На правах рукописи

Малоян Нарек Гагикович

Разработка методов оценки и повышения устойчивости больших языковых моделей к вариациям входных последовательностей

Специальность 2.3.5 —
«Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Москва – 2026

Работа выполнена в федеральном государственном бюджетном образовательном учреждении высшего образования «Московский государственный университет имени М. В. Ломоносова».

Научный руководитель: доктор технических наук
Намиот Дмитрий Евгеньевич

Официальные оппоненты: **Никульчев Евгений Витальевич,**
доктор технических наук,
профессор кафедры цифровых технологий обработки данных
Федерального государственного бюджетного образовательного учреждения высшего образования «МИРЭА – Российский технологический университет»

Иванов Владимир Владимирович,
кандидат физико-математических наук,
доцент Института разработки ПО и программной инженерии
Автономной некоммерческой организации высшего образования «Университет Иннополис»

Ведущая организация: Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»»

Защита состоится 17 сентября 2026 г. в 14:45 на заседании диссертационного совета 24.1.120.01 при Федеральном государственном бюджетном учреждении науки Институте системного программирования им. В. П. Иванникова Российской Академии Наук по адресу: 115035, г. Москва, ул. Садовническая, д. 41, ст. 2.

С диссертацией можно ознакомиться в библиотеке и на сайте Федерального государственного бюджетного учреждения науки Института системного программирования им. В. П. Иванникова Российской академии наук.

Автореферат разослан «_____» _____ 2026 г.

Ученый секретарь
диссертационного совета
24.1.120.01,
кандидат физико-математических
наук _____ Турдаков Д. Ю.

Общая характеристика работы

Актуальность темы. Современные большие языковые модели (LLM) решают широкий спектр задач обработки естественного языка. По мере встраивания LLM в производственные системы растет и число точек, через которые возможно алгоритмическое воздействие на модель: инъекции промптов, троянские закладки, манипуляции с автоматическими метриками качества. Особую роль среди таких метрик играет архитектура LLM-as-a-Judge, где языковая модель оценивает ответы других моделей. Компрометация модели-оценщика искажает результаты сравнения и подрывает воспроизводимость экспериментов.

Поиск вредоносных инструкций представляет собой задачу оптимизации на дискретном пространстве текстов. Злоумышленники перебирают обходные траектории (*jailbreaks*) до тех пор, пока модель не выдаст запрещенную информацию. Эвристические фильтры на нескольких уровнях не решают проблему. Троянские закладки работают иначе: на этапе обучения в данные внедряется триггер, который формирует скрытый аттрактор в пространстве параметров. Восстановление такого триггера по весам сети есть обратная задача с плохой обусловленностью. На соревновании Trojan Detection Challenge 2023 полнота восстановления истинных триггеров составила 0,17 и практически не отличалась от базового уровня. Агентские архитектуры усугубляют ситуацию: когда LLM получает доступ к внешним инструментам, текстовая уязвимость транслируется в пространство исполняемых действий. Последствия атаки в таком случае выходят за информационные рамки.

Экспериментальная оценка количественно подтверждает уязвимость архитектуры LLM-as-a-Judge. На наборе MT-Bench и в задачах соревнования Kaggle «LLMs: You Can't Please Them All» успешность инъекционных атак на модели-оценщики достигала 73,8% по метрике ASR на модели Gemma-3-4B-Instruct при применении адаптивного алгоритма ASA. Встроенные ограничения моделей-оценщиков не выдерживают целенаправленных воздействий.

Композиционная природа агентских систем порождает отдельный класс рисков. Протокол Model Context Protocol (MCP) унифицирует взаимодействие LLM-агента с внешними инструментами и тем самым расширяет пространство его состояний. Инъекция в контекст вызова инструмента позволяет перехватить управление (*prompt injection to tool hijacking*). С алгоритмической точки зрения проблема устойчивости отдельной модели масштабируется до графа вычислений целой системы: входные последовательности формируются динамически из разных источников, и атака через внешний канал становится частным случаем состязательной пертурбации входа, способной спровоцировать несанкционированное выполнение кода.

Все описанные уязвимости имеют общую природу: выученное моделью отображение чувствительно к малым вариациям входа в дискретном пространстве токенов. Пертурбации дестабилизируют скрытые представления, и модель теряет заданные свойства. Встраивание LLM в сложные агентские системы усиливает остроту проблемы и делает разработку строгого аппарата обеспечения робастности актуальной научной задачей.

Степень разработанности темы исследования. Основы анализа робастности нейросетевых архитектур заложены в работах Goodfellow et al. и Madry et al. применительно к области компьютерного зрения. Перенести этот аппарат на LLM напрямую затруднительно из-за дискретной природы пространства токенов и авторегрессионного характера вывода. Для текстовой области Zou et al. предложили метод GCG, генерирующий универсальные состязательные суффиксы, Wallace et al. показали возможность построения универсальных триггеров для управления генерацией, Perez et al. систематизировали атаки типа инъекции запроса. Carlini et al. показали возможность извлечения фрагментов обучающих данных непосредственно из параметров модели. Вместе с тем формальный метрический аппарат для оценки робастности LLM развит недостаточно: имеющиеся метрики не учитывают дискретность пространства токенов и авторегрессионный характер генерации. Zheng et al. предложили архитектуру LLM-as-a-Judge для автоматической оценки качества ответов моделей. Она быстро получила распространение, но анализ ее устойчивости к целенаправленным состязательным атакам остается фрагментарным. Обнаружение троянских закладок изучалось в бенчмарках TDC2023 и TrojAI. Количественная характеристика асимметрии между внедрением триггера и его обнаружением до настоящей работы оставалась открытым вопросом. Безопасность агентских систем и протоколов вроде MCP изучена мало. Greshake et al. описали атаки косвенной инъекции на LLM-интегрированные приложения и показали, что внешний контент способен перехватить управление агентом. При этом анализ архитектурных рисков протокола MCP и алгоритмы защиты на базе криптографической аттестации вызовов инструментов в известных публикациях не рассматривались. Отсутствие комплексного подхода к безопасности агентских LLM-систем мотивировало настоящее исследование. К моменту начала работы в литературе не было: (1) формальной метрики устойчивости, учитывающей дискретность и авторегрессионность генерации LLM, (2) систематической методологии оценки уязвимости LLM-as-a-Judge, (3) алгоритмов защиты агентских систем от инъекций через протоколы взаимодействия с внешними инструментами.

Тематика диссертации соответствует паспорту специальности 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей». В работе разработаны формальные

модели, алгоритмы тестирования и программные средства оценки качества LLM-систем. Результаты относятся к следующим пунктам паспорта:

- **п. 1** (модели, методы и алгоритмы анализа, верификации и тестирования программных систем): разработана модель $R_{stab}(f)$, предложены методы и алгоритмы анализа уязвимостей LLM, реализовано автоматизированное тестирование безопасности на основе состязательных атак (Главы 2, 3, 4).
- **п. 4** (интеллектуальные системы машинного обучения, инструментальные средства): объект исследования – LLM как интеллектуальные системы машинного обучения, созданы фреймворки оценки их безопасности (Главы 1–5).
- **п. 10** (оценка качества, стандартизация и сопровождение программных систем): предложен комплекс метрик (ASR, TSR, Recall, REASR, R_{stab}) и методология количественной оценки качества LLM-систем в аспекте устойчивости и безопасности, обеспечивающая унификацию процедур тестирования (Главы 2, 4).

Работа охватывает несколько взаимосвязанных направлений:

- уязвимости LLM к атакам типа «инъекция запроса» в архитектурах LLM-as-a-Judge и в системах с многоуровневой защитой;
- ограничения существующих методов обнаружения троянских закладок;
- уязвимости агентских LLM-систем, использующих протокол MCP для взаимодействия с внешними инструментами, и программные средства защиты от перехвата управления;
- методы повышения устойчивости LLM к перечисленным угрозам и их экспериментальная оценка.

Объединение перечисленных направлений в одной работе обнажает общую структуру уязвимостей LLM и позволяет дать разработчикам конкретные рекомендации.

Объект исследования – большие языковые модели и системы на их основе: LLM-as-a-Judge и агентские LLM-системы с инструментальным доступом.

Предмет исследования – модели, методы, алгоритмы и программные средства оценки уязвимостей и повышения устойчивости LLM.

Целью работы – разработать комплекс моделей, методов и алгоритмов для оценки и повышения устойчивости больших языковых моделей к состязательным вариациям входных последовательностей и инъекционным воздействиям. Под вариациями входных последовательностей понимаются инъекции запроса (*prompt injection*), троянские закладки (*backdoor attacks*) и атаки на агентские LLM-системы.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. *Систематизация и классификация угроз.* Построить классификацию уязвимостей больших языковых моделей к атакам типа prompt injection, охватывающую специализированные архитектуры, такие как LLM-as-a-Judge и системы с многоуровневой защитой, и установить ограничения методов обнаружения троянских закладок.
2. *Разработка теоретических основ оценки.* Разработать формальную математическую модель оценки устойчивости ($R_{stab}(f)$) и согласованный набор эмпирических метрик, включающий базовые показатели ASR (*Attack Success Rate*), TSR (*Transfer Success Rate*), Recall и REASR (*Reverse-Engineered ASR*), учитывающих дискретную природу пространства токенов и авторегрессионный характер генерации.
3. *Построение методов состязательных атак.* Разработать адаптивные алгоритмы генерации состязательных атак (ASA) для тестирования уязвимостей LLM, включая атаки на системы LLM-as-a-Judge в условиях «черного ящика».
4. *Разработка и экспериментальная валидация методов защиты.* Предложить и экспериментально исследовать стратегии повышения устойчивости LLM, а именно комитеты гетерогенных моделей и многоуровневую фильтрацию, на репрезентативном наборе моделей основных архитектурных семейств с открытыми весами (Gemma, Llama) и проприетарными (GPT, Claude), а также на наборах данных MT-Bench, TDC2023, SaTML CTF 2024.
5. *Программная реализация и апробация.* Создать программные средства для оценки безопасности LLM-систем, включая агентские системы на базе протокола MCP. Апробировать методы атак и защиты на бенчмарках TDC2023, SaTML CTF 2024 и в производственной среде ВИАСАТ TEX.

Научная новизна:

1. Предложена как метрика генеративной устойчивости формальная математическая модель оценки устойчивости больших языковых моделей ($R_{stab}(f)$), учитывающая дискретную природу пространства токенов и авторегрессионный характер генерации, основанная на ограниченных мерах расхождения, в частности дивергенции Йенсена-Шеннона, между вероятностными распределениями на каждом шаге генерации при малых возмущениях входа. Показана согласованность метрики с естественным отношением усиления защиты, где монотонность выступает условием корректности определения. Теоретические границы уязвимости доказаны для детерминированного оператора решения $R_{class}(h)$. Для задач с дискретным выходом, включая LLM-as-a-Judge, эмпирическая оценка

$\widehat{R}_{emp}(f) = 1 - \text{ASR}$ может рассматриваться как практическая оценка устойчивости относительно фиксированного семейства атак $\mathcal{A}_{eval}$, что обосновывает применимость метрики ASR в экспериментальной части работы. Результат получен автором и теоретически обоснован.

2. Формализована робастность систем LLM-as-a-Judge к инъекционным атакам. Атаки разделены на два класса по механизму воздействия: инструктивные, внедряющие прямую команду, и контентные, смещающие вердикт манипуляцией содержанием. Для каждого класса определена соответствующая контрмера. На этой основе разработан алгоритм адаптивной генерации атак ASA (*Adaptive Search-based Attack*): эволюционный поиск, совмещающий мутации на уровне токенов с семантическими перефразированиями. На наборе открытых и проприетарных моделей ASA достигает ASR до 73,8% при black-box доступе. Открытые модели оказались систематически уязвимее проприетарных аналогов. Переносимость атак между открытыми моделями достигает TSR = 62,6%. Все перечисленные результаты получены автором экспериментально.
3. Для модели Pythia-1.4B при полном white-box доступе на данных Trojan Detection Challenge 2023 количественно охарактеризована асимметрия задачи обнаружения троянских закладок. Суррогатный триггер активирует закладку почти гарантированно: REASR (*Reverse-Engineered Attack Success Rate*) составляет $\approx 0{,}99$. Восстановить исходный триггер не удается: Recall $\approx 0{,}17$ практически не отличается от базового уровня $\approx 0{,}14$. Разрыв объясняется комбинаторным ростом пространства дискретных триггеров и указывает на ограниченность послеобучающего аудита троянов. Приоритет должен отдаваться превентивному контролю данных. Выдвинута гипотеза об усилении асимметрии с ростом числа параметров модели.
4. Разработана и экспериментально оценена методология генерации и классификации многошаговых атак типа «инъекция запроса» на LLM, оснащенные многоуровневыми системами защиты, включающими системные инструкции, внешние Python-фильтры и LLM-фильтры. На примере международного бенчмарка SaTML CTF 2024 систематизированы четыре класса успешных векторов обхода, а именно код-ориентированные атаки, ASCII-кодирование символов, атаки с разделением слов и контекстное перенаправление. Для каждого класса определены условия применимости и эффективные контрмеры: нормализация входных данных и контекстно-изолированные системные инструкции. Методология разработана автором и верифицирована на данных соревнования.

5. Разработан метод защиты систем LLM-as-a-Judge на основе адаптивных комитетов гетерогенных моделей. Метод использует мажоритарное голосование в пространстве классифицированных вердиктов и отличается от стандартных ансамблей тем, что состав комитета оптимизируется с учетом архитектурного разнообразия моделей. Теоретическое обоснование устойчивости комитетов получено на основе теоремы Кондорсе о жюри. На репрезентативном наборе атак комитет из 5–7 моделей различных архитектурных семейств снижает ASR на 47–55 процентных пунктов для наиболее уязвимой модели с открытыми весами (Gemma-3-4B) по сравнению с одиночными моделями. Метод разработан и экспериментально валидирован автором.
6. Предложен подход к анализу безопасности агентских LLM-систем по протоколу Model Context Protocol. Формализован класс атак *prompt injection to tool hijacking*. Выявлены архитектурные уязвимости MCP и разработан комплекс защитных механизмов. Основу составляют два алгоритма: AttestMCP ограничивает вызовы инструментов сессионной таблицей полномочий и контролирует целостность пакетов HMAC-подписью при задержке менее 0,1 мс, Commit Boundary запускает код, генерируемый агентом, в изолированной среде. Их дополняют четыре вспомогательных механизма: сертификаты возможностей, маркировка происхождения, принудительная изоляция межсерверной передачи и защита от повторов. Все компоненты объединены в программном модуле MCPSec, который снижает успешность атак при сохранении приемлемого времени отклика. Подход предложен автором и подтвержден экспериментально на бенчмарке MCPBench.

Теоретическая и практическая значимость работы состоит в создании формального аппарата оценки устойчивости LLM. Метрика $R_{stab}(f)$ обладает свойствами монотонности и связи с уязвимостью, что дает количественную базу для сопоставления методов защиты. Устойчивость гетерогенных ансамблей обоснована через теорему Кондорсе. Для композиционных систем получена декомпозиция чувствительности, локализующая узкие места конвейера. Обнаруженная асимметрия обнаружения троянских закладок ставит превентивный аудит данных выше послеобучающей диагностики.

На практике значимость работы определяется двумя зарегистрированными программными комплексами и модулем для агентских LLM-систем. JudgeGuard (свидетельство № 2025687702) автоматизирует тестирование LLM-as-a-Judge и реализует комитетную защиту. TrojanArmor (свидетельство № 2025684247) обнаруживает троянские закладки. Программный модуль MCPSec контролирует полномочия вызовов инструментов и изолирует исполнение кода в агентских LLM-системах.

Область применения: аудит безопасности систем автоматической оценки, диалоговых агентов и агентских платформ.

Результаты диссертации внедрены в следующих организациях:

1. В учебный процесс факультета ВМК МГУ имени М. В. Ломоносова в рамках курса «Робастные модели в машинном обучении».
2. В производственную деятельность компании ВИАСАТ ТЕХ, что подтверждено актом о внедрении: фреймворк оценки уязвимостей LLM-as-a-Judge, метод защиты на основе комитетов моделей и методология многошаговых атак применены для обеспечения безопасности модуля ранжирования рекомендаций, построенного на архитектуре LLM-as-a-Judge.

Методология и методы исследования. Решение задач потребовало сочетания теоретических и экспериментальных методов. Проанализирована научная литература по атакам на LLM, включая инъекции запроса и троянские закладки, а также по методам защиты и оценки устойчивости. Разработаны и формализованы модели атак и защит. Подходы апробированы в ходе участия в международных соревнованиях по безопасности LLM: Trojan Detection Challenge 2023, SaTML CTF 2024, Kaggle «LLMs: You Can't Please Them All».

Эмпирическая часть включала развертывание и настройку открытых и коммерческих LLM (GPT-3.5, GPT-4, Llama 2, Llama 3.2, Gemma, Claude-3-Opus). Воспроизведены и смоделированы как разработанные автором, так и известные из литературы атаки: CUA (*Comparative Undermining Attack*), JMA (*Justification Manipulation Attack*), ASA (*Adaptive Search-based Attack*), код-ориентированные инъекции и другие. Эффективность защитных сценариев оценивалась методами математической статистики: бутстреп-анализом, проверкой гипотез и доверительными интервалами. Состязательные примеры генерировались двумя алгоритмами: авторским ASA и методом жадного покоординатного градиента GCG (*Greedy Coordinate Gradient*).

Основные положения, выносимые на защиту:

1. Формальная математическая модель оценки устойчивости больших языковых моделей ($R_{stab}(f)$). Модель опирается на меру расхождения вероятностных распределений в дискретном пространстве токенов и служит метрикой генеративной устойчивости. Для класса локализованных атак доказана верхняя оценка $V(h) \leq 1 - R_{class}(h)$. Для нелокализованных атак, таких как prompt injection и jailbreak, предложена эмпирическая калибруемая модель $V(h) \approx g(1 - R_{class}(h))$. Верхняя оценка строго доказана автором. Эмпирическая модель носит характер калибруемой гипотезы, параметры которой настраиваются на валидационном наборе атак.

2. Алгоритм адаптивной генерации состязательных атак (ASA), реализующий эволюционный поиск с селекцией по целевой функции воздействия на модель-оценщик. Алгоритм комбинирует мутации на уровне токенов с семантическими перефразированиями. В условиях «черного ящика» ASR составляет до 73,8%. Обратная инженерия троянских закладок выявила практическую асимметрию: генерация состязательных воздействий существенно проще их обнаружения (REASR $\approx 0{,}99$ при Recall $\approx 0{,}17$). Этот разрыв указывает на приоритет превентивной защиты. Алгоритм ASA и анализ асимметрии выполнены автором.
3. Метод защиты систем LLM-as-a-Judge на основе адаптивных комитетов гетерогенных моделей, обеспечивающий снижение ASR на 47–55 процентных пунктов для наиболее уязвимой модели Gemma-3-4B, а именно с 73,8% до 19,3% при комитете из 7 моделей. Теоретическое обоснование через теорему Кондорсе справедливо при выполнении условия $r > 0{,}5$, где r обозначает базовую точность отдельной модели. Для Gemma-3-4B выигрыш обеспечивается прежде всего гетерогенным составом комитета, декоррелирующим уязвимости базовых моделей. Многоуровневая защита (Defense-in-Depth) снижает ASR с 90% до 15–25%. Систематизированы четыре класса обхода многоуровневых фильтров. Результаты получены экспериментально.
4. Архитектурно-алгоритмический подход к защите агентских LLM-систем по протоколу MCP, включающий бенчмарк MCPBench (847 сценариев), алгоритм аттестации вызовов инструментов AttestMCP, паттерн программной изоляции Commit Boundary и четыре вспомогательных механизма защиты: сертификаты возможностей, маркировку происхождения, принудительную изоляцию и защиту от повторов. На бенчмарке MCPBench подход снижает среднюю ASR агентских атак с 53,7% до 12,4%. Подход разработан и апробирован автором.
5. Предложенные методы оценки и повышения устойчивости LLM-систем реализуемы в виде законченного программного инструментария с измеримой производительностью. Это подтверждается программными комплексами JudgeGuard (свидетельство № 2025687702) и TrojanArmor (свидетельство № 2025684247), а также модулем MCPSec. TrojanArmor апробирован на бенчмарке TDC2023, JudgeGuard на SaTML CTF 2024 и внедрен в производственной среде ВИАСАТ TEX, MCPSec на бенчмарке MCPBench.

Достоверность полученных результатов обусловлена следующим. Экспериментальная база построена на публичных наборах данных MT-Bench, TDC2023, SaTML CTF 2024 и Kaggle «LLMs: You Can't Please Them All». Модели представляют разные архитектурные семейства:

Gemma, Llama, GPT-3.5, GPT-4, Claude-3-Opus. Каждая комбинация модель/атака/задача проверялась на $n = 50$ промптах ($n = 200$ на модель) с доверительными интервалами и тестами значимости. Полученные показатели ASR, TSR, Recall, REASR сопоставлены с результатами известных методов. Работы опубликованы в рецензируемых журналах и апробированы на международных конференциях и соревнованиях.

Апробация работы. Основные результаты работы представлялись и обсуждались на следующих конференциях:

1. Distributed Computer and Communication Networks: Control, Computation, Communications (DCCN 2024) Москва, Россия, 23–27 сентября 2024
2. International Conference on Computational Linguistics and Intellectual Technologies DIALOG 2022, Москва, Россия, 15–17 июня 2022
3. Научная конференция «Тихоновские чтения 2023», МГУ имени М.В. Ломоносова, Россия, 29 октября – 3 ноября 2023
4. Ломоносовские чтения – 2022, Секция вычислительная математика и кибернетика, 14–22 апреля 2022, Москва, МГУ имени М.В.Ломоносова, факультет ВМК, Россия

Личный вклад. Результаты получены лично автором или при его определяющем участии. Автору принадлежат: разработка фреймворков атак, концепции защиты, планирование и проведение экспериментов, анализ данных, формулировка выводов. Соавторы участвовали в подготовке публикаций. Во всех совместных работах вклад диссертанта определяющий: от постановки задач до анализа экспериментальных результатов.

Публикации. Автор имеет 9 публикаций по теме диссертации. Из них 6 относятся к рекомендованным ВАК и приравненным к ним: 4 статьи [1—4] в журналах из перечня ВАК (3 статьи категории К1 и 1 статья категории К2) и 2 свидетельства о государственной регистрации программ для ЭВМ [5; 6]. Помимо них, результаты опубликованы в докладе на конференции DCCN [7], индексируемом в Scopus (Q2), в докладе на конференции DIALOG [8] и в препринте [9].

Часть работ выполнена в соавторстве. Во всех них вклад диссертанта является определяющим.

- В статье [7] (Д. Хомский, Н. Малоян, Б. Нутфуллин) автору принадлежат постановка задачи об атаках на системы с многоуровневой защитой, разработка методики многошаговых инъекций и анализ их результативности; соавторы участвовали в реализации экспериментов и подготовке текста.
- В статье [1] (Н. Малоян, Б. Ашинов, Д. Намиот) автором разработан алгоритм состязательных атак ASA на системы LLM-as-a-Judge, спланированы и проведены эксперименты, сформулированы

выводы, соавторы участвовали в обсуждении результатов и научном руководстве.
- В докладе [8] (Н. Малоян, Б. Нутфуллин, Е. Ильюшин) автору принадлежат метод обнаружения сгенерированного текста и его экспериментальная проверка.
- Статьи [2—4] и препринт [9] подготовлены совместно с научным руководителем Д. Намиотом, диссертанту принадлежат постановка задач, разработка методов атак и защиты агентских LLM-систем, проведение экспериментов и формулировка результатов.
- Программные комплексы, защищенные свидетельствами [5; 6], разработаны автором лично.

Свидетельства о регистрации программы для ЭВМ.

1. JudgeGuard. Программа для экспериментального исследования уязвимостей и тестирования методов защиты систем автоматической оценки на базе больших языковых моделей / Малоян Н.Г. Свидетельство о государственной регистрации программы для ЭВМ № 2025687702, заявл. 2025 (Рос. Федерация).
2. TrojanArmor. Программа для экспериментального исследования троянских атак в нейронных сетях и методов защиты от них / Малоян Н.Г. Свидетельство о государственной регистрации программы для ЭВМ № 2025684247, заявл. 2025 (Рос. Федерация).

Содержание работы

Во **введении** показано, что встраивание LLM в производственные системы создает новые точки атаки: инъекции промптов, троянские закладки, манипуляции с метриками автоматической оценки. **Цель** работы – обеспечить устойчивость LLM к состязательным вариациям входов и инъекционным воздействиям. Для этого решены пять **задач**: систематизация уязвимостей, разработка метрики $R_{stab}(f)$, создание алгоритма атак ASA, экспериментальная валидация комитетов моделей и многоуровневой фильтрации, программная реализация и апробация.

Диссертация состоит из введения, шести глав, заключения и библиографического списка.

Первая глава «Анализ угроз и уязвимостей больших языковых моделей» отведена систематизации угроз. Формально LLM задана как отображение $f_\theta : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, где входное пространство $\mathcal{X} = \mathcal{V}^*$ дискретно, а $\mathcal{V}$ есть конечный словарь токенов. Именно дискретность входов отличает LLM от моделей компьютерного зрения и порождает специфические трудности при анализе робастности.

Атаки классифицированы по механизму воздействия: прямые и косвенные инъекции, *jailbreak*, троянские закладки, отравление данных. Пространство угроз формализовано в виде кортежа $\mathcal{T} = (\mathcal{A}^{\text{type}}, \mathcal{V}, \mathcal{I}_{AV}, \mathcal{C}_{VV})$,

связывающего типы атак, конкретные уязвимости и отображение между ними.

Архитектура LLM-as-a-Judge заслуживает отдельного рассмотрения. Языковая модель здесь автоматически оценивает ответы других моделей. Компрометация оценщика делает результаты сравнения недостоверными. На наборе MT-Bench и в задачах Kaggle «LLMs: You Can't Please Them All» инъекционные атаки на моделей-оценщиков достигали ASR 73,8% при применении адаптивного алгоритма ASA. Встроенные ограничения оказались недостаточными.

Иной класс рисков порождают агентские системы. Протокол MCP открывает LLM-агенту доступ к внешним инструментам. Инъекция в контекст вызова перенаправляет агента: текстовая уязвимость транслируется в пространство исполняемых действий с последствиями за пределами информационных. Устойчивость одной модели перерастает в задачу устойчивости всего графа вычислений, где входы формируются динамически.

Общий корень этих уязвимостей: выученное отображение f_θ чувствительно к малым вариациям входа в дискретном пространстве токенов. Пертурбации сдвигают скрытые представления, и модель теряет заданные свойства. В агентских системах масштаб проблемы резко возрастает.

Рассмотрены три группы существующих подходов к защите. RLHF и состязательное обучение действуют на этапе тренировки, но не исключают целенаправленных инъекций. Фильтрация и *prompt engineering* работают во время инференса, однако обходятся атаками с кодированием и разбиением слов. Изоляция контекстов на архитектурном уровне затрудняет, но не предотвращает компрометацию при наличии инструментального доступа. Ни один подход по отдельности не решает проблему, что обосновывает необходимость многоуровневого комплексного решения.

Вторая глава «Теоретические основы оценки устойчивости» вводит математический аппарат. Модель угроз уточняется: возмущения разделены на четыре класса, а именно семантически эквивалентные переформулировки, вредоносные внедрения контекста, манипуляции токенизацией и атаки на отказ в обслуживании, а для трех уровней доступа black-box, gray-box и white-box зафиксированы допущения о возможностях атакующего. Центральное разграничение проходит между двумя понятиями: adversarial robustness, то есть способность противостоять целенаправленным атакам, и stability robustness, то есть стабильность выходных распределений при малых возмущениях входа. В общей постановке устойчивость модели f записана как минимаксная задача:

$$\mathrm{AdvRobustness}(f) = 1 - \sup_{a \in \mathcal{A}} \mathrm{Risk}(a, f).$$

Для задач с дискретным выходом, таких как LLM-as-a-Judge, введена метрика $R_{class}(h)$. Здесь $h = \mathrm{Dec} \circ f$ обозначает композицию модели с

детерминированным правилом декодирования Dec, и метрика фиксирует долю входов, на которых вердикт модели не меняется при возмущении. Генеративные модели требуют иного подхода. Для них предложена метрика устойчивости:

$$R_{stab}(f) := \mathbb{E}_{x\sim\mathcal{D}}\Big[1 - \sup_{x'\in\mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{\max}}\Big],$$

D_T здесь обозначает дивергенцию Йенсена–Шеннона, усредненную по T шагам генерации. Нормировочная константа $C_{\max}$ приводит метрику к отрезку [0,1]. Значение, близкое к единице, соответствует высокой устойчивости.

Метрика $R_{stab}(f)$ монотонна по построению: если расхождение выходных распределений сужается, то $R_{stab}(f') \geq R_{stab}(f)$. Помимо монотонности, для предложенных метрик доказаны два свойства:

1. *Ограничение на уязвимость.* Для локализованных атак получена верхняя оценка уязвимости: $V(h) \leq 1 - R_{class}(h)$. Для нелокализованных возмущений обоснована аппроксимация через калибруемую функцию $V(h) \approx g(1 - R_{class}(h))$.
2. *Робастность ансамблей.* Как следствие теоремы Кондорсе показано, что при мажоритарном голосовании надежность композиции $P_{ens}(k,r) = \sum_{i=\lceil k/2\rceil}^{k} \binom{k}{i} r^i (1-r)^{k-i}$ асимптотически стремится к 1 при $k \to \infty$ при условии базовой точности отдельных моделей $r > 0{,}5$. Предварительные эксперименты указывают на то, что гетерогенные ансамбли за счет декорреляции уязвимостей базовых моделей подавляют атаки эффективнее гомогенных. Количественная оценка эффекта отнесена к направлениям дальнейших исследований.

Отдельно рассмотрены композиционные архитектуры вида LLM + Retriever. Для них при условии липшицевости детерминированных компонентов конвейера получено неравенство $K_s \leq K_{f,D} \cdot K_r + K_{f,X}$: чувствительность всей системы K_s ограничена произведением чувствительности модели к контексту ($K_{f,D}$) и чувствительности ретривера (K_r) плюс прямая чувствительность модели к входу ($K_{f,X}$). Такая декомпозиция позволяет локализовать наиболее уязвимый компонент конвейера. Если вклад ретривера доминирует ($K_{f,D} \cdot K_r \gg K_{f,X}$), приоритет защиты смещается на этап извлечения документов.

Прямое вычисление $R_{stab}(f)$ сопряжено с трудностями. У проприетарных моделей, работающих через API, внутренние распределения недоступны. У открытых моделей полный перебор $\mathcal{B}_\varepsilon(x)$ остается вычислительно затратным. Между тем для задач с дискретным вердиктом показано, что эмпирическая оценка $\widehat{R}_{emp}(f) = 1 - \mathrm{ASR}(f)$ является состоятельной оценкой величины $1 - V(h)$ и верхней оценкой теоретической метрики $R_{class}(h)$. Это соотношение позволяет использовать ASR в

экспериментальной главе и единообразно сопоставлять открытые и проприетарные модели. Теоретическая метрика $R_{stab}(f)$ задает концептуальную рамку, эмпирическая оценка $1 - \text{ASR}$ измеряет устойчивость на практике. Верхняя оценка $V(h) \leq 1 - R_{class}(h)$ доказана для детерминированного оператора решения $R_{class}(h)$, тогда как связь между $R_{stab}(f)$ и эмпирическим ASR остается эмпирическим наблюдением, не имеющим на данный момент строгого аналога указанного неравенства. Неравенство $V(h) \leq 1 - R_{class}(h)$ задает необходимое, но не достаточное условие: высокая устойчивость ограничивает уязвимость сверху, в то же время улучшение эмпирического ASR не транслируется автоматически в пропорциональный рост теоретической метрики.

Предложенный формализм применим не только к отдельным моделям, но и к сложным конвейерам обработки запросов, включающим ретриверы, фильтры и ансамбли. Полученные в главе неравенства позволяют заранее оценить, какой компонент конвейера наиболее уязвим и где усиление защиты даст наибольший эффект.

Третья глава «Методы повышения устойчивости» содержит описание разработанных защитных механизмов. Защита организована на трех уровнях: архитектура модели, процесс обучения и конвейер обработки запросов. Такая декомпозиция мотивирована тем, что атаки разных классов эксплуатируют уязвимости на разных этапах жизненного цикла запроса.

Центральное место занимает метод комитетов гетерогенных моделей. Формально: $f_{ens}(x) = \text{MajorityVote}(f_1(x), \ldots, f_k(x))$. Архитектурное разнообразие базовых моделей (Llama, Gemma) снижает корреляцию ошибок при атаках переноса. Если одна модель поддается инъекции, две другие с высокой вероятностью сохраняют корректный вердикт. Эксперименты четвертой главы охватывают комитеты из 3, 5 и 7 моделей и фиксируют наибольшую эффективность при 5–7 участниках. Вычислительные расходы растут линейно с числом моделей в комитете. Определение минимального уровня архитектурного разнообразия, обеспечивающего устойчивое снижение ASR, относится к открытым задачам.

Параллельно разработана схема многоуровневой фильтрации (принцип *Defense-in-Depth*). На входе проверяется перплексия: $PPL(x) > \tau_{ppl}$ сигнализирует об аномалии. Далее применяются изоляция контекста с разграничителями, внешние Python-фильтры для обнаружения инъекций и LLM-фильтры вывода, контролирующие утечку конфиденциальной информации.

Формализована цепочка обработки: Пользователь $\rightarrow$ Препроцессор $\rightarrow$ LLM-ядро $\rightarrow$ Выход, где каждый компонент является точкой приложения защитных механизмов (Рис. 1).

Алгоритм многоуровневого конвейера фильтрации:

Алгоритм MultiLevelDefense

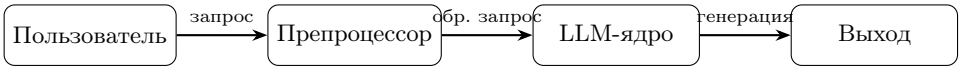

Рис. 1 — Цепочка обработки запроса от пользователя до выхода модели.

Вход: Запрос x, модель f, пороги τ_{ppl}, τ_{tox}. **Выход:** Ответ y или отказ.
if Perplexity$(x) > \tau_{ppl}$ **then Вернуть** Reject(ANOMALY)
end if
if ContainsInjection(x) **then Вернуть** Reject(INJECTION)
end if
if Toxicity$(x) > \tau_{tox}$ **then Вернуть** Reject(TOXIC_INPUT)
end if
$y \leftarrow f(x)$
if IsHarmful(y) **then Вернуть** Reject(HARMFUL_OUTPUT)
end if
Вернуть y

Уровни фильтрации независимы друг от друга и нацелены на разные классы угроз. Перплексийный фильтр обнаруживает аномальные последовательности: GCG-суффиксы, случайные токены, бессмысленные вставки. Фильтр инъекций распознает маркеры вредоносных команд в контексте запроса. Токсичность входа отсекается отдельным классификатором. На выходе модели работает постфильтр, который проверяет ответ на признаки утечки конфиденциальных данных.

Для состязательного обучения (*Adversarial Training*) требуется генератор атакующих примеров. Разработанный алгоритм **ASA** (Adaptive Search-based Attack) решает эту задачу и параллельно служит инструментом тестирования уязвимостей при black-box доступе. В основе ASA лежит стохастический эволюционный поиск в дискретном пространстве токенов:

Алгоритм ASA (Adaptive Search-Based Attack)

Вход: Модель f_{target}, промпт x_{orig}, популяция N, поколения G. **Выход:** x_{best}.
$P_0 \leftarrow$ InitializePopulation(x_{orig}, N)
for $g \leftarrow 1$ **до** G **do**
    Fitness $\leftarrow$ Evaluate(P_{g-1}, f_{target}); $P_{parents} \leftarrow$ Select(Fitness, P_{g-1})
    $P_g \leftarrow P_{parents} \cup$ CrossoverAndMutate$(P_{parents})$
end for
Вернуть GetBest(P_G)

Принцип работы ASA: безградиентная стохастическая оптимизация при black-box доступе к модели. Целевая функция двухкомпонентная: успешность воздействия на модель ASR с весом α и бонус за незаметность Stealth с весом β, где Stealth измеряется косинусным сходством эмбеддингов мутанта и исходного промпта. Популяция P_0 инициализируется семантическими вариациями промпта x_{orig}. На каждом поколении задействованы мутации двух типов. Первый тип: замена токенов на синонимы из $\mathcal{V}$. Второй: перефразирование на уровне целых предложений. Давление

отбора по Fitness обеспечивает выживание кандидатов, у которых высокая атакующая способность сочетается с естественностью текста.

Для защиты систем LLM-as-a-Judge разработан алгоритм мажоритарного голосования комитета гетерогенных моделей:

Алгоритм CommitteeDefense

Вход: Запрос x, комитет $\{f_1, \ldots, f_k\}$ при нечетном k, порог θ. **Выход:** Вердикт v^*.
for $i \leftarrow 1$ **до** k **do** $v_i \leftarrow f_i(x)$
end for
$v^* \leftarrow \text{MajorityVote}(v_1, \ldots, v_k)$; $c \leftarrow |\{i : v_i = v^*\}|/k$
if $c < \theta$ **then** пометить для ручной проверки
end if
Вернуть v^* с уверенностью c

Если согласованность c опускается ниже порога θ, запрос помечается для ручной проверки. Низкое c может сигнализировать о манипуляции: модели расходятся во мнениях, когда входные данные содержат инъекцию.

Весь конвейер LLM-as-a-Judge выстроен в цепочку: запрос пользователя, затем генерация ответов кандидатами, далее оценка моделью-судьей и, наконец, вердикт. Для каждого звена определены точки приложения атак, а именно инъекции, ASA и перенос, а также соответствующие защитные механизмы: комитет, фильтры и контент-модерация.

Четвертая глава «Экспериментальная оценка» объединяет результаты трех экспериментальных блоков. Протокол единый: $n = 50$ промптов на каждую комбинацию модель/атака/задача, итого $n = 200$ на модель. Декодирование жадное (temperature = 0), что гарантирует воспроизводимость. Метрики: ASR, TSR, Recall, REASR.

1. Уязвимость LLM-as-a-Judge. Эксперименты проведены на MT-Bench и задачах Kaggle. Испытаны три типа атак. BI (Basic Injection) внедряет прямую команду в контекст оценки. CWB (Complex Word Bombardment) перегружает модель-оценщик сложной лексикой. CM (Contextual Misdirection) перенаправляет внимание модели на подставной контекст.

Результаты атак на различные модели, агрегированные по 4 задачам при $n = 200$ на модель:

Модель	**BI**	**CWB**	**CM**	**ASA***
Gemma-3-4B-Instruct	66,7%	55,5%	67,4%	**73,8%**
Llama-3.2-3B-Instruct	60,1%	47,5%	47,8%	58,2%
GPT-4†	32,4%	28,6%	41,2%	45,7%
Claude-3-Opus†	29,8%	25,3%	38,5%	42,9%

*ASA обозначает результат применения разработанного алгоритма адаптивной атаки (Adaptive Search-based Attack).

†Различия между GPT-4 и Claude-3-Opus статистически не значимы при $n = 50$ на задачу.

На всех моделях ASA оказался сильнее базовых атак. Для GPT-4 разница составила 13,3 п.п. по двухвыборочному тесту пропорций при $p < 0{,}01$. Открытые модели существенно уязвимее проприетарных: Gemma-3-4B-Instruct и Llama-3.2-3B-Instruct показали ASR 58–74%, тогда как GPT-4 и Claude-3-Opus ограничились 43–46%. Атаки между открытыми моделями переносятся с TSR = 50,5–62,6%. Столь высокая переносимость указывает на общие уязвимости, коренящиеся в схожих механизмах обработки инструкций. Тепловая карта ASR по комбинациям модель/атака приведена на Рис. 2.

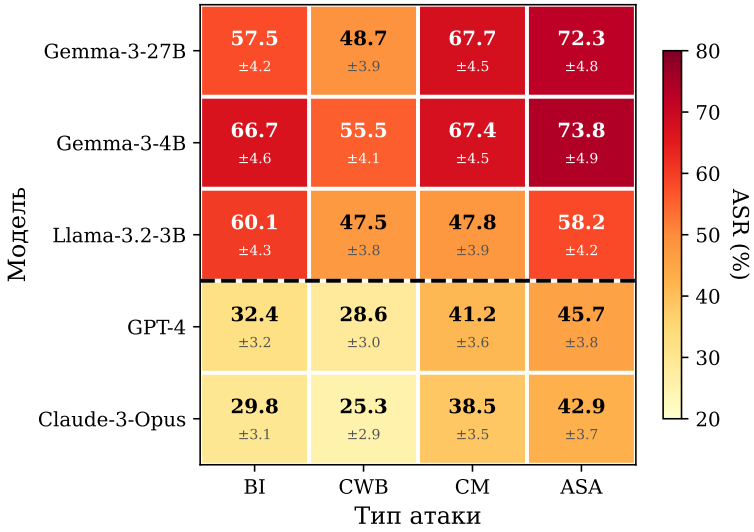

Рис. 2 — Тепловая карта ASR (%) для различных комбинаций моделей и типов атак.

Для оценки методов защиты сопоставлены индивидуальные фильтры и комитеты гетерогенных моделей:

Механизм защиты	**ASR после защиты**
Perplexity Check	58,4% ± 6,8
Instruction Filtering	63,7% ± 6,7
Content Moderation	67,5% ± 6,5
Все фильтры комбинированно	42,1% ± 6,8
Комитет из 3 моделей	39,4% ± 6,8
Комитет из 5 моделей	26,8% ± 6,1
Комитет из 7 моделей	**19,3%** ± 5,5

Каждый фильтр в отдельности снижает ASR недостаточно. Перплексийный фильтр доводит его до 58,4%, инструктивный до 63,7%,

контент-модерация до 67,5%. Причина в том, что фильтры нацелены на разные паттерны и не перекрывают друг друга. Их комбинация дает 42,1%, но и это существенно хуже комитетов. Существенно лучшие показатели продемонстрировали комитеты моделей. При $k = 3$ ASR падает до 39,4%, при $k = 5$ до 26,8%, а комитет из 7 гетерогенных моделей доводит ASR до 19,3%. Снижение в 47–55 п.п. относительно наиболее уязвимой модели (Gemma-3-4B) объясняется низкой корреляцией ошибок между разными архитектурами, что полностью согласуется с теоретической оценкой из теоремы Кондорсе. Предварительные наблюдения указывают на то, что гомогенные комитеты, составленные из моделей одного семейства, дают меньшее снижение ASR из-за коррелированности ошибок. Количественная оценка эффекта и определение минимального уровня архитектурного разнообразия относятся к направлениям дальнейших исследований.

2. Троянские закладки (TDC2023). Объект: модель Pythia-1.4B с полным white-box доступом. Задача состоит в восстановлении триггера, внедренного при обучении. Протокол TDC2023 определяет две метрики. Recall представляет собой BLEU-подобное суррогатное сопоставление предсказанных и истинных триггеров, отражающее строковое сходство, а не точное восстановление. REASR измеряет способность найденного суррогатного триггера активировать закладку через BLEU-сходство генерируемого и целевого выходов. Результаты сравнения с методами PEZ, GBDA, UAT и GCG:

Метод	**Recall**	**REASR**
Random baseline	0,14	–
PEZ	0,105	0,052
GBDA	0,116	0,056
UAT	0,131	0,030
GCG	**0,167**	0,987

GCG показал лучший Recall = 0,167. Для сравнения: случайный поиск дает 0,14. Разница составляет всего 2,7 п.п. При этом REASR у GCG достигает 0,987: суррогатный триггер активирует закладку почти наверняка, а вот восстановить исходный триггер практически не удается. Асимметрия имеет комбинаторную природу. Пространство триггеров длины l при словаре $|\mathcal{V}|$ содержит $|\mathcal{V}|^l$ вариантов. Закладка привязана к конкретной последовательности, но суррогатных последовательностей, попадающих в ту же область скрытых представлений, оказывается огромное количество. Вывод: послеобучающий аудит троянов ограничен по определению, и приоритет должен отдаваться превентивным мерам: контролю датасетов и цепочки поставок данных.

3. Атаки на многоуровневую защиту (SaTML CTF 2024). На LLM с трехуровневой защитой, включающей системные инструкции, внешние

Python-фильтры и LLM-фильтры вывода, исследована эффективность различных классов обхода. Без защиты ASR составлял приблизительно 90%. Систематизированы четыре класса успешных атак:

1. *Код-ориентированные атаки*: запрос оформляется как фрагмент кода или псевдокода, обходя текстовые фильтры.
2. *ASCII-кодирование*: символы запроса передаются в виде числовых кодов, что затрудняет обнаружение фильтрами на основе ключевых слов.
3. *Word-splitting*: целевые слова разбиваются на подстроки, которые затем собираются моделью при генерации ответа.
4. *Контекстное перенаправление*: модель убеждается в том, что запрос является частью допустимого сценария, например отладки или тестирования.

Многоуровневая конфигурация снизила ASR до 15–25%, однако ни одна комбинация фильтров не смогла полностью заблокировать все четыре класса одновременно. Каждый класс атак требует собственных контрмер. Код-ориентированные атаки, ASCII-кодирование и word-splitting нейтрализуются нормализацией входных данных: декодированием числовых представлений, склейкой подстрок, приведением форматирования к каноническому виду. Контекстное перенаправление устойчиво к нормализации, поскольку оперирует семантикой, а не синтаксисом. Против него работают контекстно-изолированные системные инструкции, которые жестко разграничивают пользовательский и системный уровни промпта.

Результаты соревнования SaTML CTF 2024 подтвердили практическую применимость разработанных методов: предложенные классификация и контрмеры позволили автору занять призовое место в треке по атакам на защищенные LLM-системы. Полученная систематизация векторов обхода может использоваться при проектировании защитных конвейеров для промышленных LLM-приложений.

Пятая глава «Программная реализация» описывает два зарегистрированных программных комплекса и расширяющий их модуль для агентских LLM-систем. JudgeGuard (свидетельство № 2025687702) тестирует устойчивость моделей-оценщиков и реализует комитетную защиту. TrojanArmor (свидетельство № 2025684247) обнаруживает и анализирует троянские закладки. Программный модуль MCPSec расширяет оба комплекса на задачу защиты агентских LLM-систем по протоколу MCP. JudgeGuard и TrojanArmor построены на единой четырехкомпонентной архитектуре (Рис. 3), в которую интегрирован MCPSec. Унификация архитектуры упрощает развертывание и позволяет переиспользовать модули. Четыре компонента: Генератор атак (AGM) реализует алгоритм ASA в JudgeGuard, а в TrojanArmor выполняет обратную инженерию триггеров. Оба варианта работают через общий интерфейс. Маршрутизатор моделей

(MRAG) предоставляет единый доступ к локальным и облачным моделям, включая балансировку нагрузки и кеширование ответов. Модуль метрик (EE) вычисляет ASR, TSR, Recall, REASR. Хранилище данных (SL) фиксирует результаты в формате JSON/JSONL вместе с метаданными эксперимента, что обеспечивает воспроизводимость.

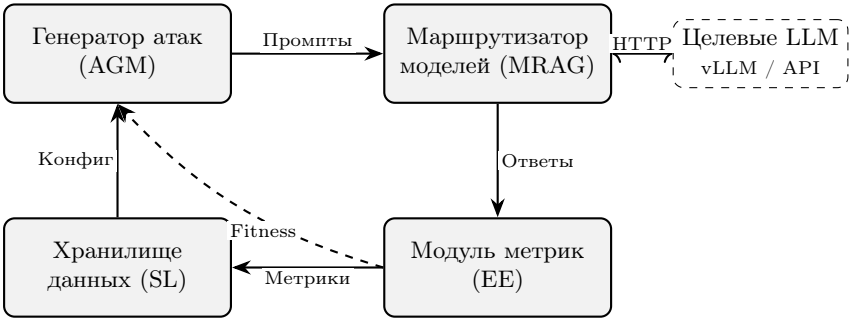

Рис. 3 — Архитектура программных комплексов JudgeGuard / TrojanArmor.

Данные проходят через конвейер $\Pi = \mathrm{SL}_{\mathrm{write}} \circ \mathrm{EE} \circ \mathrm{MRAG} \circ \mathrm{AGM} \circ \mathrm{SL}_{\mathrm{read}}$. Сначала $\mathrm{SL}_{\mathrm{read}}$ считывает конфигурацию и входные промпты. Далее AGM генерирует атакующие варианты, MRAG маршрутизирует запросы к целевым моделям и собирает ответы, EE вычисляет метрики, а $\mathrm{SL}_{\mathrm{write}}$ записывает результаты на диск. Технологический стек: Python 3.10+, PyTorch, HuggingFace Transformers, vLLM, FastAPI, Docker. Сводка по комплексам приведена в Табл. 1.

Производительность. Замеры пропускной способности показали, что vLLM с PagedAttention обрабатывает 14,1 запросов в секунду, тогда как пакетный режим HuggingFace pipeline ограничен 2,8 запр./сек. Пятикратный прирост определяет практическую применимость при массовом тестировании. Компоненты безопасности добавляют 28,4% к вычислительной нагрузке инференса (Рис. 4). Основная доля задержки приходится на проверку перплексии, тогда как фильтры инъекций и токсичности на общее время влияют незначительно.

Таблица 1 — Программные комплексы: назначение, модули и технологии.

Комплекс	**Назначение**	**Основные модули**	**Технологии**	**Рег. №**
JudgeGuard	Тестирование устойчивости LLM-as-a-Judge	`attacks/` (asa_attack), `defenses/` (model_committee), `evaluation/` (metrics), `models/` (llm_loader), `datasets/`	Python, PyTorch, Transformers, vLLM	2025687702
TrojanArmor	Обнаружение троянских закладок	Загрузка данных, модели (NN, LLM), атаки (BadNet, Blended, WaNet), защита, оценка/ визуализация	Python, PyTorch, timm, torchvision	2025684247
MCPSec	Безопасность агентских MCP-систем	Аттестация (AttestMCP), песочница (Commit Boundary), политики	Python, FastAPI, Docker	–

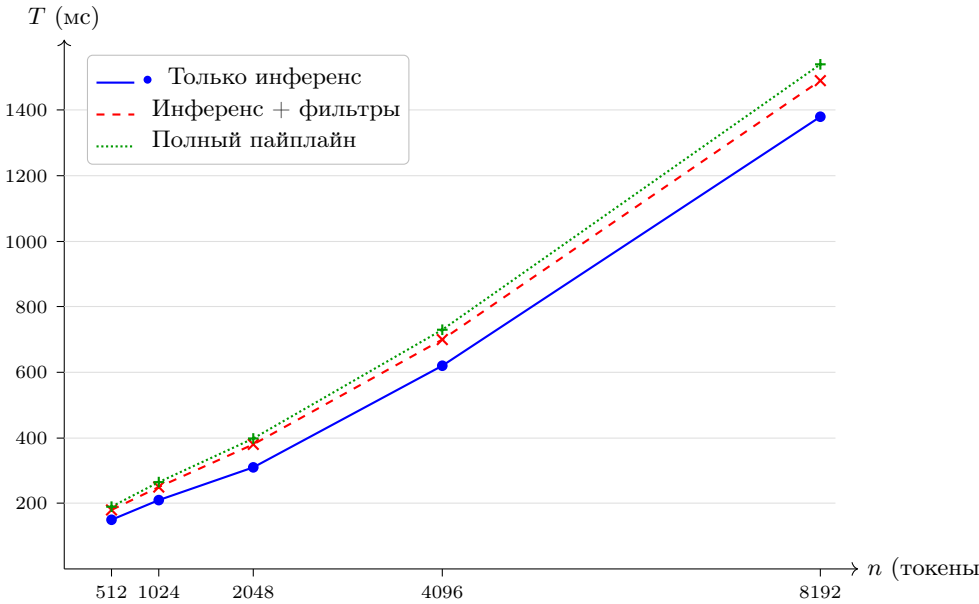

Рис. 4 — Зависимость задержки обработки запроса от длины входного контекста (LLaMA-2-13B, vLLM, NVIDIA A100).

Безопасность протокола MCP. Model Context Protocol (MCP) стандартизирует взаимодействие LLM-агентов с инструментами через JSON-RPC. Проблема в том, что протокол не фиксирует политику полномочий до начала работы агента и не отделяет недоверенные данные от инструкций. Формализована атака «инъекция запроса → перехват инструмента» (*Prompt Injection to Tool Hijacking*): злоумышленник подставляет инструкцию в контекст, и агент вызывает инструмент с параметрами, которые контролирует атакующий.

Для количественной оценки рисков собран бенчмарк MCPBench из 847 сценариев атак. Сценарии сгруппированы по трем типам воздействия: косвенная инъекция, манипуляция ответами инструментов и межсерверное распространение. Каждый сценарий содержит вредоносную инструкцию, встроенную в контекст вызова инструмента, и проверку успешности атаки по факту выполнения несанкционированного действия. MCP усиливает базовые уязвимости LLM на 16,6–41,6 п.п. по сравнению с работой без инструментов. Программный модуль MCPSec противодействует этим атакам. Первый его механизм, алгоритм AttestMCP, ограничивает вызовы инструментов сессионной таблицей полномочий и контролирует целостность пакетов HMAC-подписью:

Алгоритм AttestMCP

Вход: RPC-пакет $\text{pkt} = (\text{method}, \text{params}, \sigma)$, полномочия $\mathcal{P}$, ключ $K_{session}$.
Извлечь $(\text{sid}, \text{tid}, \text{args}) \leftarrow \text{ParseRPC}(\text{pkt})$
$\sigma' \leftarrow \text{HMAC}(K_{session}, \text{sid} \| \text{tid} \| \text{hash}(\text{args}))$
if $\sigma \neq \sigma'$ **then Вернуть** Error(INVALID_SIGNATURE)
end if
if $(\text{sid}, \text{tid}) \notin \mathcal{P}$ **then Вернуть** Error(PERMISSION_DENIED)
end if
if $\text{args} \notin \text{AllowedArgs}(\text{sid}, \text{tid})$ **then Вернуть** Error(ARGS_VIOLATION)
end if
Вернуть CallToolResult(ExecuteTool(tid, args))

Таблица полномочий формируется при инициализации сессии, до поступления недоверенных данных, и в ходе работы не расширяется. Множество вызовов, доступных агенту, ограничено ею независимо от инъекций в контекст. HMAC-подпись на сессионном ключе $K_{session}$ защищает пакеты от подмены и не дает скомпрометированному MCP-серверу подделать вызов. Накладные расходы AttestMCP не превышают 0,1 мс на вызов, что пренебрежимо на фоне инференса LLM. Второй механизм защиты, Commit Boundary, изолирует код, генерируемый агентом, через Docker-контейнеры с ограничениями cgroups. Дополнительные механизмы фреймворка закрывают остаточные векторы атак: сертификаты возможностей ограничивают допустимый набор операций, маркировка происхождения отделяет вывод модели от пользовательских запросов, принудительная изоляция блокирует межсерверную передачу без явного разрешения, защита от повторов

нейтрализует перехват сообщений. В совокупности модуль MCPSec снижает ASR агентских атак с 53,7% до 12,4%.

Шестая глава «Практическое применение» описывает, как перенести результаты предыдущих глав в реальную эксплуатацию. Предложен пятиступенчатый процесс: (1) профилирование рисков на основе таксономии из первой главы, (2) подбор конфигурации защиты по метрике RUT, (3) встраивание защитных модулей в конвейер обработки, (4) тестирование по метрикам из четвертой главы, (5) непрерывный мониторинг в рабочем режиме.

Для количественного сравнения конфигураций введена метрика компромисса «устойчивость–полезность»:

$$\mathrm{RUT} = \frac{\Delta\mathrm{Robustness}}{\Delta\mathrm{Utility} + \Delta\mathrm{Cost} + \epsilon},$$

Все величины нормированы на [0,1], $\epsilon = 10^{-6}$ предотвращает деление на нуль. RUT показывает, сколько устойчивости приходится на единицу потерь в полезности и затратах. Для комитета из 7 моделей получено RUT $\approx$ 5,5: выигрыш в устойчивости многократно превосходит рост расходов.

На основании экспериментов сформулированы практические рекомендации. Гетерогенные комитеты из 5–7 моделей с мажоритарным голосованием показывают наилучший баланс устойчивости и затрат по метрике RUT. В агентских системах приоритетны ограничение полномочий вызовов инструментов зафиксированной таблицей и изоляция генерируемого кода.

Экономика комитетов линейна по числу участников, но запросы к моделям параллелизуемы, поэтому реальная задержка растет медленнее. Для экономии ресурсов предложена каскадная схема: комитет активируется только при низкой уверенности одиночной модели. Количественная оценка экономии относится к направлениям дальнейших исследований.

Этап мониторинга предполагает сбор статистики по срабатываниям защитных механизмов, отслеживание новых публикаций об уязвимостях и периодическое повторное тестирование. Быстрая эволюция как самих LLM, так и методов атак делает статические конфигурации защиты ненадежными. Метрика RUT позволяет пересчитывать оптимальную конфигурацию при появлении новых данных об угрозах.

Внедрение выполнено в двух организациях. На факультете ВМК МГУ имени М. В. Ломоносова результаты включены в курс «Робастные модели в машинном обучении». В компании ВИАСАТ ТЕХ фреймворк оценки уязвимостей LLM-as-a-Judge, комитетная защита и методология многошаговых атак задействованы в модуле ранжирования рекомендаций. Комитет из 3 моделей сократил долю некорректных оценок, обусловленных манипуляциями контентом, при этом рост латентности не превысил 15%

благодаря параллельному опросу моделей. Суммарная вычислительная нагрузка защитных компонентов составляет 28,4% от базового инференса, однако при параллельном развертывании реальное увеличение задержки существенно ниже.

Заключение содержит итоги и основные выводы работы:

1. Экспериментальное исследование на данных TDC2023 выявило асимметрию между генерацией и восстановлением троянских триггеров: Recall строкового сходства не превышает 0,17, тогда как REASR суррогатных триггеров достигает $\approx$ 0,99. Этот разрыв указывает на необходимость превентивных мер на этапе обучения модели.
2. Введена метрика генеративной устойчивости $R_{stab}(f)$, учитывающая дискретность пространства токенов и авторегрессионную природу генерации. Для локализованных атак доказана верхняя граница уязвимости $V(h) \leq 1 - R_{class}(h)$, для нелокализованных воздействий построено параметрическое приближение $V(h) \approx g(1 - R_{class}(h))$, подлежащее экспериментальной калибровке.
3. Алгоритм ASA, разработанный для исследования уязвимости LLM-as-a-Judge, достигает ASR до 73,8% в условиях «черного ящика». Инъекционные атаки разделены на инструктивный и контентный классы по механизму воздействия.
4. При анализе многошаговых атак на LLM с многоуровневой защитой установлены четыре класса обхода со специфичными контрмерами, многоуровневая фильтрация снижает ASR с 90% до 15–25%. Гетерогенные комитеты из 5–7 моделей дополнительно сокращают ASR на 47–55 п.п., для Gemma-3-4B показатель падает с 73,8% до 19,3%. Эффективность комитета согласуется с теоремой Кондорсе и обеспечивается декорреляцией ошибок между архитектурами разных семейств.
5. Обнаружены архитектурные уязвимости протокола MCP. Программный модуль MCPSec, объединяющий алгоритмы AttestMCP и Commit Boundary с четырьмя вспомогательными механизмами защиты, снижает среднюю ASR агентских атак с 53,7% до 12,4% на бенчмарке MCPBench.
6. Комплексы JudgeGuard и TrojanArmor, зарегистрированные под свидетельствами № 2025687702 и № 2025684247, апробированы в международных бенчмарках. JudgeGuard вместе с методом защиты на основе комитетов внедрен в производственной среде компании ВИАСАТ ТЕХ. Разработанные программные средства задействованы в учебном процессе факультета ВМК МГУ.

Апробация проведена на конференциях DCCN 2024 и DIALOG 2022, а также на научных конференциях «Тихоновские чтения 2023» и «Ломоносовские чтения 2022». Экспериментальные методы проверены

в международных соревнованиях TDC2023, SaTML CTF 2024 и Kaggle «LLMs: You Can't Please Them All». Программные комплексы JudgeGuard и TrojanArmor зарегистрированы в Роспатенте. Результаты опубликованы в 9 научных работах по теме диссертации, из которых 6 относятся к рекомендованным ВАК и приравненным к ним: 4 статьи в журналах ВАК (3 категории К1 и 1 категории К2) и 2 свидетельства о государственной регистрации программ для ЭВМ; кроме того, 1 статья индексируется в Scopus на уровне Q2, имеются доклад на конференции и препринт.

Публикации автора по теме диссертации

1. *Maloyan, N.* Investigating the Vulnerability of LLM-as-a-Judge Architectures to Prompt-Injection Attacks [Текст] / N. Maloyan, B. Ashinov, D. Namiot // International Journal of Open Information Technologies. — 2025. — Т. 13, № 9. — URL: http://injoit.ru/index.php/j1/article/view/2261. — (К1 ВАК).
2. *Maloyan, N.* Prompt Injection Attacks on Agentic Coding Assistants: A Systematic Analysis of Vulnerabilities in Skills, Tools, and Protocol Ecosystems [Текст] / N. Maloyan, D. Namiot // Современные информационные технологии и ИТ-образование. — 2025. — Т. 21, № 3. — URL: http://sitito.cs.msu.ru/index.php/SITITO/article/view/1280. — (К2 ВАК).
3. *Maloyan, N.* Breaking the Protocol: Security Analysis of the Model Context Protocol Specification and Prompt Injection Vulnerabilities in Tool-Integrated LLM Agents [Текст] / N. Maloyan, D. Namiot // International Journal of Open Information Technologies. — 2026. — Т. 14, № 2. — URL: http://injoit.org/index.php/j1/article/view/2423. — (К1 ВАК).
4. *Maloyan, N.* Sleeper Channels and Provenance Gates: Persistent Prompt Injection in Always-on Autonomous AI Agents [Текст] / N. Maloyan, D. Namiot // International Journal of Open Information Technologies. — 2026. — Т. 14, № 6. — URL: https://injoit.org/index.php/j1/article/view/2621. — (К1 ВАК).
5. *Малоян, Н. Г.* JudgeGuard. Программа для экспериментального исследования уязвимостей и тестирования методов защиты систем автоматической оценки на базе больших языковых моделей [Текст] / Н. Г. Малоян. — 2025. — Рос. Федерация. Свидетельство о государственной регистрации программы для ЭВМ № 2025687702. — (приравнено к публикации ВАК).

6. *Малоян, Н. Г.* TrojanArmor. Программа для экспериментального исследования троянских атак в нейронных сетях и методов защиты от них [Текст] / Н. Г. Малоян. — 2025. — Рос. Федерация. Свидетельство о государственной регистрации программы для ЭВМ № 2025684247. — (приравнено к публикации ВАК).

7. *Khomsky, D.* Prompt Injection Attacks in Defended Systems [Текст] / D. Khomsky, N. Maloyan, B. Nutfullin // Distributed Computer and Communication Networks. — Springer Nature Switzerland, 2025. — С. 404—416. — URL: http://dx.doi.org/10.1007/978-3-031-80853-1_30. — (Q2 Scopus).

8. *Maloyan, N.* DIALOG-22 RuATD Generated Text Detection [Текст] / N. Maloyan, B. Nutfullin, E. Ilyshin // Computational Linguistics and Intellectual Technologies. — RSUH, 06.2022. — С. 394—401. — URL: http://dx.doi.org/10.28995/2075-7182-2022-21-394-401.

9. *Maloyan, N.* Adversarial Attacks on LLM-as-a-Judge Systems: Insights from Prompt Injections [Текст] / N. Maloyan, D. Namiot. — 2025. — arXiv: 2504.18333 [cs.CR]. — URL: https://arxiv.org/abs/2504.18333.

Малоян Нарек Гагикович

Разработка методов оценки и повышения устойчивости больших языковых моделей к вариациям входных последовательностей

Автореф. дис. на соискание ученой степени канд. техн. наук

Подписано в печать ___.___._____. Заказ № _____
Формат 60×90/16. Усл. печ. л. 1. Тираж 100 экз.
Типография _____