Пример презентации проекта

Интеграция анализа безопасности кода в конвейер разработки: от локальной автоматизации к стандартизации DevSecOps

Резюме проекта (Executive Summary)

Успешная реализация полностью интегрированной системы контроля безопасности и качества кода, объединяющей анализ исходного кода, автоматизацию сборки и управление задачами в единый workflow. Проект демонстрирует способность адаптировать критичное решение под нетиповые требования заказчика, сохраняя высокий уровень безопасности и надежности инфраструктуры.

ROI расчет (3-летний период):

Затраты на внедрение:

- Лицензия анализатора кода (3 года): 7 млн. руб.
- Разработка интеграции с SonarQube/GitLab/Jenkins (3 месяца × 2 инженера): 600 тыс. руб.
- Обучение команды (10 дней работы): 100 тыс. руб.

Итого: 7,8 млн руб.

Экономия за 3 года:

- Снижение затрат на аудит: 450 × 0.9 × 3 = 1,215 млн руб.
- Избежание инцидентов ИБ (1 инцидент через уязвимость в коде): от 3 млн руб.
- Ускорение TTM (30% ускорение × 50 релизов/год × 3 года × 100 тыс./день): 23,5 млн руб.

Итого: 24,715 млн руб + устранение инцидентов

- ROI =

Меры по обеспечению ИБ информационных систем заказчика

Меры по обеспечению ИБ

Анализ защищенности ИС и ПО

Построение подсистемы защиты информации

Анализ защищенности

Анализ защищенности ПО
(объект оценки: ПО)

10110
00110

Поиск новых уязвимостей в разработанном самостоятельно или с привлечением подрядчика ПО (включая статический анализ кода)

Анализ защищенности ИС
(объект оценки: ИС)

Поиск известных уязвимостей в компонентах используемого в инфраструктуре стороннего ПО

Ключевые задачи заказчика

Проблема: Заказчик (финансовая организация) использовал разрозненные инструменты для проверки качества кода, без интеграции анализа безопасности в CI/CD-конвейер. Разработчики работали с разными интерфейсами, увеличивалось время вывода обновлений в production, резко возрастали риски информационной безопасности при работе с чувствительными финансовыми данными.

- Выстроить процесс контроля безопасности кода приложений на базе уже существующего контроля качества кода
- Обеспечить интеграцию анализатора кода в сборочную инфраструктуру

Критерии выбора решения

1. Высокая точность обнаружения уязвимостей в мобильном и веб-ПО, разработанном на Java-подобных и JavaScript-подобных языках
2. Возможность интеграции со сборочной инфраструктурой и инструментальным средством контроля качества кода
3. Удобство выполнения сканирования кода как в ручном, так и автоматическом режимах в приемлемые сроки

Вызов первый

Необходимость разработки дополнительного функционала

Необходимо было оперативно разработать дополнительный модуль интеграции с инструментальным средством анализа качества кода SonarQube.

Реализация нового функционала в кратчайшие сроки

После получения запроса на доработку возможности новой интеграции «из коробки» мы провели следующую работу:

- Оценили, насколько данная функция будет востребована на рынке
- Подготовили проект доработок Solar appScreener по запросу заказчика
- Через 3 месяца реализовали и начали использовать новую функцию

Вызов второй

Внедрение Solar appScreener в условиях локдауна

- Планировавшаяся командировка для выезда в страну заказчика отменилась из-за локдауна.
- Поставленные задачи пришлось решать удаленно.

Организация удаленной работы

Удаленная работа над проектом стала возможна благодаря:

- удобному интерфейсу и интеграции «из коробки»;
- подробной англоязычной документации по эксплуатации и развертыванию решения;
- проведению регулярных встреч по ВКС;
- проработке единичных сложных моментов через удаленное подключение.

Вызов третий

Необходимость вынесения БД на отдельный сервер

В соответствии с внутренними правилами заказчика от него был запрос на размещение БД, используемой в составе Solar appScreener, на отдельном сервере.

Гибкая настройка размещения компонентов Solar appScreener согласно требованиям заказчика

Несмотря на нетиповую конфигурацию заказчика, особенности нашего решения, разработанного на базе микросервисной архитектуры, позволили развернуть его компоненты на разных серверах, чтобы обеспечить его стабильную работу.

Вызов третий

Необходимость вынесения БД на отдельный сервер

В соответствии с внутренними правилами заказчика от него был запрос на размещение БД, используемой в составе Solar appScreener, на отдельном сервере.

Гибкая настройка размещения компонентов Solar appScreener согласно требованиям заказчика

Несмотря на нетиповую конфигурацию заказчика, особенности нашего решения, разработанного на базе микросервисной архитектуры, позволили развернуть его компоненты на разных серверах, чтобы обеспечить его стабильную работу.

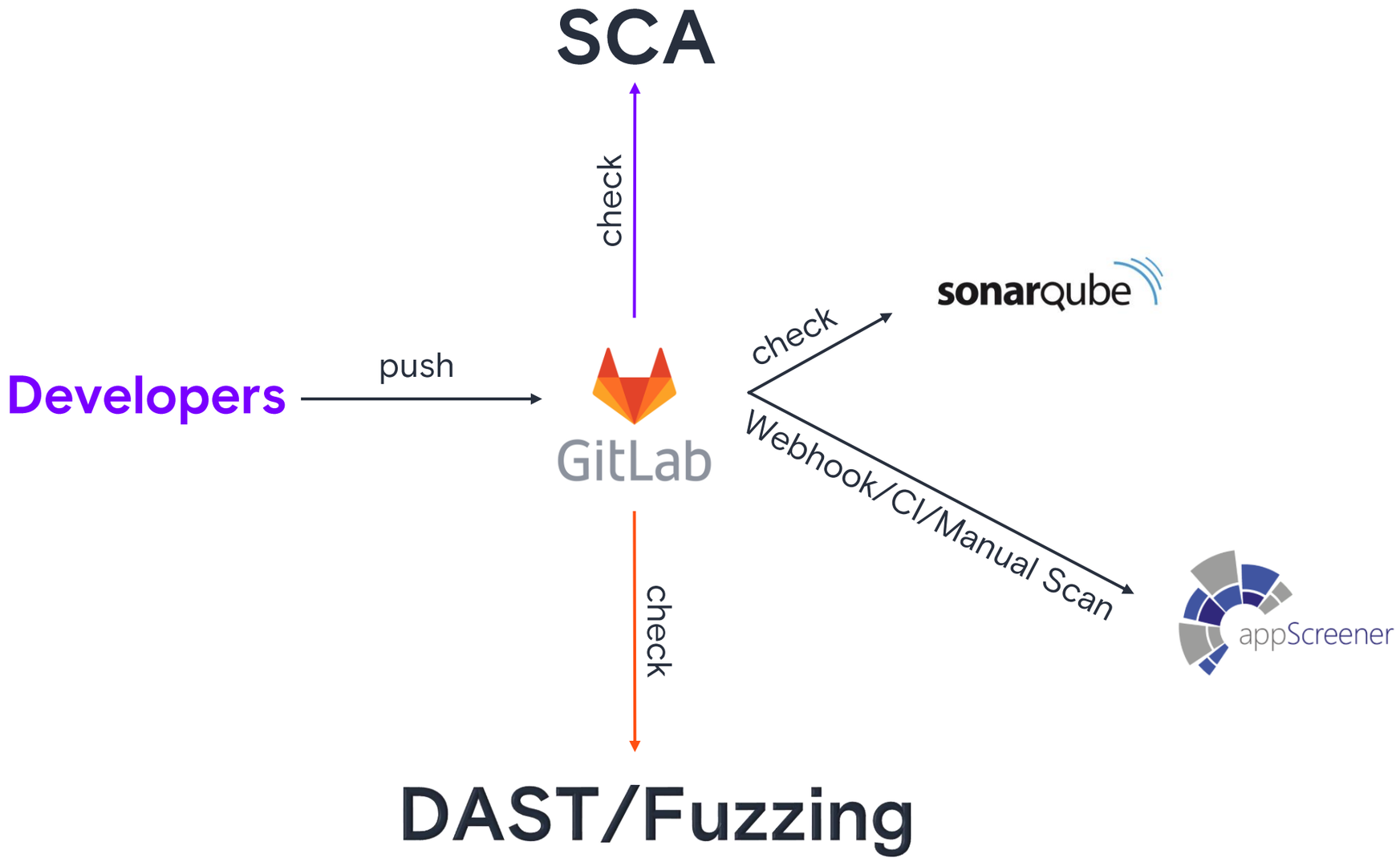
Краткие выводы и результаты

Вызов	Решение	Результат
Требование интеграции с SonarQube «из коробки» не существовало	Разработали модуль интеграции за 3 месяца и оценили его востребованность на рынке	Новая функция внедрена, доступна для других клиентов
Невозможность командировки из-за локдауна	Организовали полностью удаленное внедрение с регулярными ВКС и поддержкой через remote access	Проект реализован по плану, без задержек
Нетиповая конфигурация размещения БД	Использовали гибкость микросервисной архитектуры для разделения компонентов на разные серверы	Конфигурация работает стабильно в production

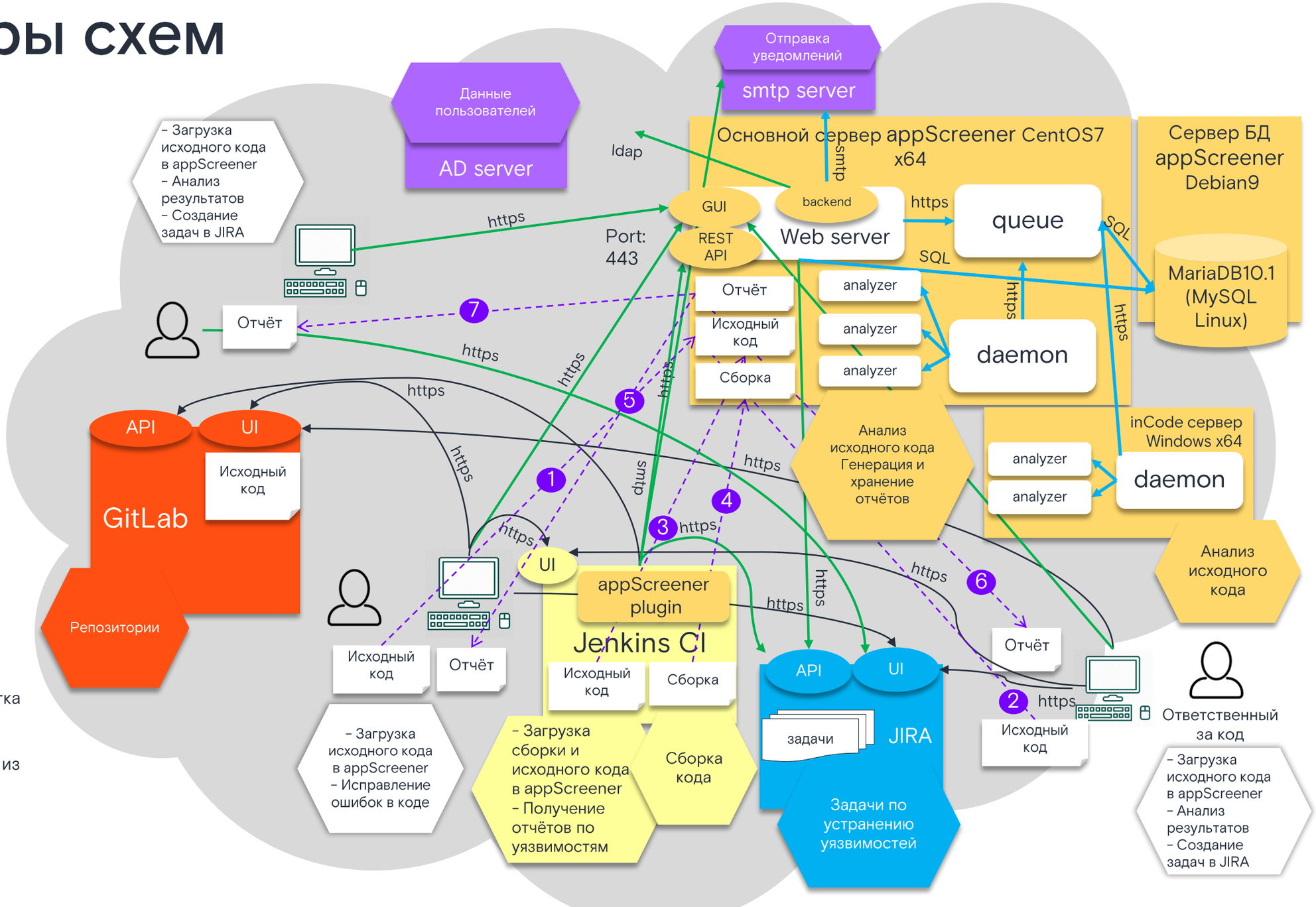
Краткие выводы и результаты

Метрика	До внедрения	После внедрения	Улучшение	Доказательство
Время обнаружения уязвимостей	2-3 недели (ручная проверка)	2-5 минут (автоматическое сканирование при коммите)	-99.5%	Логи GitLab/Jenkins, отчеты анализатора кода
Затраты на ручной аудит кода в год	500 тыс. руб. (2 security-специалиста на 50% ставки)	50 тыс. руб. (администрирование инструмента)	-90%	Зарплатные ведомости, счета на оплату труда
Количество инцидентов ИБ в год	8-12 (критичные уязвимости в production)	0-1 (отловлены на этапе разработки)	-90%	Отчеты SOC, логи инцидентов, tickets в JIRA
Среднее время исправления уязвимости	14 дней (выявление → реш → деплой)	3 дня (оповещение → исправление → регресс-тест)	-79%	История коммитов, time-tracking в JIRA
TTM (Time-to-Market) для обновлений	30 дней	14 дней	-53%	Лог релизов, метрики развертывания в production
Покрытие кода автоматическим анализом	10% (точечные проверки)	100% (каждый коммит)	+900%	Отчеты SonarQube, статистика по веткам Git
Количество критичных уязвимостей в production	3-4 в год	0	-100%	Логи систем мониторинга, отчеты пентестов
Средняя стоимость исправления одной уязвимости	250 тыс. руб. (человеко-часы + простой)	15 тыс. руб. (автоматическое исправление + тестирование)	-94%	Сметы проектов, счета на исправление инцидентов

Примеры схем



Примеры схем



Взаимодействие ИС происходит с помощью встроенных в продукты протоколов/API, разработка которых не требуется. Система доступна только из локальной сети.

Итоги проекта

Выстроенный процесс контроля безопасности кода, интегрированный в сборочную инфраструктуру заказчика

5

языков программирования

3

месяца работы над проектом

1

нетиповая конфигурация анализатора кода

Интегрировали Solar appScreener с SonarQube

Благодаря этому стало возможно контролировать безопасность и качество кода через единый интерфейс

Интегрировали Solar appScreener с GitLab

Это позволило автоматически забирать исходные коды на анализ согласно требованиям заказчика

Интегрировали Solar appScreener с Jenkins

Теперь можно автоматически анализировать исходные коды всего проекта на финальном этапе его сборки

Дополнительные материалы

Ссылки на материалы
https://habr.com/ru/companies/solarsecurity/articles/533828/
https://habr.com/ru/companies/solarsecurity/articles/533828/
https://habr.com/ru/companies/solarsecurity/articles/533828/

Доп. Материалы:

data (11).xlsx

data (11).xlsx

data (11).xlsx