Как внедрять MLSecOps: пять этапов практического road-map

Источник. Блог компании INFERA Security на HABR

https://habr.com/ru/companies/infera_security/articles/1067538/

MLSecOps переносит принципы DevSecOps на ИИ-системы. Он делает безопасность измеримой, воспроизводимой и встроенной в разработку и эксплуатацию, а не «прикрученной» в конце. В этой статье мы описываем практический road-map из пяти этапов, который позволяет последовательно выстроить этот контур: от видимости и политик к управляемому доступу, контролю кода, runtime-защите и полноценным AI Security Operations.

Этапы нельзя перепрыгивать. Каждый последующий опирается на результаты предыдущего. Расскажем про конкретные шаги, типы инструментов и метрики, по которым можно понять, что система действительно работает.

Этапы внедрения MLSecOps

Сроки ориентировочные: этапы 0–1 — 1–2 месяца, этап 2 — 2–3 месяца, этап 3 — 3–6 месяцев, этап 4 — непрерывно.

Каждый следующий этап опирается на предыдущий: нет смысла покупать Red Teaming, пока не видно, кто и как использует ИИ.

Рис. Этапы внедрения MLSecOps

Этап 0. Инвентаризация и политики (1–2 месяца)

Сначала ответьте на скучные вопросы: какие модели и ИИ-инструменты реально используются, где живут агенты, какие данные куда отправляются. Без этого шага любой инструмент будет охранять непонятно что.

Что сделать:

1. Собрать каталог ИИ-активов: модели, датасеты, промпт-шаблоны, агенты, MCP-серверы, интеграции, векторные базы. Это и есть AI-BOM (AI Bill of Materials).
2. Провести обследование (discovery) по рабочим станциям и инфраструктуре: конфиги ИИ-агентов, браузерные сессии внешних чат-ботов, ключи внешних провайдеров в файлах и переменных окружения. Как правило, находится заметно больше, чем ожидалось.

3. Зафиксировать политику допустимого использования: какие инструменты разрешены, какие классы данных можно обрабатывать, что маскируется, какие уязвимости блокируют релиз.
4. Построить модель угроз под ИИ-компоненты и назначить владельца ИИ-риска.
5. Зафиксировать точку отсчёта по модели зрелости, чтобы через полгода было с чем сравнивать.

Типы инструментов: генераторы SBOM/AI-BOM (CycloneDX, SPDX, Syft), моделирование угроз (OWASP Threat Dragon), рамки зрелости и управления риском (DSOMM, OWASP SAMM, NIST AI RMF, MITRE SAFE-AI / ATLAS), сканеры обнаружения ИИ-активов на конечных точках.

Этап 1. Управляемый доступ к LLM

Поднимите единый шлюз и переведите на него обращения из IDE, CLI и сервисов. Первый быстрый эффект – видимость: кто, куда и что отправляет. Второй – маскировка секретов и персональных данных до отправки во внешнюю модель. Третий – квоты токенов и список разрешённых моделей.

Что сделать:

1. Развернуть шлюз во всех трёх режимах включения в трафик и отсечь прямые исходящие соединения к внешним провайдерам моделей.
2. Включить детекторы секретов и персональных данных на запрос, маскирование до отправки и повторную проверку ответа.
3. Задать список разрешённых моделей и сервисов, разделить политики по командам и проектам.
4. Ввести квоты и бюджеты токенов на пользователя, команду и приложение, с алертами на аномалии.
5. Настроить экспорт событий шлюза в SIEM. Без этого шлюз останется «вещью в себе».

Типы инструментов: LLM-шлюз и обратный прокси, детекторы секретов и персональных данных, движок квот и бюджетов, runtime-guardrails, экспорт в SIEM.

Этап 2. Контроль кода

Подключите проверки в IDE (или другой точке интеграции с LLM), настройте риск-ориентированное ревью PR, определите правила блокировки в CI/CD.

Что сделать:

1. Раскатать IDE-плагины, интегрированные с LLM: настройте хук на проверку кода на уязвимости и интегрируйтесь со сканерами (либо используйте INFERA AI.SafeCode для этого).
2. Ввести в pull request отдельный сигнал «здесь ИИ-сгенерированный код» и повысить требования к ревью для критичных участков.
3. Поставить промпт-шаблоны под версионный контроль и заводить на их правки такое же ревью, как на код.
4. Настроить гейты в CI/CD: подтверждённые High/Critical, утечки секретов, критичные нарушения политик. Всё остальное – предупреждением.
5. Свести находки в ASOC/ASPM, чтобы дедуплицировать и приоритизировать, а не рассылать разработчикам семь отчётов из семи сканеров.

Типы инструментов: SAST/DAST, SCA, детекторы секретов, сканеры моделей и сериализованных весов, подпись артефактов, происхождение сборки (SLSA), агрегация находок (ASOC/ASPM), мониторинг зависимостей.

На этом этапе появляются первые метрики: доля ИИ-сгенерированного кода с находками, MTTR, повторяющиеся классы уязвимостей.

Этап 3. Runtime-защита и Red Teaming

Для продуктового ИИ включите детект инъекций и утечек в рантайме, ограничьте права агентов, настройте контроль действий и издержек. Именно здесь напрямую реализуются требования Agent Runtime Security и закрываются ключевые риски OWASP Top 10 for Agentic Applications (Tool Misuse, Identity & Privilege Abuse, Memory & Context Poisoning, Insecure Inter-Agent Communication, Rogue Agents и др.).

Что сделать:

1. Включить инспекцию обоих направлений на шлюзе: запрос и ответ.
2. Развернуть три слоя контроля агента: сканирование артефактов до загрузки, детерминированный допуск на вызове, контроль исходящего трафика и системных вызовов.
3. Раздать ИИ-агентам минимальные права, завести песочницы и обязательное подтверждение человеком для необратимых операций.
4. Внедрить брокер секретов: агент не должен видеть настоящие ключи.
5. Поставить предохранители на итерации, токены и частоту отказов, чтобы зацикленный агент не съел месячный бюджет за ночь.
6. Запустить регулярный ML Red Teaming и замкнуть его результаты на политики AI Firewall. Тестирование необходимо проводить после каждого дообучения модели и смены промпт-шаблонов, а не «по настроению».

Типы инструментов: AI Firewall и guardrails, ML Red Teaming, движок политик для агентов, прокси с DLP, сенсоры eBPF/ETW, песочницы исполнения, сканеры MCP-серверов и скиллов.

Этап 4. Операционный контур (непрерывно)

Сведите всё в AI Security Operations: сквозная видимость от промпта до действия агента, передача обогащённых событий в SOC, приоритизация по бизнес-рискам, SLA на исправление, регулярный аудит.

Что сделать:

1. Ввести сквозной идентификатор сообщения и сшивать по нему события шлюза, конечной точки и приложения в одну сессию. Стандарты уже есть: W3C Trace Context и семантические соглашения OpenTelemetry GenAI.
2. Настроить неизменяемый журнал вердиктов: цепочка хешей, ссылка на версию политики, подпись. Это то, что вы будете предъявлять на разборе инцидента.
3. Написать сценарии реагирования именно на ИИ-инциденты: аварийный останов агента, отзыв доверия к артефакту, ротация утёкшего ключа.
4. Настроить мониторинг дрейфа и аномалий потребления.
5. Регулярно пересматривать политики и прогонять их в пробном режиме на записанных трассах, прежде чем раскатывать.

Две оговорки

Во-первых, этапы нельзя перепрыгивать. Например, Red Teaming без инвентаризации даст красивый отчёт о системах, которыми никто не управляет.

Во-вторых, параллельно с этапом 1 стоит запустить программу security champions в командах разработки. Опыт DSOMM показывает, что без «своих» людей в командах контроль воспринимается как внешняя угроза и обходится.

Метрики, по которым видно, что система работает

Метрики нужны не для отчёта руководству, а чтобы понять, насколько хорошо выстроенная система работает в реальности.

Минимальный набор:

- доля обращений к LLM, проходящих через шлюз (цель должна быть близка к 100%);
- количество заблокированных отправок секретов и персональных данных во внешние модели в месяц;
- доля ИИ-сгенерированного кода, прошедшего проверки до коммита;
- MTTR по Critical/High и доля повторяющихся классов уязвимостей;
- результаты регулярного Red Teaming: сколько атак из прошлого прогона теперь блокируются (прямая проверка того, что цикл замкнут);
- доля вызовов агентов, ушедших на ручное одобрение (если показатель растёт, значит политика слишком широкая, и одобряющие решения сотрудники скоро начнут штамповать не глядя);
- расход токенов по командам и аномалии потребления.

Частые ошибки внедрения

Собрали то, на чём спотыкаются чаще всего.

Начать с инструмента, а не с инвентаризации. Самая дорогая ошибка. Просто купленный сканер закрывает 5% поверхности, о которой вы знали, и не увидит 95%, о которой не знали.

Поставить вероятностный детектор на базе LLM в линию исполнения. Задержка, невоспроизводимые вердикты и невозможность объяснить на разборе, почему в среду запрос прошёл, а в четверг уже нет. Дорогие семантические проверки должны идти асинхронно.

Fail-open «на время пилота». Режим, который включают на неделю, а выключают через год после инцидента. Если при падении контура трафик идёт напрямую, то контура у вас нет.

Усталость одобрений. Если в очередь ручного одобрения валится всё подряд, через две недели её начнут прокликивать не глядя. В очередь должны попадать только неоднозначные случаи.

Тестировать модель один раз перед запуском. Дообучение и правка системного промпта меняют поведение модели не меньше, чем рефакторинг меняет поведение кода.

Считать, что MLSecOps заменит базовую гигиену. Это надстройка над зрелой ИБ, а не альтернатива ей. Если в организации не наведён порядок с секретами и управлением доступом, ИИ просто добавит новых способов этим воспользоваться.

Как это устроено у нас

Мы собрали базовый контур безопасности для реализации перечисленных требований. Логика простая: три модуля закрывают три уровня контроля – обращение к модели, код и действие агента. Все модули работают под единой моделью политик.

INFERA AI.Firewall – контроль обращений к моделям

Единая точка контроля всех обращений к LLM. Включается в трафик всеми возможными способами: Guardrails API для внутренних приложений, LLM Gateway для агентов в IDE и пользователей через OpenWebUI и прозрачным forward-прокси с TLS-инспекцией для браузерного Shadow AI.

Что делает: фильтрует запросы и ответы, блокирует промпт-инъекции, маскирует персональные данные и секреты до отправки, держит квоты токенов на уровне пользователя, агента, приложения или команды, перераспределяет задачи между моделями с учётом свободных лимитов. Маскировщик персональных данных построен на распознавании именованных сущностей по 64 классам плюс регулярные выражения.

Встроенный сканер ML Red Teaming замыкает цикл: найденная уязвимость сразу превращается в правило защиты, после чего прогон повторяется. Сканер может использоваться и для проверки самих LLM-моделей моделей до их допуска в контур.

INFERA AI.SafeCode – контроль кода

Закрывает контроль кода в IDE и в конвейере: единое информационное пространство в DevSecOps/MLSecOps-контуре, подтверждение уязвимостей доказательствами прямо в IDE и автоматическое превращение high/critical-находок в задачи разработчику с рекомендациями по исправлению. Прямая интеграция с ИИ-агентом для исправления найденных уязвимостей в режиме написания кода.

Для сценария с ИИ-ассистентом здесь важны три вещи: скан и маскирование промпта до отправки, инжект guardrail-директив в контекст ассистента и репо-скан на секреты до коммита. То есть контроль срабатывает в момент, когда разработчик собирается отправить лишнее, а не когда это вся чувствительная информация уже отправилась наружу.

INFERA AI.SafeAgent – контроль действий агента

Это внешний контур принудительного контроля: он видит каждое действие агента, оценивает его по единой политике и останавливает опасное до исполнения не требуя правки кода агента. Над ними фактически формируется решающий слой: реестры артефактов и агентов, компилятор политик, корреляция событий по сквозному идентификатору сессии, очередь ручных одобрений, неизменяемый аудит и консоль SOC.

Что из этого важно на практике:

- подключение без правки кода агента как через API для сетевого контроля, так и на хосте;
- дорогие контентные проверки уходят на границу сети, где используются AI.Firewall, на хосте всё проверяется быстрыми политиками;
- брокер секретов включен прямо в AI.SafeAgent на хосте. Благодаря этому агент видит заглушку, в то время как настоящий ключ подставляется на границе исходящего трафика;
- работа в изолированной среде, когда весь контур функционирует полностью офлайн, без скрытой телеметрии.

MLSecOps – это не модная надстройка, а перевод ИИ-разработки на рельсы инженерной дисциплины: понятные активы, угрозы, ответственность и измеримый риск.

Наследуйте из DevSecOps всё, что наследуется: гейты, ревью, мониторинг, security champions. Достраивайте то, чего в нём нет: шлюз, AI-BOM, Red Teaming, контроль агентов. Идите по этапам: от видимости к контролю, от контроля к операционному контуру.

И держите в голове главное про ИИ-агентов: устойчивость модели к инъекции – это свойство модели, а не вашей системы, и рассчитывать на неё нельзя. Гарантию нужно строить на уровне контура так, чтобы опасное действие останавливалось до исполнения независимо от того, что модель себе «надумала».

Компании, которые пройдут этот путь системно, получат не просто «галочку» в отчёте по безопасности, а возможность внедрять ИИ на основе измеримой зрелости и масштабировать агентные технологии без пропорционального роста рисков.