От сотен алертов к доказанным уязвимостям: как эволюционирует DevSecOps при объединении 7 сканеров в единый пайплайн

Источник. Блог компании INFERA Security на HABR

https://habr.com/ru/companies/infera_security/articles/1061862/

Классический подход к Application Security сегодня переживает кризис. Интеграция разрозненных SAST, DAST и SCA-инструментов в CI/CD часто приводит к одному и тому же результату: разработчики тонут в сотнях «критических» алертов, до 80% из которых – это false positives и информационный шум. AppSec-команды тратят недели на ручной триаж, а уязвимости исправляются уже после сборки или, что хуже, после инцидента.

Ситуацию усугубляет бум AI-помощников (GitHub Copilot, Cursor, Claude). Разработка ускорилась кратно, но вместе с ней масштабировалась и генерация небезопасного кода. Разработчик теперь может сгенерировать за час то, что раньше писал день, при этом уязвимости тоже начинают появляться с такой же скоростью. Если процесс безопасности остался прежним, AppSec быстро превращается в узкое место.

В этой статье мы разберем архитектуру и механику работы INFERA AI.SafeCode – платформы непрерывного анализа кода, которая отказывается от концепции «просто показать список подозрений» в пользу автоматического доказательства уязвимостей и MLSecOps-подхода.

Статья продуктовая, но мы ее публикуем на HABR не как рекламную, а как концептуальную. Хотим показать, как меняются подходы к безопасной разработке и насколько неэффективным становится «разрозненный» AppSec в эпоху ИИ и вайб-кодинга.

Ведь современная поверхность атаки стала шире. Уязвимость может быть не только в вашем коде, а в open-source библиотеке, а секрет может лежать в .env, тестовом конфиге или старом коммите. Ошибка авторизации может проявиться только в работающем API, когда один и тот же endpoint вызывается под разными ролями. SSRF может подтверждаться только через OOB-callback, а бизнес-логика вообще часто не выглядит как классическая уязвимость для статического анализатора.

Формально AppSec-инструменты у большинства есть: отдельно SAST, отдельно SCA, secret scanner, DAST, fuzzing и API security тоже отдельно. В результате команда разработки получает несколько независимых списков алертов, которые плохо связываются между собой. **И главная проблема не в отсутствии сканеров, а в отсутствии единой картины рисков и доказанной приоритезации задач.**

Главная архитектурная проблема традиционного AppSec – это изоляция инструментов. SAST не знает, что делает DAST, а SCA не учитывает контекст вызовов.

Архитектура: 7 движков под одним капотом

INFERA AI.SafeCode – платформа для анализа исходного кода и open-source компонентов с целью обнаружения уязвимостей и аномалий в режиме реального времени, внедрения подхода DevSecOps / MLSecOps в процессы разработки ПО. Помогает выявить и предотвратить уязвимости ещё на стадии проектирования и кодирования, а не находить их после развертывания и сборки ПО.

Решение объединяет 7 сканеров в единый DevSecOps / MLSecOps-контур, находит потенциальные уязвимости и подтверждает их доказательствами:

1. **SAST** – анализ исходного кода в IDE и CI/CD в реальном времени;
2. **SCA** – анализ open-source компонентов и транзитивных зависимостей;
3. **Secrets** – поиск и верификация утечек секретов;
4. **DAST** – динамическое тестирование работающего приложения;
5. **Pentest** – AI-агент для имитации ручного пентеста в песочнице;
6. **Code Fuzzing** – поиск crash-уязвимостей и edge-cases;
7. **API Fuzzing** – глубокий анализ REST, GraphQL, gRPC и WebSocket.

Платформа нормализует и дедублицирует их результаты, формируя единый реестр рисков по коду, API и зависимостям. Автоматически распределяет high/critical-уязвимости в понятные задачи для разработчика с объяснениями, доказательствами и рекомендациями по исправлению.

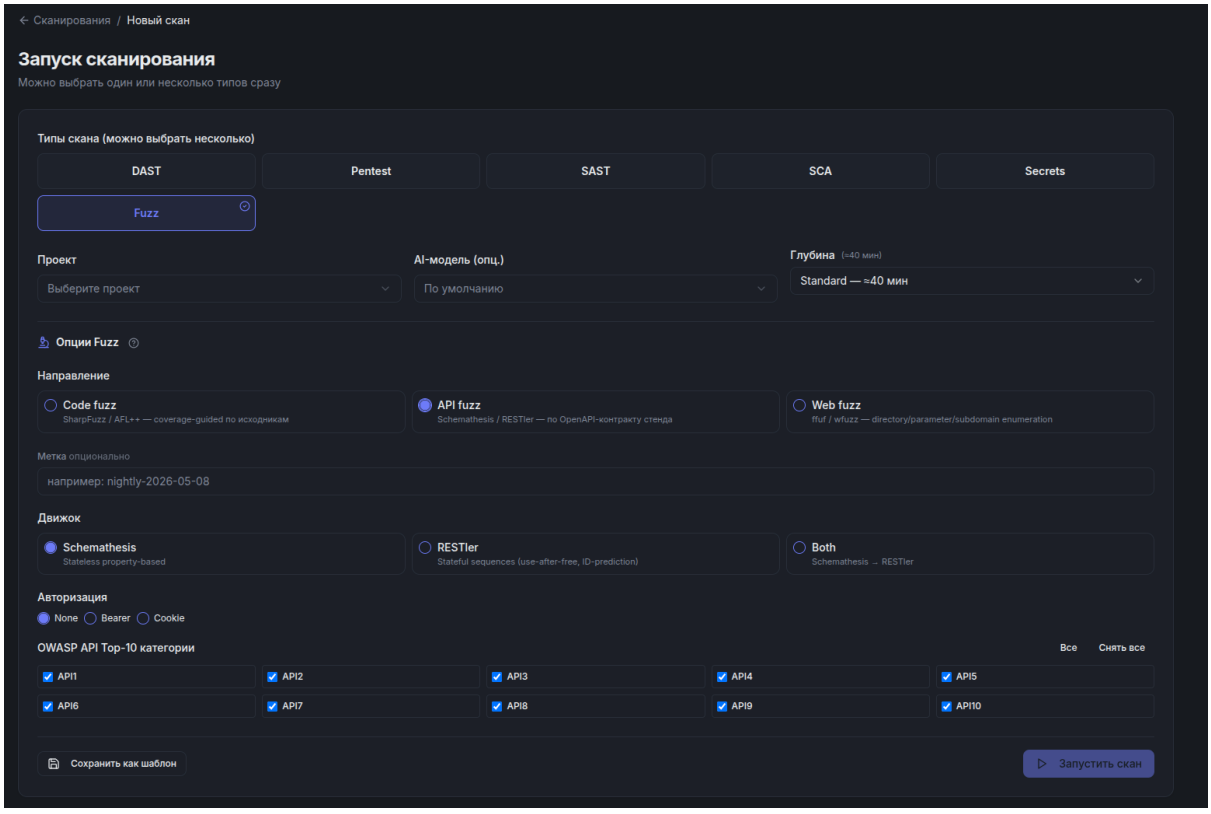

Рис.1 INFERA AI.SafeCode. Семь сканеров в одной платформе.

Как сканеры работают в связке

Платформа реализует различные связанные сценарии, которые трудно воспроизвести набором разрозненных инструментов. Обычный сканер находит потенциальную проблему и делегирует проверку человеку. INFERA меняет эту парадигму за счет кросс-валидации между движками. Рассмотрим на примере.

От SAST к Fuzzing

Если статический анализ (SAST) находит подозрительное место (например, потенциальную SQL-инъекцию), INFERA автоматически собирает fuzzing-каркас под конкретную функцию. Далее в дело вступает Code Fuzzing: он начинает бомбардировать функцию мутированными данными. Если происходит краш или эксплуатация – разработчик получает неопровержимое доказательство (proof-of-exploit) со стектрейсом. Если уязвимость нельзя подтвердить, она не блокирует pipeline и не отвлекает команду.

От DAST к Pentest

Динамический сканер (DAST) находит ошибки авторизации (например, IDOR), заходя в приложение под разными ролями и сравнивая ответы сервера. Затем подключается AI Pentest-агент. Используя данные от SAST (понимание структуры кода) и DAST, агент в изолированной песочнице (sandbox) строит граф атаки глубиной до 12 шагов, пытаясь развить уязвимость до RCE или выгрузки базы данных, имитируя действия реального злоумышленника.

В итоге, SAST может найти потенциально опасный sink в коде. DAST может показать, что до этого участка реально можно добраться через endpoint. Fuzzing может подтвердить crash. API Fuzzing покажет, что endpoint отдаёт чужие данные при вызове под другой ролью.

В итоге команда видит не семь разрозненных предупреждений, а одну подтверждённую проблему с доказательством, контекстом и рекомендацией по исправлению. Это важный сдвиг: от «у нас есть алерт» к «у нас есть воспроизводимая уязвимость с доказательством риска».

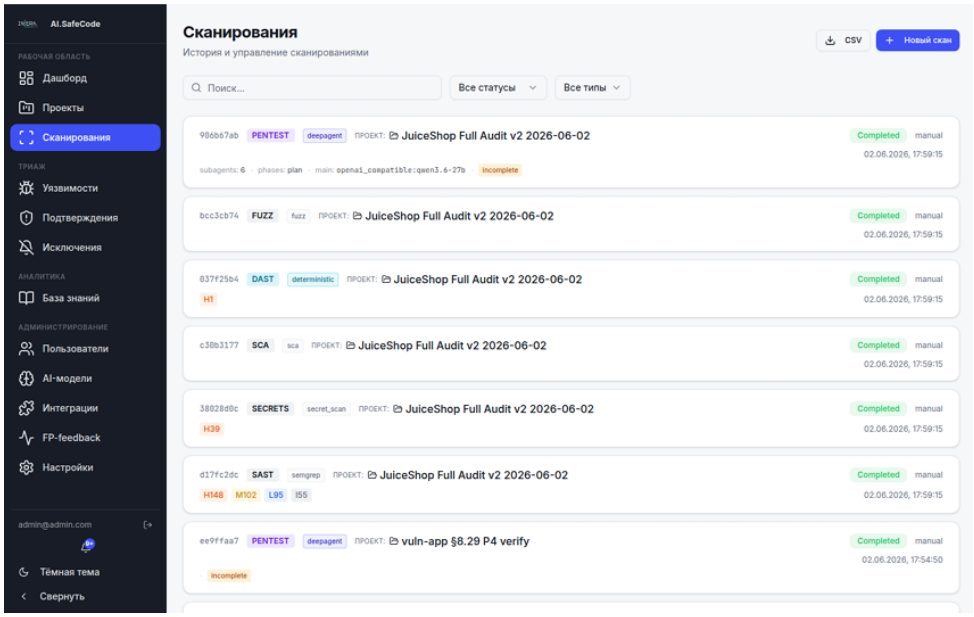

Рис. 2. INFERA AI.SafeCode. Результаты сканирования.

Security как часть IDE, а не блокер релиза

Безопасность должна работать со скоростью разработки. INFERA смещает фокус влево (Shift-Left) до максимума – прямо в момент написания кода:

- **проверка на лету:** анализ кода осуществляется в момент его написания сразу в IDE, найденная проблема подсвечивается до коммита;
- **autoFix:** к каждой подтвержденной уязвимости прилагается не просто абстрактная ссылка на CWE, а конкретная рекомендация и кнопка «Применить AutoFix»;
- **интеграция с Git:** в GitLab/GitHub комментарии от платформы появляются на проблемных строках сразу с готовым pull-request на исправление;
- **контроль vibe coding:** валидация кода, написанного ИИ-помощниками, предотвращение попадания ИИ-галлюцинаций и уязвимостей в кодовую базу.

Если уязвимость нельзя подтвердить, она не блокирует pipeline как High или Critical. Она может остаться в системе как неподтверждённая, но не отвлекает команду так же, как доказанный RCE или валидный production-токен. Это снижает шум и меняет отношение разработчиков к security-проверкам. Когда pipeline падает только по подтверждённым рискам, блокировка воспринимается не как каприз сканера, а как инженерный факт.

Большая часть современной бизнес-логики живёт в API. Именно там часто появляются ошибки, которые не выглядят как классические SQLi или XSS, но приводят к серьёзным инцидентам: пользователь получает чужой объект, может изменить чужой заказ, видит лишние поля, меняет роль через PATCH-запрос или обходит ограничение на уровне frontend.

В INFERA AI.SafeCode **сканер API Fuzzing** анализирует REST, GraphQL, gRPC и WebSocket API. Платформа может использовать OpenAPI-спецификации, GraphQL schema, .proto, AsyncAPI и другие описания интерфейсов. На их основе строятся stateful-сценарии: авторизация, создание объекта, получение, изменение, удаление, повтор запроса под другой ролью, проверка обязательных и лишних полей, граничные значения, malformed payloads, JWT bypass, mass assignment.

Это тоже важно для AI-сгенерированного кода. ИИ хорошо генерирует типовые CRUD-обработчики, но часто не знает реальные правила доступа конкретной системы. Функционально endpoint может работать идеально, но с точки зрения безопасности быть открытой дверью.

Open-source зависимости сегодня являются частью продукта не меньше, чем собственный код. INFERA AI.SafeCode выполняет SCA-анализ open-source компонентов, выявляет известные CVE, помогает оценивать риски зависимостей и формировать картину supply chain. Для разработки ПО, поставляемого на объекты КИИ, такая проверка особенно важна: нужно понимать, какие компоненты используются, какие из них уязвимы, какие версии требуют обновления и где есть риск использования небезопасного или неподдерживаемого пакета.

И здесь также важна приоритезация. Если уязвимый код не вызывается из приложения, приоритет один. Если есть путь от пользовательского ввода до уязвимой функции – совсем другой. Поэтому простого списка CVE недостаточно: **нужна связь между зависимостью, кодом приложения и реальной достижимостью.**

Поиск секретов давно стал обязательной частью SDLC, но и здесь есть проблема «шума». В репозиториях часто находятся тестовые ключи, примеры из документации, старые токены, фейковые значения и реальные production-секреты. Для AppSec и DevOps это принципиально разные ситуации.

INFERA AI.SafeCode ищет секреты в коде и конфигурациях, а также может проверять их валидность. Если найденный токен действительно работает, это уже не «потенциальная проблема», а инцидент, который требует немедленного отзыва ключа, ротации и расследования. На практике это помогает отличать информационный мусор от реального риска.

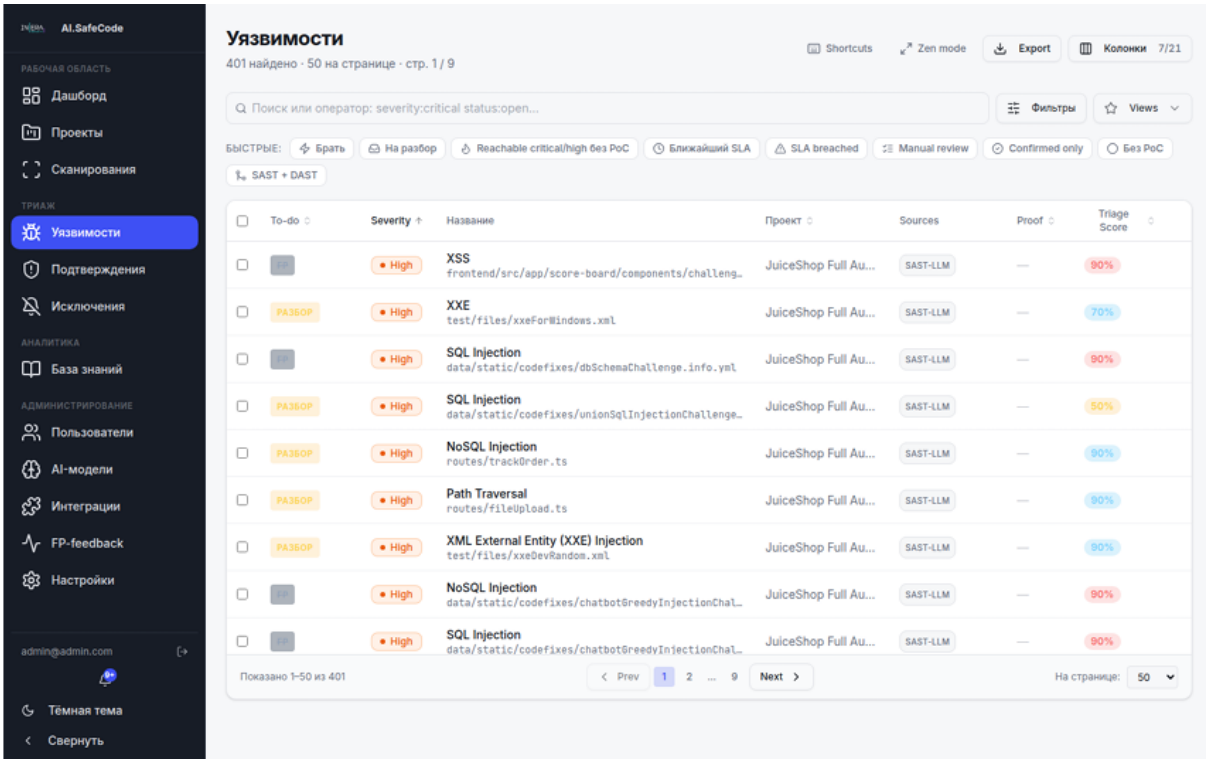

Рис. 3. INFERA AI.SafeCode. Единый список уязвимостей по всей организации с сортировкой и фильтрами.

Поверхность атаки и измеримый риск

В INFERA AI.SafeCode центральная сущность – это **проект**, внутри которого объединяются репозитории, стенды, сканы, политики безопасности, находки, метрики, поверхность атаки и история изменений.

Это удобно по двум причинам.

Во-первых, разработчик и AppSec смотрят на один и тот же объект, а не на разные отчёты из разных систем. Во-вторых, риск оценивается не по отдельному скану, а по состоянию проекта в целом: какие уязвимости открыты, какие подтверждены, какие нарушают SLA, какие эндпоинты связаны с опасными sink, какие зависимости требуют обновления, какие секреты валидны, какие проверки сейчас выполняются.

Сканы можно запускать вручную или автоматически, управлять их жизненным циклом, ставить на паузу, отменять, перезапускать, настраивать глубину, движки, пресеты и шаблоны. Результаты нормализуются и дедублируются в единый список находок.

Хронологический лог сканирования с таймстемпами сохраняется и доступен после завершения.

SafeCode показывает актуальный срез по проектам: сколько открыто Critical и High, где нарушен SLA, какие сканы идут сейчас, какие проекты требуют внимания, как меняется MTTR (Mean Time to Remediation), как часто запускаются проверки, какие классы уязвимостей повторяются чаще всего.

Это помогает перейти от режима «раз в месяц выгрузили отчёт» к постоянному управлению риском. AppSec видит, где накапливается долг. Руководитель разработки понимает, какие команды стабильно исправляют проблемы, а где security-дефекты возвращаются снова. CISO получает не только технический список уязвимостей, но и картину процесса.

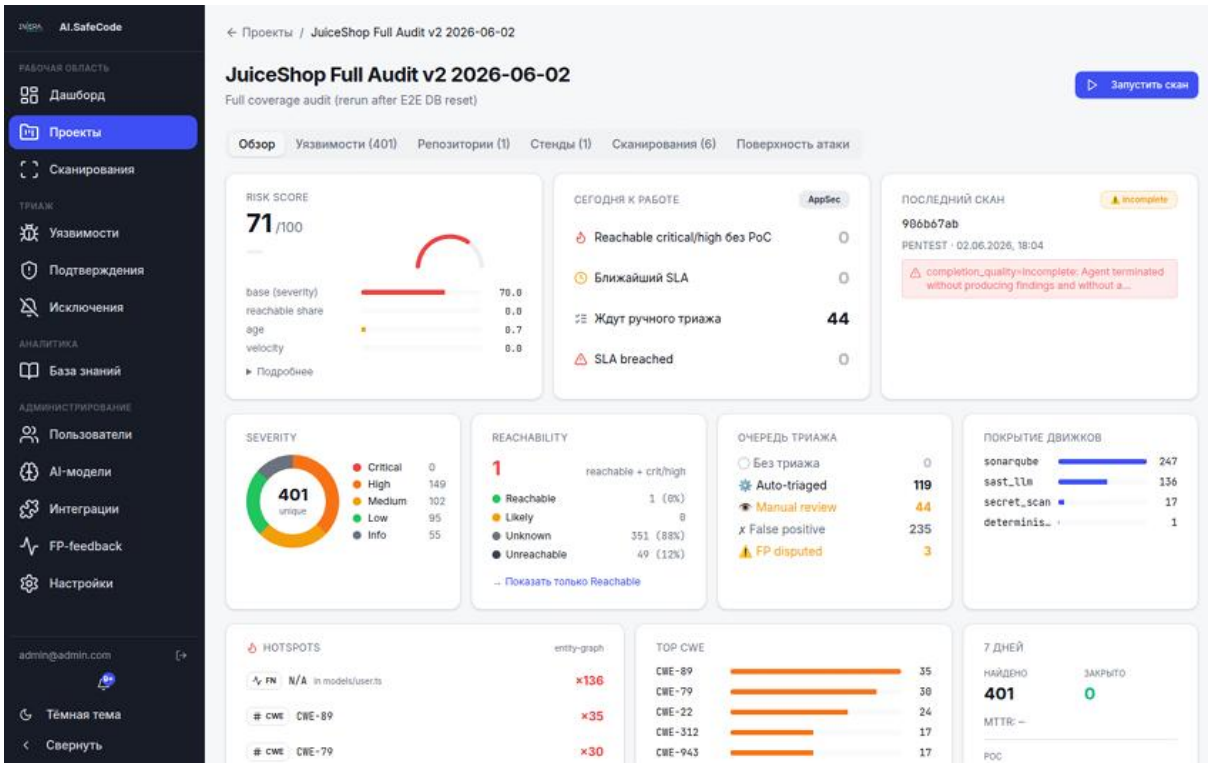

Рис.4. INFERA AI.SafeCode. Аналитика по проектам.

Поверхность атаки – это еще один отдельный слой платформы, который показывает не только список уязвимостей, но и то, где приложение реально доступно и какие точки входа связаны с рисками.

В одном месте можно увидеть recon-данные по проекту: субдомены, открытые порты, технологии, эндпоинты, сертификаты, публично доступные внутренние сервисы, цепочки преобразования entry к sink, кластеры по CWE-семьям и количество открытых находок на каждом эндпоинте.

Для DevOps это список того, что нужно закрыть или ограничить. Для AppSec это карта достижимости уязвимых участков. Для CISO это более понятная картина того, как код, API и инфраструктура складываются в реальную поверхность атаки.

Для специалистов по информационной безопасности платформа предлагает переход от ручного триажа к управляемому процессу:

1. **Картирование поверхности атаки (Attack Surface)**

Платформа строит полную карту приложения: точки входа (entry points), цепочки от entry к sink, кластеризацию по семействам CWE.

Благодаря DAST и recon-модулям, формируется сводка по доступным субдоменам, открытым портам и API-эндпоинтам. AppSec-инженер видит, какие именно эндпоинты ведут к уязвимым участкам и достижимы ли они в реальности.

2. **Единая картина рисков и метрики**

Вместо разрозненных отчетов – агрегированный дашборд. CISO видит актуальный срез по всем проектам:

- обнаруженные открытые порты внутренних сервисов (сразу формируется таск для сетевиков/DevOps);
- статистику по MTTR (Mean Time to Remediation) и нарушениям SLA;
- сжатый список дедублицированных уязвимостей с доказанной приоритизацией.

3. **Compliance и On-Premise**

Для кровавого энтерпрайза и объектов КИИ реализована возможность Self-hosted развертывания. Система поддерживает строгий Enterprise-контроль доступа, возможность загрузки собственной базы уязвимостей заказчика и обеспечивает соответствие требованиям регуляторов (Приказы ФСТЭК №117, №21, ГОСТ Р 56939-2024, ФЗ-152).

4. **Поддерживаемый стек**

Архитектура платформы адаптирована под современную разработку.

INFERA AI.SafeCode обнаруживает следующие типы уязвимостей: инъекция команд ОС, секреты в коде, отсутствие защиты от XSS, SQL- инъекции, командные инъекции, некорректная обработка входных данных, раскрытие чувствительной и конфиденциальной информации, использование MDS для хеширования, слабая криптографическая стойкость и/или шифрование, обработка файлов и т.д.

Реализована широкая поддержка языков программирования: Python, JavaScript/TypeScript, Java, C#/.NET, Go, Rust, PHP, Ruby, C/C++, Swift и других.

MLSecOps: когда безопасность нужна не только коду приложения

Термин MLSecOps становится актуальным там, где компании разрабатывают собственные решения на базе ИИ: модели, пайплайны обучения, конфигурации, API для inference, обработку датасетов, интеграции с LLM, prompt-шаблоны и агентные сценарии.

В таких проектах уязвимости могут быть не только в коде. Риски появляются в конфигурациях моделей, обработке входных данных, API вокруг inference, секретах в notebook, зависимостях ML-библиотек, небезопасных пайплайнах и доступах к данным.

INFERA AI.SafeCode подходит для такого сценария как часть MLSecOps-контура: анализирует код, конфигурационные файлы, зависимости, API и работающие сервисы,

а также помогает встроить проверки в привычный SDLC. Но об этом мы расскажем в нашей следующей статье.

Итог

Код пишется непрерывно, зависимости обновляются постоянно, API меняются быстро, AI-ассистенты ускоряют разработку, а злоумышленники не ждут окончания квартального аудита.

В такой среде AppSec должен работать с той же скоростью, что и разработка. Но это невозможно, если команда живёт в режиме ручного разбора бесконечных алертов из разрозненных инструментов.

Платформа INFERA AI.SafeCode предлагает другой подход и дает разработчикам, AppSec и CISO управляемость:

- разработчик видит, что именно нужно исправить;
- AppSec работает с подтверждёнными рисками, а не с шумом;
- CISO получает картину состояния безопасности,
- CTO – возможность ускорять разработку без неконтролируемого роста технического и security-долга.