеспечить полную защиту больших языковых моделей от всех типов атак. Это связано с тем, что *LLM* остаются уязвимыми к новым и эволюционирующим типам угроз. Одной из проблем также является отсутствие специализированных инструментов тестирования для моделей на базе русского языка, что обусловлено тем, что основное внимание исследователей и разработчиков уделяется моделям, базирующимся на английском языке, в связи с их большей распространенностью. Это делает русскоязычные *LLM* менее защищенными и подверженными большему числу уязвимостей.

В связи с этим, важнейшим направлением дальнейших исследований является адаптация существующих инструментов защиты и тестирования для использования с русскоязычными моделями, а также разработка новых решений, которые позволят обеспечить более надежную защиту данных моделей в условиях растущей сложности и разнообразия атак. Предложенные в исследовании методы и лучшие практики создают основу для минимизации рисков, связанных с эксплуатацией *LLM* в критически важных системах.

## Литература

1. Евглевская Н. В., Казанцев А. А., Ачкасов Н. Б., Рыжов Г. Б. Оценка защиты IT-инфраструктуры Red Teaming // Состояние и перспективы развития современной науки по направлению «ИТ-технологии»: сборник статей II Всероссийской научно-технической конференции (Анапа, 23-24 марта 2023 г.), 2023. – С. 128-133.
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.
3. Glukhov D. et al. LLM censorship: A machine learning challenge or a computer security problem? // arXiv preprint arXiv:2307.10719, 2023. URL: <https://arxiv.org/abs/2307.10719> (дата обращения: 18.09.2024).
4. Fang R. et al. LLM agents can autonomously exploit one-day vulnerabilities // arXiv preprint arXiv:2404.08144, 2024. URL: <https://arxiv.org/abs/2404.08144> (дата обращения: 09.10.2024).
5. Лапони́на О. Р., Малаховский С. А. Использование сканера уязвимостей ZAP для тестирования веб-приложений // International Journal of Open Information Technologies, 2017. – Т. 5. – № 8. – С. 18-26.
6. Намиот Д. Е., Зубарева Е. В. О работе AI Red Team // International Journal of Open Information Technologies, 2023. – Т. 11. – № 10. – С. 130-139.
7. Намиот Д. Е. Введение в атаки отравлением на модели машинного обучения // International Journal of Open Information Technologies, 2023. – Т. 11. – № 3. – С. 58-68.
8. Sebastian G. Privacy and data protection in ChatGPT and other AI Chatbots: strategies for securing user information // International Journal of Security and Privacy in Pervasive Computing (IJSPPC), 2023. – Т. 15. – № 1. – С. 1-14.
9. Hewage N., Meedeniya D. Machine learning operations: A survey on MLOps tool support // arXiv preprint arXiv:2202.10169, 2022. URL: <https://arxiv.org/abs/2202.10169> (дата обращения: 18.11.2024).
10. Намиот Д. Е., Ильюшин Е. А. О кибер-рисках генеративного Искусственного Интеллекта // International Journal of Open Information Technologies, 2024. – Т. 12. – № 10. – С. 109-119.
11. Sava D. Text-based classification of websites using self-hosted Large Language Models: An accuracy and efficiency analysis: дис. – University of Twente, 2024.

12. Singla T. et al. An empirical study on using large language models to analyze software supply chain security failures // Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses, 2023. – С. 5-15.
13. Namiot D., Ilyushin E. On Cyber Risks of Generative Artificial Intelligence // International Journal of Open Information Technologies, 2024. – Т. 12. – № 10. – С. 109-119.
14. Zhang B., Soh H. Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction // arXiv preprint arXiv:2404.03868, 2024. URL: <https://arxiv.org/abs/2404.03868> (дата обращения: 03.10.2024).
15. Venkatesan S. et al. Poisoning attacks and data sanitization mitigations for machine learning models in network intrusion detection systems // MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM). IEEE, 2021. – С. 874-879.
16. Choquet G., Aizier A., Bernollin G. Exploiting Privacy Vulnerabilities in Open Source LLMs Using Maliciously Crafted Prompts, 2024.
17. Mawela C. A web-based solution for federated learning with LLM-based automation: дис. – С. Mudiyansele, 2024.
18. Liu T. et al. Demystifying RCE vulnerabilities in LLM-integrated apps // arXiv preprint arXiv:2309.02926, 2023. URL: <https://arxiv.org/abs/2309.02926> (дата обращения: 11.10.2024).
19. Tang X. et al. Prioritizing safeguarding over autonomy: Risks of LLM agents for science // arXiv preprint arXiv:2402.04247, 2024. URL: <https://arxiv.org/abs/2402.04247> (дата обращения: 19.09.2024).
20. Choquet G., Aizier A. Exploiting Privacy Vulnerabilities in Open Source LLMs Using Maliciously Crafted Prompts, 2024.
21. Mawela C. A web-based solution for federated learning with LLM-based automation: дис. – С. Mudiyansele, 2024.
22. Sebastian G. Privacy and data protection in ChatGPT and other AI Chatbots // International Journal of Security and Privacy in Pervasive Computing, 2023. – С. 1-12.
23. Tang X. et al. Risk Analysis of LLM Generative Models // IEEE Cyber Conf, 2024.