SORCA

Анализ происхождения ПО и состава рисков

Руководство администратора и пользователя

Версия 0.2.4-beta

(Листов – 29)

ООО «СОРКА»
Москва
2026

Содержание

1 Основные характеристики

1.1 Назначение программы

Программное обеспечение «СОРКА» разработано с целью автоматизации процесса композиционного анализа программного обеспечения, выявления заимствованных компонентов, уязвимостей, секретов и формирования отчётных материалов по результатам анализа.

Программное обеспечение «СОРКА» предназначено для:

- автоматизированного анализа исходных текстов программного обеспечения, архивов с исходными текстами, репозиториев и SBOM-файлов;
- автоматизированного формирования перечня компонентов программного обеспечения с указанием наименований, версий, экосистем, типов зависимостей и областей применения;
- выявления компонентов с открытым исходным кодом и сопоставления компонентов со ссылками на репозитории или архивы исходных текстов;
- расчёта контрольных сумм архивов исходных текстов компонентов;
- анализа зависимостей компонентов, включая прямые и транзитивные зависимости, с учётом анализа и данных пакетных менеджеров;
- выявления уязвимостей компонентов программного обеспечения по БДУ, NVD, GHSA и др.;
- сопоставления уязвимостей с компонентами и их версиями;
- отображения уязвимостей по уровням критичности с возможностью фильтрации, сортировки, разметки статусов и добавления экспертных комментариев;
- поиска секретов, ключей, паролей, токенов и иных чувствительных данных в анализируемых исходных текстах;
- разметки найденных секретов по статусам с сохранением экспертных комментариев;
- определения языков программирования компонентов по репозиториям и архивам исходных текстов;
- формирования и хранения структуры проектов и подпроектов с наследованием настроек анализа;
- запуска анализа отдельных проектов, подпроектов и групп подпроектов в том числе повторного анализа с переносом ранее выполненной экспертной разметки, где это применимо;
- ведения общего справочника компонентов и связанных с ними репозиториев или архивов исходных текстов;
- исключения компонентов из состава учитываемого SBOM по ручной разметке или на основании шаблонов фильтрации;
- формирования списков доверенных компонентов с указанием источника поставки;
- разметки компонентов по признакам поверхности атаки и функций безопасности;
- формирования SBOM в формате CycloneDX, включая SBOM по требованиям ФСТЭК;
- импорта SBOM и результатов анализа с сохранением состава компонентов, уязвимостей, секретов, ссылок на исходные тексты и экспертной разметки;
- формирования HTML, PDF, CSV, XLSX и JSON отчётов по уязвимостям, секретам, компонентам и SBOM;

- настройки состава экспортируемых отчётов с учётом фильтров по уязвимостям, компонентам и секретам;
- формирования сводных представлений по проектам, подпроектам, запускам, компонентам, уязвимостям и секретам.

1.2 Условия выполнения программы

ПО «СОРКА» предназначено для использования на технических средствах под управлением операционных систем семейства Linux и Windows, обеспечивающих запуск контейнеров Docker. Рекомендуемой средой эксплуатации является Linux x64, в том числе Astra Linux Special Edition.

Серверная часть поставляется и эксплуатируется в контейнере Docker. Для корректной работы требуется поддержка архитектуры x64.

Веб-интерфейс доступен через браузер и может использоваться с любого устройства, имеющего доступ к серверу. Для работы с веб-интерфейсом на клиентском устройстве достаточно современного браузера.

Применение ПО рассчитано на персонал, имеющий опыт работы с анализом состава программного обеспечения, управлением проектами анализа, интерпретацией сведений об уязвимостях, SBOM и результатах проверки исходных текстов.

1.3 Среда функционирования

Минимальные характеристики технических средств, для функционирования программного обеспечения представлены в таблице 1.

Таблица 1 – Минимальные характеристики технических средств для развёртывания сервера

Параметр	Значение
Процессор	архитектура x64
Оперативная память	16 ГБ
Хранилище (SSD)	100 ГБ
Система контейнеризации	Docker

Рекомендуемые характеристики технических средств, для функционирования программного обеспечения представлены в таблице 2.

Таблица 2 – Рекомендуемые характеристики технических средств для развёртывания сервера

Параметр	Значение
Процессор	архитектура x64, 3.7 ГГц и выше
Оперативная память	64 ГБ
Хранилище (SSD NVMe)	600 ГБ
Система контейнеризации	Docker

Для работы пользователей с веб-интерфейсом «СОРКА» достаточно любого технического средства, имеющего сетевой доступ к серверу и установленный современный веб-браузер. Специальные требования к процессору, объёму оперативной памяти и хранилищу клиентского устройства не предъявляются.

2 Развёртывание

Развёртывание выполняется на сервере с установленной и настроенной контейнерной средой Docker. Перед установкой необходимо убедиться, что техническое средство соответствует минимальным требованиям, имеет доступ к образу в реестре контейнеров и располагает достаточным объёмом свободного дискового пространства для хранения базы данных, рабочих каталогов анализа, кэша и отчётных материалов.

Поставка включает контейнерный образ программного обеспечения, файл конфигурации окружения, файл описания сервисов Docker Compose, скрипт первичной инициализации и файл лицензии. В конфигурации задаются параметры запуска контейнера, порт веб-интерфейса, имена томов Docker, параметры подключения к встроенной базе данных, путь к лицензии и иные эксплуатационные настройки, в том числе логин и пароль первого пользователя в системе.

Для развёртывания необходимо выполнить перечисленные ниже действия.

1. Установить Docker и Docker Compose на сервере развёртывания.
2. Получить контейнерный образ SORCA из реестра контейнеров.
3. Подготовить файл конфигурации окружения .env.
4. Запустить сервис SORCA с использованием Docker Compose.
5. Выполнить скрипт «init» для первичной инициализации ПО.
6. Проверить состояние контейнера и доступность веб-интерфейса.
7. Перейти в веб-интерфейс SORCA через браузер по адресу сервера и указанному порту.
8. Выполнить первичную настройку: загрузить лицензию, проверить параметры анализа, состояние баз уязвимостей и учётные записи пользователей.

Данные должны храниться в постоянных Docker-томах. К таким данным относятся база данных, лицензия, рабочие файлы анализов, кэш анализатора, локальные базы уязвимостей и служебные материалы. При обновлении версии контейнер может быть заменён, однако постоянные тома должны сохраняться, так как в них содержатся пользовательские проекты, результаты анализов и настройки системы.

После запуска – веб-интерфейс доступен пользователям с клиентских устройств через современный браузер при наличии сетевого доступа к серверу.

Пример команд для развёртывания:

```
docker pull qwer.sorca.ru/sorca:0.2.4-beta
docker compose --env-file .env up -d
chmod +x ./init.sh
./init.sh (.\init.ps1 – для windows)
```

Пример команд для обновления установленного ПО:

```
docker pull qwer.sorca.ru/sorca:0.2.4-beta
docker stop sorca
docker rm sorca
# поменять версию ПО в .env
docker compose --env-file .env up -d
```

После успешного обновления и проверки, можно удалить docker образ с устаревшей версией ПО.

ВНИМАНИЕ! Не удаляйте .env файл, полученный для развёртывания ПО.

3 Веб-интерфейс

3.1 Вкладка «Главная»

Данная вкладка содержит общую информацию по приложению и проведенным анализом доступных проектов (рисунок 1):

- общее количество активных проектов (подпроектов);
- общее количество запусков проектов (подпроектов);
- общее количество найденных уязвимостей в активных проектах (подпроектах);
- статус актуальности базы уязвимостей;
- срок действия лицензии;
- топ-5 уязвимостей активных проектов (подпроектов);
- график с распределением по языкам активных проектов (подпроектов);
- график с распределением по лицензиям активных проектов (подпроектов).

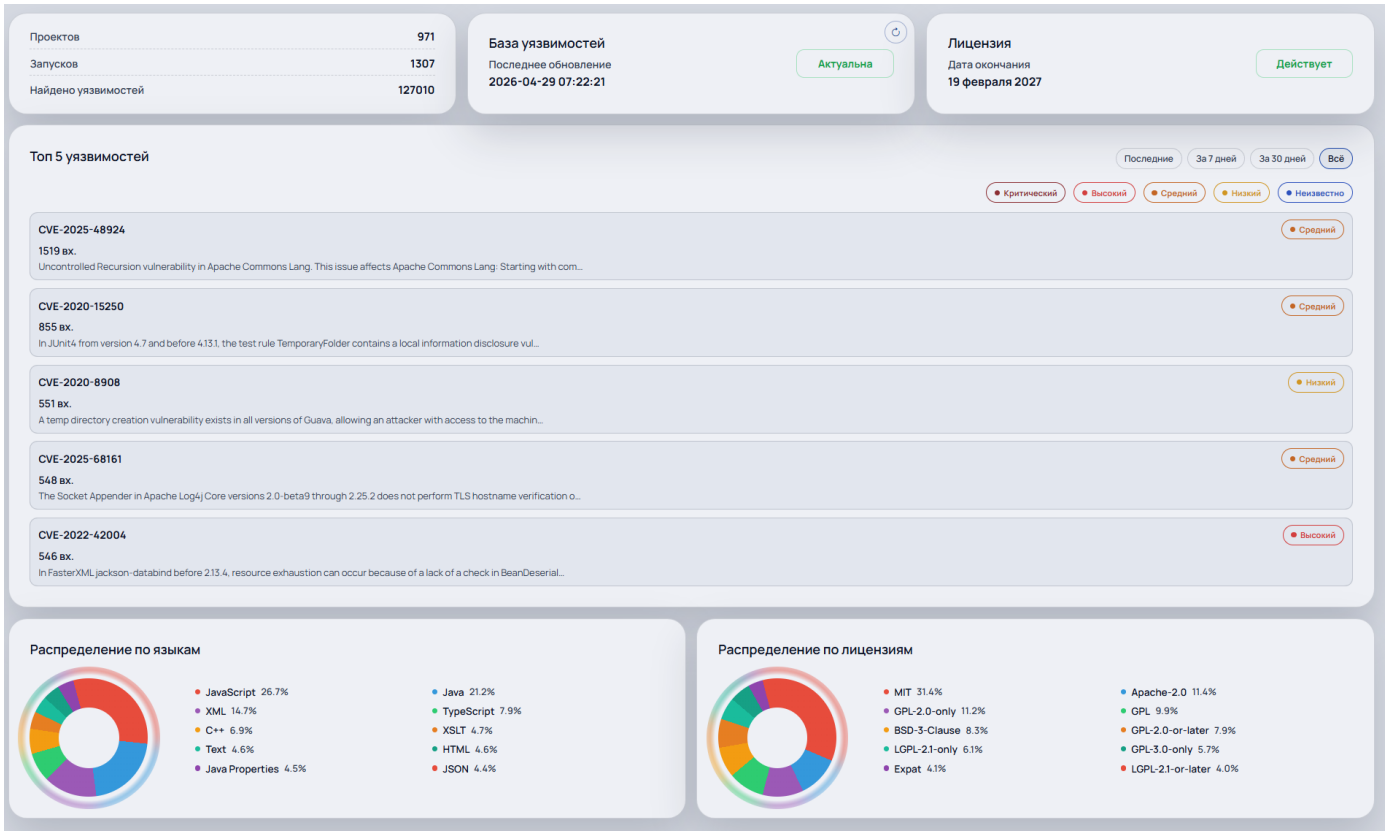

Рисунок 1 – Вкладка «Главная»

3.2 Вкладка «Проекты»

3.2.1 Создание и настройка проекта

По нажатии кнопки «Создать проект» открывается модальное окно, позволяющее произвести предварительную настройку проекта.

Окно содержит следующие настройки:

- Вкладка «Проект» (рисунок 2):
  - поле «Название проекта»;
  - выпадающий список «Родительский проект» – позволяет присвоить создаваемому проекту уже существующий «родительский» проект;

- выпадающий список «Подпроекты» – позволяет присвоить создаваемому проекту уже существующий подпроект;
- поле «Новые подпроекты» – позволяет присвоить создаваемому проекту новые подпроекты, создавая «вложенность».

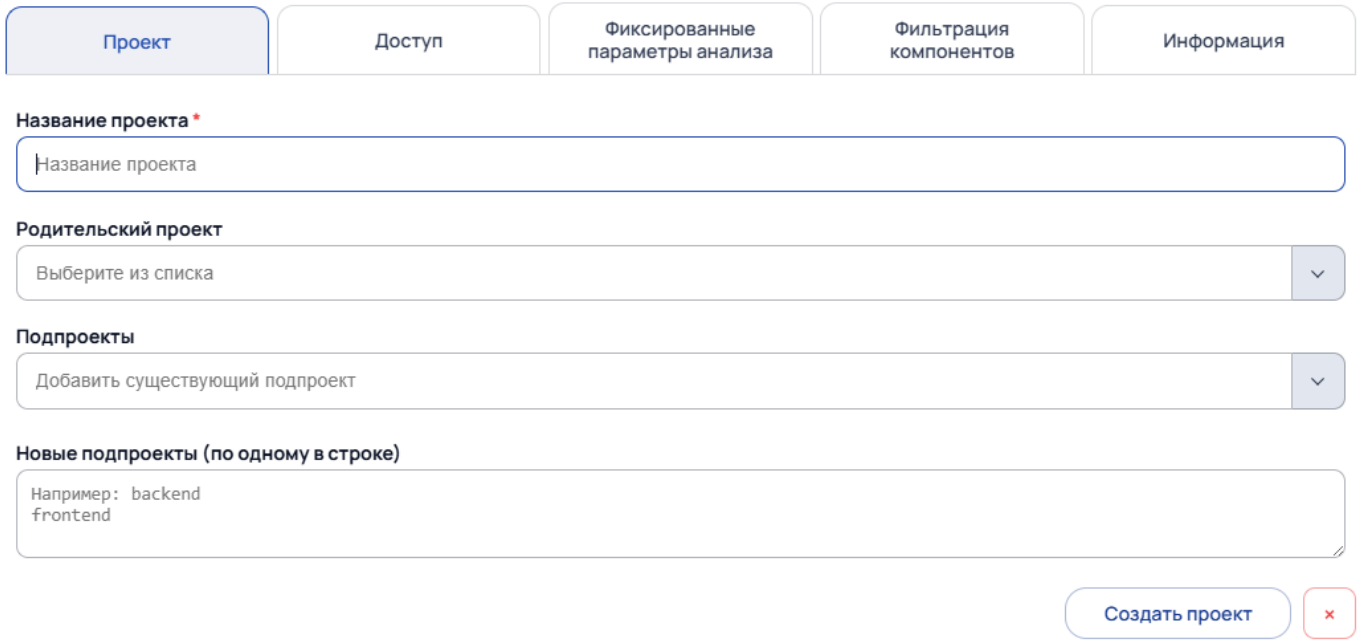

Рисунок 2 – Модальное окно создания проекта

- Вкладка «Доступ» (рисунок 3):
  - выпадающий список «Группы доступа» – позволяет присвоить группе пользователей доступ к создаваемому проекту;
  - выпадающий список «Пользователи доступа» – позволяет присвоить конкретному пользователю доступ к создаваемому проекту.

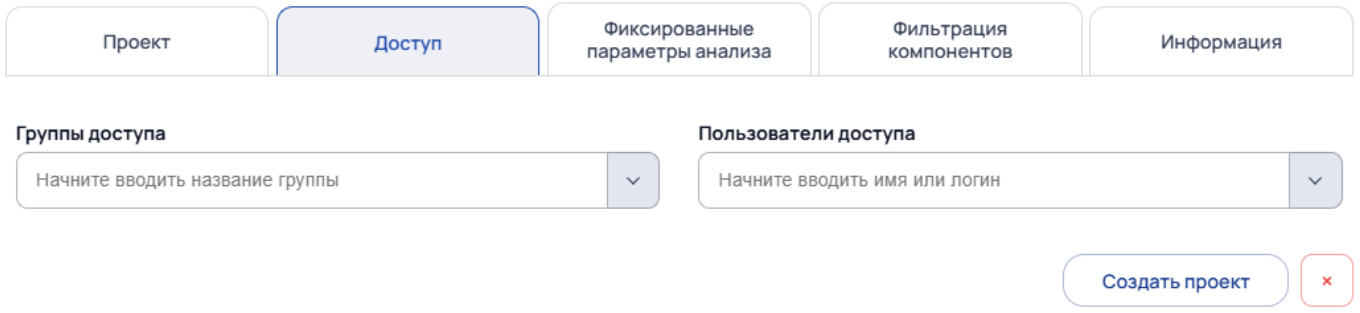

Рисунок 3 – Модальное окно создания проекта

- Вкладка «Фиксированные параметры анализа» (рисунок 4):
  - позволяет жестко фиксировать параметры анализа проекта для обычных пользователей. Описание параметров анализа представлено в п. 3.2.2 настоящего руководства;
  - позволяет жестко задавать анализируемые манифесты компонентов проекта. Правило задаётся в формате `целевой_манифест=файл_проекта`. Например, `conanfile.py=project.build.py` скажет анализу считать `project.build.py` Conan-рецептом. Правила проекта наследуются подпроектами.

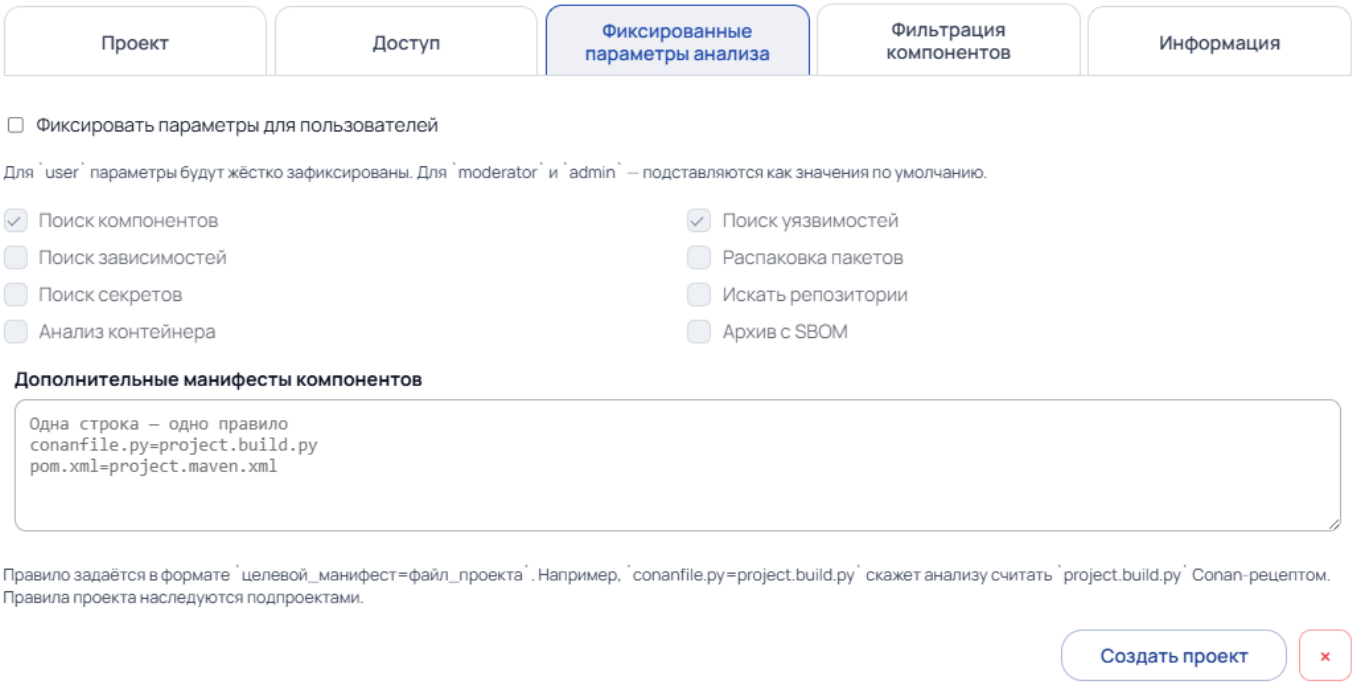

Рисунок 4 – Модальное окно создания проекта

– Вкладка «Фильтрация компонентов» (рисунок 5);
  – позволяет исключать компоненты из анализа по заданным шаблонам. Поддерживаются правила `name=` и `purl=` для поиска по подстроке, а также `name~=` и `purl~=` для регулярных выражений. Без `*` используется поиск по подстроке. Правила проекта наследуются всеми подпроектами;
  – позволяет задавать правила для разметки доверенных компонентов («provided_by» в рамках SBOM-файла). Каждый список задаёт значение `GOST:provided_by` и правила компонентов, к которым оно применяется. Списки наследуются подпроектами;

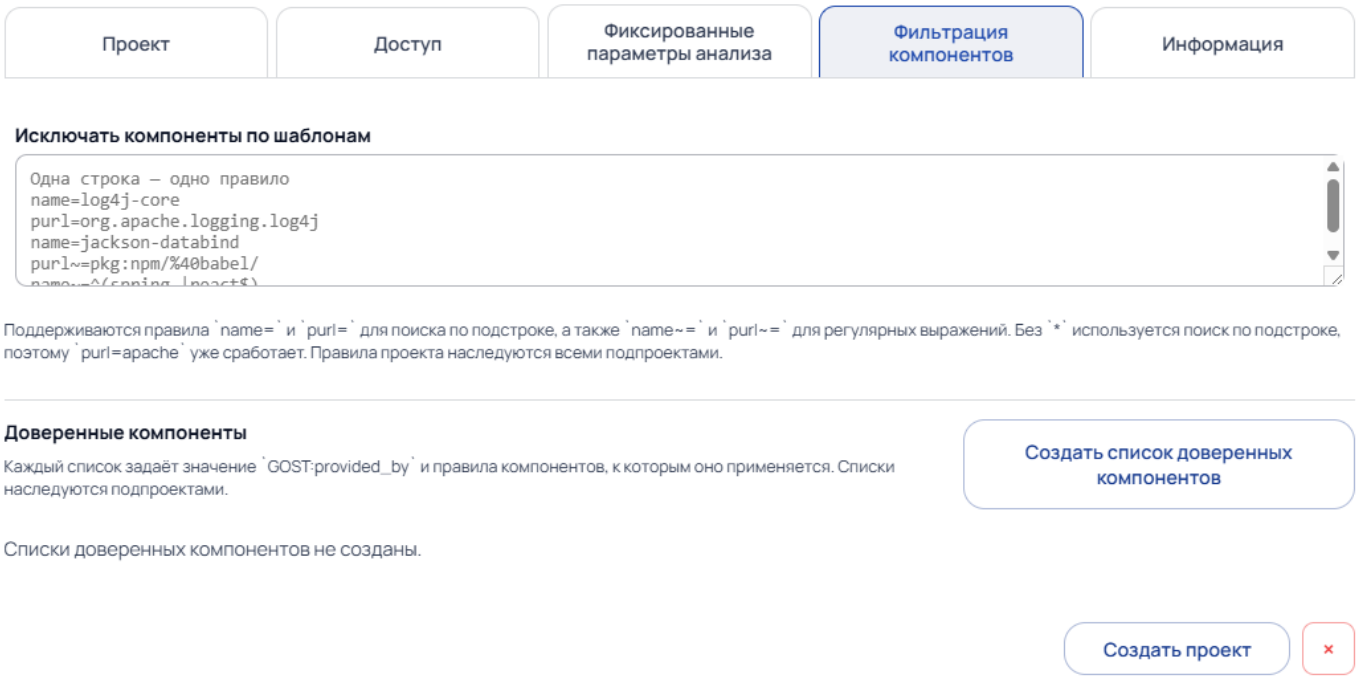

Рисунок 5 – Модальное окно создания проекта

– вкладка «Информация» (рисунок 6) – позволяет фиксировать информацию о проекте, которая в дальнейшем будет отображаться в формируемых SBOM-файлах:
  – поле «Название продукта»;
  – поле «Версия продукта»;
  – поле «Версия SBOM»;
  – поле «Заявитель» – название организации, разрабатывающей продукт.

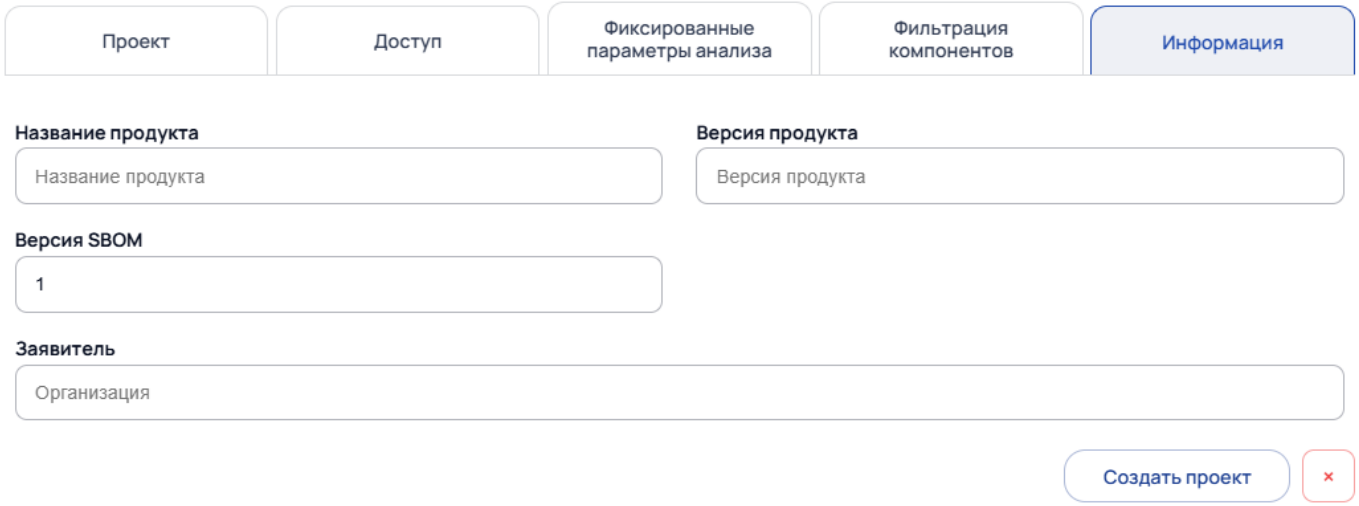

Рисунок 6 – Модальное окно создания проекта

После настройки и сохранения проект появится в общем списке проектов во вкладке «Проекты» (рисунок 7).

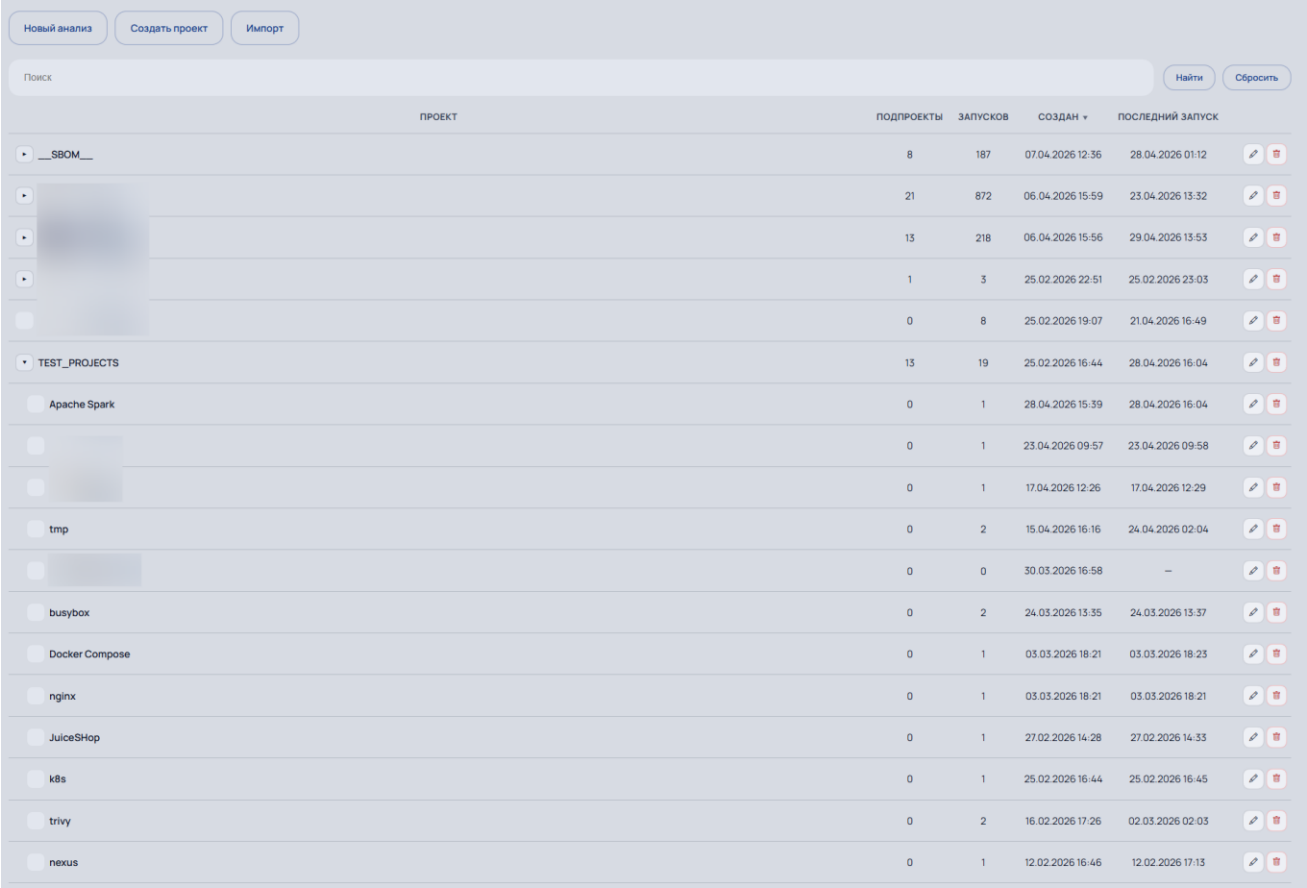

Рисунок 7 – Список проектов

3.2.2 Настройка и запуск анализа

По нажатии кнопки «Новый анализ» открывается модальное окно (рисунок 8), позволяющее произвести предварительную настройку анализа перед запуском:

- выбор проекта из общего списка проектов;
- выбор источника анализа, а именно:
  - по ссылке на репозиторий с проектом:
    - с возможностью рекурсивной загрузки submodule (после git clone выполняется git submodule update -init -recursive);
  - через загрузку архива с проектом;
- настройка анализа:
  - параметры анализа:
    - чекбокс «Поиск компонентов» – формирует список заимствованных компонентов проекта;
    - чекбокс «Поиск уязвимостей» – проверяет заимствованные компоненты проекта на предмет наличия в них известных уязвимостей;
    - чекбокс «Поиск зависимостей» – ищет зависимости проекта через системы сборки и менеджеры пакетов;
    - чекбокс «Распаковка пакетов» – Извлекает содержимое пакетов (.rpm/, .apk/, .nupkg/, .whl/, .deb) и Java-архивов (.jar/, .war/, .ear) внутри архива проекта;
    - чекбокс «Поиск секретов» – ищет ключи, токены и пароли в файлах проекта;
    - чекбокс «Искать репозитории» – поиск ссылок на репозитории и архивы с исходными кодами заимствованных компонентов проекта;
  - режимы анализа:
    - чекбокс «Анализ контейнера» – анализирует установленные пакеты в контейнерах без учета транзитивности;
    - чекбокс «Архив с SBOM» – анализ архива с несколькими SBOM JSON (каждый файл анализируется отдельно).

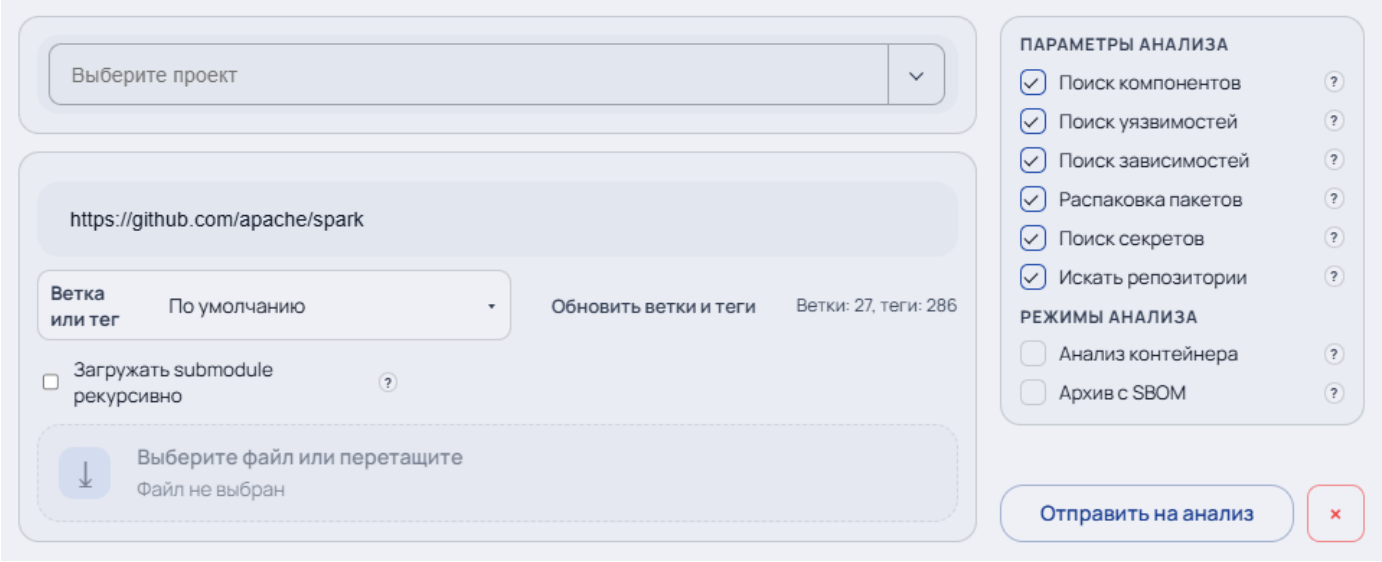

Рисунок 8 – Новый анализ

3.2.3 Запуск анализа

После запуска анализа откроется страница со статусом (рисунок 9). Страница со статусом анализа содержит следующую информацию:

- наименование загруженного архива/репозитория:
- время старта анализа;
- продолжительность анализа в реальном времени;
- сконфигурированные ранее параметры анализа;
- статус-бары с этапами анализа;
- кнопка «Нагрузка» – позволяет отслеживать затрачиваемые ресурсы сервера на проведение анализа;
- кнопка «К результатам анализа» – позволяет сразу перейти на страницу с результатами анализа проекта.

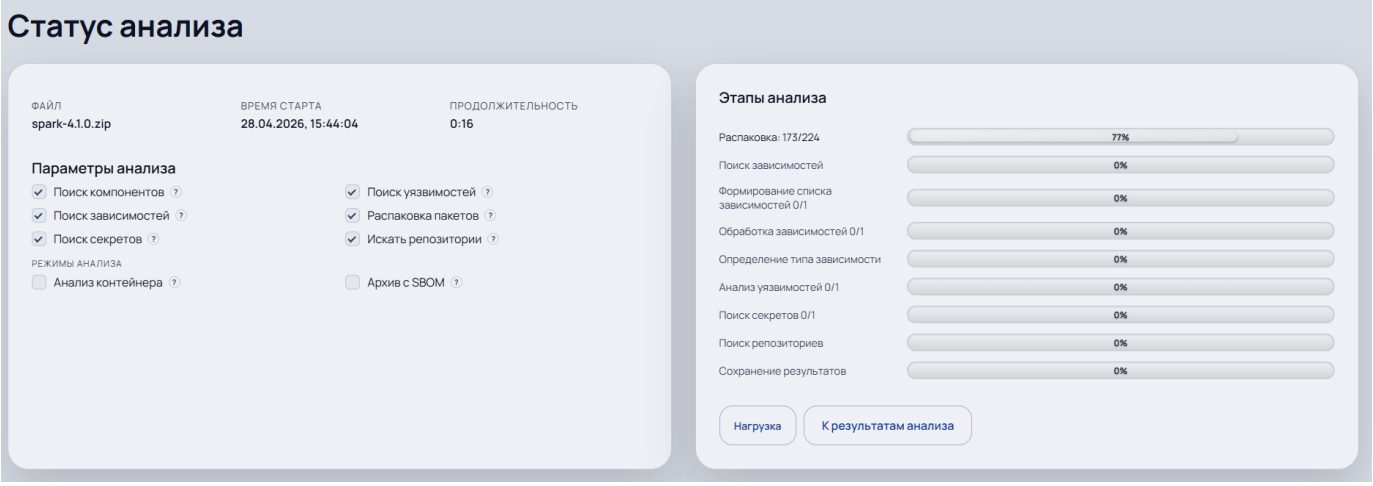

Рисунок 9 – Статус анализа

3.2.4 Работа с проектом

3.2.4.1 Общие сведения о проекте

По завершении анализа в созданном ранее проекте (рисунок 7) появится карточка с результатами анализа (рисунок 10).

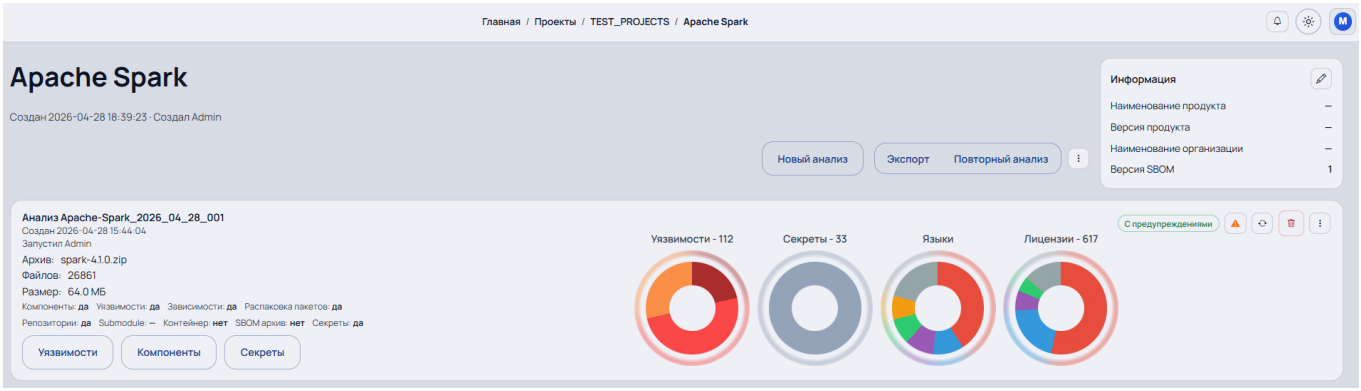

Рисунок 10 – Карточка с результатами анализа

Карточка анализа содержит следующие кнопки:

- кнопка «Повторный анализ по SBOM» ( ) – после получения результатов анализа, информация о них сохраняется в СУБД в формате SBOM-файла, который можно автоматически повторно анализировать для обнаружения новых уязвимостей;
- кнопка «Удалить анализ» ( );
- кнопка «Переместить анализ» ( ) – позволяет переместить анализ в другой проект.

На главной странице проекта также расположены следующие данные:

- кнопка «Новый анализ» – переносит в модальное окно (рисунок 8) для создания нового анализа проекта. Полезно, когда в проект были внесены изменения и нужно провести новый анализ внутри проекта.
- кнопка «Экспорт» – формирует файл формата JSON с метаданными о проекте. Полезно, когда нужно перенести работу над проектом на другой экземпляр SORCA.
- поле «Информация» – дублирует поле из модального окна создания проекта (рисунок 6).
- кнопка «Параметры» ( ), которая позволяет:
  - формировать отчеты:
    - об уязвимостях (в форматах HTML, PDF, CSV, XLSX);
    - о секретах (в форматах HTML, PDF, CSV, XLSX);
    - SBOM:
      - единым SBOM JSON (полезно, когда в проекте много подпроектов и нужно получить общий SBOM по всему проекту, фиксируются только последние анализы подпроектов);
      - «Сырой» SBOM JSON;
      - SBOM JSON по формату требований ФСТЭК России;
      - SBOM odt по формату требований ФСТЭК России;
      - SBOM JSON фиксацией уязвимостей в компонентах.
  - осуществлять следующие действия:
    - переместить проект – позволяет переместить проект в другой проект в качестве подпроекта, а также сделать проект корневым, если он уже является подпроектом.
    - переместить анализы одного проекта в другой проект;
    - сделать копию проекта;
    - выполнить поиск репозиториев/архивов с исходными текстами заимствованных компонентов для последнего выполненного анализа;
    - просмотреть компоненты проекта (полезно, когда в проекте много подпроектов и нужно получить общую сводку по всем компонентам проекта);
    - просмотреть уязвимости проекта (полезно, когда в проекте много подпроектов и нужно получить общую сводку по всем уязвимостям проекта);
    - сравнить выполненные анализы;
    - редактировать проект;
    - удалить проект.

3.2.4.2 Общие сведения об анализе

При переходе по карточке откроется страница с подробной информацией об анализе (рисунок 11):

– Общая информация:
  – статус анализа;
  – наименование архива/репозитория проекта;
  – количество файлов в проекте;
  – размер проекта;
  – дата и время начала анализа;
  – дата и время завершения анализа;
  – график выявленных уязвимостей проекта;
  – график выявленных секретов проекта;
  – график распределения проекта по языкам программирования;
  – график распределения лицензий в заимствованных компонентах проекта.

Основную часть страницы занимает рабочее пространство с выявленным уязвимостями, перечнем заимствованных компонентов, выявленным секретами и выявленными лицензиями заимствованных компонентов.

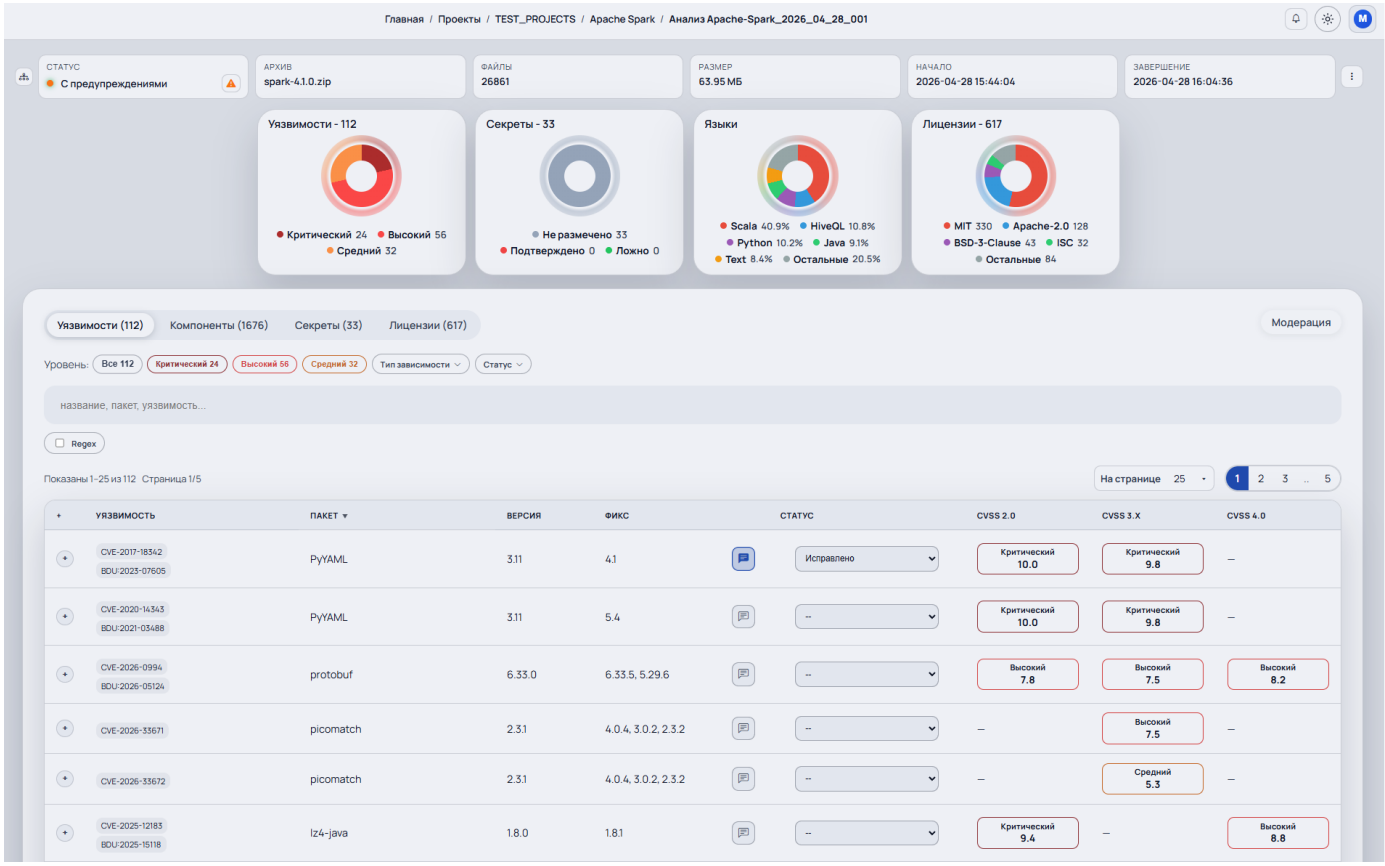

Рисунок 11 – Информация о проекте после анализа

3.2.4.3 Работа с уязвимостями

При переходе на вкладку с уязвимостями открывается список выявленных уязвимостей. Для работы с уязвимостями доступно 2 режима:

- Разметка (рисунок 12) – позволяет проводить анализ и разметку выявленных уязвимостей, и содержит следующую информацию:
  - ID уязвимости – выводятся идентификаторы BDU (при наличии, имеет приоритетный вывод) и CVE;
  - наименование уязвимого компонента;
  - версия уязвимого компонента;
  - версия уязвимого компонента с исправленной уязвимостью;
  - поле «Статус», которое содержит:
    - поле «Комментарий» – для проведения разметки уязвимостей;
    - выпадающий список «Статус», который содержит:
      - статус «Подтверждена»;
      - статус «Исправлена»;
      - статус «Не подтверждена»;
      - статус «Дубликат»;
      - статус «Ложная»;
      - статус «Не будет исправлено».
    - стандарт критичности уязвимости CVSS 2.0;
    - стандарт критичности уязвимости CVSS 3.x;
    - стандарт критичности уязвимости CVSS 4.0.

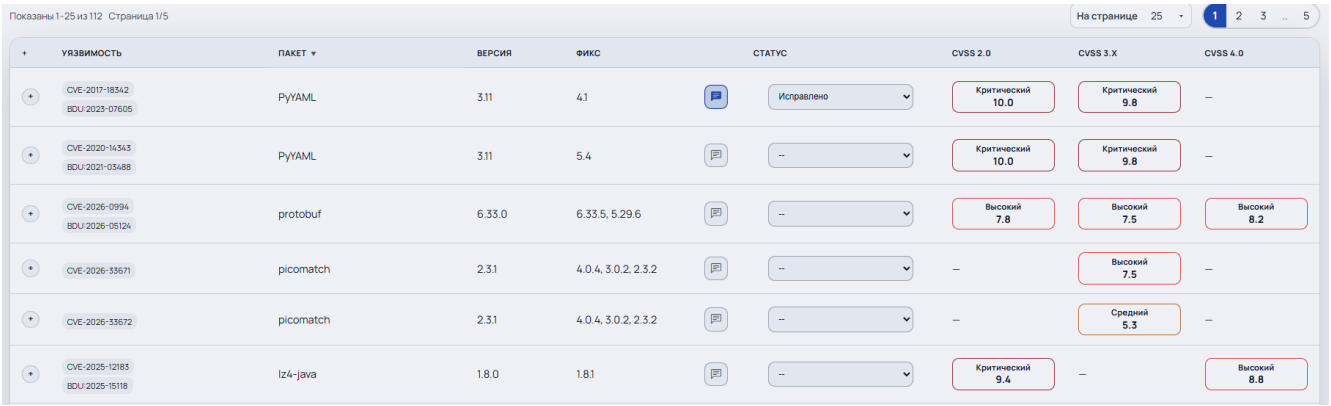

Рисунок 12 – Режим разметки уязвимостей

- модерация (рисунок 13) – позволяет проводить анализ и модерацию размеченных уязвимостей, и содержит следующую информацию:
  - ID уязвимости – выводятся идентификаторы BDU (при наличии, имеет приоритетный вывод) и CVE;
  - наименование уязвимого компонента;
  - версия уязвимого компонента;
  - критичность уязвимости по высшей планке подсчета на основе каждого стандарта CVSS.
  - статус уязвимости;
  - комментарий к уязвимости после проведения разметки.

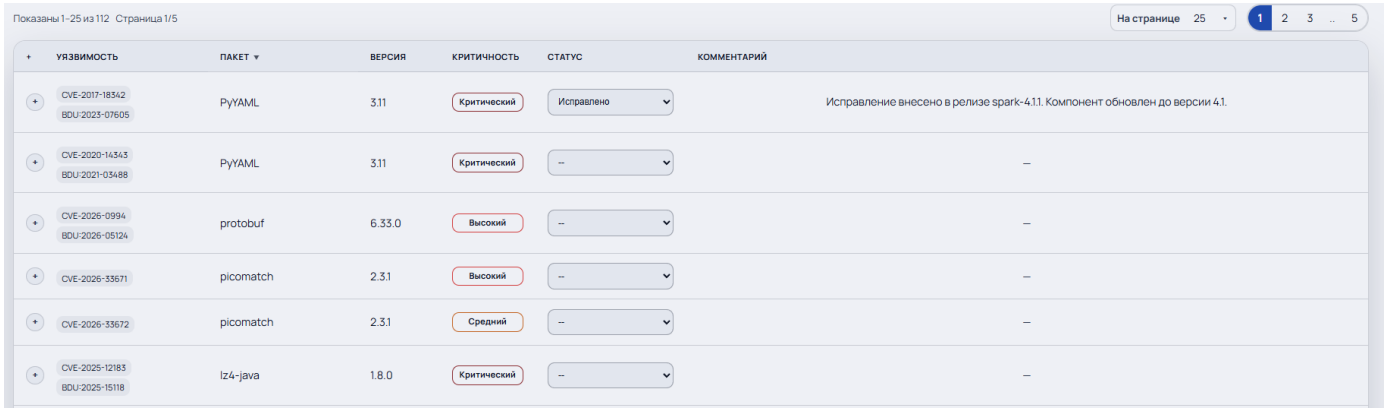

Рисунок 13 – Режим модерации разметки

В каждом режиме также доступна кнопка «Больше» (+) (рисунок 14), являющаяся кнопкой раскрытия списка, который содержит следующую информацию:

- поле «Описание» – содержит основную информацию об уязвимости;
- поле «Даты» – содержит даты публикации уязвимости и обновления информации о ней;
- поле «Тип зависимости» (при наличии) – содержит информацию о типе зависимости (применимо для Java и JavaScript);
- поле «Цель» – указывает на используемый при анализе пакетный менеджер
- поле «Ссылки» – содержит ссылки на все возможные источники с информацией об уязвимости;
- поле «CVSS» – содержит числовой рейтинг (CVSS) и вектор атаки уязвимости;
- поле «Файл» – указывает на путь к файлу, в котором был обнаружен уязвимый компонент.
- поле «Комментарий» – содержит комментарий разметки уязвимости.

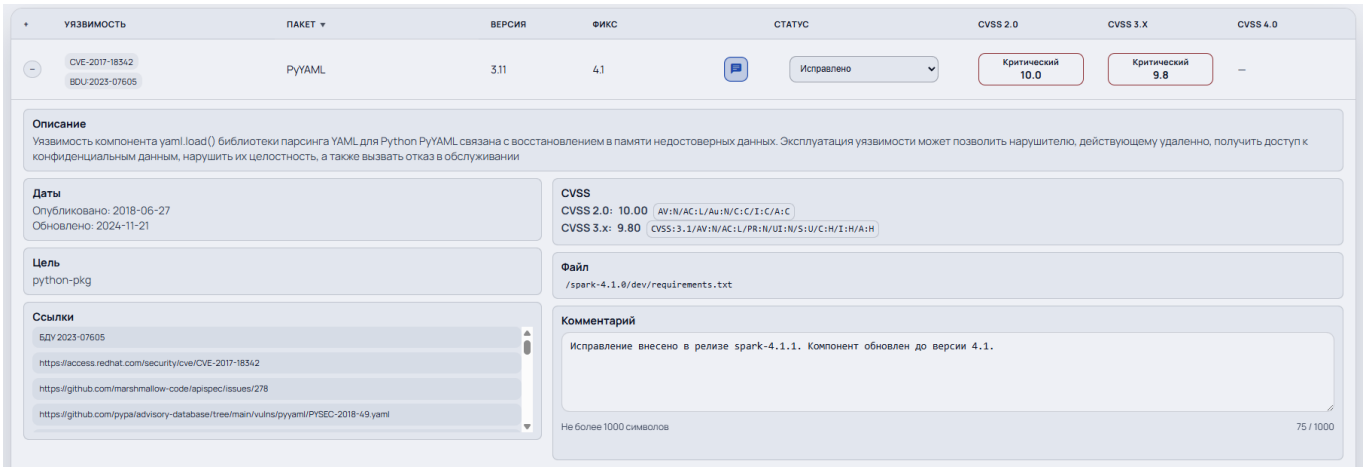

Рисунок 14 – Подробная информация об уязвимости

Для удобного анализа уязвимостей представлена фильтрация по:

- уровню критичности;
- типу зависимости:
  - для Maven:
    - compile – По умолчанию. Зависимости с этим scope доступны на всех этапах и упаковываются в финальный артефакт (JAR, WAR и другие форматы).

- provided – Используется на этапе сборки и тестирования проекта. Предполагается, что зависимость предоставит среда выполнения (например, контейнер сервлетов или сервер приложений).
- runtime – Зависимости с этим scope нужны для выполнения исходного кода, но не для компиляции.
- test – Зависимости с этим scope нужны только для компиляции и запуска тестов, а не для производственного кода.
- system – Зависимости с этим scope не извлекаются из репозитория Maven, а ссылаются на локальную систему. Этот scope обычно не рекомендуется, так как он обходит управление зависимостями Maven.
- import – Используется только в Maven 2.0.9 и выше. Применяется в разделе dependency Management файла pom для импорта информации об управлении зависимостями из других файлов POM в текущий проект.

– для JavaScript:
- runtime – Основные зависимости, необходимые для работы приложения в продакшене. Включают библиотеки и модули, на которые приложение полагается во время выполнения.
- dev – Зависимости, необходимые только для разработки. Обычно включают фреймворки тестирования, инструменты сборки, линтеры и другие утилиты, используемые на этапе разработки.
- optional – Зависимости, которые не критичны для работы приложения, но могут быть использованы, если доступны. Если установка одной из этих зависимостей не удалась, это не приведёт к ошибке.
- peer – Зависимости, которые должны быть установлены разработчиком совместно с пакетом. Указывают, что нужные зависимости уже должны быть установлены на стороне разработчика, либо они должны быть доступны в рабочем окружении, чтобы пакет мог корректно функционировать.

– статусу разметки уязвимостей.

3.2.4.4 Работа с компонентами

При переходе на вкладку с компонентами (рисунок 15) открывается список выявленных заимствованных компонентов. Вкладка содержит следующую информацию:

- наименование компонента;
- версия компонента;
- язык программирования, на котором написан компонент;
- поле «Поверхность атаки» – для определения компонента, как лежащего на поверхности атаки;
- поле «Функции безопасности» – для определения компонента, как выполняющего функции безопасности;
- поле «Репозиторий» – для фиксации ссылок репозиториев и архивов (с функцией подсчета контрольных сумм по алгоритму STREEBOG-256), которые ведут к исходным текстам компонента;
- поле «Пакет» – для отображения ссылок на ресурсы, содержащие информацию о компоненте;

– Поле «Файл» – указывает на путь к файлу, в котором был обнаружен компонент.

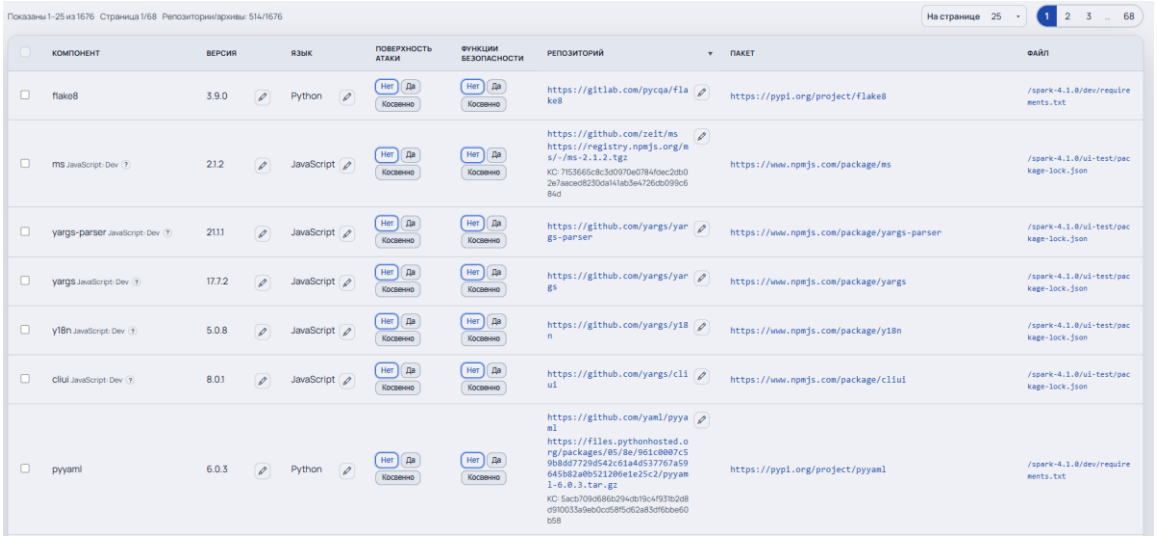

Рисунок 15 – Информация о выявленных заимствованных компонентах

Для анализа транзитивности выявленных компонентов представлена кнопка «Дерево зависимостей» (), которая ведет к графику зависимостей анализа (рисунок 16).

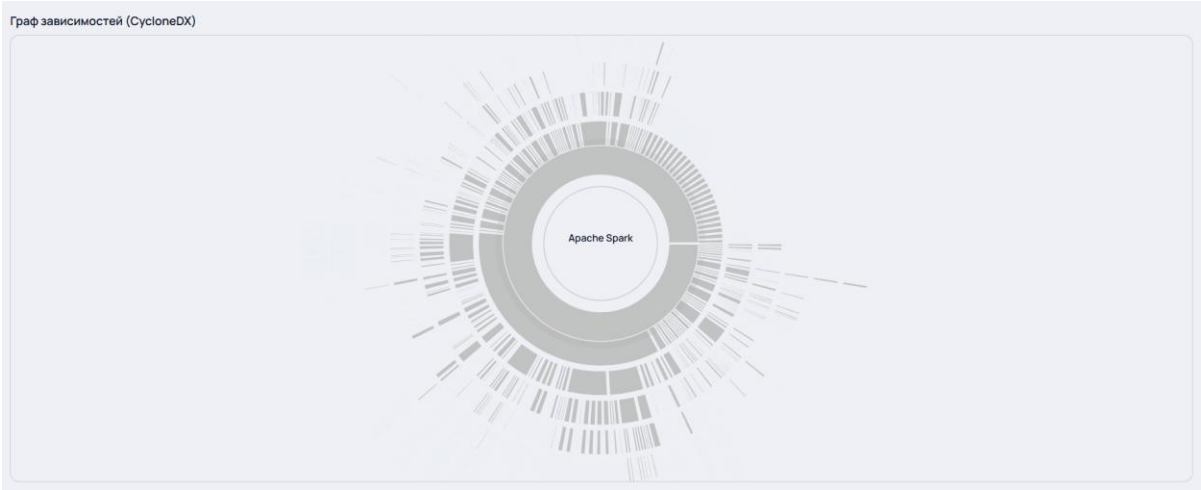

Рисунок 16 – График зависимостей

Для удобного анализа компонентов представлена фильтрация по:

– компонентам, версия которых неизвестна;
– компонентам, для которых не был найден репозиторий/архив с исходными текстами;
– компонентам, для которых был найден репозиторий/архив с исходными текстами;
– компонентам, отмеченным, как лежащим на поверхности атаки;
– компонентам, отмеченным, как реализующим функции безопасности;
– типу зависимости:
  – для Maven:
    – compile – По умолчанию. Зависимости с этим scope доступны на всех этапах и упаковываются в финальный артефакт (JAR, WAR и другие форматы).
    – provided – Используется на этапе сборки и тестирования проекта. Предполагается, что зависимость предоставит среда выполнения (например, контейнер сервлетов или сервер приложений).

- runtime – Зависимости с этим scope нужны для выполнения исходного кода, но не для компиляции.
- test – Зависимости с этим scope нужны только для компиляции и запуска тестов, а не для производственного кода.
- system – Зависимости с этим scope не извлекаются из репозитория Maven, а ссылаются на локальную систему. Этот scope обычно не рекомендуется, так как он обходит управление зависимостями Maven.
- import – Используется только в Maven 2.0.9 и выше. Применяется в разделе dependency Management файла pom для импорта информации об управлении зависимостями из других файлов POM в текущий проект.

– для JavaScript:
- runtime – Основные зависимости, необходимые для работы приложения в продакшене. Включают библиотеки и модули, на которые приложение полагается во время выполнения.
- dev – Зависимости, необходимые только для разработки. Обычно включают фреймворки тестирования, инструменты сборки, линтеры и другие утилиты, используемые на этапе разработки.
- optional – Зависимости, которые не критичны для работы приложения, но могут быть использованы, если доступны. Если установка одной из этих зависимостей не удалась, это не приведёт к ошибке.
- peer – Зависимости, которые должны быть установлены разработчиком совместно с пакетом. Указывают, что нужные зависимости уже должны быть установлены на стороне разработчика, либо они должны быть доступны в рабочем окружении, чтобы пакет мог корректно функционировать.

3.2.4.5 Работа с секретами

При переходе на вкладку с секретами (рисунок 17) открывается список выявленных секретов. Вкладка содержит следующую информацию:

- поле «Правило» – указывает на правило, по которому найден секрет;
- поле «Файл» – указывает на путь к файлу, в котором был обнаружен секрет;
- поле «Строка» – указывает на строку в файле, в которой был обнаружен секрет;
- поле «Статус» – для указания статуса секрета:
  - ложный;
  - подтвержденный;
- поле «Комментарий» – для проведения разметки уязвимостей.

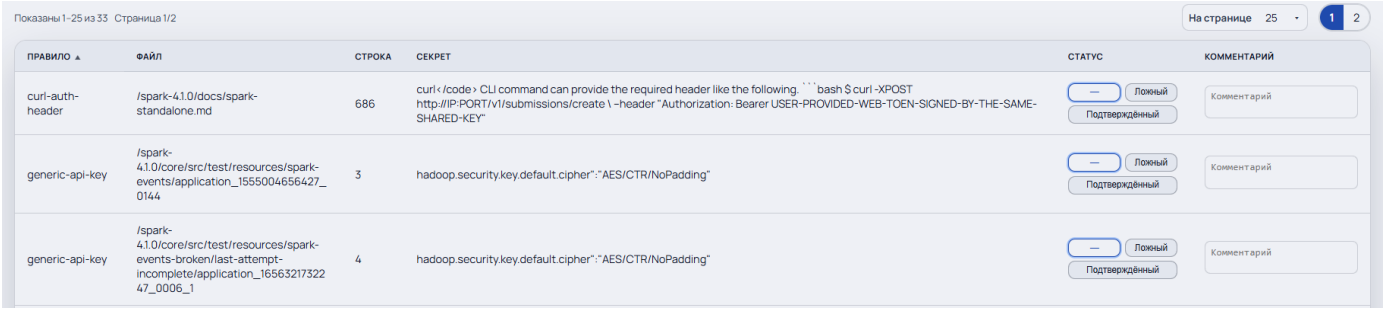

Рисунок 17 – Информация о выявленных секретах

3.2.4.6 Дополнительные действия с анализом

Кнопка «Параметры» ( ) позволяет:

- провести повторный анализ;
- провести поиск репозиториев/архивов с исходными текстами компонентов в случае, если при запуске анализа данный параметр выставлен не был;
- получить языки программирования компонентов в случае, если при запуске анализа данный параметр выставлен не был;
- скопировать разметку анализа в другой анализ;
- импортировать разметку (полезно, когда нужно перенести разметку анализа на другой экземпляр SORCA).

3.2.4.7 Экспорт анализа

Кнопка «Настройка экспорта» ( ) позволяет провести предварительную гибкую настройку отчетов анализа перед выгрузкой (рисунки 18-19).

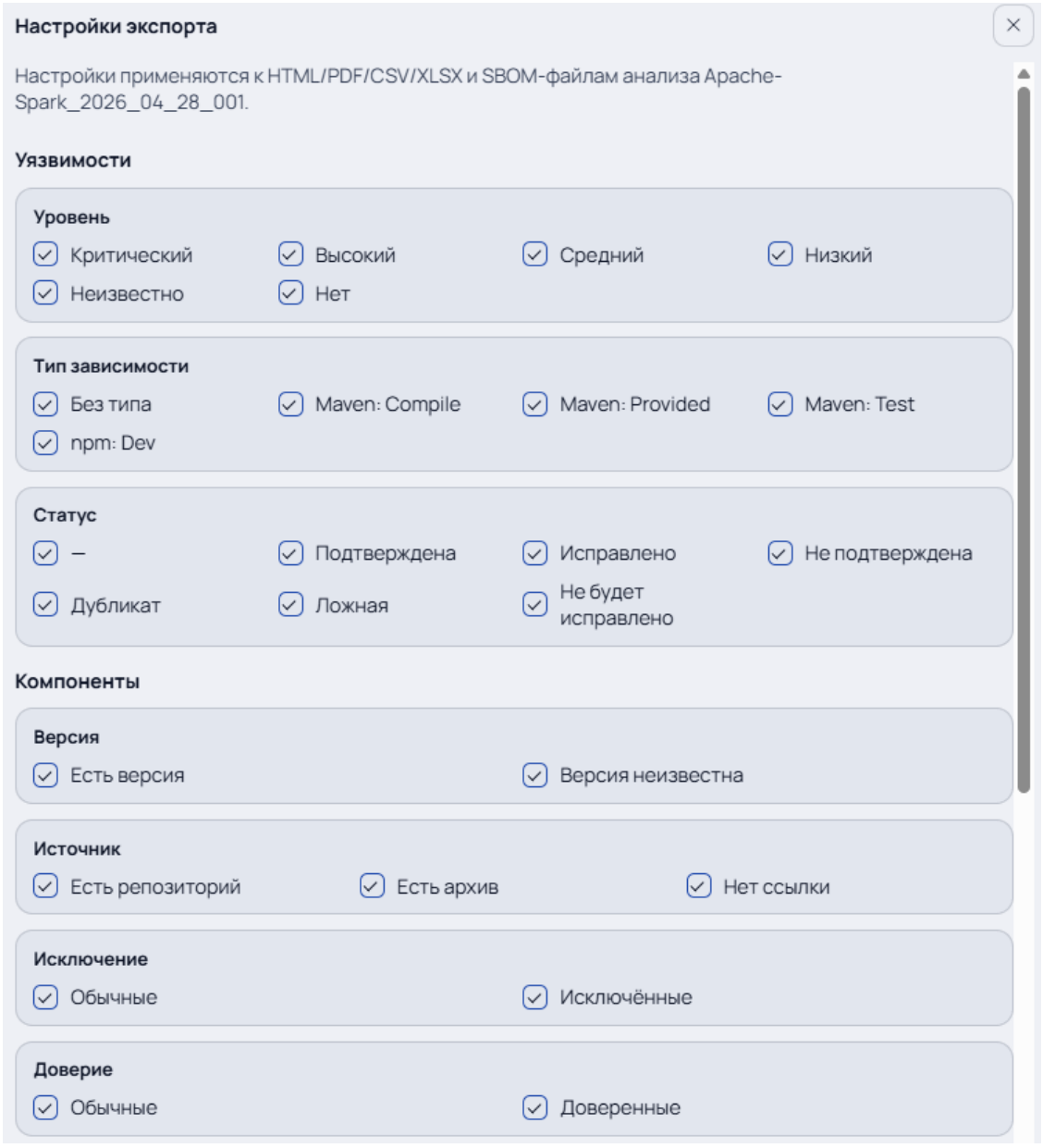

Рисунок 18 – Настройка экспорта

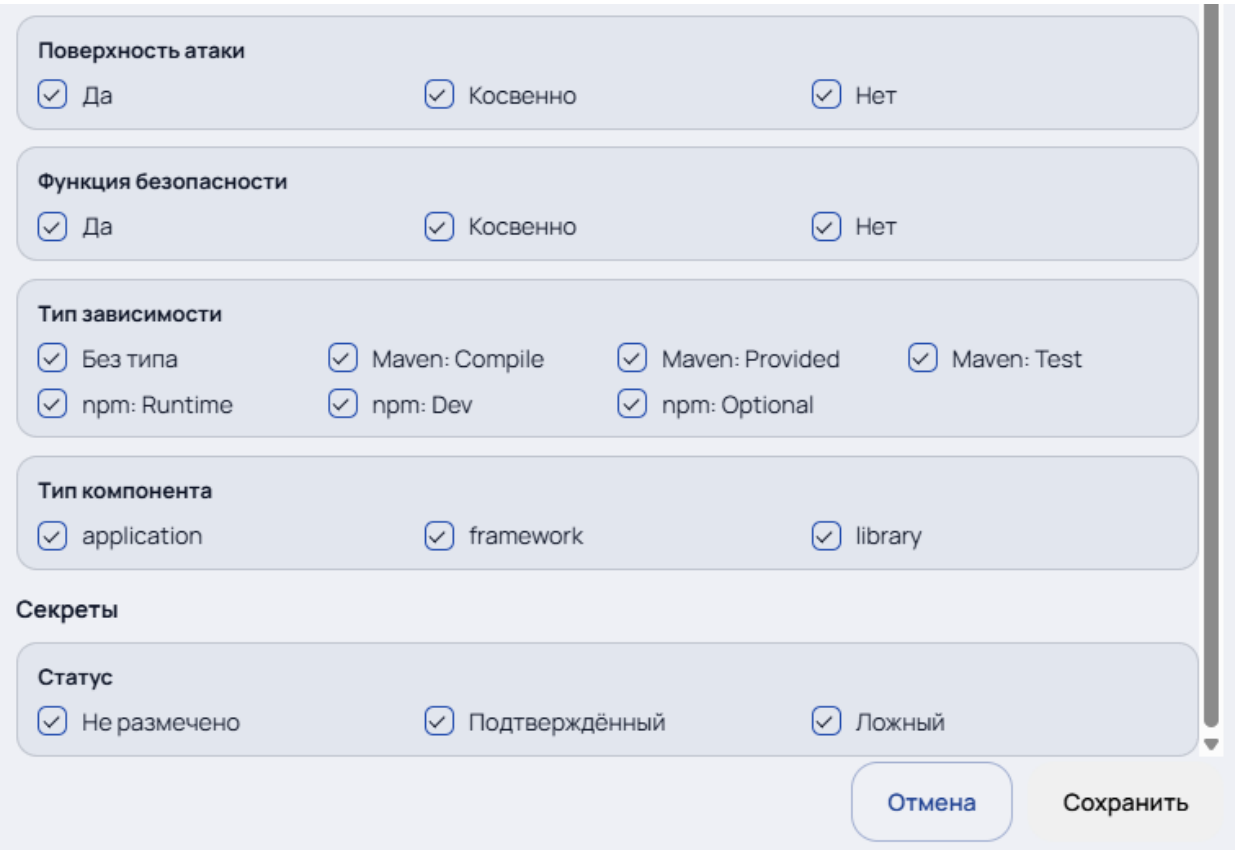

Рисунок 19 – Настройка экспорта

После настройки экспорта можно формировать отчеты:

- об уязвимостях (в форматах HTML, PDF, CSV, XLSX);
- о секретах (в форматах HTML, PDF, CSV, XLSX);
- SBOM:
  - единым SBOM JSON (полезно, когда в проекте много подпроектов и нужно получить общий SBOM по всему проекту, фиксируются только последние анализы подпроектов);
  - «Сырой» SBOM JSON;
  - SBOM JSON по формату требований ФСТЭК России;
  - SBOM odt по формату требований ФСТЭК России;
  - SBOM JSON фиксацией уязвимостей в компонентах.

3.3 Вкладка «Параметры»

Данная вкладка является панелью администратора с функцией управления всем приложением в целом.

Вкладка содержит в себе следующие вкладки:

- вкладка «Обзор» (рисунок 20) – содержит общую информацию о приложении:
  - серверное время;
  - информация о базе данных;
  - информация о свободном месте на сервере;
  - версия приложения;
  - количество репозиториев/архивов с исходными текстами компонентов, сохраненных в базе данных;
  - статус актуальности базы уязвимостей;

– количество и список запущенных задач;
– срок действия лицензии.

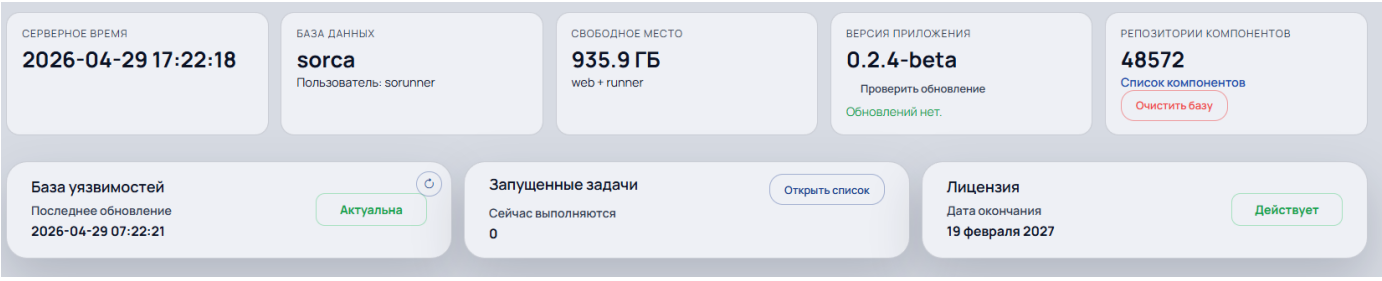

Рисунок 20 – Вкладка «Обзор»

– вкладка «Пользователи» (рисунок 21) – отвечает за создание и обзор пользователей;

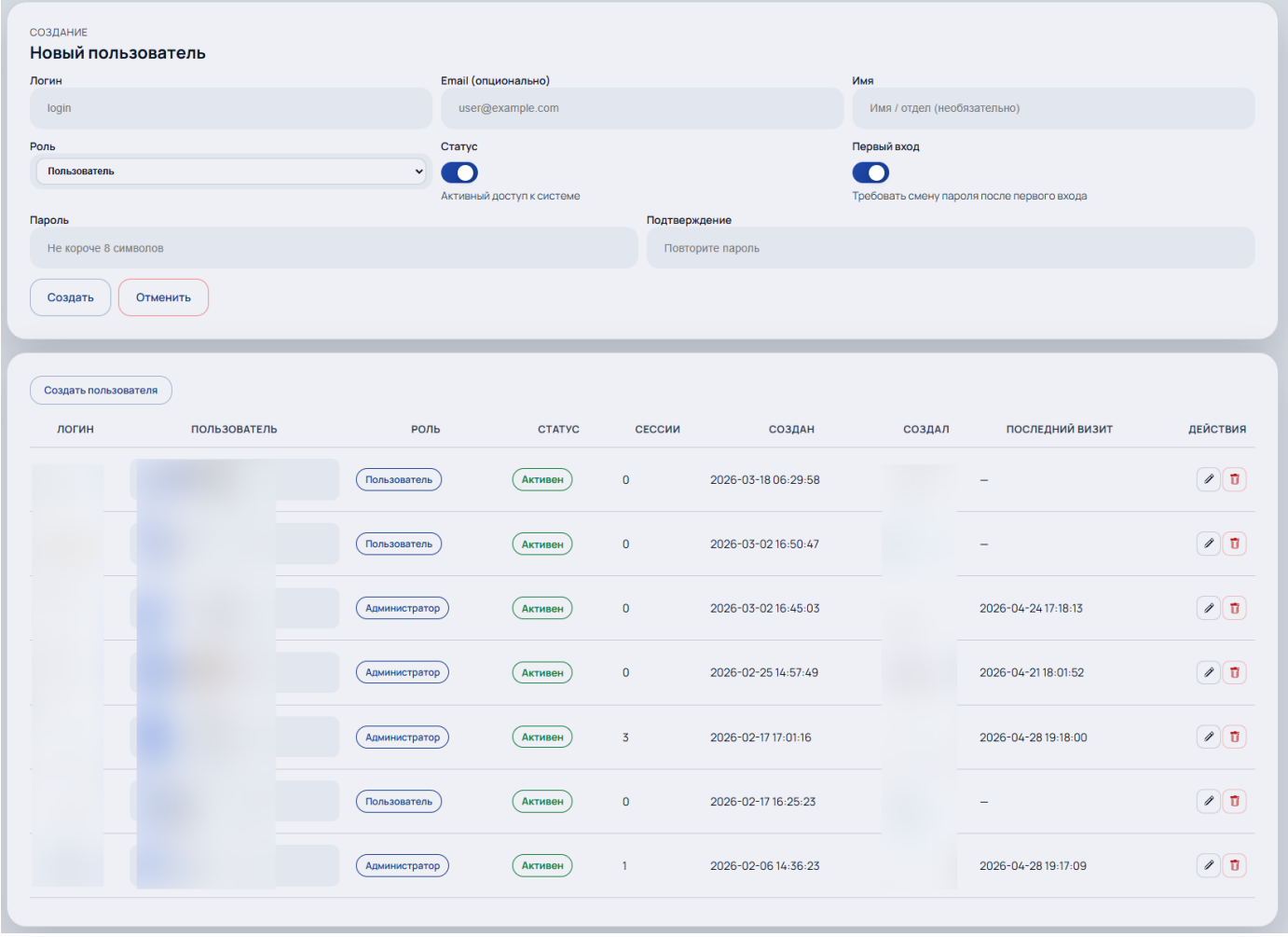

Рисунок 21 – Вкладка «Пользователи»

– вкладка «Параметры анализа» (рисунок 22) – содержит параметры:
  – модуль «Глубина распаковки»:
    – параметр «Каталоги» – Устанавливает глубину анализа каталогов после распаковки.
    – параметр «Архивы» – Устанавливает возможное количество распаковываемых архивов в процессе анализа.
  – модуль «Производительность»:
    – параметр «Потоки анализа» – По умолчанию — половина доступных потоков CPU.

- параметр «Потоки поиска репозиториев» – По умолчанию — 8 потоков.

– модуль «Поиск репозиториев во время анализа»:
  – параметр «Поиск репозиториев» – Если выключено, новые repo URL берутся только из локальной БД.
  – параметр «Поиск архивов» – Если выключено, новые ссылки на архивы берутся только из локальной БД.
– модуль «Время»:
  – параметр «Ожидание решения по базе уязвимостей, мин» – Если пользователь не ответил, анализ продолжится без обновления (только при наличии локальной БД).
  – параметр «Часовой пояс интерфейса» – Время хранится в UTC, отображается в выбранной зоне. По умолчанию — системная зона: UTC+00:00 · Etc/UTC.
– модуль «Сохранение»:
  – параметр «Сохранять бинарные файлы, как компоненты» – Отключите, чтобы не сохранять компоненты с непонятным purl.
  – параметр «Подсчёт КС загружаемого архива» – Выберите алгоритмы (STREEBOG-256, MD5, SHA-1, SHA-256, SHA-512), чтобы сохранять контрольные суммы исходного файла.

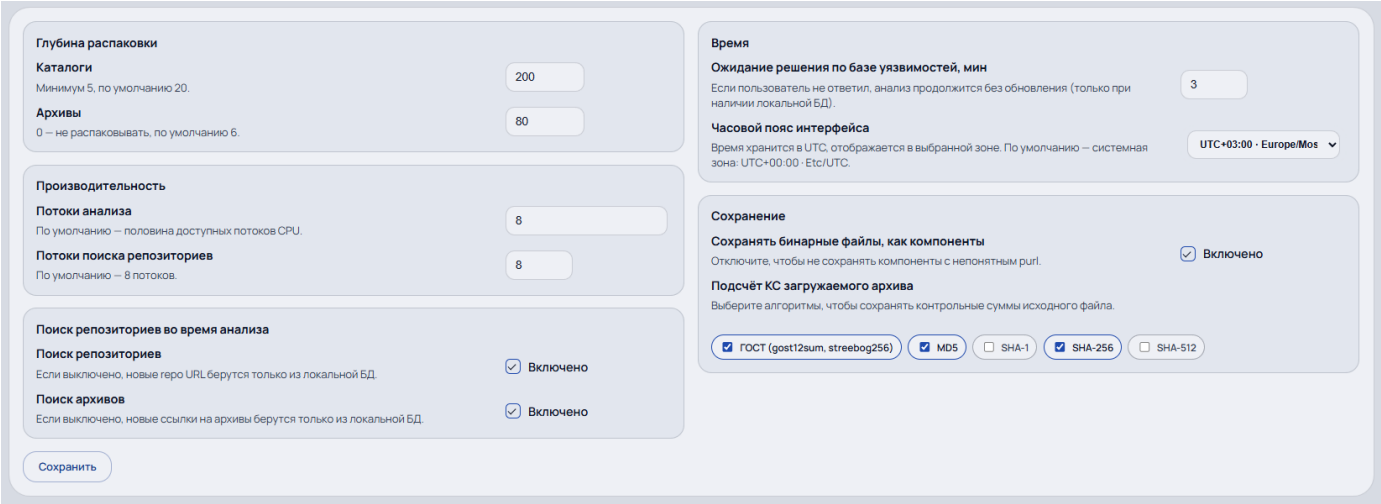

Рисунок 22 – Вкладка «Параметры анализа»

– вкладка «APT репозитории» (рисунок 23) – позволяет вносить в конфигурацию анализов репозитории, которые используются для получения информации о пакетах операционных систем на базе пакетного менеджера APT.

Рисунок 23 – Вкладка «APT репозитории»

– вкладка «RPM репозитории» (рисунок 24) – позволяет вносить в конфигурацию анализов репозитории, которые используются для получения информации о пакетах операционных систем на базе пакетного менеджера RPM.

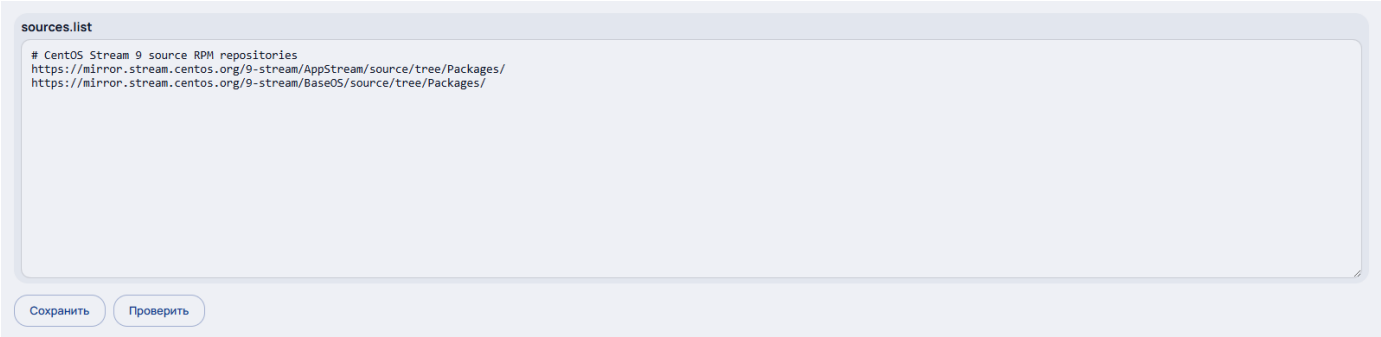

Рисунок 24 – Вкладка «RPM репозитории»

– вкладка «Нагрузка» (рисунок 25) – позволяет отслеживать затрачиваемые ресурсы сервера на проведение анализов.

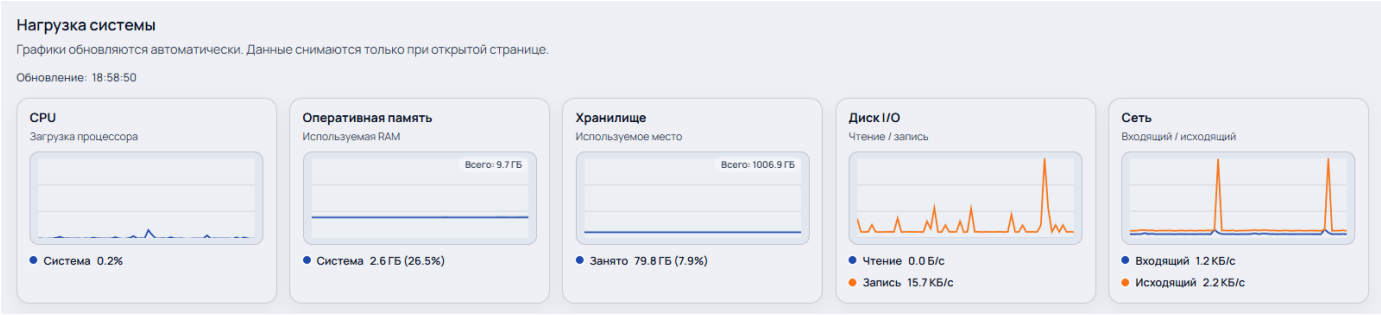

Рисунок 25 – Вкладка «Нагрузка»

– вкладка «Лицензия» (рисунок 26) – позволяет добавлять лицензию для активации функционала приложения.

Рисунок 26 – Вкладка «Лицензия»

– вкладка «Репозитории компонентов» (рисунок 27) – позволяет отслеживать и управлять базой данных репозиториев/архивов с исходными текстами компонентов.

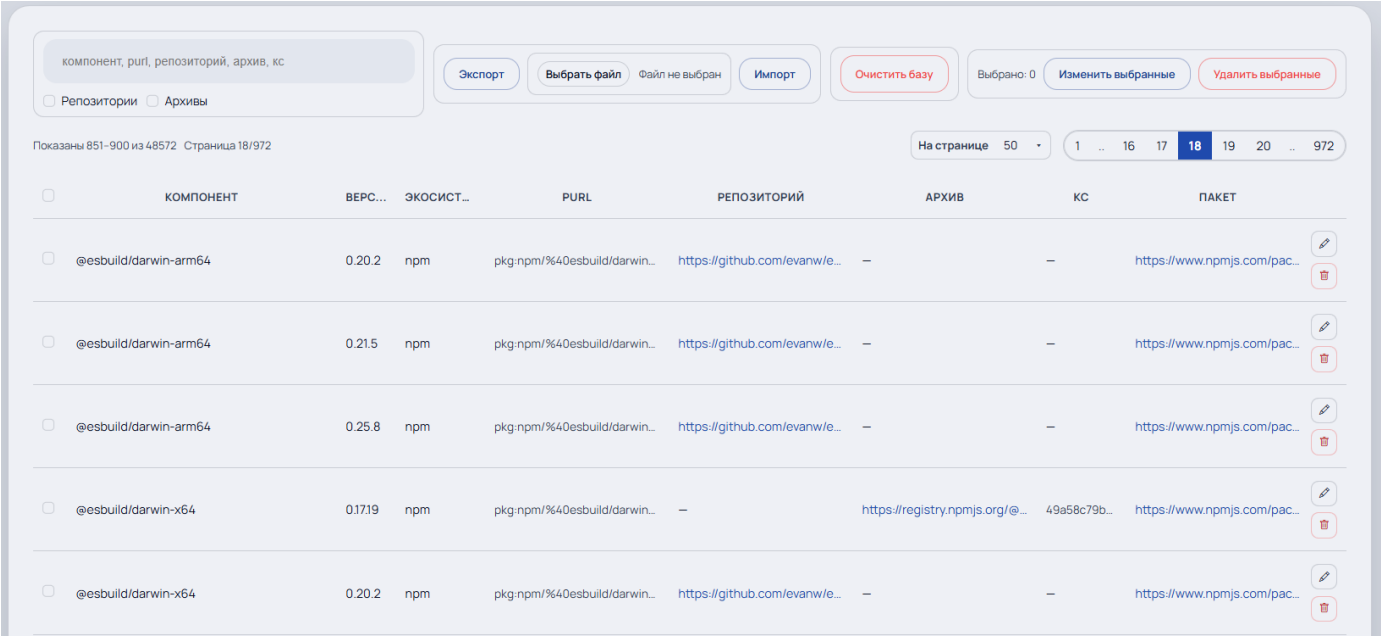

Рисунок 27 – Вкладка «Репозитории компонентов»

– вкладка «Группы» (рисунок 28) – позволяет создавать группы пользователей.

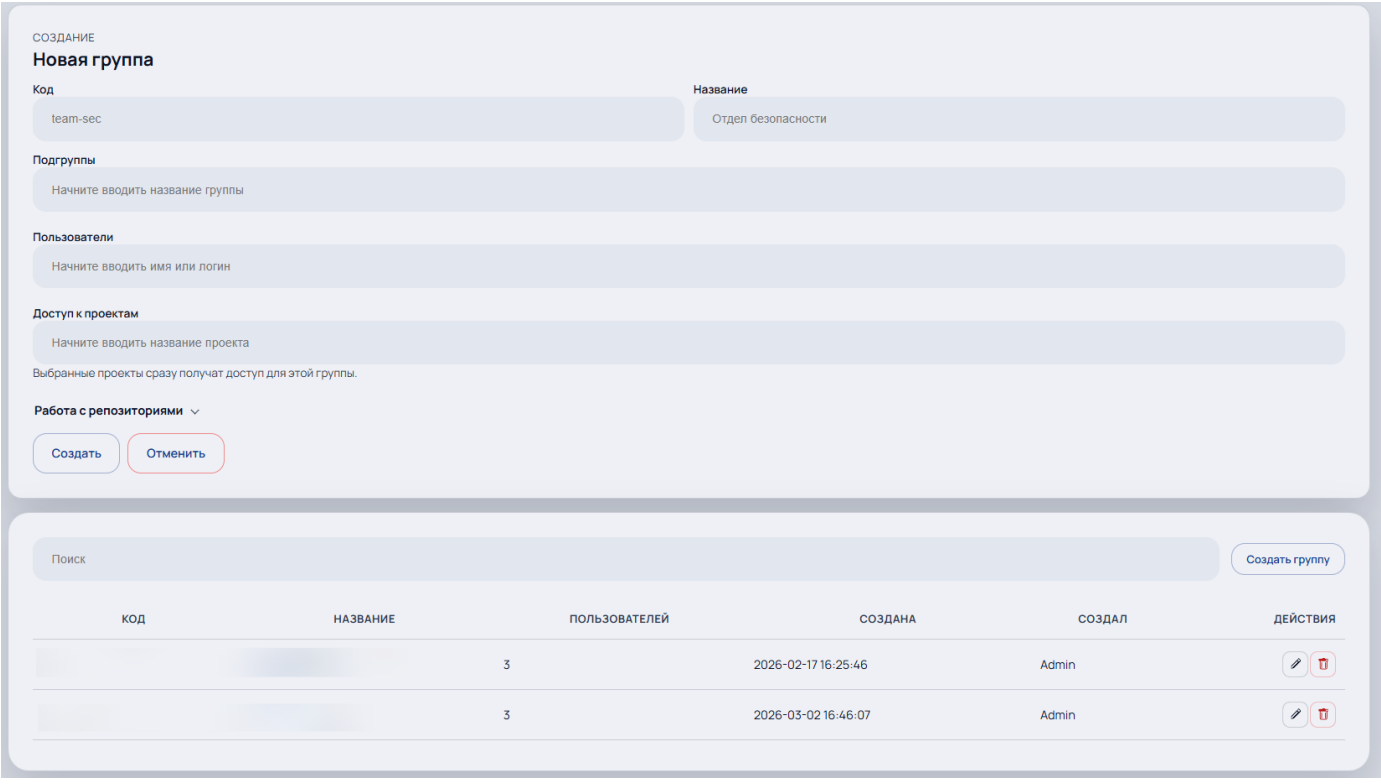

Рисунок 28 – Вкладка «Группы»

– вкладка «Журнал» (рисунок 29) – позволяет отслеживать действия в приложении.

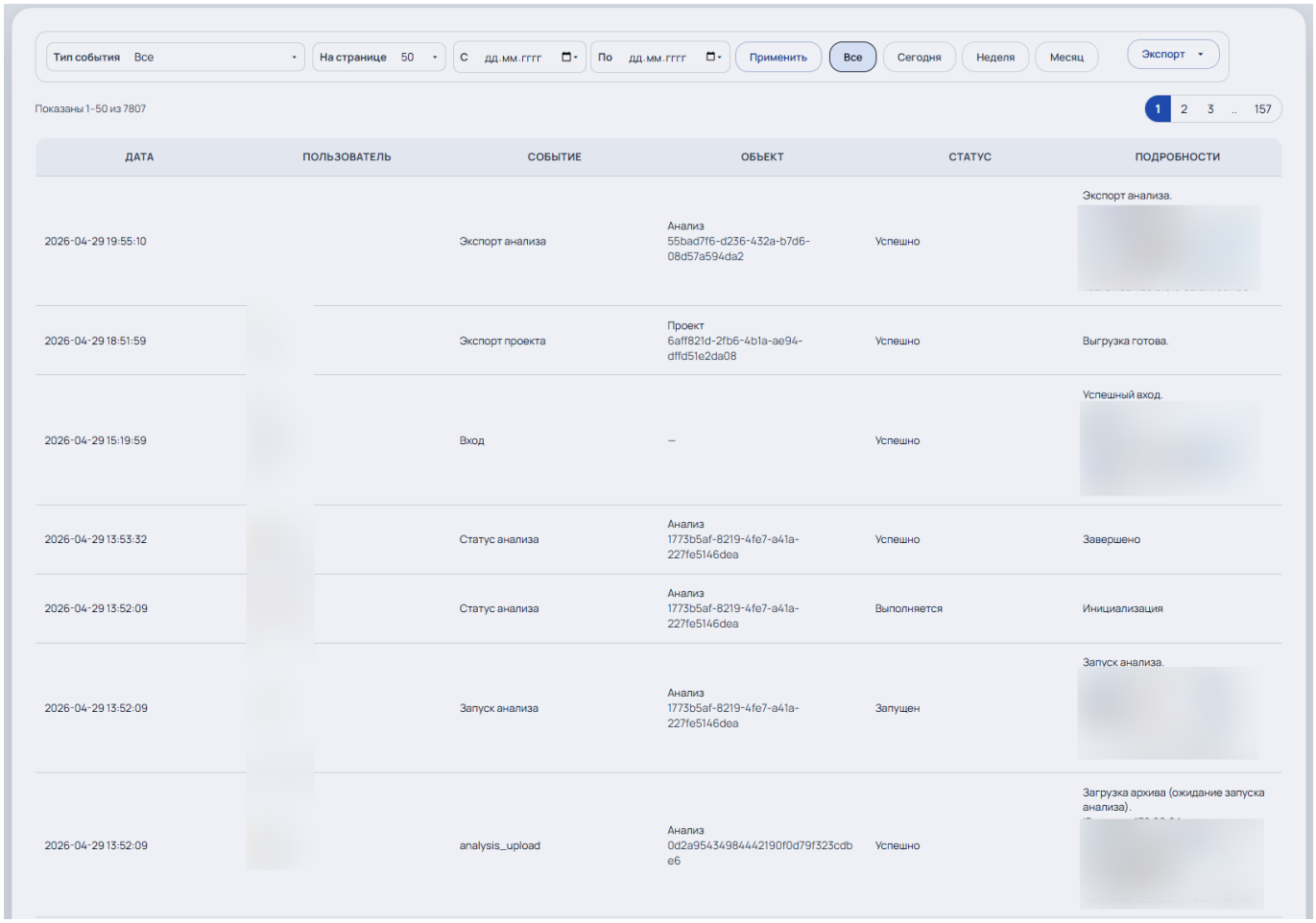

Рисунок 29 – Вкладка «Журнал»

3.4 Вкладка «Профиль»

Вкладка содержит основную информацию о профиле пользователя (рисунок 30), а также позволяет:

– редактировать профиль (рисунок 31);
– добавлять доступ к закрытым репозиториям Github и Gitlab (рисунок 32);
– генерировать API-ключ для доступа консольного агента к основному приложению, а также скачивать конфигурационный файл для консольного агента и непосредственно сам консольный агент (рисунок 33).

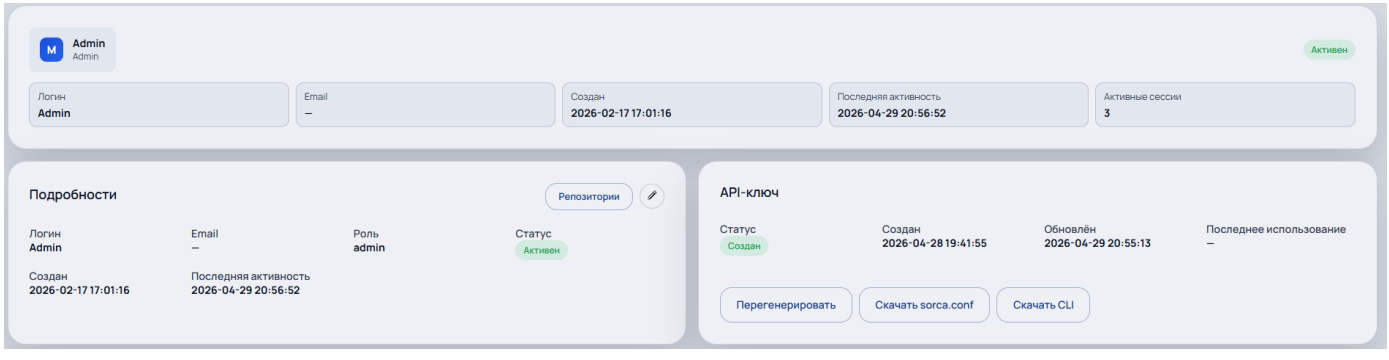

Рисунок 30 – Основная информация о профиле пользователя

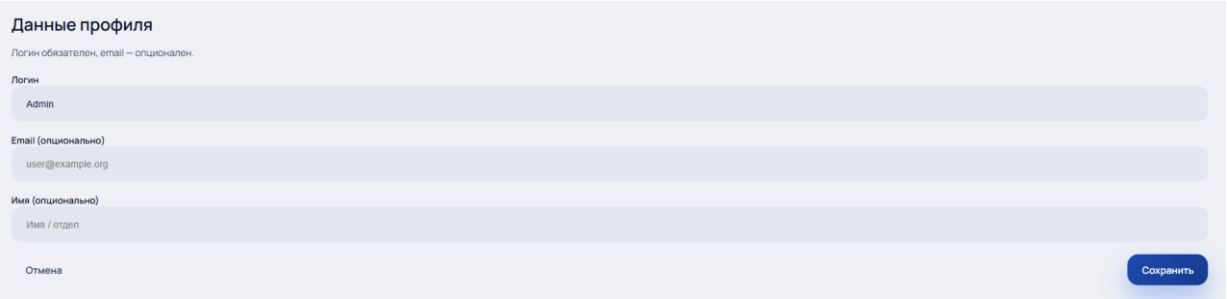

Рисунок 31 – Редактирование профиля

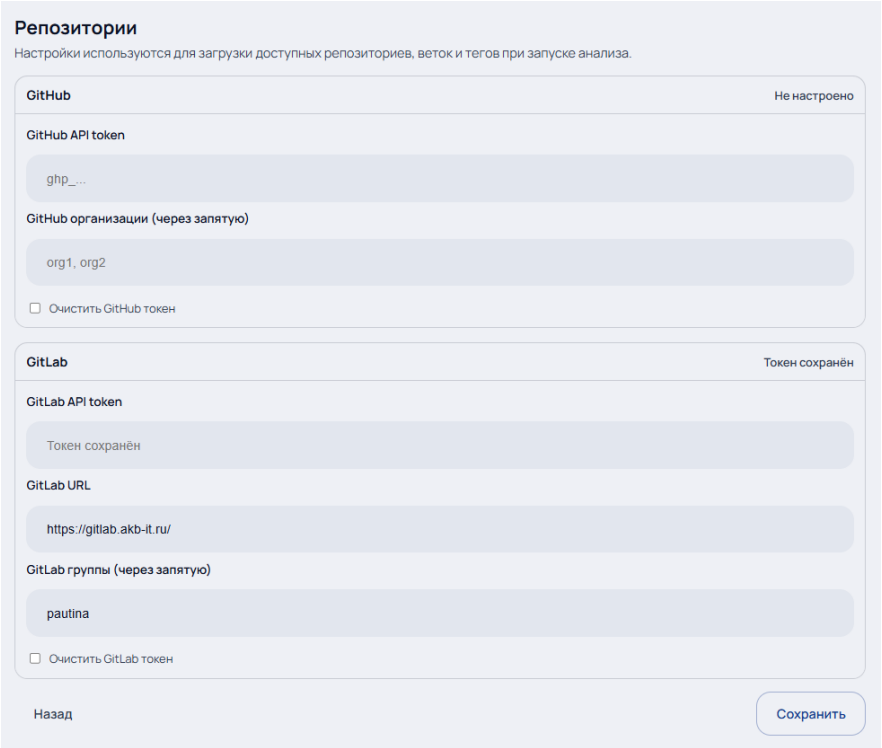

Рисунок 32 – Работа с репозиториями

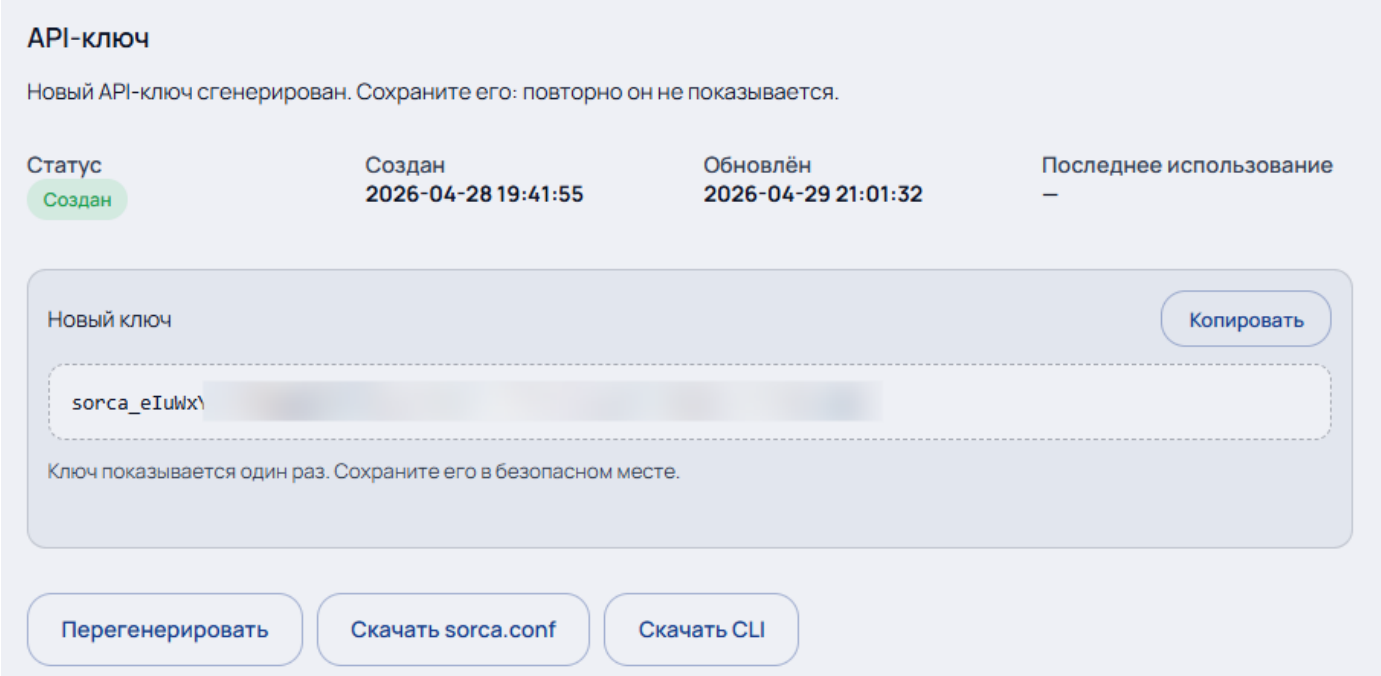

Рисунок 33 – Работа с API

4 Консольный агент

Перед началом работы с консольным агентом требуется произвести конфигурацию файла sorca.conf. Пример содержимого конфигурационного файла представлен в таблице 3.

Таблица 3 – Средства контроля и тестирования

Ключ	Описание	Пример значения
server_url=	Адрес сервера приложения, к которому будет обращаться агент	http://localhost:18080
api_key=	API ключ (рисунок 33) для обращения к серверу приложения	sorca_eIuWxYU2ghpfHbsDEkHiEE jwpnkgndTZEYE87462V6Xkqp5gIJ
project_id=	ID проекта	55bad7f6-d236-432a-b7d6-08d57a594da2
project_name=	Название проекта	console-demo
default_mode=	watch – запускает анализ и ждёт завершения, периодически опрашивая статус раз в poll_interval_sec=, также выводит текущий статус в консоль =sumbit – запускает и не ждёт ничего, возвращает task_id	watch/submit
poll_interval_sec=	Интервал опроса сервера в процессе ожидания результатов анализа	2
components_scan=	Формирует список заимствованных компонентов проекта	true/false
vuln_scan=	Проверяет заимствованные компоненты проекта на предмет наличия в них известных уязвимостей	true/false
deps_pull=	Ищет зависимости проекта через системы сборки и менеджеры пакетов	true/false
jar_unpack=	Извлекает содержимое Java-архивов (.jar/, .war/, .ear) внутри архива проекта	true/false
pkg_unpack=	Извлекает содержимое пакетов (.rpm/, .apk/, .nupkg/, .whl/, .deb) внутри архива проекта	true/false
container_scan=	Анализирует установленные пакеты в контейнерах без учета транзитивности	true/false
sbom_archive=	Анализ архива с несколькими SBOM JSON (каждый файл анализируется отдельно)	true/false
repo_lookup=	Поиск ссылок на репозитории и архивы с исходными кодами заимствованных компонентов проекта	true/false
secret_scan=	Ищет ключи, токены и пароли в файлах проекта	true/false

Примечание. project_id и project_name одновременно не задаются!

Параметры конфигурации:

- По умолчанию sorca.conf ищется рядом с бинарным файлом, затем в /srv/sorca.
- Если файл не найден, агент предложит создать его интерактивно.
- Если проект не задан, сервер создаст проект вида console-<индекс>.

Короткие опции (флаги) представлены в таблице 4. Команды для использования консольного агента представлены в таблице 5.

Таблица 4 – Опции

Опция	Описание
-c, --conf, --config <path>	Путь к sorca.conf
-u, --url, --server <url>	Адрес сервера SORCA
-k, --key, --api-key <key>	API-ключ пользователя
-p, --project <name>	Имя проекта
-i, --pid, --project-id <uuid>	Идентификатор проекта
-a, --aid, --analysis-id <uuid>	Использовать конкретный анализ
-t, --type <kind>	Тип отчёта: vulns|secrets
-o, --out, --output <path>	Путь для сохранения файла отчёта
-s, --severity <level>	Фильтр уровня: critical|high|medium|low|unknown
-n, --limit <count>	Ограничить число строк вывода
-w, --watch	Следить за прогрессом после запуска
-q, --submit-only	Только отправить анализ и вернуть task_id
-D, --param <key>=<bool>	Переопределить параметр анализа

Таблица 5 – Команды

Команда	Описание	Доступные опции
sorca <архив>	Запустить анализ по умолчанию и следить за прогрессом	
sorca run <архив> [опции]	То же, что и запуск по умолчанию	Принимают архив как позиционный аргумент и опции -c/-u/-k/-p/-i/-w/-q/-D
sorca submit <архив> [опции]	Только отправить анализ и вернуть task_id	
sorca status <task_id> [опции]	Получить текущий статус задачи	Принимают task_id как позиционный аргумент и опции -c/-u/-k
sorca watch <task_id> [опции]	Следить за статусом существующей задачи	
sorca projects [опции]	Показать доступные проекты	Показывает корневые проекты, доступны опции -c/-u/-k
sorca subprojects [опции]	Показать подпроекты проекта	Требует -p <имя> или -i <uuid>, доступны опции -c/-u/-k
sorca summary [опции]	Показать сводку последнего анализа проекта	Показывает сводку по -a <analysis_id> либо по последнему анализу проекта из -p/-i
sorca vulns [опции]	Показать уязвимости последнего анализа проекта	Показывает уязвимости по -a <analysis_id> либо по последнему анализу проекта из -p/-i. Дополнительно поддерживает -s <severity> и -n <limit>
sorca report [опции]	Скачать HTML-отчёт	Скачивает HTML-отчёт по -a <analysis_id> либо по последнему анализу проекта из -p/-i. Требует -t vulns|secrets.

Команда	Описание	Доступные опции
		-o может указывать директорию или полный путь к файлу. Если задана директория, имя файла выбирается автоматически
sorca init [опции]	Создать или обновить sorca.conf	Создаёт или обновляет sorca.conf, доступна опция -c <path>
sorca -h | --help	Показать справку	