

Безопасное внедрение облачных ИИ-агентов в процессы разработки: опыт «Лаборатории Касперского»



kaspersky

Краткое содержание и терминология

ИИ-агенты — программные системы на базе больших языковых моделей (LLM), способные самостоятельно взаимодействовать с файлами, базами данных и другими системами. ИИ-агенты нашли широкое применение во всех областях, связанных с разработкой, внедрением и эксплуатацией ПО: они могут создавать и проверять программный код, проводить тестирование, исправлять ошибки, обнаруживать уязвимости и пр. Внедрение ИИ-агентов часто идёт «снизу вверх» в обход политик и процедур ИБ, что порождает «теневой ИИ» и, как следствие, создаёт значительные риски как для конечных продуктов, так и для инфраструктуры организации. Создание политик и процессов, позволяющих комфортно и безопасно использовать ИИ-инструменты, позволяет снизить риски при одновременном повышении производительности организации.

Использование ИИ-агентов сопряжено с рисками, возникающими из самой природы LLM: уязвимость к вредоносному вводу, недоверенность «когнитивного ядра», непрозрачность работы, вероятностное поведение.



Облачные ИИ-агенты подключаются к внешним сервисам, предоставляющим доступ к большим языковым моделям (Claude, ChatGPT и другие). В качестве облачных ИИ-агентов могут использоваться как официальные программные пакеты от разработчиков моделей, такие как Claude Code, Codex, так и сторонние приложения, например OpenCode, подключённые к облачным сервисам. Использование облачных ИИ-агентов позволяет использовать передовые ИИ-модели без необходимости в организации дорогой вычислительной инфраструктуры, однако создаёт дополнительные риски: утечки коммерческой тайны, персональных данных, внесение вредоносных изменений в ответы модели и компрометацию инфраструктуры организации. Облачные ИИ-агенты фактически создают новый, непрозрачный периметр организации, связанный с её внутренней инфраструктурой.

В документе рассматриваются основные риски, связанные с внедрением облачных ИИ-агентов в процессы разработки ПО, и подходы «Лаборатории Касперского» к их снижению. Приведены практические решения из опыта внедрения ИИ-агентов для разработчиков R&D.

Предлагаются следующие решения для безопасного использования облачных ИИ-агентов в разработке:



Определение требований пользователей, модели угроз, компенсирующих мер



Изоляция агентов: на уровне сети, учётной записи, рабочих станций разработчиков



Категоризация кода и документации: некритичные — с доступом для ИИ-агентов, критичные — доступ запрещён



Журналирование доступа к моделям и действий агента на нескольких уровнях системы



Использование LLM-фаерволла для внедрения правил фильтрации, блокирования доступа к критичному коду и данным, наблюдаемости взаимодействий с облачной моделью

Краткий контекст

В этом документе мы рассматриваем конкретный пример внедрения ИИ-агентов, использующих облачные LLM, в процессы разработки в R&D «Лаборатории Касперского». ИИ-агенты используются в «Лаборатории Касперского» уже достаточно давно, однако из-за чувствительности кода, связанного с защитой пользователей от кибератак, действовали строгие ограничения — разработчикам были доступны только модели с открытыми весами, развернутые в нашем внутреннем LLM-сервисе. Несмотря на многочисленные преимущества моделей, развернутых в контуре компании, высокий спрос на агентные системы означает высокие капиталовложения для обеспечения достаточной производительности сервиса. По некоторым оценкам, переход от взаимодействия с моделями по модели чата к использованию агентов приводит к росту нагрузки на сервис на порядки. Одновременно с этим характеристики облачных моделей как по производительности, так и по сложности задач, которые они могут решать, зачастую превосходят характеристики открытых моделей: модели, которые разработчики могли использовать в личных целях, показывали лучшие результаты, чем используемые в рабочей среде. Запрет использования облачных ИИ-агентов создавал риски — например, непрозрачного использования личных доступов к ИИ-моделям и утечки через них коммерческой тайны компании. В результате было принято решение о внедрении в процессы разработки ряда проектов агентов на базе облачных сервисов LLM.

Пилотный проект

Первым этапом в проекте внедрения был пилотный проект, разбитый на несколько потоков для тестирования разных ИИ-агентов и сервисов. Были определены первоначальные требования пользователей-разработчиков, а затем на основе уже существующего перечня рисков при использовании облачных ИИ-моделей наши эксперты ИБ составили новый, специфичный для агентского режима. В процессе его проработки были приняты во внимание мировой опыт и сложившиеся практики, а также существующие фреймворки¹, методологии² и рекомендации, которые анализировались с учетом специфики нашей компании.

1 https://cheatsheetseries.owasp.org/cheatsheets/AI_Agent_Security_Cheat_Sheet.html#key-risks

2 <https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/>



Основные риски использования облачных ИИ-агентов в разработке



1. Непреднамеренное выполнение вредоносных инструкций агентом в инфраструктуре организации

LLM генерирует команды и вызовы инструментов, которые могут быть деструктивными, причем происходить это может даже без внешнего воздействия. Яркий пример — инцидент с ИИ-агентами OpenAI, в ходе которого неверная конфигурация окружения привела к непреднамеренному взлому инфраструктуры компании HuggingFace¹. Возможные последствия: нарушение целостности инфраструктуры, утечки и потеря данных, компрометация сервисов и др.



2. Косвенная промпт-инъекция² и Living-off-the-Agent

Злоумышленник может подставить в запрос к LLM вредоносные инструкции, причем они могут оказаться в файлах внешних программных пакетов, скиллах (навыках), которыми разработчики расширяют функционал моделей, и даже в служебных файлах скомпрометированных моделей. Кроме того, злоумышленники, уже получившие доступ к устройству, могут использовать установленных на нем агентов в своих целях, например, для поиска и эксфильтрации конфиденциальных данных, как это произошло в инциденте с билд-системой Nx (атака singularity). Возможные последствия: выполнение нежелательных операций, утечки или потеря данных, обход защитных проверок.



3. Вредоносные MCP и скиллы

ИИ-агенты могут самостоятельно загружать и использовать сторонние навыки и MCP-сервера, которые могут содержать вредоносный код³ или подгружать его после выполнения инструкции, вызванной косвенной промпт-инъекцией. Известны реальные инциденты, связанные как с вредоносными файлами скиллов⁴, так и с MCP-серверами, выдающими опасные инструкции, ведущие к эксфильтрации чувствительных данных с устройства пользователя⁵. Возможные последствия: аналогично риску 1.



4. Раскрытие конфиденциальных данных

Агент может передавать или сохранять секреты, коммерческую тайну, персональные данные и другую конфиденциальную информацию в облачный сервис. Известны случаи, когда это приводило к реальным утечкам конфиденциальных данных⁶. Возможные последствия: утечка коммерческой тайны или персональных данных, обучение облачной модели на полученных данных и дальнейшее её раскрытие третьим лицам.



5. Отсутствие возможности расследовать причины инцидента

Недостаток логирования и аудита действий ИИ-агентов ограничивает возможности дальнейшего анализа в случае ошибок или намеренной компрометации. Возможные последствия: длительное восстановление, повторные инциденты.



6. Внесение в код вредоносных функций и уязвимостей

Возможно внесение уязвимых фрагментов кода и вредоносных закладок в разрабатываемый код или инфраструктуру — как непреднамеренно, так и в результате косвенных промпт-инъекций. Возможные последствия: атаки на цепочку поставок через продукты, их скрытая эксплуатация, несанкционированный доступ к инфраструктуре организации или её клиентов.

1 https://www.rbc.ru/technology_and_media/04/09/2026/6a9aa1ac89f0a318e75345c3

2 <https://www.kaspersky.ru/resource-center/threats/prompt-injection>

3 <https://www.kaspersky.ru/blog/supply-chain-attacks-in-2025/41594/>

4 <https://www.kaspersky.ru/blog/moltbot-enterprise-risk-management/41329/>

5 <https://www.pillar.security/blog/deadbugz-currently-active-mcp-supply-chain-campaign>

6 <https://www.forbes.com/sites/janakirammsv/2026/06/29/prompts-are-the-new-malware-as-enterprise-ai-defenses-fall-behind/>

Пилотная реализация

Пилотные проекты были запущены параллельно для нескольких групп разработчиков и отличались только ИИ-агентами и моделями.

УЗ – учетная запись пользователя
ИС – интеллектуальная собственность



Для снижения рисков была построена инфраструктура, обеспечивающая максимальную изоляцию агентов от инфраструктуры: агенты запускались на специально выделенных VDI-машинах, изолированных от основной сети. Сетевой доступ предоставлялся только к ограниченному количеству внутренних сервисов (авторизация, обмен файлами, контроль версий) и серверам облачной модели, а разработчики подключались через сервис удаленного рабочего стола VDI. Для организации доступа только к конкретным веткам исходных текстов, разрешённых для агента, применялись возможности нашей собственной системы контроля версий для монорепозитариев — Ast¹.

Реальное использование показало, что данный подход не оптимален:

Решили отказаться от использования VDI

В нашей инфраструктуре VDI дают меньше гибкости разработчику из-за специфики задач системного программирования. Ряд подразделений пишут код, который отлаживается только на специфичных аппаратных конфигурациях, что привело к необходимости дублировать среды разработки и отладки. Мы перешли на рабочие станции разработчиков.

Поняли, что агент должен получать больше контекста

Ему нужен доступ к документации, требованиям, задачам и другим рабочим элементам трекинг-систем

Агенту нужен более широкий доступ к коду для возможности сборки и написания тестов для сложных компонентов

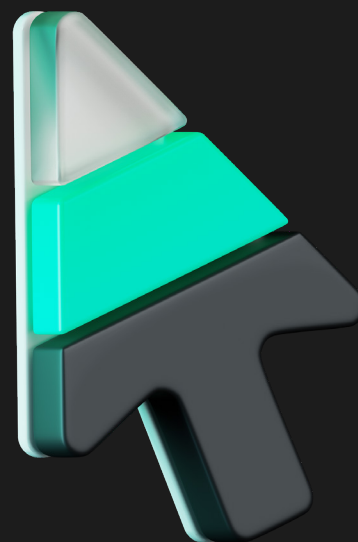
По результатам тестирования пилотной схемы были выбраны наиболее подходящие ИИ-агент и семейство моделей, но техническую реализацию проекта было решено изменить. Нужно отметить, что приведенные ниже меры применимы к разным провайдерам и агентным обязательствам (agent harness) и не являются специфичными для выбранных в рамках пилота.

¹ <https://ecode.ozone.tech/archive/2024/talks/20006303/>

Требования к реализации

Опыт проведения пилотных проектов показал, что разработчикам необходима большая гибкость и более широкий набор легко доступных инструментов для эффективного использования ИИ-агентов:

- 1 **Контролируемый доступ к исходному коду проектов** — требуется для понимания контекста ИИ-агентом, разработки тестов, сборки и тестирования компонентов. Предоставление полного доступа к исходным текстам проектов, защищающих пользователей от кибератак, создаёт риски внедрения вредоносного кода, внесения уязвимостей в критичные части продукта и утечки коммерческой тайны, поэтому код был разделён на две категории — критичный код, который доступен только людям-разработчикам, и некритичный код, доступный агентам. Определены критерии отнесения кода к критичному и порядок предоставления доступа к нему.
- 2 **Доступ к системам разработки: требования и задачи в таск-трекере, документация в корпоративной вики.** Полный доступ с правами разработчика создаёт риски утечек, поэтому были определены категории критичных проектов, доступ к которым для агентов запрещён.
- 3 **Запуск ИИ-агентов на рабочих станциях разработчиков.** Именно отрицательный опыт и выводы предыдущего этапа показали, что если не создать среду, которая будет удобна разработчикам, то результатом может стать разрозненная и неконтролируемая инфраструктура «теневого ИИ», применяемого в обход политик ИБ. Поиск баланса удобства и изоляции стал самым серьёзным вызовом. Запуск агента без изоляции создаёт все вышеописанные риски, поэтому необходимо было создать безопасную среду, которая оградит системы от ошибок и вредоносных действий агента и обеспечит контролируемый и прозрачный доступ только к разрешённым ресурсам и системам разработки.

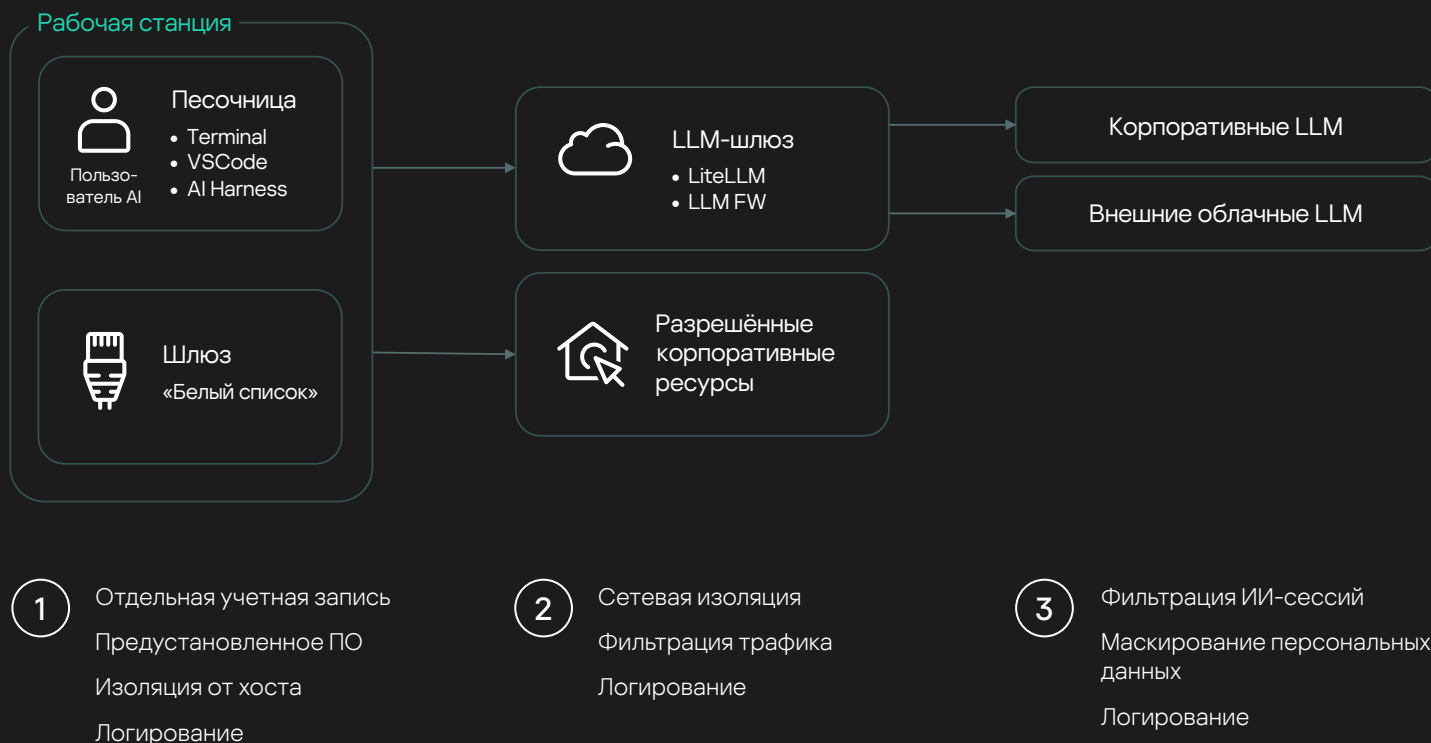


Реализация

По результатам пилотных проектов схема использования ИИ-агентов была переработана. При этом учитывались базовые принципы информационной безопасности, подход *secure-by-design*¹, пожелания пользователей и требования экспертов управления информационной безопасности. Безопасность использования агентов обеспечивается на нескольких уровнях: техническом — решениями для изоляции, настройки и запуска рабочей среды, процессном — через разграничение доступа к ресурсам и выделение учётных записей, и экспертном — мониторингом событий внутренним SOC.

¹ <https://trends.rbc.ru/trends/industry/cmrm/69145be59a79471fb4365740>

Рабочая среда



Основа решения — это программная изоляция рабочей среды агента от внешней среды, в нашем случае — системы разработчика. Мы протестировали несколько вариантов: полнофункциональные виртуальные машины не подходили из-за значительного падения производительности, контейнеризация подходила только для Linux (например, Windows-контейнеры Docker сильно ограничены и кратно медленнее нативной ОС) и не компенсировала риск компрометации ядра ОС хоста, VDI был исключён после тестирования в пилотных проектах. В результате для каждой из ОС определили рабочее решение, удовлетворяющее пользователей по производительности и при этом позволяющее закрыть ИБ-риски.

Для запуска рабочей среды были разработаны и поддерживаются готовые пакеты: разработчику достаточно загрузить и запустить установщик, который произведёт все настройки ОС, средств изоляции и инструментов разработки — реализация принципа «удобства пользователя» (в данном случае разработчика).

Изоляция окружения («песочница»)

Изоляция контейнера необходима для компенсации риска компрометации системы разработчика и исполнения на ней вредоносного кода.



Windows: Windows Sandbox

В отличие от виртуальной машины, создающей отдельную копию ОС, Windows Sandbox использует (но не может изменять) системные файлы хоста, что даёт большие преимущества по утилизации дискового пространства и производительности: не нужно хранить отдельный образ, песочница запускается за секунды, а не минуты. Кроме того, не требуется устанавливать ОС в виртуальную машину, выполняется лишь запуск готового набора файлов, используемых совместно с хост-системой. Штатное обновление Windows обновляет также и версию песочницы, что упрощает сопровождение и позволяет переиспользовать текущие механизмы обновлений.



Linux: Kata Containers

Открытый проект, реализующий Docker-совместимый интерфейс, но обеспечивающий при этом большую изоляцию: контейнер запускается внутри отдельной лёгкой VM со своим ядром. Права контейнера ограничены изолированным ядром песочницы, что снижает риск компрометации хост-системы. По скорости запуска Kata Containers значительно ближе к обычному контейнеру, чем к полноценной VM с отдельной установкой ОС.



MacOS: Tart VM + softnet

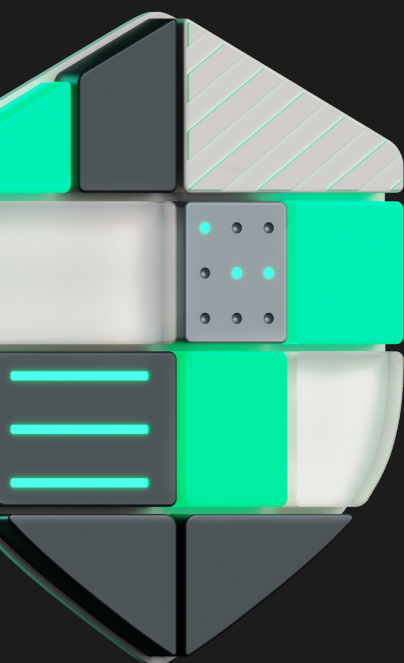
На Apple Silicon используется виртуальная машина tart. В отличие от решений для Windows и Linux, позволяющих настроить сетевую маршрутизацию своими средствами, здесь дополнительно необходим softnet — host-side фильтр, который контролирует соединения от виртуальной машины. В tart запускается полноценный macOS-гость (с Xcode), а защита вынесена на хост. Использование полноценной виртуальной машины на macOS оправдано, т. к. ОС не имеет встроенной «лёгкой» песочницы уровня Windows Sandbox или Kata.



Слой сетевой изоляции: контейнер-роутер между песочницей и сетью

Это не менее важный слой безопасности, который тоже необходимо реализовать. Сетевая изоляция снижает риск компрометации инфраструктуры и утечки критичных данных. Ограничение сетевых соединений контролирует доступ агента к сетевым ресурсам, разрешая доступ только к тем, которые входят в согласованный «белый список», и обеспечивает журналирование сетевых соединений. Это снижает риск, что несанкционированные действия агента останутся незамеченными.

Параллельно с песочницей запускается отдельный шлюз (Gateway VM) — лёгкий образ виртуальной машины, которая выполняет функции роутера для агента и является единственным доступным для него сетевым маршрутом. Шлюз функционирует независимо от песочницы, и его настройки защищены от изменения агентом.



Шлюз:

- 1 Выделяет агенту IP-адрес и функционирует как DNS-сервер (DHCP/dnsmasq) — песочница не выбирает адрес сама
- 2 Фильтрует исходящий трафик только по «белому списку» пар адрес/порт, все остальные пакеты отклоняются
- 3 Делает доступным только ограниченный набор доменных имен из согласованного «белого списка»

Сетевые ограничения реализованы единообразно на всех ОС: изолированный агент имеет сетевую связность только с шлюзом, доступ к внутренней инфраструктуре обеспечивается шлюзом с фильтрацией, а все остальные соединения заблокированы на уровне гипервизора и настроек сети, недоступных к изменению из песочницы, что создаёт дополнительный слой изоляции.

LLM Gateway: контроль доступа к облачной модели

LLM Gateway (шлюз) — ключевой элемент схемы, являющийся единой точкой контроля потоков данных между агентом и языковой моделью. ИИ-агент осуществляет всё взаимодействие с облачной моделью через LLM Gateway, прямой доступ к облачным API запрещён. Это обеспечивает наблюдаемость, контроль доступа и учет потребляемых ресурсов.

LLM Gateway для облачных моделей был построен на основе уже имеющегося в компании шлюза¹, который реализовывал доступ к LLM, развернутому в периметре компании, и был создан на базе доработанного решения с открытым исходным кодом (LiteLLM). Для решения задач безопасности в этот шлюз было интегрировано наше внутреннее решение класса LLM Firewall для обеспечения безопасного взаимодействия с большими языковыми моделями. Итоговая конфигурация позволяет достичь следующих целей:



Контроль доступа и маршрутизация

При запуске песочницы сервис выделяет агенту ключ авторизации для доступа к модели через LLM Gateway. Шлюз проверяет ключ авторизации, определяет доступные для пользователя облачные ИИ-модели и обеспечивает авторизацию в облачном сервисе, не раскрывая ключи доступа клиенту.



Полное журналирование

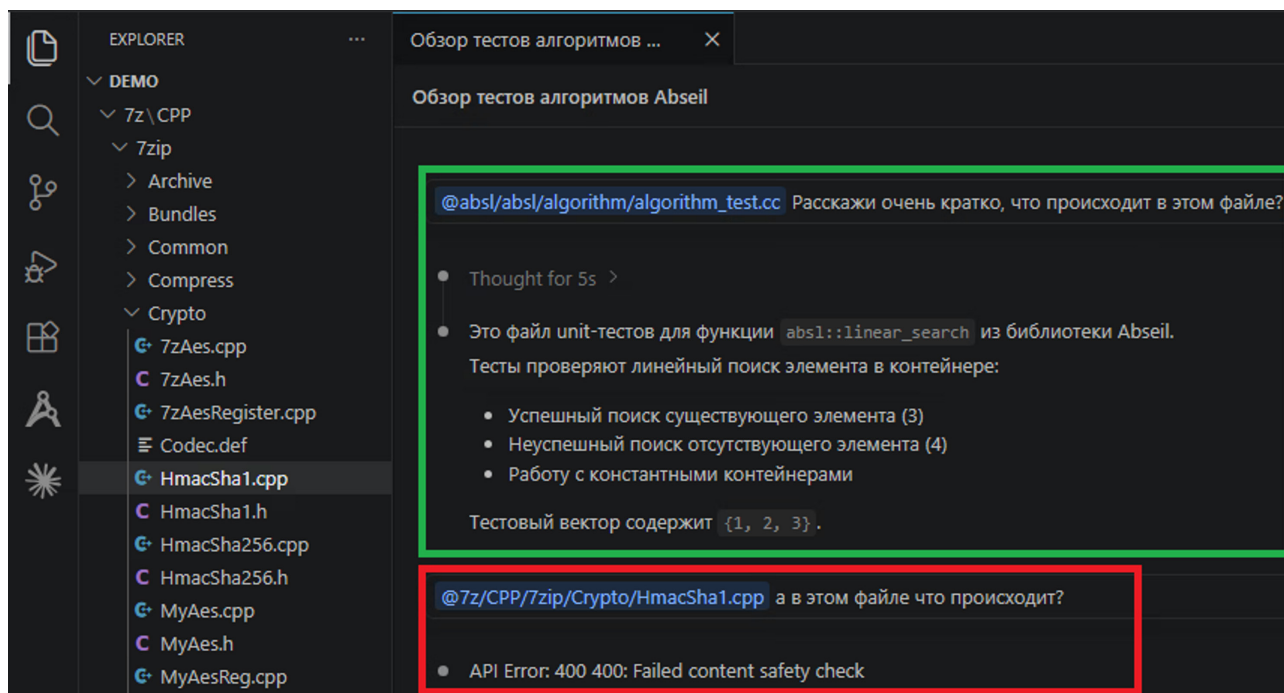
Все обращения к моделям журналируются с использованием корпоративного кластера OpenSearch. При этом LLM Firewall в случае обнаружения (и, соответственно, маскирования) чувствительных данных генерирует дополнительные события, доступные для анализа специалистам управления ИБ, а события детектирования обращений к чувствительному коду передаются напрямую в SIEM для анализа в качестве инцидентов.



Защита секретов, конфиденциальных и персональных данных

LLM Firewall проверяет исходящие запросы на предмет наличия чувствительных и персональных данных или секретов (например, внутренних IP, электронных писем сотрудников или ключей доступа), причем анализ производится как для промптов разработчика, так и для выводов результатов вызова инструментов. Для обеспечения бесперебойной работы агента LLM Firewall при обнаружении чувствительных данных осуществляет не блокировку запроса, а маскирование: подмену этих данных на заглушку (например, sk-key-123... превращается в <API_KEY1>). Данная заглушка отправляется в облако, а при получении ответа подменяется обратно на исходное значение, чтобы агент мог воспользоваться секретом без утечки его в облако. Правила детектирования были построены на базе уже существующих правил модуля DLP и внедренной в процессы «Лаборатории Касперского» в рамках развития SDL/DevSecOps-практик системы поиска секретов.

¹ <https://jugspeakers.info/ru/information/talks/talk/6619>



Защита критичного кода

Перечисленные выше меры не гарантируют, что агент случайно не получит доступ к участкам кода, признанного чувствительным. Отдельный модуль LLM Firewall исключает утечку чувствительного кода (даже изменённого или переформулированного) во внешнее облако. Он использует комбинацию чётких, нечётких и семантических подходов, которые позволяют достичь максимальной полноты детектирования. Первый слой анализа определяет, что файл, который читает агент, является чувствительным, а более глубокий анализ позволяет предотвратить ситуации, когда агент использует нестандартные способы получения доступа к защищаемому коду.

Мониторинг: обеспечение наблюдаемости и реагирования

Один из основных рисков — непрозрачность действий ИИ-агента, особенно в отсутствие постоянного наблюдения со стороны разработчика. Для компенсации риска требуется обеспечить журналирование на нескольких уровнях: хост-машины, среды исполнения (песочницы), непосредственно агента, канала взаимодействия с облачной моделью. На каждом из этих уровней реализованы новые или подключены штатные syslog-коллекторы:



Хост-машина разработчика

Установщик отчитывается о начале, завершении и проблемах установки



Слой сетевой изоляции — шлюз

Публикует свой старт, получение или компиляцию белого списка и возможные ошибки шага настройки, а также каждое заблокированное исходящее соединение



Агент в песочнице

Публикует каждое действие агента через хуки, такие как вызов инструментов Bash/PowerShell, mcp_* и пр.



События ОС в песочнице

Полный журнал ОС песочницы передаётся в syslog-коллектор



LLM Gateway

Журналирование всех запросов и ответов облачной ИИ-модели и вызовов инструментов



LLM Firewall

Журналирование и инвентаризация использования MCP и скиллов, событий детектирования критичного кода и маскирования персональных данных

На каждом этапе запуска среды исполнения агента реализовано критическое условие: ошибка включения любого из источников журналирования приводит к остановке. Таким образом гарантируется полная наблюдаемость для аналитиков SOC.

Учётные записи и доступ к внутренним ресурсам

Для возможности полнофункциональной разработки с помощью ИИ-агентов требуется сетевой доступ к внутренним ресурсам — сервисам контроля версий, спискам требований, документации. Использование регулярных учётных записей разработчиков создаёт риски утечки и доступа к критичному коду и данным, поэтому запуск и использование агентов разрешены только для специальных сервисных учётных записей, уникальных для каждого разработчика. Сервисная учётная запись для ИИ-агента имеет фиксированный узнаваемый префикс, по умолчанию не наследует какие-либо права доступа и получает доступ только к разрешённым репозиториям с некритичным кодом. Процесс предоставления дополнительных доступов регламентирован: владелец ресурса обязан убедиться в том, что не открывает доступ к критичному коду для ИИ-агента.

Изменения в процессе SDL

Код, разработанный при участии ИИ-агентов, считается заведомо недоверенным. Использование такого кода без дополнительных проверок создаёт риски атаки на цепочки поставок и внедрения вредоносного кода или скрытых уязвимостей. Для компенсации этих рисков такой код в обязательном порядке проходит дополнительные проверки: обязательное код-ревью с участием человека, статический анализ (SAST), динамический анализ (DAST), композиционный анализ (SCA) и другие практики безопасной разработки (Secure Development Lifecycle/РБПО), внедрённые в компании в соответствии с установленными процессами. Кроме того, используются и ИИ-автоматизации, такие как дополнительное ревью кода с помощью LLM.

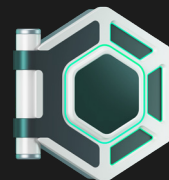
Результат

Внедрение ИИ-агентов в процессы разработки программного обеспечения — сложный процесс, и использование таких инструментов без специально разработанных мер и процессов защиты создаёт дополнительные риски для организации. Схема, разработанная и реализованная департаментами R&D, ИБ и ИТ «Лаборатории Касперского», компенсирует риски и обеспечивает:

Комфортное и безопасное использование ИИ-агентов на рабочих станциях разработчиков



Ограниченный и контролируемый доступ к внутренним ресурсам



Прозрачность действий ИИ-агентов



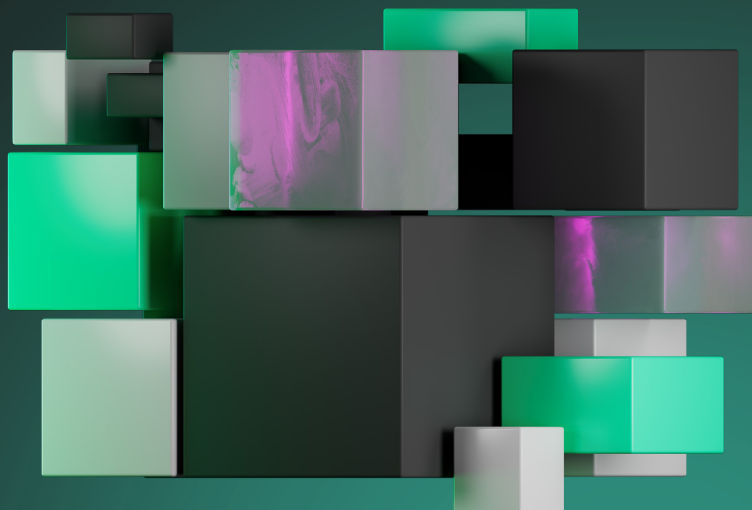
Контроль и фильтрацию персональных данных и критичных данных компании



Чек-лист — безопасное внедрение облачных ИИ-агентов в процессы разработки

Данный чек-лист дополняет и уточняет выпущенный ранее чек-лист¹ по внедрению ИИ-агентов в процессы разработки пунктами и рекомендациями, специфичными для использования облачных моделей.

- | | |
|--|--|
|  Обеспечена ли безопасность рабочих станций разработчиков? | ИИ-агенты с доступом к системам разработчиков могут производить неправомерные действия, исполнять вредоносные команды и код, приводить к утечкам критичных кода и данных. Обеспечьте изоляцию среды исполнения, установку EDR на рабочие станции, желательно подключение к MDR. |
|  Обеспечена ли сетевая изоляция ИИ-агентов от инфраструктуры организации? | Неконтролируемый доступ к сетевой инфраструктуре создаёт риски компрометации внутренних сервисов и длительного неправомерного доступа злоумышленников. Обеспечьте ограниченный доступ среды исполнения ИИ-агента только к ограниченному списку ресурсов, согласованному со службой ИБ. |
|  Проведена ли категоризация кода и данных? | Исходный код и документация могут содержать коммерческую тайну. Доступ ИИ-агента к ним приведёт к утечке. Определите категории критичного кода и данных, доступ к которым для ИИ-агентов запрещён, и порядок отнесения к этой категории. |
|  Определены ли процессы доступа ИИ-агентов к ресурсам организации? | Использование регулярных учётных данных и токенов доступа создаёт риски утечки критичного кода и данных, коммерческой тайны организации. Определите процесс создания и использования выделенных учётных данных для ИИ-агентов и предоставления доступа к внутренним ресурсам. |



¹ <https://lp.kaspersky.com/ru/ai/>

✓ **Обеспечена ли наблюдаемость взаимодействий ИИ-агентов с облачными моделями и окружением?**

Трафик, передаваемый между агентом и моделью, может содержать персональные данные и коммерческую тайну, что создаёт риски компрометации и утечек. Внедрите LLM-шлюз как единую точку доступа к облачным LLM для обеспечения наблюдаемости и LLM Firewall для обнаружения и предотвращения передачи в облако персональных данных и конфиденциальной информации.

✓ **Обеспечена ли защита от LLM-специфичных угроз?**

Внешние данные, такие как скиллы, MCP-серверы, плагины или программные пакеты, могут содержать непрямые промпт-инъекции, вредоносную нагрузку и инструкции к агенту по скачиванию и запуску вредоносного кода. Используйте LLM Firewall для детектирования промпт-инъекций и EDR-решения для обнаружения подозрительной активности на хосте. Решения класса Managed Detection and Response помогут в круглосуточном мониторинге и реагировании на вредоносную активность.

✓ **Определён ли процесс безопасной разработки для ИИ-агентов?**

Код, изменённый или разработанный при использовании ИИ-агентов, — недоверенный и может содержать ошибки, уязвимости и вредоносные фрагменты, что создаёт риски атаки на цепочку поставок как для компании-разработчика, так и для её клиентов. Внедрите обязательные процессы код-ревью такого кода с участием человека-разработчика, статический анализ и проверки на уязвимости.

Эксперты «Лаборатории Касперского» готовы поделиться своим опытом по построению прозрачного и безопасного процесса применения агентов на базе больших языковых моделей как в процессах разработки, так и для более широкой ИИ-трансформации бизнеса. Свяжитесь с нами, если вас интересует консультация по безопасному внедрению ИИ-решений или пилотирование ИИ-специфичных решений для безопасности, таких как LLM Firewall.

Авторы исследования:

Владислав Тушканов

Руководитель группы исследования технологий машинного обучения «Лаборатории Касперского»

Александр Лискин

Руководитель управления исследования угроз «Лаборатории Касперского»

Игорь Кузнецов

Директор Глобального центра исследований и анализа угроз «Лаборатории Касперского»