Apsafe

DevSecOps за 10 дней

Apsafe — безопасность, которая
не мешает разработке

apsafe.ru

Что сегодня обсудим

- Почему большинство команд страдает от DevSecOps
- Каким должен быть «идеальный» процесс
- Как Apsafe снимает ключевые боли
- Две реальные истории успеха
- Подробнее про Apsafe

Спикеры

Илья Новойдарский

Эксперт

Руководитель группы инновационных решений направления безопасной разработки УЦСБ

Евгений Тодышев

Спикер

Руководитель направления безопасной разработки УЦСБ

Анастасия Камалова

Спикер

Пресейл инженер направления безопасной разработки УЦСБ

Виктор Тимашков

Спикер

AppSec-аналитик группы инновационных решений направления безопасной разработки УЦСБ

эксперт в области DevSecOps

Илья Новойдарский

Руководитель группы инновационных решений
направления безопасной разработки УЦСБ

01

Почему команды страдают от DevSecOps

Долго, дорого сложно

Долгое внедрение

3–6 месяцев на внедрение классических сканеров

Высокие затраты

Рост затрат пропорционален росту разработки

Ложные срабатывания

Сканеры генерируют тысячи ложных срабатываний за одно сканирование

Кадровый голод

Требуется команда из 4-х человек для работы процесса, включая редкого AppSec-аналитика

Проблема №1
Нам некому и некогда

Найм своей команды

От 4-х человек, срок найма от 6 месяцев,
в среднем 6 млн руб. в год на человека

Конкуренция за экспертов

Опытные эксперты предпочитают работать в командах с уже высоким уровнем зрелости DevSecOps процессов, а их привлечение требует делать оффер на 30-50% выше текущего предложения

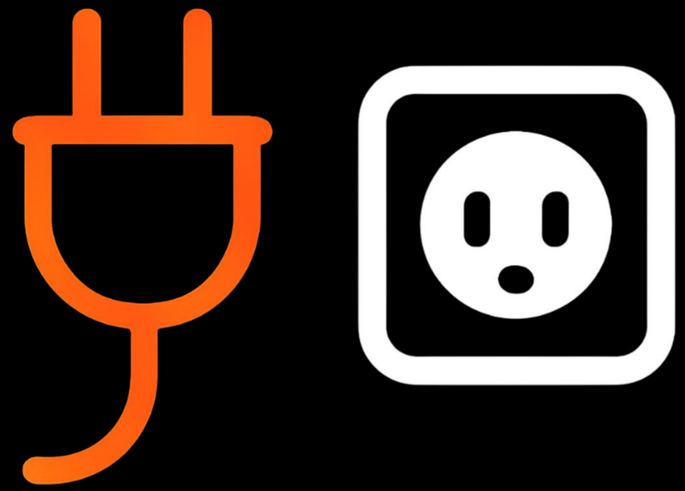
Проблема №2
Ложные срабатывания

До 90% срабатываний составляет шум

Потеря времени на поиск актуальных уязвимостей и риск задержки релиза или выпуска релиза с уязвимостями

Проблема №3
Отсутствие интеграций

- Результаты сканирования складываются вручную в Vulnerability Management систему
- Нет сквозного трекинга статуса уязвимости → дубли, просроченные уязвимости, потери времени на актуализации статуса
- Управление процессом затрудняется по мере увеличения масштаба разработки, аналогично растут финансовые затраты



Проблема №4
Комплаенс-давление

Нарушение требований = штрафы и риск приостановки деятельности

Требуется непрерывный журнал аудита и отчеты для комплаенса

02

Как выглядит идеальный процесс

Идеальный процесс

Автономность AppSec

Работает «за кадром» и не мешает разработке

Shift-left

Автоматические сканы при запуске CI/CD конвейера

Только актуальные уязвимости

Разработчики получают уязвимости в виде тикета в дефект-менеджмент системе

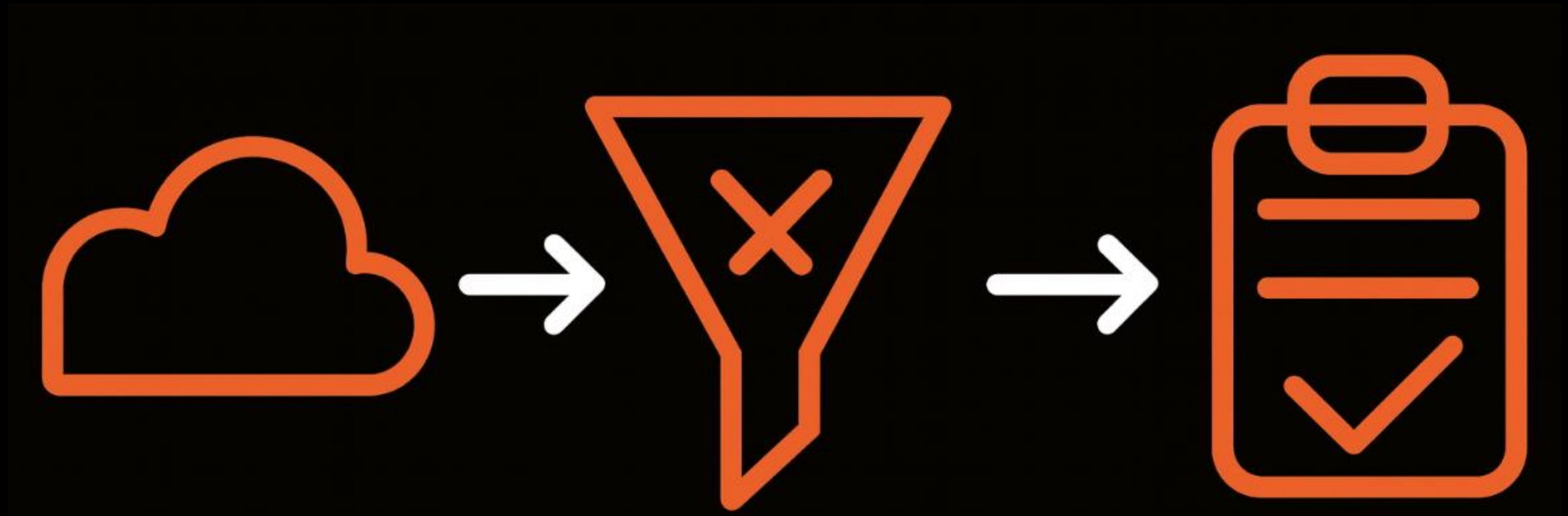
Ноль ручных действий

Разработчикам не нужно самим заниматься развитием безопасной разработки

Отчетность быстро

Для сбора данных не требуется привлекать аналитика, данные доступны по запросу

DevSecOps как услуга



Платформа берет на себя инфраструктуру

Аналитики вручную отсекают «шум»

Клиент получает только подтвержденные уязвимости

psafe

Вопросы

спикер

Пресейл инженер Apsafe

Анастасия Камалова

Пресейл инженер направления безопасной разработки УЦСБ

03

Как Apsafe снимает ключевые боли

Представляем Apsafe

DevSecOps-as-a-Service
Предоставляет процесс, инструменты, кроме устранения уязвимостей и Operate стадии DSO

Подключение

Первые актуальные уязвимости через 10 дней (или ранее)

Триаж сработок

Разработчики получают только релевантные тикеты в дефект-трекер (любой, у которого есть API)

Понятный прайс

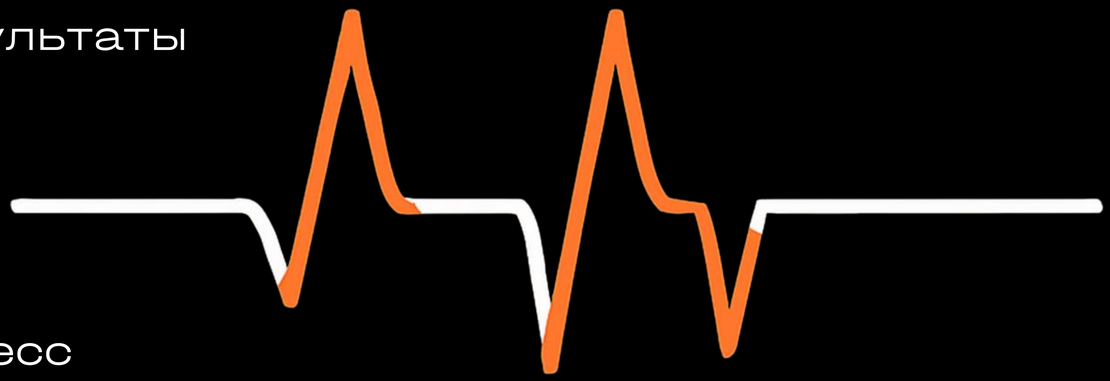
Прозрачная модель оплаты — по числу KSLOC (тыс. строк исходного кода), месячных проверок и по практике (SAST, DAST, SCA)

Сканирование

Код загружается в облако, запускаются сканы SAST, DAST, SCA

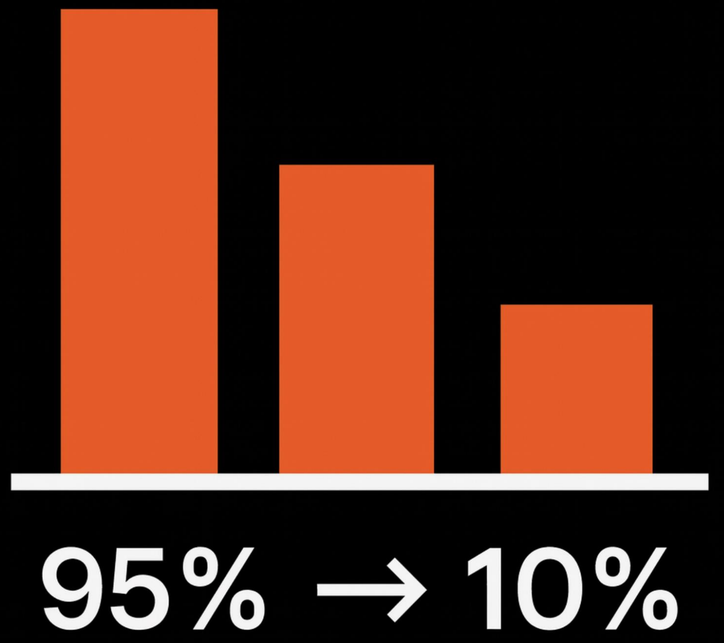
Как решаем проблему «Некому, некогда»

- Платформа + эксперты УЦСБ делают всё «под ключ»
- Сканы, анализ, приоритизация — готовые результаты
- Экспертный отчёт в удобном формате
- Можем интегрировать ваш сканер в наш процесс



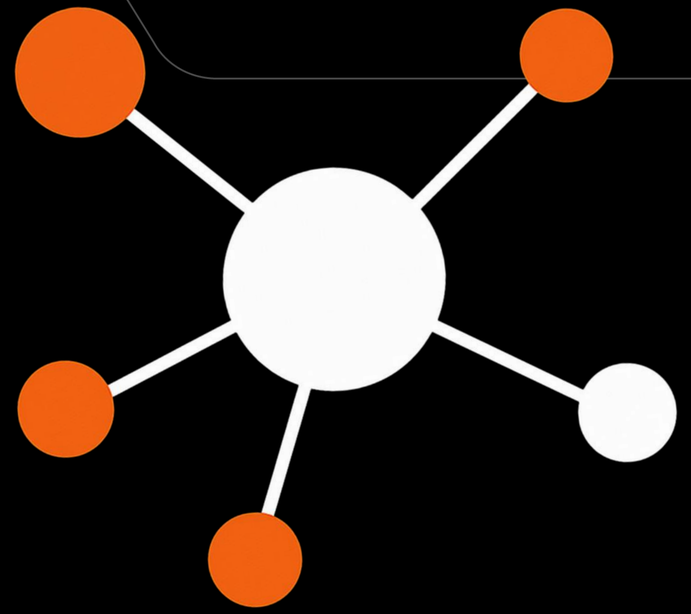
Как решаем проблему «Ложные срабатывания»

- Ложные срабатывания фильтруются в ручном режиме
- Подтвержденные уязвимости отправляются в таск-трекер только после ревью
- Разработчикам остаётся ~10 % актуальных уязвимостей → экономия времени



Как решаем проблему «Отсутствие интеграций»

- Нативные коннекторы: Jira, Redmine, YouTrack
(возможно подключение вашего трекера)
- Подключение к платформе меньше 10 дней
- Осуществляется интеграция со сканерами заказчика



Как решаем проблему «Комплаенс»

- Формирование отчётов под Приказ ФСТЭК №239, ГОСТ 56939, PCI DSS v4, Приказ ФСТЭК №117 (будет в силе с 01.03.2026)
- Передаем аудиторам описание нашей платформы для подтверждения соответствия

Базовые стандарты безопасной разработки

СТАНДАРТ		ЧТО ДЕЛАТЬ?
ГОСТ Р 56939-2024	1	Внедрить статический и динамический анализ, SCA, проверку supply chain. Всё — как часть процесса разработки.
ГОСТ Р 71207-2024	2	Использовать сертифицированный статический анализатор с нормальной точностью, API-интеграцией, фильтрацией ложных срабатываний.
ГОСТ Р 58412-2019	3	Выявлять угрозы безопасности на всех этапах SDLC. Защищать сами процессы разработки.
ГОСТ Р ISO/IEC 27034 (серия)	4	Построить secure SDLC: определить уровни доверия и применять меры ASC.
ГОСТ Р 59793-2021	5	Обязательное прохождение стадий создания АС. Безопасность должна быть включена в ТЗ и проектирование.
ГОСТ Р 57580.2-2018	6	Зрелость процессов разработки с ИБ ≥ уровень 4. Периодический аудит и повышение зрелости.

Что применять всем, как best practice

Обязательные стандарты для соответствия

СТАНДАРТ		ЧТО ДЕЛАТЬ?
ФЗ 152-ФЗ + Приказ ФСТЭК №21	1	Анализ кода на НДВ, пентесты, защищённое программирование — при высокой угрозе (1–2 тип).
ФСТЭК 239 (КИИ)	2	Статический анализ, фаззинг, динамический анализ (1-я категория), устранение уязвимостей, информирование пользователей.
ФСТЭК 117 (ГИС)	3	Соблюдение ГОСТ 56939, контроль разработки (включая подрядчиков), регулярное сканирование кода, устранение уязвимостей.

1: Операторы перс. данных

2: Субъекты КИИ

3: Гос. ИС

Отраслевые стандарты для соответствия

СТАНДАРТ		ЧТО ДЕЛАТЬ?
ГОСТ 57580.1-2017	1	Для ОУД-4: анализ кода, контроль уязвимостей, либо сертифицированное ПО.
Минэнерго 1215	2	Обязательная система патч-менеджмента и оповещения об уязвимостях со стороны разработчиков.
ГОСТ 60880 / IEC 62859 (АЭС)	3	Проверка на скрытые функции и “unsafe code”, особенно в критичных подсистемах.
PCI DSS v4.0	4	SDLC с безопасными практиками: обучение, контроль среды, устранение топ-уязвимостей, обновления, защита веб-приложений (WAF/DAST).

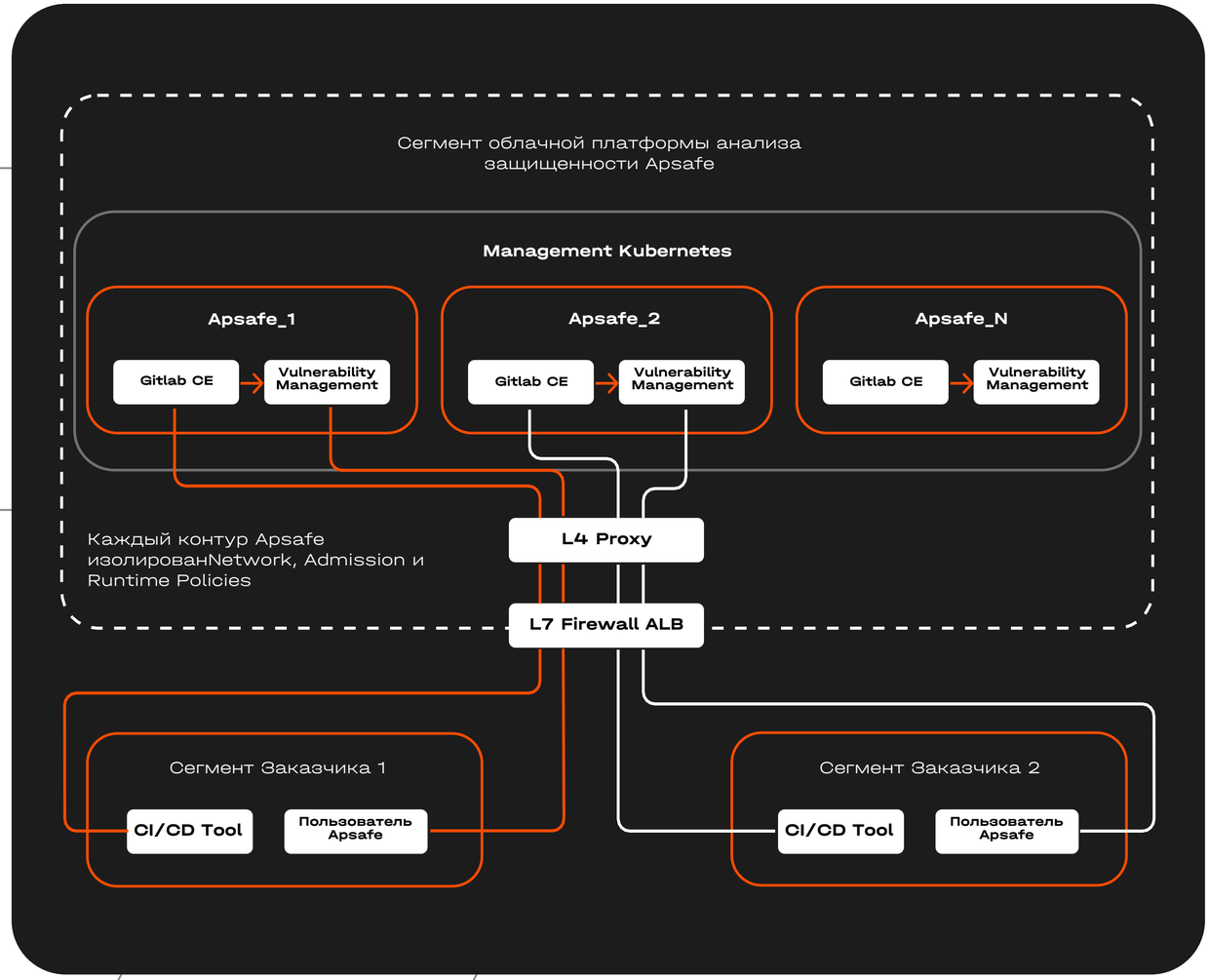
1: Финансовые организации

2: Субъекты электроэнергетики

3: Системы управления АЭС

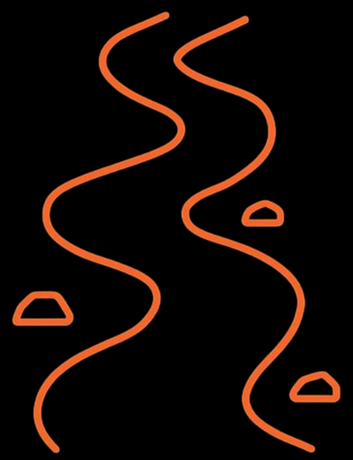
4: Работа с платежными картами

Архитектура под капотом

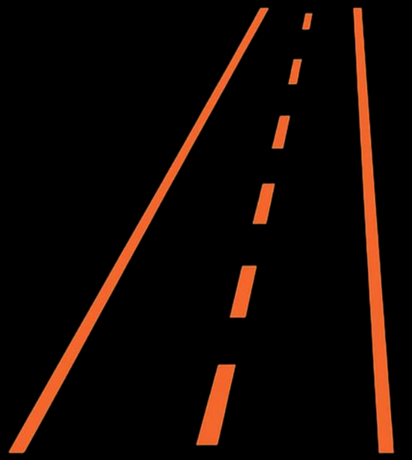


Финансовая выгода

- Сокращение времени на внедрение:
  6 месяцев → 10 дней
- Снижение затрат на экспертов:
  4 ставки → 0
- Убираем ложные срабатывания



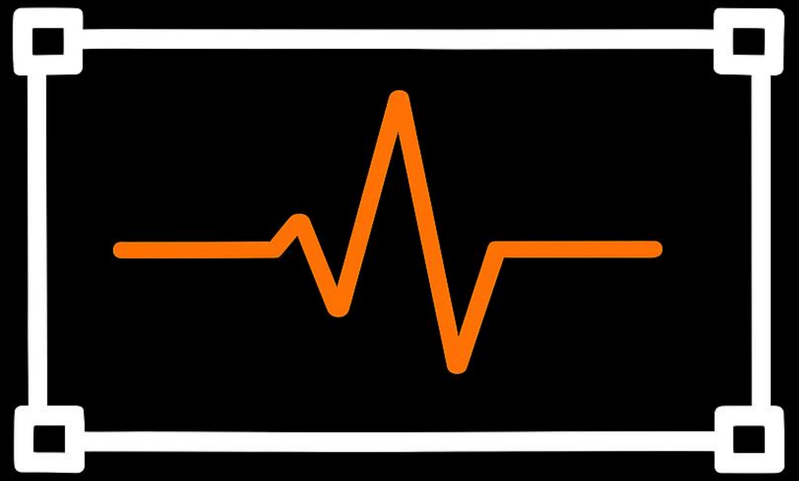
6 месяцев,
4 эксперта



10 дней
0 экспертов

Container Security
безопасность в рантайме

- Container Security
- Runtime Policy
- Admission Policy
- Network Policy
- Под капотом Luntry
- Подключаем к SOC

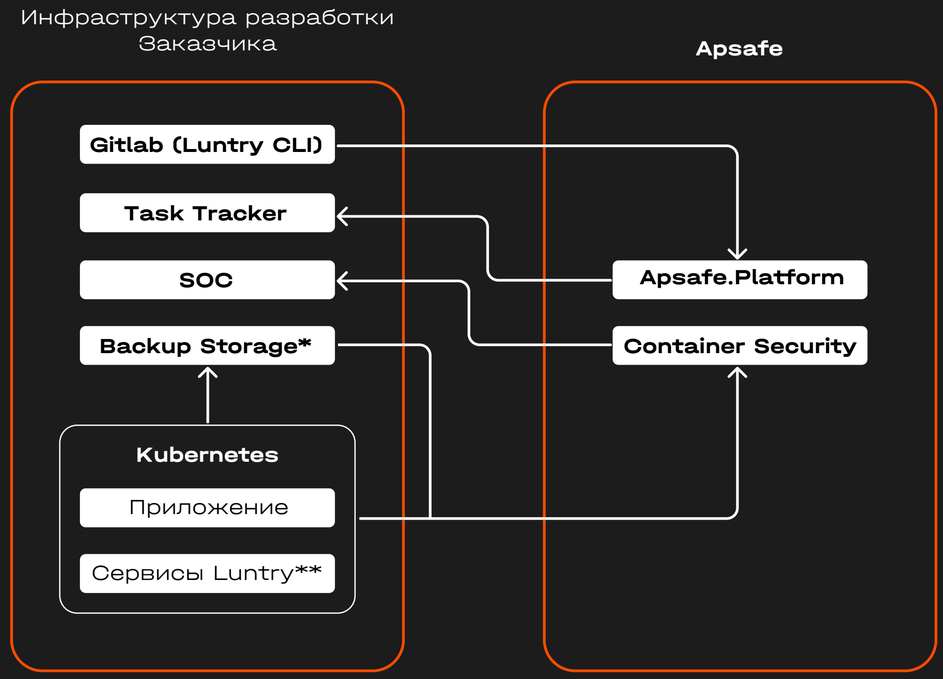


Container Security процесс

Как это работает

Проводим обучение Luntry, после чего адаптируются все политики без блокировок. Далее возможно подключение к SOC или отправка событий.

По мере роста уровня зрелости процесса, перенесем инсталляцию Luntry в контур клиента.



*Хранилище снимков файловой системы и оперативной памяти (необходимо при включении функции реагирования на аномалии)

**Sensor, Sbom-operator, Networker, Compliance, Collector, Cmd-agent, Watcher

psafe

Вопросы

спикер

AppSec-аналитик Apsafe

Виктор Тимашков

AppSec-аналитик группы инновационных решений направления безопасной разработки УЦСБ

04

Два реальных кейса клиентов

СибКом Цифра
кейс №1

- Провели все согласования и настроили доступы (6–13 марта)
- Интегрировали платформу безопасности приложения с Jira клиента — уязвимости сразу попадают в таск-трекер
- На старте выявили 4 уязвимости высокого уровня критичности
- За несколько месяцев регулярных проверок найдены десятки уязвимостей
- Консультируем разработчиков по вопросам безопасной разработки

Банк Синара
кейс №2

- Интеграция прошла с 14 по 21 октября
- 29 октября — заказчик получил первый отчёт
- На старте выявлено:
  - уязвимые компоненты в продуктах
  - уязвимые конфигурации окружения
  - уязвимости в исходном коде
- За полгода работы платформы было выявлено множество уязвимостей разной критичности. На текущий момент большая часть уязвимостей устранена командой Заказчика
- Осуществлена интеграция с таск-трекером, отправка уязвимостей автоматизирована
- Реализовали пожелания клиента в новой версии Apsafe Platform:
  - Улучшен интерфейс
  - Добавлены метрики и отчётность
  - Добавлена персонализированная аналитика для каждой команды разработки

05

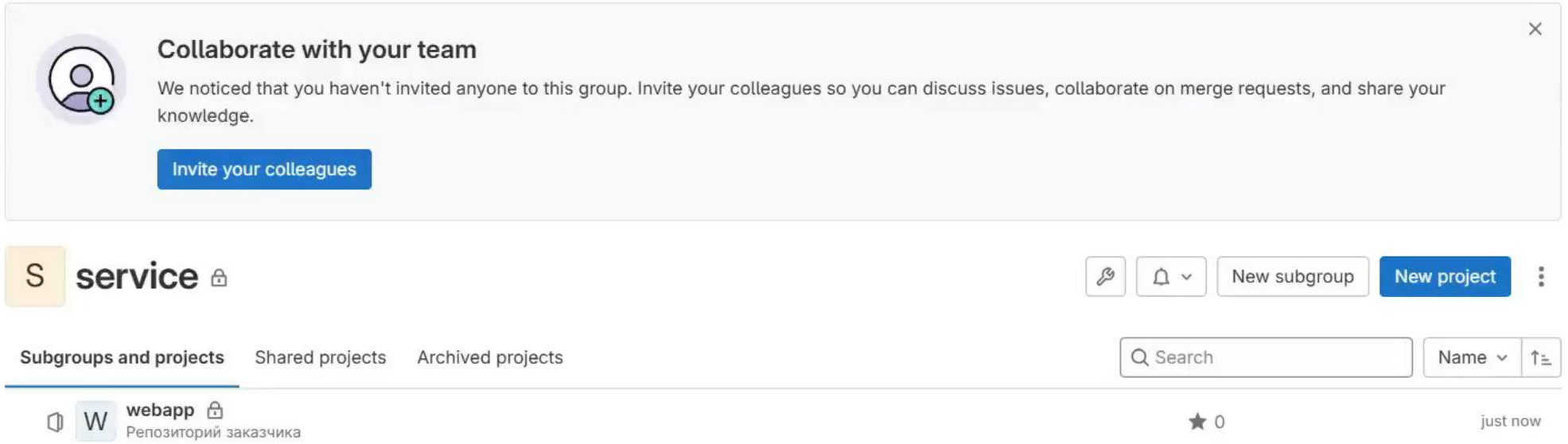
Подробнее про Apsafe

Демонстрация Apsafe

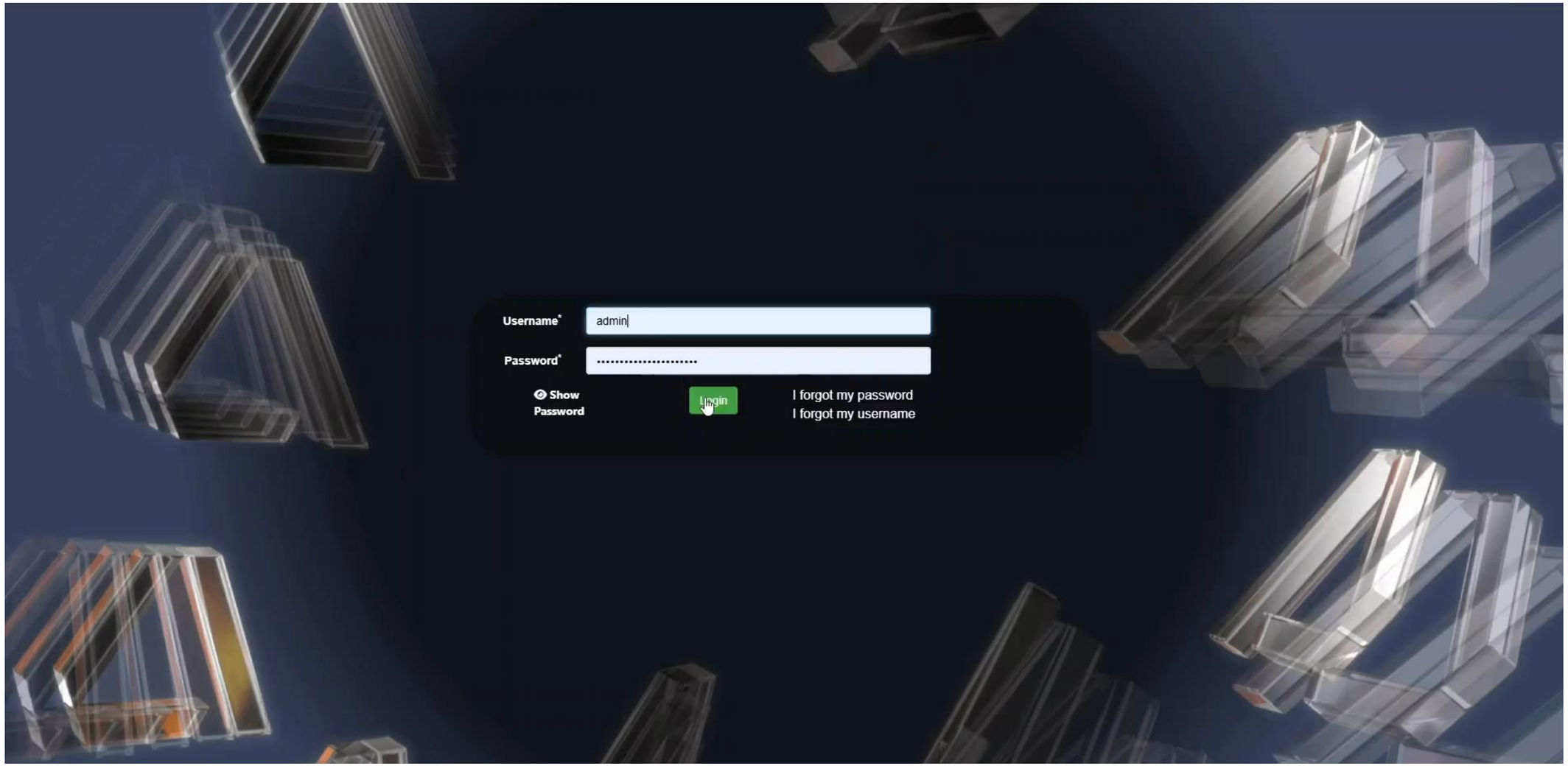
Рассмотрим:

Как выглядит отправка кода на проверку из CI/CD, как это выглядит в платформе и в каком виде уязвимость попадает разработчикам.

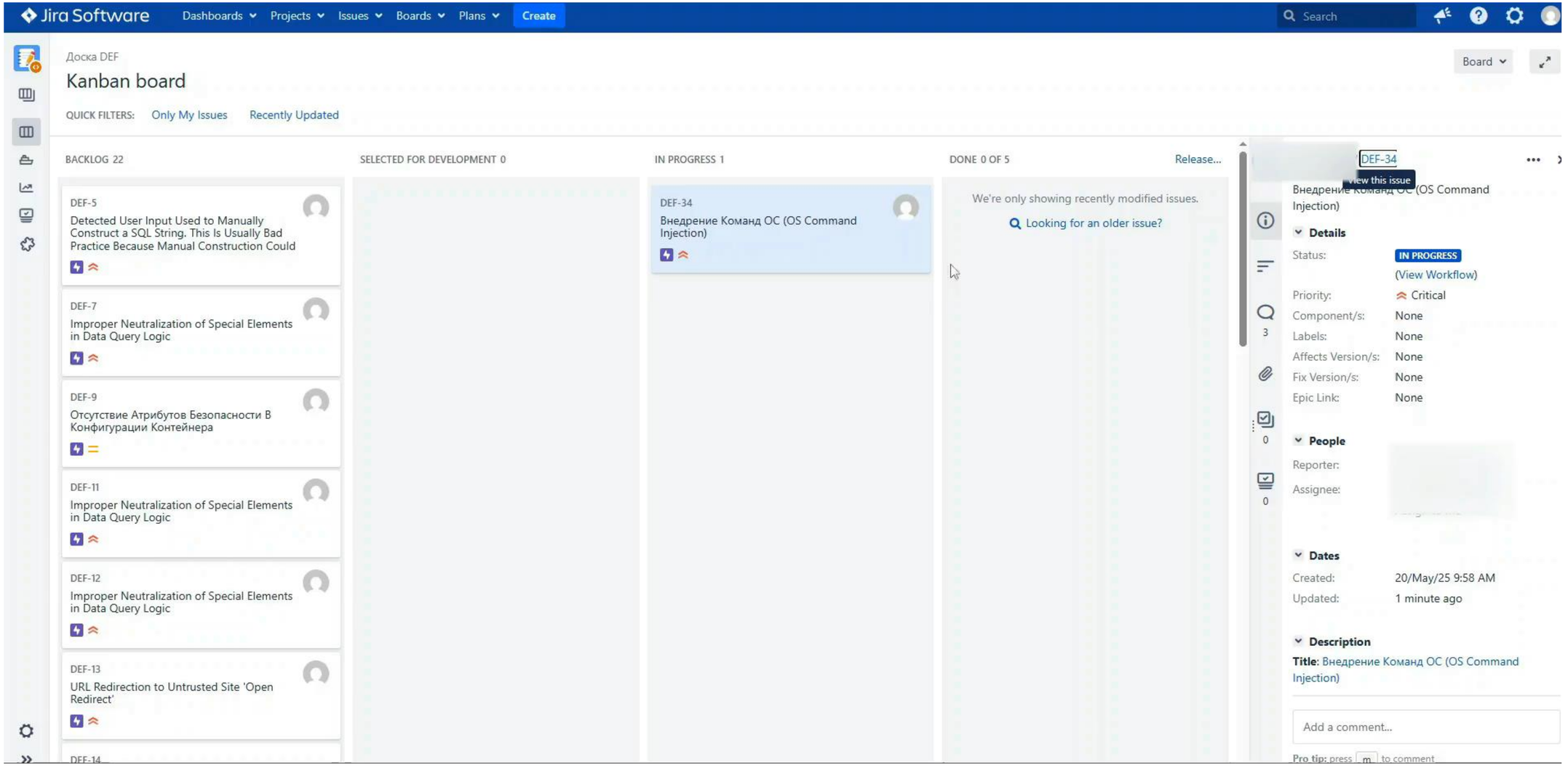
Отправка кода из CI/CD



Что происходит в платформе



Как разработчик видит уязвимость



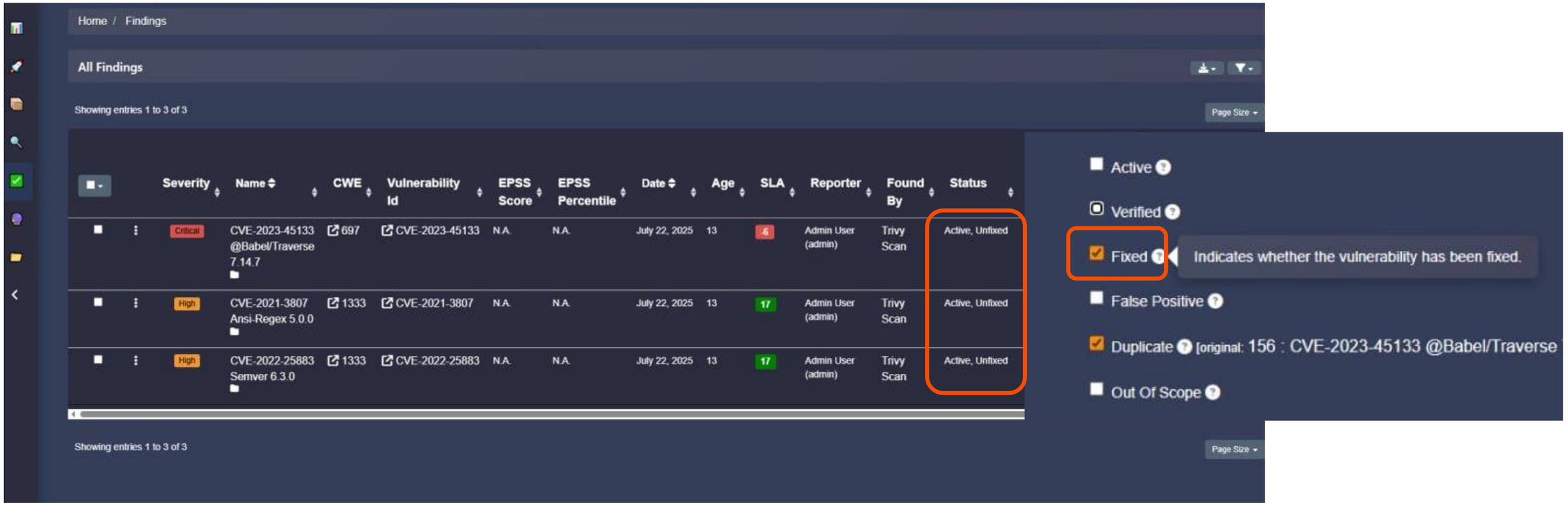
Ключевые фичи Apsafe

- Сканирование
  - Автоматический запуск сканирования
  - Нет задержки разработки

```yaml
apsafe-sync:
  stage: build
  rules:
    - if: $CI_COMMIT_BRANCH == "main" && $CI_PIPELINE_SOURCE != "pipeline"
  before_script:
    - apk add git curl rsync
    - git config --global user.email "security-check@asaas.ussc.ru"
    - git config --global user.name "USSC_Security_Check"
    # получение информации о текущем коммите
    - current_branch=$(git rev-parse --abbrev-ref HEAD)
    - commit_hash=$(git log -1 --pretty=format:"%H")
    - commit_message=$(git log -1 --pretty=format:"%s")
    - commit_author=$(git log -1 --pretty=format:"%an <%ae>")
    - commit_history=$(git log -3 --pretty=format:"%H %an <%ae>\n" | tr "\n" ' ')
  script:
    - git clone -b ${APSAFE_SYNC_GITLAB_BRANCH} https://${GITLAB_ACCESS_TOKEN_NAME}:${GITLAB_ACCESS_TOKEN}@gitlab-apsafe.com/customer/${APSAFE_SYNC_GITLAB_REPOSITORY} /tmp/apsafe_sync
    - find /tmp/apsafe_sync/ -maxdepth 1 -type f ! -name '.gitlab-ci.yml' -exec rm -rf {} +
    - find /tmp/apsafe_sync -maxdepth 1 -type d ! -name '.git' ! -name '.' ! -name '..' ! -name 'apsafe_sync' -exec rm -rf {} +
    # синхронизаация файлов
    - rsync -av ./ /tmp/apsafe_sync --exclude apsafe_sync --exclude .git --exclude .gitlab-ci.yml > /dev/null
    - cd /tmp/apsafe_sync
    - git add . *
    - >
      git commit -m "Commit Hash: $commit_hash\nCommit Message: $commit_message\nCommit Author: $commit_author\nCurrent Branch: $current_branch\nCommit History:\n$commit_history"
    # зеркалирование репозитория
    - git remote set-url origin https://${GITLAB_ACCESS_TOKEN_NAME}:${GITLAB_ACCESS_TOKEN}@gitlab-apsafe.com/customer/${APSAFE_SYNC_GITLAB_REPOSITORY}
    - git push -u origin HEAD -ff  || true
  after_script:
      - rm -rf /tmp/apsafe_sync
```

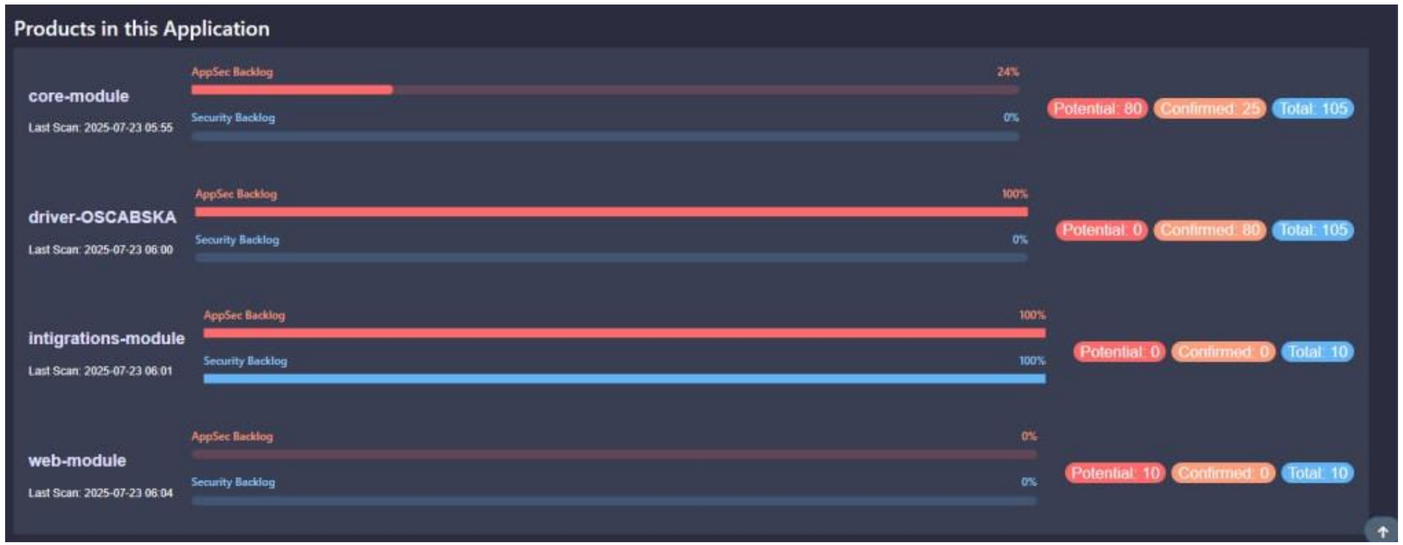
Ключевые фичи Apsafe

- Управление уязвимостями
  - Полный цикл «Detected → Fixed»



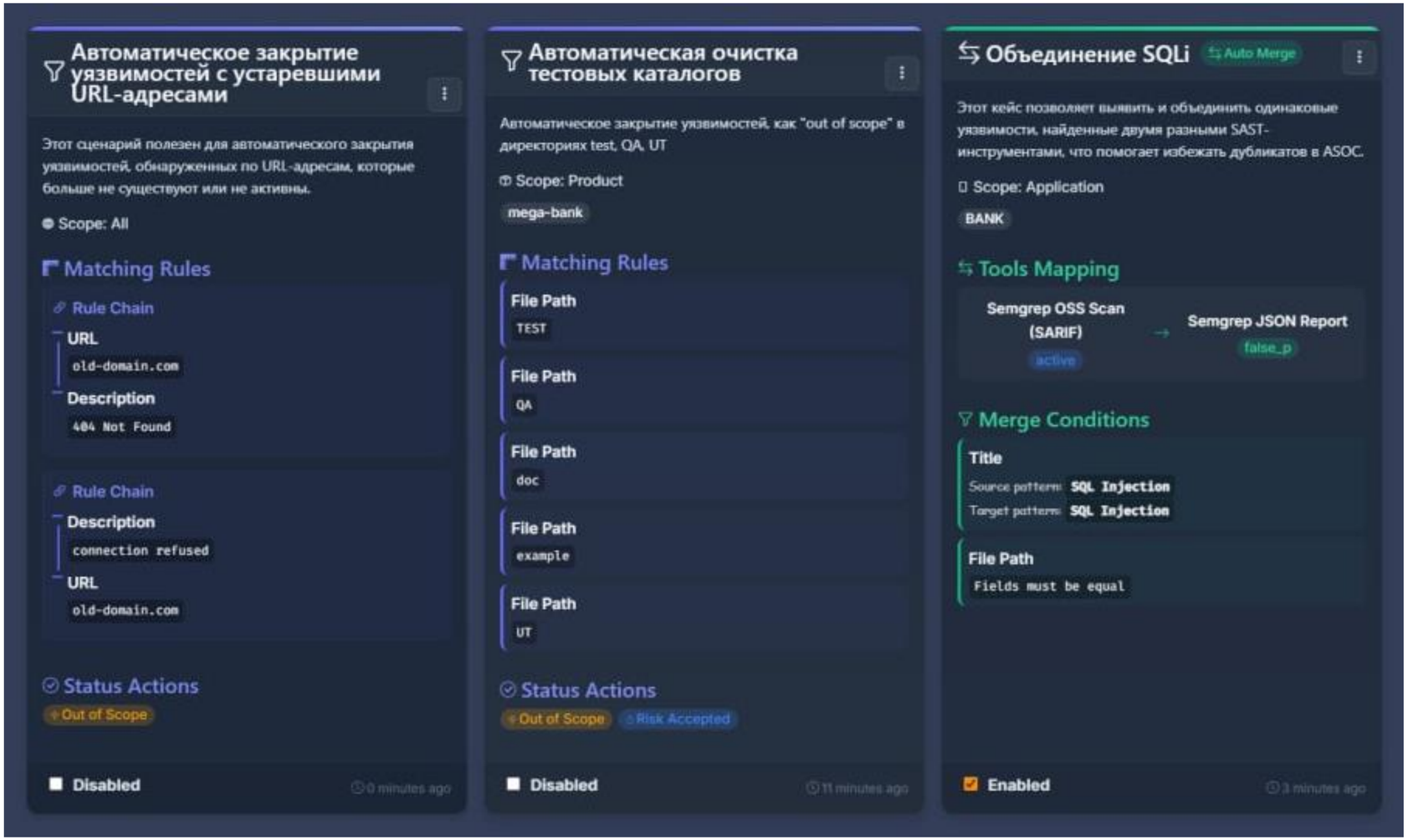
Ключевые фичи Apsafe

- Управление уязвимостями
  - Объединение микросервисов в сущность «Приложение/команда разработки», общая статистика



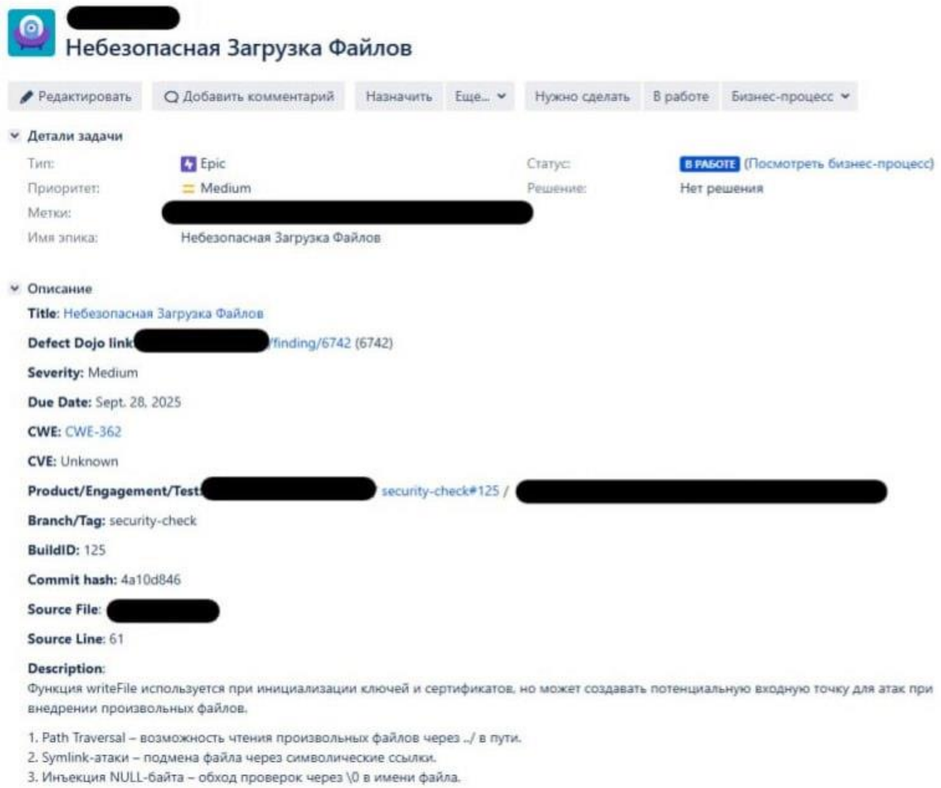
Ключевые фичи Apsafe

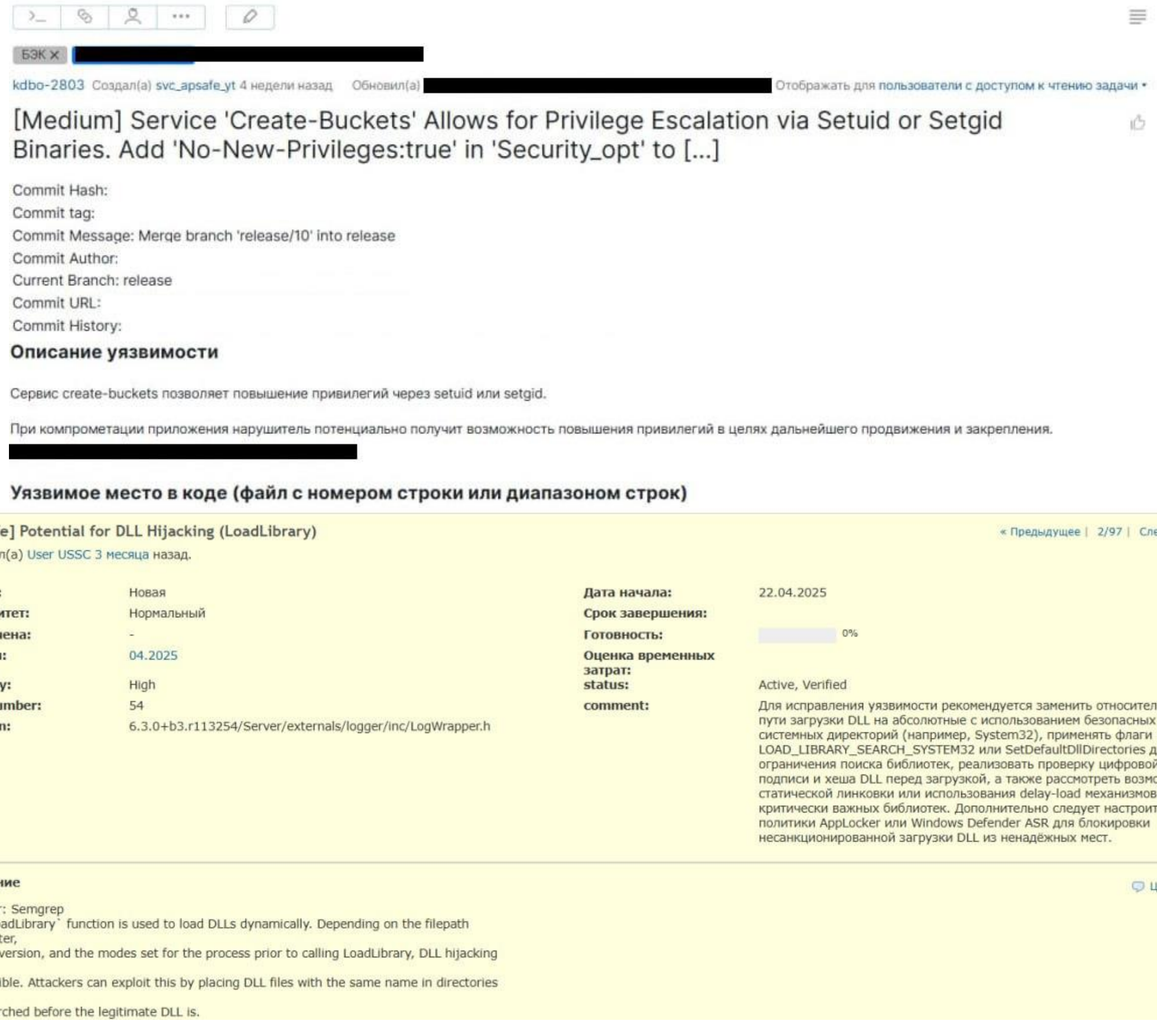
- Управление уязвимостями
  - Настраиваемые правила автотриажа



Ключевые фичи Apsafe

- Интеграции
  - Выгрузка дефектов в таск-трекеры: уже есть поддержка Jira, Redmine YouTrack





Roadmap Apsafe

- Проверка коммитов в релизах
- Автоматизированные отчеты под Приказ ФСТЭК №239, ГОСТ 56939, PCI DSS v4, Приказ ФСТЭК №117 (вступает в силу с 01.03.2026)
- Двусторонняя синхронизация статусов уязвимостей между Apsafe.Platform и дефект-трекером
- Улучшение качества сканирования за счет кастомных правил

psafe

Вопросы

СПАСИБО!

Получите обзор топ-угроз под вашу архитектуру

Заполните анкету по QR-коду и мы расскажем,
от чего вам следует защищаться

cybersec@ussc.ru