
Яндекс

Яндекс

Наш опыт автоматизации сканирования веб-приложений

Эльдар Заитов

whoami

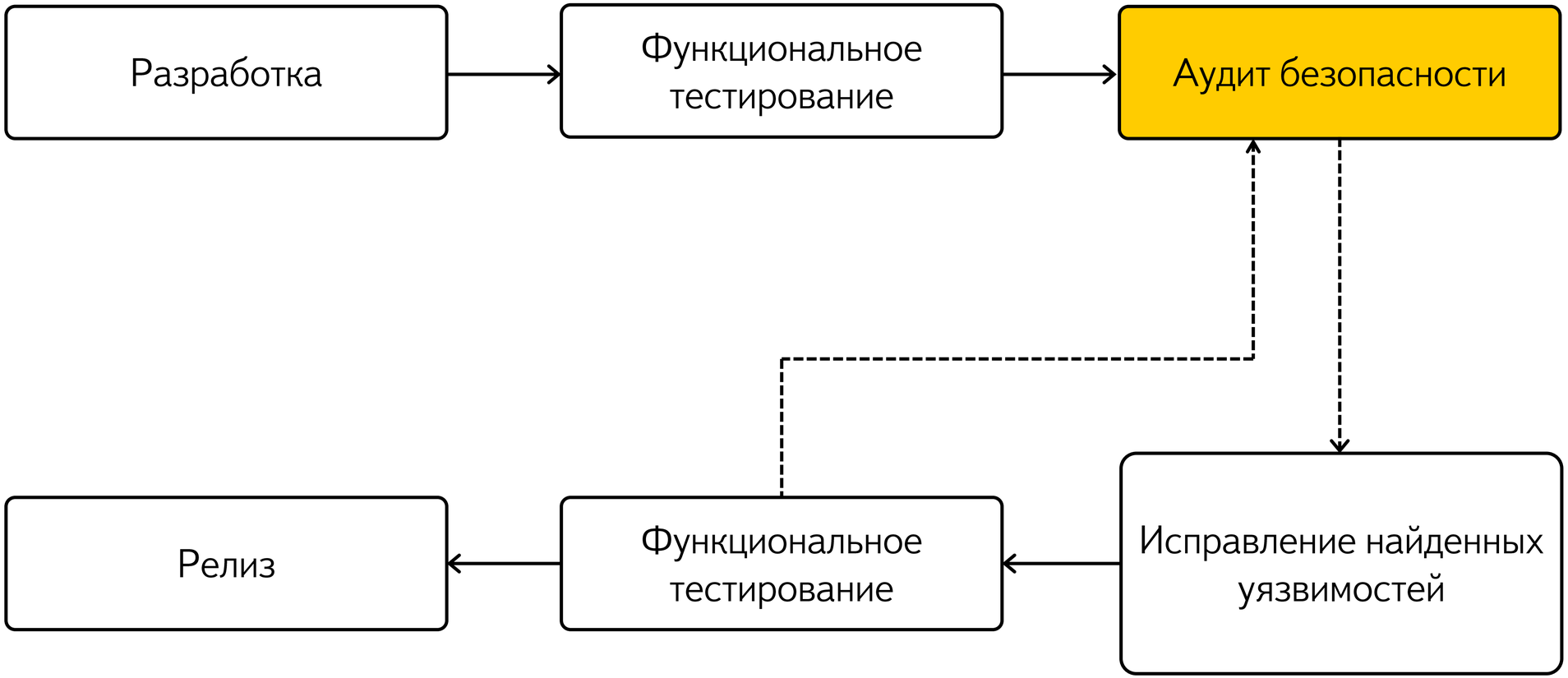
- Application Security Engineer at Yandex
- More Smoked Leet Chicken CTF team
- CTFtime.org

Product Security Team

- Internal policies, requirements, recommendations and best practices
- Secure common components
- Consultations and trainings
- Manual and automated security testing

Типичный подход к сканированию

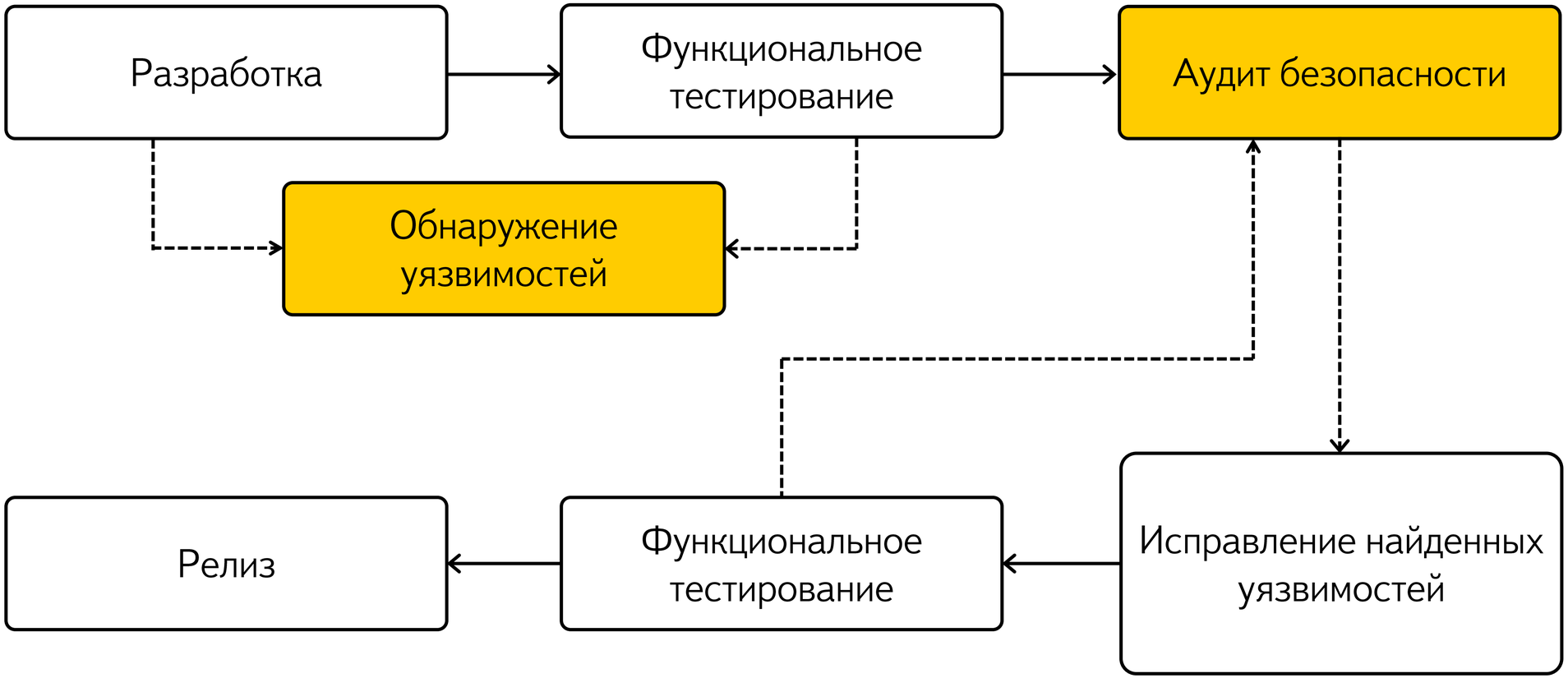
Релизный цикл v1.0



Что не так?

- Эффективно только при (пере-)запуске сервиса
- Отнимает время — ИБ становится «бутылочным горлышком»
- Простые уязвимости находятся перед самым релизом
- Стимулирует команды сервисов постараться избежать аудита или перенести сроки исправления уязвимостей

Релизный цикл v1.1



Чем лучше

- Простые уязвимости находятся и исправляются на этапе разработки
- Есть возможность акцентировать внимание на критичных уязвимостях при предрелизном аудите
- Команда тестирования сервиса и разработки лучше знает сервис

Научим тестировщиков запускать сканер?

SaaS* сканер v1.0

1. Создаём тикет в JIRA на XXX и просим настроить сканер;

2. Согласовываем с ответственными за тестируемый сервер администраторами время сканирования, просим его заказать доступ с YYY на тестируемый сервер но нужному протоколу (обычно tcp/80,tcp/443);

3. После того, как Служба ИБ "дала добро", заходим на ZZZ и добавляем задачу на сканирование соответствующей цели. Можно сканировать как одноразово, так и по расписанию;

**Software-as-a-Service*

В борьбе за автоматизацию

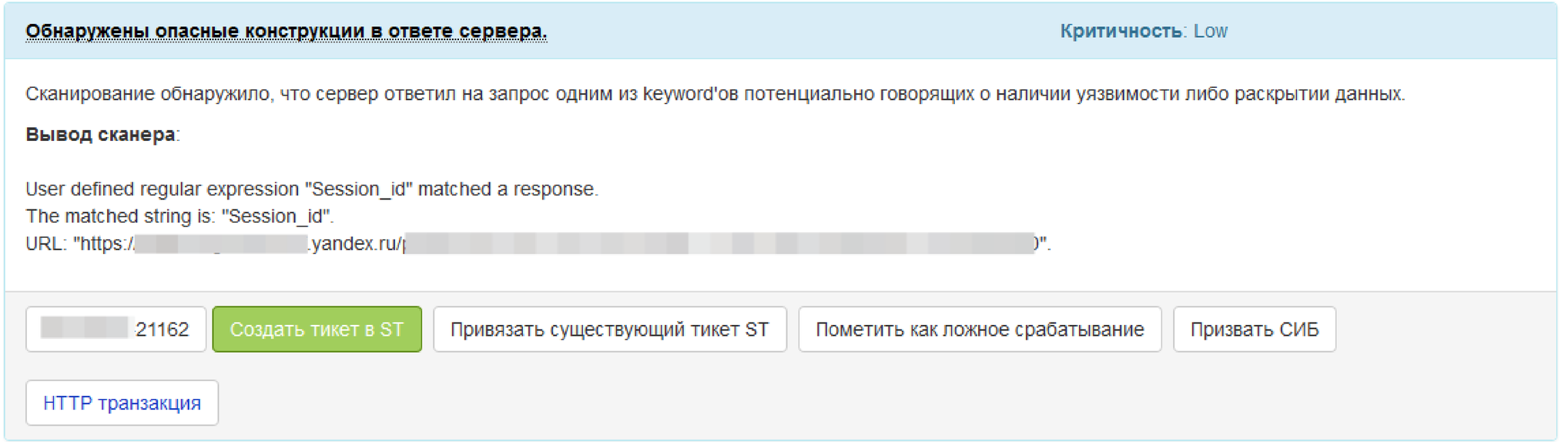
SaaS сканер v1.1

- REST API для управления сканированием
- Провязка с общим фреймворком для запуска автотестов (Aqua)
- Сканирующие агенты «рядом» с кластером Selenium нод

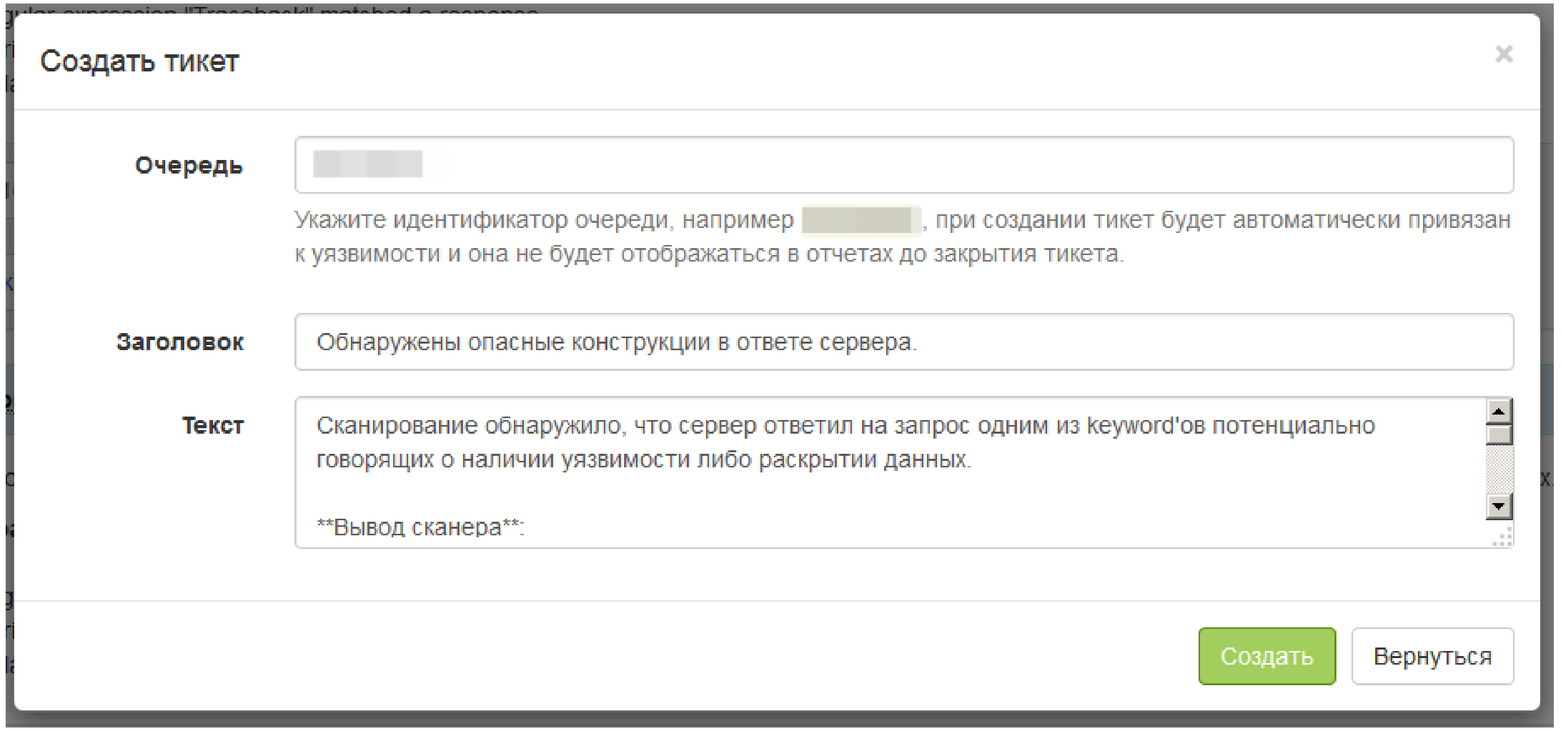
SaaS сканер v1.1

- Человекочитаемые отчеты
- Возможность управления шумом
- «Склейка» уязвимостей по сервису
- Провязка с таск-трекером
- Ограниченное время сканирования

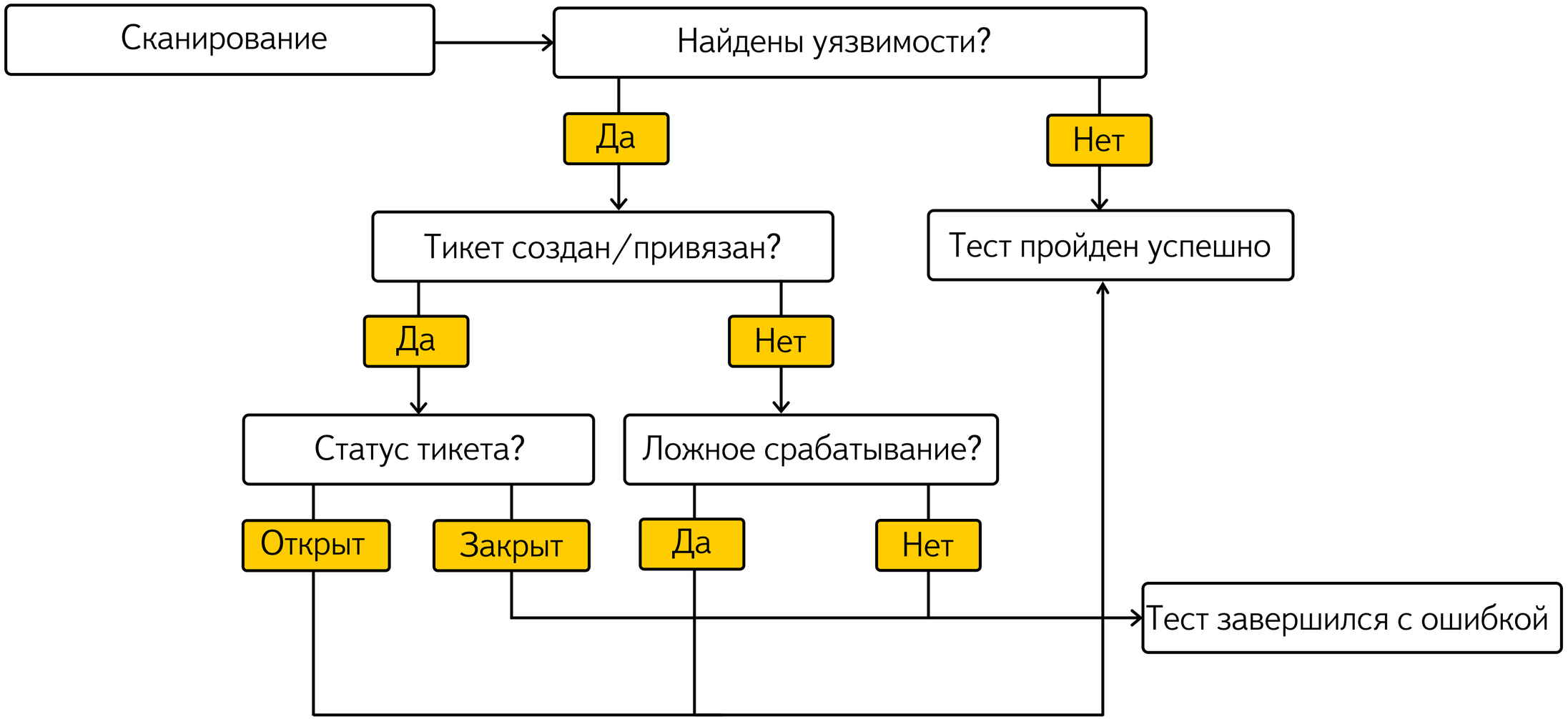
Отчет


Обнаружены опасные конструкции в ответе сервера.
Критичность: Low
Сканирование обнаружило, что сервер ответил на запрос одним из keyword'ов потенциально говорящих о наличии уязвимости либо раскрытии данных.
Вывод сканера:
User defined regular expression "Session_id" matched a response.
The matched string is: "Session_id".
URL: "https:/ .yandex.ru/
21162
Создать тикет в ST
Привязать существующий тикет ST
Пометить как ложное срабатывание
Призвать СИБ
HTTP транзакция

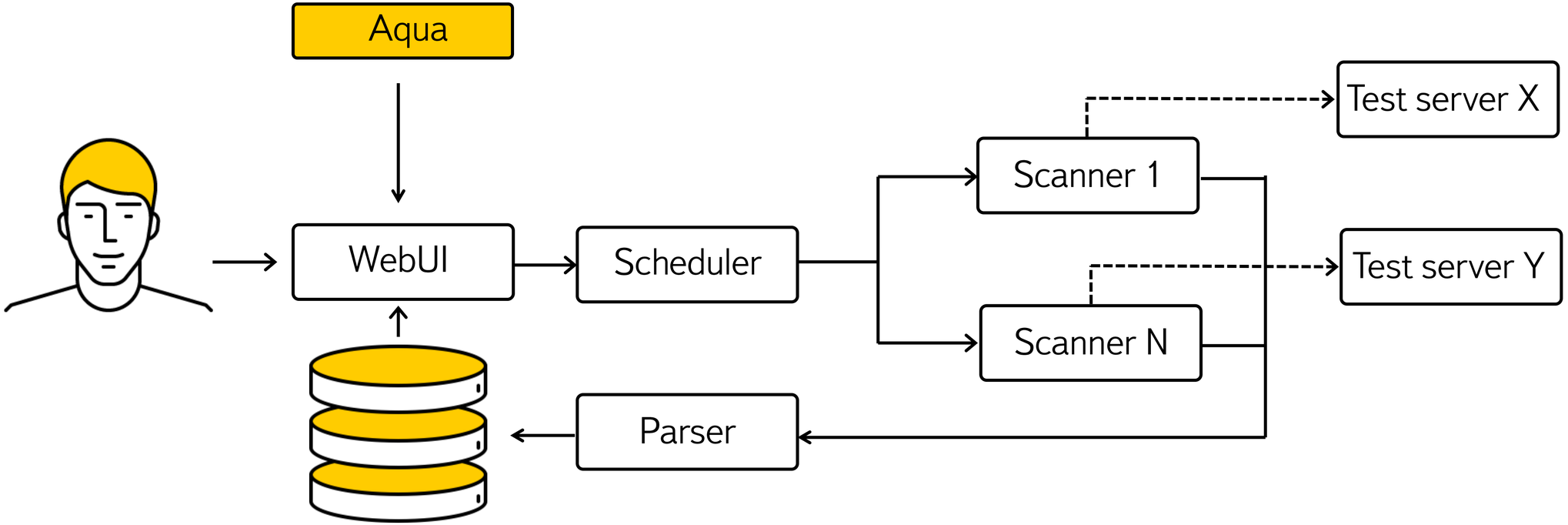
Отчет


Создать тикет
Очередь
Укажите идентификатор очереди, например , при создании тикет будет автоматически привязан к уязвимости и она не будет отображаться в отчетах до закрытия тикета.
Заголовок
Обнаружены опасные конструкции в ответе сервера.
Текст
Сканирование обнаружило, что сервер ответил на запрос одним из keyword'ов потенциально говорящих о наличии уязвимости либо раскрытии данных.
Вывод сканера:
Создать
Вернуться

Алгоритм разбора результата сканирования



Что внутри?


Aqua
WebUI
Scheduler
Scanner 1
Scanner N
Test server X
Test server Y
Parser

Плюсы

- Масштабируемость (агенты в OpenStack)
- Одна точка входа, сколько угодно сканеров (мы используем w3af)
- Возможность быстро дописать и прогнать любые проверки

В борьбе за максимальное покрытие

Challenges

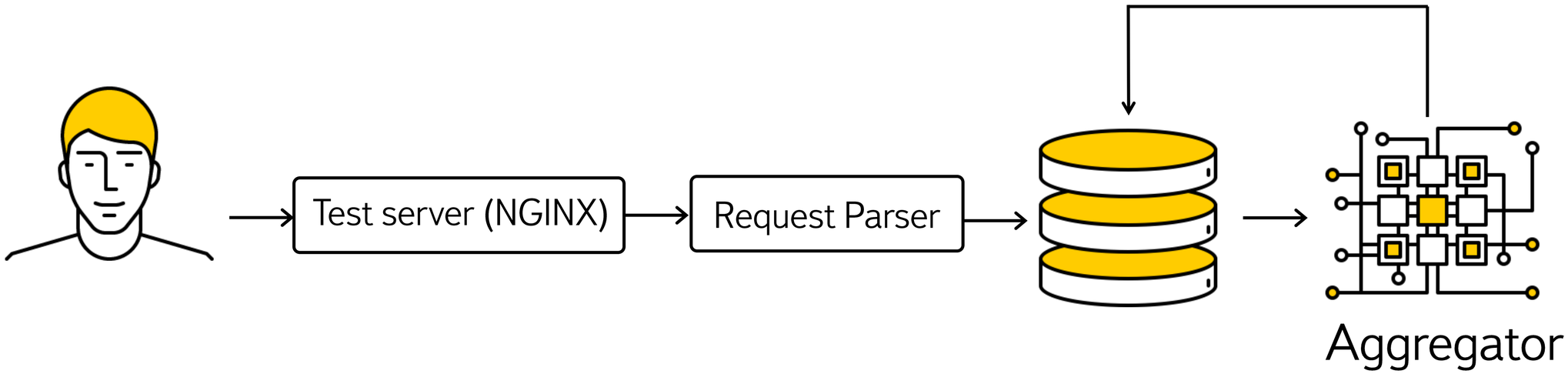
- Проблема обхода Web2.0 ⟶ Отказоустойчивость

У

Унификация

Source: https://ru.wikipedia.org/wiki/Орлан_(скафандр)

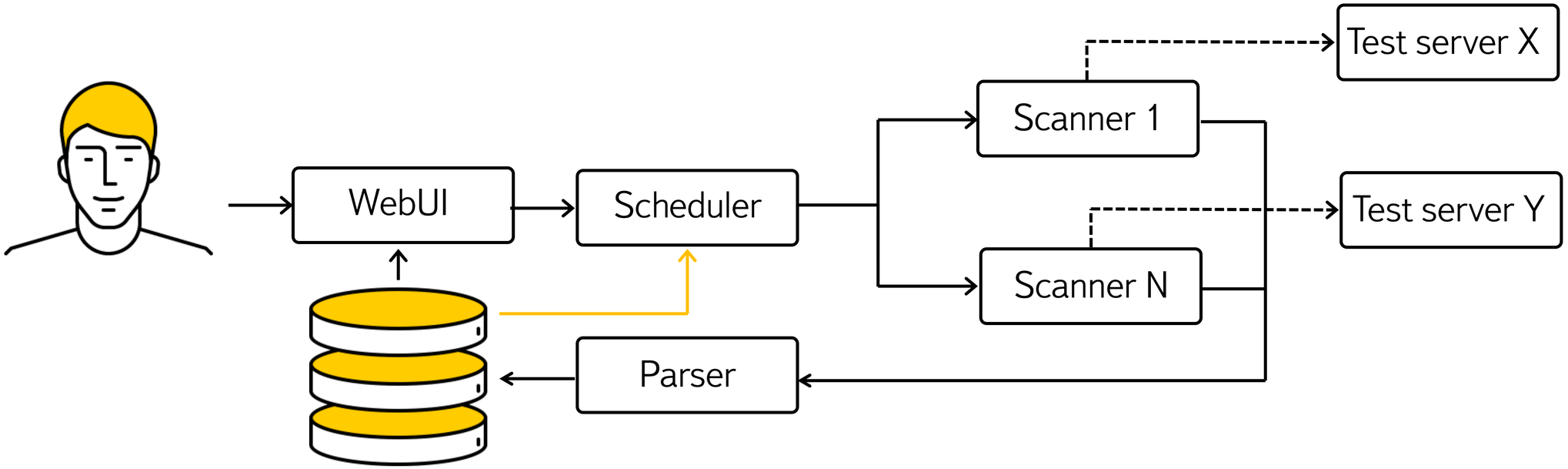
Как увеличить покрытие?



Агрегируем запросы

- Сервис
- Метод запроса
- URI, параметры итд.
- Content-Type
- Заголовки, тело, что угодно

Подмешиваем в сканирование



Бонусы

- Автоматическое детектирование новой функциональности
- Семплы запросов для ручного анализа

Challenges

- CSRF токены
- Complex data formats
- Улучшение алгоритмов агрегации
- Широкое внедрение

Спасибо за внимание!
Вопросы?

Контакты

Эльдар Заитов

ezaitov@yandex-team.ru

kyprizel