Solar appScreener

Руководство пользователя

Solar appScreener

Версия 3.13.1

Июнь 2023

СОДЕРЖАНИЕ

1. ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Термин	Расшифровка
АРМ	Автоматизированное рабочее место
ОС	Операционная система
ПО	Программное обеспечение
CLI	Command Line Interface – интерфейс командной строки
CLT	Command Line Tool – инструмент командной строки
REST	Representational State Transfer – передача состояния представления
SDLC	System Development Life Cycle – жизненный цикл разработки системы
VCS	Version Control System – система управления версиями
WAF	Web Application Firewall – межсетевой экран для защиты веб-приложений

2. ГЛОССАРИЙ

API (Программный интерфейс приложения, Application Programming Interface) — интерфейс, который определяет взаимодействие программы с другой программой.

CI/CD система — система, которая объединяет практики непрерывной интеграции, непрерывной доставки и непрерывного развёртывания. CI/CD системы могут быть встроены в процесс разработки по методике SDLC и SSDLC. Примеры: **TeamCity**, **Jenkins**.

Continuous Delivery (Непрерывная доставка, CD) — практика разработки программного обеспечения, при которой команды производят программное обеспечение в короткие циклы.

Continuous Deployment (Непрерывное развёртывание, CD) — практика разработки программного обеспечения, которая заключается в использовании автоматизированного тестирования для проверки правильности и стабильности изменений в коде для быстрого, автономного развёртывания в производственной среде.

Continuous Integration (Непрерывная интеграция, CI) — практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки и выполнении автоматизированных сборок проекта для выявления потенциальных ошибок и решения интеграционных проблем.

ID проекта (Project ID)— первые 6 символов UUID проекта. ID проекта может быть использован при поиске проекта в общем списке проектов, но UUID является полным идентификатором проекта. Пример ID проекта: d4d1e2.

Software Development Lifecycle (SDLC) — методика разработки, которая обеспечивает качество и правильность работы программного обеспечения. Методика SDLC состоит из таких этапов: анализ требований, дизайн системы, разработка, тестирование, эксплуатация, поддержка.

Secure Software Development Lifecycle (Secure SDLC, SSDLC, DevSecOps) — методика разработки программного обеспечения, которая используется организациями для создания безопасных приложений. При SSDLC на каждом этапе SDLC выполняется ряд дополнительных действий по обеспечению безопасности. Например, анализ рисков на этапе анализа требований, оценка рисков на этапе архитектуры, применение статического анализа кода на этапе разработки, проверка выполнения требований безопасности на этапе тестирования, мониторинг угроз и реагирование на инциденты на этапах эксплуатации и поддержки.

UUID (Universally unique identifier) — 128-битное число, которое используется для идентификации информации. Пример: d4d1e2da-6b82-4350-829b-d3883592f4c8.

Version Control System (VCS, Система контроля версий) — программный инструмент, который служит для записи изменений в файлы и отслеживания изменений, внесённых в код. Примеры: **Git**, **Subversion**.

VCS хостинг — веб-сервис для хостинга проектов и их совместной разработки. Примеры: **GitLab**, **GitHub**, **Bitbucket**.

Web Application Firewall (WAF) — приложение с набором фильтров, предназначенных

для обнаружения и блокировки сетевых атак на веб-приложение. Примеры: **ModSecurity**, **Imperva SecureSphere**, **F5**. В интерфейсе Solar appScreener приводятся инструкции по настройке этих систем для обнаруженных уязвимостей.

Webhook — механизм оповещения пользователей веб-сервиса о событиях в нём.

XPath — язык запросов к элементам XML-документов.

Безопасность приложения (Application security) — набор мер, принимаемых для повышения безопасности приложения, часто путём обнаружения, исправления и предотвращения уязвимостей безопасности.

Интеграция (Integration) — обмен данными между системами с возможной последующей обработкой.

Интегрированная среда разработки (Integrated Development Environment, IDE) — приложение, которое предоставляет возможности для разработки программного обеспечения. IDE обычно состоит из редактора исходного кода, средств автоматизации сборки и отладчика. Примеры: **Eclipse**, **IntelliJ IDEA**.

Интерфейс командной строки (Терминал, Консоль, Command-line interface, CLI) — интерфейс, который обрабатывает команды для программы в виде строк текста.

Исполняемый файл (Executable file) — скомпилированный файл. Примеры расширений исполняемых файлов: `.exe`, `.com`, `.bat`, `.bin`, `.dmg`, `.app`.

Конфигурационный файл (Configuration file) — файл с настройками приложения.

Плагин (Расширение) — независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения её функциональных возможностей.

Система отслеживания ошибок (Bug tracking system, Bug tracker) — программа, разработанная для учёта и контроля ошибок, найденных в программном обеспечении, которая позволяет следить за процессом устранения этих ошибок. Примеры: **Jira**, **Redmine**.

Скрипт (Script) — последовательность команд для автоматического выполнения задачи.

Токен авторизации API (Токен, API authorization token) — набор символов, который предназначен для аутентификации пользователей для выполнения действий в системе без использования пользовательского интерфейса.

Трасса (Trace) — набор инструкций программы, выполняемых последовательно. В Solar appScreener трасса для вхождения изображается в виде диаграммы, отображающей путь распространения данных уязвимости, который выявлен с помощью анализа потока данных.

3. ВВЕДЕНИЕ

Настоящий документ представляет собой руководство пользователя программного комплекса Solar appScreener.

4. СВЕДЕНИЯ О SOLAR APPSCREENER

4.1. Назначение комплекса

Solar appScreener предназначен для анализа приложений с целью выявления уязвимостей информационной безопасности и недекларированных возможностей.

С помощью Solar appScreener можно восстанавливать исходный код приложений из рабочего файла с использованием технологии декомпиляции (обратной инженерии).

4.2. Описание возможностей

Solar appScreener предоставляет следующие возможности:

- Анализ приложений для ОС Android и iOS.
- Анализ приложений на языках ABAP, Apex, C#, C/C++, COBOL, Dart, Delphi, GO, Groovy, HTML, Java, JavaScript, Kotlin, LotusScript, Objective-C, Pascal, PHP, PL/SQL, Python, Perl, Ruby, Rust, Scala, Solidity, Swift, T-SQL, TypeScript, VB.NET, VBA, VBScript, Visual Basic, Vyper или 1C.

  Приложения могут загружаться для анализа как в виде файлов исходного или бинарного кода, так и напрямую из репозитория. В случаях, когда исходный код недоступен, можно загружать в Solar appScreener исполняемые файлы приложения. Для мобильных приложений на Android или iOS достаточно скопировать ссылку на приложение из Google Play или App Store соответственно.
- Анализ конфигурационных файлов.
- Мониторинг изменения уровня безопасности приложения.

  В Solar appScreener реализована возможность генерации отчётов для получения информации по результатам анализа проектов. Отчёты можно генерировать в форматах PDF, CSV или DOCX и отправлять по почте. Форматы JSON и SARIF доступны через вызов API. Также результаты анализа можно просматривать и сравнивать непосредственно в веб-интерфейсе Solar appScreener.
- Формирование рекомендаций по настройке WAF (для веб-приложений).

  В Solar appScreener предусмотрен механизм генерации детальных рекомендаций по настройке WAF, блокирующих возможности эксплуатации ряда уязвимостей до момента их устранения.
- Интеграция в процесс разработки.

  С помощью инструмента командной строки Solar appScreener взаимодействует с системами непрерывной интеграции (Continuous Integration), позволяя наладить непрерывный процесс контроля качества исходного кода и снижая временные затраты. Кроме того, Solar appScreener позволяет автоматизировать проверку новых сборок ПО и может быть встроен в процесс безопасной разработки (SDLC).

4.3. Соответствие классификации CWE

Solar appScreener предоставляет функциональность для работы с уязвимостями в соответствии с классификацией CWE.

Common Weakness Enumeration (CWE™) — это перечень распространённых типов уязвимостей программного и аппаратного обеспечения, имеющих последствия для безопасности. Под «уязвимостями» понимаются дефекты, уязвимости и другие ошибки в программной или аппаратной реализации, коде, дизайне или архитектуре, которые делают системы, сети или оборудование потенциально подверженными атакам злоумышленников. CWE List и связанные с ним классификации служат инструментом для идентификации и описания этих уязвимостей.

В Solar appScreener можно:

- просмотреть ссылки на соответствующий уязвимости идентификатор недостатка (CWE item) (версия 4.0) в карточке правила или его метках (подробнее в разделе Правила);
- искать/фильтровать правила по классификации уязвимостей CWE;
- создавать пользовательские правила со ссылками на соответствующие идентификаторы недостатков (CWE items) (подробнее в разделе Пользовательские правила);
- соотносить найденные уязвимости с соответствующими идентификаторами недостатков (CWE items);
- создавать отчёты в форматах PDF, CSV и DOCX в соответствии с классификацией уязвимостей CWE/SANS Top 25 (подробнее в разделе Экспорт отчёта).

4.4. Требования к АРМ пользователя

4.4.1. Требования к аппаратному обеспечению

АРМ пользователя Solar appScreener должно быть оборудовано персональным компьютером с подключением к внутренней сети компании.

4.4.2. Требования к программному обеспечению

В состав программного обеспечения компьютера для АРМ пользователя Solar appScreener должна входить программа-клиент, предоставляющая пользователю возможность навигации и просмотра веб-ресурсов (браузер). Рекомендуемые браузеры (актуальные версии):

- **Mozilla Firefox**;
- **Google Chrome**;
- **Safari**;
- **Internet Explorer**;
- **Microsoft Edge**.

5. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

5.1. Авторизация

Ссылка для входа в веб-интерфейс Solar appScreener (далее UI) предоставляется администратором. При переходе по ссылке пользователь попадает на страницу авторизации.

Чтобы войти в систему, введите логин и пароль и нажмите кнопку **Войти** (рис. 5.1).

Вход в систему может быть осуществлен с помощью логина (в формате <user> или <user@domain>) и пароля учётной записи **LDAP**. Чтобы настроить доступ с помощью **LDAP**, обратитесь к администратору.

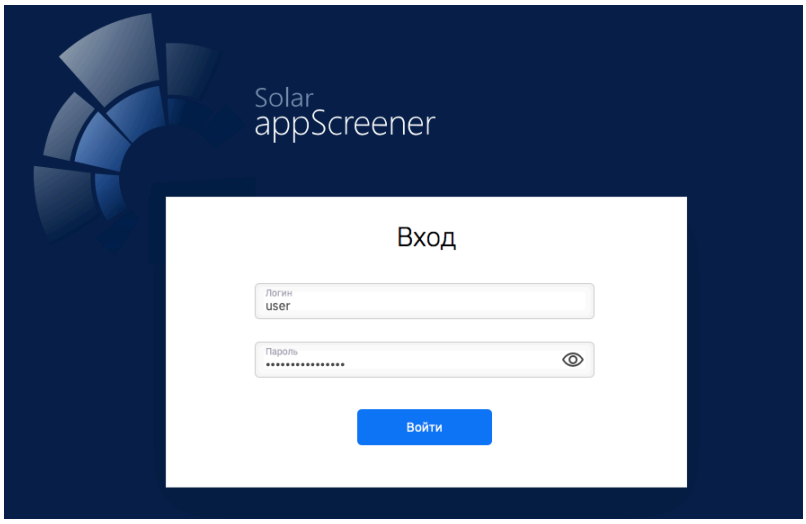


Рис. 5.1: Страница Авторизации

При введении неверных учётных данных на экране отобразится сообщение **Неверный логин и/или пароль.** При превышении числа попыток аутентификации с неверным паролем ваш аккаунт будет временно заблокирован. Количество попыток аутентификации и продолжительность блокировки устанавливается администратором системы (по умолчанию лимит попыток входа — 5, срок блокировки — 5 часов).

Прежде чем начать работу с Solar appScreener, ознакомьтесь с Пользовательским соглашением и нажмите **Принимаю**(рис. 5.2).

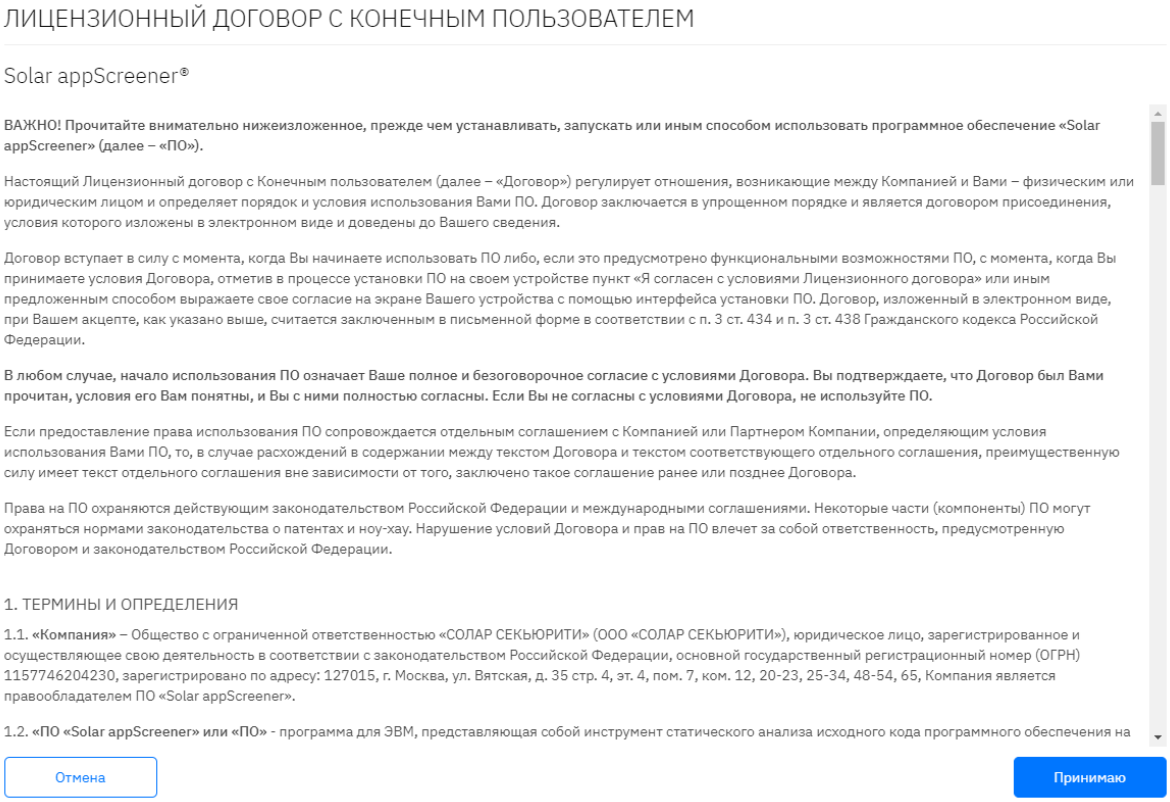


Рис. 5.2: Пользовательское соглашение

После успешного входа в систему отображается краткая инструкция. После просмотра инструкции отображается **Домашняя страница**. Повторный просмотр краткой инструкции доступен на странице **О продукте**.

5.2. Главное меню

В верхней части страницы расположено главное меню, которое предоставляет доступ к разделам **Домашняя страница**, **Проекты**, **Правила и наборы**, **Аналитика**, **О продукте**, **Личный кабинет**.

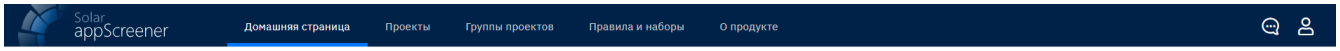


Рис. 5.3: Главное меню

Для доступа к форме обратной связи нажмите  .

5.2.1. Домашняя страница

Домашняя страница (рис. 5.4) предназначена для загрузки и сканирования новых проектов. **Проектом** здесь и далее называется загрузка в Solar appScreener приложения и его сканирование с целью выявления уязвимостей. Под **сканированием** следует понимать анализ исходного или бинарного кода приложения и выявление фрагментов кода, содержащих уязвимости. **Уязвимости** — недостатки в коде приложения, которые могут быть использованы злоумышленниками и вызывать нарушение корректной работы

приложения. Подробное описание запуска проекта представлено в разделе Создание проекта.

На **Домашней странице** также можно:

- ознакомиться со списком проектов — отображаются последние 4 проекта с последующей ссылкой на страницу **Проекты** с полным списком проектов;
- увидеть статистику по количеству сканирований с учётом статусов. Нажмите на статус, чтобы посмотреть список соответствующих проектов.

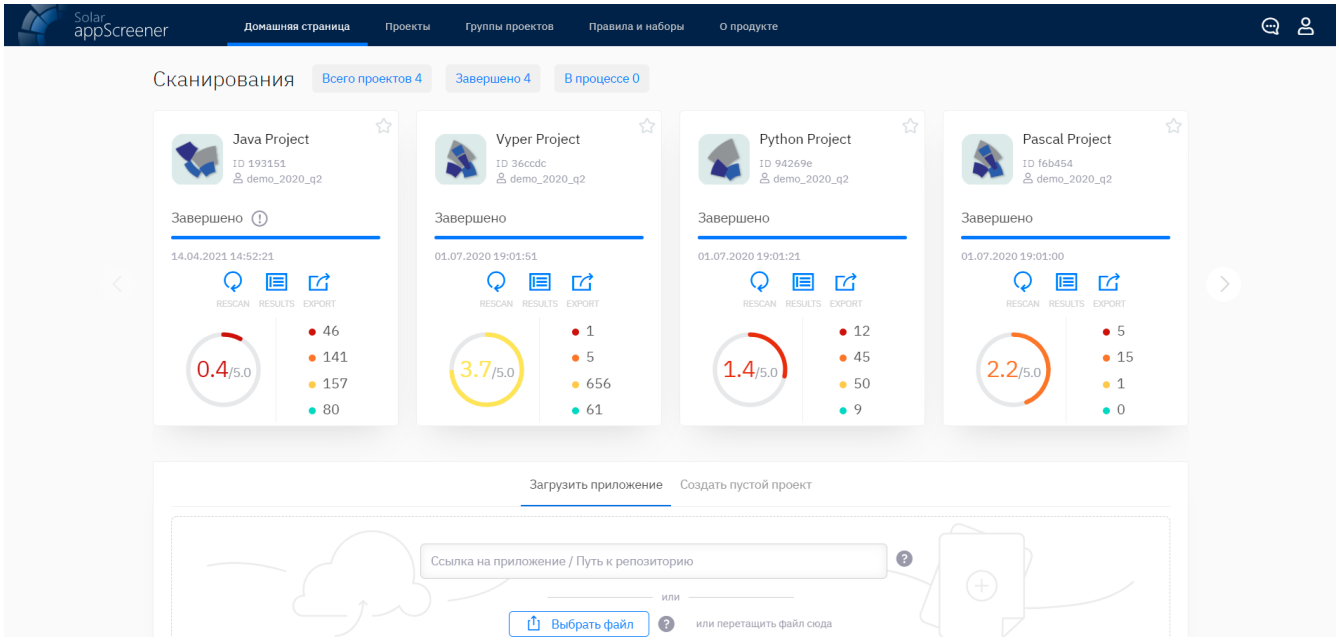


Рис. 5.4: Домашняя страница

5.2.2. Проекты

Страница **Проекты** (рис. 5.5) предназначена для управления проектами. Все проекты представлены в виде списка с краткими характеристиками.

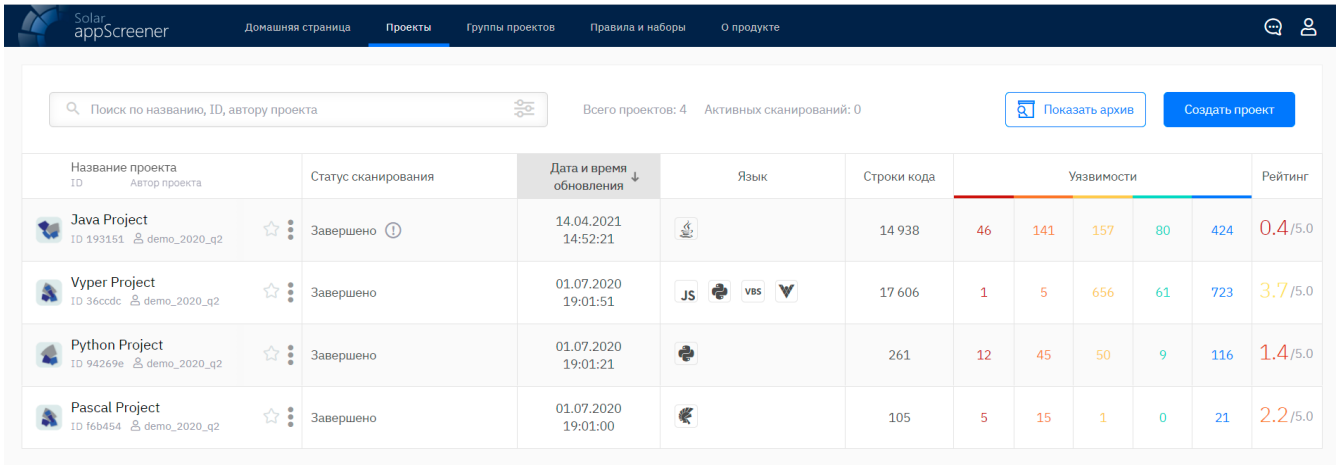


Рис. 5.5: Проекты

Для каждого проекта отображаются следующие данные:

- логотип, название проекта, автор (пользователь, загрузивший проект), ID проекта (первые шесть символов UUID проекта);

Рис. 5.6: Проекты: название

- статус последнего сканирования;

Рис. 5.7: Проекты: статус

- меню действий:
  - копировать UUID проекта;
  - посмотреть подробные результаты последнего сканирования;
  - запустить сканирование;
  - выгрузить отчёт;
  - настроить проект;
  - архивировать проект.

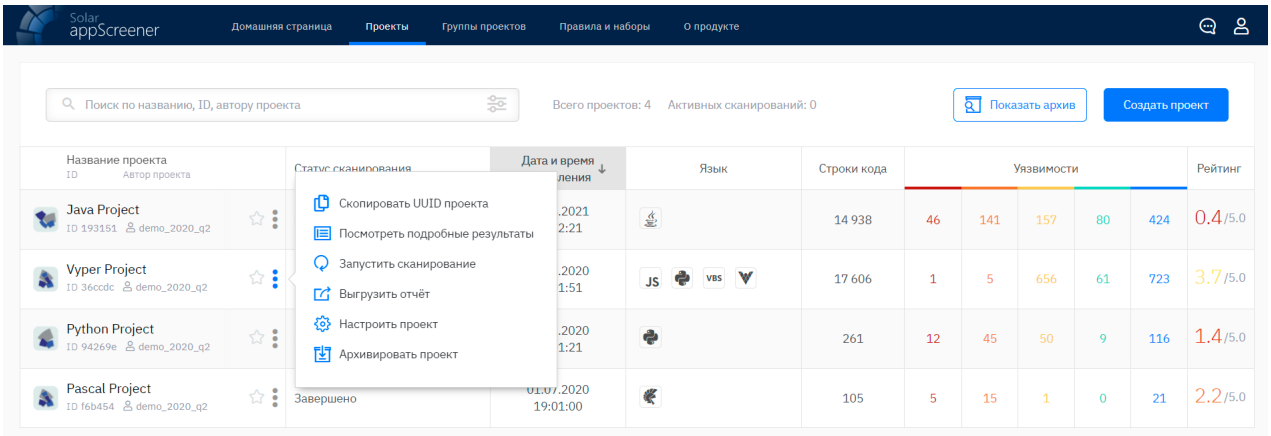


Рис. 5.8: Проекты: действия

- кнопка добавления в Избранное

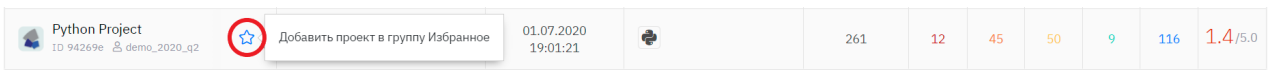


Рис. 5.9: Проекты: добавить в Избранное

- дата и время последнего сканирования;

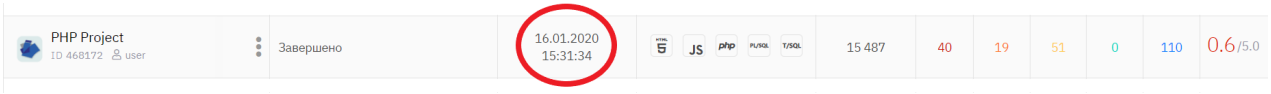


Рис. 5.10: Проекты: дата и время

- языки, для которых был произведен анализ;

Рис. 5.11: Проекты: тип приложения

- количество строк кода в проекте;

Рис. 5.12: Проекты: строки кода

- количество уязвимостей критического, среднего, низкого и информационного уровней, а также общее количество уязвимостей;

Рис. 5.13: Проекты: количество уязвимостей

- рейтинг приложения.

Рис. 5.14: Проекты: рейтинг

В appScreener уязвимости поделены на четыре категории: критические, среднего уровня, низкого уровня и информационного уровня.

1. Критические уязвимости с большой вероятностью приводят к компрометации конфиденциальных данных и нарушению целостности системы.
2. Уязвимости среднего уровня могут с меньшей вероятностью привести к компрометации конфиденциальных данных и нарушению целостности системы либо являются менее серьёзными нарушениями безопасности.
3. Уязвимости низкого уровня могут стать потенциальной угрозой безопасности.
4. Уязвимости информационного уровня сигнализируют о нарушении хороших практик программирования.

Рейтинг приложения вычисляется исходя из количества критических уязвимостей и уязвимостей среднего уровня. Влияние критических уязвимостей больше, чем влияние уязвимостей среднего уровня, и не учитывает объем кода. Уязвимости среднего уровня учитываются из расчёта их количества на общее число строк исходного кода.

Список можно отсортировать по названию, статусу последнего сканирования, по дате и по рейтингу. Для этого нажмите на соответствующий заголовок, повторное нажатие меняет порядок сортировки (рис. 5.15).

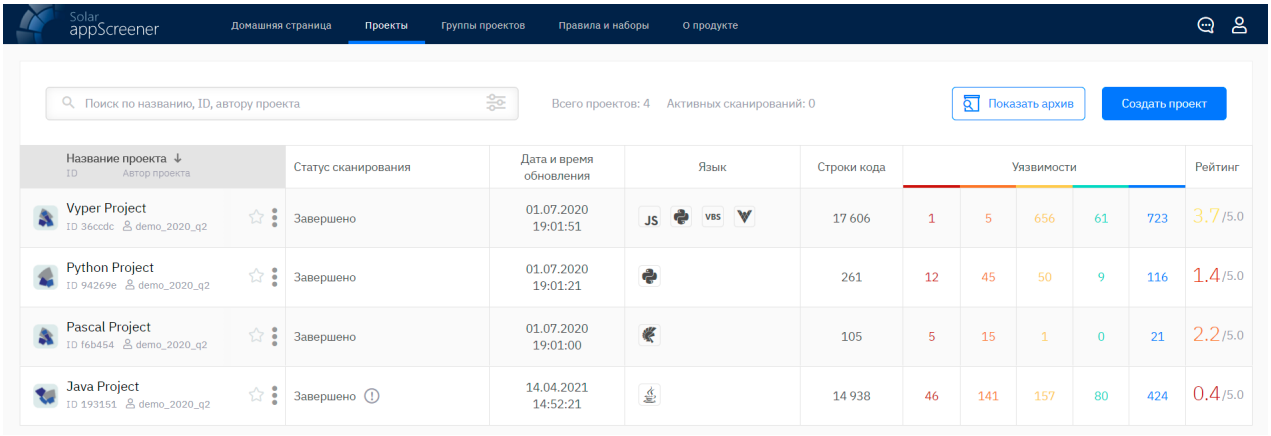


Рис. 5.15: Сортировка по названию

Для скрытия ненужных в данный момент проектов существует возможность архивации. Архивированный проект сохраняется в системе, но становится недоступным для работы. Выберите **Архивировать проект** в меню действий, чтобы добавить проект в архив. Проект, находящийся в архиве, можно найти, нажав **Показать архив** на странице **Проекты**.

Для удобной навигации по проектам предусмотрены поиск и фильтры (рис. 5.16). Поиск позволяет искать проекты по названию, ID проекта или автору. Чтобы установить фильтры, нажмите на иконку фильтров и настройте один или несколько параметров:

- статус сканирования — выбрать из списка статусы сканирования;
- дата обновления — задать временной диапазон;
- языки — выбрать один или несколько языков программирования;
- рейтинг — задать диапазон для рейтинга последнего сканирования в проекте;
- количество уязвимостей каждого из уровней критичности — задать диапазон для количества уязвимостей критического, среднего, низкого или информационного уровня;
- наличие в группе проектов.

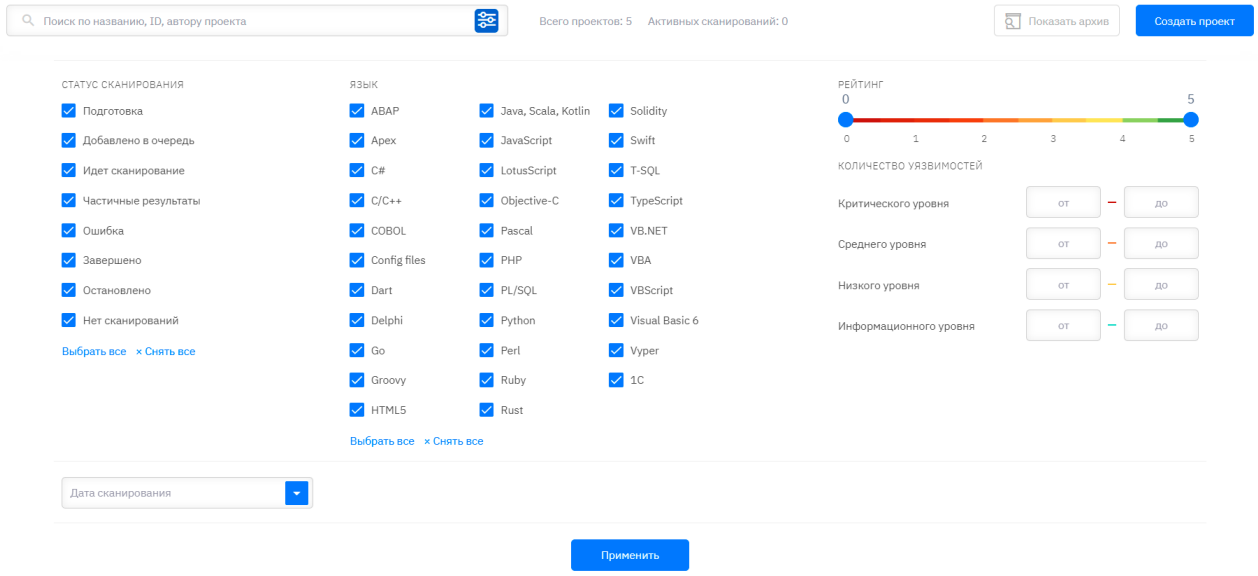


Рис. 5.16: Фильтры проектов

Чтобы установить фильтры, нажмите на кнопку **Применить**. После применения фильтров в правой части страницы появится количество отобранных проектов и кнопка **Сбросить**. Нажатие на кнопку отменит фильтрацию.

Чтобы перейти на страницу конкретного проекта, нажмите на его название в списке. Подробнее про управление конкретным проектом в разделе Управление проектом.

5.2.2.1. Очередь сканирований

На вкладке **Очередь сканирований** можно управлять очередью сканирований в системе, поднимать/опускать приоритет сканирований. Система поддерживает 4 уровня приоритета сканирований: Низкий, Средний, Высокий и Эксклюзивный. По умолчанию сканирования запускаются со **Средним** приоритетом.

Искать сканирование в очереди можно по ID, названию проекта и автору сканирования.

В основной части страницы находится список всех активных сканирований в системе. Он представлен в виде таблицы со столбцами:

- Название проекта — кликабельно, по клику происходит переход на страницу **Обзор** (при наличии доступа в проект у пользователя);
- Сканирование — первые 6 символов UUID сканирования (по кнопке может быть скопирован целиком) и автор сканирования;
- Дата создания;
- Статус;
- Приоритет.

Все столбцы поддерживают сортировку, по умолчанию отсортированы по приоритету. Сканирования с одинаковым приоритетом сортируются по дате создания: первым отображается и будет просканирован проект, запущенный *раньше*.

Обратите внимание: изменение приоритета сканирования затрагивает только проекты в очереди и не повлияет на прогресс сканирований, которые уже выполняются.

5.2.3. Группы проектов

Проекты в Solar appScreener можно объединять в группы. Используя группы, можно выполнять действия с несколькими логически связанными проектами одновременно. Также для групп доступна сводная информация и аналитика. Работать с группами можно в разделе **Группы проектов** (рис. 5.17).

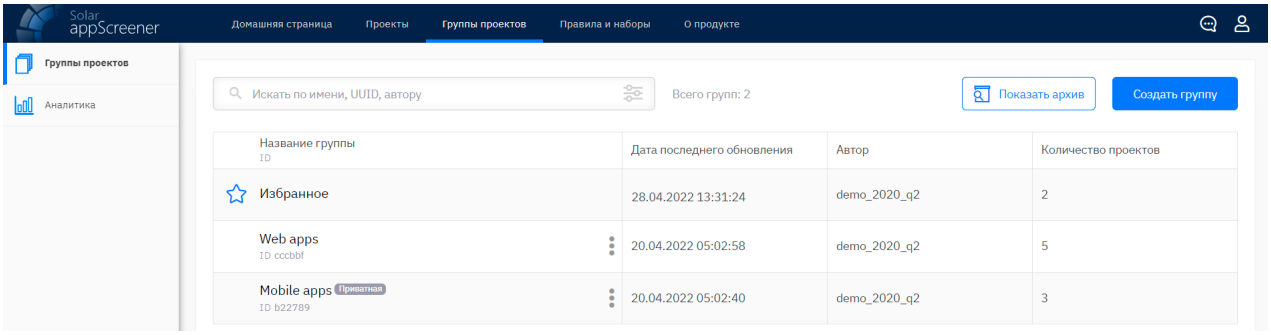


Рис. 5.17: Группы проектов

Список групп отображается в основной части страницы. Для каждой группы отображаются следующие данные:

- название группы;
- ID группы;
- видимость;
- меню действий:
  - копировать UUID группы;
  - проекты — перейти к списку проектов;
  - настроить группу;
  - архивировать группу.
- дата и время обновления;
- автор;
- количество проектов в группе.

Список можно отсортировать. Для этого нажмите на соответствующий заголовок, повторное нажатие меняет порядок сортировки. Для удобной навигации также предусмотрен поиск и фильтры. Поиск позволяет искать группы проектов по названию, UUID группы или автору. Чтобы установить фильтры, нажмите на иконку фильтров и настройте один или несколько параметров:

- видимость — выбрать публичные или приватные группы;
- количество проектов в группе — можно указать диапазон;
- дата создания;
- дата последнего обновления;
- наличие проектов — добавить проекты, которые должна содержать группа.*

* *Обратите внимание: при выборе нескольких проектов фильтр работает как условие ИЛИ, то есть результаты поиска будут содержать группы проектов с хотя бы одним из указанных проектов.*

Чтобы установить фильтры, нажмите на кнопку **Применить**. После применения фильтров в правой части страницы появится количество отобранных проектов и кнопка **Сбросить**. Нажатие на кнопку отменит фильтрацию.

5.2.3.1. Создание группы проектов

Чтобы создать группу проектов, нажмите **Создать группу**, укажите имя группы и выберите проекты, которые следует в неё включить. Также можно включить проекты из существующих групп. Чтобы группа отображалась у всех пользователей, выберите опцию **Публичная**. После выполнения этих действий нажмите **Сохранить** (рис. 5.18).

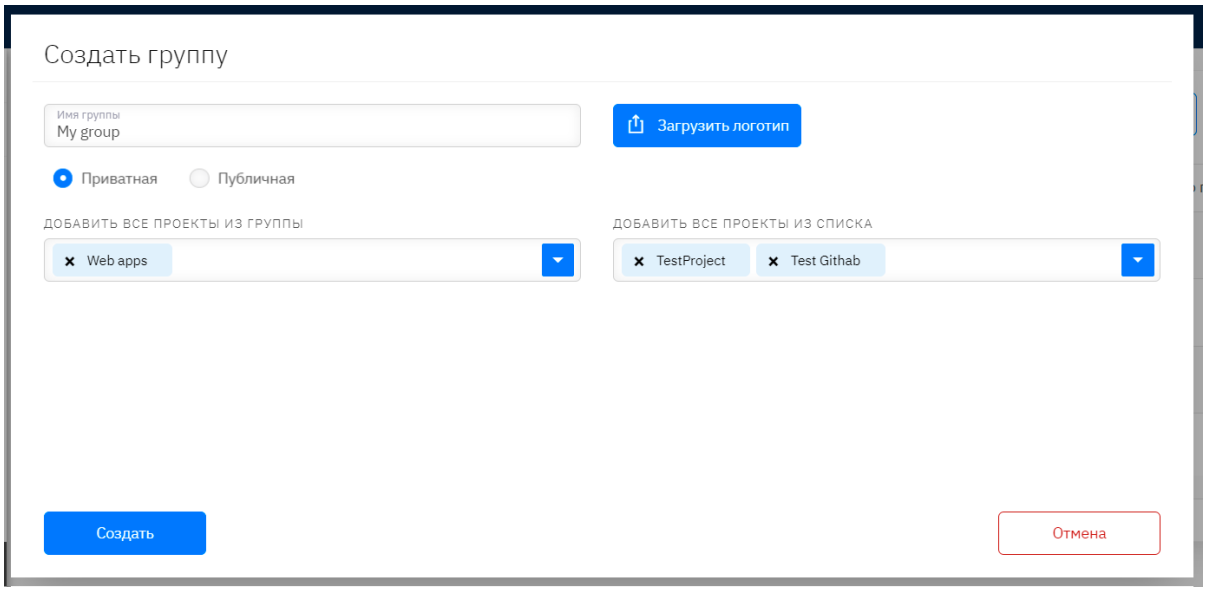


Рис. 5.18: Создание группы проектов

5.2.3.2. Работа с группой проектов

Чтобы перейти к конкретной группе проектов, кликните по её названию в списке групп.

На вкладке **Обзор** представлена общая статистика по сканированиям в группе. Динамику результатов сканирований проектов можно проследить на графиках. В верхней части страницы можно выбрать тип значений (суммарное или среднее) и период для отображения. Сводная информация по сканированиям представлена в таблице **Статистика группы**.

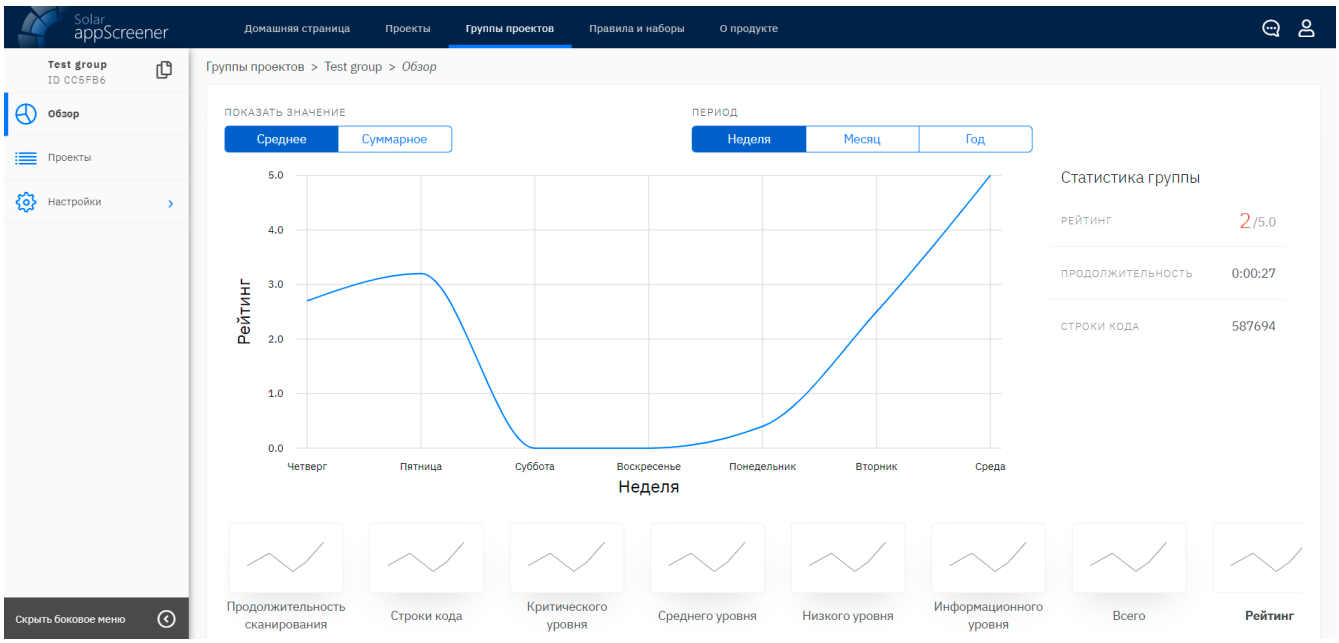


Рис. 5.19: Группа проектов: Обзор

Действия с проектами в группе доступны на вкладке **Проекты**. Здесь можно просмотреть список всех проектов в группе, добавить/удалить проект или поместить/извлечь проект из архива группы. Обратите внимание: при удалении проекта происходит только его удаление из группы, но не из системы.

Управлять группой и правами пользователей в группе можно во вкладке **Настройки**. В подразделе **Управление группой** можно редактировать данные группы, поместить/извлечь группу из архива или удалить группу. В подразделе **Jira** можно привязать группу проектов appScreener к проекту **Jira**.

5.2.3.3. Аналитика

Раздел **Аналитика** предназначен для просмотра общей статистики по всем сканированиям в системе и сравнения результатов анализа по группам.

На вкладке **Результаты** (рис. 5.20) можно проследить динамику результатов сканирований проектов. Добавьте группу к статистике, чтобы сравнить результаты. Для просмотра информации по всем проектам, выберите **Все проекты** в списке групп. Аналогично разделу **Статистика группы** тип значений и период для отображения можно настроить. Для каждой из групп на графиках отображаются данные:

- количество сканирований (при отображении суммарного значения) или рейтинг (при отображении среднего значения);
- продолжительность сканирований;
- количество строк кода;
- количество уязвимостей (с учётом уровня критичности).

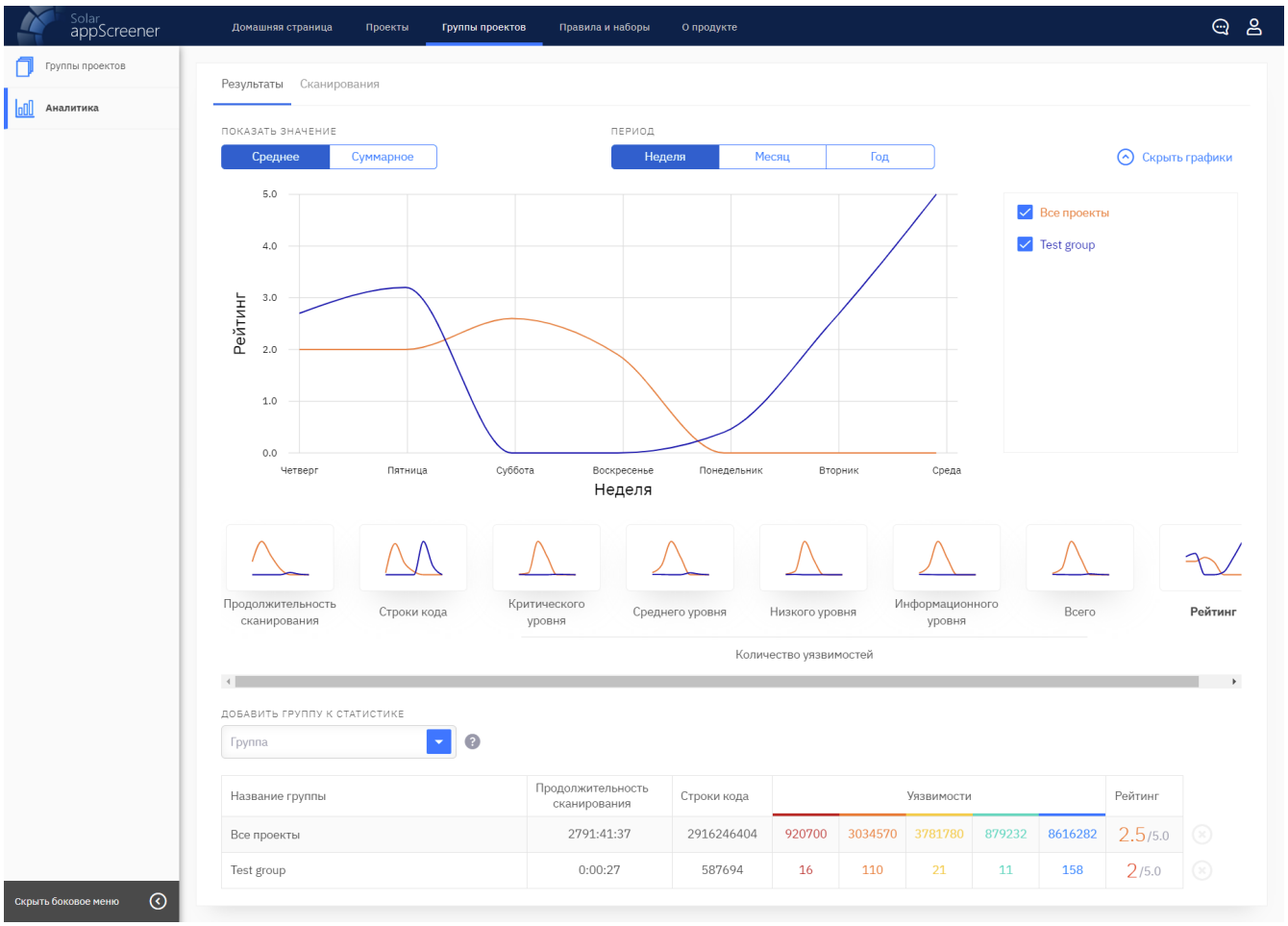


Рис. 5.20: Результаты сканирований по группам проектов

Для просмотра аналитики по сканированиям перейдите на вкладку **Сканирования**. В таблице отображаются данные о количестве проектов в группе, количестве сканирований и их статусах. Чтобы убрать группу из статистики, нажмите на иконку крестика в конце строки нужной группы.

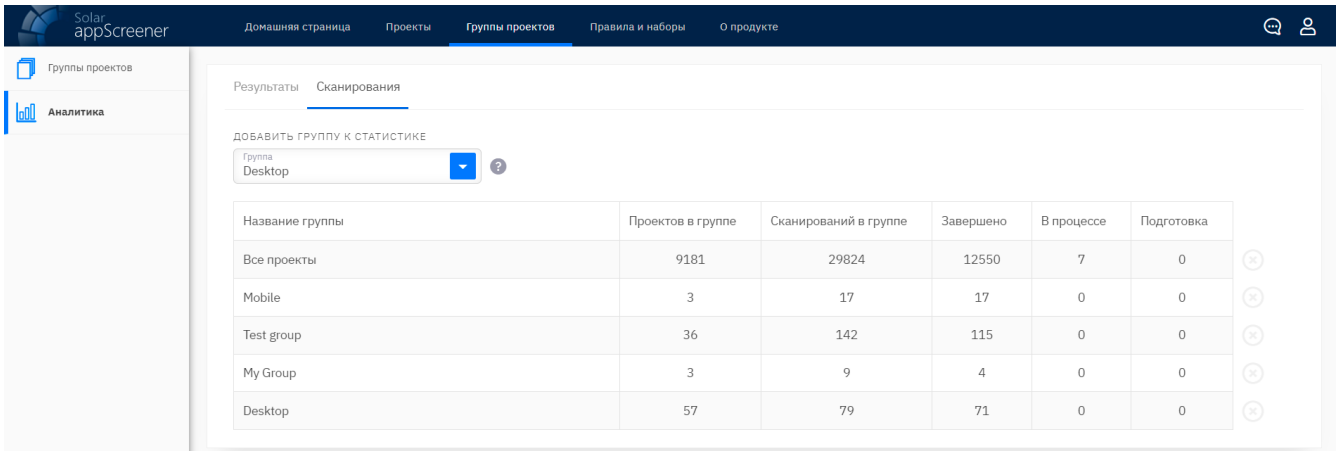


Рис. 5.21: Просмотр статистики по группам проектов

5.2.4. О продукте

Страница **О продукте** (рис. 5.22) служит для предоставления пользователю общей информации о работе с Solar appScreener. В верхней части страницы можно переключаться между следующими разделами:

- Инструкция;
- Общая информация;
- Анализ мобильных приложений;
- Анализ веб-приложений;
- Инструкции по настройке WAF.

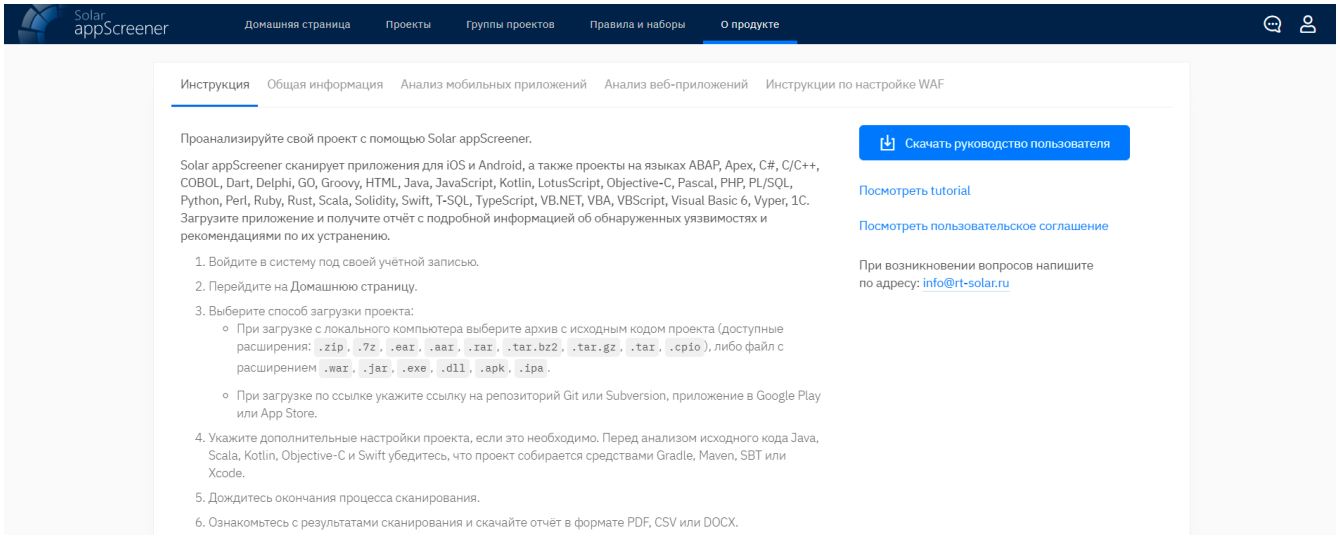


Рис. 5.22: О продукте

В разделе **Инструкция** представлено краткое описание запуска анализа. Из этого раздела можно скачать руководство пользователя и включить/отключить отображение подсказок в интерфейсе.

В разделе **Общая информация** перечисляются основные возможности продукта.

В разделах **Анализ мобильных приложений** и **Анализ веб-приложений** приведены распространенные уязвимости для мобильных и веб-приложений соответственно.

В разделе **Инструкции по настройке WAF** описана возможность генерировать рекомендации по настройке средств защиты периметра.

5.2.5. Личный кабинет

Личный кабинет (рис. 5.23) открывается при наведении курсора на иконку в правом углу верхнего меню. Появляется выпадающее меню с пунктами: **Профиль**, **Настройки**, **Выйти**.

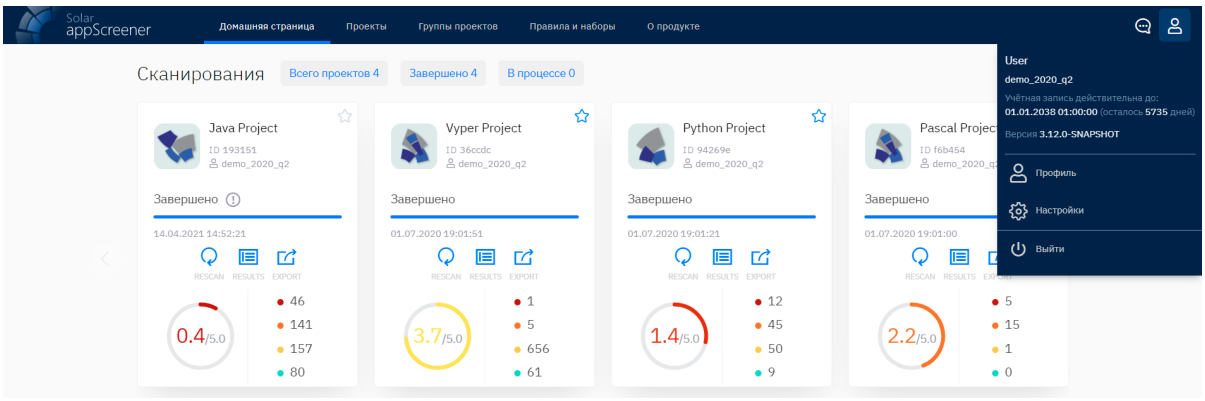


Рис. 5.23: Личный кабинет

5.2.5.1. Профиль

В разделе **Профиль** можно выполнить следующие действия:

- ознакомиться с информацией об учётной записи;
- ознакомиться с информацией об ограничениях лицензии;
- настроить оповещения;
- выбрать язык интерфейса.

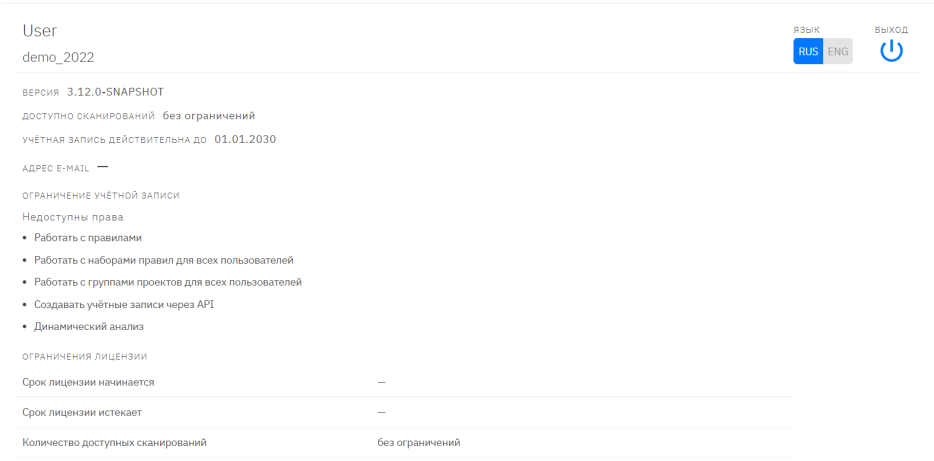


Рис. 5.24: Профиль

Если вы хотите получать почтовые оповещения о завершённых сканированиях, воспользуйтесь переключателем. По желанию к оповещению можно добавить краткую информацию о результатах и статистику по языкам (для проектов SAST) или текст ошибки в случае, если сканирование будет завершено с ошибкой.

5.2.5.2. Настройки доступа

5.2.5.2.1. Токен и пароль

На вкладке **Токен и пароль** можно получить активный токен авторизации и изменить пароль учётной записи.

Токен авторизации предназначен для аутентификации пользователя при выполнении действий в Solar appScreener без использования UI. Например, чтобы запускать

сканирования напрямую через CLT или автоматизировать действия в системе с помощью скриптов. Чтобы получить токен авторизации API:

1. Нажмите **Создать токен**.
2. Введите пароль учётной записи.
3. Укажите время действия токена.
4. Нажмите **Получить активный токен**.

Токен авторизации появится в соответствующем поле. Ознакомиться с информацией о всех активных токенах можно в таблице.

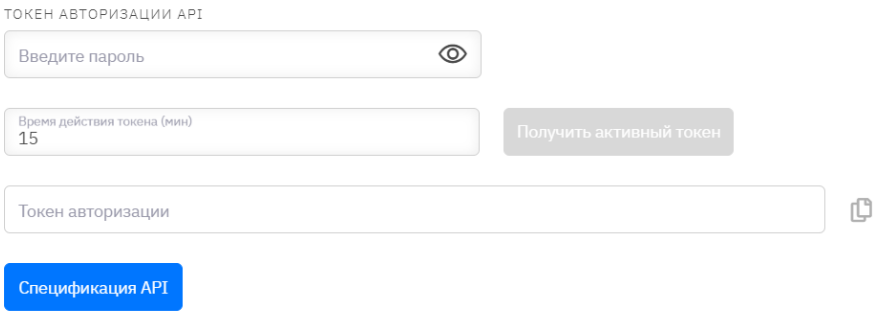


Рис. 5.25: Токен авторизации API

В соответствии с требованиями информационной безопасности **пароль** учётной записи должен регулярно обновляться. Незадолго до истечения срока действия текущего пароля вы получите уведомление.

Для **смены пароля:**

1. Укажите текущий пароль.
2. Укажите новый пароль и повторите его в следующем текстовом поле.
3. Нажмите **Сохранить**.

По истечению срока действия пароля произойдёт автоматический выход из системы на всех устройствах. Для повторного входа требуется установить новый пароль.

5.2.5.2.2. Jira

Для того чтобы привязать аккаунт Jira (рис. 5.26):

1. Введите URL сервера Jira.
2. Введите логин и пароль от аккаунта Jira.
3. Нажмите **Привязать аккаунт**.

В результате этих действий в разделе **Личный кабинет** будет указан привязанный аккаунт Jira. В этом же разделе можно **Отвязать аккаунт** и **Проверить соединение** с Jira.

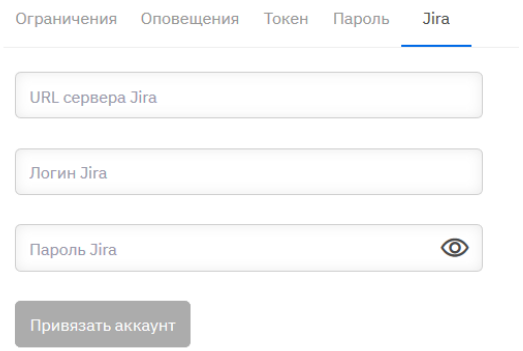


Рис. 5.26: Привязка аккаунта Jira

5.2.5.2.3. Приватный репозиторий

В разделе **Приватный репозиторий** можно работать с учётными данными, необходимыми для анализа файлов из закрытых репозиториев. Сохранённые в разделе учётные записи можно использовать в различных проектах в системе.

Вы можете добавить/редактировать учётные данные 4 типов:

- логин и пароль - укажите имя пользователя и пароль от ресурса, которому требуется аутентификация;
- токен доступа - предоставьте токен, используемый для авторизации на стороннем ресурсе;
- SHH-ключ - предоставьте приватный SSH-ключ (может быть введён вручную или загружен файлом) и при необходимости отредактируйте конфигурацию SSH клиента (доступно только при выключенном переключателе);
- форма авторизации - используются только для анализа DAST; все поля обязательны для заполнения, за исключением регулярных выражений в сообщении о входе/выходе (будет достаточно указать только одно из них).

Добавленные учётные записи отображаются в виде списка на соответствующих вкладках. Чтобы отредактировать данные или настроить доступ к ним других пользователей системы, выберите нужную учётную запись из списка.

Данные учётной записи также можно заполнить перед началом сканирования. Выбор опции **Использовать данные при пересканировании проекта** сохранит данные в зашифрованном виде в настройках проекта для последующих сканирований.

Логин и пароль

Чтобы добавить учётные данные этого типа, задайте название записи и укажите имя пользователя и пароль от необходимого ресурса. Также вы можете настроить доступ др

5.2.5.3. Настройки системы

5.2.5.3.1. Сканирование

В подразделе **Сканирование** можно выполнять действия с шаблонами настроек сканирования. Шаблоны позволяют не настраивать конфигурацию сканирования вручную перед каждым запуском, а в один клик выставлять часто используемые настройки для анализа. На странице можно:

- создать шаблон;
- внести изменения в существующие шаблоны;
- выбрать шаблон по умолчанию.

Вы можете самостоятельно выбирать, изменять и удалять значение шаблона по умолчанию для запуска анализа. Если вы ничего не укажете или шаблон будет удалён, вашим шаблоном по умолчанию будет значение, установленное администратором системы. Если администратор не назначит иной шаблон, для запуска сканирования будет использоваться системный шаблон.

Для удобной навигации по шаблонам предусмотрена возможность поиска по названию или автору шаблона.

Создание шаблона настроек сканирования

Чтобы создать шаблон настроек:

1. Нажмите на кнопку **Создать шаблон**. После этого откроется форма создания шаблона настроек.
2. В форме создания шаблона настроек задайте название шаблона.
3. Отметьте чекбокс **Шаблон настроек по умолчанию**, чтобы использовать выбранный шаблон по умолчанию при запуске нового анализа.
4. При необходимости добавьте описание шаблона настроек.
5. Укажите, будет шаблон публичным или приватным. **Публичный шаблон** будет доступен для использования всем пользователям системы. **Приватный шаблон** будет доступен только автору шаблона и администратору системы.
6. Шаблон хранит в себе информацию о конфигурации настроек пунктов следующих блоков:
   - анализировать языки;
   - настройки Java, Scala, Kotlin;
   - настройки C/C++;
   - настройки Javascript;
   - общие настройки;
   - настройки репозитория Git;
   - настройки приватного репозитория;
   - настройки кодировки;
   - наборы правил.
7. Нажмите **Сохранить**. После успешного сохранения система вернёт вас в подраздел **Сканирование**.

Важно обратить внимание:

При дальнейшем обновлении системы настройки старых шаблонов будут переноситься без изменений. Это означает, например, что при добавлении поддержки нового языка программирования вам нужно выбрать вручную добавленный язык в вашем шаблоне, если вы хотите его анализировать. В противном случае, при запуске сканирования анализ этого языка запущен не будет.

Изменение шаблона настроек сканирования

Чтобы изменить шаблон настроек:

1. Нажмите на название шаблона в списке. После этого откроется форма изменения шаблона настроек.
2. Внесите желаемые изменения.
3. Нажмите **Сохранить**. После успешного сохранения система вернёт вас в подраздел **Сканирование** раздела **Настройки**.

Для удаления шаблона настроек после выполнения шага 2 нажмите кнопку **Удалить**.

Важно обратить внимание:

При редактировании собственного шаблона вы можете изменить любые настройки. При работе с шаблонами других пользователей или системным шаблоном настройки недоступны для редактирования. Вы можете:

- просмотреть шаблон;
- установить выбранный шаблон в качестве шаблона по умолчанию для запуска анализа;
- скопировать настройки шаблона и использовать копию для создания собственного шаблона.

5.2.5.3.2. Экспорт отчёта

В подразделе **Экспорт отчёта** можно выполнять действия с шаблонами настроек экспорта отчёта. Шаблоны позволяют в один клик выставлять часто используемую конфигурацию отчёта. На странице можно:

- создать шаблон;
- внести изменения в существующие шаблоны;
- выбрать шаблон по умолчанию.

Для удобной навигации по шаблонам предусмотрена возможность поиска по названию или автору шаблона.

Создание шаблона экспорта отчёта

Чтобы создать шаблон:

1. Нажмите на кнопку **Создать шаблон**. После этого откроется форма создания шаблона экспорта отчёта.
2. Задайте название шаблона.
3. Отметьте чекбокс **Шаблон экспорта по умолчанию**, чтобы использовать выбранный шаблон по умолчанию при генерации отчёта.
4. При необходимости добавьте описание шаблона настроек.
5. Укажите, будет шаблон публичным или приватным. **Публичный шаблон** будет доступен для использования всем пользователям системы. **Приватный шаблон** будет доступен только автору шаблона и администратору системы.
6. Настройте видимость шаблона в списке на странице **Экспорт отчёта** проекта.
7. Шаблон хранит в себе информацию о конфигурации настроек пунктов отчёта. Подробнее о настройках экспорта см. Экспорт отчёта.
8. Нажмите **Сохранить**. После успешного сохранения система вернёт вас в подраздел **Экспорт отчёта**.

Изменение шаблона экспорта отчёта

Чтобы изменить шаблон экспорта:

1. Нажмите на название шаблона в списке. После этого откроется форма изменения шаблона.
2. Внесите желаемые изменения.
3. Нажмите **Сохранить**. После успешного сохранения система вернёт вас в подраздел **Экспорт отчёта** раздела **Настройки**.

Для удаления шаблона экспорта после выполнения шага 2 нажмите кнопку **Удалить**.

6. ОПИСАНИЕ РАБОТЫ С SOLAR APPSCREENER

6.1. Создание проекта

В интерфейсе Solar appScreener реализованы следующие способы создания проекта:

- запуск сканирования приложения, загруженного с локального компьютера;
- запуск сканирования приложения, загруженного по ссылке;
- создание пустого проекта, у которого нет сканирований.

6.1.1. Создание пустого проекта

На **Домашней странице** можно создать пустой проект. Данная возможность позволяет создать проект и произвести его настройку перед запуском сканирования.

Чтобы создать пустой проект, введите название и нажмите **Создать проект**. При необходимости нажмите **Показать настройки** и установите настройки анализа. Подробнее про настройки анализа в разделе Общие.

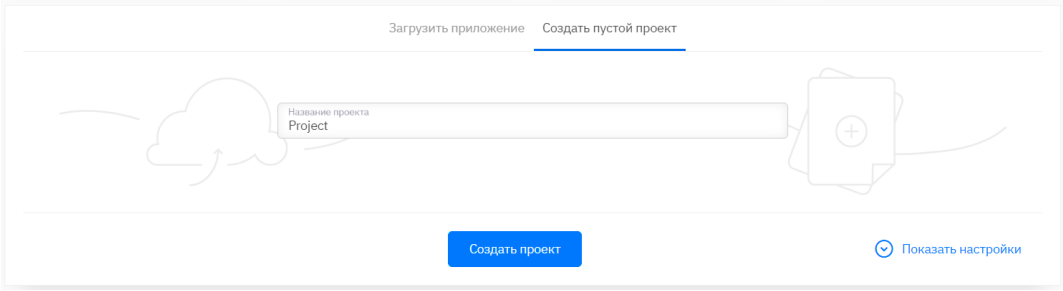


Рис. 6.1: Создание пустого проекта

В созданном проекте можно настроить интеграции. Подробнее про интеграции Solar appScreener в разделе Автоматическое сканирование.

6.1.2. Запуск сканирования в UI

Чтобы запустить новое сканирование в UI Solar appScreener:

1. Перейдите на **Домашнюю страницу**.
2. Загрузите проект.
3. Настройте анализ вручную или воспользуйтесь одним из готовых шаблонов запуска анализа (подробнее о **Настройках** в разделе Настройки и о **Шаблонах** в разделе Настройки).
4. Нажмите **Начать сканирование**.

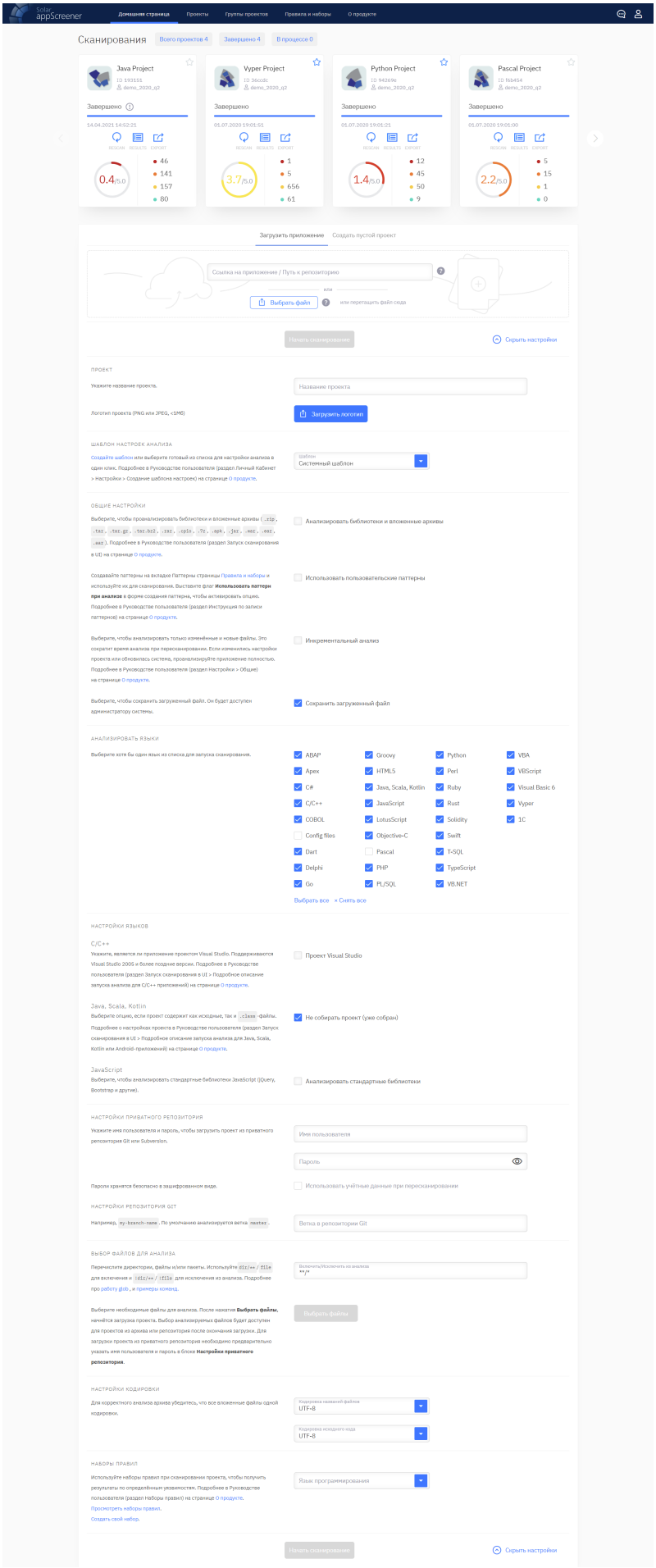


Рис. 6.2: Новое сканирование

Подробное описание запуска анализа в UI Solar appScreener с учётом особенностей языка проекта:

1. **Java, Scala, Kotlin или Android-приложение**
2. **iOS-приложение**
3. **C/C++ приложение**
4. **macOS-приложение**
5. **1C-приложение**
6. **Остальные приложения**

6.1.2.1. Подробное описание запуска анализа для Java, Scala, Kotlin или Android-приложений

В Solar appScreener реализованы следующие способы загрузки Java, Scala, Kotlin или Android-проектов для анализа.

- **Загрузить приложение с локального компьютера**. Загрузите архив с исходным кодом и/или байт-кодом приложения (поддерживаются следующие форматы архивов: ZIP, 7z, RAR (до версии 4.0), EAR, AAR, tar.bz2, tar.gz, tar, cpio), либо исполняемый файл приложения (`.war`-, `.jar`-, `.apk`- или `.aab`-файл).
- **Загрузить приложение по ссылке**. Есть два возможных способа загрузки приложения по ссылке:
  - Для Android приложений есть возможность указать ссылку на Google Play следующего вида: https://play.google.com/store/apps/details?id=package.
  - Указывается ссылка на репозиторий с исходным кодом проекта (поддерживается **Git** и **Subversion**). Машина, на которой установлен Solar appScreener, должна иметь доступ к указанному репозиторию. С помощью указанной ссылки должно быть возможно скачивание кода из репозитория с использованием команды `git clone` (для **Git**) или `svn export` (для **Subversion**). Примеры ссылок:
  - https://gitlab.example.com/myproject.git (**Git**)
  - ssh://gitlab.example.com/myproject (**Git**)
  - https://svn.example.com/mysvnproject/trunk/ (**Subversion**)
  - svn://svn.example.com/mysvnproject/branches/my-branch (**Subversion**)
  - svn+ssh://svn.example.com/mysvnproject/ (**Subversion**)

Чтобы проанализировать код из приватного репозитория, укажите учётные данные. Подробнее про настройки анализа в разделе Настройки.

Со списком расширений файлов, которые анализируются при загрузке архива или по ссылке на репозиторий, можно ознакомиться в приложении (табл. 8.1). Чтобы проанализировать вложенный архив, установите дополнительную настройку **Анализировать библиотеки и вложенные архивы**.

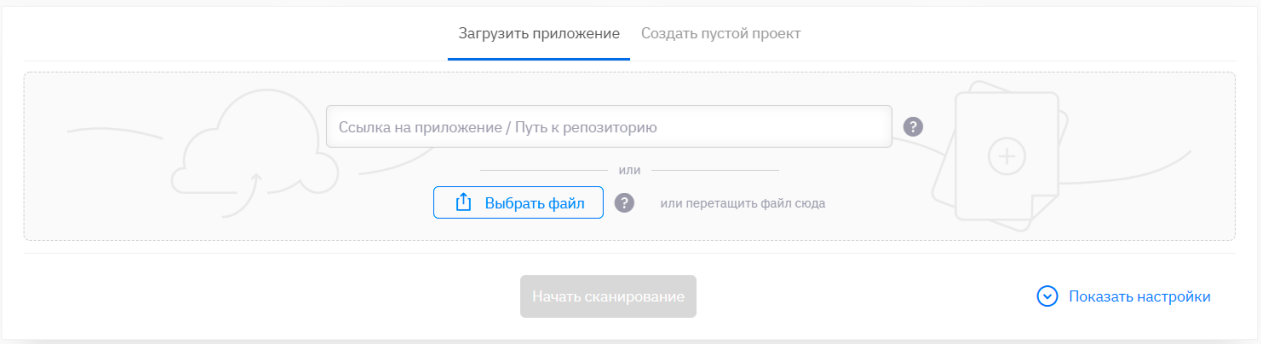


Рис. 6.3: Загрузка приложения

В процессе анализа Solar appScreener рассматривает промежуточное представление кода (его модель), а не сам текст исходного кода или инструкции исполняемого кода. Для приложений, написанных на языках, компилирующихся в байт-код Java (языки Java, Scala, Kotlin, мобильные приложения для платформы Android), в качестве модели используется представление, близкое к самому байт-коду, поэтому для анализа он необходим. Результаты анализа в любом случае отображаются на исходный код либо переданный пользователем, либо полученный путем декомпиляции. Отображение происходит с точностью до строки исходного кода, где была обнаружена уязвимость.

- При анализе архива помимо исходного кода рекомендуется помещать в архив `.class`-файлы проекта (байт-код), соответствующие переданному исходному коду. Это соответствует архивации директории проекта после успешного завершения сборки. В таком случае при запуске анализа можно установить дополнительную настройку **Не собирать проект (уже собран)**.
- При анализе исходного кода в виде архива или по ссылке на репозиторий, если не устанавливать дополнительную настройку **Не собирать проект (уже собран)**, Solar appScreener проведет автоматическую сборку проекта с помощью средств **Maven**, **Gradle** или **SBT**. В таком случае проект должен собираться без ошибок на машине, где установлен анализатор Solar appScreener.
- При анализе Java, Scala, Kotlin или Android-приложений можно установить дополнительную настройку **Анализировать библиотеки и вложенные архивы**. В таком случае будут проанализированы рекурсивно вложенные `.jar`-файлы, а также `.class`-файлы, для которых в архиве не было обнаружено исходного кода (при анализе архива или по ссылке на репозиторий).
- Для анализа JSP-страниц структура их расположения должна соответствовать структуре `.war`-файла.

Подробнее дополнительные настройки анализа описаны в разделе Общие.

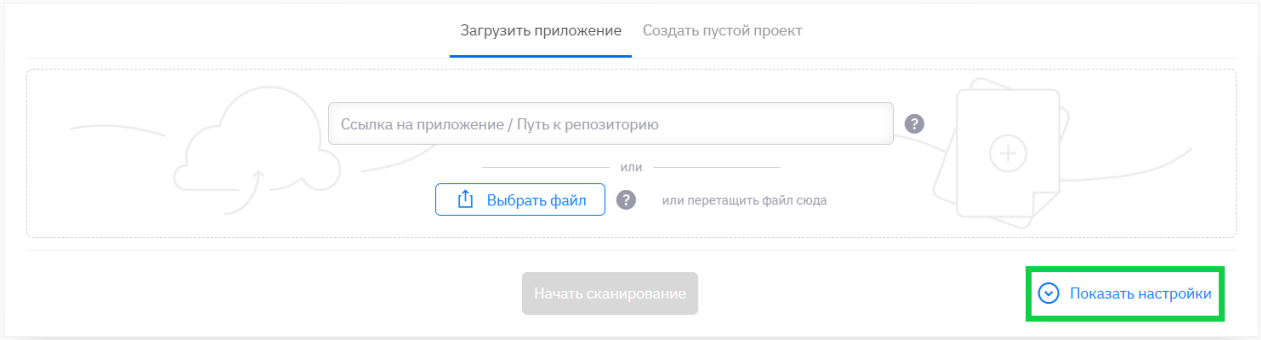


Рис. 6.4: Показать настройки

6.1.2.2. Подробное описание запуска анализа для iOS-приложений

В Solar appScreener реализованы следующие способы загрузки iOS-приложений для анализа.

- **Загрузить приложение с локального компьютера**. Загрузите архив с исходным кодом (поддерживаются следующие форматы архивов: ZIP, 7z, RAR (до версии 4.0), EAR, AAR, tar.bz2, tar.gz, tar, cpio) или исполняемый файл приложения (`.ipa`-файл, который должен содержать `armv8` (`aarch64`) бинарный файл приложения).
- **Загрузить приложение по ссылке**. Есть два возможных способа загрузки приложения по ссылке:
  - Указывается ссылка на App Store следующего вида:

    https://apps.apple.com/us/app/name/id23423432432?mt=8.
  - Указывается ссылка на репозиторий с исходным кодом проекта (поддерживается **Git** и **Subversion**). Машина, на которой установлен Solar appScreener, должна иметь доступ к указанному репозиторию. С помощью указанной ссылки должно быть возможно скачивание кода из репозитория с использованием команды `git clone` (для **Git**) или `svn export` (для **Subversion**). Примеры ссылок:
    - https://gitlab.example.com/myproject.git (**Git**)
    - ssh://gitlab.example.com/myproject (**Git**)
    - https://svn.example.com/mysvnproject/trunk/ (**Subversion**)
    - svn://svn.example.com/mysvnproject/branches/my-branch (**Subversion**)
    - svn+ssh://svn.example.com/mysvnproject/ (**Subversion**)

    Чтобы проанализировать код из приватного репозитория, укажите учётные данные. Подробнее про настройки анализа в разделе Настройки.

Со списком расширений файлов, которые анализируются при загрузке архива или по ссылке на репозиторий, можно ознакомиться в приложении (табл. 8.1). Чтобы проанализировать вложенный архив, установите дополнительную настройку **Анализировать библиотеки и вложенные архивы**.

Чтобы получить `.ipa`-файл, который поддерживается системой:

1. Соберите приложение в **Xcode** и получите директорию с расширением `.app`;
2. Упакуйте её в `.zip`-архив;

3. Переименуйте расширение `.zip` в `.ipa`.

Чтобы провести анализ исходного кода (либо в виде архива, либо по ссылке на репозиторий), соберите проект без ошибок стандартными командами сборки **Xcode** версии <= 11.5.

Подробнее дополнительные настройки анализа описаны в разделе Общие.

6.1.2.3. Подробное описание запуска анализа для C/C++ приложений

В Solar appScreener реализованы следующие способы загрузки C/C++ проектов для анализа.

- **Загрузить приложение с локального компьютера**. Загрузить архив с исходным кодом (поддерживаются следующие форматы архивов: ZIP, 7Z, RAR (до версии 4.0), EAR, AAR, tar.bz2, tar.gz, tar, cpio), либо исполняемый файл приложения(`.dll` или `.exe`-файл).

  Если сборка проекта не настроена, рекомендуется добавить в архив заголовочные файлы зависимостей. Это улучшит результаты анализа.

- **Загрузить приложение по ссылке**.

  Указывается ссылка на репозиторий с исходным кодом проекта (поддерживается **Git** и **Subversion**). Машина, на которой установлен Solar appScreener, должна иметь доступ к указанному репозиторию. С помощью указанной ссылки должно быть возможно скачивание кода из репозитория с использованием команды `git clone` (для **Git**) или `svn export` (для **Subversion**). Примеры ссылок:

  - https://gitlab.example.com/myproject.git (**Git**)
  - ssh://gitlab.example.com/myproject (**Git**)
  - https://svn.example.com/mysvnproject/trunk/ (**Subversion**)
  - svn://svn.example.com/mysvnproject/branches/my-branch (**Subversion**)
  - svn+ssh://svn.example.com/mysvnproject/ (**Subversion**)

Чтобы проанализировать код из приватного репозитория, укажите учётные данные. Подробнее про настройки анализа в разделе Настройки.

Со списком расширений файлов, которые анализируются при загрузке архива или по ссылке на репозиторий, можно ознакомиться в приложении (табл. 8.1). Чтобы проанализировать вложенный архив, установите дополнительную настройку **Анализировать библиотеки и вложенные архивы**.

Чтобы провести анализ исходного кода (либо в виде архива, либо по ссылке на репозиторий), соберите проект без ошибок указанными средствами:

- Для сборки C/C++ на **Linux/macOS** поддерживаются CMake проекты (версии <= 3.17.3), которые собираются командами `mkdir build && cd build && cmake .. && make` в среде, на которой настроен анализатор Solar appScreener. Поддерживаемые версии компиляторов:

  - **Apple Clang** <= 11.0.3
  - **Open source Clang** <= 10.0.0
  - **GCC** <= 10.1

- Для сборки C/C++ на **Linux**, помимо CMake, поддерживаются:
  - проекты Makefile:
  - проекты Autotools:
  - сборка через пользовательский **неинтерактивный** shell-скрипт. Исполняемый файл shell-скрипта с именем **build.sh** должен быть предоставлен вместе с остальными файлами проекта. Скрипт должен собирать проект по вызову из локации в файловом дереве проекта.
- Для сборки C/C++ на **Windows** поддерживаются проекты Visual Studio (2005, 2008, 2010, 2012, 2013, 2015, 2017, 2019) (выберите соответствующую опцию в настройках анализа), которые собираются командой `msbuild <project_name>.sln`.

Таким образом, либо CMake/Visual Studio проект должен сам устанавливать свои зависимости, либо они должны быть установлены на машине с анализатором Solar appScreener заранее.

Подробнее дополнительные настройки анализа описаны в разделе Общие.

6.1.2.4. Подробное описание запуска анализа для macOS-приложений

В Solar appScreener реализованы следующие способы загрузки macOS-проектов для анализа.

- **Загрузить приложение с локального компьютера**. Загрузите архив с исходным кодом (поддерживаются следующие форматы архивов: ZIP, 7z, RAR (до версии 4.0), EAR, AAR, tar.bz2, tar.gz, tar, cpio) или исполняемый файл приложения (`.app`-файл, упакованный в ZIP-архив, поддерживаются бинарные файлы архитектур x86 и x64).
- **Загрузить приложение по ссылке**. Указывается ссылка на репозиторий с исходным кодом проекта (поддерживается **Git** и **Subversion**). Машина, на которой установлен Solar appScreener, должна иметь доступ к указанному репозиторию. С помощью указанной ссылки должно быть возможно скачивание кода из репозитория с использованием команды `git clone` (для **Git**) или `svn export` (для **Subversion**). Примеры ссылок:
  - https://gitlab.example.com/myproject.git (**Git**)
  - ssh://gitlab.example.com/myproject (**Git**)
  - https://svn.example.com/mysvnproject/trunk/ (**Subversion**)
  - svn://svn.example.com/mysvnproject/branches/my-branch (**Subversion**)
  - svn+ssh://svn.example.com/mysvnproject/ (**Subversion**)

  Чтобы проанализировать код из приватного репозитория, укажите учётные данные. Подробнее про настройки анализа в разделе Настройки.

Со списком расширений файлов, которые анализируются при загрузке архива или по ссылке на репозиторий, можно ознакомиться в приложении (табл. 8.1). Чтобы проанализировать вложенный архив, установите дополнительную настройку **Анализировать библиотеки и вложенные архивы**.

Чтобы получить исполняемый файл приложения, который поддерживается системой:

1. Соберите приложение в **Xcode** и получите директорию с расширением `.app`;

2. Упакуйте её в ZIP-архив.

Чтобы провести анализ исходного кода (либо в виде архива, либо по ссылке на репозиторий), соберите проект без ошибок стандартными командами сборки **Xcode** версии <= 11.5.

Подробнее дополнительные настройки анализа описаны в разделе Общие.

6.1.2.5. Подробное описание запуска анализа для 1С-приложений

Поддерживается 8 версия 1С:Предприятие.

Вы можете выгрузить исходный код из 1С:Предприятие двумя способами: пользовательским и автоматизированным.

Пользовательский вариант

1. Укажите нужную информационную базу (ИБ) и нажмите кнопку **Конфигуратор**.

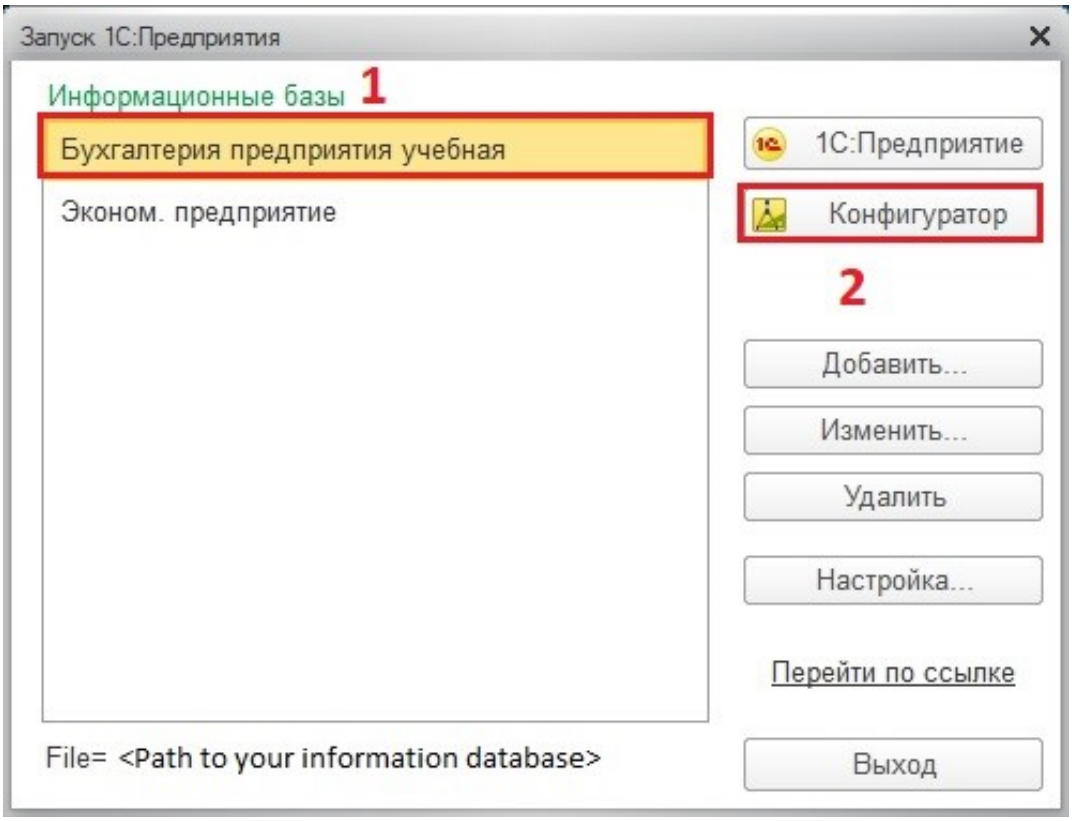


Рис. 6.5: Главное меню

2. В тулбаре выберите **Конфигурация->Выгрузить конфигурацию в файлы**.

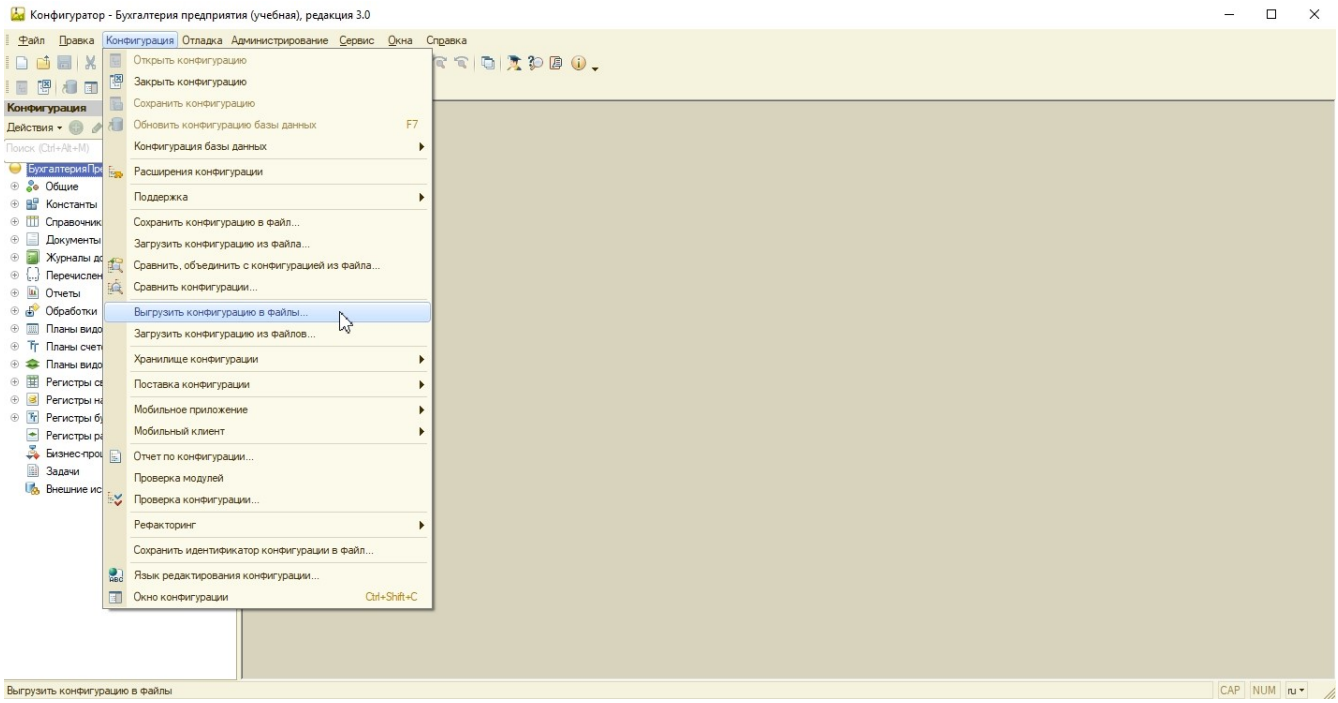
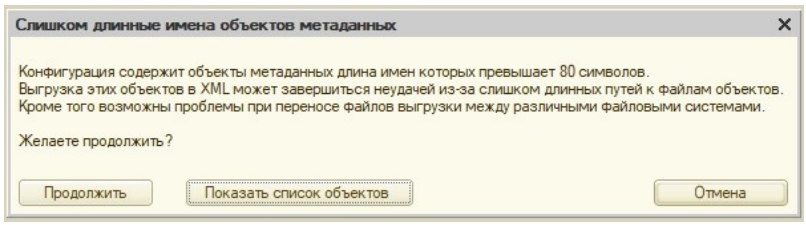


Рис. 6.6: Экспорт

3. Возможно предупреждение о слишком длинных именах объектов.



4. Нажмите кнопку **Показать список объектов**.

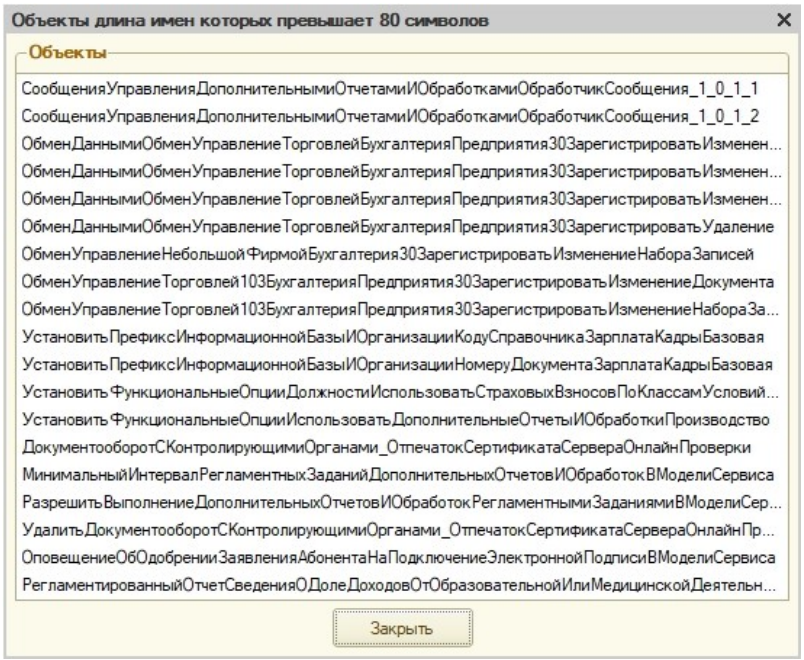


Рис. 6.7: Список объектов

5. Измените имена этих объектов либо проигнорируйте предупреждение и нажмите кнопку **Продолжить**.
6. Укажите путь до директории выгрузки файлов конфигурации.

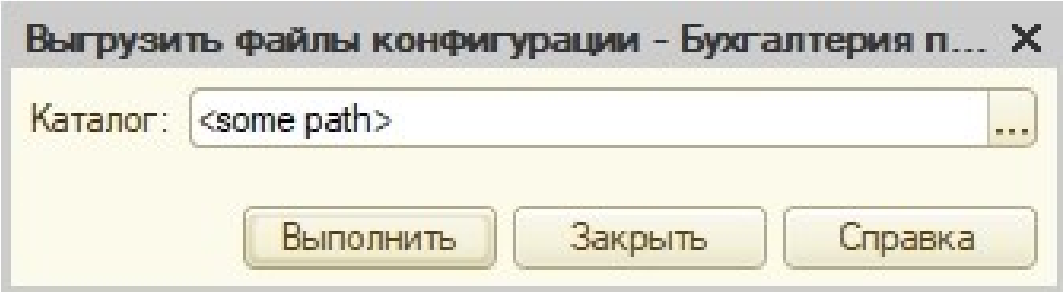


Рис. 6.8: Путь до директории выгрузки файлов конфигурации

7. Прогресс экспорта отображается в левом нижнем углу экрана.

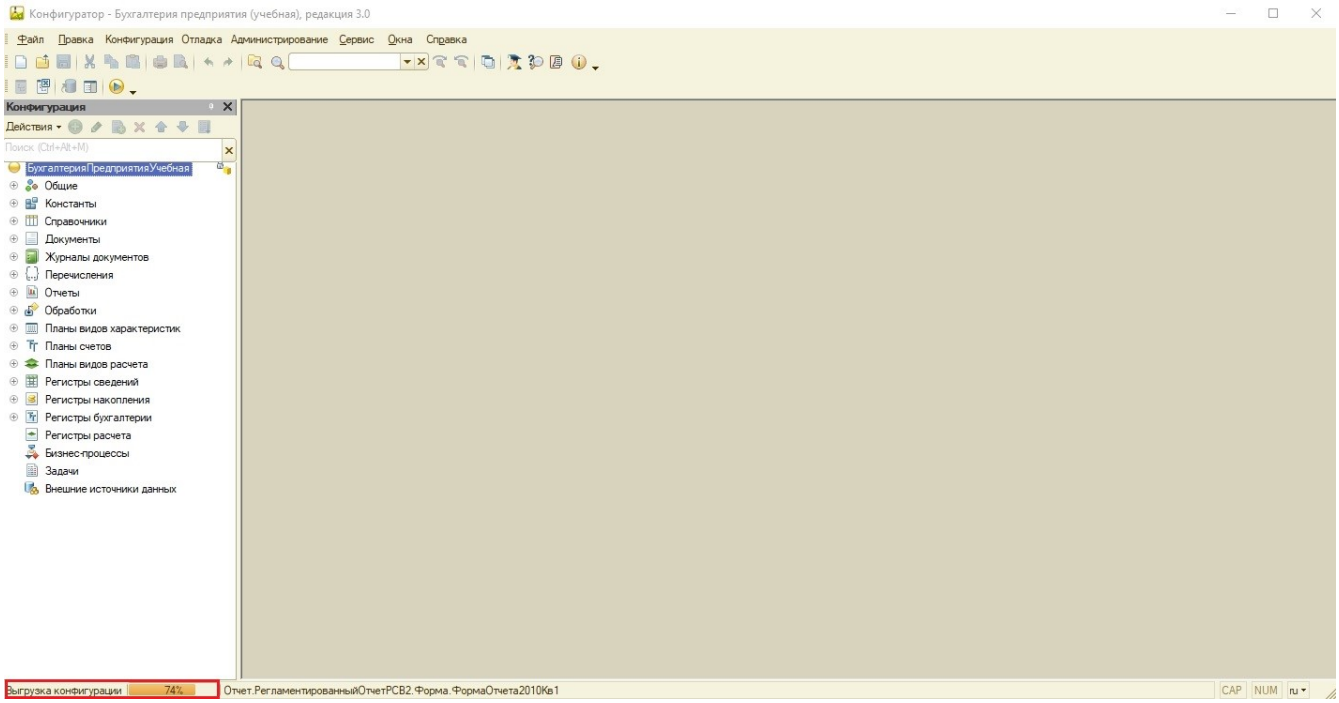
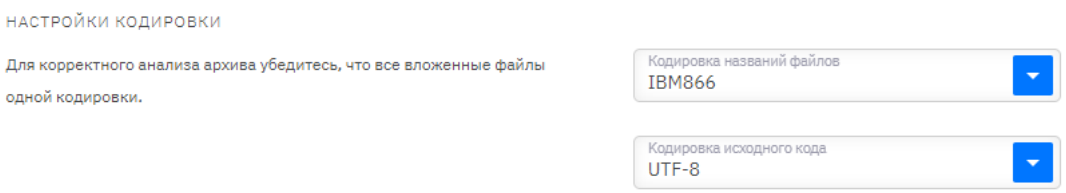


Рис. 6.9: Прогресс экспорта

8. Добавьте в архив директорию, полученную в результате выполненных действий.
9. Отправьте получившийся архив на анализ, выбрав в настройках проекта кодировку названий файлов **IBM866**.



Автоматизированный вариант

Инструкция реализована на примере ОС **Windows** для клиентов двух типов: «толстого» и «тонкого».

Для автоматической выгрузки кода из определенной конфигурации:

1. Если вы вели работу с конфигуратором в графическом режиме, завершите её.
2. Откройте для редактирования файл **1c_extractor.bat** со следующим содержанием (значения переменных заданы для наглядности).

   Для «толстого» клиента:

```
@echo off
set binpath=C:\"Program Files (x86)"\1cv8\8.3.13.1690\bin\1cv8.exe
set dbpath="C:\Users\Ivanov\Documents\InfoBaseIvanov"
set username="Ivanov"
set userpass="Password"
set folder="C:\Users\Ivanov\Desktop\ExtractModules"
set logs="C:\Users\Ivanov\Desktop\1C_extract_logs.log"
start %binpath% CONFIG /N%username% /P%userpass% /F%dbpath%
/DumpConfigToFiles%folder% /OUT%logs%
```

   Для «тонкого» клиента:

```
@echo off
set binpath=C:\"Program Files (x86)"\1cv8\8.3.13.1690\bin\1cv8c.exe
set server="server-base\InfoBaseIvanov"
set username="Ivanov"
set userpass="Password"
set folder="C:\Users\Ivanov\Desktop\ExtractModules"
set logs="C:\Users\Ivanov\Desktop\1C_extract_logs.log"
start %binpath% CONFIG /N%username% /P%userpass% /S%server%
/DumpConfigToFiles%folder% /OUT%logs%
```

   Пояснения к ключам:

   - **/N<значение>** имя пользователя для доступа к базе данных и учётной записи;
   - **/P<значение>** пароль доступа к учётной записи и базе данных. Если у пользователя нет пароля, этот параметр можно опустить;
   - **/F<путь>** используется для файловых баз, вместо <путь> вставьте путь к каталогу, где расположена ИБ, а не к файлу 1CD;

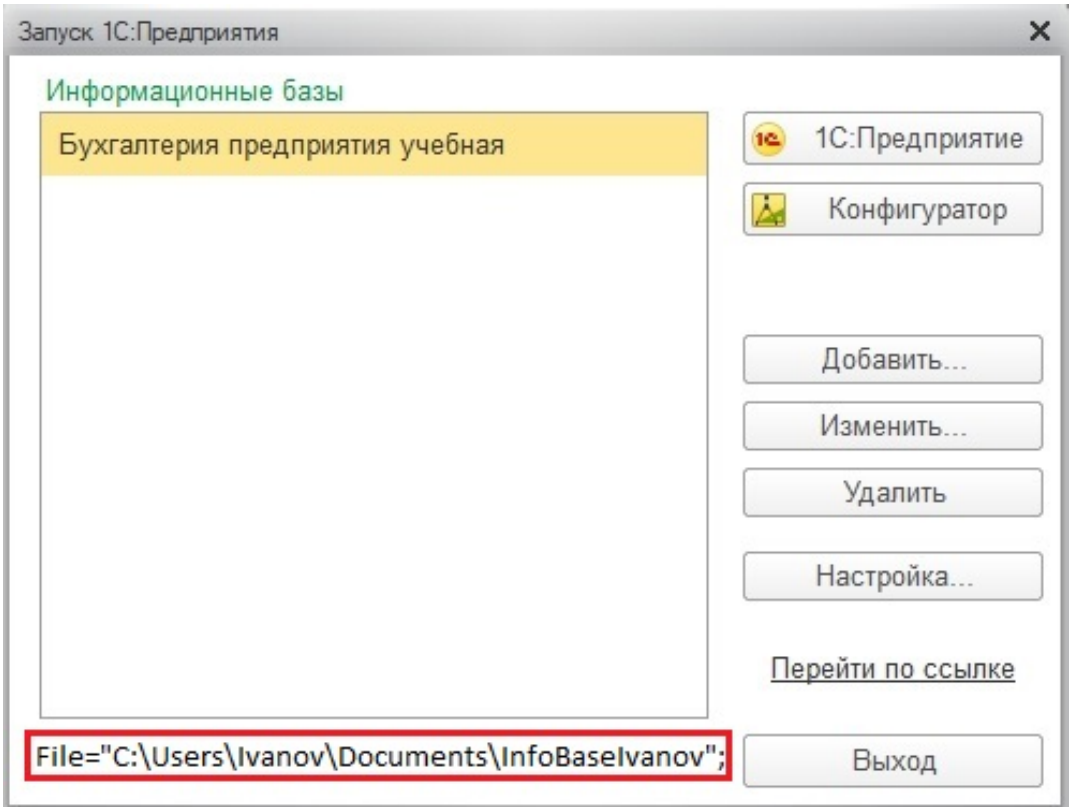


Рис. 6.10: Путь к базе

- **/S<адрес>** адрес информационной базы, хранящейся на сервере 1С:Предприятия 8, складывается следующим образом: <Имя компьютера, работающего сервером приложений>\<Ссылочное имя информационной базы, известное в рамках сервера 1С:Предприятия 8>
- **/OUT<путь>** путь до файла с логами процесса;
- **/DumpConfigFiles<путь>** выгрузка свойств объектов метаданных конфигурации, вместо <путь> указать каталог для расположения файлов.

3. Определите путь до исполняемого файла **1cv8.exe** («толстый» клиент) или **1cv8c.exe** («тонкий» клиент). Основной каталог, в который устанавливается 1С **C:\Program files\1Cv8** или **C:\Program Files (x86)\1Cv8**. Внутри каталога находятся другие каталоги с подверсиями. Уже в них находится папка **bin** с основным исполняемым файлом версии.

   Введите этот путь в строке **set binpath=...** .

4. Заполните оставшиеся переменные **username**, **userpass**, **server**, **folder**, **logs** для «тонкого» клиента и **username**, **userpass**, **dbpath**, **folder**, **logs** для «толстого» в соответствии с пояснениями для ключей, данными выше.

5. Сохраните все изменения в файле **1c_extractor.bat**.

6. Запустите **cmd** от имени Администратора, перейдите в директорию **1c_extractor.bat**, выполните команду `call 1c_extractor.bat`

7. Время выполнения экспорта зависит от объёма конфигурации и в среднем составляет 5-10 минут, в это время будет невозможно получить доступ к **Конфигуратору** через

интерактивный режим.

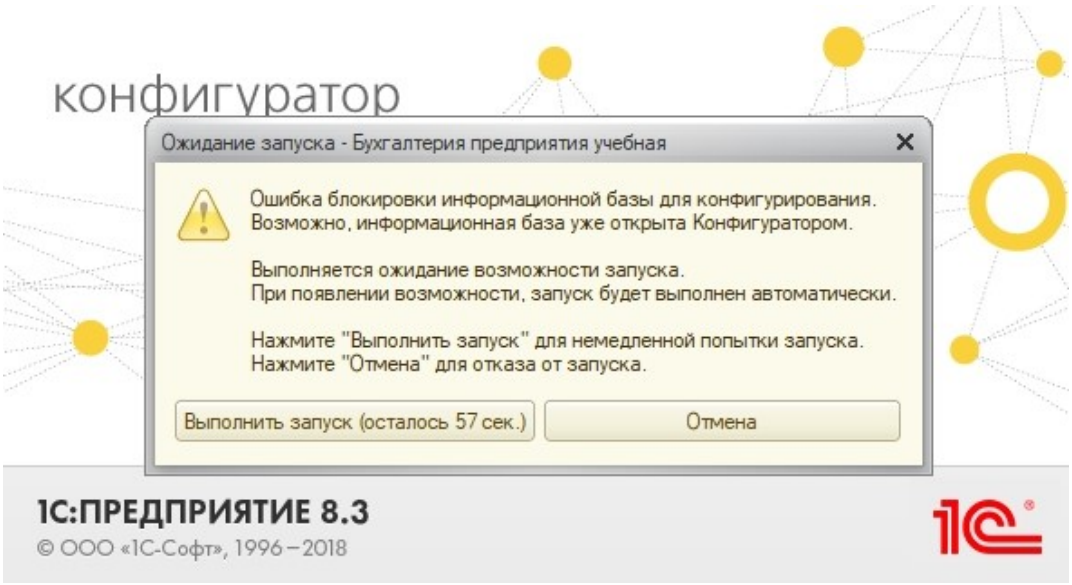


Рис. 6.11: Ожидание

8. В результате в директории, указанной в переменной **folder**, окажутся все файлы объектов метаданных конфигурации.
9. Добавьте эту директорию в архив и отправьте его на анализ, выбрав в настройках кодировку названий файлов **IBM866**.

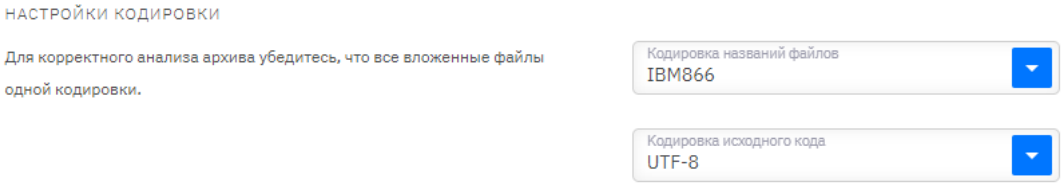


Рис. 6.12: Кодировка

6.1.2.6. Подробное описание запуска анализа для остальных приложений

В Solar appScreener реализованы следующие способы загрузки для анализа проектов, написанных на языках ABAP, Apex, C#, COBOL, Dart, Delphi, GO, Groovy, HTML, JavaScript, LotusScript, Objective-C, Pascal, PHP, PL/SQL, Python, Perl, Ruby, Rust, Solidity, Swift, T-SQL, TypeScript, VB.NET, VBA, VBScript, Visual Basic или Vyper.

- **Загрузить приложение с локального компьютера**. Загрузите архив с исходным кодом (поддерживаются следующие форматы архивов: ZIP, 7Z, RAR (до версии 4.0), EAR, AAR, tar.bz2, tar.gz, tar, cpio).
- **Загрузить приложение по ссылке**. Указывается ссылка на репозиторий с исходным кодом проекта (поддерживается **Git** и **Subversion**). Машина, на которой установлен Solar appScreener, должна иметь доступ к указанному репозиторию. С помощью указанной ссылки должно быть возможно скачивание кода из репозитория с использованием команды `git clone` (для **Git**) или `svn export` (для **Subversion**). Примеры ссылок:

- https://gitlab.example.com/myproject.git(**Git**)
- ssh://gitlab.example.com/myproject (**Git**)
- https://svn.example.com/mysvnproject/trunk/ (**Subversion**)
- svn://svn.example.com/mysvnproject/branches/my-branch (**Subversion**)
- svn+ssh://svn.example.com/mysvnproject/ (**Subversion**)

Чтобы проанализировать код из приватного репозитория, укажите учётные данные. Подробнее про настройки анализа в разделе Настройки.

Со списком расширений файлов, которые анализируются при загрузке архива или по ссылке на репозиторий, можно ознакомиться в приложении (табл. 8.1). Чтобы проанализировать вложенный архив, установите дополнительную настройку **Анализировать библиотеки и вложенные архивы**.

Подробнее про дополнительные настройки анализа, которые доступны при запуске сканирования после нажатия на кнопку **Показать настройки** (рис. 6.4), в разделе Общие.

6.1.3. Запуск сканирования из командной строки

*Для того чтобы посмотреть раздел **help**, выполните команду:* `java -jar clt.jar -help`.

Запуск сканирования из командной строки доступен только при условии предустановленной Java 11 или новее. Для запуска сканирования из командной строки следует выполнить команду:

```
java -jar clt.jar -rest [rest URL] -token [token] start [options]
```

- **-rest** адрес API.
- **-token** токен авторизации API; можно получить в интерфейсе пользователя в разделе **Личный кабинет** (см. раздел Личный кабинет).

Аргументы (options):

Обязательные аргументы:

- **-type** возможные значения: FILE, LINK, REPO.
- **-path** используется, если type=FILE (путь к каталогу или файлу для анализа).
- **-link** используется, если type=LINK (Google Play или App Store URL).
- **-vcs.url** используется, если type=REPO (URL репозитория).

  Указывается ссылка на репозиторий с исходным кодом проекта (поддерживается **Git** и **Subversion**). Машина, на которой установлен Solar appScreener, должна иметь доступ к указанному репозиторию. С помощью указанной ссылки должно быть возможно скачивание кода из репозитория с использованием команды `git clone` (для **Git**) или `svn export`(для **Subversion**). Примеры ссылок:

  - https://gitlab.example.com/myproject.git (**Git**)
  - ssh://gitlab.example.com/myproject (**Git**)
  - https://svn.example.com/mysvnproject/trunk/ (**Subversion**)
  - svn://svn.example.com/mysvnproject/branches/my-branch (**Subversion**)
  - svn+ssh://svn.example.com/mysvnproject/ (**Subversion**)

Чтобы проанализировать код из приватного репозитория, укажите учётные данные. Подробнее про настройки анализа в разделе Настройки.

Необязательные аргументы:

- **-disableSsl**.
- **-name** название проекта.
- **-languages** языки, которые необходимо включить в анализ:

  ABAP, APEX, CCPP, COBOL, CS, DART, DELPHI, GO, GROOVY, HTML5, JAVA, JAVASCRIPT, LOTUSSCRIPT, OBJC, ONES, PASCAL, PHP, PLSQL, PYTHON, PERL, RUBY, RUST, SOLIDITY, SWIFT, TSQL, TYPESCRIPT, VBSCRIPT, VB, VBA, VBNET, VYPER или любой набор из этих языков, разделенный запятой, например: JAVA,CS,VB;

  по умолчанию анализируются все языки.
- **-vcs.branch** ветка репозитория, если не master (при анализе приложений по ссылке на репозиторий).
- **-vcs.login** логин репозитория.
- **-vcs.password** пароль репозитория.
- **-vcs.saveCredentials** использовать учётные данные при пересканировании.
- **-fileSelector** директории, файлы и/или пакеты, которые необходимо включить/исключить из анализа, разделенные точкой с запятой.
- **-iconPath** путь к логотипу проекта.
- **-ruleSet** UUID наборов правил, разделённые запятой.
- **-sourceEncoding** кодировка исходного кода.
- **-nameEncoding** кодировка названий файлов.
- **-useExtraRules**
- **-analyzelibs** анализировать ли библиотеки или вложенные архивы.
- **-analyzeJsLibs** анализировать стандартные библиотеки JavaScript.
- **-isVisualStudio** проект Visual studio.
- **-nobuild** не собирать проект (для Java-проектов).
- **-saveFile** полностью сохранить загружаемый файл.
- **-incremental** инкрементальный анализ. Значение по умолчанию false.
- **-uuid** UUID проекта, в котором будет запущено сканирование.

UUID проекта можно получить в боковом меню проекта в интерфейсе. Справа от логотипа проекта отображается ID (первые шесть символов UUID проекта). Чтобы скопировать в буфер полный UUID, нажмите на иконку .

Пример:

```
java -jar clt.jar -rest http://<installation_address>/app/api/v1
-token jfghkdjghfkdjgfdkg start -type LINK -link
https://play.google.com/store/apps/details
id=com.redphx.deviceid
```

Для проверки статуса сканирования из командной строки следует выполнить команду:

```
java -jar clt.jar -rest [rest URL] -token [token] status [options]
```

- **-rest** адрес API.
- **-token** токен авторизации API; можно получить в интерфейсе пользователя в разделе **Личный кабинет** (см. раздел Личный кабинет).

Аргументы (options):

- **-scanid** UUID сканирования.

UUID сканирования можно получить в информации о сканировании. Чтобы скопировать в буфер UUID сканирования, нажмите на иконку .

Пример:

```
java -jar clt.jar -rest http://<installation_address>/app/api/v1 -token
kijlkjlkjlkjlkjlkjgkuy status -scanid b001eab1-ba6c-4b05-9066-e84d594210e2
```

Для проверки уровня безопасности сканирования из командной строки следует выполнить команду:

```
java -jar clt.jar -rest [rest URL] -token [token] score [options]
```

- **-rest** адрес API.
- **-token** токен авторизации API; можно получить в интерфейсе пользователя в разделе **Личный кабинет** (см. раздел Личный кабинет).

Аргументы (options):

- **-scanid** UUID сканирования.

UUID сканирования можно получить в информации о сканировании. Чтобы скопировать в буфер UUID сканирования, нажмите на иконку .

Пример:

```
java -jar clt.jar -rest http://<installation_address>/app/api/v1 -token
kijlkjlkjlkjlkjlkjgkuy score -scanid b001eab1-ba6c-4b05-9066-e84d594210e2
```

Для экспорта отчёта в формате PDF из командной строки следует выполнить команду:

```
java -jar clt.jar -rest [rest URL] -token [token] export [options]
```

Для экспорта отчёта в формате CSV из командной строки следует выполнить команду:

```
java -jar clt.jar -rest [rest URL] -token [token] export [options]
'-general.format' CSV
```

- **-rest** адрес API.
- **-token** токен авторизации API; можно получить в интерфейсе пользователя в разделе **Личный кабинет** (см. раздел Личный кабинет).

Аргументы (options):

Аргументы включаются в отчёт, если они принимают значение **true**. По умолчанию аргументы принимают значение **false**, если не указано иное.

Обязательные аргументы:

- **-path** путь до директории, куда следует поместить отчёт.
- **-project** UUID проекта.

UUID проекта можно получить в боковом меню проекта в интерфейсе. Справа от логотипа проекта отображается ID (первые шесть символов UUID проекта). Чтобы скопировать в буфер полный UUID, нажмите на иконку .

Опциональные аргументы:

- **-scans** UUID сканирований, которые следует включить в отчёт. Если сканирований несколько, то перечислять их следует через запятую.
- **-filter.classFiles** включить в отчёт уязвимости в `.class`-файлах (включается по умолчанию).
- **-filter.critical** включить в отчёт уязвимости критического уровня (включается по умолчанию).
- **-filter.info** включить в отчёт уязвимости информационного уровня.
- **-filter.jira** включить в отчёт уязвимости с созданными задачами в jira (включается по умолчанию).
- **-filter.low** включить в отчёт уязвимости низкого уровня.
- **-filter.medium** включить в отчёт уязвимости среднего уровня (включается по умолчанию).
- **-filter.stdLibs** включить в отчёт уязвимости в стандартных библиотеках (включается по умолчанию).
- **-filter.waf** включить в отчёт уязвимости без инструкций по настройке WAF (включается по умолчанию).
- **-fuzzy.included** включить в отчёт данные FLE. При включении укажите:
  - **-fuzzy.critical** настройка значения FLE для критических уязвимостей (по умолчанию 0).
  - **-fuzzy.info** настройка значения FLE для уязвимостей информационного уровня (по умолчанию 0).
  - **-fuzzy.low** настройка значения FLE для уязвимостей низкого уровня (по умолчанию 0).

- **-fuzzy.medium** настройка значения FLE для уязвимостей среднего уровня (по умолчанию 0).

- **-general.contents** включить оглавление (включается по умолчанию).
- **-general.included** включить настройки экспорта (включается по умолчанию).
- **-general.locale** настроить язык отчёта (по умолчанию английский).
- **-languages** включить анализ языков в отчёт. Если языков несколько, то перечислять их следует через запятую, без пробелов (по умолчанию включены все языки). Доступные значения: ABAP, APEX, CS, CCPP, COBOL, CONFIG, DART, DELPHI, GO, GROOVY, HTML5, JAVA, JAVASCRIPT, LOTUSSCRIPT, KOTLIN, OBJC, PASCAL, PHP, PLSQL, PYTHON, PERL, RUBY, RUST, SCALA, SOLIDITY, SWIFT, TSQL, TYPESCRIPT, VBNET, VBA, VBSCRIPT, VB, VYPER, ONES.
- **-projectInfo.scanHistory** количество последних сканирований в отчёте (по умолчанию 0):
  - -1— не выгружать историю сканирований;
  - 0 — выгрузить всю историю сканирований;
  - >0 — число последних сканирований.
- **-projectInfo.securityDynamic** включить динамику уровня безопасности (включается по умолчанию).
- **-projectInfo.vulnerabilityDynamic** включить динамику количества уязвимостей (включается по умолчанию).
- **-results.included** включить подробные результаты (включается по умолчанию). При включении укажите:
  - **-results.comment** включить комментарии к уязвимостям (включается по умолчанию).
  - **-results.confirmed** включить подтверждённые уязвимости (включается по умолчанию).
  - **-results.entryNum** настроить количество вхождений уязвимости (по умолчанию 0):
    - -1 — не выгружать вхождения;
    - 0 — выгрузить все вхождения;
    - >0 — выгрузить не более чем вхождений.
  - **-results.jiraInfo** включить информацию о задачах в jira (включается по умолчанию).
  - **-results.notProcessed** включить необработанные уязвимости (включается по умолчанию).
  - **-results.rejected** включить отклонённые уязвимости.
  - **-results.sourceCodeNum** настроить размер контекста (по умолчанию 7).
    - -1 — не выгружать исходный код;
    - all — выгрузить весь исходный код;
    - >=0 — выгрузить введённое количество строк до и после строки с уязвимостью.
  - **-results.traceNum** включить элементы трассы (по умолчанию 1):

  - -1 — не выгружать элементы трассы;
  - 0 — выгрузить все элементы;
  - 1 — выгрузить только первый и последний элементы.
- **-scanInfo.included** включить настройки сканирования (включается по умолчанию). При включении укажите:
  - **-scanInfo.errorInfo** включить информацию об ошибках сканирования (включается по умолчанию).
  - **-scanInfo.foundChart** включить диаграмму найденных уязвимостей (включается по умолчанию).
  - **-scanInfo.langStat** включить статистику по языкам (включается по умолчанию).
  - **-scanInfo.settings** включить настройки сканирования (включается по умолчанию).
  - **-scanInfo.typeChart** включить диаграмму найденных уязвимостей (включается по умолчанию).
- **-sort** настроить метод классификации уязвимостей. Доступные значения: CR, OWASP_13, OWASP_14, OWASP_16, OWASP_17, OWASP_21, MASVS_L1, MASVS_L2, MASVS_L1_R, MASVS_L2_R, PCI_DSS, HIPAA, CWE_SANS_11, CWE_SANS_21. Значение по умолчанию: CR.
- **-table.included** включить список уязвимостей (включается по умолчанию). При включении в отчёт укажите:
  - **-table.confirmed** включить в список подтверждённые уязвимости (включается по умолчанию).
  - **-table.entryNum** список вхождений уязвимости в списке уязвимостей (по умолчанию 0):
    - -1 — не выгружать вхождения;
    - 0 — выгрузить все вхождения;
    - >0 — выгрузить введённое количество вхождений.
  - **-table.notProcessed** включить в список необработанные уязвимости (включается по умолчанию).
  - **-table.rejected** включить в список отклонённые уязвимости.
- **-waf.included** включить инструкции по настройке WAF (включается по умолчанию). При включении в отчёт укажите:
  - **-waf.confirmed** включить инструкции по настройке WAF для подтверждённых уязвимостей (включается по умолчанию).
  - **-waf.f5** включить рекомендации по настройке **F5** (включается по умолчанию).
  - **-waf.imperva** включить рекомендации по настройке **Imperva SecureSphere** (включается по умолчанию).
  - **-waf.mod** включить рекомендации по настройке **ModSecurity** (включается по умолчанию).
  - **-waf.notProcessed** включить инструкции по настройке WAF для необработанных уязвимостей (включается по умолчанию).

- **-waf.rejected** включить инструкции по настройке WAF для отклонённых уязвимостей.

- **-comparison.included** включать ли в отчёт сравнение с предшествующим сканированием. При включении в отчёт укажите:
  - **-comparison.ScanUuid** UUID предшествующего сканирования для сравнения (обязательный параметр).
  - **-comparison.fixed** включить в отчёт устранённые уязвимости.
  - **-comparison.newIssue** включать ли в отчёт новые уязвимости (включается по умолчанию).
  - **-comparison.saved** включить в отчёт сохранившиеся уязвимости (включается по умолчанию).
  - **-comparison.entryNum** включить количество вхождений уязвимости (по умолчанию 0):
    - -1 — не выгружать вхождения;
    - 0 — выгрузить все вхождения;
    - >0 — выгрузить введённое количество вхождений.
  - **-comparison.scanSettings** включить в отчёт настройки сравнения сканирований (включается по умолчанию).

Пример:

```
java -jar clt.jar -rest http://<installation_address>/app/api/v1 -token
kljkjljlkjljklkjk export -scanid ec59395b-4372-47b1-95a2-4d48b044ff0b
-path C:\test -default
```

Важно обратить внимание:

Раздел **Информация о сканировании** не будет включён в отчёт, если вы не укажете UUID необходимого сканирования (аргумент **-scans**). UUID сканирования можно получить в информации о сканировании. Чтобы скопировать в буфер UUID сканирования, нажмите на иконку  .

Описанная выше функциональность также доступна через REST.

6.1.4. Запуск сканирования из инструментов сборки

Запуск сканирования возможен из средств сборки программ для Java, Scala, Kotlin: **Maven**, **Gradle**, **SBT**.

Запуск производится автоматически с помощью инструмента командной строки Solar appScreener **clt.jar**. В настройках для каждого сборщика передаются следующие параметры:

- **-token** токен авторизации API; можно получить в интерфейсе пользователя в разделе **Личный кабинет** (см. раздел Личный кабинет).
- **-rest** адрес API.

- путь к файлу **clt.jar**.

В **Eclipse** настройки запуска сканирования устанавливаются с помощью пункта меню **appScreener->Settings**, запуск сканирования производится с помощью пункта меню **appScreener->Run**. (см. раздел Eclipse)

6.2. Управление проектом

Управление проектом состоит из разделов **Обзор**, **Подробные результаты**, **Сканирования**, **Экспорт отчёта**, **Сравнение сканирований** и **Настройки**. Переключение между этими разделами осуществляется через меню в левой части страницы. Меню может быть представлено в виде иконок с текстом или только иконок, что регулируется нажатием на кнопку в левом нижнем углу страницы (рис. 6.13).

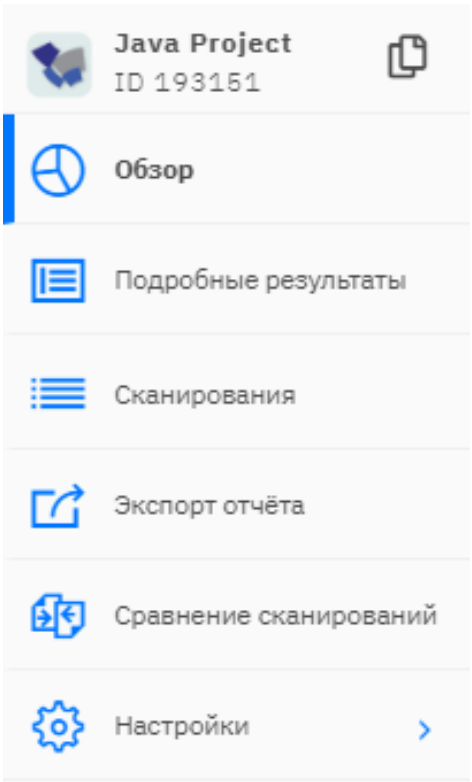


Рис. 6.13: Боковое меню

Справа от логотипа проекта отображается ID (первые шесть символов UUID проекта). Чтобы скопировать в буфер полный UUID, нажмите на .

На страницу **Обзор** можно перейти, нажав на название проекта на странице **Проекты** или на **Домашней странице** (если проект входит в шесть последних запущенных проектов).

На страницы **Подробные результаты** или **Экспорт отчёта** можно перейти, нажав на соответствующие кнопки быстрой навигации на странице **Проекты** или на **Домашней странице** (если проект входит в шесть последних запущенных проектов).

6.2.1. Обзор

В разделе **Обзор** (рис. 6.15) в правом верхнем углу можно выбрать сканирование, для которого будет отображаться статистика по сканированию. Нажмите на иконку ⓘ, чтобы отобразились параметры запуска анализа для выбранного сканирования.

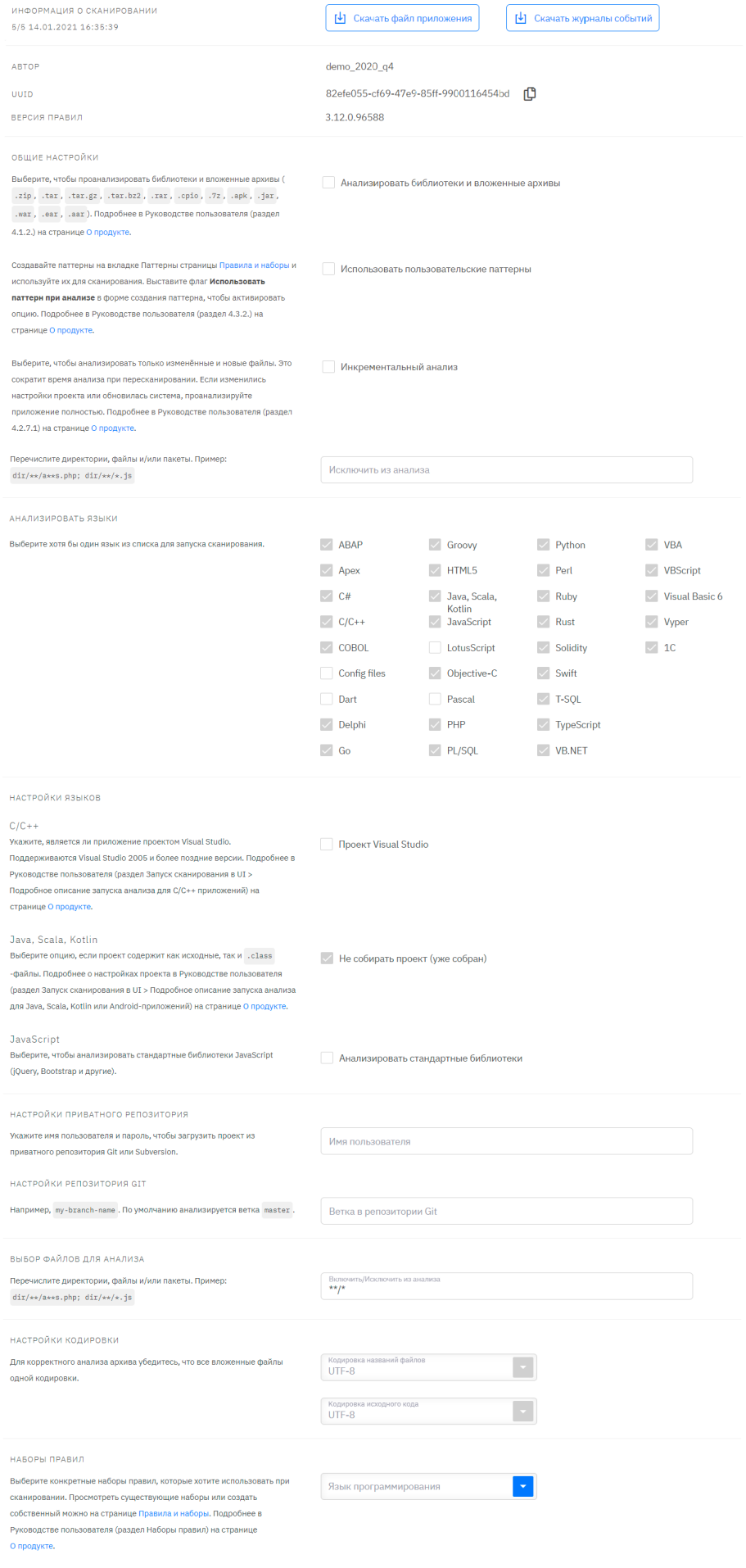


Рис. 6.14: Параметры запуска анализа

На странице **Обзор** представлена следующая информация:

- рейтинг;
- статус сканирования;
- продолжительность сканирования;
- количество строк кода;
- количество уязвимостей каждого уровня критичности;
- статистика по проанализированным языкам;
- статистика по проанализированным файлам;
- графическая информация по сканированию и проекту:
  - диаграмма с количеством уязвимостей каждого уровня критичности в сканировании;
  - график уровня безопасности проекта;
  - график количества уязвимостей в проекте;
  - диаграмма с самыми распространенными уязвимостями в сканировании;
  - накопительная диаграмма со статистикой количества уязвимостей по языкам.

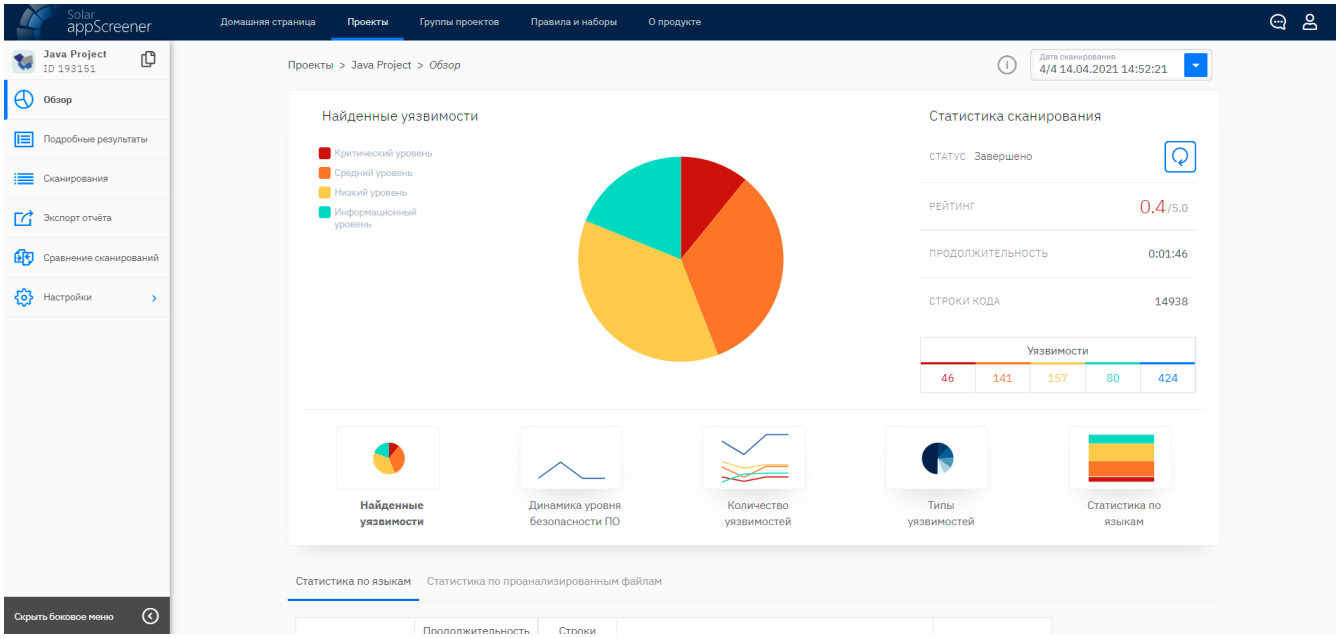


Рис. 6.15: Обзор

Если в данный момент приложение не сканируется, можно запустить новое сканирование, нажав на иконку . Если сканирование находится в процессе анализа, то его можно приостановить или остановить, нажав на иконку .

6.2.2. Подробные результаты

На вкладке **Подробные результаты** (рис. 6.16) отображается информация по каждой из обнаруженных уязвимостей для выбранного сканирования. Переключаться между результатами разных сканирований можно с помощью списка сканирований в правом верхнем углу.

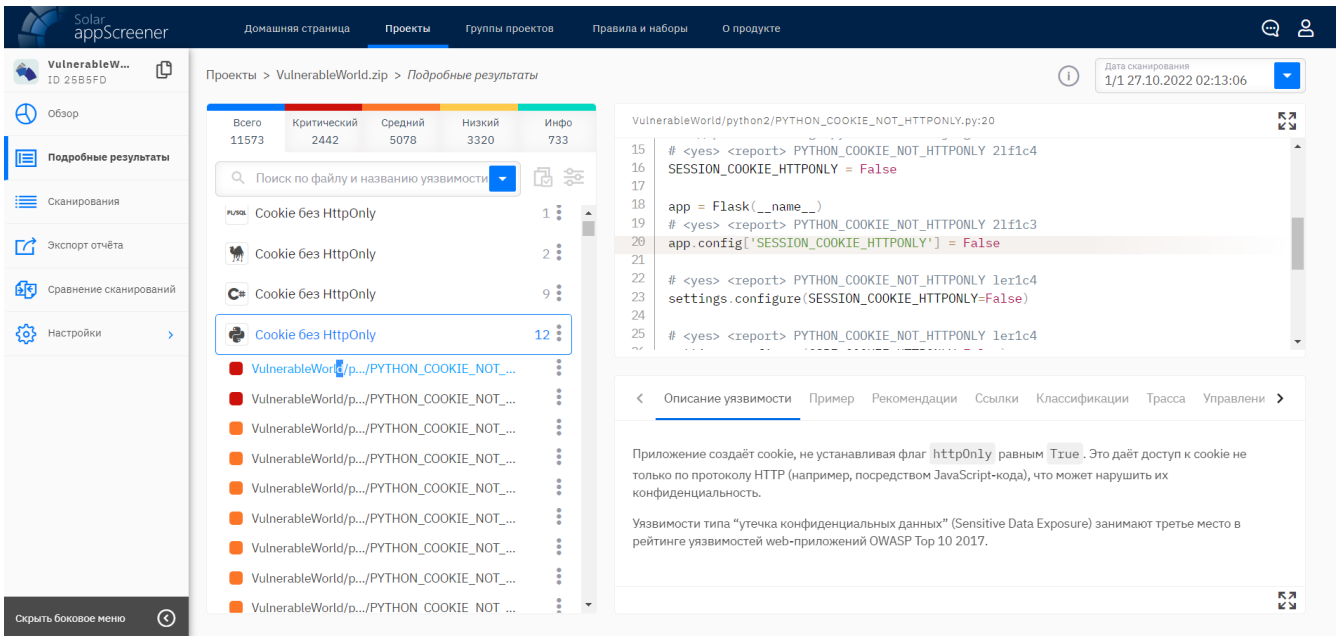


Рис. 6.16: Подробные результаты

В левой части страницы представлен список вхождений уязвимостей, сгруппированный по типам уязвимостей. При этом в верхнем меню можно выбрать, уязвимости какого уровня требуется отобразить. По желанию тип группировки можно изменить, следующие опции доступны по нажатию на иконку фильтров:

- по типу уязвимости;
- по файлу;
- по точке эксплуатации;
- по источнику уязвимости;
- по пакету.

Для удобной навигации по уязвимостям также предусмотрен поиск по названию уязвимости и файлу/ам с вхождением, а также фильтры (рис. 6.17).

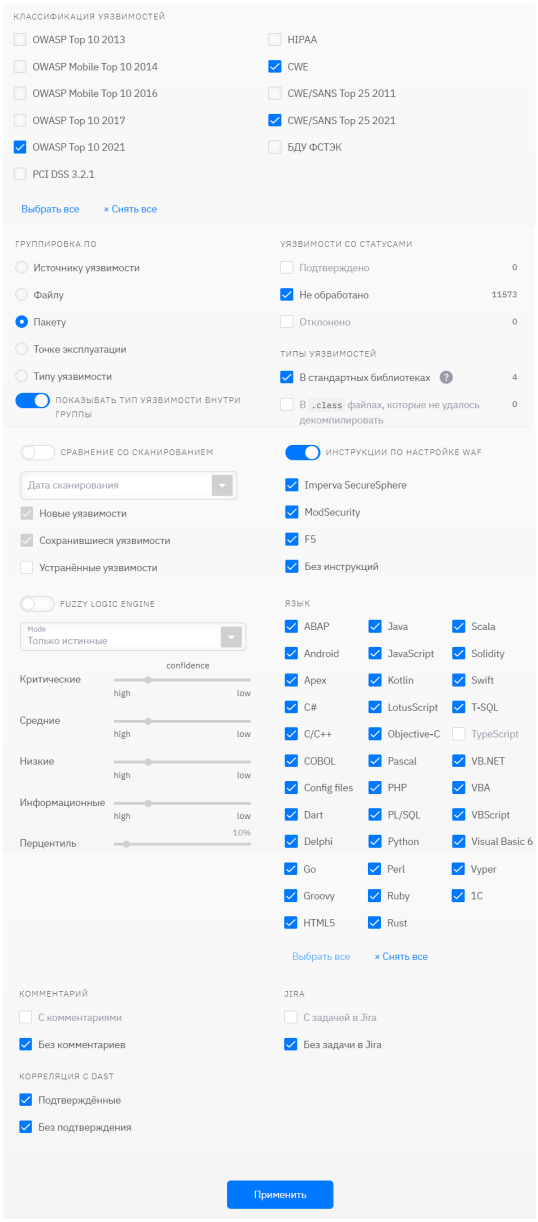


Рис. 6.17: Фильтры результатов

Фильтровать результаты можно по следующим параметрам:

- классификация уязвимостей
- статусы уязвимостей
- типы уязвимостей
- при наличии двух и более успешных сканирований в проекте можно сравнить текущее сканирование с одним из предшествующих и отобразить уязвимости в соответствии с их статусом (если при сравнении уязвимости не отслеживаются, перезапустите сканирование, изменив кодировку в настройках анализа)
- инструкции по настройке WAF
- включить Fuzzy Logic Engine с одним из режимов:

- только истинные — отображаются уязвимости с высокой вероятностью того, что вхождения являются реальной уязвимостью;
- только важные — отображаются уязвимости, на которые надо обратить внимание в первую очередь;
- пользовательский — есть возможность настроить чувствительность Fuzzy Logic Engine, перемещая ползунок в разные положения. Крайнее левое положение ползунка характеризует вхождения с самой высокой вероятностью корректного срабатывания, крайнее правое отобразит уязвимости для любой вероятности;
- динамический — можно установить перцентиль (значение от 1 до 100), в зависимости от которого будет отображаться определенная часть/процент самых важных уязвимостей.

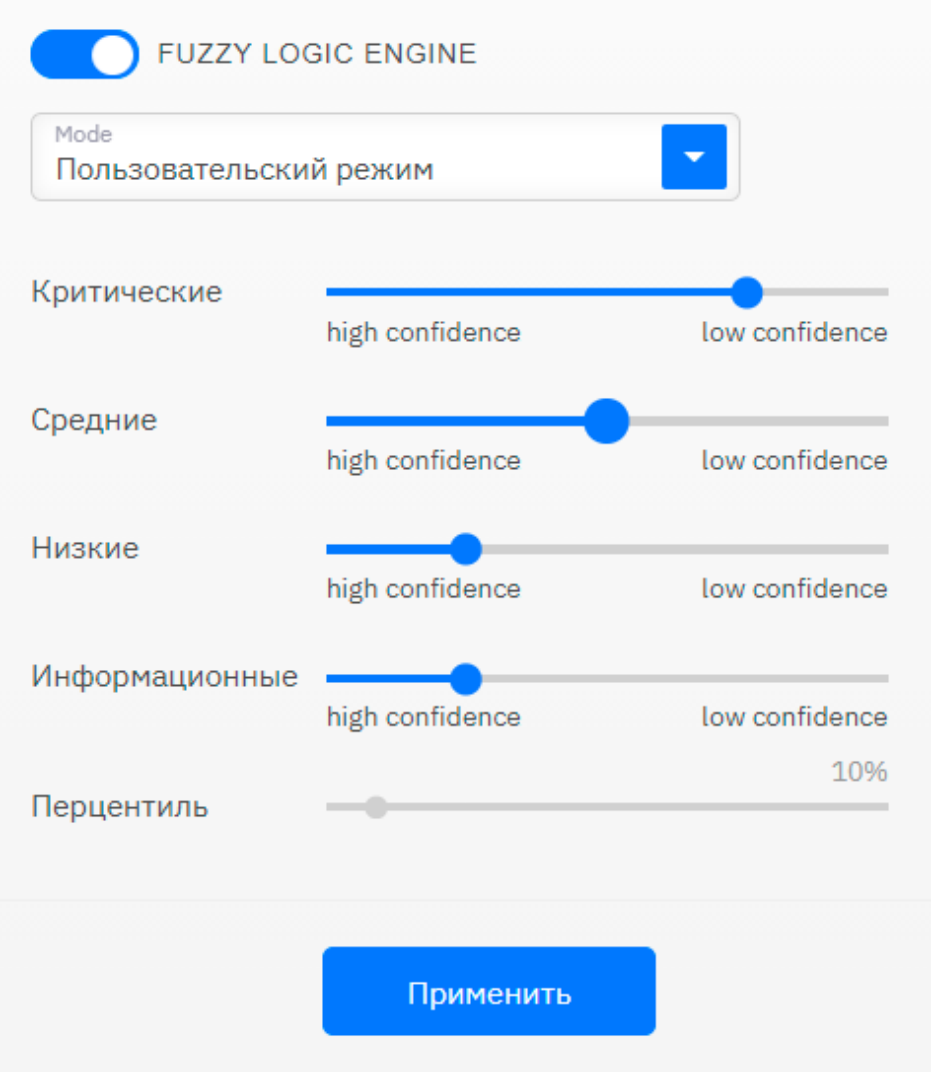


Рис. 6.18: Fuzzy Logic Engine

- языки, для которых был произведен анализ;
- наличие/отсутствие комментариев;
- наличие/отсутствие связанной задачи в **Jira**.

Фильтры применяются после нажатия на кнопку **Применить**.

Чтобы изменить статус или критичность уязвимости, нажмите на три точки справа от её названия. Вы также можете изменить статус/критичность или оставить комментарий для произвольной группы уязвимостей. Для этого перейдите в режим выбора нескольких уязвимостей, нажав на соответствующую иконку.

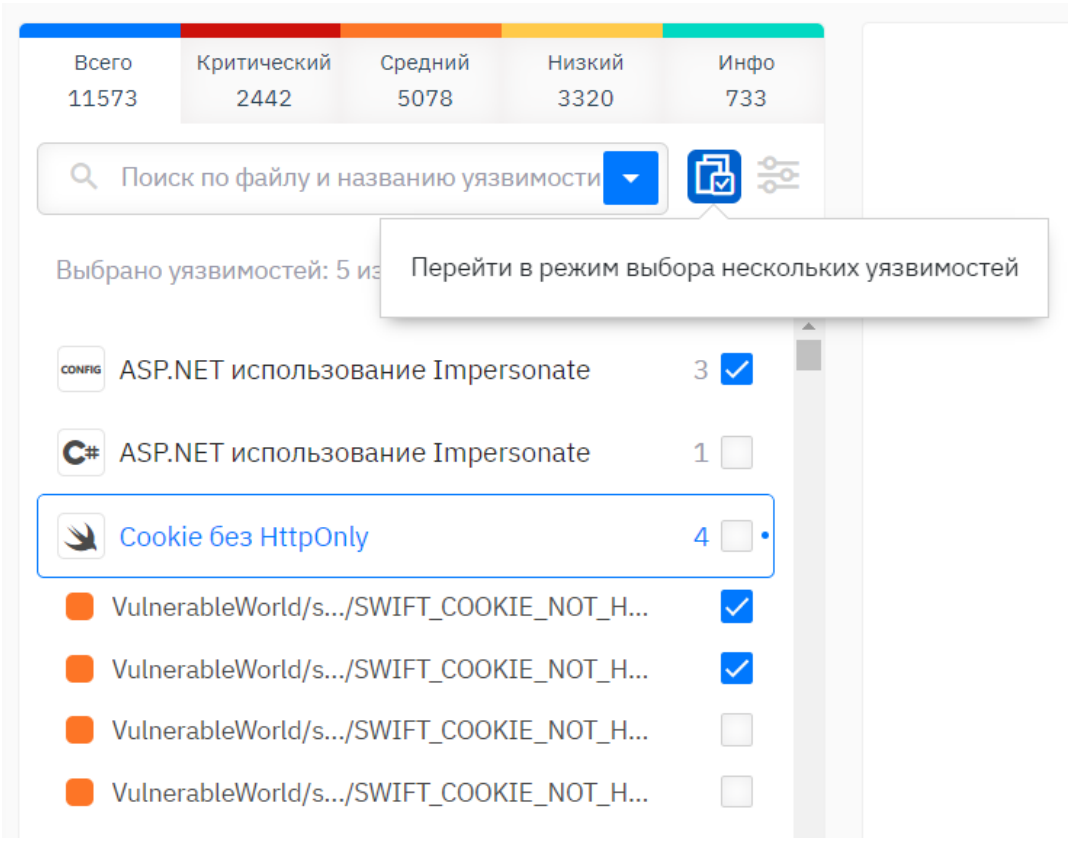


Рис. 6.19: Переход в режим выбора нескольких уязвимостей

При изменении статуса и уровня критичности уязвимостей пересчитывается уровень безопасности приложения. Уязвимости со статусом **Отклонено** не учитываются при подсчёте количества уязвимостей и рейтинга безопасности. При пересканировании изменения сохраняются (рис. 6.20).

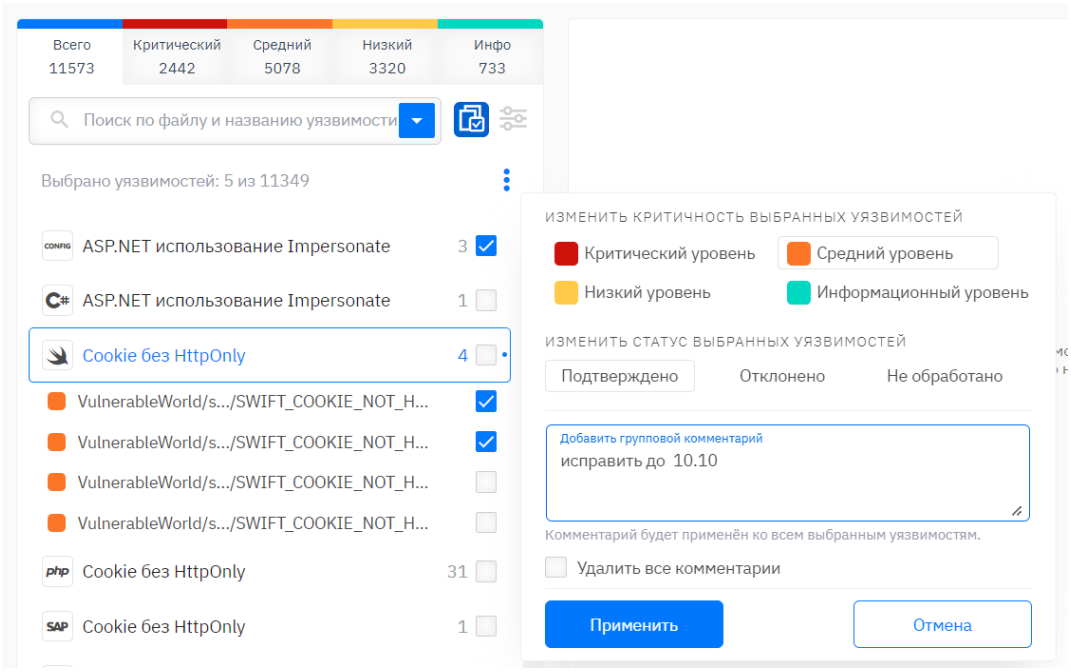


Рис. 6.20: Управление пакетом уязвимостей

После выбора конкретного вхождения уязвимости в центральной части страницы отображается соответствующий ему фрагмент исходного кода и название файла, в котором оно обнаружено. В нижней части страницы представлена информация по выбранному типу уязвимостей (рис. 6.21): **Описание**, **Пример**, **Рекомендации** по устранению, **Ссылки** на ресурсы, связанные с данным типом уязвимостей, рекомендации по **Настройке WAF** (для веб-приложений). Инструкции по настройке различных WAF отображаются на соответствующих вкладках.

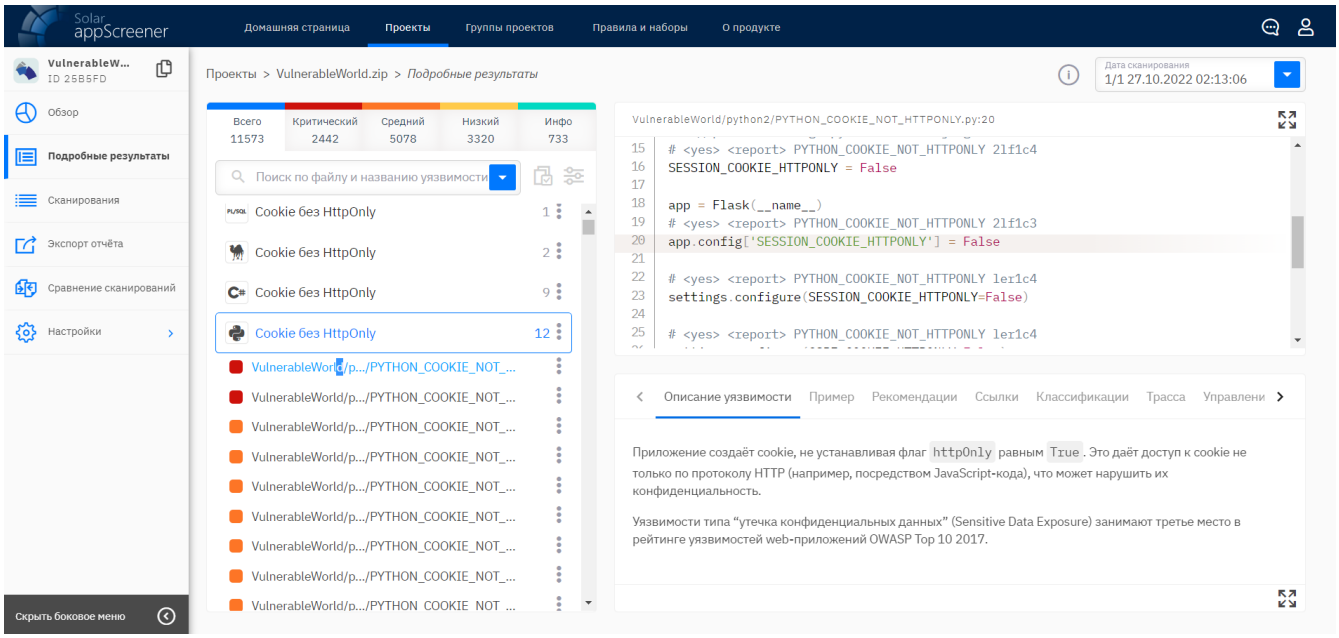


Рис. 6.21: Описание уязвимости

На вкладке **Классификации** (рис. 6.22) отображаются соответствующие пункты CWE

(версия 4.0), CWE/SANS Top 25, OWASP Top 10, OWASP Mobile 10, OWASP MASVS, PCI DSS 4.0, HIPAA и БДУ ФСТЭК (доступен, если выбран русский язык отчёта).

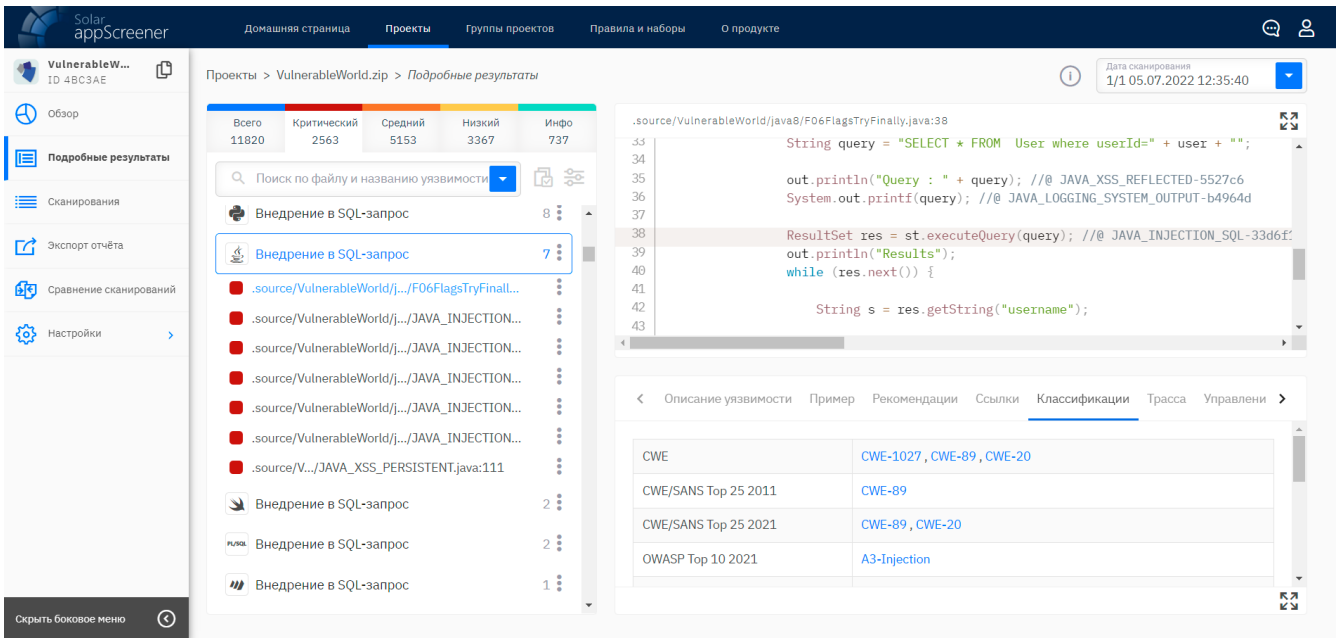


Рис. 6.22: Классификации

На вкладке **Трасса** (рис. 6.23) можно прослеживать распространение данных для уязвимости. Распространение данных представлено в виде диаграммы с кликабельными элементами, при нажатии на которые в центральной части страницы отображается соответствующий фрагмент исходного кода.

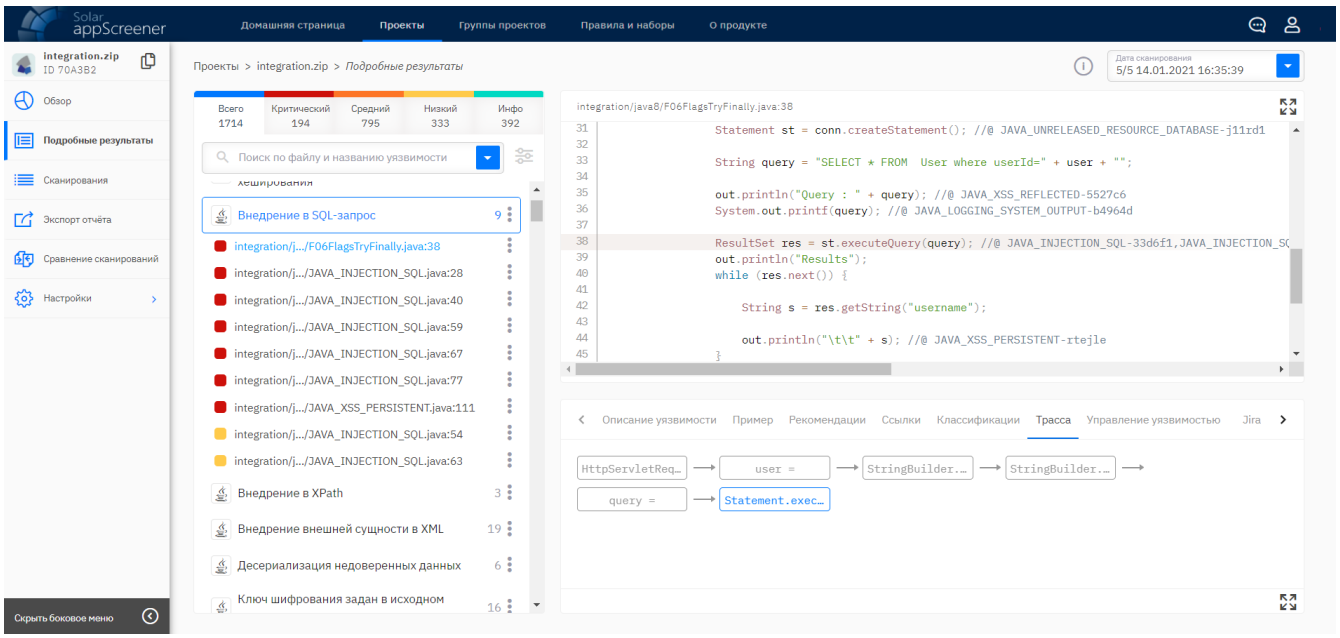


Рис. 6.23: Трасса

На вкладке **Управление уязвимостью** (рис. 6.24) можно изменить уровень критичности и статус, добавить комментарий к вхождению и посмотреть оставленные ранее комментарии.

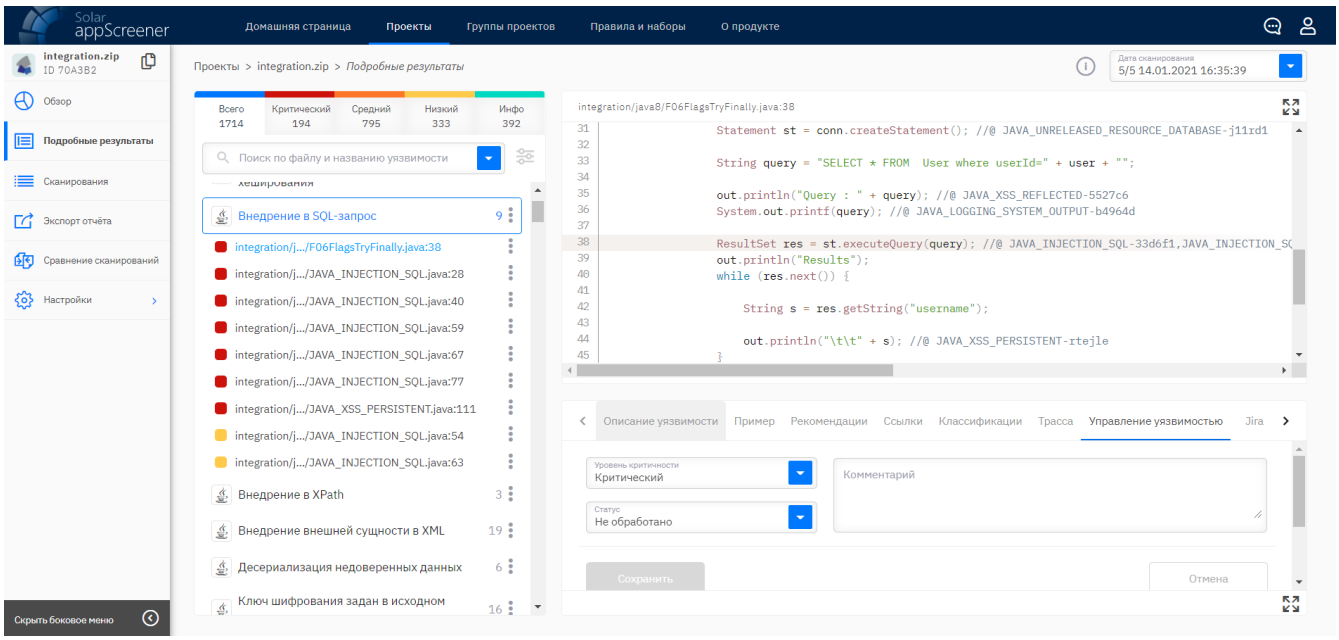


Рис. 6.24: Управление уязвимостью

На вкладке также отображается значение параметра **Confidence** для вхождения. Параметр показывает уверенность модуля Fuzzy Logic Engine, что данная уязвимость не является ложным срабатыванием. Чем больше значение параметра **Confidence**, тем меньше вероятность, что вхождение ложно-положительное.

6.2.3. Сканирования

Раздел **Сканирования** (рис. 6.25) предназначен для управления сканированиями в рамках одного проекта. Каждое сканирование имеет свой идентификатор и отметку времени. Для каждого сканирования отображаются следующие данные:

- дата и время сканирования, при нажатии на иконку ⓘ отображается информация о параметрах запуска анализа;
- меню действий:
  - выгрузить отчёт;
  - архивировать сканирование;
  - удалить сканирование.
- статус сканирования;
- языки, для которых был произведен анализ;
- продолжительность сканирования;
- количество строк кода;
- количество уязвимостей критического, среднего, низкого и информационного уровня;
- рейтинг приложения.

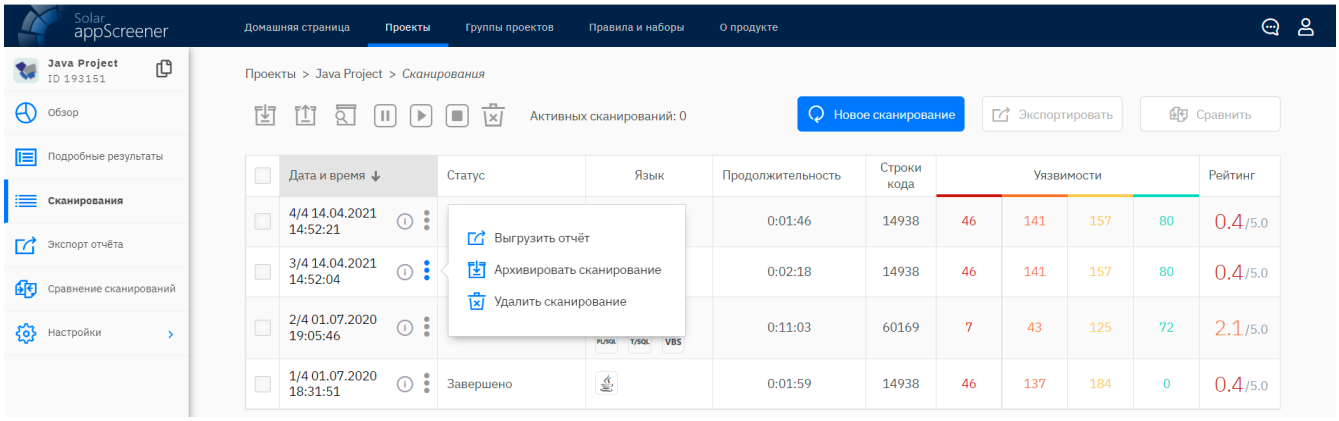


Рис. 6.25: Сканирования

Список можно сортировать по дате сканирования, продолжительности сканирования или рейтингу. Для этого нажмите на соответствующий заголовок, повторное нажатие меняет порядок сортировки.

Сравнить результаты двух выбранных сканирований можно, нажав на кнопку **Сравнить**. Сканирования, которые находятся в архиве, можно скрыть из списка, нажав на **Скрыть архив**, или отображать в списке, нажав на **Показать архив**.

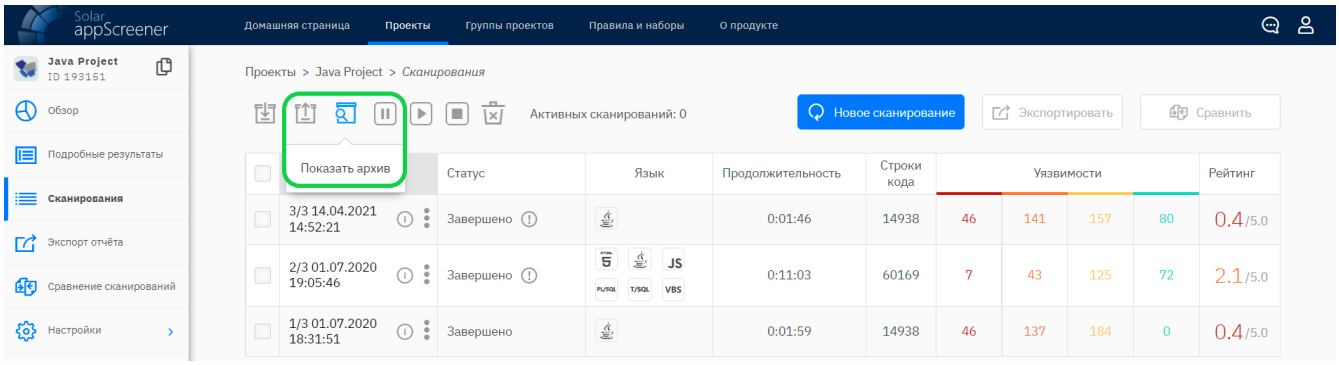


Рис. 6.26: Показать/Скрыть архив

6.2.3.1. Новое сканирование

Кнопка **Новое сканирование** (рис. 6.27) предназначена для проведения повторного сканирования в рамках одного проекта. Подробное описание запуска анализа представлено в разделе Запуск сканирования в UI.

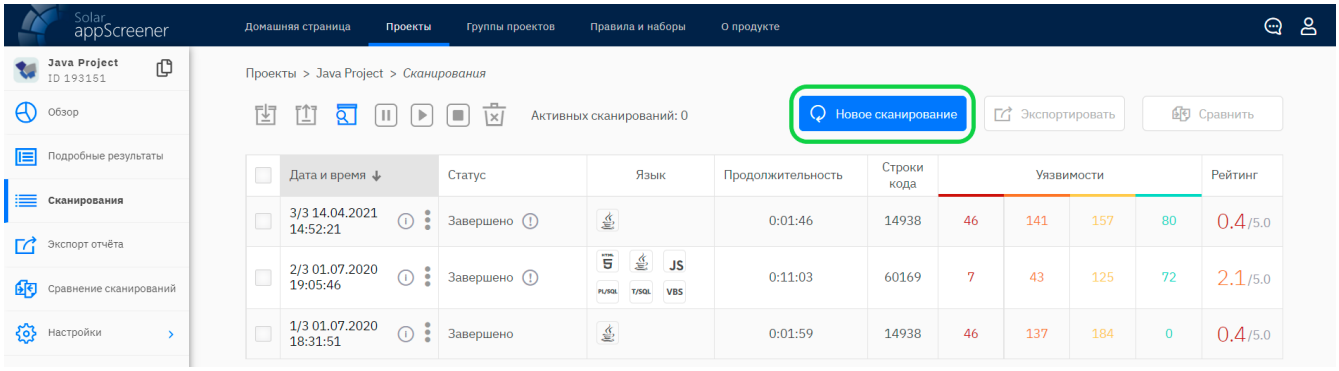


Рис. 6.27: Новое сканирование

6.2.3.2. Очередь сканирований

В appScreener можно запустить сразу несколько сканирований в одном проекте с разными настройками. Отслеживать статусы сканирований можно в графе **Статус**.

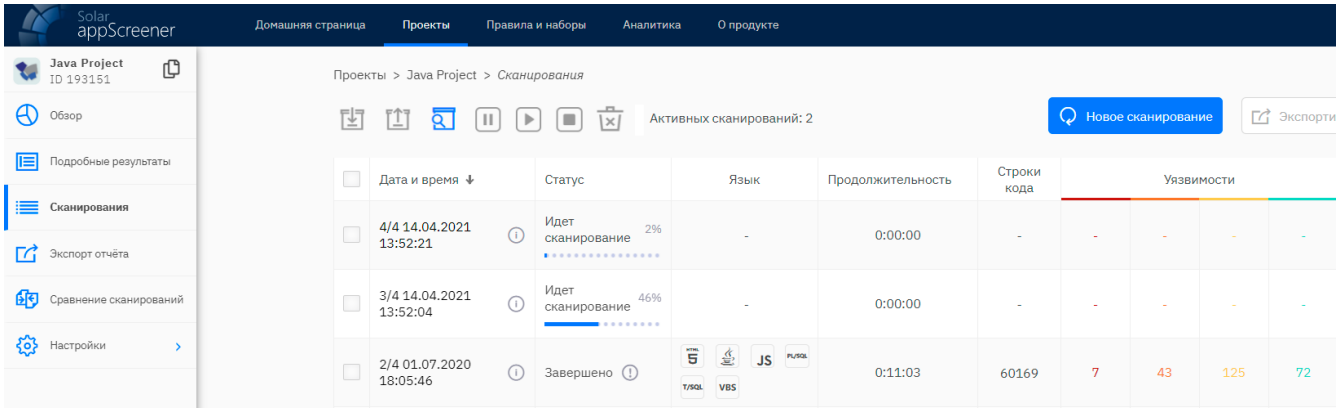


Рис. 6.28: Очередь сканирований

6.2.4. Экспорт отчёта

В разделе **Экспорт отчёта** можно выгрузить результаты сканирования в отчёт в формате PDF, CSV, DOCX или SARIF. Выберите один из готовых шаблонов настроек или задайте информацию для экспорта вручную.

Настройки отчёта включают следующие блоки:

- сканирования;
- сравнить со сканированием;
- информация о проекте;
- классификация уязвимостей;
- фильтр уязвимостей;
- список уязвимостей;
- подробные результаты;
- инструкции по настройке WAF;

- общие настройки отчёта.

6.2.4.1. Сканирования

Для экспорта отчёта выберите одно или несколько сканирований. Чтобы получить только сводную информацию по проекту, удалите все сканирования из списка.

6.2.4.2. Сравнить со сканированием

Выберите одно сканирование, чтобы опция **Сравнить со сканированием** стала доступна. В отчёт будут включены таблица сравнения, график и статистика по новым, сохранившимся и устранённым уязвимостям.

Выберите статусы уязвимостей (новые, сохранившиеся и/или устранённые) и укажите количество вхождений каждой уязвимости.

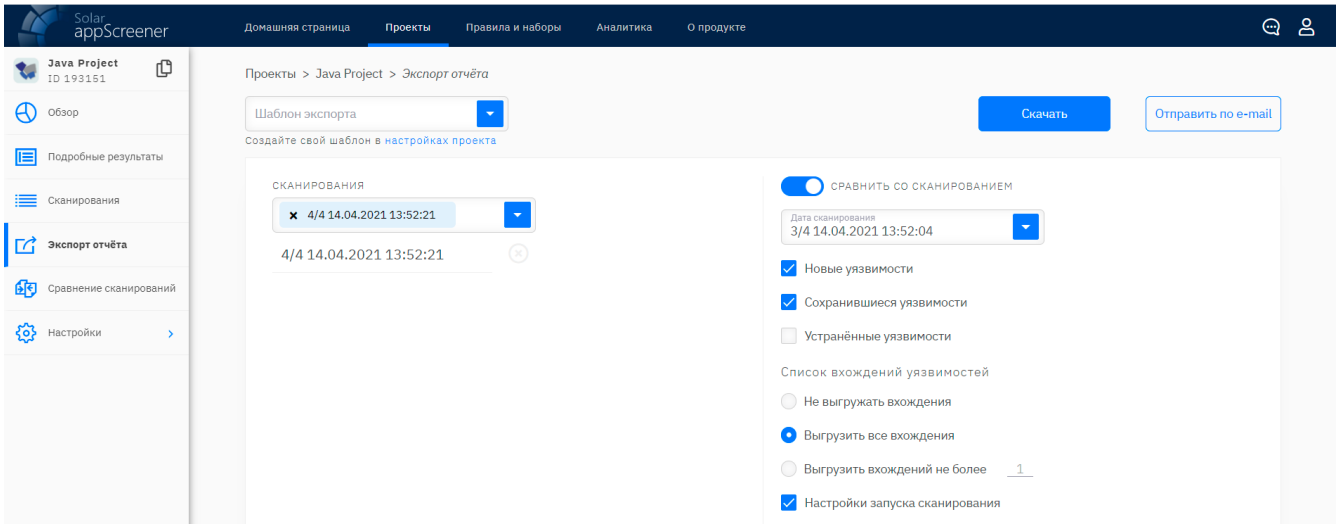


Рис. 6.29: Сравнить со сканированием

6.2.4.3. Информация о проекте

В отчёт можно включить динамику уровня безопасности, динамику количества уязвимостей и историю сканирований.

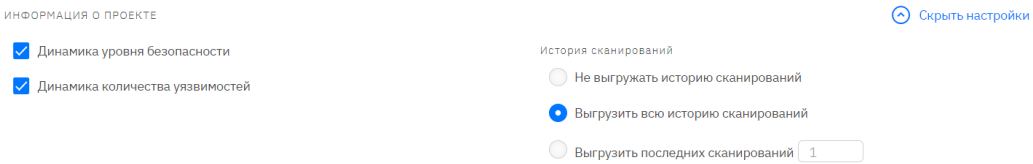


Рис. 6.30: Информация о проекте

6.2.4.4. Классификация уязвимостей

Выберите один из способов классификации уязвимостей: По критичности, OWASP Top 10 2013, OWASP Mobile Top 2014, OWASP Mobile Top 2014, OWASP Top 10 2017, OWASP

Top 10 2021, OWASP MASVS, PCI DSS 4.0, HIPAA, CWE/SANS Top 25 2011, CWE/SANS Top 25 2021, БДУ ФСТЭК (доступен, если выбран русский язык отчёта) или ОУД4 (доступен, если выбран русский язык отчёта).

При выборе **Сравнить со сканированием** доступна только классификация уязвимостей **По критичности**.

Рис. 6.31: Классификация уязвимостей

6.2.4.5. Информация о сканировании

По умолчанию будет добавлена статистика сканирования: статус, рейтинг, продолжительность, количество строк кода, количество уязвимостей.

Выберите дополнительную информацию о сканировании:

- диаграмма найденных уязвимостей;
- диаграмма типов уязвимостей;
- статистика по языкам;
- информация об ошибках сканирования;
- настройки запуска сканирования.

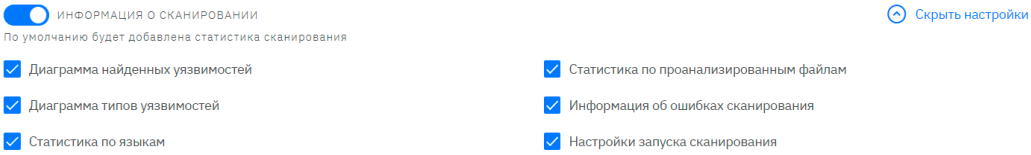


Рис. 6.32: Информация о сканировании

6.2.4.6. Фильтр уязвимостей

Выберите уязвимости по следующим параметрам: уровень критичности, типы уязвимостей, язык. Воспользуйтесь фильтром **Fuzzy Logic Engine** в одном из режимов: **Только истинные**, **Только важные**, **Пользовательский** или **Динамический** режимы. Подробнее о Fuzzy Logic Engine см. Подробные результаты

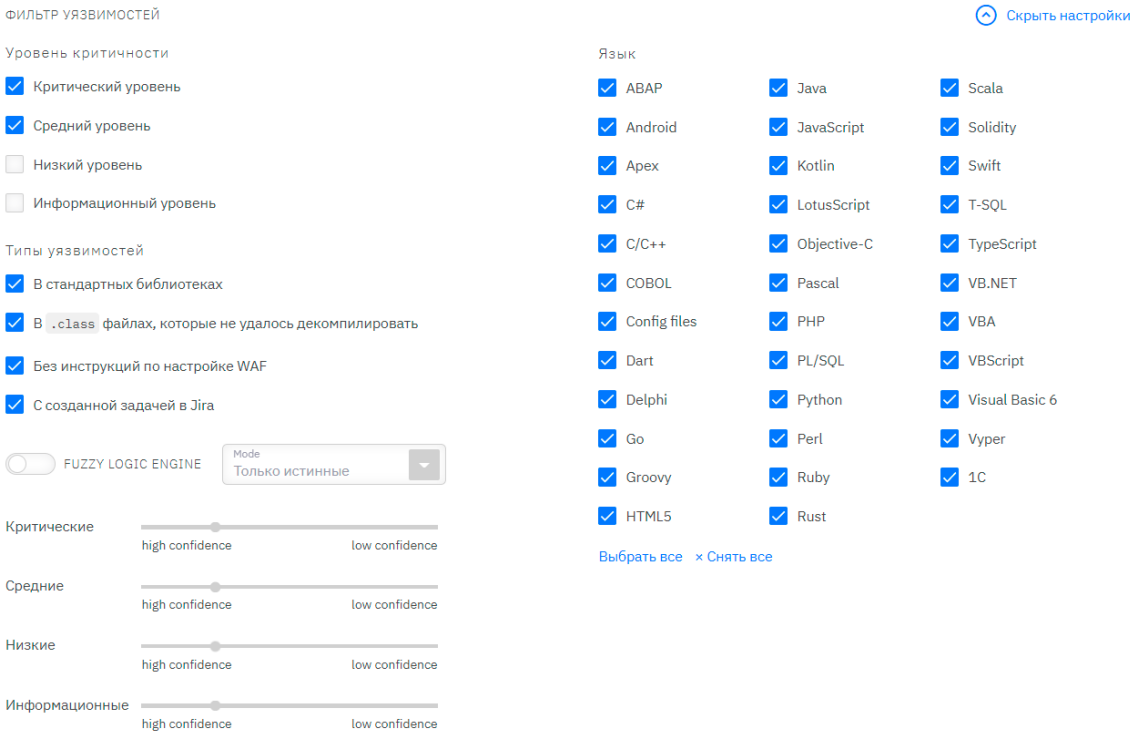


Рис. 6.33: Фильтр уязвимостей

6.2.4.7. Список уязвимостей

Выберите статусы уязвимостей и задайте количество их вхождений.

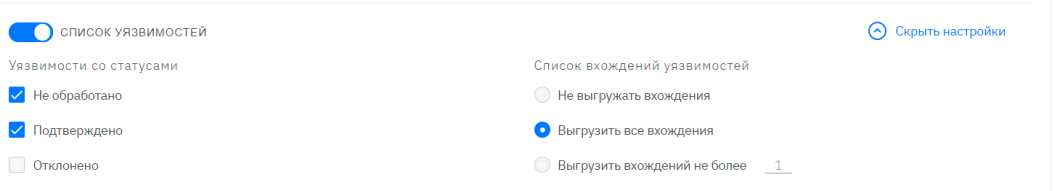


Рис. 6.34: Список уязвимостей

6.2.4.8. Подробные результаты

По умолчанию для уязвимостей будут добавлены описание, пример, рекомендации по устранению, ссылки. Дополнительно можно настроить:

- статусы уязвимостей: **Не обработано**, **Подтверждено**, **Отклонено** (подробнее в разделе Подробные результаты);
- количество вхождений уязвимостей;
- включение комментариев;
- отображение трассы;
- размер контекста исходного кода с уязвимостью.

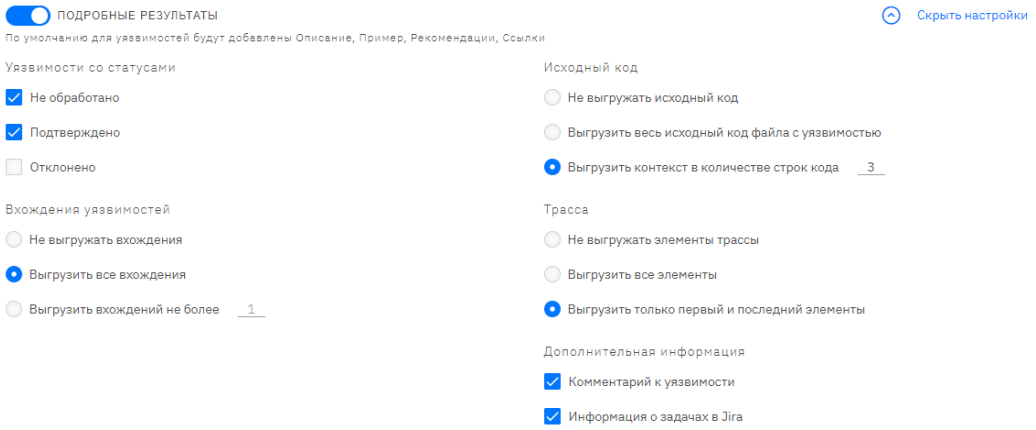


Рис. 6.35: Подробные результаты

6.2.4.9. Инструкции по настройке WAF

Укажите статусы уязвимостей, для которых будут добавлены инструкции по настройке WAF:

- **Imperva SecureSphere**;
- **ModSecurity**;
- **F5**
- **PT AF**.

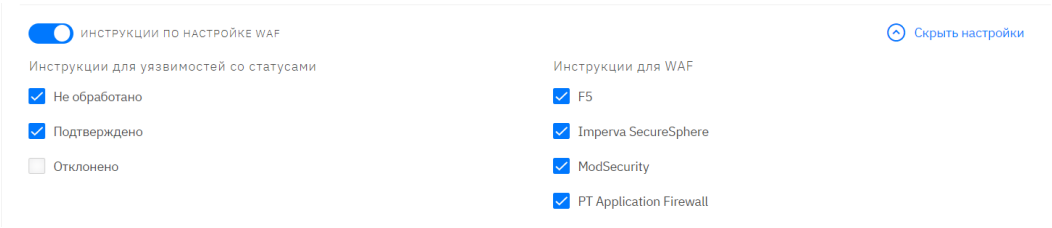


Рис. 6.36: Инструкции по настройке WAF

6.2.4.10. Общие настройки отчёта

Выберите язык, формат отчёта и при необходимости включите в него настройки экспорта и оглавление. Также можно настроить отображение статусов уязвимостей в отчёте и установить пользовательский логотип.

Обратите внимание:

Для корректного отображения данных CSV-отчёта в **Microsoft Excel** необходимо вручную выбрать в выпадающем списке **Обнаружение типов данных** опцию **Не обнаруживать типы данных** во время импорта файла. Настройка отображения статусов уязвимостей недоступна для этого формата.

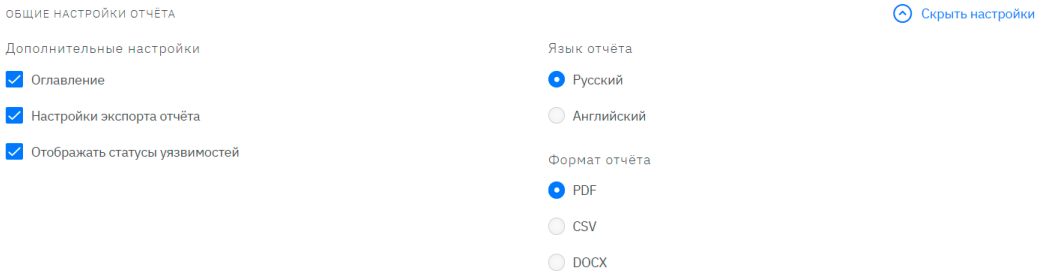


Рис. 6.37: Общие настройки отчёта

Чтобы скачать отчёт, нажмите **Скачать**.

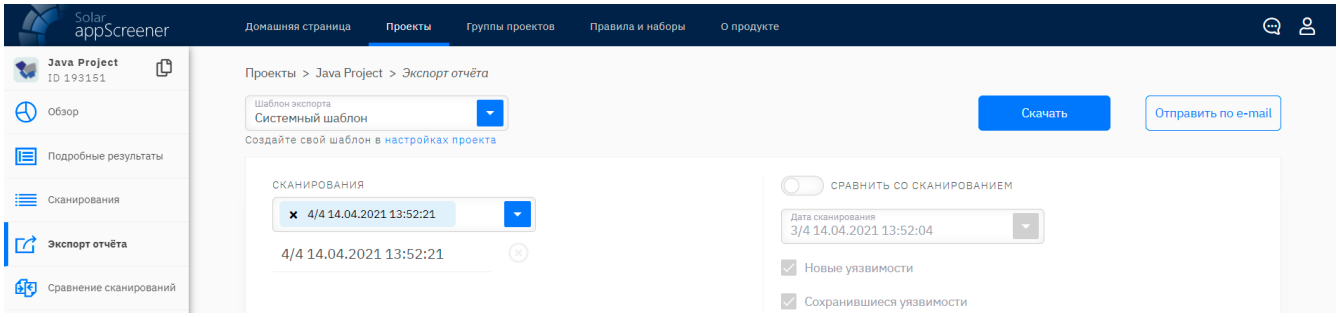


Рис. 6.38: Экпорт

Чтобы отправить отчёт по почте, нажмите **Отправить по e-mail**. В открывшейся форме укажите список адресов получателей и при необходимости отредактируйте текст письма.

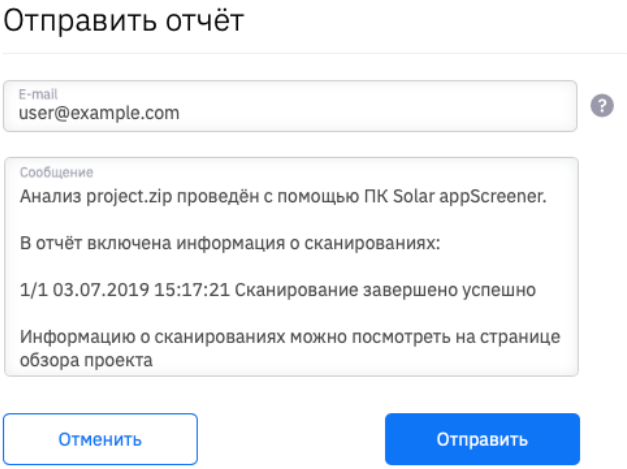


Рис. 6.39: Отправить отчёт по e-mail

6.2.5. Сравнение сканирований

В разделе **Сравнение сканирований** можно производить сравнение результатов сканирований(рис. 6.40). Чтобы сравнить результаты, выберите в верхней части страницы два сканирования. На странице отобразится количество устраненных, новых и сохранившихся уязвимостей на графике и в таблице, а также будет представлена

таблица со сравнением по дате сканирования, продолжительности, языкам, количеству строк кода, количеству уязвимостей с учётом уровня критичности и рейтингу.

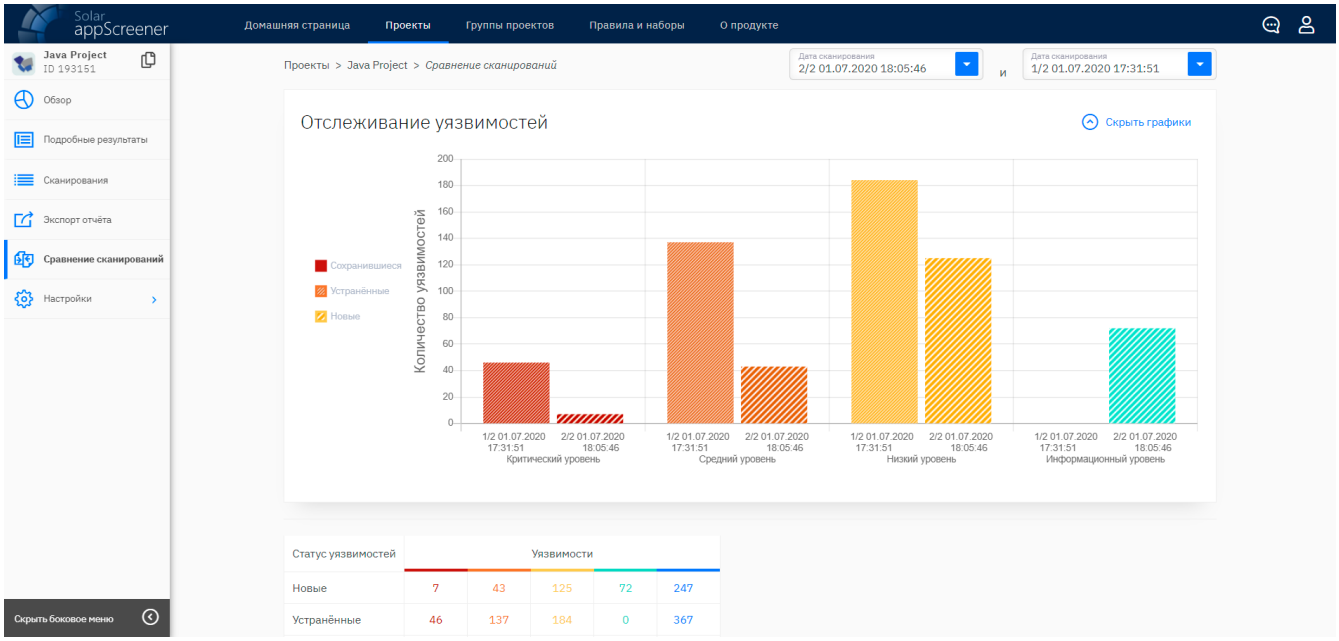


Рис. 6.40: Сравнение сканирований

6.2.6. Настройки

В разделе **Настройки** отображаются настройки проекта. В этом разделе можно переключаться между подпунктами **Общие**, **Экспорт отчёта**, **Права пользователей**, **Jira**, **Автоматическое сканирование** и **Управление проектом**.

6.2.6.1. Общие

В подпункте **Общие** (рис. 6.41) можно задать настройки для последующих сканирований:

- указать источник кода — ссылка на репозиторий или на Google Play/App Store. Чтобы загрузить содержимое из закрытого репозитория, укажите имя пользователя и пароль в блоке **Настройки приватного репозитория**;
- назначить приоритет сканирования;
- выбрать языки, которые необходимо включить в анализ;
- включить анализ библиотек и указать, необходима ли сборка проекта (для приложений Java, Scala, Kotlin и Android);
- выбрать среду сборки проекта (для C/C++ приложений);
- отметить, проводить ли межмодульный анализ проекта (для C/C++ приложений);
- включить анализ стандартных библиотек (для JavaScript приложений);
- использовать ли пользовательские паттерны (подробнее в разделе Правила и наборы);
- отметить, анализировать ли файлы конфигурации;

- отметить, произвести инкрементальный анализ или полный (для ABAP, Apex, C#, COBOL, Config files, Dart, Delphi, GO, Groovy, HTML5, JavaScript, LotusScript, PHP, PL/SQL, Python, Perl, Ruby, Rust, Solidity, T-SQL, TypeScript, VB.NET, VBA, VBScript, Visual Basic 6, Vyper или 1С-приложений). При инкрементальном анализе сканируются только изменённые или новые файлы, что сокращает время анализа при пересканировании. При изменении настроек проекта или обновлении системы рекомендуется произвести полный анализ;
- указать директории, файлы и/или пакеты, которые следует включить/исключить из анализа;
- указать ветку в репозитории Git;
- отметить, сохранять ли файл, загруженный на анализ. Файл будет доступен администратору системы;
- ввести имя пользователя и пароль закрытого репозитория, которые будут использованы для анализа;
- выбрать кодировку для исходного кода проекта;
- выбрать кодировку для названий файлов проекта;
- отметить наборы правил, которые следует применять при сканировании данного приложения. Для этого выберите язык, и для каждого отдельного языка укажите требуемые наборы правил.

Также можно выбрать шаблон настроек для запуска сканирования из списка доступных шаблонов. Подробнее о шаблонах сканирования в разделе Настройки. По умолчанию для запуска анализа используется системный шаблон.

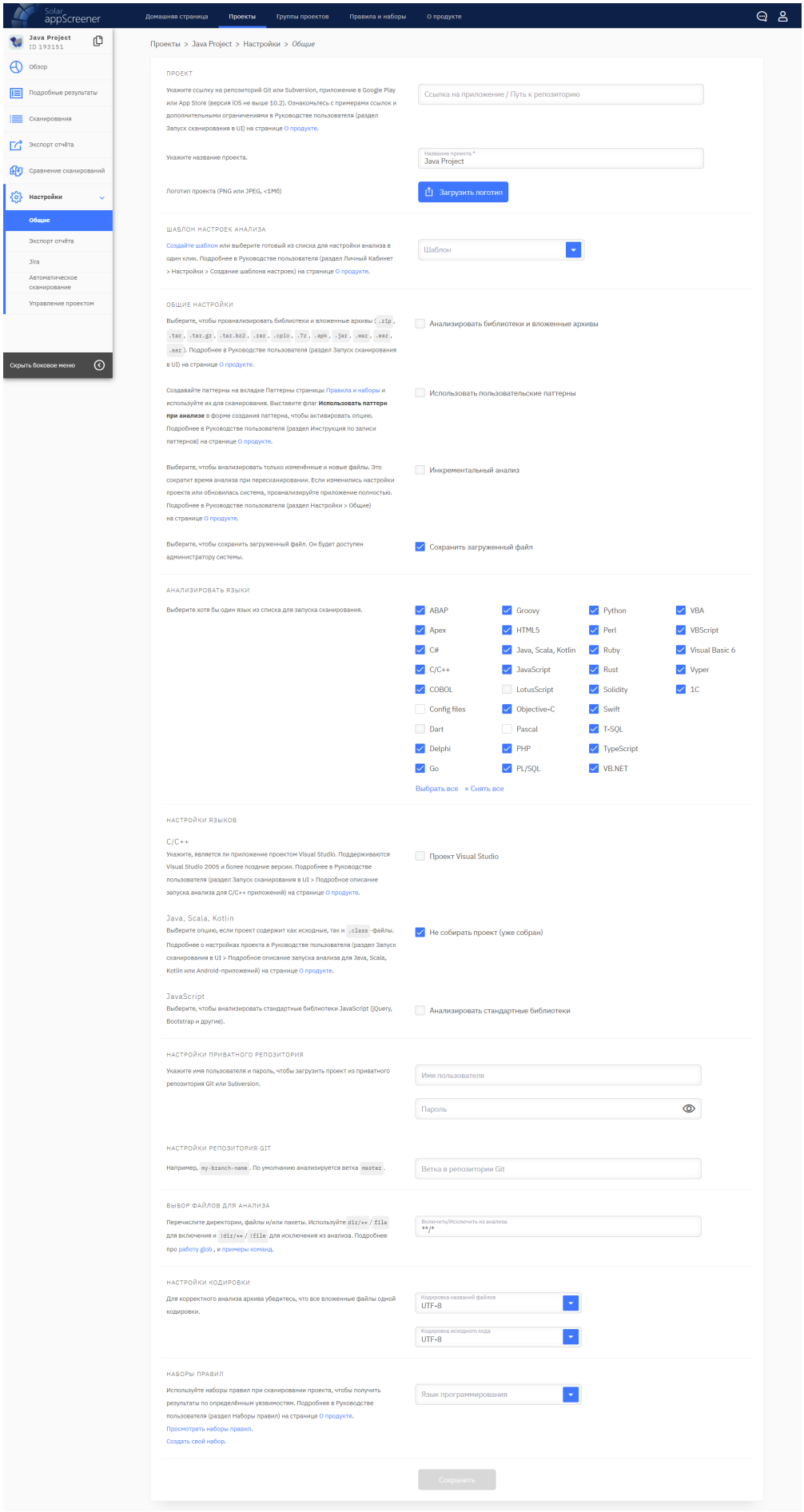


Рис. 6.41: Общие

6.2.6.1.1. Исключение файлов из анализа

Для исключения файлов из анализа используются регулярные выражения. При записи используются следующие символы:

```
** - 0+ директорий
* - 0+ символов
? - 1 символ
```

При анализе архива необходимо учитывать, что архив является корневой папкой, поэтому в таком случае необходимо в начало добавить `**`. Чтобы исключить файл из анализа, указатель на него нужно добавить к этому выражению, например `**/*; !**/*.spec.ts` исключит из анализа все `.spec.ts`. Для обработки только `.ts`, но исключения `.module.ts` можно указать: `**/*.ts; !**/*.module.ts`.

Каждое отдельное выражение отделяется `;`.

Регулярные выражения в примерах:

```
project/
├─ main/
│  ├─ src/
│  │  ├─ config/
│  │  │  ├─ somefile.some.ext
│  │  │  ├─ somefile.other.ext
│  │  ├─ somefile.ext
├─ test/
│  ├─ sometestfile.ext
├─ somefile.txt
```

`**/*` - проверять всё

```
**/main/* - проверять только папку main, проверяемые файлы:
project/main/src/config/somefile.some.ext
project/main/src/config/somefile.other.ext
project/main/src/somefile.ext
```

`**/*; !**/*.ext` - проверять все файлы, кроме .ext, проверяемые файлы: project/somefile.txt

```
**/*.ext; !**/*.some.ext - проверять все .ext файлы, кроме .some.ext
(.other.ext проверяются), проверяемые файлы:
project/main/src/config/somefile.other.ext
project/main/src/somefile.ext
project/test/sometestfile.ext
```

`project/somefile.txt`

`**/*.other.ext` - проверять только .other.ext файлы, проверяемые файлы: project/main/src/config/somefile.other.ext

`**/*; !**/sometestfile.ext - проверять всё, кроме sometestfile.ext,`

`проверяемые файлы:`

`project/main/src/config/somefile.some.ext`

`project/main/src/config/somefile.other.ext`

`project/main/src/somefile.ext`

`project/somefile.txt`

`!**/*` - не проверять ничего

6.2.6.2. Права пользователей

В подразделе **Права пользователей** можно быстро выдать доступ к проекту другим пользователям системы и настроить их права в проекте. Чтобы настроить права конкретного пользователя, кликните по его логину в списке.

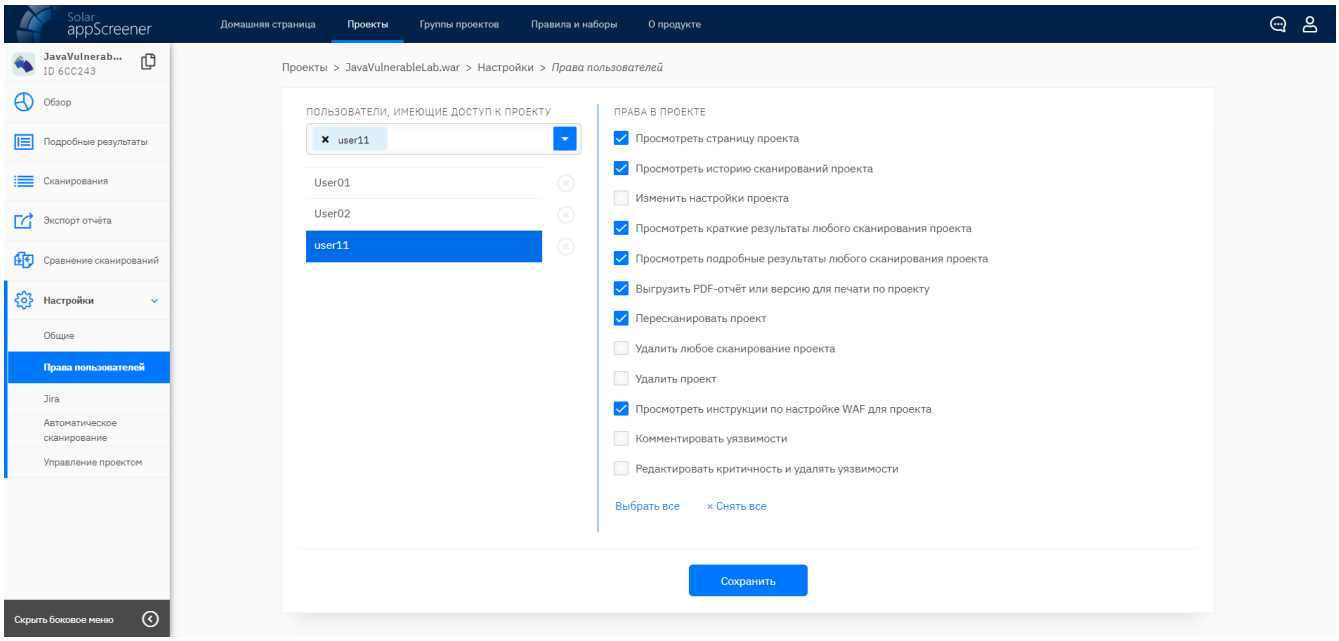


Рис. 6.42: Права пользователей

6.2.6.3. Jira

Во вкладке **Jira** можно привязывать проекты в **Jira** к проекту appScreener (подробнее см. раздел Как привязать проект в Solar appScreener к проекту в Jira).

6.2.6.4. Автоматическое сканирование

Для удобной работы с долгосрочными проектами или приложениями из магазинов можно настроить автоматическое сканирование по событию или расписанию.

Интеграция с VCS хостингом позволит автоматически запускать сканирование проекта из VCS хостинга (система поддерживает интеграции с **GitHub**, **GitLab** и **Bitbucket**). Вы можете настроить запуск анализа по расписанию, а также событиям **Push** и **Tag**. Более подробную информацию об использовании можно найти в разделе VCS хостинги.

Для приложений из магазинов Google Play и AppStore можно настроить автоматический запуск сканирования по пользовательскому расписанию или при обновлении версии.

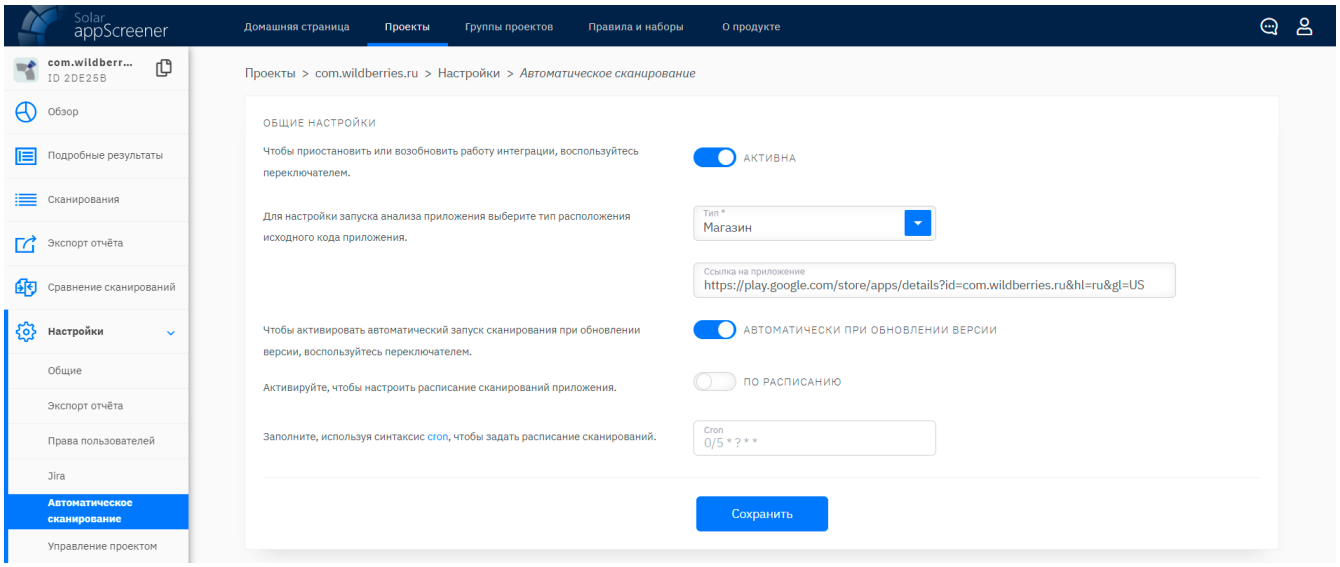


Рис. 6.43: Автоматическое сканирование при обновлении версии

6.2.6.5. Управление проектом

В подразделе **Управление проектом** можно:

- изменить название проекта;
- загрузить логотип проекта;
- добавить/извлечь из архива;
- удалить проект.

Для добавления проекта в архив нажмите **Архивировать проект**. Подробнее об архивации (см. раздел Проекты). Чтобы удалить проект без возможности восстановления, нажмите **Удалить проект** и подтвердите действие.

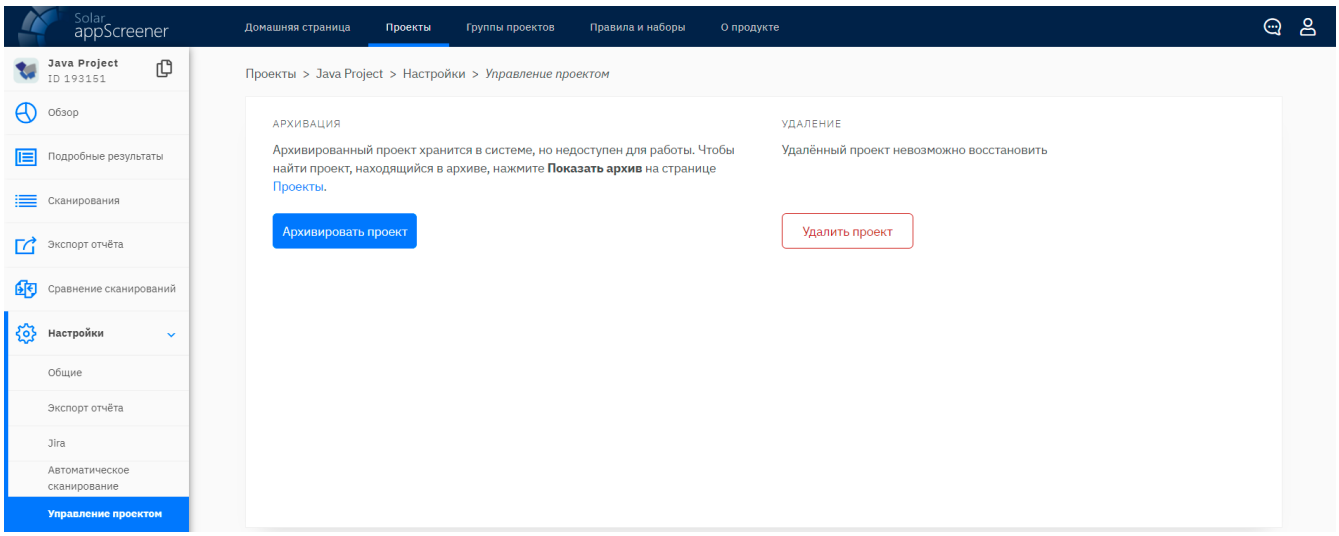


Рис. 6.44: Управление проектом

6.3. Работа с API

Доступ к функциональности appScreener также доступен через API. Web-интерфейс спецификации API реализован с помощью Swagger Codegen. Для доступа к API:

1. Перейдите в раздел **Личный кабинет** > **Настройки доступа** > **Токен и пароль**.
2. Нажмите **Создать токен**.
3. Введите пароль учётной записи.
4. Укажите время действия токена.
5. Нажмите **Получить активный токен** и скопируйте значение в буфер.
6. Нажмите **Спецификация API**.
7. На открывшейся странице вставьте значение токена в поле **Enter token here**.
8. Нажмите **Explore**.

6.3.1. Запуск сканирования

Чтобы запустить сканирование из API:

1. Кликните по кнопке **/scan/start** в разделе **Scan**.
2. Нажмите **Try it out**.
3. В теле запроса выберите файл для анализа или укажите ссылку на проект и добавьте настройки сканирования.
4. Нажмите **Execute**.

Пример метода для запуска сканирования из **приватного репозитория**:

```
curl -k -X POST "https://YOUR_SERVER/app/api/v1/scan/start" -H "accept:
application/json" -H "Authorization: Bearer TOKEN" -H "Content-Type:
multipart/form-data" -F "branch=BRANCH" -F "analyzeJsLibs=" -F "ruleSet=" -F
"checkboxNoBuild=" -F "name=" -F "repoPassword=PASSWORD" -F
"saveRepoCredentials=" -F "sourceEncoding=" -F "checkboxUseUserPatterns=" -F
"visualStudio=" -F "checkboxAnalyzeLibs=" -F "repoLogin=LOGIN" -F "saveFile=" -F
```

```
"incremental=" -F "languages=" -F "fileSelector=**/*" -F "uuid=" -F
"link=WEB_URL" -F "nameEncoding=" -F "preset="
```

Пример метода для запуска сканирования **архива с локального компьютера**:

```
curl -k -X POST "https://YOUR_SERVER/app/api/v1/scan/start" -H "accept:
application/json" -H "Authorization: Bearer TOKEN" -H "Content-Type:
multipart/form-data" -F "branch=" -F "analyzeJsLibs=" -F "ruleSet=" -F
"checkboxNoBuild=" -F "name=" -F "repoPassword=" -F "saveRepoCredentials=true"
-F "sourceEncoding=" -F "checkboxUseUserPatterns=" -F "visualStudio=" -F
"checkboxAnalyzeLibs=" -F "repoLogin=" -F "saveFile=" -F "incremental=" -F
"link=" -F "languages=" -F "fileSelector=**/*" -F "uuid=" -F
"file=PATH_TO_THE_FILE;type=application/x-msdownload" -F "nameEncoding=" -F
"preset="
```

Получить **статус сканирования**:

```
curl -k -X GET "https://YOUR_SERVER/app/api/v1/scans/SCAN_UUID/compact" -H
"accept: application/json" -H "Authorization: Bearer TOKEN"
```

6.3.2. Выгрузка отчёта

Чтобы выгрузить отчёт сканирования проекта через API:

1. Перейдите в раздел **Report**.
2. Выберите способ получения отчёта: скачать (**/report/file/**) или получить на почту (**/report/email/**).
3. Нажмите **Try it out**.
4. В теле запроса введите UUID нужного проекта и сканирования и укажите настройки отчёта.
5. Нажмите **Execute**.

Получить **отчёт в формате PDF**:

```
curl -k -X POST "https://YOUR_SERVER/app/api/v1/report/file" -H "accept:
application/octet-stream" -H "Authorization: Bearer TOKEN" -H "Content-Type:
application/json" -d "{\"projectUuid\":\"PROJECT_UUID\",\"scanUuids\":
[\"SCAN_UUID\"],\"exportSettings\":{\"uuid\":\"string\",\"projectInfoSettings\":
{\"securityLevelDynamics\":true,\"vulnNumberDynamics\":true,\"scanHistory\":0},\
"sort\":\"CR\",\"scanInfoSettings\":
{\"included\":true,\"foundVulnChart\":true,\"typeVulnChart\":true,\"langStats\":
true,\"fileStats\":true,\"scanErrorInfo\":true,\"scanSettings\":true},\
"filterSettings\":
```

{\"critical\":true,\"medium\":true,\"low\":true,\"info\":true,\"standardLibs\":
true,\"classFiles\":true,\"waf\":true,\"jira\":true,\"fuzzy\":
{\"included\":true,\"critical\":0,\"medium\":0,\"low\":0,\"info\":0,\
"percentile\":0,\"mode\":\"TRUE\"},\"lang\":\"ru\"},\"tableSettings\":
{\"included\":true,\"entriesSettings\":
{\"notProcessed\":true,\"confirmed\":true,\"rejected\":true},\"entryNum\":0},\
"detailedResultsSettings\":{\"included\":true,\"entriesSettings\":
{\"notProcessed\":true,\"confirmed\":true,\"rejected\":true},\"entryNum\":0,\
"comment\":true,\"jiraInfo\":true,\"traceNum\":0,\"sourceCodeNum\":0},\
"wafSettings\":{\"included\":true,\"entriesSettings\":
{\"notProcessed\":true,\"confirmed\":true,\"rejected\":true},\"imperva\":true,\"
mod\":true,\"f5\":true},\"generalSettings\":
{\"reportSettings\":true,\"contents\":true,\"locale\":\"ru\",\"format\":\"PDF\"}
},\"comparisonSettings\":
{\"included\":false,\"scanUuid\":\"string\",\"newIssue\":true,\"saved\":true,\
"fixed\":true,\"entryNum\":0,\"scanSettings\":true}}" > PATH_TO_THE_FILE

6.4. Правила и наборы

Для управления правилами и их наборами перейдите в раздел **Правила и наборы** (рис. 6.45).

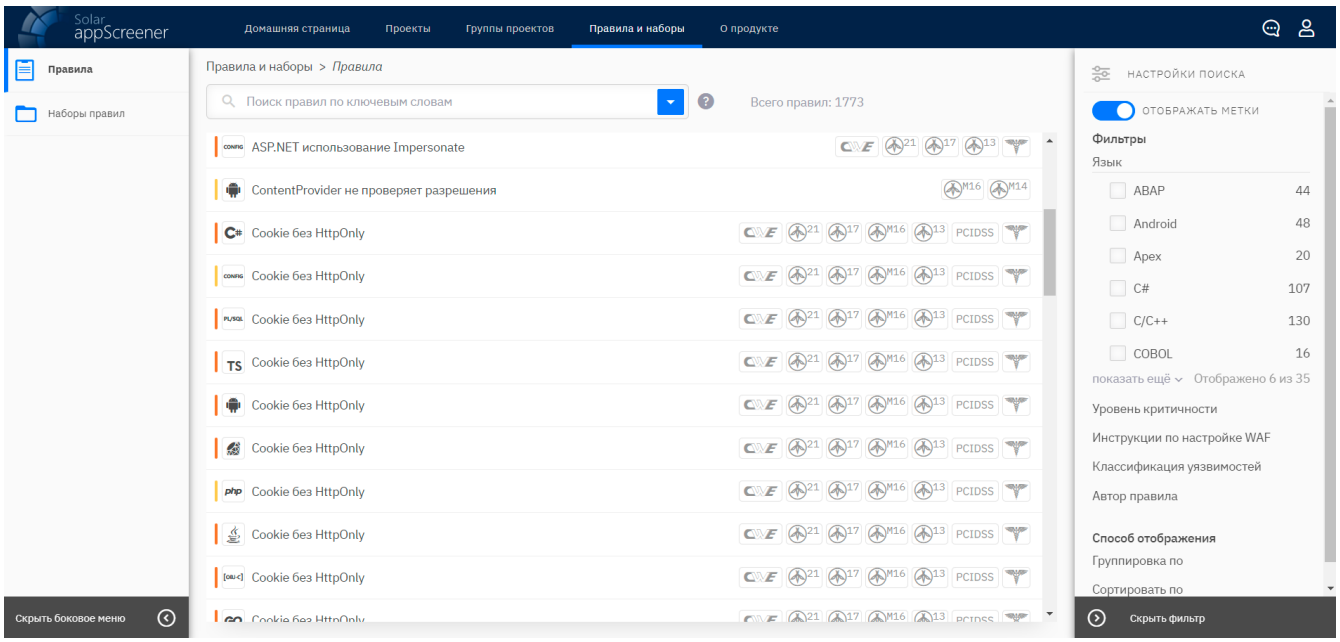


Рис. 6.45: Раздел Правила и наборы

6.4.1. Правила

Для поиска, сортировки и группировки правил воспользуйтесь поиском и фильтрами в правом боковом меню.

С помощью строки поиска можно отбирать правила по следующим значениям:

- название правила;
- автор правила;
- идентификатор правила*;
- название пользовательского паттерна;
- автор пользовательского паттерна в правиле*;
- идентификатор пользовательского паттерна.

* не все правила могут иметь данные значения.

После ввода нажмите **Enter** или выберите подходящее значение из открывшегося списка. После этого система выведет подходящие результаты. Искать правила можно сразу по нескольким значениям.

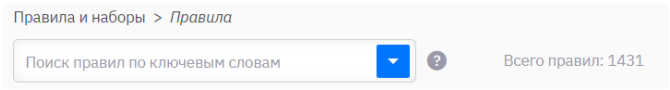


Рис. 6.46: Строка поиска правил

6.4.1.1. Фильтры

При использовании фильтров система отобразит правила, подходящие под выбранные критерии. Фильтровать правила можно по следующим значениям:

- языку;
- критичности;
- наличию инструкций по настройке WAF;
- классификации;
- наличию пользовательских паттернов в правиле;
- автору правила.

Для правил можно выбрать способ их отображения:

- группировать по:
  - не группировать;
  - названию правила;
  - языку;
  - критичности;
- сортировать по:
  - алфавиту;
  - критичности;

- языку;
- количеству правил.

Нажмите на стрелочку слева от параметра, по которому вы хотите отсортировать правила, чтобы сортировать в обратном порядке.

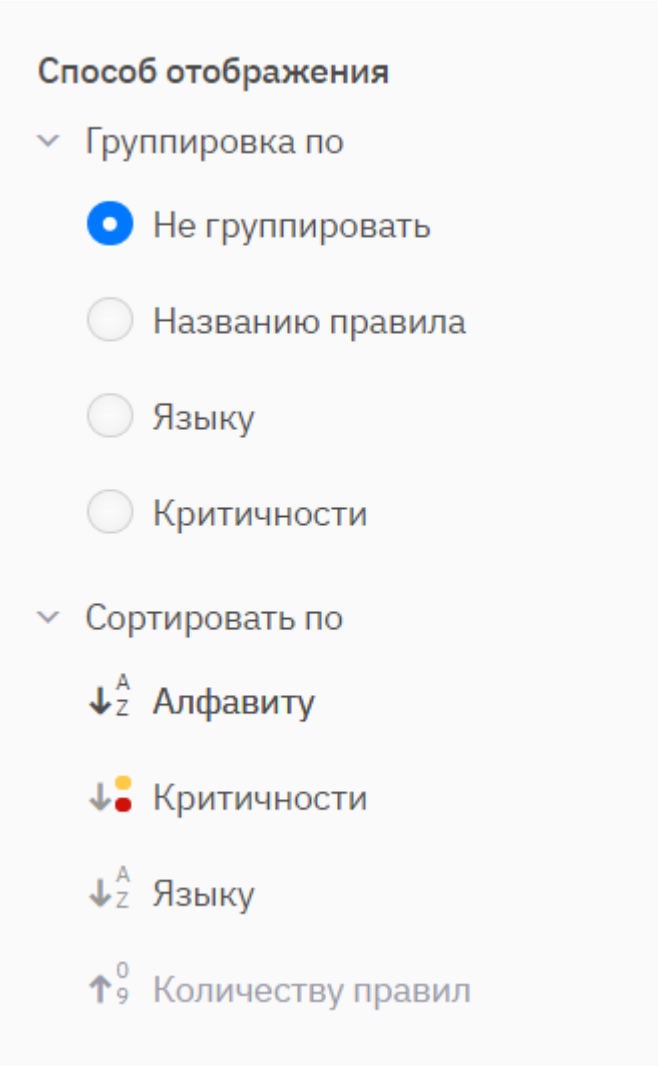


Рис. 6.47: Способ отображения правил

Для установки изначальных значений строки поиска, фильтров, сортировки и группировки нажмите **Сбросить**.

6.4.1.2. Метки

С помощью меток отображаются атрибуты правил:

- **Инструкции по настройке WAF**.

  Solar appScreener предоставляет инструкции по настройке средств: **Imperva SecureSphere**, **ModSecurity**, **F5** и **PT AF**. Каждому из них соответствует своя метка.

- **Классификации, в которые входит уязвимость**.

Уязвимости сопоставляются с пунктами классификаций: CWE, CWE/SANS Top 25 2011, CWE/SANS Top 25 2021, OWASP Top 10 2021, OWASP Top 10 2017, OWASP Mobile Top 10 2016, OWASP Mobile Top 10 2014, OWASP Top 10 2013, OWASP MASVS, PCI DSS, HIPAA и БДУ ФСТЭК. При наведении курсора на метку отображается тултип с идентификатором пункта классификации, соответствующим данной уязвимости.

- **Автор правила и паттерна**.

  Метка **Автор правила и паттерна** отображается для пользовательских правил. При наведении курсора на метку отображается тултип с автором правила и авторами паттернов.

Чтобы скрыть метки, используйте переключатель в правом боковом меню.

6.4.1.3. Информация о правиле

Для просмотра информации о найденном правиле нажмите на него. В информацию о правиле входит:

- описание уязвимости;
- пример;
- рекомендации;
- ссылки — ссылки на информационные ресурсы, связанные с уязвимостью;
- классификации — ссылки на соответствующие пункты в CWE 4.0, БДУ ФСТЭК, CWE/SANS Top 25, OWASP Top 10, OWASP Mobile Top 10, OWASP MASVS, PCI DSS, HIPAA;
- наборы правил — для добавления/исключения правила из набора;
- паттерны — для добавления новых и редактирования существующих паттернов (только для пользовательских правил);
- инструкции по настройке WAF (только для системных правил).

Паттерны определяют условия, при выполнении которых фрагмент кода помечается как уязвимый. Для хранения паттерна разработан собственный универсальный формат XML. Для добавления паттерна на вкладке **Паттерны** нажмите **Создать паттерн**, введите название, уровень критичности, уровень confidence и паттерн XML, затем нажмите **Сохранить** (рис. 6.48). Для того, чтобы паттерн был доступен всем, выберите опцию **Публичный**.

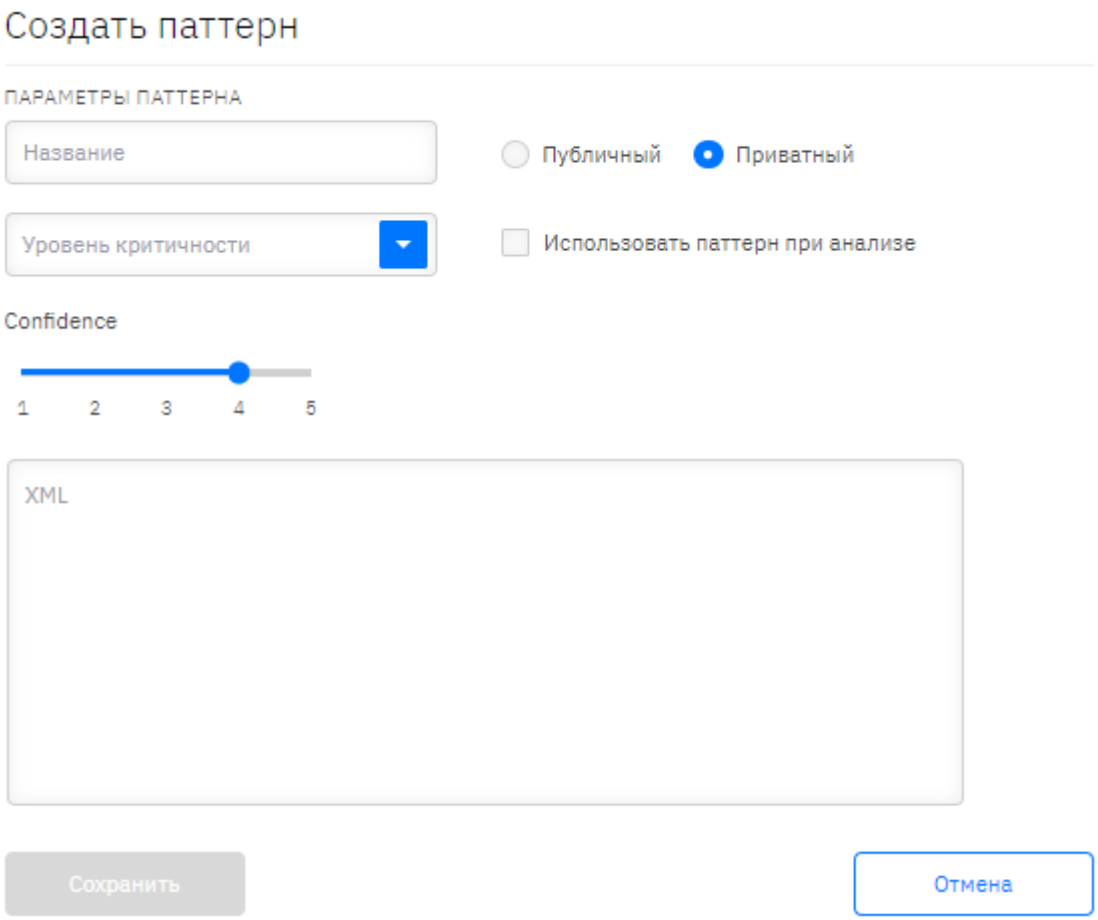


Рис. 6.48: Создание паттерна

6.4.2. Пользовательские правила

В разделе **Правила и наборы** можно работать с пользовательскими правилами для анализа.

Чтобы создать правило, в разделе **Правила** нажмите **Создать правило**. Заполните открывшуюся форму и нажмите **Сохранить**, чтобы добавить правило в систему.

Вы также можете создавать правила на основе существующих. Для этого нажмите кнопку **Создать копию правила** в карточке правила. При создании копии правила переносится информация разделов: **Описание уязвимости**, **Пример**, **Рекомендации**, **Ссылки** и **Классификации**. **Пользовательские паттерны** и **Инструкции по настройке WAF** скопированы не будут. Кнопки **Создать правило** и **Создать копию правила** не отображаются у пользователей с ограничениями учётной записи (ознакомиться с ними можно на вкладке **Ограничения** раздела **Профиль**).

Чтобы изменить информацию о правиле или удалить правило, нажмите **Редактировать** в карточке правила. Системные правила недоступны для редактирования. Однако вы можете создать пользовательское правило на основе копии системного правила и редактировать его.

6.4.3. Инструкция по записи паттернов

В Solar appScreener реализован анализ потока данных. Для анализа приложений на Java, Scala, Kotlin в качестве промежуточного представления используется байт-код Java. При анализе приложений на других языках промежуточным представлением является абстрактное синтаксическое дерево (AST). Паттерны формулируются в терминах XPath-запросов к этому представлению. Для хранения паттернов в Solar appScreener разработан собственный универсальный формат XML.

Анализ потока данных предусматривает присваивание сущностям (переменным) флагов и отслеживание их изменений при прохождении данных через программу. Паттерны разделены на три категории: источники (**sources**), прохождения (**passthrough**) и стоки (**sinks**).

Source паттерны показывают, вызовы каких методов сопряжены с постановкой флагов.

Passthrough паттерны показывают, как флаги должны меняться (например, метод, очищающий структуру данных, должен снимать с неё все флаги).

Sink паттерны показывают, какие программные конструкции при каких условиях небезопасны.

6.4.3.1. Структура XML-документа

Структура XML-документа, содержащего паттерн, должна включать следующие элементы:

- Корневой элемент **Rules** содержит несколько элементов **Rule**.
- Элемент **Rule** (правило) задаёт совокупность методов поиска уязвимостей определённого типа. Каждый элемент **Rule** содержит элементы **RuleId** и **Patterns**.
- Элемент **RuleId** содержит уникальный, отражающий смысл правила, идентификатор, например, **CRYPTO_KEY_SIZE** (недостаточный размер ключа шифрования).
- Элемент **Patterns** содержит один или несколько элементов **Pattern**.
- Элемент **Pattern** (паттерн) описывает метод поиска уязвимости для конкретного подмножества случаев. Например, если несколько криптографических библиотек реализуют методы шифрования, то каждый паттерн, входящий в правило **CRYPTO_KEY_SIZE**, задаёт метод поиска данной уязвимости для одной библиотеки или нескольких близких библиотек. Для удобства отладки каждый элемент **Pattern** имеет атрибут **patternId** — уникальный идентификатор паттерна. Идентификатор генерируется автоматически после создания паттерна.

  Элемент **Pattern** содержит подэлементы одного из двух наборов:

  - **Categories**, **Severity**, **Definitions**, **Condition**, **ChangeFlags**
  - **ReportValues**, **Categories**, **Severity**, **XPath**
- Элемент **Categories** содержит один или несколько элементов **Category**.
- Элемент **Category** определяет, к какому классу приложений относится данный паттерн, и принимает одно из значений: **JavaWeb** (веб-приложения на Java, Scala или Kotlin), **Android** (мобильные приложения), **PASCAL**, **PHP**, **CS**, **JS**, **LotusScript**, **TS**, **VBS**,

HTML5, **Python**, **Perl**, **CCPP**, **ObjectiveC**, **Swift**, **PLSQL**, **TSQL**, **ABAP**, **APEX**, **GO**, **RUBY**, **RUST**, **Groovy**, **Dart**, **DELPHI**, **VB6**, **Solidity**, **Vyper**, **COBOL**. Паттерны, относящиеся только к одному классу приложений, не применяются при анализе приложений другого класса, что сокращает время анализа.

- Элемент **Severity** отражает критичность уязвимости по шкале от 1 (низкий) до 3 (высокий). В одно правило могут входить паттерны с разным уровнем критичности.
- Элемент **Definitions** определяет сущности, участвующие в формулировке паттерна. Такими сущностями могут быть: класс (элемент **Class**), метод (элемент **Method**), вызов метода с конкретными значениями аргументов и флагами (элемент **Instruction**). Для класса и метода можно задать имя (как точным значением, так и регулярным выражением), для метода — также сигнатуру (количество и типы параметров, в том числе, неточно, посредством элемента WildCard), для инструкции — фактические значения аргументов и флагов.
- Элемент **Condition** определяет, собственно, небезопасную языковую конструкцию. Как правило, такая конструкция задаётся в терминах сущностей, определённых в **Definitions**. Элемент **Condition** содержит один или несколько элементов из списка: **ClassCondition**, **MethodCondition**, **InstructionCondition**, **CatchBlockCondition**, **VariableCondition**, **ConstantCondition**. Типичный элемент **Condition** для паттерна **sink** содержит единственное условие – вызов инструкции (**InstructionCondition**) со ссылкой (**Ref**) на небезопасный вызов, определённой в **Definitions**.
- Элемент **ChangeFlags** (только для паттернов **source** и **passthrough**) показывает, как нужно поменять флаги, если участок кода удовлетворяет условиям, описанным в **Condition**.
- Элемент **ReportValues** показывает, информацию о каких сущностях следует передать в пользовательский интерфейс. Элемент содержит один или несколько элементов **ReportValue**. Каждый **ReportValue** содержит атрибут **key** (строка-идентификатор для принимающего модуля) и элемент **Ref** с атрибутом **id**, определяющим сущность, информацию о которой необходимо передать (**id** должен быть определён в **Definitions**). В большинстве случаев передаётся информация о вызванной небезопасной инструкции, но можно передавать и информацию о классе, методе, поле, и т.д.
- Элемент **XPath** используется для записи правил, работающих на промежуточном AST-представлении, и представляет собой запрос к XML-представлению промежуточного представления программного кода.

6.4.3.2. Подробное описание элементов

6.4.3.2.1. Простые элементы

Схема допускает запись информации о многих сущностях: классах, методах, переменных и т.п.

При описании многих из них применяются общие элементы правил:

- Элемент **Name** описывает имя и включает один из подэлементов: **Value**, **Regex**. **Value** содержит точное имя сущности; **Regex** содержит регулярное выражение, определяющее имя сущности.

- Принадлежность сущности типу/классу задаётся элементами **Type** или **Class**. Элемент **Type** содержит строковое значение (имя примитивного типа либо полное имя класса). Элемент **Class** может содержать более сложные условия (см. раздел Описание класса).

6.4.3.2.2. Булева логика

Многие элементы могут содержать логические выражения. К примеру, записать условие «имя метода не совпадает с method» можно так:

```
<Method><Name><Not><Value>Method</Value></Not></Name></Method>
```

Формат записи не ограничивает глубину вложенности логических конструкций, что позволяет формулировать сложные методы поиска уязвимостей. Допускаются логические выражения для следующих элементов: **Constant**, **Parameter**, **Method**, **Field**, **Class**, **Variable**, **Argument**, **Instruction**, **CatchBlock**.

6.4.3.2.3. Идентификаторы и ссылки

Многие элементы могут использоваться с идентификаторами. Идентификатор позволяет определить сущность внутри элемента **Definitions** (определив для неё уникальный в рамках паттерна атрибут **id**) и ссылаться на неё в данном шаблоне с помощью элемента **Ref**. Элемент **Ref** содержит два атрибута: **id** (идентификатор) и **reftype** (тип ссылки; принимает одно из значений: **classref**, **methodref**, **instructionref**, **catchblockref**, **fieldref**, **variableref**, **parameterref**, **argumentref**, **constantref**). Например, задать идентификатор **class1** для класса можно так:

```
<Class id="class1"> ... </Class>
```

Ссылаться на класс в данном паттерне можно так:

```
<Class><Ref id="class1" reftype="classref"/></Class>
```

6.4.3.2.4. Константы

Записывать условия, содержащие константы, можно с помощью элемента **Constant**. Элемент **Constant** содержит один из элементов: **ConstantString**, **ConstantInt**, **ConstantBoolean**, **Null**.

ConstantString содержит один из элементов **Value**, **Regex** (аналогично **Name**).

ConstantInt содержит целочисленную константу либо логическое выражение, содержащее условия на целочисленных константах (допускается булева логика и арифметические условия **Eq** (равно), **Ne** (не равно), **Lt** (меньше), **Gt** (больше), **Le** (меньше либо равно), **Ge** (больше либо равно)).

6.4.3.2.5. Флаги паттернов

6.4.3.2.5.1. Флаги паттернов source и passthrough

Паттерны **source** и **passthrough** указывают, как необходимо изменить флаги при определённых условиях. Эти условия задаются в элементе **ChangeFlags**.

Элемент **ChangeFlags** содержит один или несколько элементов **FlagsSet**. Каждый элемент **FlagsSet** указывает, как изменить флаги для заданного набора сущностей (аргументов).

Элемент **FlagsSet** содержит элементы: **To**, **From**, **FlagsDiff** и необязательный атрибут **clearAll**, принимающий значения **true**, **false**.

Элементы **To** и **From** определяют сущности (либо ссылки на сущности, заданные в **Definitions**), участвующие в процедуре изменения флагов. Сущности, которым можно присваивать флаги: **Argument**, **Range** (см. раздел Описание инструкции в коде). Паттерн **passthrough** содержит оба элемента **To**, **From**, паттерн **source** – только элемент **To**.

Элемент **FlagsDiff** содержит строку, задающую изменение флагов. Изменения перечисляются через запятую и имеют вид:

- **+FLAG** – добавить флаг **FLAG**;
- **-FLAG** – снять флаг **FLAG**, если он есть;
- **!FLAG** – изменить значение флага **FLAG**, если он есть.

Изменение флагов происходит по следующему алгоритму:

- Если **clearAll** равен **true**, снять все флаги с сущностей, заданных в **To**.
- Проставить сущностям, заданным в **To**, флаги сущностей, заданных в **From**.
- Изменить флаги сущностей, заданных в **To**, в соответствии с **FlagsDiff**.

6.4.3.2.5.2. Флаги паттернов sink

Элемент **Flags** в паттернах **sink** задаёт условие на флаги сущностей (обычно аргументов), при выполнении которого конструкция считается небезопасной.

Элемент **Flags** состоит из одного или нескольких элементов **Flag**, каждый из которых задаёт один флаг. Допускаются логические конструкции, например, записать условие «проставлен флаг FLAG1 или не проставлен флаг FLAG2» можно так:

```
<Flags><Or><Flag>FLAG1</Flag><Not><Flag>FLAG2</Flag></Not></Or></Flags>
```

По умолчанию флаги соединяются предикатом «и», например, запись

```
<Flags><Flag>FLAG1</Flag><Flag>FLAG2</Flag></Flags>
```

задаёт условие «проставлен флаг FLAG1 и флаг FLAG2».

6.4.3.2.6. Описание класса

Элемент **Class** содержит следующие элементы:

- **Modifier** обозначает модификаторы класса.
- **Name** задаёт имя класса.
- **Supers** задаёт условие на предков класса. Например, запись вида:

  ```
  <Class includeSelf="true"><Supers><Class>...
  ```

  соответствует условию «среди предков класса есть класс, удовлетворяющий условию». Атрибут **includeSelf** принимает одно из значений **true**, **false** и показывает, следует ли включать сам класс во множество предков.
- **Method** – задаёт условие на метод класса (см. раздел Описание метода).
- **Field** – задаёт условие на поле класса.

6.4.3.2.7. Описание метода

Элемент **Method** содержит следующие элементы:

- **Modifier** обозначает модификаторы метода.
- **Name** задаёт имя метода. Если метод является конструктором, вместо **Name** применяется специальный пустой элемент **Constructor**.
- **Parameters** определяет условие на набор формальных параметров класса и включает один или несколько элементов **Parameter**, **WildCard**, **Range**.
- **Parameter** содержит элемент **Type** или **Class** и определяет наличие у метода параметра данного типа на текущей позиции.
- **WildCard** определяет наличие у метода нескольких параметров произвольных типов в количестве от **min** до **max** либо в количестве ровно **num** (**min**, **max**, **num** атрибуты элемента **WildCard**). Значение **min** по умолчанию 0, значение **max** «бесконечность» (т.е. означает «с двумя или более параметрами»; **<WildCard />** означает «с любым набором параметров»).
- **Throws** определяет типы исключений, которые класс может порождать.

6.4.3.2.8. Описание инструкции в коде

Элемент **Instruction** содержит элементы: **Class**, **Method**, **Arguments** и определяет вызов заданного метода заданного класса с заданными аргументами.

Элементы **Class**, **Method** описаны в разделах Описание класса и Описание метода.

Элемент **Arguments** определяет набор фактических аргументов вызова и содержит один или несколько элементов **Argument** или **Range**.

Элемент **Argument** определяет условие на один фактический аргумент и содержит элементы:

- **Pos** (позиция аргумента, начиная с нуля) либо **This** (если **Argument** – текущий объект), либо **Return** (если **Argument** – возвращаемое значение).
- **Type** либо **Class**.

- **Constant**, если условие на значение аргумента задаётся через константы (см. раздел Константы).
- **Flags**, если задаётся условие на флаги (см. раздел Флаги паттернов sink).

Элемент **Range** содержит подэлементы **PosMin**, **PosMax**, **Flags** и определяет группу подряд идущих аргументов, начиная с индекса **PosMin** и заканчивая (включительно) индексом **PosMax** (индексы нумеруются с нуля). **Flags** задаёт условия на флаги для группы аргументов (см. раздел Флаги паттернов sink).

6.4.3.3. Примеры паттернов

6.4.3.3.1. Примеры паттернов source/passthrough

Следующий паттерн **passthrough** указывает, что при вызове любого метода класса **org.apache.commons.httpclient.util.URIUtil** с одним или более параметрами, имя которого начинается с **encode**, возвращаемому значению необходимо присвоить все флаги нулевого аргумента и добавить флаг **URL_ENCODE**:

```
<Rule>
    <RuleId>SOURCE_00AF03C6</RuleId>
    <Patterns>
        <Pattern patternId="75b459">
            <Categories>
                <Category>JavaWeb</Category>
            </Categories>
            <Definitions>
                <Class id="class">
                    <Supers includeSelf="true">
                        <Class>
                            <Name>
<Value>org.apache.commons.httpclient.util.URIUtil</Value>
                            </Name>
                        </Class>
                    </Supers>
                    <Method id="method">
                        <Name>
                            <Regex>encode.*</Regex>
                        </Name>
                        <Parameters>
                            <WildCard min="1" />
```

```
                    </Parameters>
                </Method>
            </Class>
            <Instruction id="target_invoke">
                <Class>
                    <Ref id="class" reftype="classref" />
                </Class>
                <Method>
                    <Ref id="method" reftype="methodref" />
                </Method>
                <Arguments>
                    <Argument id="arg0">
                        <Pos>0</Pos>
                    </Argument>
                </Arguments>
            </Instruction>
        </Definitions>
        <Condition>
            <InstructionCondition>
                <Ref id="target_invoke" reftype="instructionref" />
            </InstructionCondition>
        </Condition>
        <ChangeFlags>
            <FlagsSet>
                <To>
                    <Argument>
                        <Return />
                    </Argument>
                </To>
                <From>
                    <Argument>
                        <Ref id="arg0" reftype="argumentref" />
</Argument>
                </From>
                <FlagsDiff>+URL_ENCODE</FlagsDiff>
```

```
                </FlagsSet>
            </ChangeFlags>
        </Pattern>
    </Patterns>
</Rule>
```

Паттерны **source** записываются по такой же схеме и отличаются от паттернов **passthrough** только отсутствием элемента **From** в элементе **Flagset**.

6.4.3.3.2. Пример паттерна sink

Следующий паттерн **sink** указывает, что вызов метода **dangerousMethod** класса **org.example.DangerousClass** с нулевым параметром типа **int** и ещё одним или более параметров небезопасен, если первый параметр обладает флагом **TAINTED**:

```
<Rule>
    <RuleId>TEST_RULE</RuleId>
    <Patterns>
        <Pattern patternId="123abc">
            <Categories>
                <Category>JavaWeb</Category>
            </Categories>
            <Severity>1</Severity>
            <Definitions>
                <Class id="class">
                    <Supers includeSelf="true">
                        <Class>
                            <Name>
<Value>org.example.DangerousClass</Value>
                            </Name>
                        </Class>
                    </Supers>
                    <Method id="method">
                        <Name>
                            <Regex>dangerousMethod</Regex>
                        </Name>
                        <Parameters>
                            <Parameter>
                                <Type>int</Type>
```

```
                        </Parameter>
                        <WildCard min="1" />
                    </Parameters>
                </Method>
            </Class>
            <Instruction id="target_invoke">
                <Class>
                    <Ref id="class" reftype="classref" />
                </Class>
                <Method>
                    <Ref id="method" reftype="methodref" />
                </Method>
                <Arguments>
                    <Argument>
                        <Pos>1</Pos>
                        <Flags>
                            <Flag>TAINTED</Flag>
                        </Flags>
                    </Argument>
                </Arguments>
            </Instruction>
        </Definitions>
        <Condition>
            <InstructionCondition>
                <Ref id="target_invoke" reftype="instructionref" />
            </InstructionCondition>
        </Condition>
        <ReportValues>
            <ReportValue key="bad_instruction">
                <Ref id="target_invoke" reftype="instructionref" />
            </ReportValue>
        </ReportValues>
    </Pattern>
</Patterns>
</Rule>
```

6.4.3.4. Запись паттерна для C# и PHP

Структура паттерна для **source/passthrough** паттернов для C# и PHP:

```
<Pattern>
    <Categories>
        <Category>LANG</Category>
    </Categories>
    <XPath>self::node[] </XPath>
    <Instruction>node </Instruction>
    <ChangeFlags>
        <FlagSet>
            …
        </FlagSet>
    </ChangeFlags>
</Pattern>
```

Значение текстового поля **LANG** это соответственно язык, для которого пишется правило, т.е. CS либо PHP.

Элемент **XPath** определяет вызов заданного метода заданного класса с заданными аргументами. Значение элемента **XPath** представляет собой XPath-запрос, начинающийся с оси **self** с последующим элементом:

- Для C# `member_access2` или `primary_expression_start` (в зависимости от того, что является ключевым элементом метод класса или конструктор).
- Для PHP `memberAccess` или `functionCall` (в зависимости от того, что является ключевым элементом метод класса или конструктор).

Элемент **Instruction** имеет значение:

- Для C# `member_access2` или `primary_expression_start` (зависит от того, с какого элемента начинается XPath-запрос).
- Для PHP `memberAccess` или `functionCall` (зависит от того, с какого элемента начинается XPath-запрос).

Элемент **ChangeFlags** содержит один или несколько элементов **FlagsSet**. Каждый элемент **FlagsSet** указывает, как изменить флаги для заданного набора сущностей (аргументов).

Элемент **FlagsSet** содержит элементы: **To**, **From**, **FlagsDiff** и необязательный атрибут **clearAll**, принимающий значения **true**, **false**. Элементы **To** и **From** определяют сущности, участвующие в процедуре изменения флагов. Сущности, которым можно присваивать флаги: **Argument**, **Range**.

Элемент **Argument** определяет условие на один фактический аргумент и содержит атрибут **pos** (позиции нумеруются с нуля).

Элемент **Range** содержит атрибуты **PosMin**, **PosMax** и определяет группу подряд идущих аргументов, начиная с индекса **PosMin** и заканчивая (включительно) индексом

PosMax (индексы нумеруются с нуля).

Элемент **FlagsDiff** содержит строку, задающую изменение флагов. Вид изменения задается в виде атрибута **diff**, принимающего значения:

- **add** – добавить флаг **FLAG**;
- **remove** – снять флаг **FLAG**, если он есть;
- **inverse** – изменить значение флага **FLAG**, если он есть.

Структура паттерна для **sink** паттернов для C# и PHP:

```
<Pattern>
        <XPath>//node</XPath>
</Pattern>
```

6.4.3.5. Запись паттернов для остальных языков

Для анализа языков, не компилирующихся в байт-код Java, используется промежуточное представление (AST). В этом случае паттерны поиска формулируются в терминах XPath-запросов к этому представлению.

Общая структура паттерна XML на базе **XPath**:

```
<Pattern>
    <XPath>//node</XPath>
</Pattern>
```

Разные языки по-разному транслируются в промежуточное представление. Поэтому XPath-запросы для разных исходных языков имеют различную структуру. Ниже приведены примеры XPath-запросов для поддерживаемых языков (редактируемые параметры выделены шрифтом). Запись паттерна для языка JavaScript

Вызов метода **testMethod** класса **testClass** с первым аргументом – целочисленной константой меньше 128:

```
//singleExpression[singleExpression[single Expression/text()='testClass']
[identifierName /text()='testMethod']][arguments/argumentList
[singleExpression[1]/literal/ numericLiteral[number(text())&lt; 128]]]
```

6.4.3.5.1. Запись паттерна для языка LotusScript

Вызов метода **TestMethod** класса **TestClass**:

```
//iCS_B_MemberProcedureCall[implicitCallStmt_InStmt/iCS_S_VariableOrProcedureCall
/ambiguousIdentifier[text()[1] = 'TestClass']]
[ambiguousIdentifier/ambiguousKeyword[text()[1] = 'TestMethod']]
```

6.4.3.5.2. Запись паттерна для языка TypeScript

Вызов функции **testFunction** с константным аргументом, удовлетворяющим регулярному выражению:

```
//callExpression[memberExpression/identifierName[text()[1] = 'testFunction']]
[arguments//literal[matches(text()[1],'.*val.*','i')]]
```

6.4.3.5.3. Запись паттерна для языка VBScript

Вызов метода **TestMethod** класса **TestClass** с третьим аргументом – константной строкой:

```
//iCS_B_MemberProcedureCall[implicitCallStmt_InStmt
//ambiguousIdentifier[matches(@class,
'^TestClass$','i')]][ambiguousIdentifier/ambiguousKeyword/text
()[1]='TestMethod'][argsCall/argCall[3]//literal[text()[1]]
```

6.4.3.5.4. Запись паттерна для языка HTML5

Атрибут **attribute** со значением **bad_value**:

```
//htmlElement[htmlAttribute[htmlAttributeName[matches(text()[1], 'attribute', 'i')]]
[htmlAttributeValue[matches(text()[1], 'bad_value', 'i')]]]
```

6.4.3.5.5. Запись паттерна для языка Python

Вызов метода **test_method** класса **TestClass** с аргументом строкового типа длиной меньше 16 символов:

```
//power[atom[1]/text()[1]='TestClass'][trailer[1]
/text()[1]='test_method'][trailer[2]/arglist/argument//
arith_expr/term/factor/power/atom/string
[string-length(text()[1])&amp;lt;16]]
```

6.4.3.5.6. Запись паттерна для языка Perl

Вызов метода **insecure_operator()** с двумя аргументами:

```
//listOperatorCall[listOperatorName[text()[1] = 'insecure_operator']]
[list/operators[operator[1]][operator[2]][not(operator[3])]]
```

6.4.3.5.7. Запись паттерна для языка PL/SQL

Вызов процедуры **test_proc** класса **test_class** с константным аргументом:

```
//function_call[routine_name[./id/id_expression
/regular_id[matches(text(), '^test_class$','i')]][./id_expression/regular_id
[matches(text(),'^test_proc$','i')]]][function_argument[.//constant]]
```

6.4.3.5.8. Запись паттерна для языка Ruby

Вызов метода **test_method** с целочисленным аргументом:

```
//function_call[function_name[1]/id/text() [1]='test_method']
[function_call_param_list[1]/function_call_params/function_param/
function_unnamed_param/int_result/int_t]
```

6.4.3.5.9. Запись паттерна для языка T-SQL

Вызов функции **TESTFUNC** с аргументом – константной строкой, равной **TEST_STRING_CONSTANT**:

```
//function_call[scalar_function_name/func_proc_name
/id/simple_id[matches(text()[1],'^TESTFUNC$','i')]
][expression_list/expression[1]/constant[matches
(text()[1],'^TEST_STRING_CONSTANT$','i')]]
```

6.4.3.5.10. Запись паттерна для языков Objective-C и C

Вызов функции **TEST_FUNC1** или **TEST_FUNC2**:

```
//CallExpr[./*[1]//FunctionDecl[@name='TEST_FUNC1'or @name='TEST_FUNC2']]
```

6.4.3.5.11. Запись паттерна для языка C++

Вызов функции с именем **bad_function**:

```
<Patterns>
  <Pattern id="kjdfbgv132">
    <Condition>
      <callExpr>
        <callee>
          <functionDecl>
            <hasName id="bad_function"/>
          </functionDecl>
        </callee>
      </callExpr>
```

```
    </Condition>
    <Action>
      <Report id="bad_function" severity="1" confidence="5"/>
    </Action>
  </Pattern>
</Patterns>
```

6.4.3.5.12. Запись паттерна для языка ABAP

Вызов функции **test_func**:

```
//statement/callStatement/call_method_static_short/methodId1/methodId[1]
/anySingleToken[matches(text()[1], 'test_func', 'i')]
```

6.4.3.5.13. Запись паттерна для языка 1С

Вызов конструктора **ТестовыйКонструктор** с численным аргументом, заданным в исходном коде:

```
//constructorStatement[identifier[matches(text()[1],
'^(ТестовыйКонструктор)$', 'i')]]
[argumentList[argument[1]/expression/primaryExpression
/simpleIdentifier/dataType/literal/numericLiteral]]
```

6.4.3.5.14. Запись паттерна для языка Apex

Вызов конструктора со значением пятого аргумента **bad_value**:

```
//expression/creator[createdName[matches(text()[1], '^Test$', 'i')]]
[classCreatorRest//expressionList/expression[5]//literal[matches(text()[1],
'^bad_value$', 'i')]]
```

6.4.3.5.15. Запись паттерна для языка Go

Вызов функции **testFunc**:

```
//primaryExpr[primaryExpr/operand/operandName[matches(text()[1],
'^testFunc$', 'i')]]
```

6.4.3.5.16. Запись паттерна для языка Rust

Заданная в исходном коде переменная **my_variable**:

```
//let[assignee/pattern//identifier[matches(text(), 'my_variable')]]
[initialization/expression[count(*) = 1 and literal[text()[1]]]]
```

6.4.3.5.17. Запись паттерна для языка Groovy

Вызов метода **testMethod**:

```
//expression[callExpressionRule[selectorName/IDENTIFIER
[matches(text()[1],'^testMethod$', 'i')]]]
```

6.4.3.5.18. Запись паттерна для языка Dart

Вызов метода **testMethod** класса **testClass**:

```
//postfixExpression[primary/identifier/identifierNotFUNCTION[text()[1]
= 'testClass']][selector/assignableSelector/unconditionalAssignableSelector/
identifier/identifierNotFUNCTION[text()[1] = 'testMethod']]
```

6.4.3.5.19. Запись паттерна для языка Delphi

Вызов функции **test_func**:

```
//statement[.//designator//ident[matches(text()[1], 'test_func')]]
[.//expression//factor[(intNum|stringFactor[string-length(text()[1]) > 0])]]
```

6.4.3.5.20. Запись паттерна для языка Pascal

Вызов функции **test_func**:

```
//methodOrFunctionId/extendedIdentifier/identifier/ident
[matches(text()[1], '^test_func$', 'i')]
```

6.4.3.5.21. Запись паттерна для языка VB.NET

Вызов метода **TestMethod** класса **TestClass** с третьим аргументом – константной строкой:

```
//iCS_B_MemberProcedureCall[implicitCallStmt_InStmt
//ambiguousIdentifier[matches(@class,
'^TestClass$','i')]][ambiguousIdentifier/ambiguousKeyword/text
()[1]='TestMethod'][argsCall/argCall[3]//literal[text()[1]]
```

6.4.3.5.22. Запись паттерна для языка VBA

Вызов функции **TestFunction** со строковым аргументом, заданным в исходном коде:

```
//iCS_S_VariableOrProcedureCall[ambiguousIdentifier[matches(text()[1],
'^TestFunction$', 'i')]][subscripts/subscript[1]//ambiguousIdentifier/literal]
```

6.4.3.5.23. Запись паттерна для языка Visual Basic 6.0

Вызов метода **TestMethod** класса **TestClass** с третьим аргументом – константной строкой:

```
//iCS_B_MemberProcedureCall[implicitCallStmt_InStmt
//ambiguousIdentifier[matches(@class,
'^TestClass$','i')]][ambiguousIdentifier/ambiguousKeyword/text
()[1]='TestMethod'][argsCall/argCall[3]//literal[text()[1]]
```

6.4.3.5.24. Запись паттерна для языка Solidity

Вызов функции **testfunc**:

```
//atom_name[text()[1]='insecure_function']
```

6.4.3.5.25. Запись паттерна для языка Vyper

Вызов функции **insecure_function()**:

```
//functionName/identifier[matches(text()[1], '^testfunc$', 'i')]
```

6.4.3.5.26. Запись паттерна для языка COBOL

Заданная в исходном коде переменная **sensitiveData**:

```
//moveToStatement[moveToSendingArea/literal]
/identifier//cobolWord[matches(text()[1], '^sensitiveData$', 'i')]
```

6.4.4. Наборы правил

Для поиска наборов воспользуйтесь строкой поиска набора по названию и фильтрами в правом боковом меню. После нажатия на название набора откроется список языков, которые содержатся в выбранном наборе. Для просмотра правил в наборе выберите нужный язык.

В поле поиска набора по названию после ввода значения нажмите **Enter** или выберите подходящее значение из открывшегося списка. Искать наборы можно сразу по нескольким значениям.

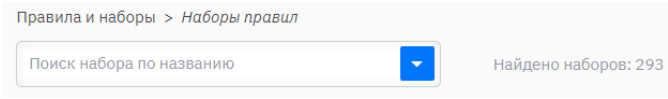


Рис. 6.49: Строка поиска наборов

Фильтровать наборы можно по языку и типу автора набора. Для установки изначальных значений строки поиска и фильтров нажмите **Сбросить**.

Результатом поиска наборов будут списки, содержащие подходящие наборы. Нажмите на правило из набора, чтобы посмотреть информацию о нём. Чтобы изменить пользовательский набор, нажмите на кнопку (рис. 6.50). На вкладке **Редактировать набор** можно выполнить следующие действия:

- посмотреть список правил, входящих в набор;
- изменить название набора, добавить/удалить правила из набора, сделать набор общим/личным. По завершении редактирования нажмите на кнопку **Сохранить**;
- удалить набор.

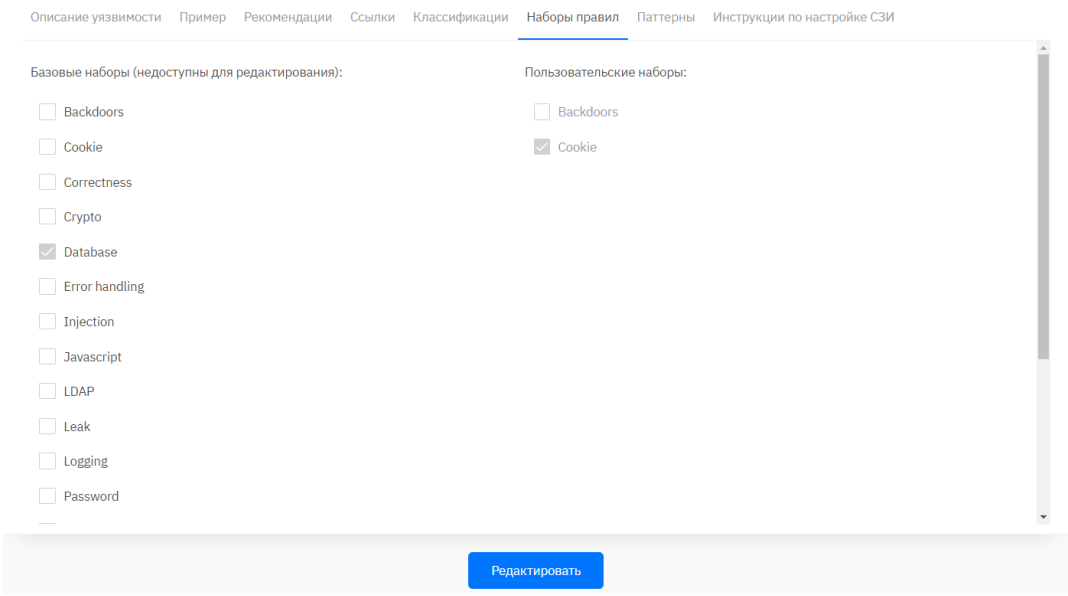


Рис. 6.50: Редактирование набора правил

Чтобы создать собственный набор правил, на вкладке **Наборы правил** нажмите **Создать набор**. Введите его название и выберите правила, которые будут в него входить (рис. 6.51). Можно выбирать как отдельные правила, так и группы правил, входящие в уже существующий набор. Набор можно сделать публичным или приватным. Для завершения создания набора нажмите **Сохранить**.

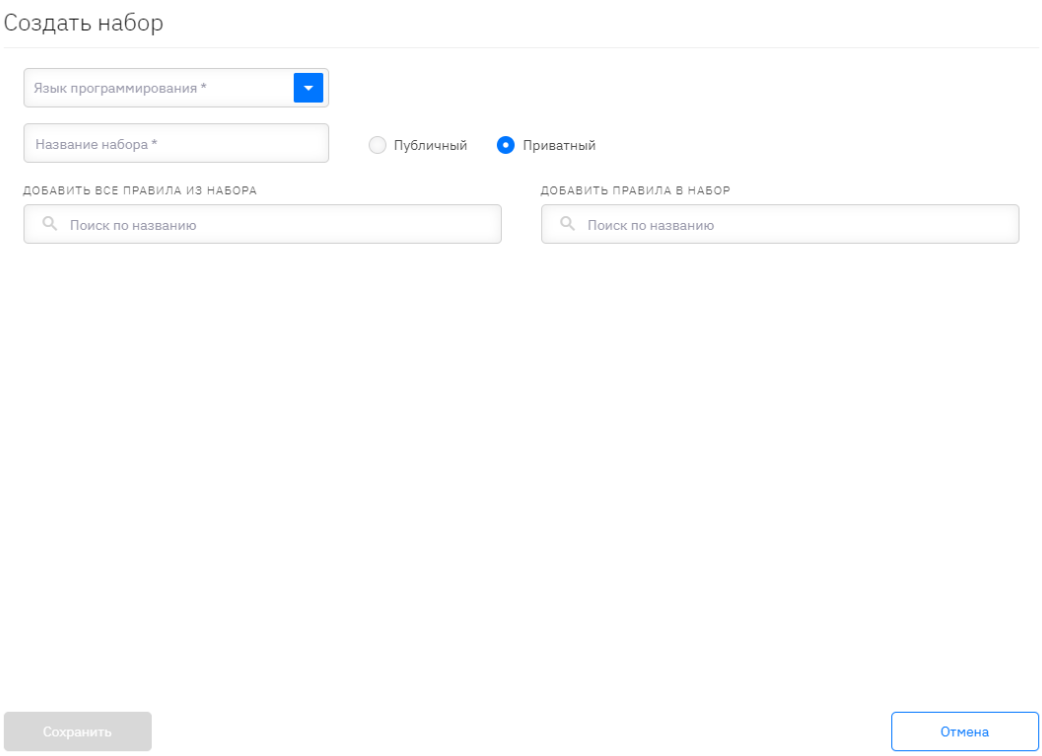


Рис. 6.51: Создание пользовательского набора правил

6.5. Интеграции Solar appScreener

Все плагины для интеграции с Solar appScreener находятся в директории **/opt/appScreener** (Linux) либо **C:\appScreener** (Windows). Для интеграции с **Jira** плагина не требуется.

6.5.1. Jira

Solar appScreener использует для интеграции **Jira REST API v2**.

6.5.1.1. Как привязать проект в Solar appScreener к проекту в Jira

Для того чтобы привязать проект в Solar appScreener к проекту в Jira:

1. Перейдите на страницу **Проекты**.
2. Выберите проект и перейдите на вкладку **Jira** в разделе **Настройки**.
3. Нажмите на кнопку **Привязать проект**.

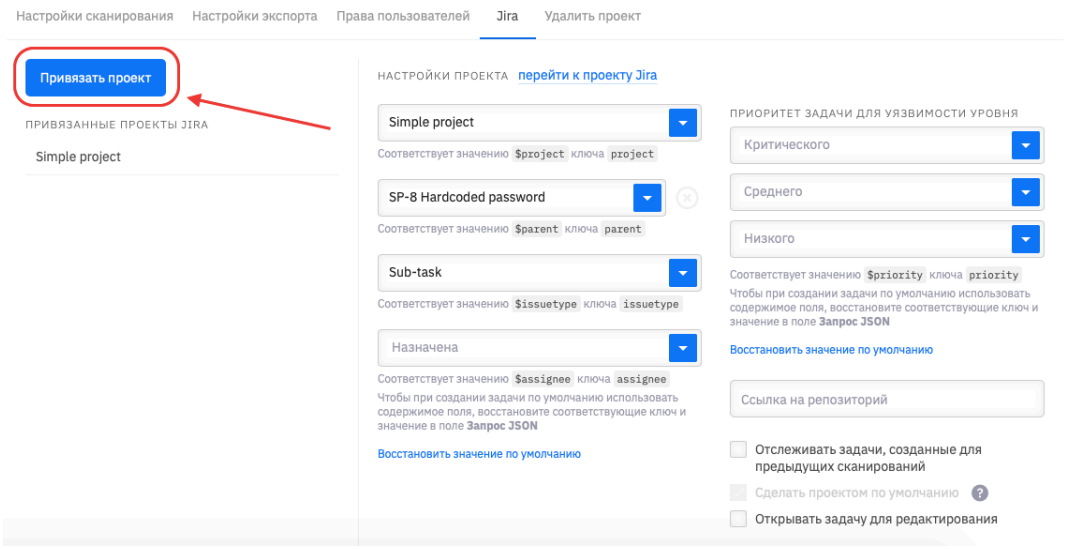


Рис. 6.52: Привязать проект Jira

4. Выберите из списка проект Jira и тип задачи по умолчанию при создании новых задач Jira.
5. Укажите опциональные значения формы, которые также будут использоваться по умолчанию при создании новых задач в Jira.
6. Настройте автоматическое создание задач в Jira по результатам сканирования, если необходимо.

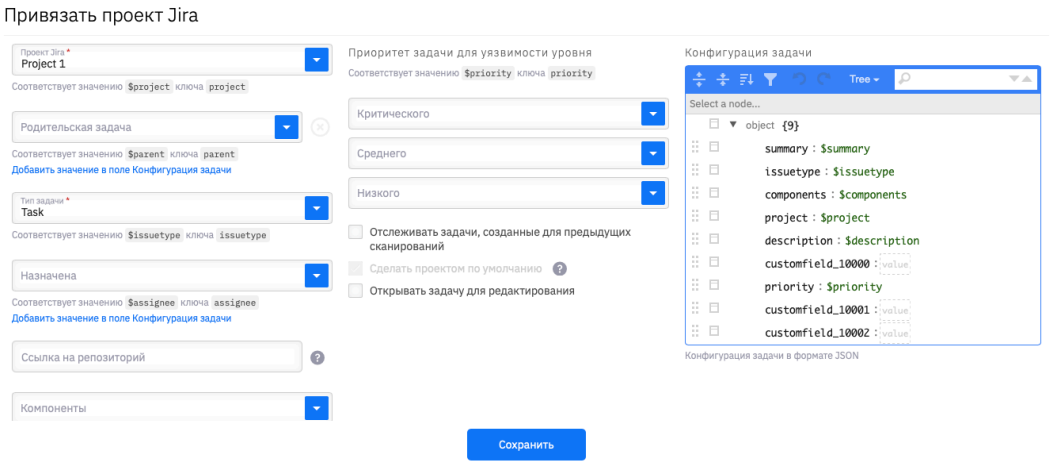


Рис. 6.53: Настроить параметры проекта Jira

Чтобы настроить уже привязанный проект, выберите его из списка и отредактируйте поля формы.

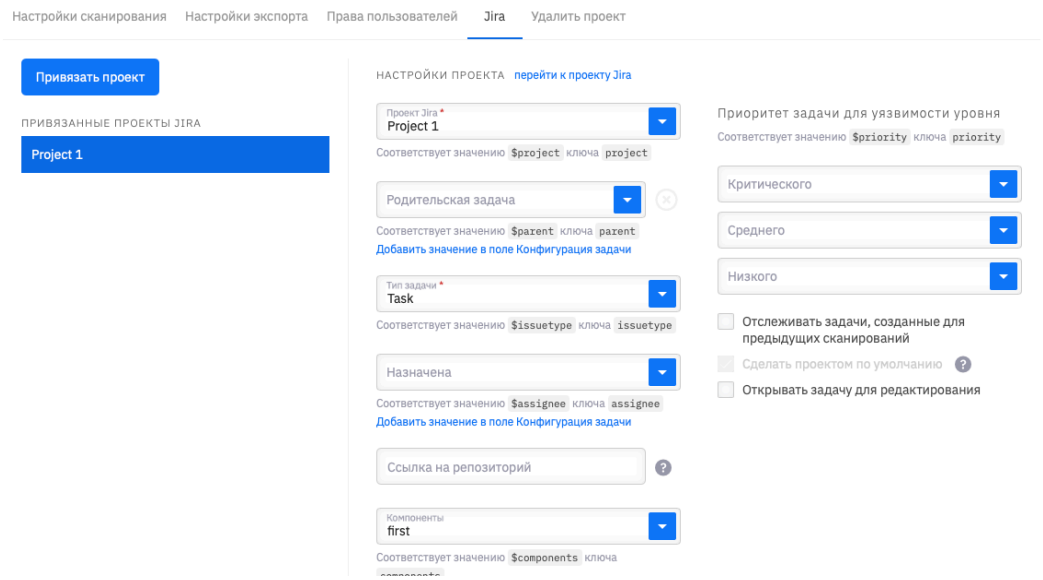


Рис. 6.54: Изменить параметры проекта Jira

6.5.1.2. Создание задачи в Jira

Если проект в Solar appScreener привязан к проекту в Jira, через интерфейс Solar appScreener можно создавать задачи в Jira. Для этого выполните действия:

1. Перейдите в раздел **Подробные результаты**.
2. Выберите конкретную уязвимость.
3. Перейдите на вкладку **Jira**, которая располагается в правой нижней части страницы (на данной вкладке отображается список уже созданных задач в Jira).
4. Нажмите на кнопку **Создать задачу**.

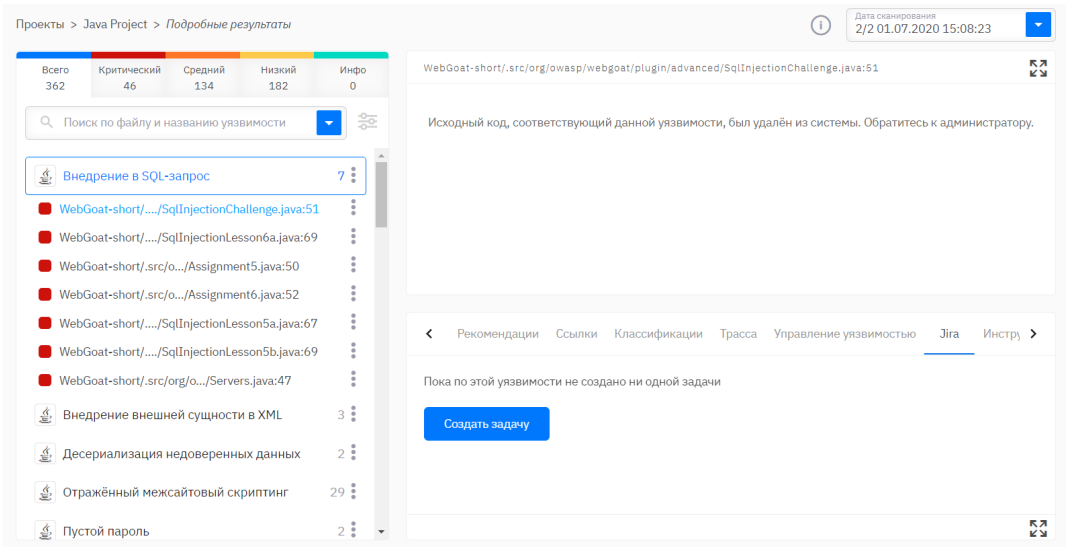


Рис. 6.55: Создать задачу в Jira

5. Укажите проект, родительскую задачу (опционально), тип задачи, компоненты (опционально), тему задачи, приоритет (опционально), кому назначена задача (опционально), описание задачи в формате Jira (опционально).

6. Если в задаче есть другие обязательные поля, для них в поле **Конфигурация задачи** будут сгенерированы пары ключ и значение. Явно укажите значение, которое необходимо задать в задаче.
7. Нажмите на кнопку **Создать**.

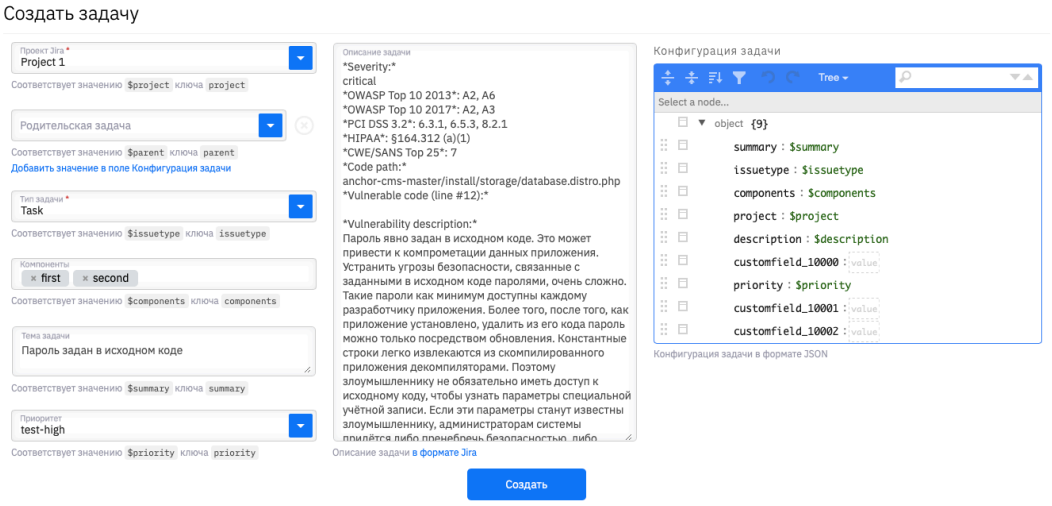


Рис. 6.56: Параметры задачи в Jira

В результате этих действий создается задача в Jira. Для просмотра задачи в Jira кликните на название задачи в списке. Для удаления задачи из интерфейса appScreener нажмите на кнопку удаления.

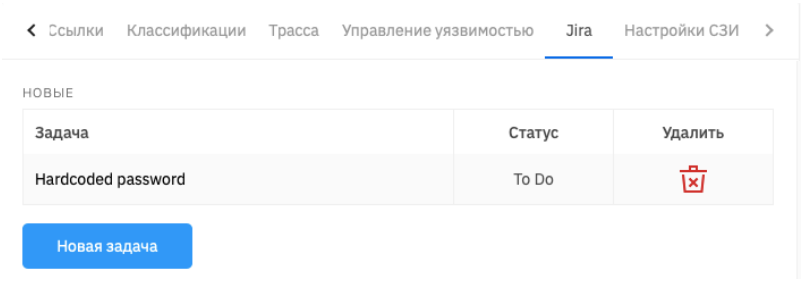


Рис. 6.57: Список задач в Jira

6.5.2. Jenkins

Solar appScreener поддерживает **Jenkins 2.164** и более поздние версии.

6.5.2.1. Установка расширения в Jenkins

1. Перейдите на страницу настроек Jenkinks **Manage Jenkins** (**http://<installation_address>:8090/jenkins/manage**, **<installation address>** адрес машины, на которой установлен Jenkins).

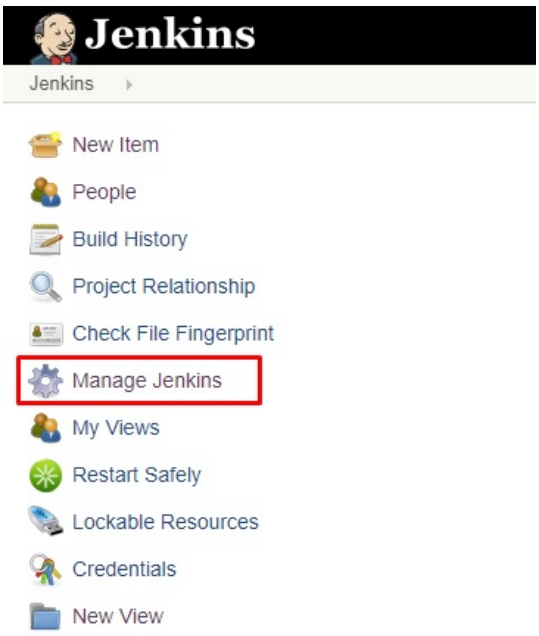


Рис. 6.58: Jenkins: Настройки Jenkins

2. Выберите **Manage Plugins** (**Управление плагинами**).

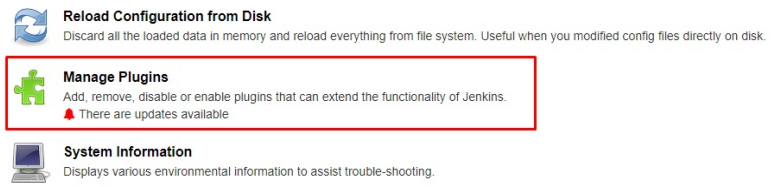


Рис. 6.59: Jenkins: Управление плагинами

3. Выберите вкладку **Advanced** (**Дополнительно**).

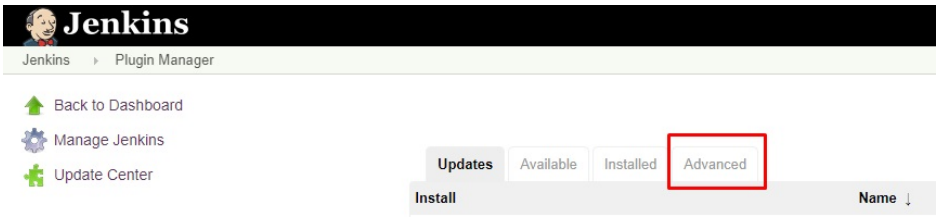


Рис. 6.60: Jenkins: Дополнительные настройки

4. В пункте **Upload Plugin** (**Загрузить плагин**) нажмите кнопку **Choose the file** (**Выберите файл**).

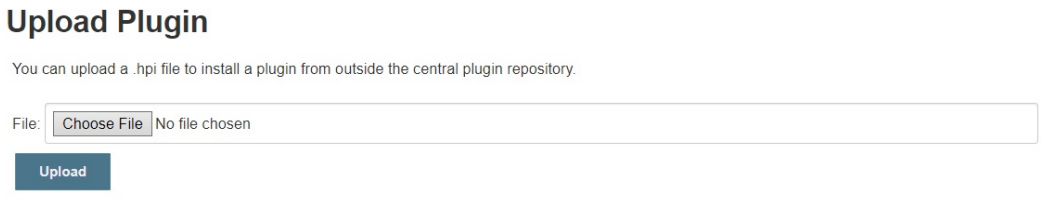


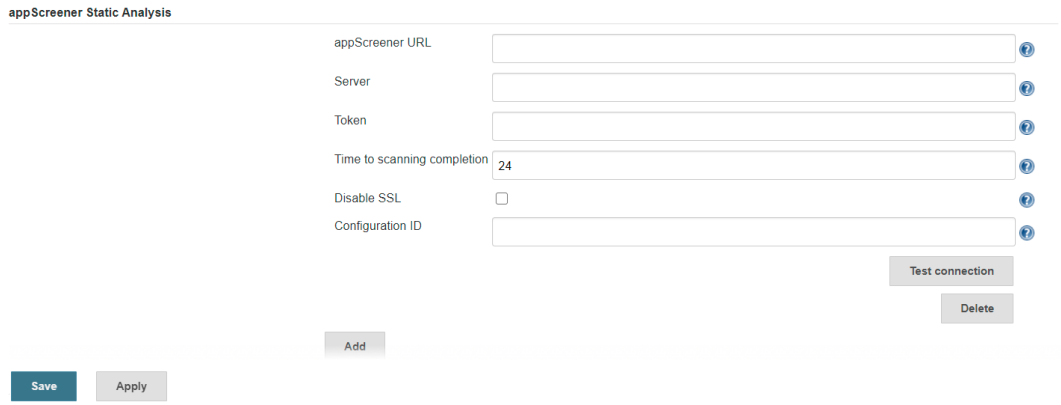
Рис. 6.61: Jenkins: Загрузить плагин

5. Выберите файл расширения в открывшемся всплывающем окне.
6. Нажмите **Upload** (**Загрузить**).

6.5.2.2. Общие настройки расширения

1. Перейдите на страницу настроек Jenkinks **Manage Jenkins** (**http://<installation_address>:8090/jenkins/manage**, **<installation_address>** адрес машины, на которой установлен Jenkins).
2. Выберите **Configure System** (**Конфигурация системы**).
3. Найдите пункт **appScreener Static Analysis** (**Статический анализ appScreener**) и заполните поля:
   - введите appScreener URL, например **http://<installation_address>**, **<installation_address>** адрес машины, на которой установлен appScreener;
   - введите адрес REST API для appScreener, например **http://<installation_address>/app/api/v1/**, **<installation_address>** адрес машины, на которой установлен appScreener;
   - введите токен, его можно получить в интерфейсе самого appScreener в разделе **Личный кабинет** (**Account**);
   - введите время ожидания завершения сканирования;

Поле **Configuration ID** (**ID конфигурации**) будет заполнено автоматически после сохранения настроек. Оно указывается в качестве значения параметра **configUuid** в **Jenkins Pipeline Script** и хранит в себе глобальные настройки плагина.



4. Нажмите кнопку **Save** (**Сохранить**) внизу страницы.

6.5.2.3. Конфигурация анализа и отчёта

Конфигурацию анализа и отчёта можно осуществить двумя способами:

- через задачу со свободной конфигурацией;
- через **Jenkins Pipeline**.

6.5.2.3.1. Задача со свободной конфигурацией

1. Выберите нужный **Item** (далее нажмите **Configure** (**Настройки**) и перейдите к пункту 5) или создайте новый **New Item** (**Создать Item**).

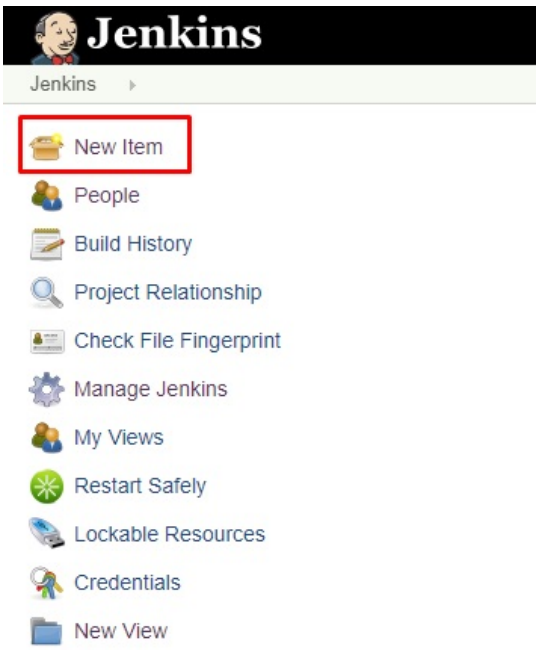


Рис. 6.62: Jenkins: Новый Item

2. Введите имя Item'а.
3. Выберите **Freestyle project** (**Создать задачу со свободной конфигурацией**).

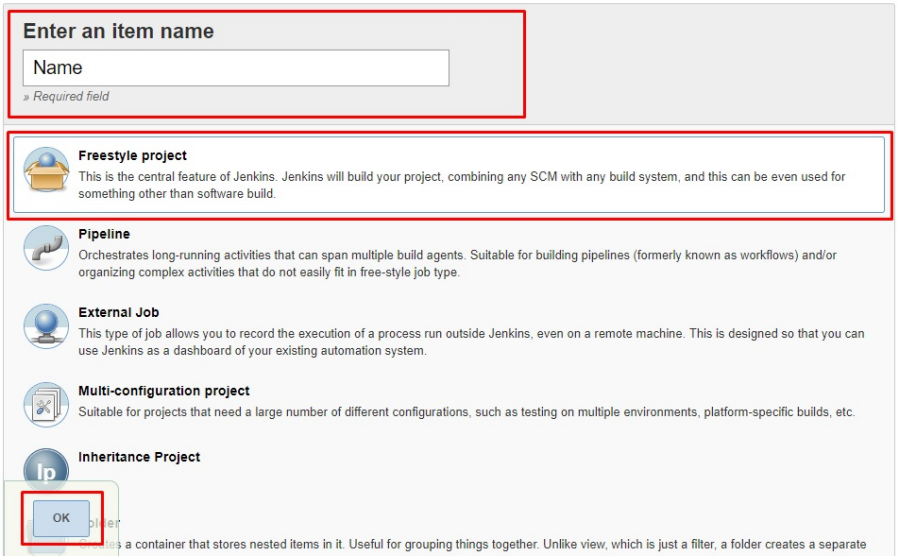


Рис. 6.63: Jenkins: Задача со свободной конфигурацией

4. Нажмите **OK**.
5. В пункте **Build** (**Сборка**) в поле **Add Build Step** (**Добавить шаг сборки**) выберите **Execute appScreener Static Analysis** (**Выполнить статический анализ appScreener**).
6. В раскрывшемся меню:
   1. Введите UUID проекта.

Project UUID это идентификатор существующего проекта appScreener. Скопировать **Project UUID** можно в боковом меню проекта. Пример: 9feefaf0-4c17-47fe-b1f5-f7f64d4da722. Настройка конфигураций ниже перенастроит проект appScreener, если текстовое поле **Project UUID** не пустое. Если текстовое поле **Project UUID** пустое, плагин Jenkins создаст новый проект в appScreener.

2. Выберите сервер.
3. Настройте остальные конфигурации сканирования (если потребуется). Подробнее о конфигурациях сканирования см. Настройки.

7. В пункте **Post-build Actions** (**Послесборочные операции**) в поле **Add Build Step** (**Добавить шаг сборки**) выберите **appScreener Static Analysis report export in PDF** (**Экспорт pdf-отчёта статического анализа appScreener**) и сделайте настройку (см. раздел Экспорт отчёта).
8. Нажмите **Save** (**Сохранить**).

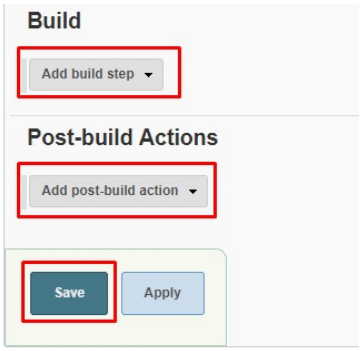


Рис. 6.64: Jenkins: Послесборочные операции

6.5.2.3.2. Pipeline

1. Выберите нужный **Item** (далее нажмите **Configure** (**Настройки**) и перейдите к пункту 5) или создайте новый **New Item** (**Создать Item**).
2. Введите имя Item'а.
3. Выберите **Pipeline**.
4. Нажмите **OK**.
5. В пункте **Pipeline** нажмите **Pipeline Syntax**.

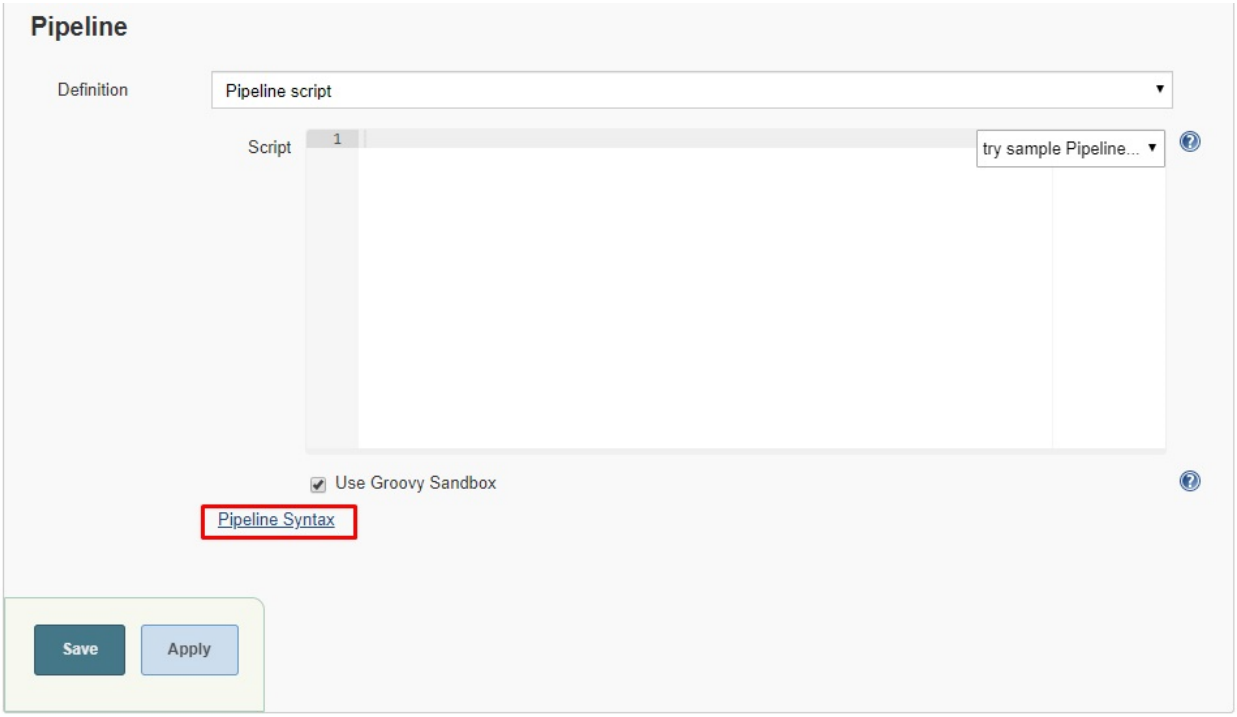


Рис. 6.65: Jenkins: Pipeline Syntax

6. В открывшейся вкладке в браузере в пункте **Steps** в поле **Sample Step** выберите **step: General Build Step**.
7. В пункте **Build Step** выберите **Execute appScreener Static Analysis** (**Выполнить статический анализ appScreener**).

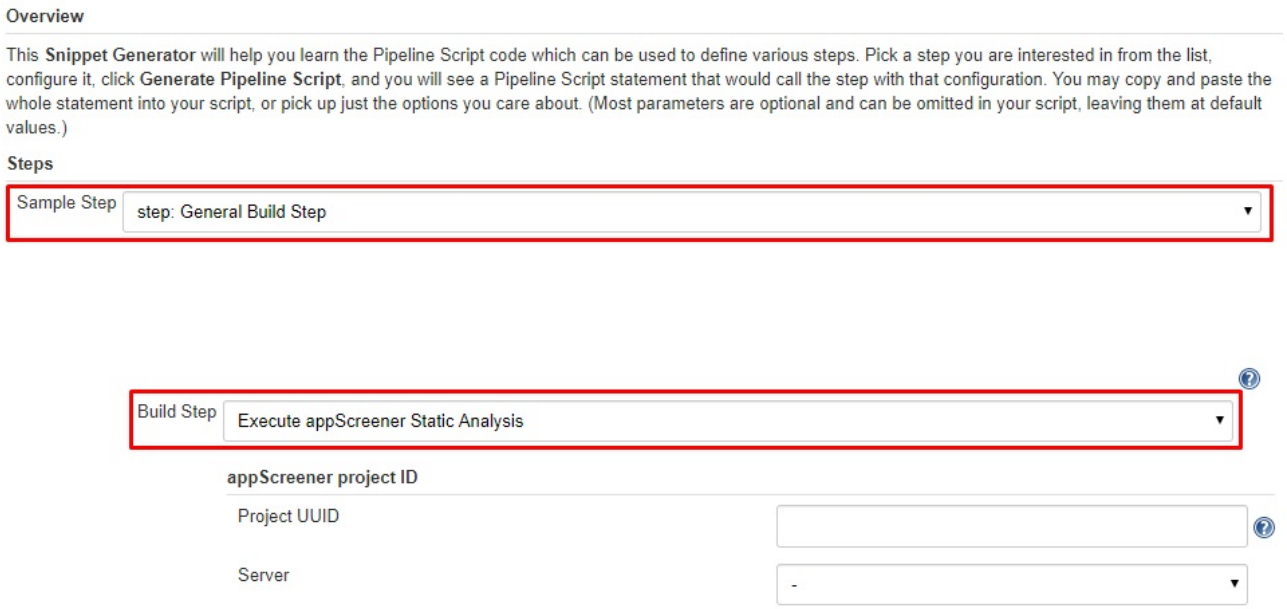


Рис. 6.66: Jenkins: Сборка

8. В появившемся пункте **appScreener project ID** (**Идентификатор проекта appScreener**):
   1. Введите UUID проекта.

Project UUID это идентификатор существующего проекта appScreener. Скопировать **Project UUID** можно в боковом меню проекта. Пример: 9feefaf0-4c17-47fe-b1f5-f7f64d4da722 . Настройка конфигураций ниже перенастроит проект appScreener, если текстовое поле **Project UUID** не пустое. Если текстовое поле **Project UUID** пустое, плагин Jenkins создаст новый проект в appScreener.

2. Выберите сервер.
3. Настройте остальные конфигурации сканирования (если потребуется). Подробнее о конфигурациях сканирования см. Настройки.

9. Нажмите **Generate Pipeline Script**.

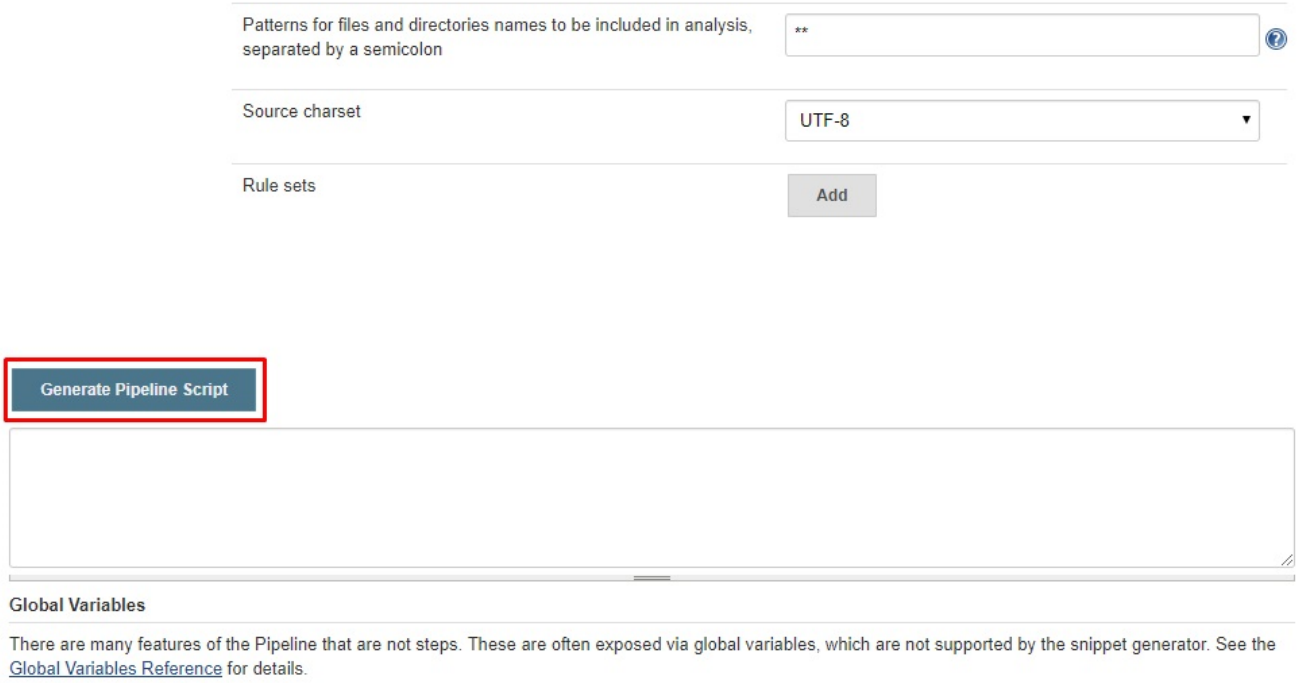


Рис. 6.67: Jenkins: Генерация скрипта Pipeline

10. Скопируйте появившийся скрипт для его использования на странице конфигурации проекта (шаг 5) в поле **Script**.
11. Вернитесь во вкладку **Pipeline Syntax**.
12. В пункте **Build Step** выберите **appScreener Static Analysis report export in PDF** (**Экспорт PDF-отчёта статического анализа appScreener**).

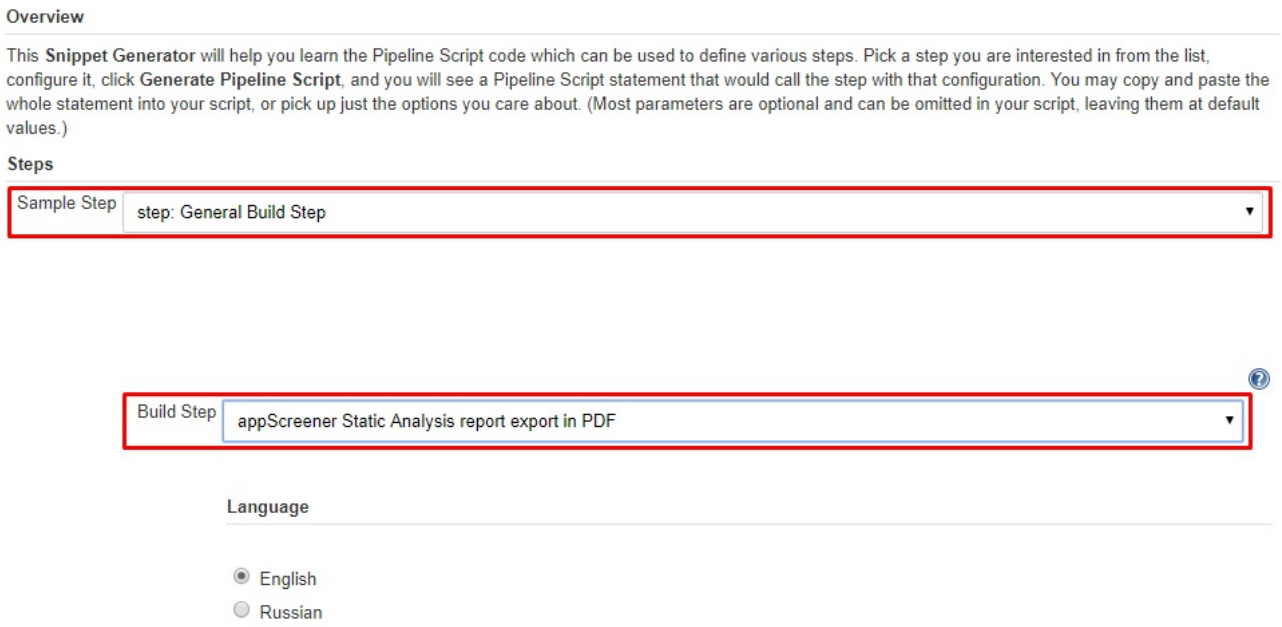


Рис. 6.68: Jenkins: Экспорт отчёта в PDF

13. В появившемся пункте настройте конфигурации формирования отчёта (если потребуется).
14. Нажмите **Generate Pipeline Script**.
15. Скопируйте появившийся скрипт для его использования на странице конфигурации проекта (шаг 5) в поле **Script**.

Пример:

```
pipeline {
            agent any
            stages {
                stage('code_obtain') {
                    steps{
                        git 'https://github.com/example.git'
                    }
                }
                stage('analysis') {
                    steps{
                        step([$class: 'InCodeBuilder', analyzeConfig: true,
                        analyzeJsLibs: false, analyzeLib: true, configUuid:
                        '75ed295c-728a-409a-9c5d-de85ad30b932', dirExc: '',
                        dirInc: '**', extrules: false, incremental: false,langs:
                        [[checked: false, name: 'Java, Scala, Kotlin', value:
```

```
            'JAVA'], [checked: false, name: 'PHP', value: 'PHP'],
            [checked: true, name: 'C#', value: 'CS'], [checked:
            false, name: 'JavaScript', value: 'JAVASCRIPT'],
            [checked: false, name: 'LotusScript', value: 'LOTUS'],
            [checked: false, name: 'TypeScript', value:
            'TYPESCRIPT'], [checked: false, name: 'VBScript', value:
            'VBSCRIPT'], [checked: false, name: 'HTML5', value:
            'HTML5'], [checked: false, name: 'Python', value:
            'PYTHON'], [checked: true, name: 'C/C++', value: 'CCPP'],
            [checked: false, name: 'Objective-C', value: 'OBJC'],
            [checked: false, name: 'Swift', value: 'SWIFT'],
            [checked: false, name: 'PL/SQL', value: 'PLSQL'],
            [checked: false, name: 'T-SQL', value: 'TSQL'],
            [checked: false, name: 'ABAP', value: 'ABAP'], [checked:
            false, name: '1C', value: 'ONES'], [checked: false, name:
            'Apex', value: 'APEX'], [checked: false, name: 'Go',
            value: 'GO'], [checked: false, name: 'Ruby', value:
            'RUBY'], [checked: false, name: 'Rust', value: 'RUST'],
            [checked: false, name: 'Groovy', value: 'GROOVY'],
            [checked: false, name: 'Dart', value: 'DART'],
            [checked: false, name: 'Delphi', value: 'DELPHI'],
            [checked: false, name: 'VBA', value: 'VBA'], [checked:
            false, name: 'Visual Basic 6', value: 'VB'], [checked:
            false, name: 'Solidity', value: 'SOLIDITY'], [checked:
            false, name: 'COBOL', value: 'COBOL']], mobileApp: false,
            noBuild: true, projectUuid: '', ruleSets: [],
            sourceEncoding: 'UTF-8', visualStudio: false])
        }
    }
    stage('report_extraction') {
        steps{
            step([$class: 'InCodePublisher', classificationVul: 'CR',
            comparisonScan: '', criticalVul: true, f5: false,
            fixedIssues: false, impervaSecure: false,
            includeComparison: false, includeDeleted: false,
```

```
                        includeNotDecompiled: false, includeStd: false,
                        lowLevelVul: false, midLevelVul: true, modSecurity: false,
                        newIssues: true, savedIssues: true, scanDuration: true,
                        scanHistory: false, score: true, useLanguage: 'en',
vulTable: true])
                    }
                }
            }
        }
```

16. Нажмите **Save** (**Сохранить**).

6.5.2.4. Конфигурация Build Failure Conditions

Для того, чтобы создавать **build failure conditions** на основе переменных в виде метрик в **post-build step**:

1. Выберите нужный **Item** из списка.

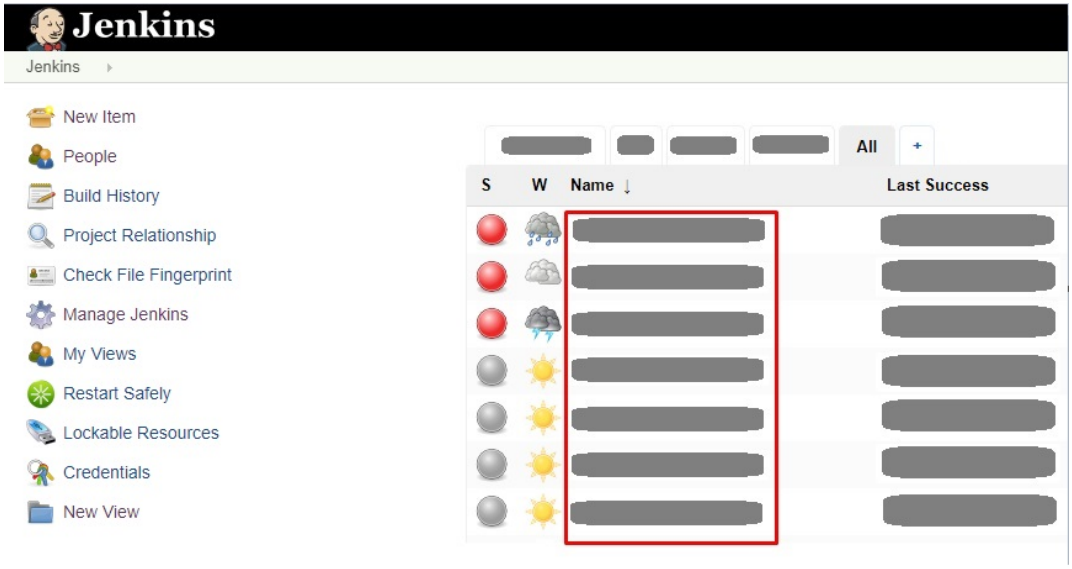


Рис. 6.69: Jenkins: Выбор Item'а

2. Нажмите **Configure** (**Настройки**).

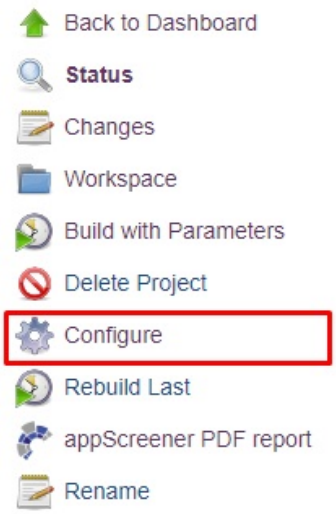


Рис. 6.70: Jenkins: Выбор настроек

3. В пункте **Post-build Actions** (**Послесборочные операции**) нажмите **Add post-build action** (**Добавить шаг после сборки**).

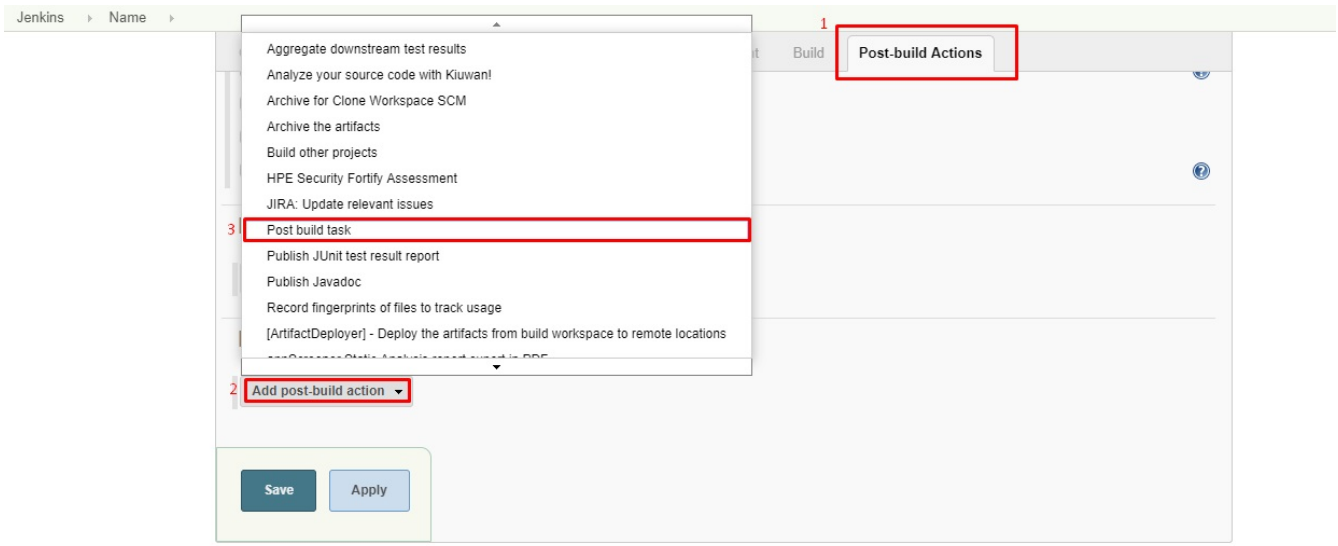


Рис. 6.71: Jenkins: Выбор послесборочных операций

4. Выберите **Post build task**.
5. Выберите появившийся раздел **Post build task**.
6. В разделе **Post build task** укажите **build failure conditions** в формате bash скрипта. Например, скрипт:

```
if [[ $SCORE < 3.5 || $CRITICAL > 10 || $LOW > 30 ]]; then
  echo "vulnerable app"
  exit 1
else
```

```
  exit 0
fi
```

означает, что сборка не будет завершена для проектов с рейтингом ниже 3,5, или количеством критических уязвимостей больше 10, или количеством уязвимостей низкого уровня больше 30.

Используйте глобальные переменные:

- Плагин создает переменные среды для сборки:
  - **PROJECT_ID** UUID проекта в appScreener;
  - **SCAN_ID** UUID сканирования в appScreener;
  - **SCAN_URL** ссылка на результаты сканирования в appScreener;
  - **SERVER** адрес REST API;
  - **SERVER_UUID** идентификатор сервера, присваиваемый при настройке глобальной конфигурации;
  - **PDF_URL** адрес PDF-отчёта;
  - **SCORE** рейтинг проекта в appScreener;
  - **TOTAL** общее количество уязвимостей в проекте;
  - **CRITICAL** количество уязвимостей критического уровня;
  - **MEDIUM** количество уязвимостей среднего уровня;
  - **LOW** количество уязвимостей низкого уровня;
  - **INFO** количество уязвимостей информационного уровня.
- Следующие параметры рассчитываются при установке чекбокса **Включить в отчёт сравнение со сканированием**. Они рассчитываются относительно сканирования, UUID которого указан в поле **UUID сканирования для сравнения**.
  - **NEW_TOTAL** общее количество новых уязвимостей в проекте;
  - **NEW_CRITICAL** количество новых уязвимостей критического уровня;
  - **NEW_MEDIUM** количество новых уязвимостей среднего уровня;
  - **NEW_LOW** количество новых уязвимостей низкого уровня;
  - **NEW_INFO** количество новых уязвимостей информационного уровня.

Эти переменные можно потом использовать для других шагов сборки.

7. Активируйте чекбокс **Escalate script execution status to job status**.

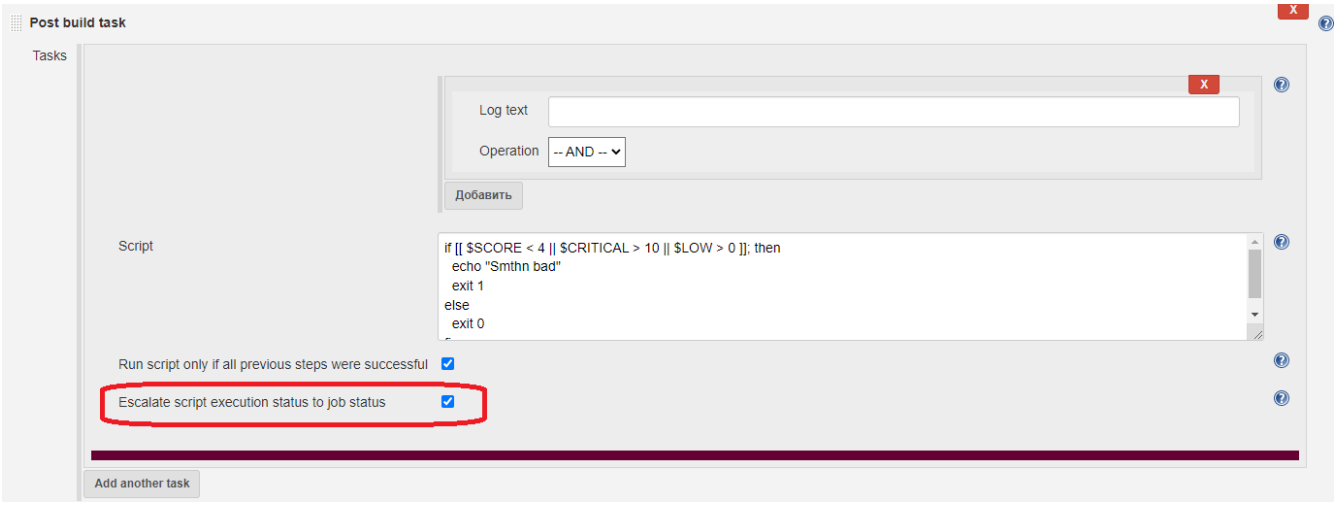


Рис. 6.72: Jenkins: Сохранение послесборочных операций

7. Нажмите **Save** (**Сохранить**).

6.5.2.5. Результаты и PDF-отчёт

Для того чтобы просмотреть результаты сканирования:

1. Выберите нужный **Item** из списка.

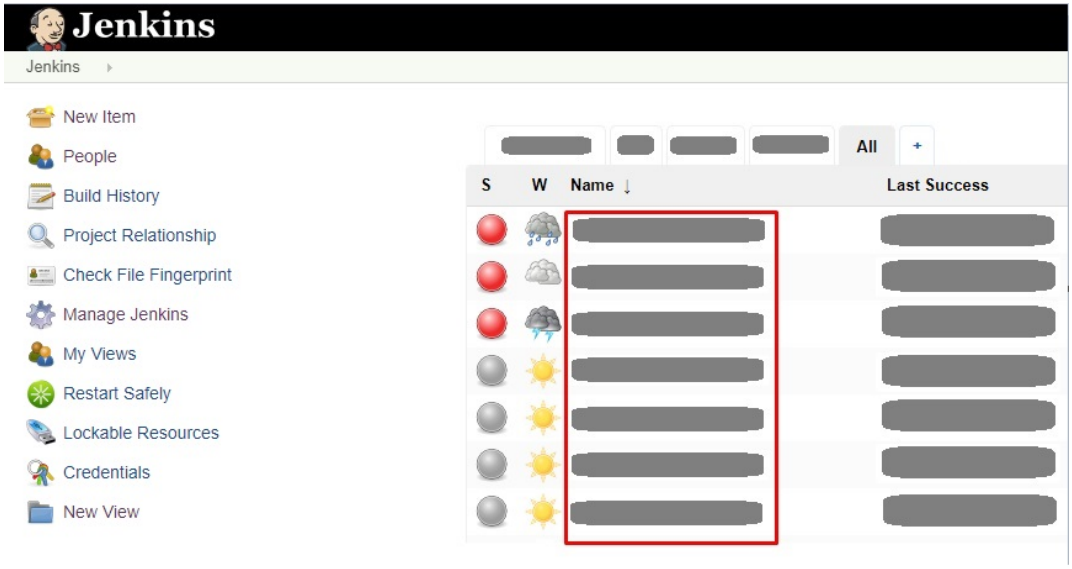


Рис. 6.73: Jenkins: Выбор Item'а

2. Для просмотра PDF-отчёта нажмите **appScreener PDF report** (**Pdf-отчёт appScreener**) или скачайте по ссылке в результатах сканирования (шаг 5).

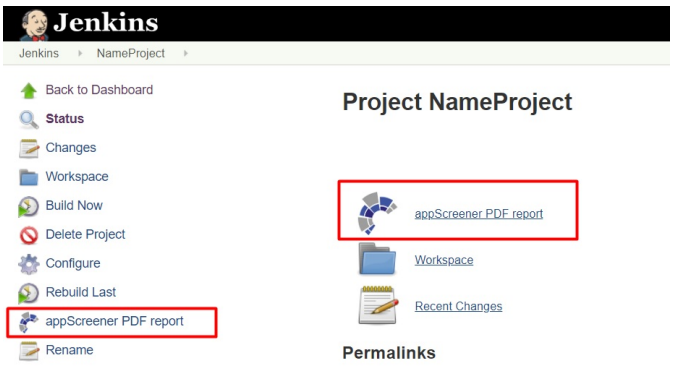


Рис. 6.74: Jenkins: PDF отчёт

3. Нажмите на нужный номер сборки в блоке **Build History** (**История сборок**) располагающемся под боковым меню.

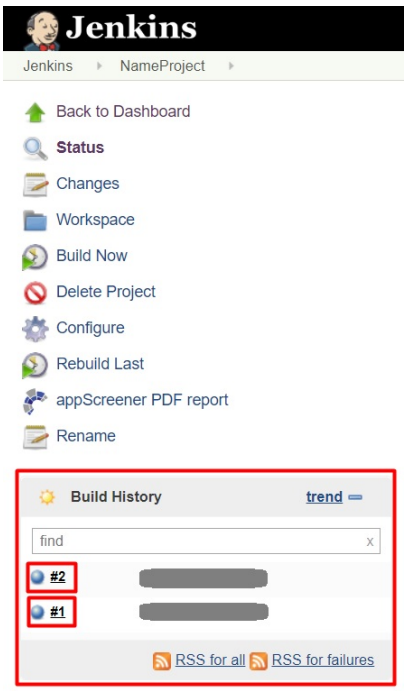


Рис. 6.75: Jenkins: Номер сборки

4. Для просмотра журнала событий (logs) нажмите **Console Output** (**Вывод консоли**) в боковом меню.

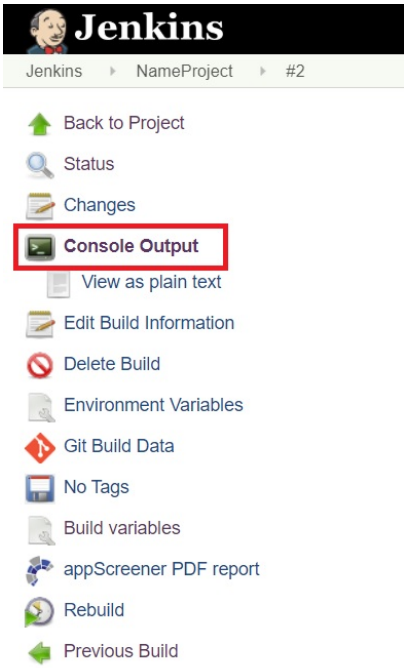


Рис. 6.76: Jenkins: Консоль

5. Для просмотра результатов сканирования нажмите **Build variables** (**Переменные сборки**). В пункте PDF_URL находится ссылка на скачивание PDF-отчёта.

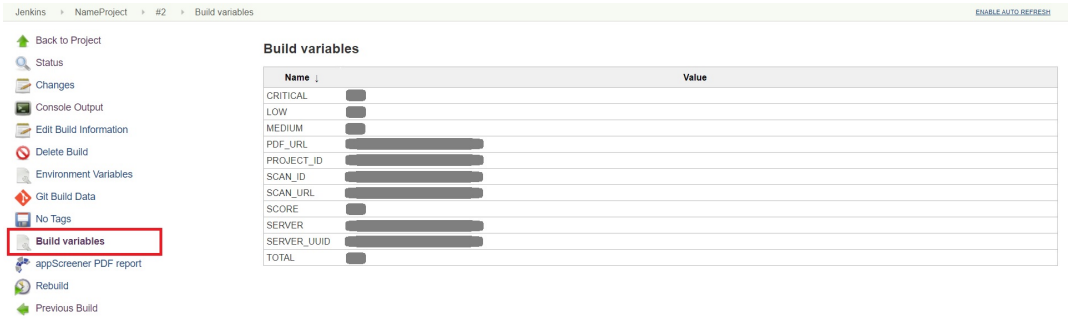


Рис. 6.77: Jenkins: Результаты сканирования

6.5.3. Azure DevOps Server

Solar appScreener поддерживает **Team Foundation Server 2018**, **Azure Devops Server 2019**, **Azure Devops Server 2020** и **Azure DevOps Services**.

6.5.3.1. Установка расширения

1. Перейдите на страницу управления расширениями (**http://<installation_address>/_gallery**, **<installation_address>** адрес машины, на которой установлен Azure DevOps Server).
2. Нажмите на кнопку **Manage Extensions** (**Управление расширениями**) в правой нижней части страницы.

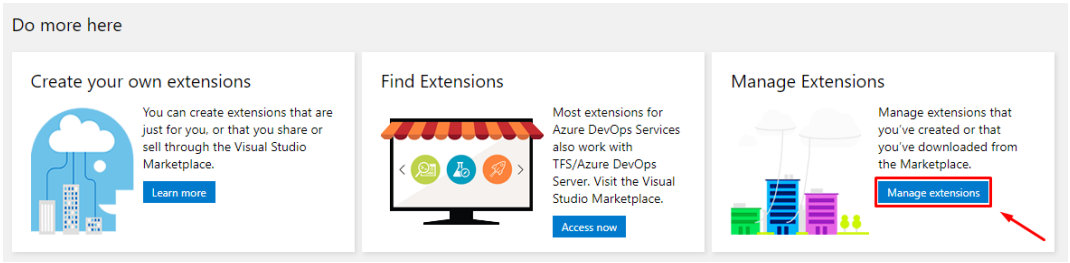


Рис. 6.78: Azure DevOps Server: Управление расширениями

3. Нажмите на кнопку **Upload extension** (**Загрузить расширение**).

Рис. 6.79: Azure DevOps Server: Отправить новое расширение

4. Выберите файл расширения в открывшемся всплывающем окне.
5. Нажмите на кнопку **Upload** (**Загрузить**).

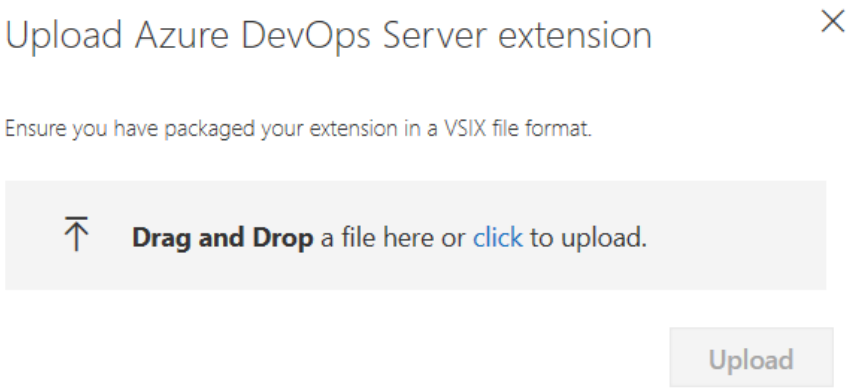


Рис. 6.80: Azure DevOps Server: Отправка

6. Выберите из списка расширение appScreener.

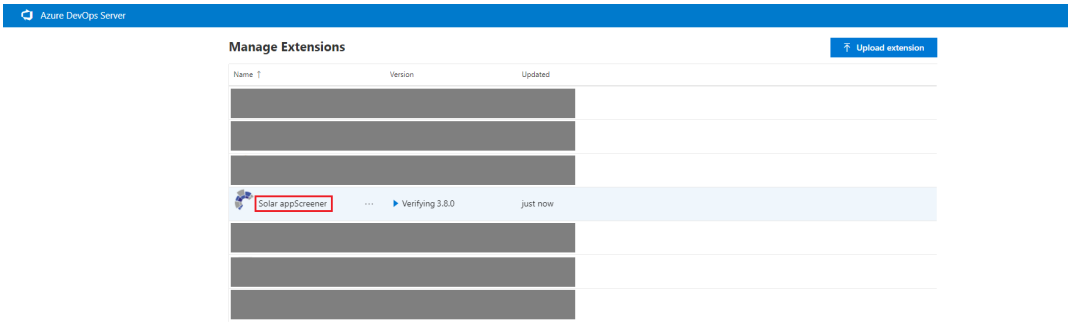


Рис. 6.81: Azure DevOps Server: Выбор расширения

7. Нажмите на кнопку **Get it free** (**Получить бесплатно**) и выберите коллекцию, для которой хотите установить расширение, например **DefaultCollection**.

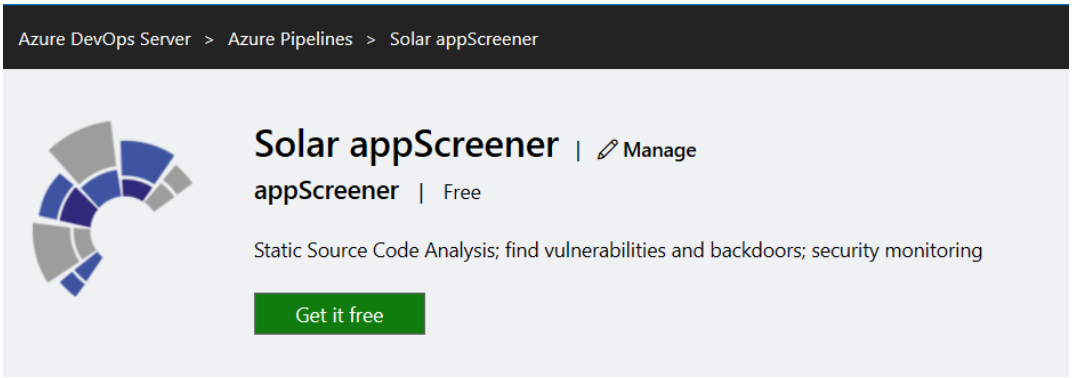


Рис. 6.82: Azure DevOps Server: Установка

8. Нажмите на кнопку **Install** (**Установка**).

   Если все шаги выполнены, на странице отобразится текст **You are all set!** (**Все готово**). В любом проекте выбранной коллекции будет доступна задача как шаг определения сборки.

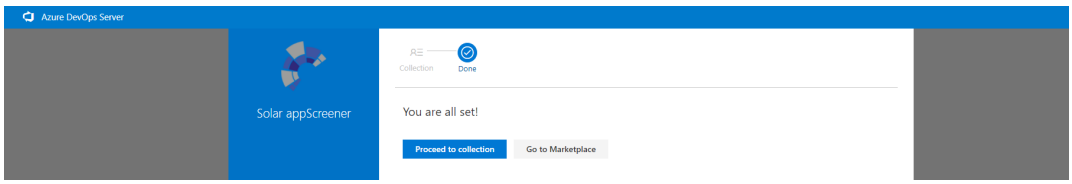


Рис. 6.83: Azure DevOps Server: Подтверждение

6.5.3.2. Добавление шага для сборки в Azure DevOps Server

1. Перейдите на страницу нужного проекта коллекции, для которого было установлено расширение (**http://<installation_address>/DefaultCollection/projectexample**, **<installation_address>** адрес машины, на которой установлен Azure DevOps Server).
2. Перейдите по пути **Pipelines** -> **Pipelines** (**Сборка и выпуск** -> **Сборки**).

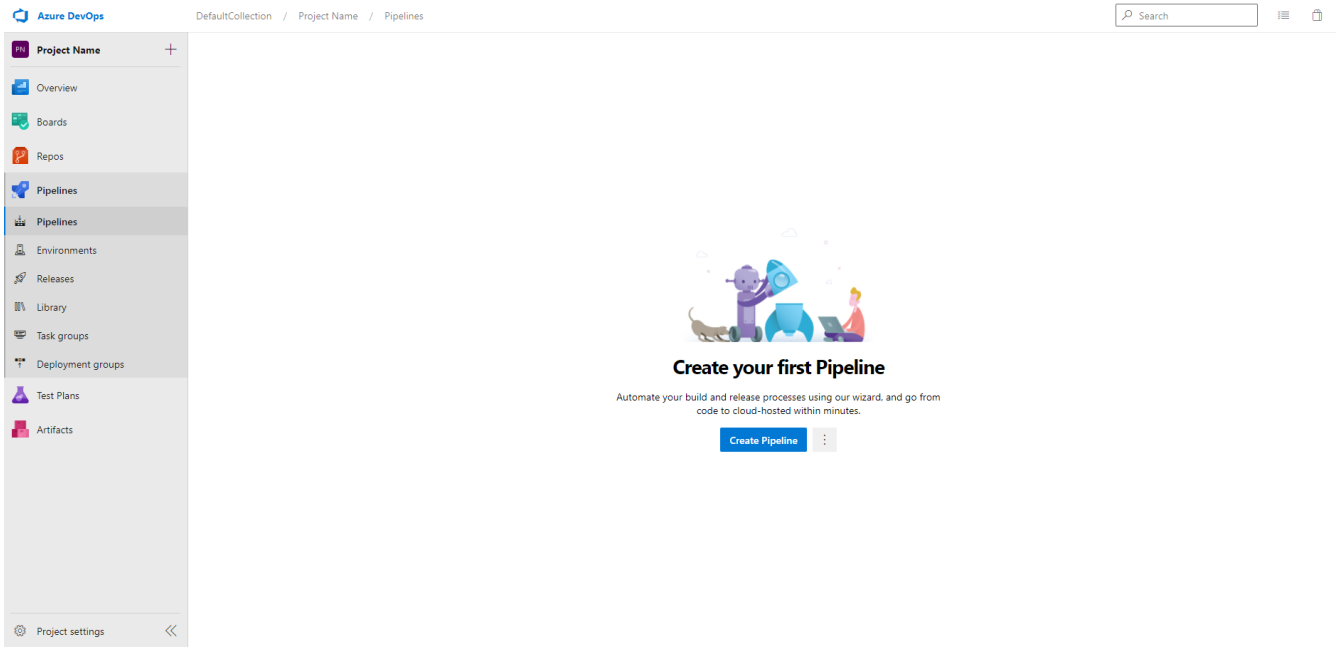


Рис. 6.84: Azure DevOps Server: Сборка и выпуск

3. Нажмите на три точки и нажмите **Edit** (**Изменить**) для имеющегося определения сборки или создайте новое, нажав **Create Pipeline** (**Создать сборку**) и выбрав шаблон. Чтобы включить автоматическую настройку UUID соответствующего проекта appScreener:
   1. Нажмите **Admin settings**.
   2. Перейдите в раздел **Security**.
   3. Нажмите **Project collection build services**.
   4. Выберите значение **Allow** напротив пункта **Edit build definition**.
   5. Нажмите **Save changes**.
4. Нажмите **Add Task** (**Добавить задачу**).

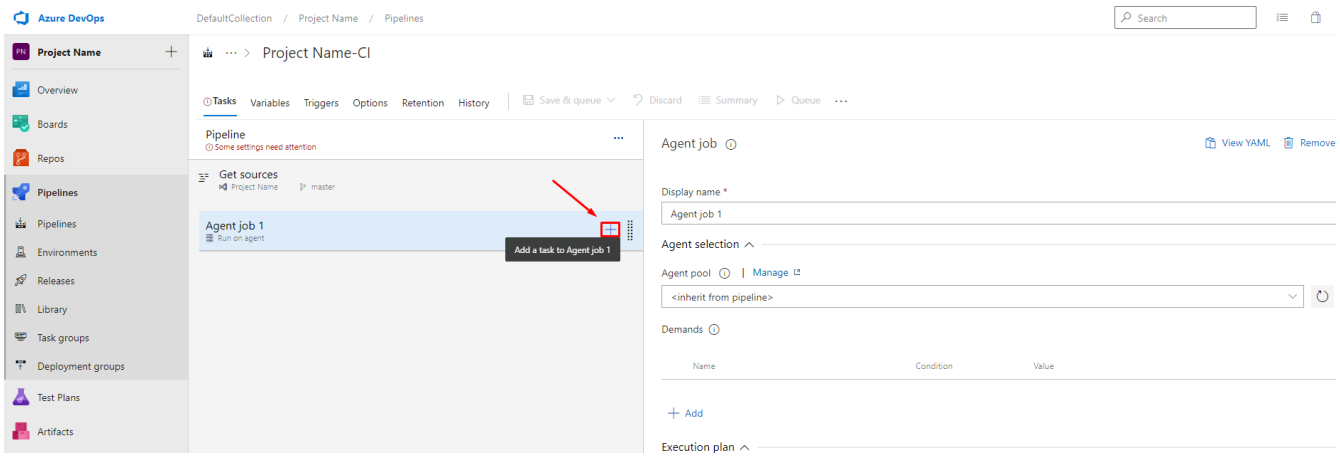


Рис. 6.85: Azure DevOps Server: Добавление задачи

5. Найдите **Run Solar appScreener SAST** и нажмите **Add** (**Добавить**).

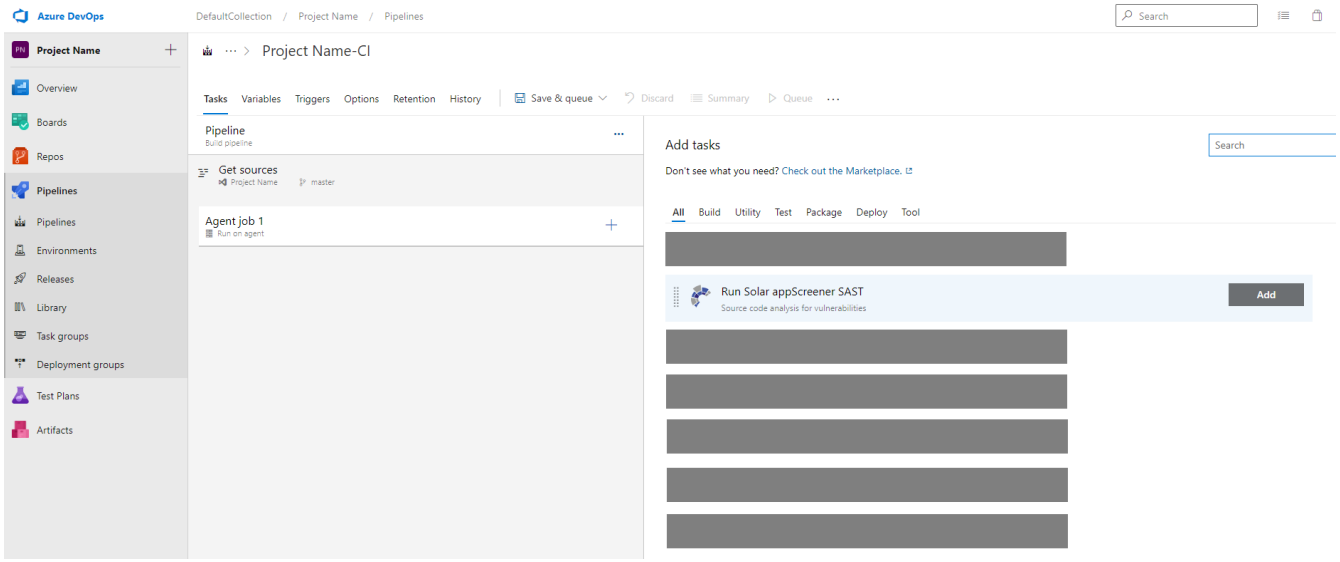


Рис. 6.86: Azure DevOps Server: Выбор задачи

6. Выберите добавленный шаг сборки.
7. Добавьте подключение к серверу appScreener из списка или создайте новое:
    1. Справа от **appScreener server end point** нажмите на кнопку **New** (**Создать**).

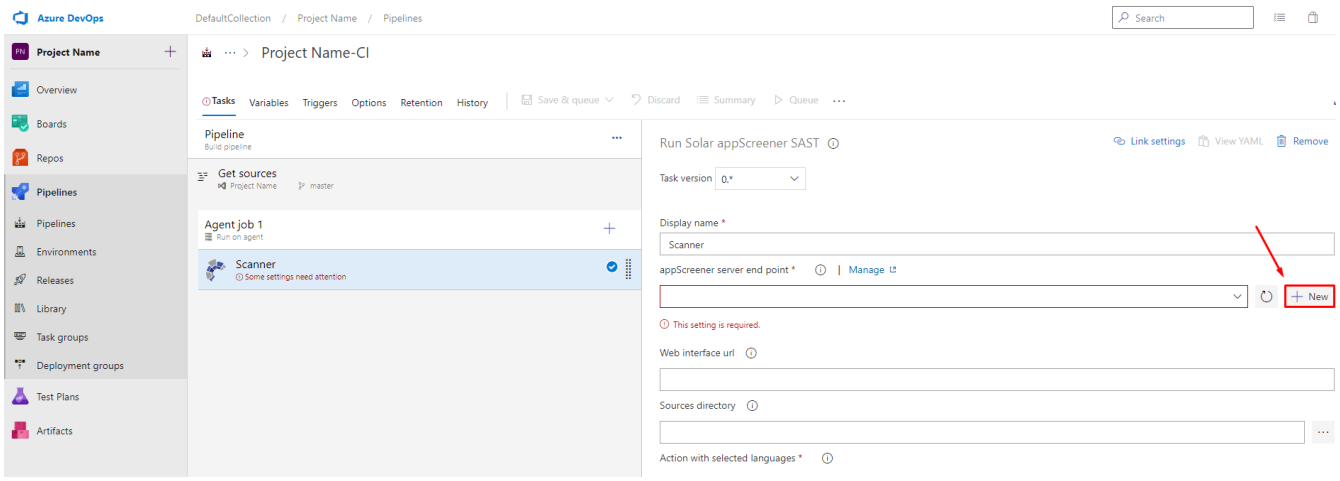


Рис. 6.87: Azure DevOps Server: Редактировать задачу

2. В появившемся окне введите адрес API (например, **http://<installation_address>/app/api/v1/ <installation_address>** адрес машины, на которой установлен appScreener) и токен. Токен можно получить в разделе **Личный кабинет** (при получении токена рекомендуется установить длительное время действия токена).

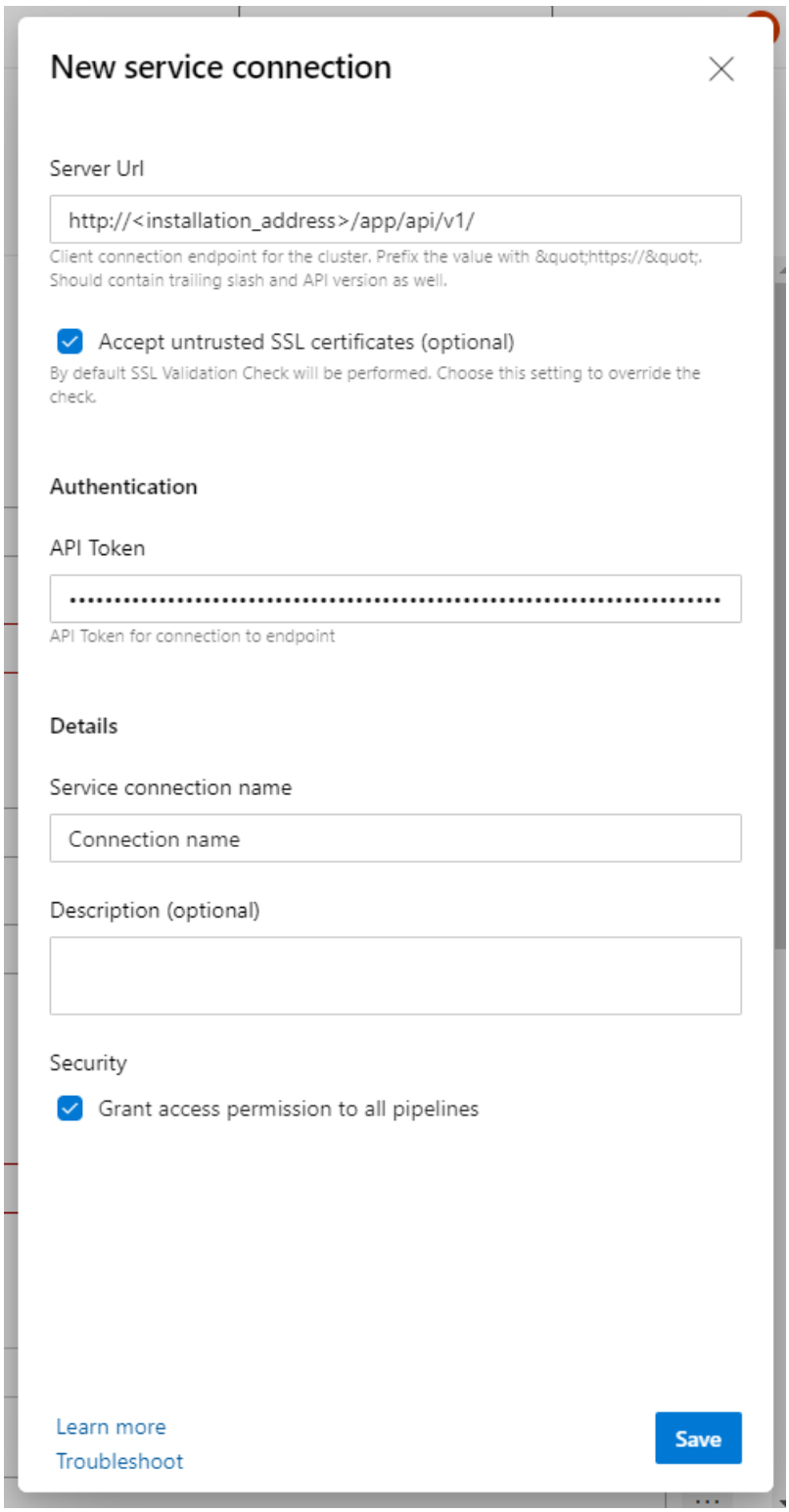


Рис. 6.88: Azure DevOps Server: Добавление подключения

3. Нажмите **Save**.

8. Настройте необходимые параметры анализа. Подробнее о настройках анализа в разделе Общие.

9. Укажите дополнительные параметры в меню **General analysis settings**: **Use extra rules** (**Использовать дополнительные правила**), **Incremental analysis** (**Инкрементальный анализ**), **Analyze libraries and nested archives** (**Анализировать**

библиотеки и вложенные архивы).

10. Настройте **Failure Conditions**:
11. В разделе **Task failure conditions** активируйте опцию **Enable failing on condition**.
12. Установите **Failure Conditions** в зависимости от значений (Score condition, Critical issues number condition, Medium issues number condition, Low issues number condition, Info issues number condition).

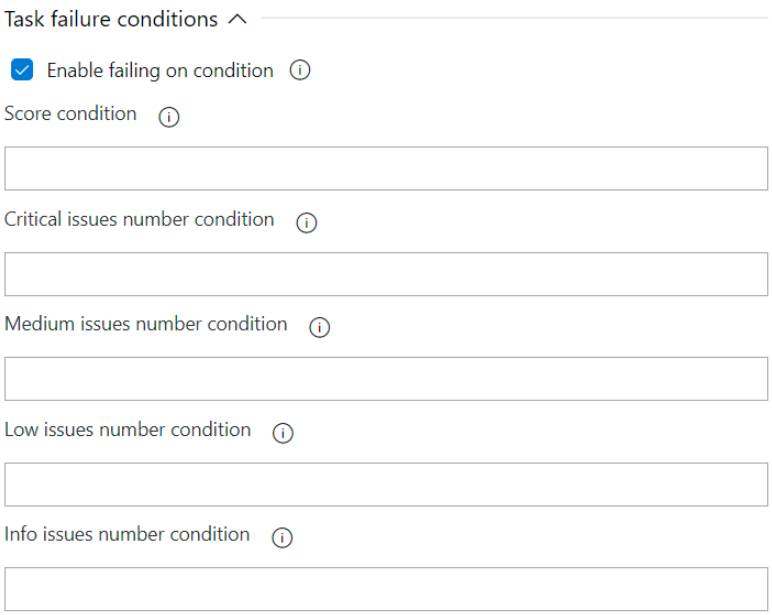


Рис. 6.89: Azure DevOps Server: Failure conditions

13. Нажмите в верхнем правом углу **Save and queue** (**Сохранить и поместить в очередь**) и затем ещё раз **Save and queue** (**Сохранить и поместить в очередь**).

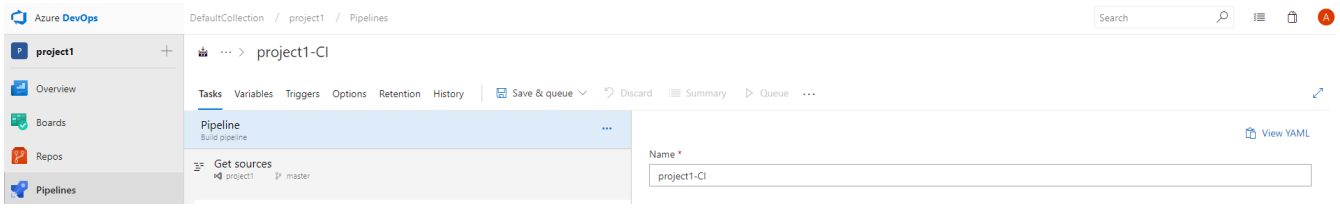


Рис. 6.90: Azure DevOps Server: Сохранение

10. Дождитесь окончания сборки и перейдите на страницу результатов.
11. Чтобы скачать отчёт, перейдите во вкладку **appScreener code analysis results** и откройте ссылку на отчёт в новой вкладке браузера.

6.5.4. TeamCity

Solar appScreener поддерживает **TeamCity 2017.2.2.** и более поздние версии.

6.5.4.1. Интеграция Solar appScreener через плагин

Инструкция по использованию плагина к TeamCity:

1. Перейдите на страницу, на которой установлен TeamCity, например: **http://<installation_address>:8111/teamcity/**
2. Установите плагин: перейдите по пути **Administration->Plugins List->Upload plugin zip**. В некоторых случаях для корректной работы плагина может потребоваться удаление его предыдущих версий.
3. Настройте соединение:
    1. Перейдите по пути **Administration->Integrations->appScreener**.
    2. В разделе **appScreener Static Analysis** введите адрес API (должен заканчиваться слешем, например: **http://<installation_address>/app/api/v1/**, **<installation_address>** адрес машины, на которой установлен appScreener) и токен, который можно получить в веб-интерфейсе appScreener на странице аккаунта в разделе **Токен** (см. раздел Личный кабинет).
    3. Нажмите на кнопку **Test connection** (при успешной проверке подключения появится надпись **Successful**).
    4. Нажмите **Save**.
4. Добавьте **appScreener SAST Build Step** в настройках сборки и укажите для него необходимые настройки.
5. Добавьте **Build Features** (доступны только при наличии **appScreener SAST Build Step**):
    - **appScreener PDF report** настройте экспорт отчёта с результатами сканирования (см. раздел Экспорт отчёта);
    - **appScreener statistics** включает в себя оценку безопасности, количество уязвимостей каждого уровня критичности, продолжительность сканирования и количество строк кода (посмотреть статистику можно в **Build->Parameters->Reported statistic values**);
6. Настройте **Failure Conditions**:
    1. В настройках сборки нажмите **Failure Conditions**.
    2. Нажмите **Add failure condition** и выберите **Fail build on metric change**.
    3. Установите **Failure Conditions** в зависимости от значений **appScreener statistics** (appScreener LOC; info, low, medium, critical vulnerabilities; scan duration; score).

6.5.4.2. Интеграция Solar appScreener с помощью Command Line Runner

Для интеграции Solar appScreener в процесс, обеспечиваемый TeamCity, с помощью The Command Line Runner необходимо:

1. С помощью настройки существующей конфигурации **Build** добавить **Build Step** с типом **Command Line**;
2. Разработать скрипт запуска сканирования в Solar appScreener:

- параметры проекта и окружения доступны через конструкцию вида %key% (список параметров);
- пример скрипта запуска сканирования:

```
java -jar <path>/clt.jar -rest http://<installation_address>/app/api/v1
-token xxx -name TeamCityCLTTest -languages JAVA -type FILE -path
%teamcity.build.checkoutDir%
```

При создании **Build Step** с типом **Command Line** рекомендуется использовать **Custom script** в качестве **Run**, а не **Executable with parameters**.

В следующем примере используется **Build Step PowerShell**. Данный скрипт запускает сканирование, получает его идентификатор и ожидает его завершения. После завершения сканирования можно скачать PDF-отчёт, используя CLI.

```
$out = java -jar <path>\clt.jar -rest
http://<installation_address>/app/api/v1 -token xxx
-name TeamCityTest -languages JAVA -type FILE -path
%teamcity.build.checkoutDir%;
$sep = "ScanUuid: ";
$line = $out.Where{$_.Contains($sep)}.Item(0);
$splitted = $line -split $sep;
$id = $splitted[1];
For ($i=0; $i -lt 12; $i++) {
    $out = java -jar <path>\clt.jar -rest
    http://<installation_address>/app/api/v1 -token xxx
    -cmd status -scanid $id;
    $sep= "Status: ";
    $line = $out.Where{$_.Contains($sep)}.Item(0);
    $splitted = $line -split $sep;
    $status = $splitted[1];
    write-host $( '##teamcity[message text=''{0}'']' -f $status );
    if ($status -eq "Scan completed"){
        break;
    }
    Start-Sleep -s 5;
}
java -jar <path>\clt.jar -rest http://<installation_address>/app/api/v1
-token xxx -cmd export -scanid $id –path <path> -default;
```

6.5.5. JSON API

JSON API предоставляет возможность выгружать результаты сканирования и информацию о найденных уязвимостях в формате JSON. Для использования API Solar appScreener необходима предварительная настройка. Инструкция по настройке API доступна в руководстве администратора. Для того чтобы ознакомиться со спецификацией:

1. Перейдите на страницу **Личный кабинет** (см. раздел Личный кабинет).
2. Получите токен авторизации API.
3. Нажмите **Спецификация API**.
4. Введите токен авторизации API.
5. Нажмите **Explore**, после чего появится список возможных запросов.

Для того чтобы сделать запрос:

1. Нажмите **Authorize** и повторно введите токен авторизации.
2. Нажмите на требуемый запрос.
3. Нажмите **Try it out**.
4. При необходимости укажите параметр.
5. Нажмите **Execute**.

После выполнения этих действий на экране появится соответствующий cURL-запрос и ответ в формате JSON.

Спецификация API в Solar appScreener отвечает стандарту OpenAPI. Спецификацию API можно использовать для генерации клиентских и серверных библиотек доступа к JSON API с помощью инструментальных средств **OpenAPI Generator** или **Swagger Codegen**.

Доступ к спецификации в формате JSON может быть получен по протоколу HTTP с использованием GET-запроса **/app/api/v1/openapi.json** по токену авторизации API. Например, для доступа к спецификации в формате JSON можно использовать следующую команду cURL:

```
curl -H 'Authorization: Bearer <token>'
<installation_address>/app/api/v1/openapi.json -o openapi.json
```

В результате исполнения команды в текущей директории будет сохранён файл **openapi.json**, который далее может быть передан инструментальному средству генерации библиотек доступа к JSON API. Например, для генерации Java-библиотеки доступа к JSON API с использованием OpenAPI Generator можно использовать следующую команду:

```
java -jar openapi-generator-cli.jar generate -g java -i openapi.json
```

Более подробная информация о стандарте OpenAPI и возможностях инструментальных средств генерации библиотек доступа к API может быть получена в официальной документации.

Ссылки:

- спецификация OpenAPI;
- OpenAPI Generator;
- Swagger Codegen;
- другие реализации и инструментальные средства.

6.5.6. IntelliJ IDEA

6.5.6.1. Установка плагина

1. В IDEA перейдите в **File -> Settings -> Plugins** (для macOS **IntelliJ IDEA -> Preferences -> Plugins**).
2. Нажмите на шестеренку в правом верхнем углу.

Рис. 6.91: Настройка конфигурации

3. Выберите **Install Plugin from Disk...**

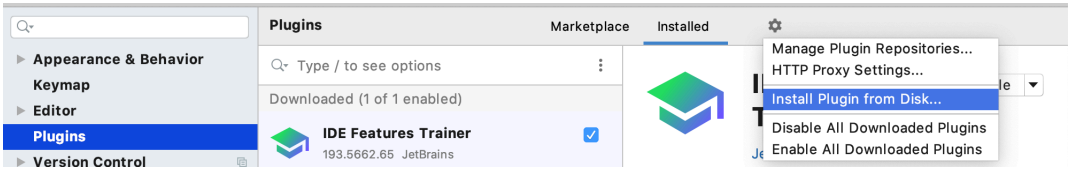


Рис. 6.92: Выбор плагина

4. Укажите путь до архива.

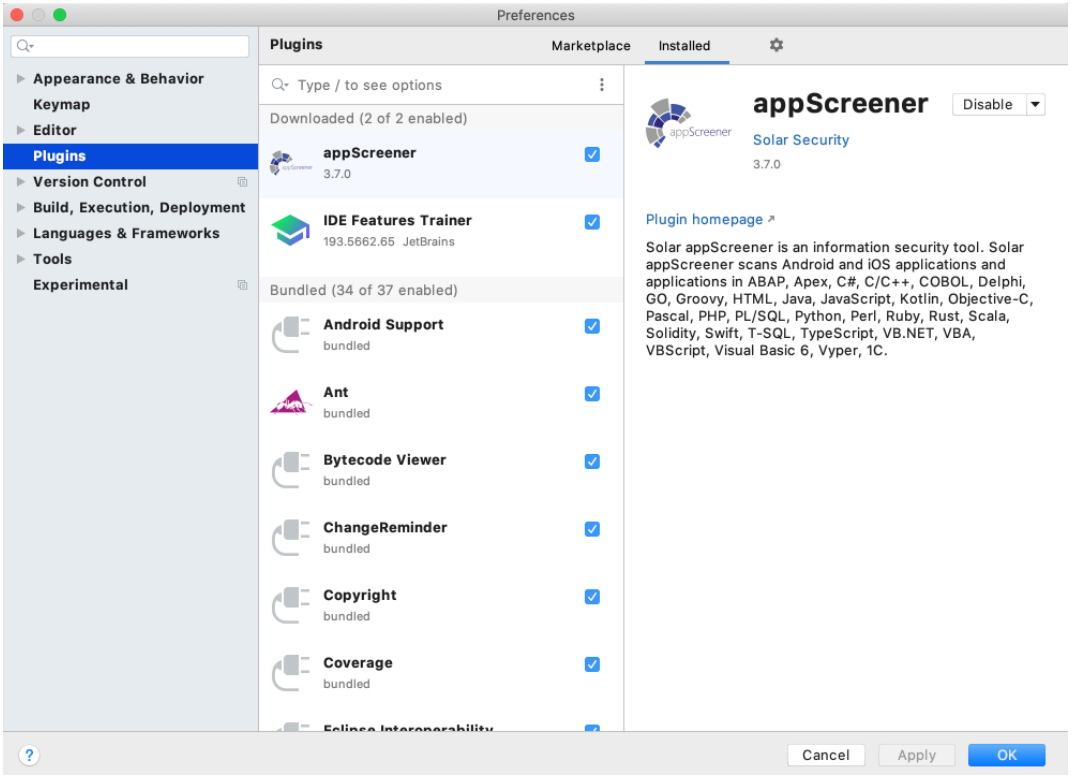


Рис. 6.93: Установленный плагин

6.5.6.2. Использование

В IDEA создайте новую конфигурацию: нажмите **Add Configuration** или **Edit Configurations**, если в проекте уже есть добавленные конфигурации. Выберите из списка шаблонов appScreener и нажмите **+**.

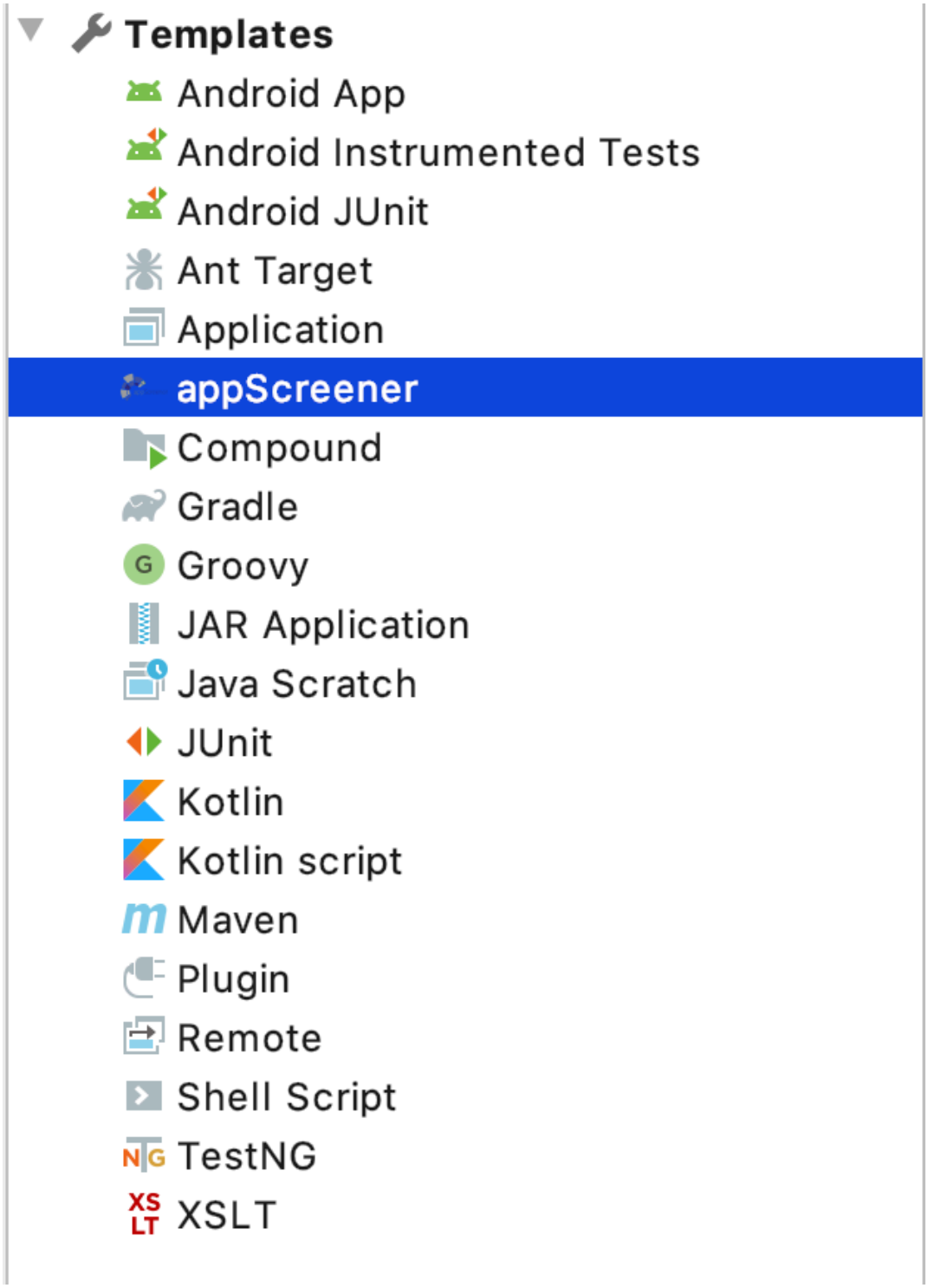


Рис. 6.94: Добавить конфигурацию

В добавленной конфигурации укажите значения:

- **Project path** путь до проекта, который нужно проанализировать;
- **Project name** название проекта;
- **CLT path** путь до **clt.jar**;
- **Token** токен авторизации, можно получить в **Личном кабинете**;

- **URL** путь до установки: **http://<installation_address>/app/api/v1/**;
- Нажмите **OK**.

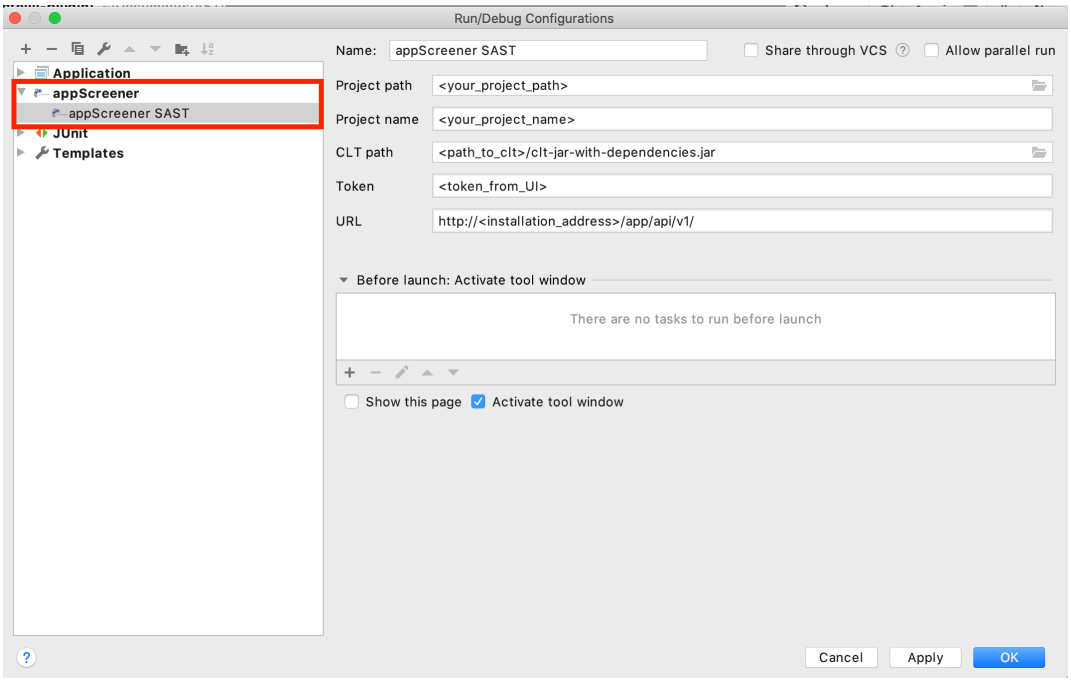


Рис. 6.95: Значения конфигурации

6.5.6.3. Запуск

1. Из списка конфигураций выберите созданную для **appScreener**:

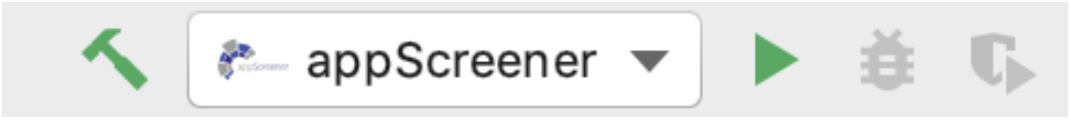


Рис. 6.96: Выбор конфигурации

2. Нажмите **Пуск**.

6.5.7. Eclipse

6.5.7.1. Установка плагина

1. Для установки плагина завершите работу среды разработки Eclipse.
2. Поместите файл плагина из архива (файл с расширением `.jar`) в директорию **eclipse_directory/dropins/plugins** (**eclipse_directory** — путь, по которому установлен Eclipse). Если есть старая версия плагина, её необходимо удалить.
3. Перезагрузите Eclipse. После перезагрузки (первая загрузка может проходить долго, 3-5 минут) в тулбаре появится пункт **appScreener**.

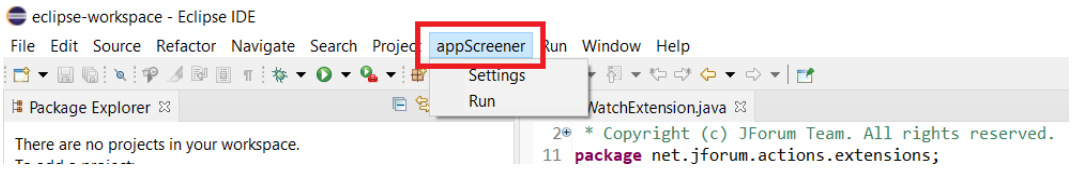


Рис. 6.97: Eclipse: установка

6.5.7.2. Использование

1. В тулбаре выберите **appScreener**.
2. Выберите пункт **Settings**:
    1. Укажите путь до директории проекта, который необходимо отправить на анализ (либо выделите нужный проект в **Project Explorer**).
    2. Укажите путь к файлу **clt.jar** в **CLT path** (вместе с названием файла, пример: `C:\ipr\jas\clt\target\clt-jar-with-dependencies.jar`).
    3. Введите токен, его можно получить в интерфейсе appScreener в разделе **Личный кабинет** (**Account**).
    4. Укажите название проекта (опционально).

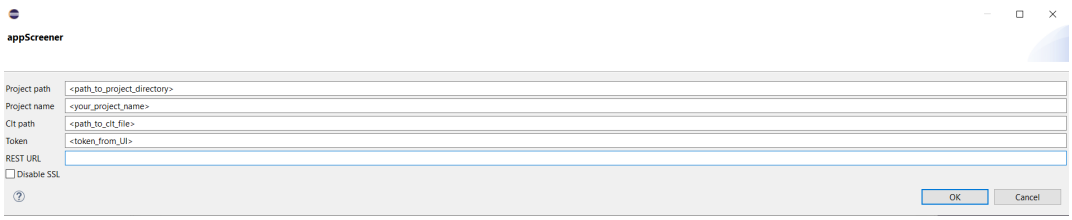


Рис. 6.98: Eclipse: настройки

6.5.7.3. Запуск

1. Перейдите в пункт меню **appScreener**.
2. Запустите сканирование, нажав **Run**. Результаты сканирования можно просмотреть в интерфейсе appScreener.

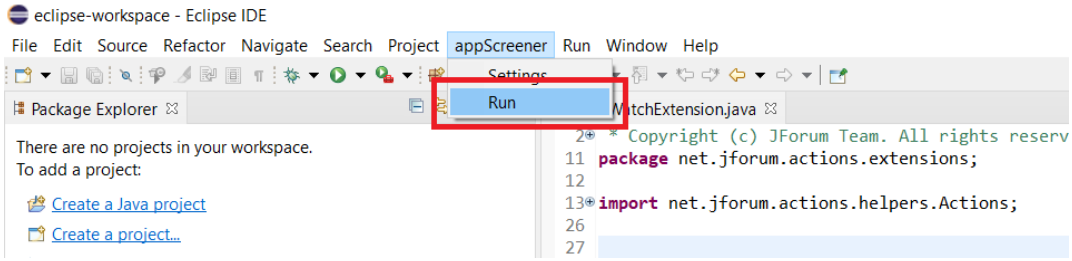


Рис. 6.99: Eclipse: запуск

6.5.8. Visual Studio

6.5.8.1. Установка расширения

1. Откройте расширение с помощью Visual Studio.

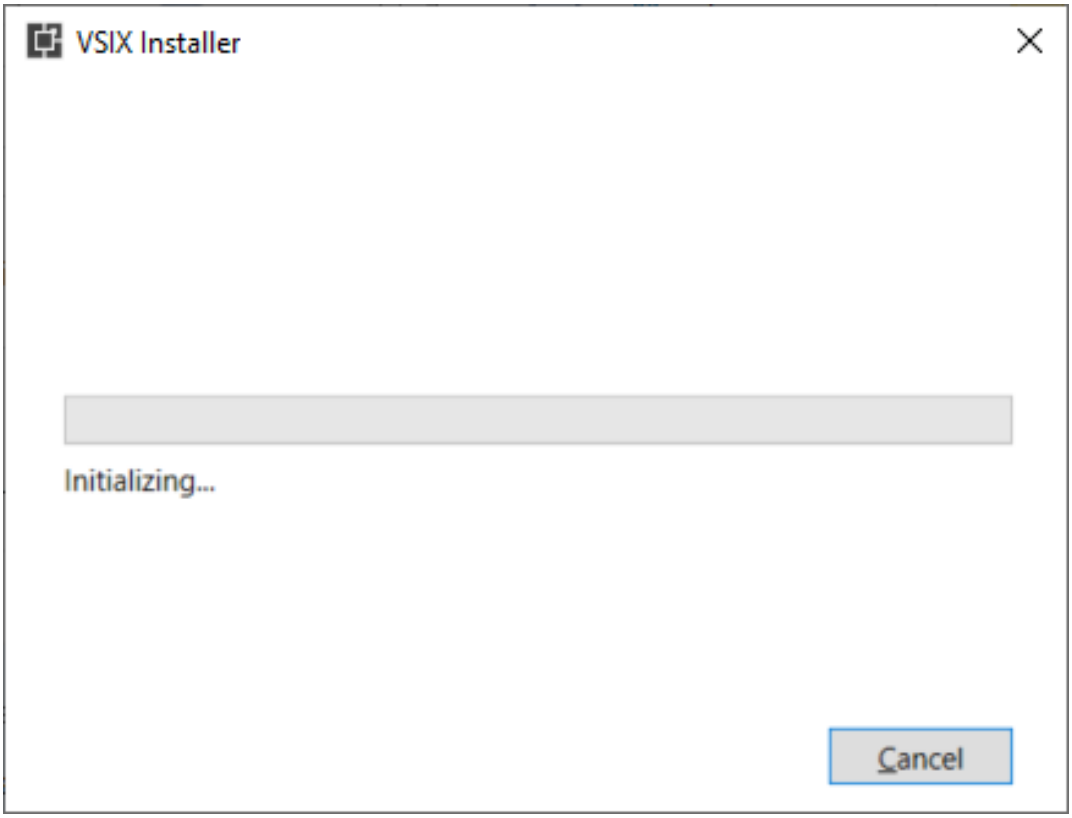


Рис. 6.100: Visual Studio: Запуск

2. Выберите IDE установки.

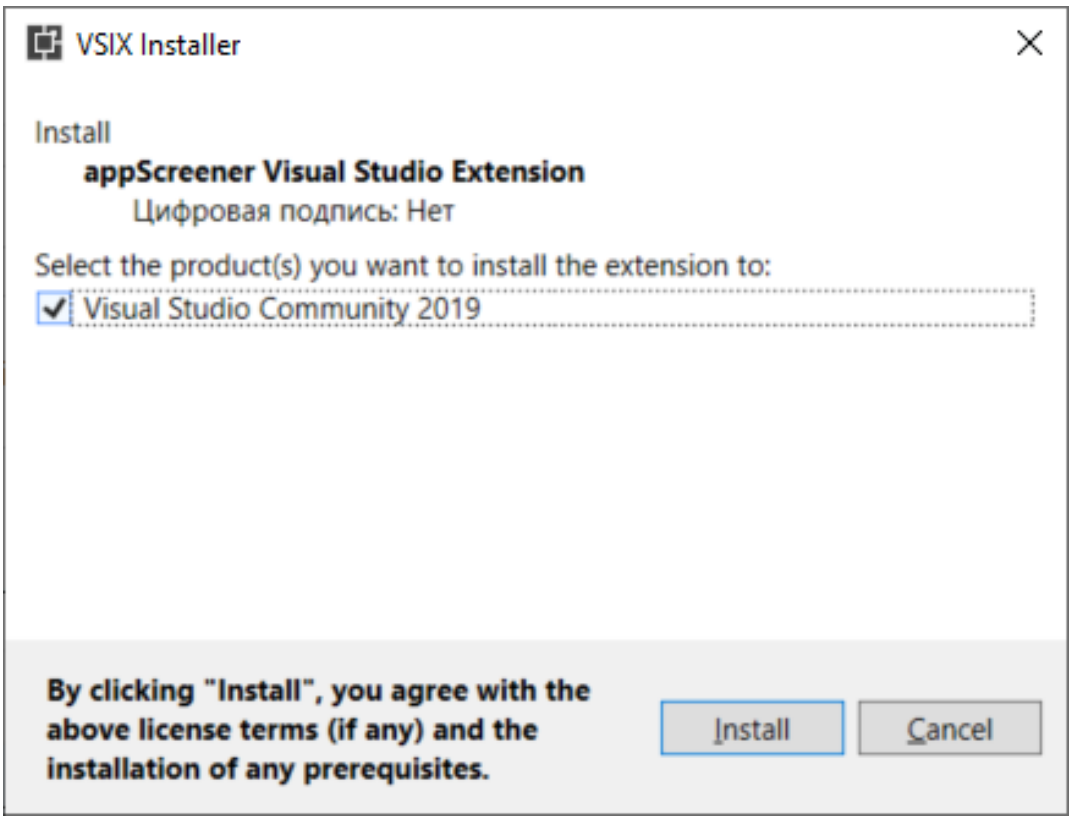


Рис. 6.101: Visual Studio: Выбор IDE

3. Дождитесь завершения.

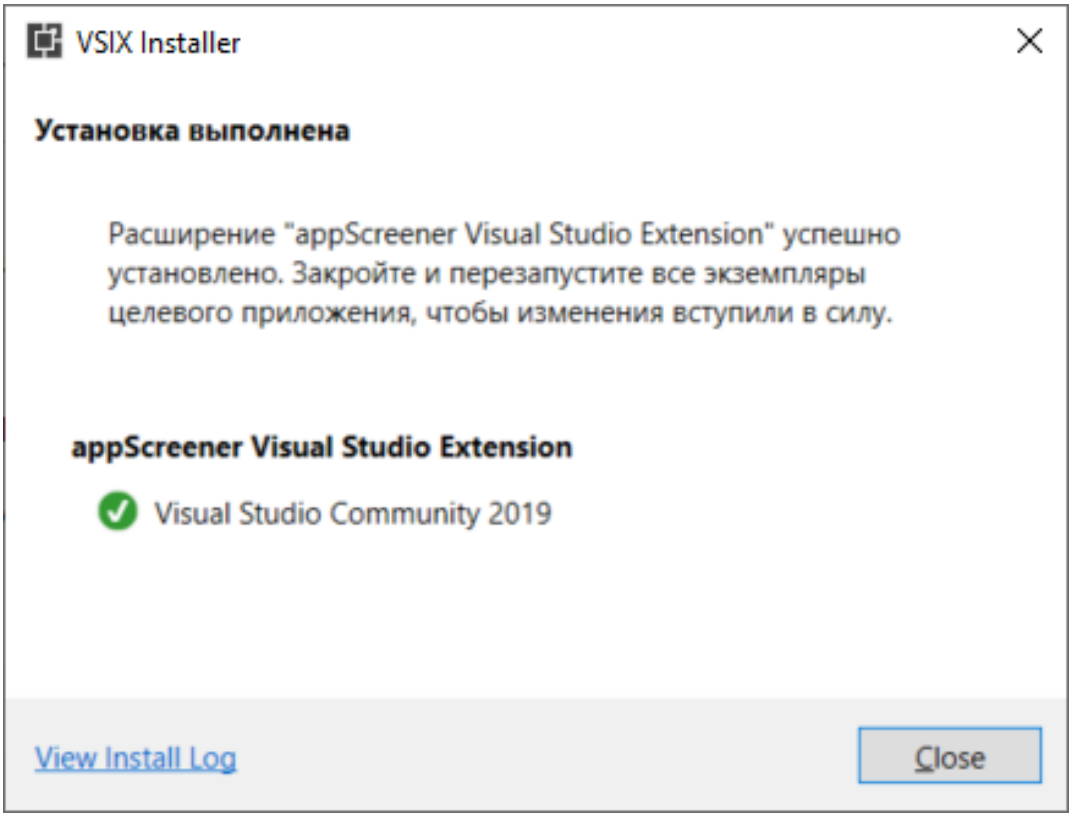


Рис. 6.102: Visual Studio: Установка расширения

6.5.8.2. Использование

1. Запустите Visual Studio, откройте любой проект, создайте новый или продолжите без проекта.

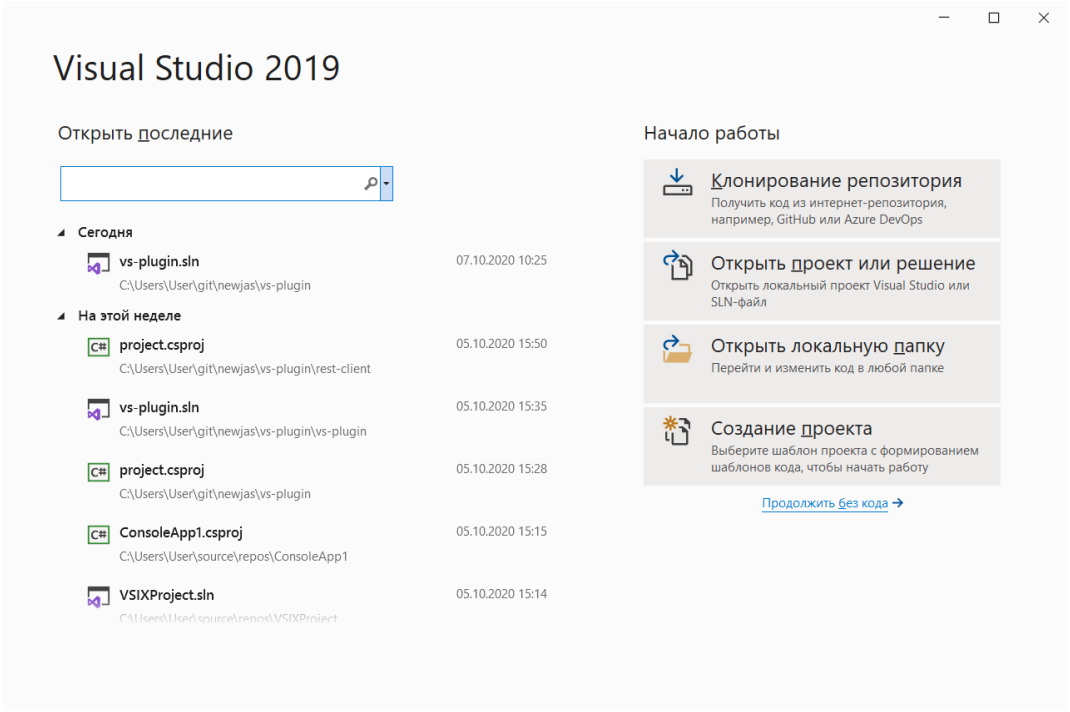


Рис. 6.103: Visual Studio: Открытие проекта

2. Перейдите в параметры.

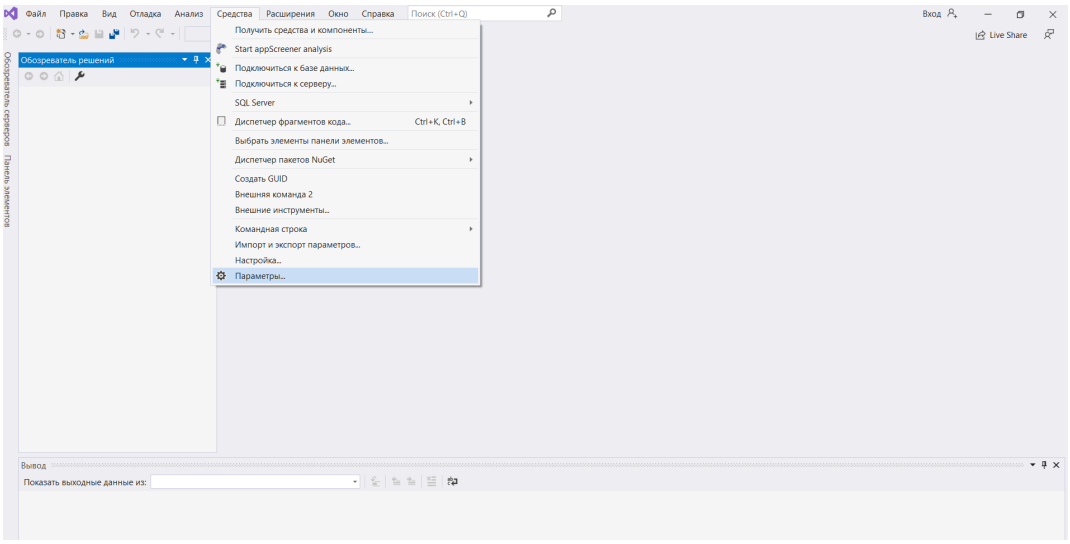


Рис. 6.104: Visual Studio: Параметры

3. Выберите **appScreener Settings**.

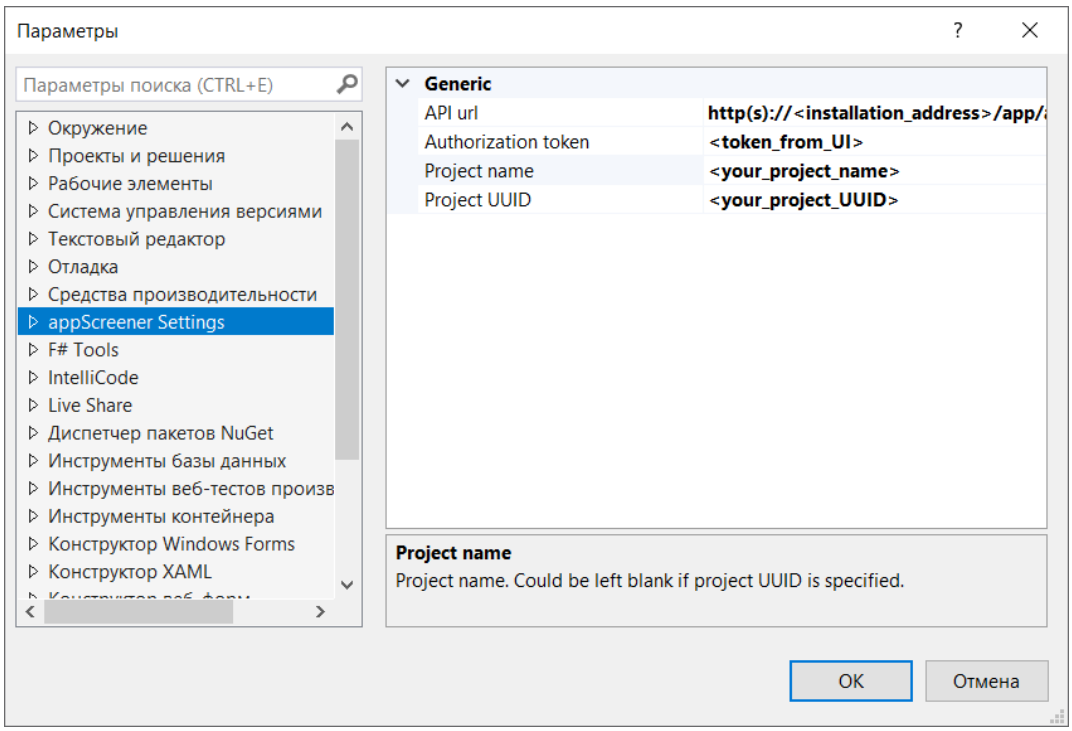


Рис. 6.105: Visual Studio: Выбор настроек

4. Заполните необходимые поля:
   - введите appScreener URL, например **http://<installation_address>**, **<installation_address>** адрес машины, на которой установлен appScreener;
   - введите токен, его можно получить в интерфейсе appScreener в разделе **Личный кабинет** (**Account**);
   - введите название проекта (опционально) и его UUID — идентификатор проекта в **appScreener**, соответствующего проекту в **VisualStudio**.

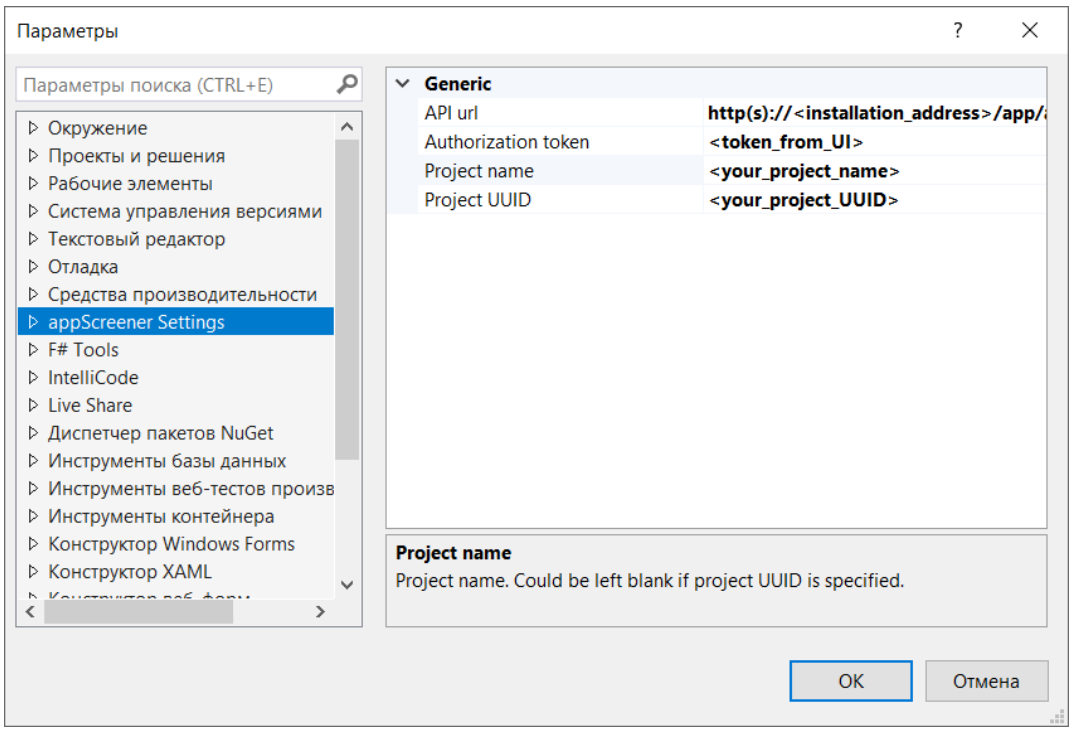


Рис. 6.106: Visual Studio: Параметры

5. Чтобы запустить сканирование, необходимо предварительно создать проект либо открыть существующий.

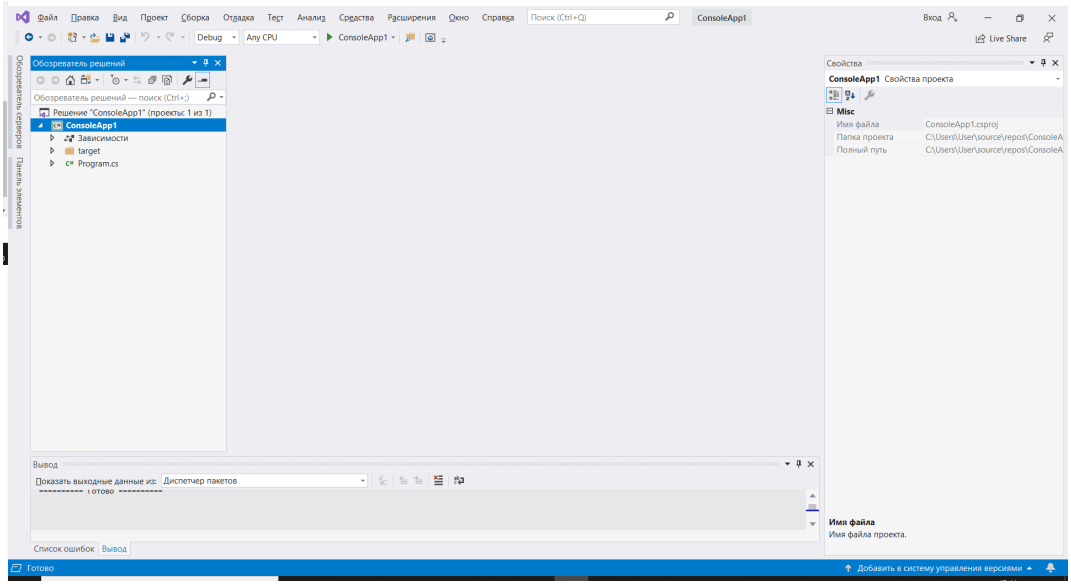


Рис. 6.107: Visual Studio: Выбор проекта

6. Запуск сканирования осуществляется выбором **Start appScreener analysis**.

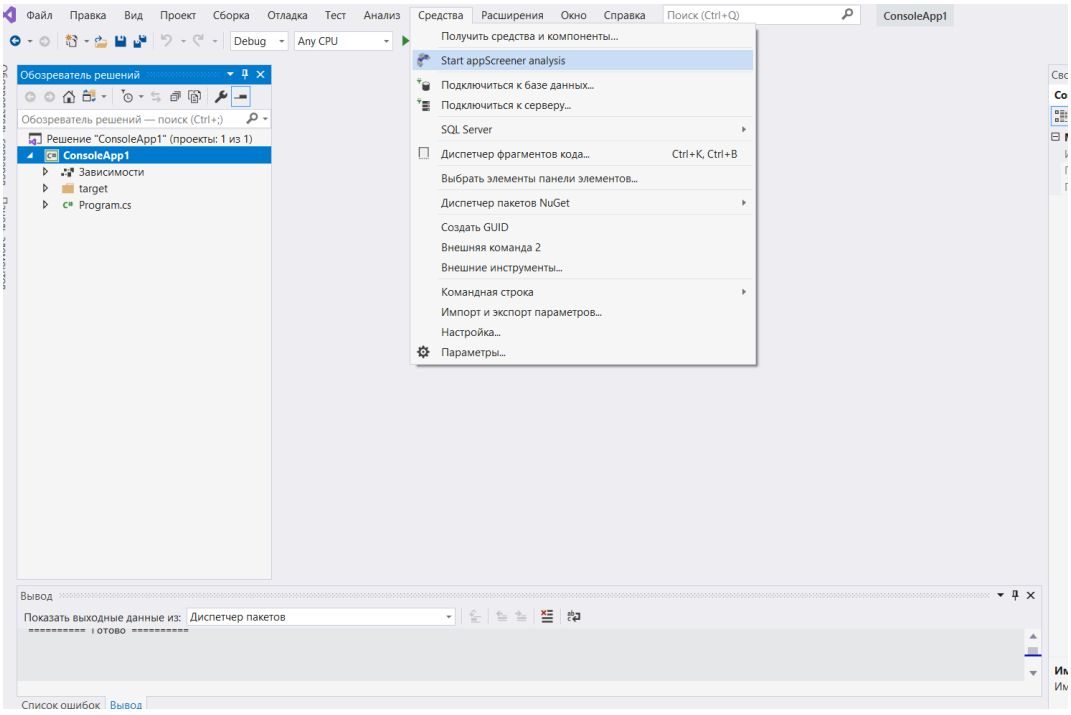


Рис. 6.108: Visual Studio: Запуск анализа

6.5.9. SonarQube

Solar appScreener поддерживает **SonarQube 7.9.** или более поздней версии.

6.5.9.1. Установка расширения

1. Остановите сервер SonarQube.
2. Удалите предыдущие версии расширения из папки **$SONARQUBE_HOME/extensions/plugins**.
3. Поместите `.jar`-файл расширения в папку **$SONARQUBE_HOME/extensions/plugins**.
4. Запустите сервер SonarQube.

6.5.9.2. Настройки расширения

Расширение конфигурируется тремя параметрами:

- **Authentication token** токен авторизации, можно получить на странице профиля в **appScreener** (см. раздел Токен);
- **Project UUID** идентификатор проекта в **appScreener**, соответствующего проекту в **SonarQube**;

  UUID проекта можно получить в боковом меню проекта в интерфейсе appScreener. Справа от логотипа проекта отображается ID (первые шесть символов UUID проекта). Чтобы скопировать в буфер полный UUID, нажмите на иконку копирования.
- **URL** адрес REST API, например **http://<installation_address>/app/api/v1/**, **<installation_address>** адрес машины, на которой установлен appScreener.

Разделяют глобальные настройки и настройки на уровне проекта. Глобальные действуют по умолчанию во всех проектах, если они не переопределены в конкретном проекте.

6.5.9.2.1. Глобальные настройки

1. Перейдите на вкладку **Administration**.
2. В выпадающем списке **Configuration** выберите **General Settings**.
3. Перейдите на вкладку **appScreener Plugin**.

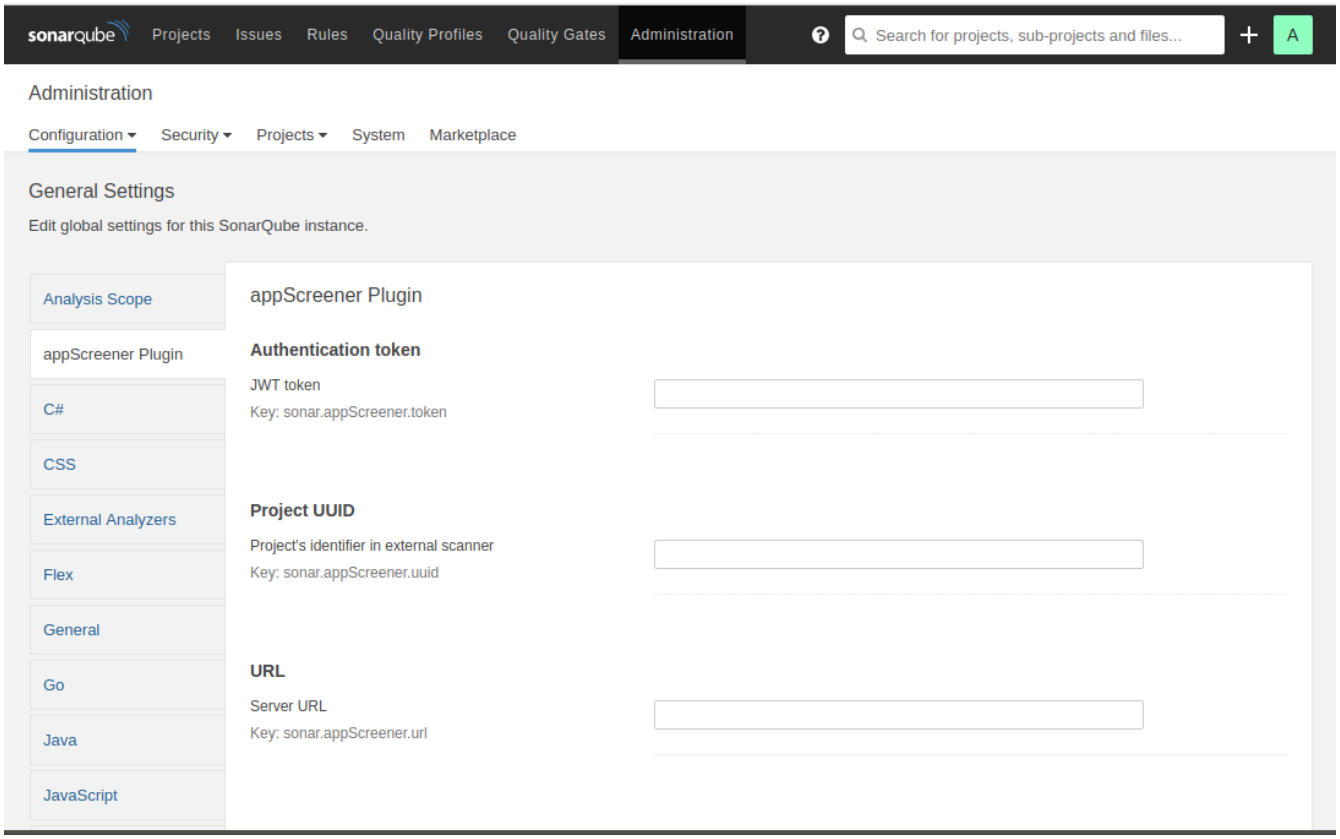


Рис. 6.109: SonarQube: Глобальные настройки

4. Заполните поля ввода необходимыми значениями.
5. Нажмите **Save** для каждого измененного поля.

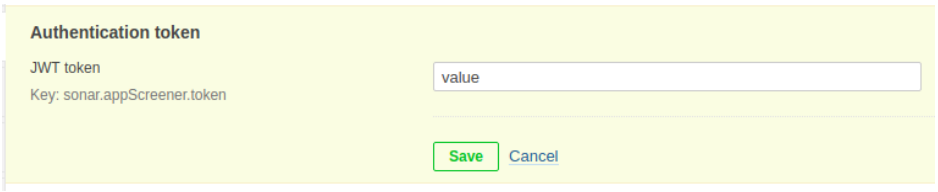


Рис. 6.110: SonarQube: Изменённые поля

6.5.9.2.2. Настройки проекта

1. Перейдите на вкладку **Projects** и выберите проект.
2. На странице проекта в выпадающем списке **Administration** выберите **General Settings**.

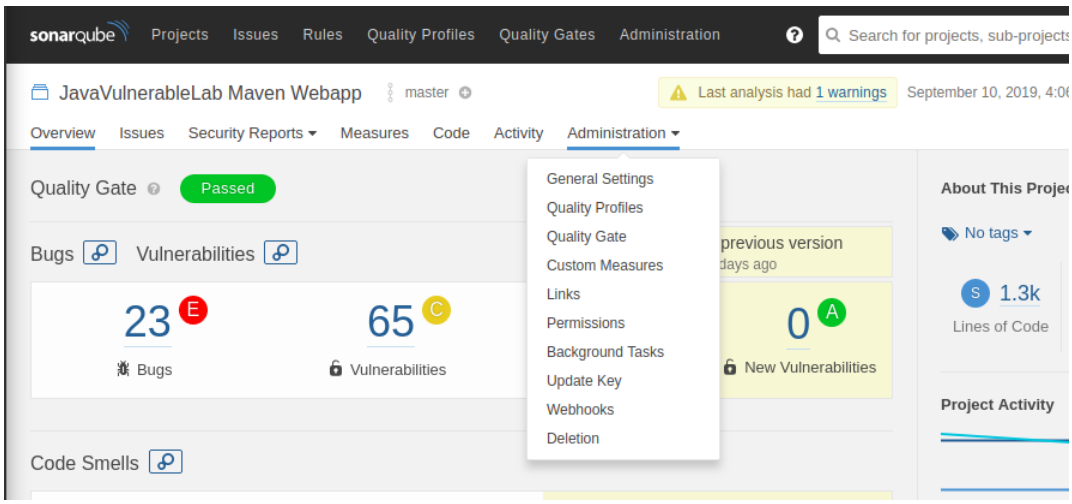


Рис. 6.111: SonarQube: Страница проекта

3. Перейдите на вкладку **appScreener Plugin**.

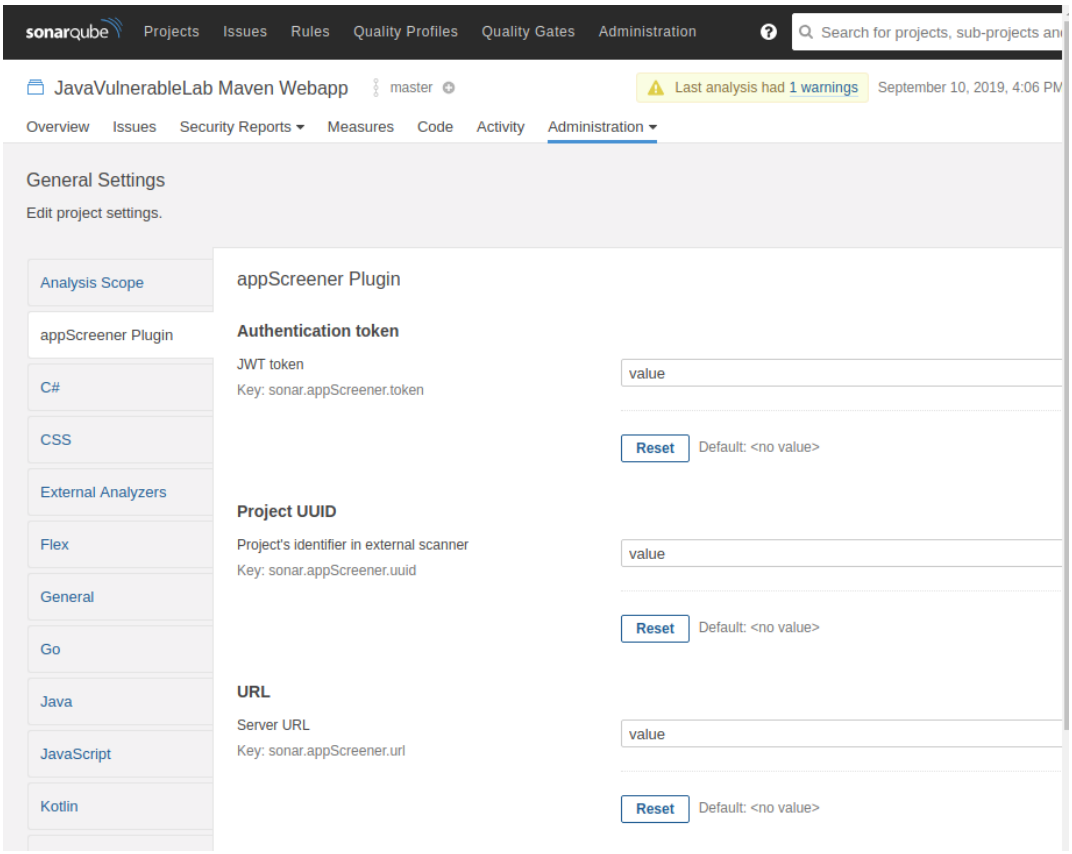


Рис. 6.112: SonarQube: Поля проекта

4. Заполните необходимые поля.
5. Нажмите **Save** для каждого изменённого поля.

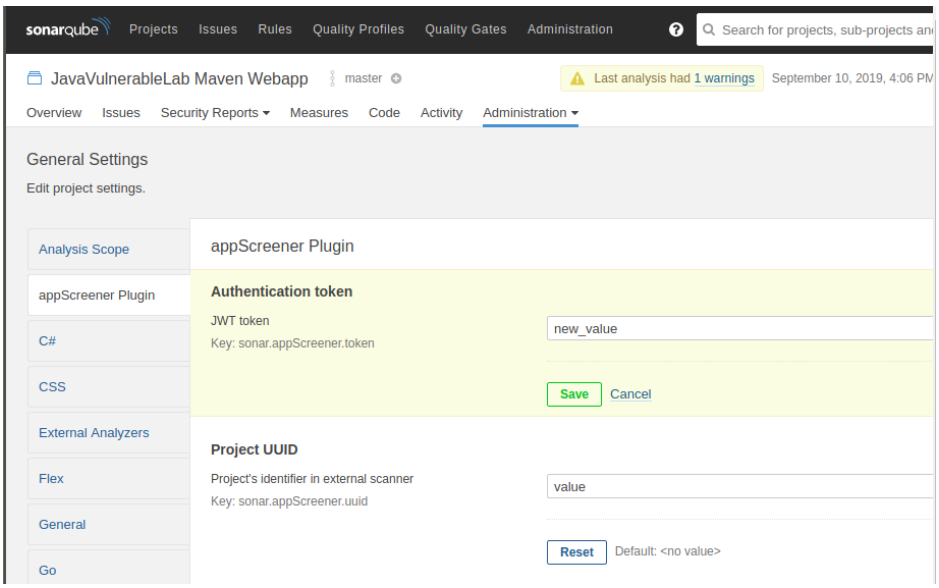


Рис. 6.113: SonarQube: Поле проекта

6.5.9.2.3. Запуск сканирования

Расширение не запускает новое сканирование в appScreener. При запуске сканирования в SonarQube, расширение загружает результаты последнего сканирования appScreener для указанного проекта (поле **Project UUID** в настройках). Расширение сохраняет найденные в appScreener уязвимости в базу данных SonarQube, далее с ними можно работать как с другими объектами SonarQube (понижать/повышать приоритет, назначать ответственного и т.п.).

Часть уязвимостей в приложениях на Java, JavaScript и Python может не отображаться в интерфейсе SonarQube. Чтобы посмотреть полный список уязвимостей, перейдите в интерфейс appScreener.

6.5.10. VCS хостинги

Интеграция с VCS хостингом позволяет автоматически запускать сканирование проекта из хостинга при срабатывании определённых событий или по расписанию. Система поддерживает интеграции с хостингами **GitHub**, **GitLab** и **Bitbucket**. Интеграция происходит с помощью механизма **webhook**.

Чтобы создать интеграцию, нужно настроить её в системе appScreener и в самом хостинге. **Webhook** — механизм оповещения пользователей хостинга о событиях в нём. Хостинги могут уведомлять пользователя о различных событиях в зависимости от хостинга. Solar appScreener поддерживает **Push-события** и **Tag-события**. Информацию о **webhook** и их настройке можно найти в документации соответствующих хостингов: GitHub, GitLab и Bitbucket.

6.5.10.1. Настройка интеграции в appScreener

Чтобы создать интеграцию с VCS хостингом:

1. В боковом меню выбранного проекта выберите вкладку **Автоматическое сканирование** (**Настройки** -> **Автоматическое сканирование**).

2. Выберите тип расположения исходого кода (VCS хостинг) и конкретный хостинг, с которым нужно провести интеграцию.
3. После выбора VCS хостинга появится возможность скопировать ссылку для webhook и токен интеграции. Токен интеграции необходим только для интеграции с **GitHub** и **GitLab**. Скопируйте и используйте ссылку для webhook и токен интеграции при настройке webhook в самом VCS хостинге.
4. Выберите события, по которым будет запускаться сканирование: push в ветку (push-событие), создание метки (tag-событие) и/или сканирование по расписанию.
5. В случае push-события укажите ветки, push в которые будет запускать сканирование. В случае tag-события укажите метки, при создании которых будет запускаться сканирование. Для запуска сканирования в appScreener по tag-событию в **GitLab** и **Bitbucket** достаточно, чтобы webhook срабатывал на push-события. Если запуск сканирования по tag-событию не требуется, нужно отключить переключатель **Создание метки** в настройках интеграции в appScreener. Значения в полях **Ветка** и **Метка** указываются в формате регулярного выражения. По умолчанию система будет обрабатывать все запросы выбранного типа события.
6. В случае сканирования по расписанию активируйте соответствующий переключатель и задайте расписание сканирований в виде cron-выражения.
7. Для сохранения интеграции нажмите кнопку **Сохранить**.

Важно обратить внимание:

- если пользователь, создавший интеграцию, будет удалён из системы appScreener, интеграция будет нерабочей. Чтобы продолжить работоспособность интеграции, нужно создать её повторно;
- машина, на которой установлен Solar appScreener, должна иметь доступ к репозиторию, с которым осуществляется интеграция. В случае приватного репозитория, нужно указать логин и пароль для входа в репозиторий. Это можно сделать в настройках проекта на вкладке **Общие** (боковое меню проекта -> **Настройки** -> **Общие**);
- со списком расширений файлов, которые анализируются при загрузке проекта из репозитория, можно ознакомиться в приложении (табл. 8.1).

6.6. Динамический анализ

В Solar appScreener реализована возможность сканировать веб-ресурсы в режиме динамического анализа. Модуль динамического анализа передаёт на вход случайные или заведомо неверные данные и анализирует реакцию приложения на них. Результаты, полученные в ходе динамического анализа, могут быть использованы для корреляции с данными проекта статического анализа.

6.6.1. Создание проекта

В интерфейсе Solar appScreener реализованы следующие способы создания проекта динамического анализа:

- запуск сканирования приложения по ссылке;
- создание пустого проекта, у которого нет сканирований.

6.6.1.1. Создание пустого проекта

Чтобы создать пустой проект, введите название и нажмите **Создать проект**. При необходимости нажмите **Показать настройки** и установите настройки анализа. Подробнее про настройки анализа в разделе Общие.

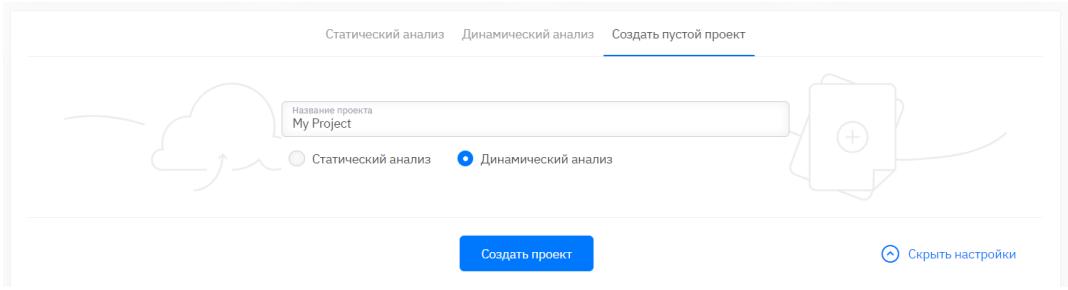


Рис. 6.114: Создание пустого проекта

В созданном проекте можно настроить интеграции. Подробнее про интеграции Solar appScreener в разделе Автоматическое сканирование.

6.6.1.2. Запуск сканирования

Чтобы запустить новое сканирование в UI:

1. Перейдите на **Домашнюю страницу**.
2. Укажите URL ресурса для анализа.
3. Настройте анализ (подробнее о **Настройках** в разделе Настройки).
4. Нажмите **Начать сканирование**.

Во время анализа проект DAST проходит через статусы: **Добавлено в очередь**, **Идет сканирование**, **Завершено**. Процесс сканирования состоит из нескольких этапов: номер текущего этапа будет отображаться в карточке и на странице **Обзор** проекта, а прогресс-бар будет заполняться от 0 до 100% для каждого из этапов.

6.6.2. Управление проектом

Управление проектом состоит из разделов **Обзор**, **Подробные результаты**, **Сканирования**, **Экспорт отчёта**, **Сравнение сканирований** и **Настройки**. Переключение между этими разделами осуществляется через меню в левой части страницы.

Справа от логотипа проекта отображается ID (первые символы UUID проекта). Чтобы скопировать в буфер полный UUID, нажмите на .

На страницу **Обзор** можно перейти, нажав на название проекта на странице **Проекты** в разделе **Динамический анализ** или на **Домашней странице** (если проект входит в шесть последних запущенных проектов).

На страницы **Подробные результаты** или **Экспорт отчёта** можно перейти, нажав на соответствующие кнопки быстрой навигации на странице **Проекты** или на **Домашней странице** (если проект входит в шесть последних запущенных проектов).

6.6.2.1. Обзор

В разделе **Обзор** в правом верхнем углу можно выбрать сканирование, для которого будет отображаться статистика по сканированию. Нажмите на иконку , чтобы отобразились параметры запуска анализа для выбранного сканирования.

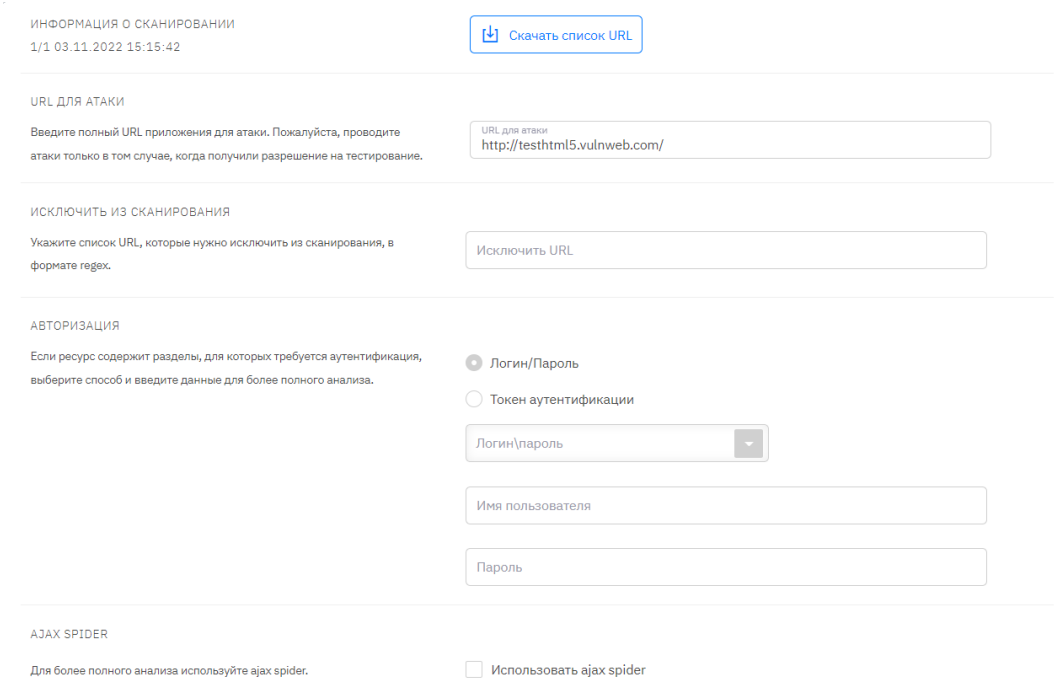


Рис. 6.115: Параметры запуска анализа

На странице **Обзор** представлена следующая информация:

- рейтинг;
- статус сканирования;
- продолжительность сканирования;
- количество уязвимостей каждого уровня критичности
- графическая информация по сканированию и проекту:
  - диаграмма с количеством уязвимостей каждого уровня критичности в сканировании;
  - график уровня безопасности проекта;
  - график количества уязвимостей в проекте;
  - диаграмма с самыми распространенными уязвимостями в сканировании.

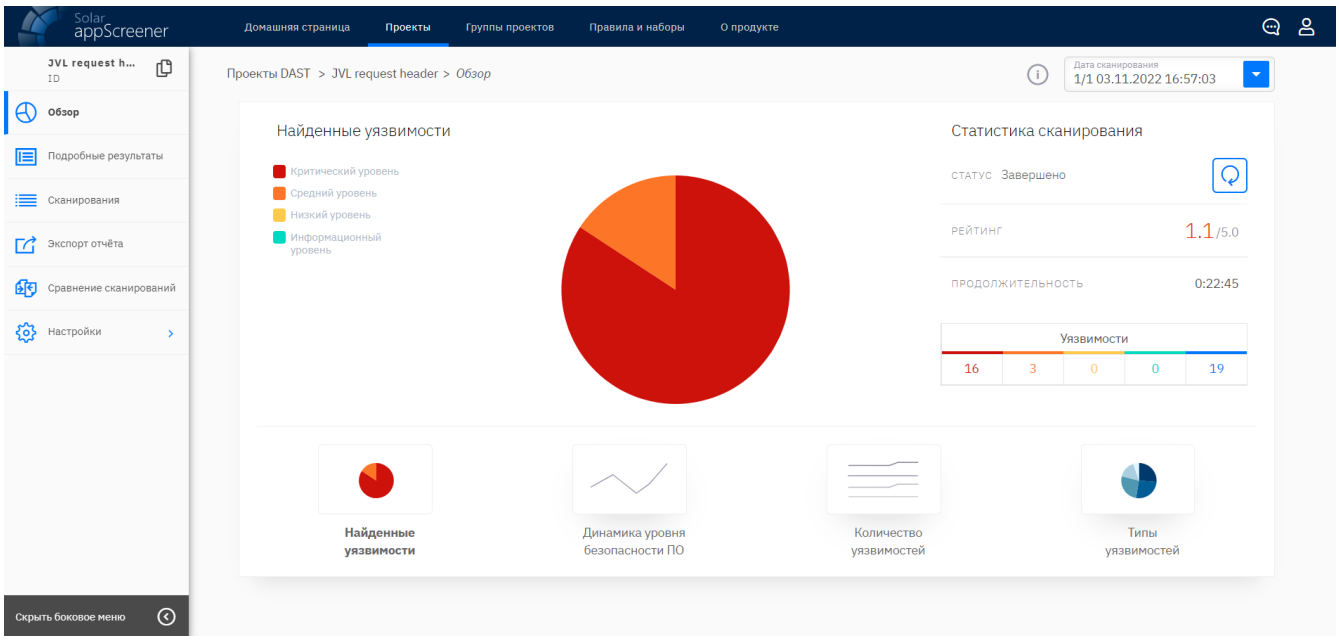


Рис. 6.116: Обзор

Если в данный момент приложение не сканируется, можно запустить новое сканирование, нажав на иконку . Если сканирование находится в процессе анализа, его можно остановить, нажав на иконку .

6.6.2.2. Подробные результаты

На вкладке **Подробные результаты** отображается информация по каждой из обнаруженных уязвимостей для выбранного сканирования. Переключаться между результатами разных сканирований можно с помощью списка сканирований в правом верхнем углу.

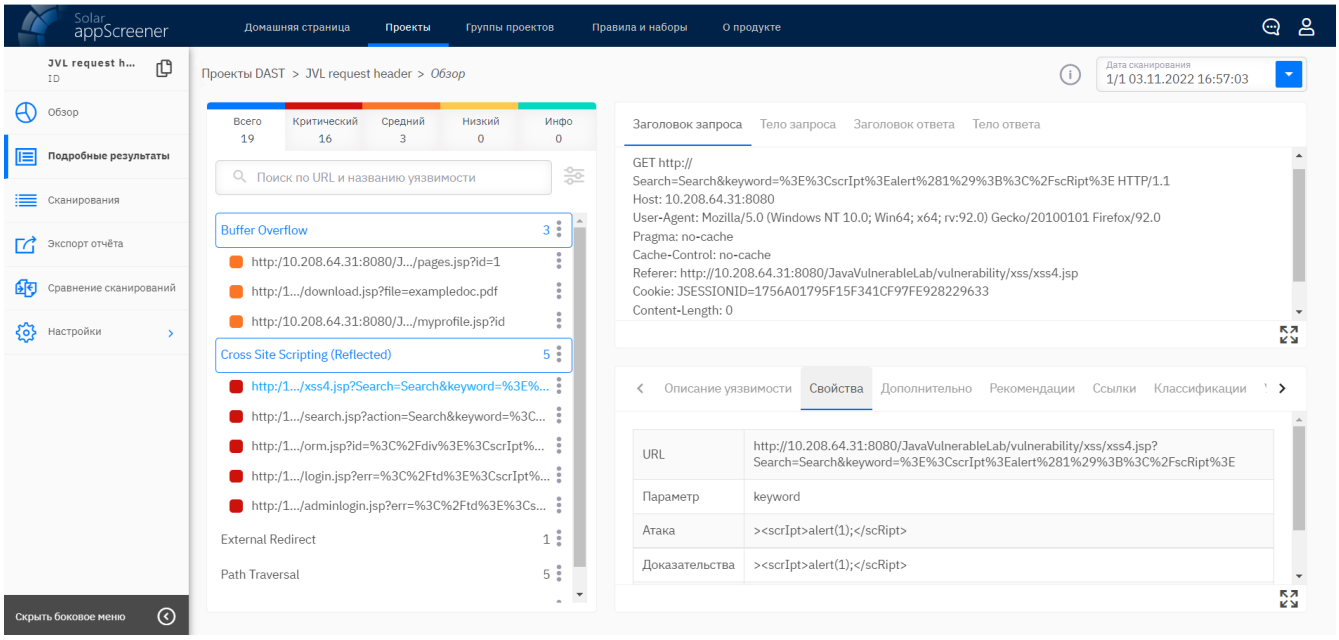


Рис. 6.117: Подробные результаты

В левой части страницы представлен список вхождений уязвимостей, сгруппированный по типам. В верхнем меню можно выбрать, уязвимости какого уровня требуется отобразить. Для удобной навигации по уязвимостям предусмотрен поиск по URL и названию уязвимости, а также фильтры (рис. 6.118).

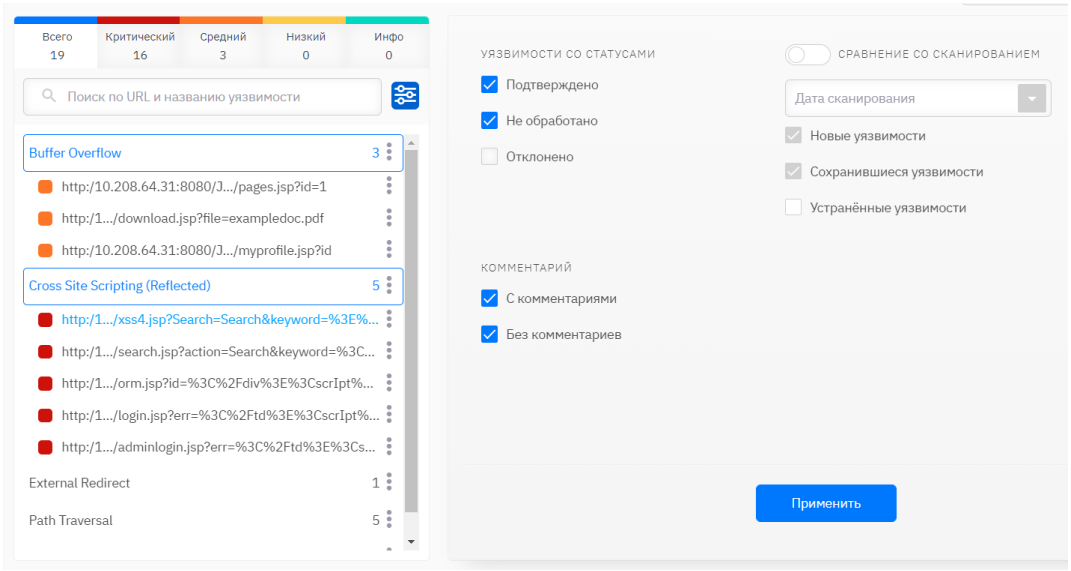


Рис. 6.118: Фильтры результатов

Фильтровать результаты можно по следующим параметрам:

- статусы уязвимостей для отображения:
  - подтверждено;
  - не обработано;
  - отклонено.
- наличие комментария:
  - с комментариями;
  - без комментариев.
- при наличии двух и более успешных сканирований в проекте, можно сравнить текущее сканирование с одним из предшествующих и отобразить уязвимости в соответствии с их статусом. Для этого выберите соответствующие настройки:
  - новые уязвимости — новые уязвимости, по отношению к выбранному из списка сканированию;
  - сохранившиеся уязвимости — уязвимости, обнаруженные в выбранном из списка сканировании и в текущем сканировании;
  - устранённые уязвимости — уязвимости, обнаруженные в выбранном из списка сканировании, но не обнаруженные в текущем сканировании.

Фильтры применяются после нажатия на кнопку **Применить**.

Нажмите на три точки рядом с названием уязвимости или конкретного вхождения, чтобы изменить критичность и статус. При изменении статуса и уровня критичности уязвимости пересчитывается уровень безопасности приложения. Уязвимости со статусом **Отклонено** не учитываются при подсчёте количества уязвимостей и рейтинга безопасности. При пересканировании изменения сохраняются.

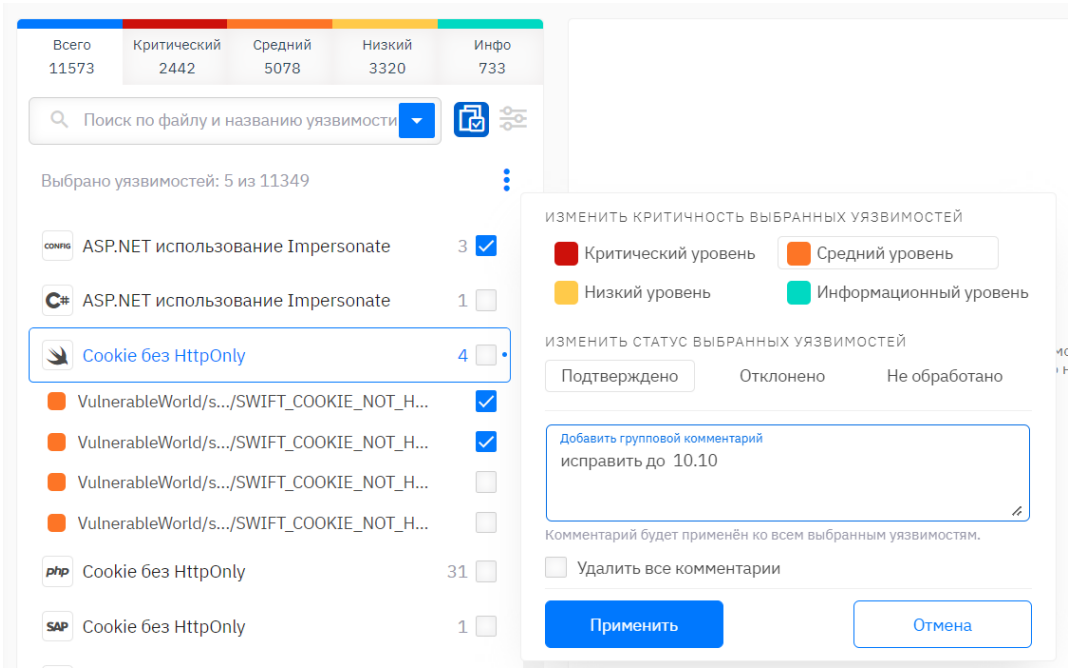


Рис. 6.119: Управление пакетом уязвимостей

После выбора конкретного вхождения уязвимости в центральной части страницы отображается соответствующие ему заголовок и тело запроса и заголовок и тело ответа. В нижней части страницы представлена информация об уязвимости (рис. 6.120): **Описание**, **Свойства**, **Дополнительно**, **Рекомендации**, **Ссылки**, **Классификации**, **Управление уязвимостью** и **Jira**.

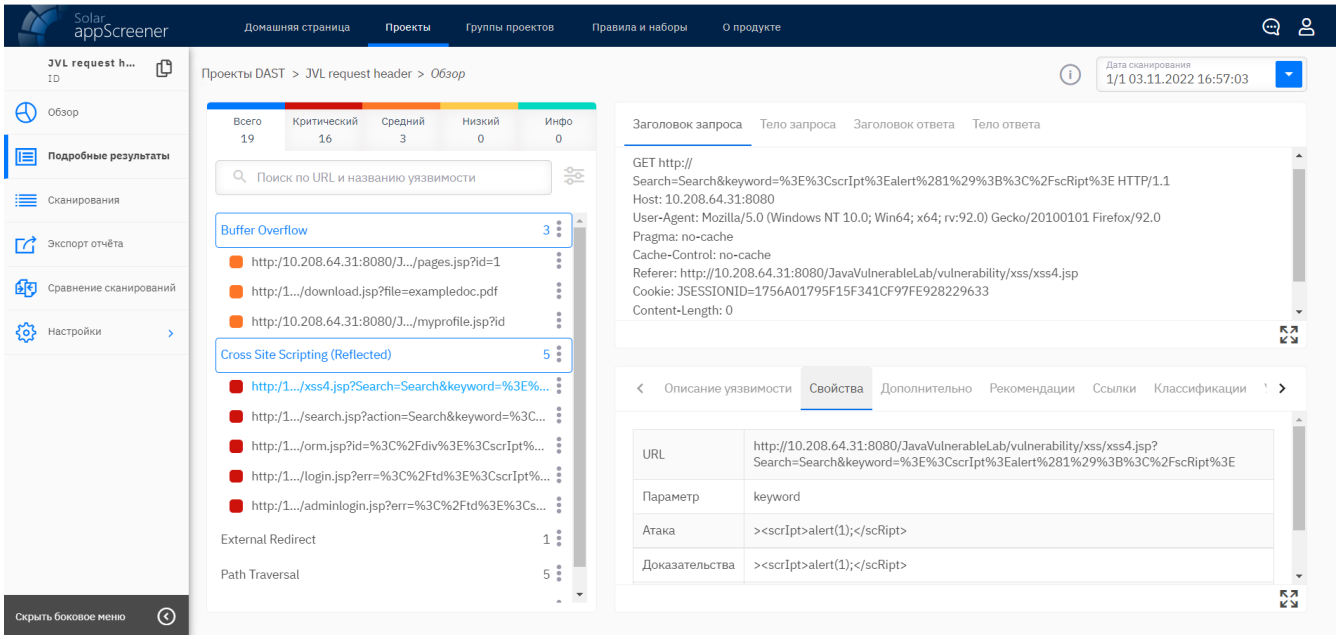


Рис. 6.120: Свойства уязвимости

На вкладке **Классификации** (рис. 6.121) отображаются соответствующие пункты CWE, WASC, OWASP Top 10 2017, OWASP Top 10 2021 и WSTG-v4.2.

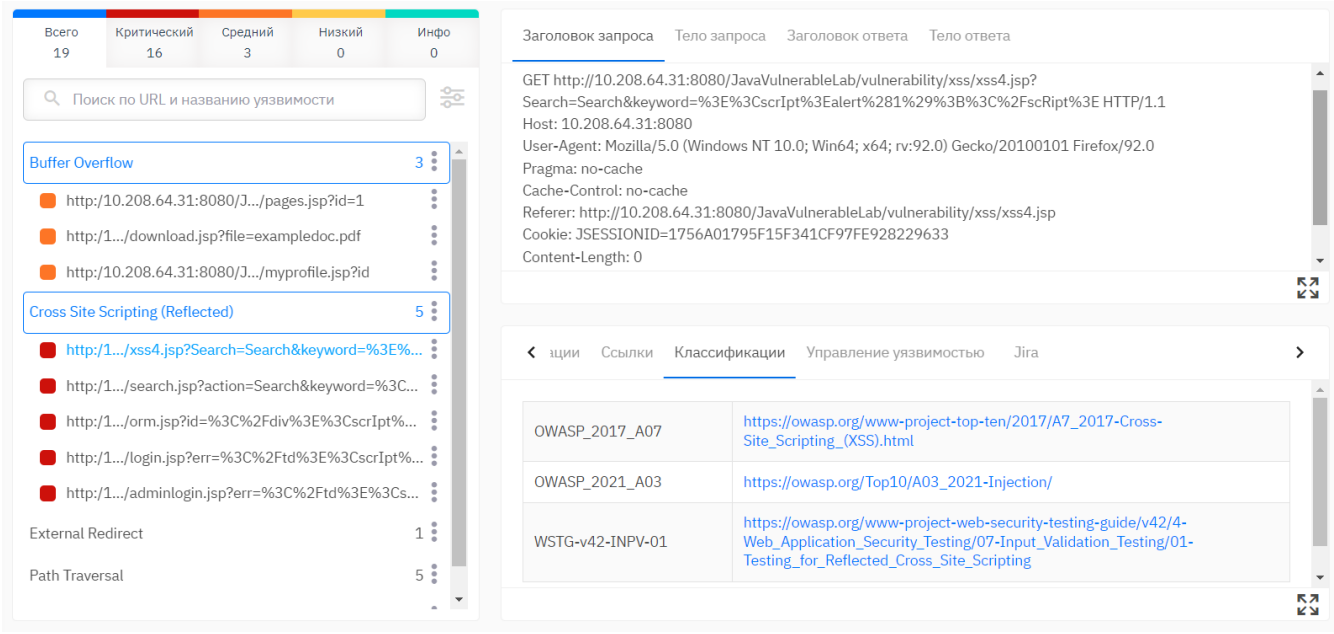


Рис. 6.121: Классификации

На вкладке **Управление уязвимостью** (рис. 6.122) можно изменить уровень критичности и статус, добавить комментарий к вхождению и посмотреть оставленные ранее комментарии.

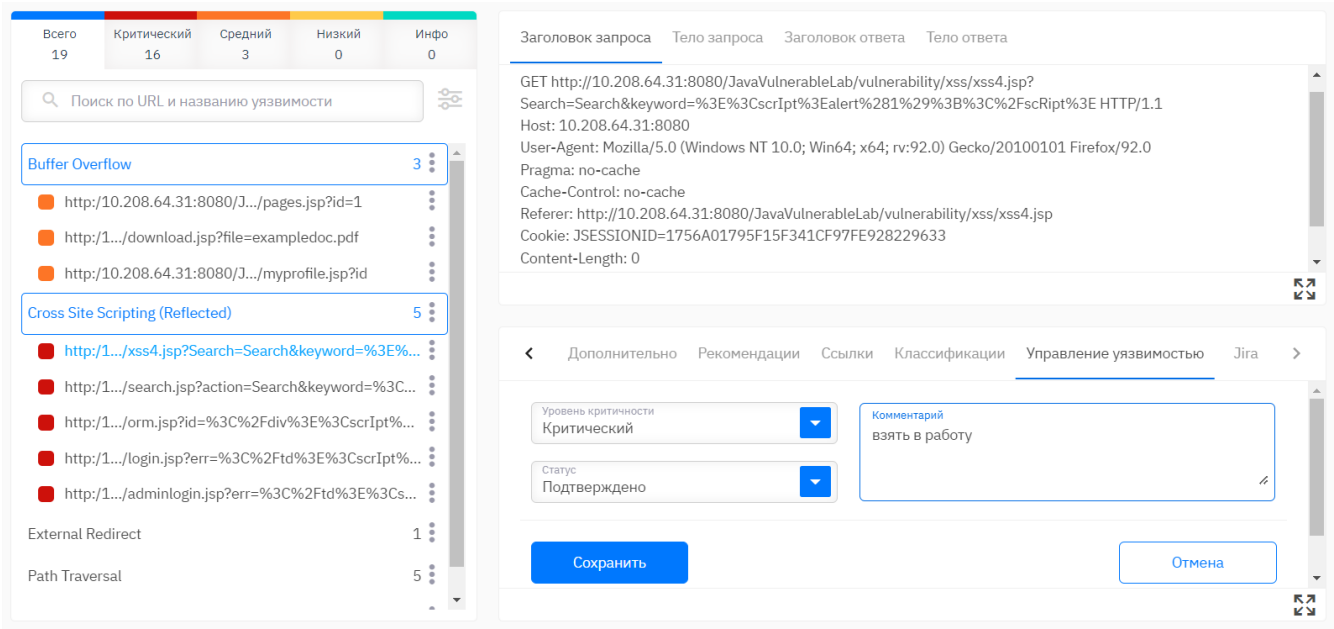


Рис. 6.122: Управление уязвимостью

6.6.2.3. Сканирования

Раздел **Сканирования** предназначен для управления сканированиями в рамках одного проекта. Для каждого сканирования отображаются следующие данные:

- дата и время сканирования, при нажатии на иконку ⓘ отображается информация о параметрах запуска анализа;

- меню действий:
  - выгрузить отчёт;
  - архивировать сканирование;
  - удалить сканирование.
- статус сканирования;
- продолжительность сканирования;
- количество уязвимостей критического, среднего, низкого и информационного уровня;
- рейтинг приложения.

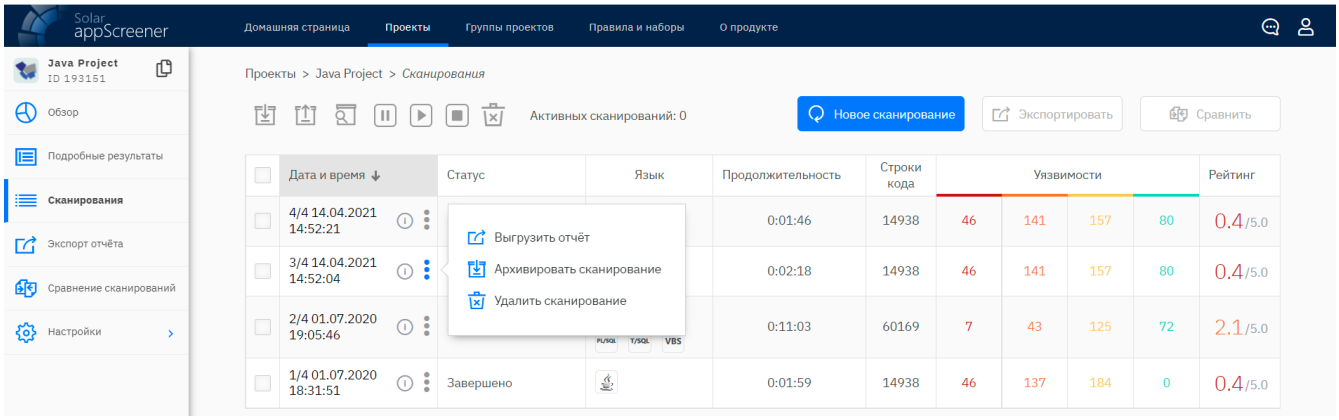


Рис. 6.123: Сканирования

Список можно сортировать по дате сканирования, продолжительности сканирования или рейтингу. Для этого нажмите на соответствующий заголовок, повторное нажатие меняет порядок сортировки.

Сравнить результаты двух выбранных сканирований можно, нажав на кнопку **Сравнить**. Сканирования, которые находятся в архиве, можно скрыть из списка, нажав на **Скрыть архив**, или отображать в списке, нажав на **Показать архив**.

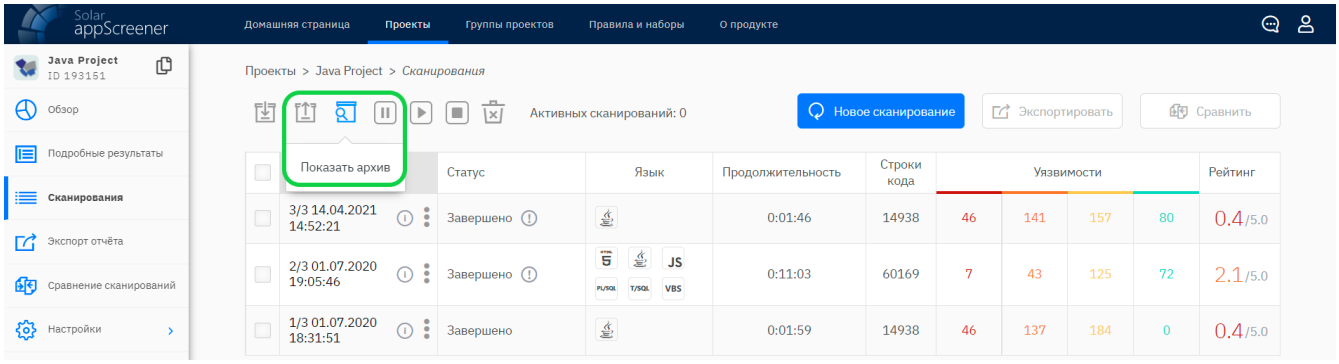


Рис. 6.124: Показать/Скрыть архив

Для проведения повторного сканирования в рамках одного проекта нажмите **Новое сканирование**.

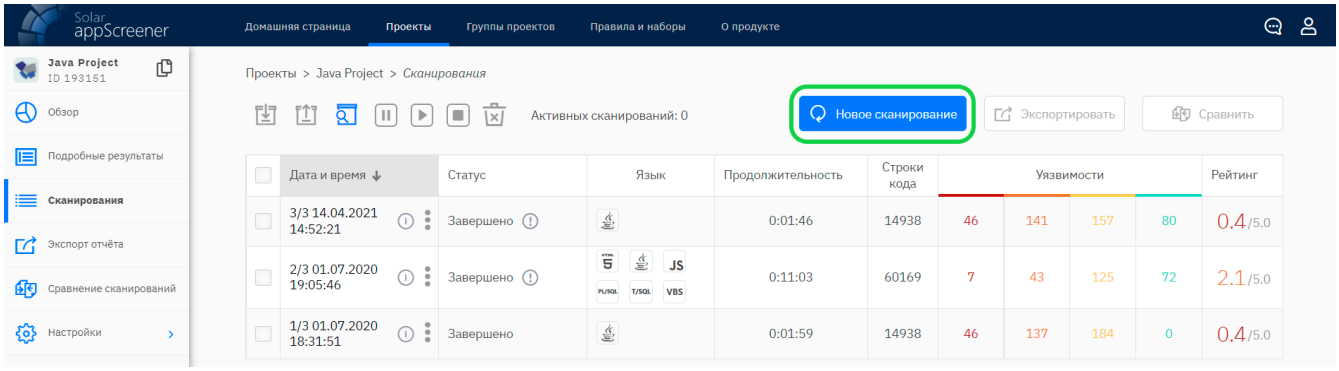


Рис. 6.125: Новое сканирование

В appScreener можно запустить сразу несколько сканирований в одном проекте с разными настройками. Отслеживать статусы сканирований можно в графе **Статус**.

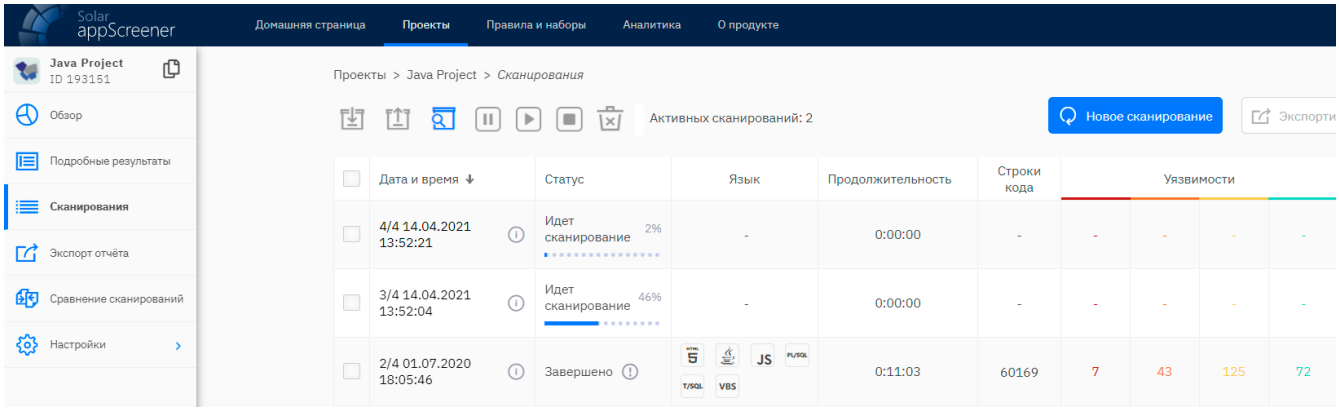


Рис. 6.126: Очередь сканирований

6.6.2.4. Экспорт отчёта

В разделе **Экспорт отчёта** можно выгрузить результаты сканирования в отчёт в формате PDF, CSV или DOCX. Выберите один из готовых шаблонов настроек или задайте информацию для экспорта вручную.

Настройки отчёта включают следующие блоки:

- сканирования;
- сравнить со сканированием;
- информация о проекте;
- фильтр уязвимостей;
- список уязвимостей;
- подробные результаты;
- общие настройки отчёта.

Сканирования

Для экспорта отчёта выберите одно или несколько сканирований. Чтобы получить только сводную информацию по проекту, удалите все сканирования из списка.

Сравнить со сканированием

Выберите одно сканирование, чтобы опция **Сравнить со сканированием** стала доступна. В отчёт будут включены таблица сравнения, график и статистика по новым, сохранившимся и устранённым уязвимостям.

Выберите статусы уязвимостей (новые, сохранившиеся и/или устранённые) и укажите количество вхождений каждой уязвимости.

Информация о проекте

В отчёт можно включить динамику уровня безопасности и историю сканирований.

Информация о сканировании

По умолчанию будет добавлена статистика сканирования: статус, рейтинг, продолжительность, количество уязвимостей.

Выберите дополнительную информацию о сканировании:

- диаграмма найденных уязвимостей;
- диаграмма типов уязвимостей;
- информация об ошибках сканирования;
- настройки запуска сканирования.

Фильтр уязвимостей

Выберите уязвимости уровню критичности и типу.

Список уязвимостей

Выберите статусы уязвимостей и задайте количество их вхождений.

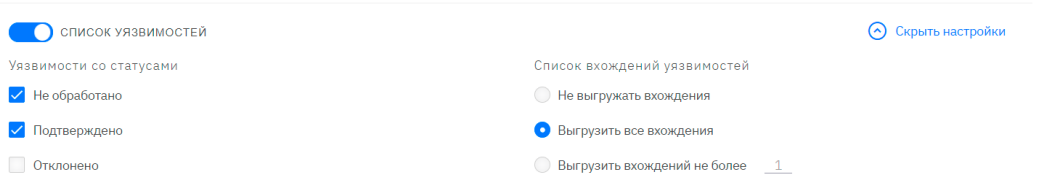


Рис. 6.127: Список уязвимостей

Подробные результаты

По умолчанию для уязвимостей будут добавлены описание, свойства, дополнительно, рекомендации по устранению, ссылки. Также можно настроить:

- статусы уязвимостей: **Не обработано**, **Подтверждено**, **Отклонено** (подробнее в разделе Подробные результаты);
- количество вхождений уязвимостей;
- включение комментариев;
- включение запроса/ответа.

Общие настройки отчёта

Выберите язык, формат отчёта и при необходимости включите в него настройки экспорта и оглавление. Также можно настроить отображение статусов уязвимостей в отчёте и установить пользовательский логотип.

Обратите внимание:

Для корректного отображения данных CSV-отчёта в **Microsoft Excel** необходимо вручную выбрать в выпадающем списке **Обнаружение типов данных** опцию **Не обнаруживать типы данных** во время импорта файла. Настройка отображения статусов уязвимостей недоступна для этого формата.

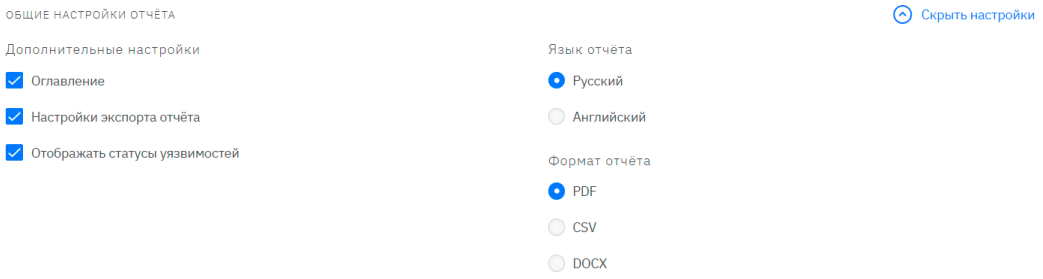


Рис. 6.128: Общие настройки отчёта

Чтобы скачать отчёт, нажмите **Скачать**.

Чтобы отправить отчёт по почте, нажмите **Отправить по e-mail**. В открывшейся форме укажите список адресов получателей и при необходимости отредактируйте текст письма.

6.6.2.5. Сравнение сканирований

В разделе **Сравнение сканирований** можно производить сравнение результатов сканирований. Чтобы сравнить результаты, выберите в верхней части страницы два сканирования. На странице отобразится количество устраненных, новых и сохранившихся уязвимостей на графике и в таблице, а также будет представлена таблица со сравнением по дате сканирования, продолжительности, количеству уязвимостей с учётом уровня критичности и рейтингу.

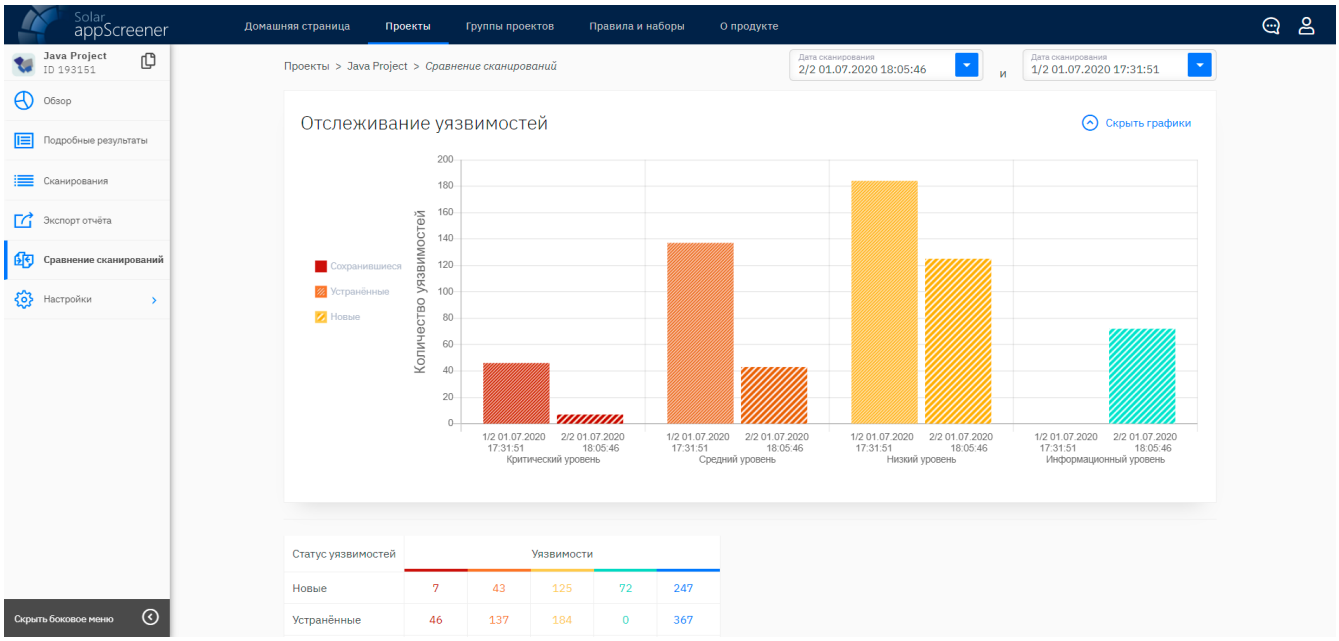


Рис. 6.129: Сравнение сканирований

6.6.2.6. Настройки

В разделе **Настройки** отображаются настройки проекта. В этом разделе можно работать с сущностями: **Общие**, **Права пользователей**, **Jira**, **Автоматическое сканирование** и **Управление проектом**.

6.6.2.6.1. Общие

В подразделе **Общие** (рис. 6.130) можно задать настройки для последующих сканирований:

- указать URL приложения для атаки;
- указать URL, которые нужно исключить из анализа;
- выбрать способ авторизации и заполнить необходимые данные для ресурсов, требующих аутентификации;
- настроить использование ajax-паука (используйте как дополнение для более качественного, глубокого анализа);
- указать URL OpenAPI схемы (используйте, чтобы анализатор отправлял на вход приложению специальные запросы в соответствии с предоставленной схемой).

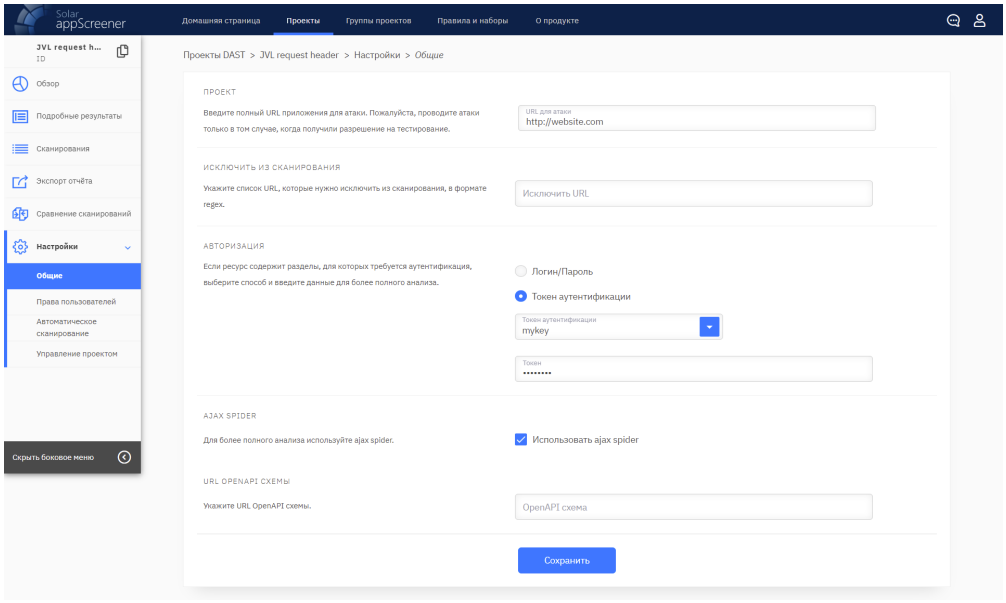


Рис. 6.130: Общие

В подразделе **Права пользователей** можно быстро выдать доступ к проекту другим пользователям системы и настроить их права в проекте.

На вкладке **Jira** можно привязать проект в **Jira** к проекту appScreener (подробнее см. раздел Как привязать проект в Solar appScreener к проекту в Jira).

6.6.2.6.2. Автоматическое сканирование

Для удобной работы с долгосрочными проектами можно настроить автоматическое сканирование. Для настройки введите URL приложения, аутентификационные данные (если необходимо) и задайте расписание.

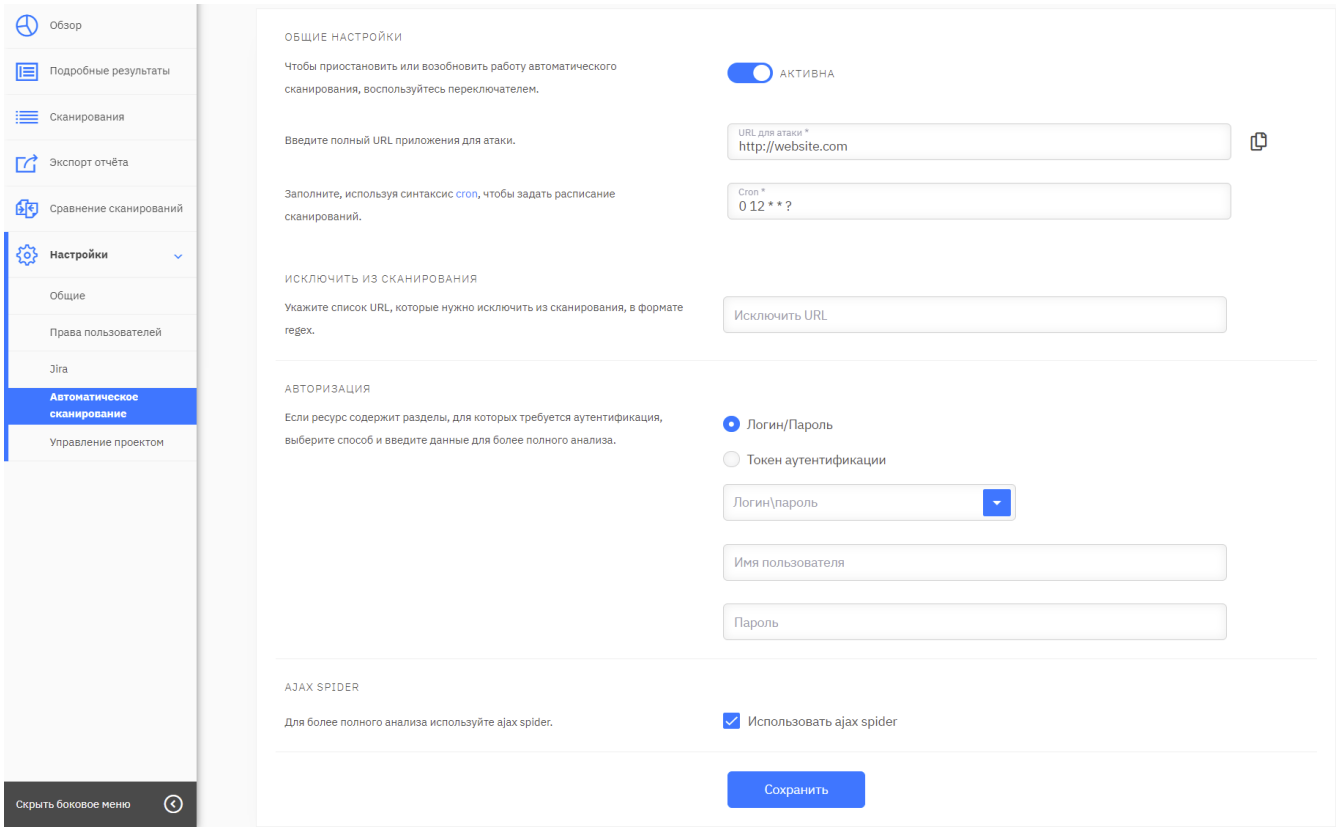


Рис. 6.131: Сканирование по расписанию

6.6.2.6.3. Управление проектом

В подразделе **Управление проектом** можно редактировать данные проекта, а также архивировать или удалить проект. Архивированные проекты продолжают храниться в системе. Чтобы удалить проект без возможности восстановления, нажмите **Удалить проект** и подтвердите действие.

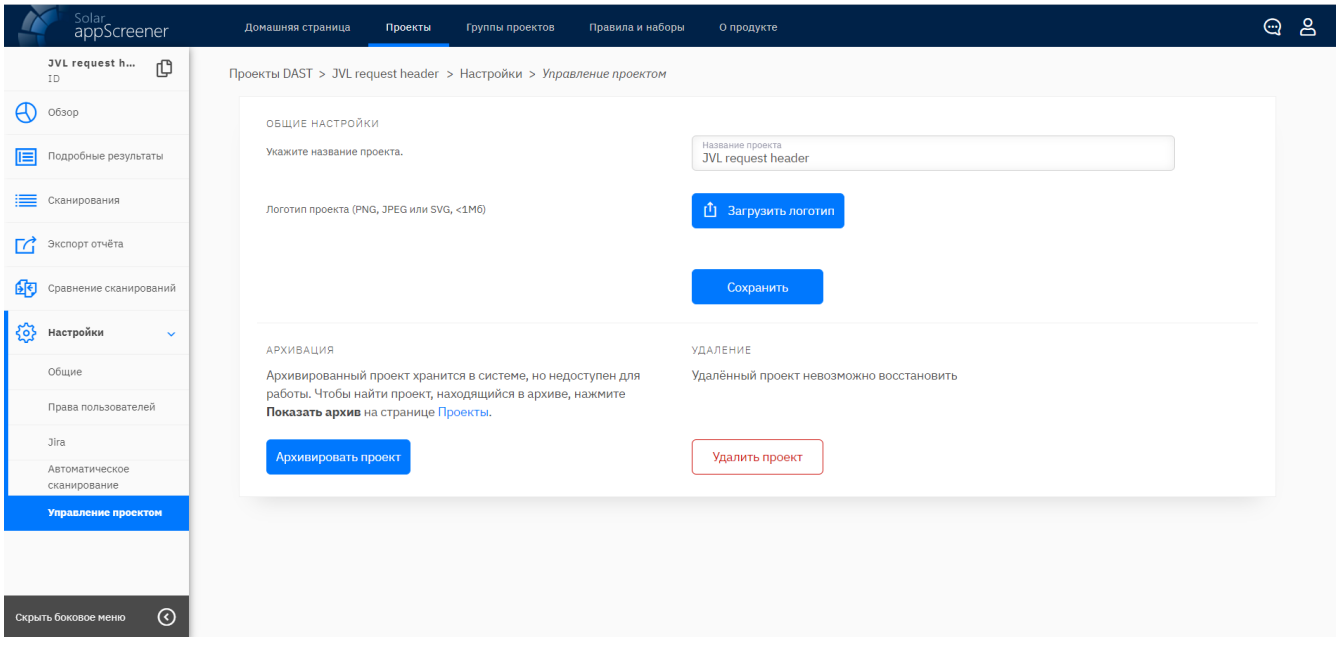


Рис. 6.132: Управление проектом

6.6.3. Корреляция результатов с проектами статического анализа

В случае, когда одно приложение сканируется методами статического и динамического анализа, Solar appScreener предоставляет возможность корреляции результатов исследований.

6.6.3.1. Запуск сканирования

Чтобы настроить корреляцию результатов:

1. Раскройте настройки анализа проекта SAST.
2. В блоке **Корреляция результатов с проектом DAST** выберите проект динамического анализа, соответствующий проекту SAST. Для корреляции результатов будет использовано последнее успешное сканирование проекта динамического анализа.
3. При необходимости укажите дополнительные настройки анализа.
4. Нажмите **Начать сканирование**.

6.6.3.2. Подробные результаты

Ознакомиться со списком уязвимостей и корреляцией результатов статического и динамического анализа можно в разделе **Подробные результаты** проекта.

Уязвимости, подтверждённые методом динамического анализа, будут помечены буквой **D** справа от названия. При активации опции **Показывать тип уязвимости внутри группы** метка отображается справа от названия уязвимости. Если данная опция выключена, метка указывается напротив каждого вхождения уязвимости, тип уязвимости которого был обнаружен методом динамического анализа.

Обратите внимание:

Если в выбранном для корреляции проекте DAST будет запущено новое сканирование, результаты и метки в разделе **Подробные результаты** будут автоматически обновлены по его завершению. Уязвимости со статусом **Отклонено** в корреляции не учитываются.

6.6.3.3. Экспорт отчёта

Чтобы добавить информацию о корреляции в отчёт сканирования, нужно активировать опцию **Отображать метки корреляции результатов** в блоке **Общие настройки отчёта**. Опция доступна для форматов PDF, DOCX и CSV.

6.7. Анализ состава ПО

В Solar appScreener реализована возможность сканировать приложения с целью выявления уязвимых компонент и зависимостей в open-source библиотеках в режиме **Анализ состава ПО**.

6.7.1. Создание проекта

В интерфейсе Solar appScreener реализованы следующие способы создания проекта анализа состава ПО:

- запуск сканирования приложения, загруженного с локального компьютера;
- запуск сканирования приложения по ссылке на репозиторий;
- создание пустого проекта, у которого нет сканирований.

6.7.1.1. Создание пустого проекта

Чтобы создать пустой проект, введите название и нажмите **Создать проект**. При необходимости нажмите **Показать настройки** и установите настройки анализа для будущих сканирований. Подробнее про настройки анализа в разделе Общие.

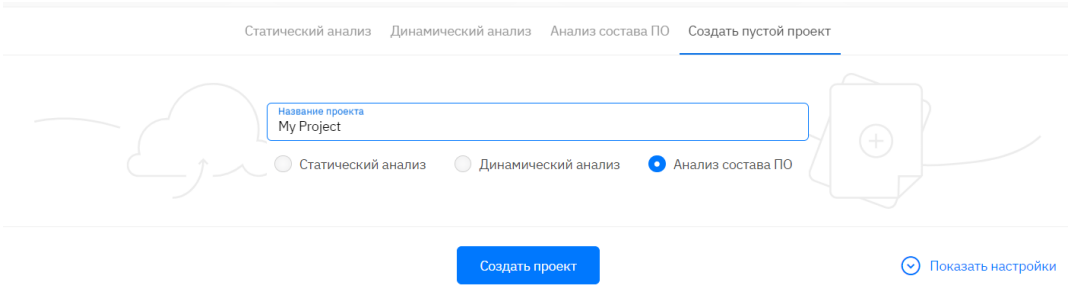


Рис. 6.133: Создание пустого проекта

В созданном проекте можно настроить интеграции. Подробнее про интеграции Solar appScreener в разделе Автоматическое сканирование.

6.7.1.2. Запуск сканирования

Чтобы запустить новое сканирование в UI:

1. Перейдите на **Домашнюю страницу**.
2. Загрузите проект в виде архива с исходным кодом, ссылки на репозиторий с исходным кодом проекта или SBOM файла в формате Cyclone DX (архив со SBOM файлом или ссылка на SBOM файл в репозитории приведет к ошибке сканирования).
3. Настройте анализ (подробнее о **Настройках** в разделе Настройки).
4. Нажмите **Начать сканирование**.

6.7.2. Инструкция по сборке SBOM файла

Проекты Swift/Objective-C (cocoapods)

Для проектов, написанных на Swift, Objective-C, можно воспользоваться инструментом **cyclonedx-cocoapods**. Пример команды, с помощью которой можно создать SBOM файл в формате CycloneDX:

```
cyclonedx-cocoapods --path /path/to/project --output /path/to/bom.xml
```

где `--path` путь до проекта, `--output` путь до файла SBOM.

В результате получится файл с расширением `.xml`. Чтобы конвертировать его в `.json`, можно использовать **cyclonedx-cli**. Пример команды для конвертации CycloneDX-XML в CycloneDX-JSON:

```
cyclonedx convert --input-file /path/to/bom.xml --input-format xml --output-file
/path/to/bom.json --output-format json
```

где:

`--input-file` - путь до конвертируемого файла;

`--input-format` - формат исходного файла;

`--output-file` - путь до итогового файла;

`--output-format` - итоговый формат файла.

Обратите внимание: данный генератор не строит дерево транзитивных зависимостей.

Для генерации SBOM файла для многих языков программирования можно воспользоваться инструментом **cdxgen**. Пример команд, с помощью которых можно создать SBOM файл:

Проекты JavaScript

```
cd /path/to/project
```

```
cdxgen -t node.js -o /path/to/sbom.json
```

где `-t` тип проекта, `-o` путь до файла SBOM.

Проекты Java/Scala/Kotlin (Maven/Gradle)

```
cd /path/to/project
```

```
cdxgen -t java -o /path/to/sbom.json
```

Проекты C/C++ (conan)

```
cd /path/to/project
```

```
cdxgen -t c/c++ -o /path/to/sbom.json
```

Обратите внимание: дерево транзитивных зависимостей будет простроено только если зависимости описаны в `conan.lock`.

Проекты PHP (Composer)

```
cd /path/to/project
```

```
cdxgen -t php -o /path/to/sbom.json
```

Проекты Swift (SwiftPM)

```
cd /path/to/project
```

```
cdxgen -t swift -o /path/to/sbom.json
```

Проекты C# (.Net)

```
cd /path/to/project
```

```
cdxgen -t .Net -o /path/to/sbom.json
```

Обратите внимание: дерево транзитивных зависимостей будет сгенерировано только в присутствии файлов `project.assets.json`, `packages.lock.json`.

Проекты на других языках

Список генераторов для разных языков программирования представлен по ссылке. Все генераторы поддерживают формат SBOM CycloneDX.

6.7.3. Управление проектом

Управление проектом состоит из разделов **Обзор**, **Подробные результаты**, **Сканирования**, **Экспорт отчёта**, **Сравнение сканирований** и **Настройки**. Переключение между этими разделами осуществляется через меню в левой части страницы.

Справа от логотипа проекта отображается ID (первые символы UUID проекта). Чтобы скопировать в буфер полный UUID, нажмите на  .

На страницу **Обзор** можно перейти, нажав на название проекта на странице **Проекты** в разделе **SCA** или на **Домашней странице** (если проект входит в шесть последних запущенных проектов).

На страницы **Подробные результаты** или **Экспорт отчёта** можно перейти, нажав на соответствующие кнопки быстрой навигации на странице **Проекты** или на **Домашней странице** (если проект входит в шесть последних запущенных проектов).

6.7.3.1. Обзор

В разделе **Обзор** в правом верхнем углу можно выбрать сканирование, для которого будет отображаться статистика по сканированию. Нажмите на иконку  , чтобы отобразились параметры запуска анализа для выбранного сканирования.

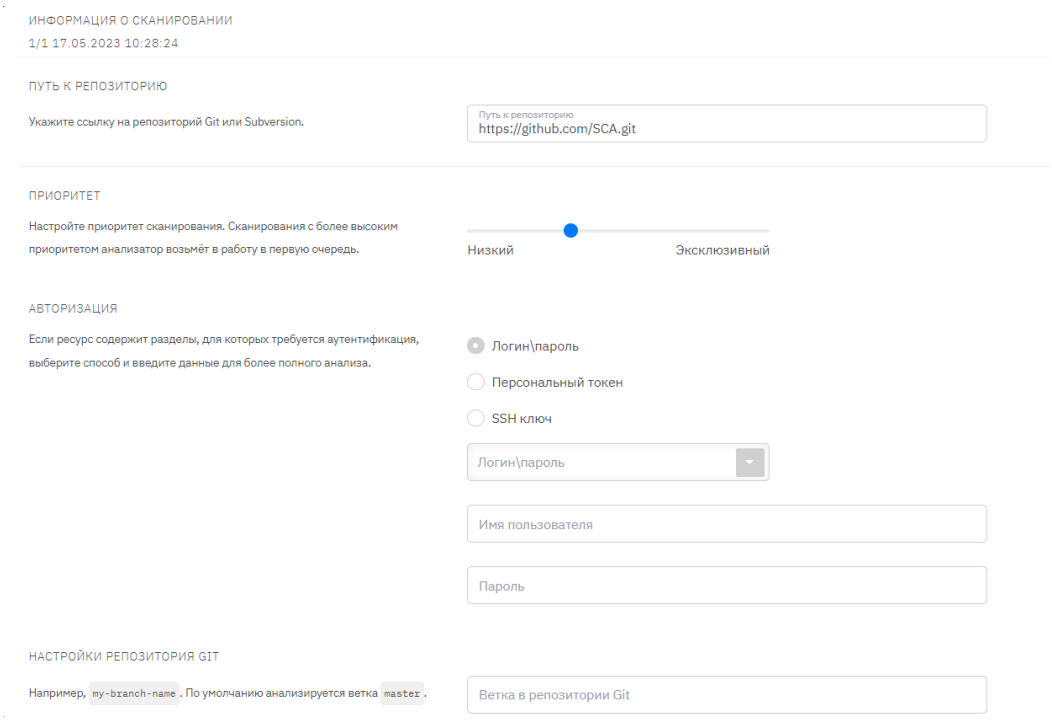


Рис. 6.134: Параметры запуска анализа

На странице **Обзор** представлена следующая информация:

- рейтинг;
- статус сканирования;
- продолжительность сканирования;
- общее количество компонент;
- количество уязвимых компонент;
- графическая информация по сканированию и проекту:
  - диаграмма с количеством уязвимостей каждого уровня критичности в сканировании;
  - график уровня безопасности проекта;
  - график количества уязвимостей в проекте;
  - диаграмма с наиболее уязвимыми компонентами.

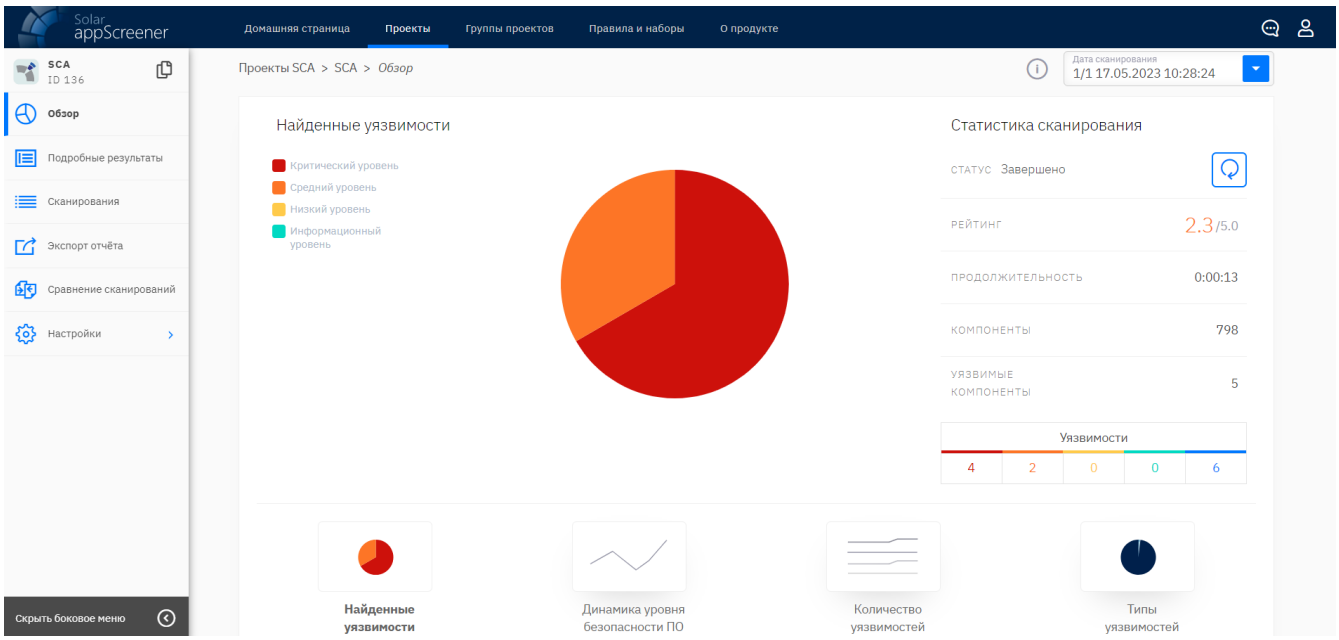


Рис. 6.135: Обзор

Если в данный момент приложение не сканируется, можно запустить новое сканирование, нажав на иконку . Если сканирование находится в процессе анализа, его можно остановить, нажав на иконку .

6.7.3.2. Подробные результаты

На вкладке **Подробные результаты** отображается информация по каждой из обнаруженных уязвимостей для выбранного сканирования. Переключаться между результатами разных сканирований можно с помощью списка сканирований в правом верхнем углу.

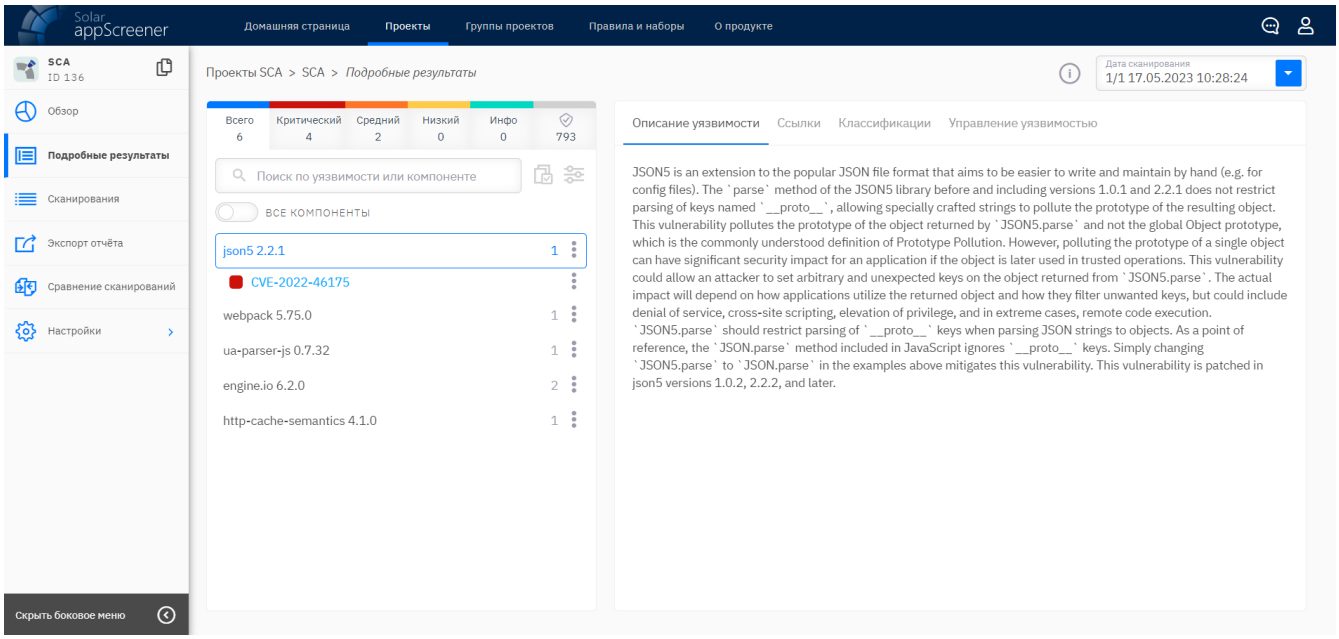


Рис. 6.136: Подробные результаты

В левой части страницы представлен список вхождений уязвимостей, сгруппированный по названию библиотек и версий. Если компонента содержит зависимости, они будут отмечены соответствующими тэгами: **D** для прямых, **T** для транзитивных зависимостей. Связанные зависимости отображаются по наведению курсора на тэг.

В верхнем меню можно выбрать, уязвимости какого уровня требуется отобразить. Для удобной навигации по уязвимостям предусмотрен поиск по названию уязвимости или компоненте, а также фильтры (рис. 6.137).

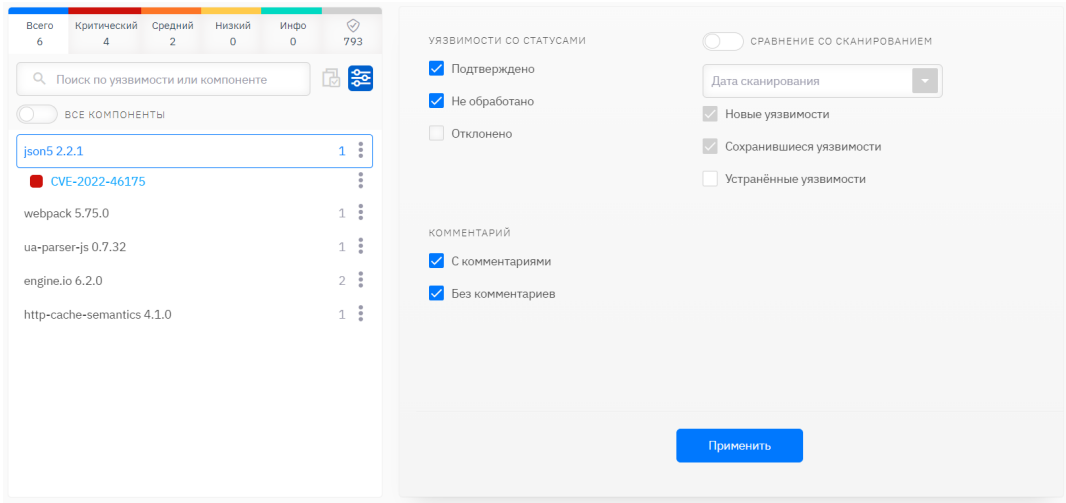


Рис. 6.137: Фильтры результатов

Фильтровать результаты можно по следующим параметрам:

- статусы уязвимостей для отображения:
  - подтверждено;
  - не обработано;
  - отклонено.
- наличие комментария:
  - с комментариями;
  - без комментариев.
- при наличии двух и более успешных сканирований в проекте, можно сравнить текущее сканирование с одним из предшествующих и отобразить уязвимости в соответствии с их статусом. Для этого выберите соответствующие настройки:
  - новые уязвимости — новые уязвимости, по отношению к выбранному из списка сканированию;
  - сохранившиеся уязвимости — уязвимости, обнаруженные в выбранном из списка сканировании и в текущем сканировании;
  - устранённые уязвимости — уязвимости, обнаруженные в выбранном из списка сканировании, но не обнаруженные в текущем сканировании.

Фильтры применяются после нажатия на кнопку **Применить**.

Нажмите на три точки рядом с названием уязвимости, чтобы изменить критичность и статус. При изменении статуса и уровня критичности уязвимости пересчитывается уровень безопасности приложения. Уязвимости со статусом **Отклонено** не учитываются при подсчёте количества уязвимостей и рейтинга безопасности. При пересканировании изменения сохраняются.

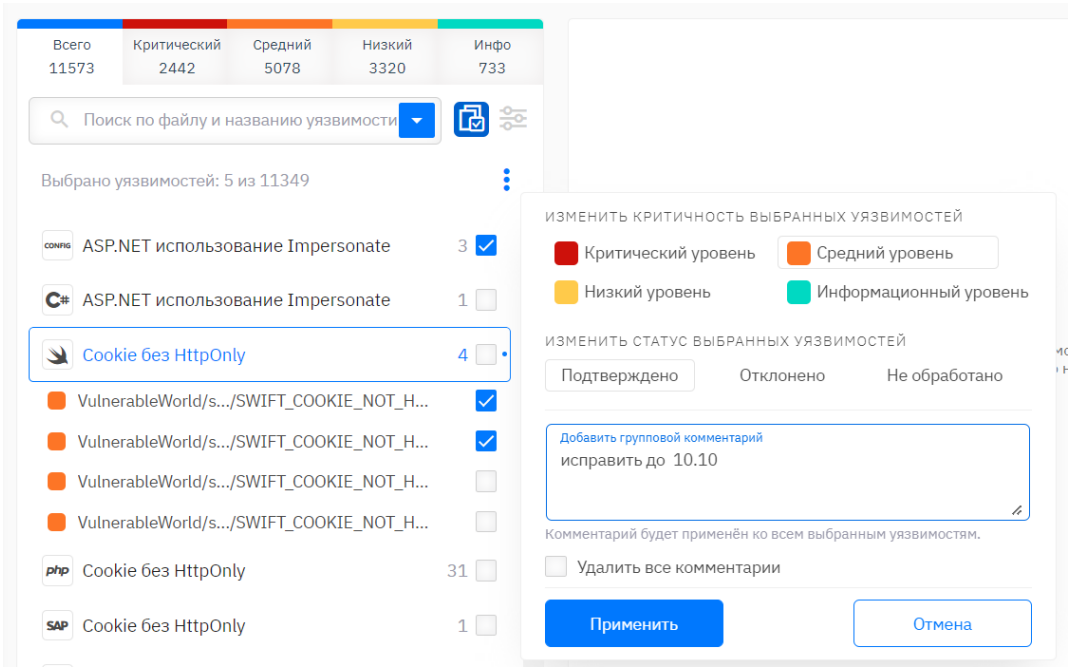


Рис. 6.138: Управление пакетом уязвимостей

После выбора конкретной уязвимости в центральной части страницы отображается следующая информация (рис. 6.139): **Описание уязвимости**, **Ссылки**, **Классификации**, **Управление уязвимостью**.

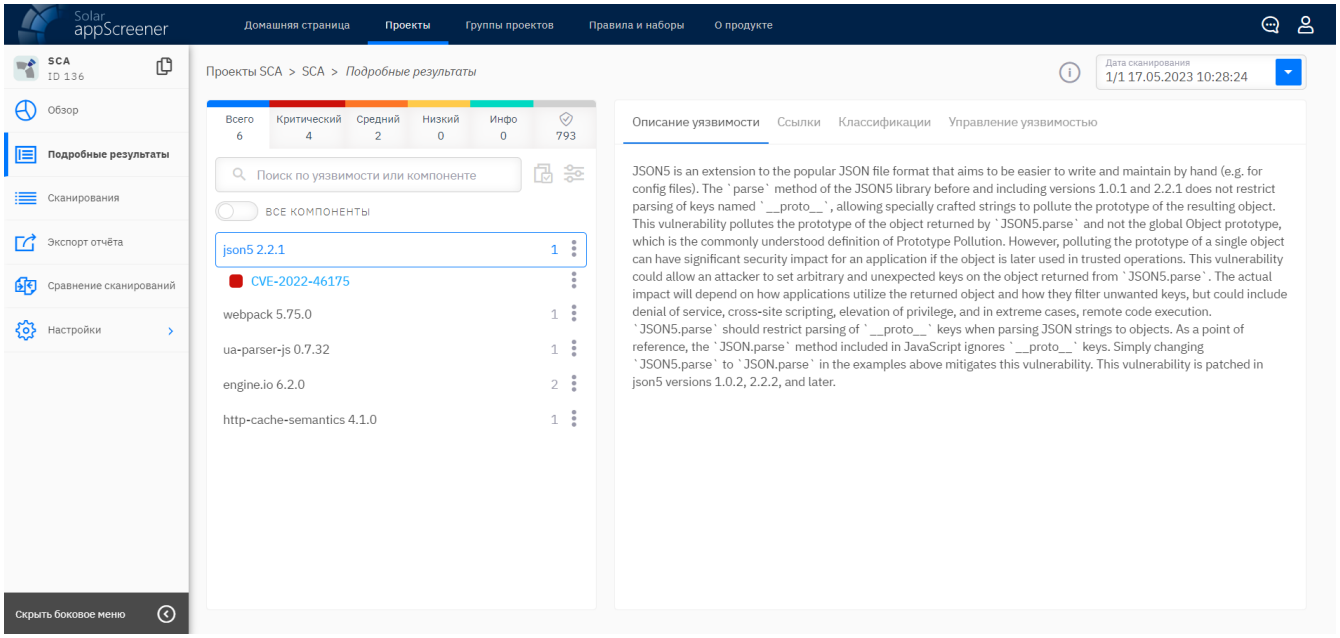


Рис. 6.139: Свойства уязвимости

На вкладке **Управление уязвимостью** (рис. 6.140) можно изменить уровень критичности и статус, добавить комментарий к уязвимости и посмотреть оставленные ранее комментарии.

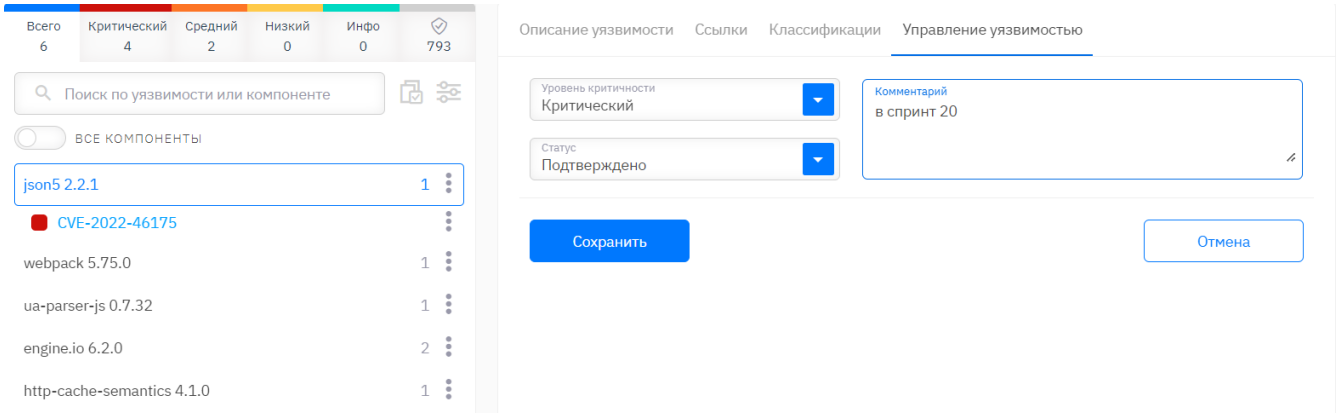


Рис. 6.140: Управление уязвимостью

6.7.3.3. Сканирования

Раздел **Сканирования** предназначен для управления сканированиями в рамках одного проекта. Для каждого сканирования отображаются следующие данные:

- дата и время сканирования, при нажатии на иконку ⓘ отображается информация о параметрах запуска анализа;
- меню действий:
  - выгрузить отчёт;
  - архивировать сканирование;
  - удалить сканирование.
- статус сканирования;
- продолжительность сканирования;
- общее количество компонент;
- количество уязвимых компонент;
- количество уязвимостей критического, среднего, низкого и информационного уровня;
- рейтинг приложения.

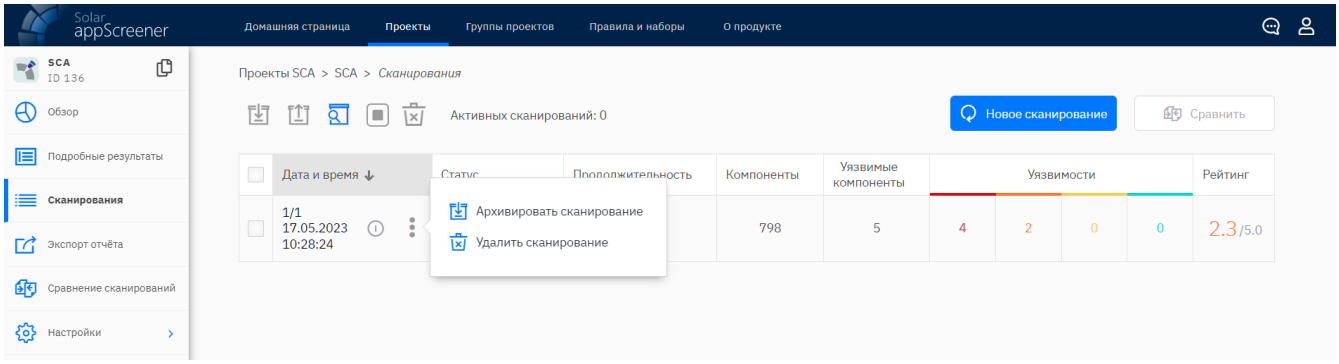


Рис. 6.141: Сканирования

Список можно сортировать по дате сканирования, продолжительности сканирования или рейтингу, общему количеству компонент или уязвимым компонентам. Для этого нажмите

на соответствующий заголовок, повторное нажатие меняет порядок сортировки.

Сравнить результаты двух выбранных сканирований можно, нажав на кнопку **Сравнить**. Сканирования, которые находятся в архиве, можно скрыть из списка, нажав на **Скрыть архив**, или отображать в списке, нажав на **Показать архив**.

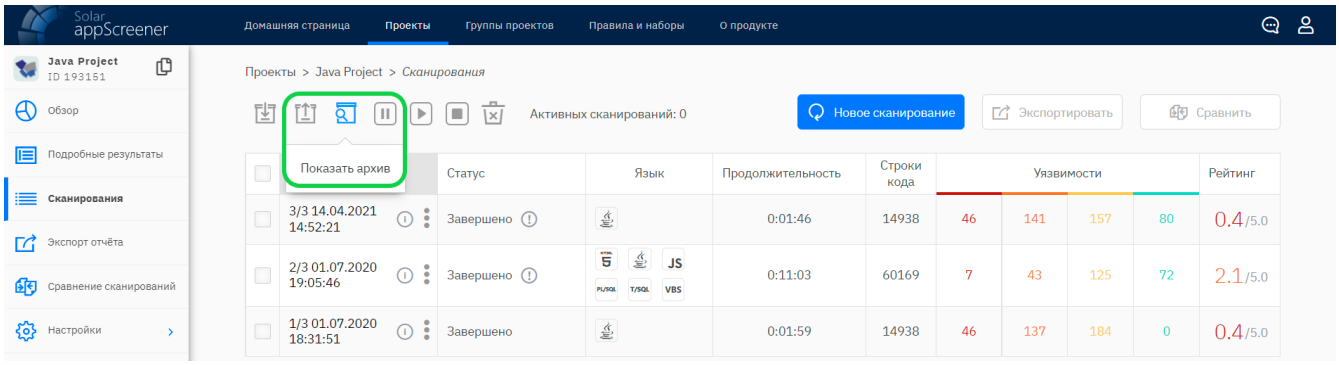


Рис. 6.142: Показать/Скрыть архив

Для проведения повторного сканирования в рамках одного проекта нажмите **Новое сканирование**.

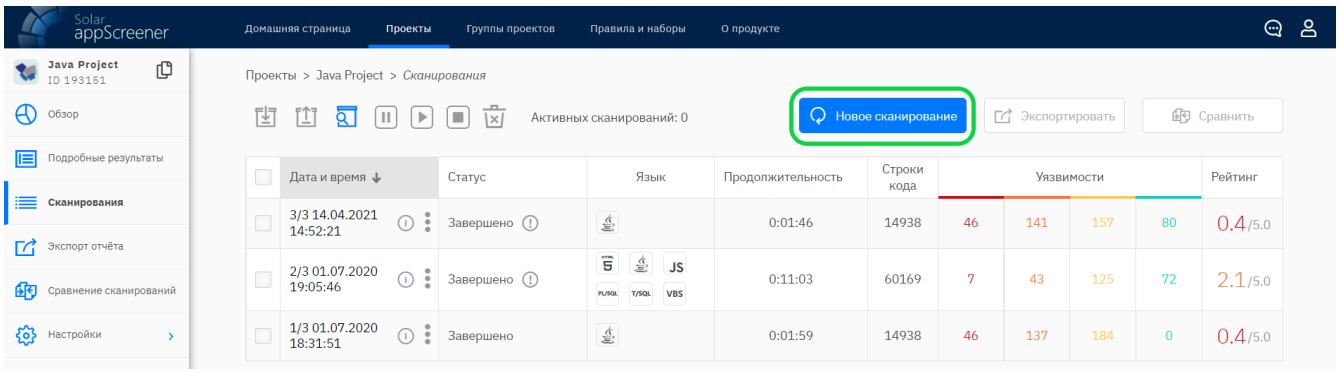


Рис. 6.143: Новое сканирование

В appScreener можно запустить сразу несколько сканирований в одном проекте с разными настройками. Отслеживать статусы сканирований можно в графе **Статус**.

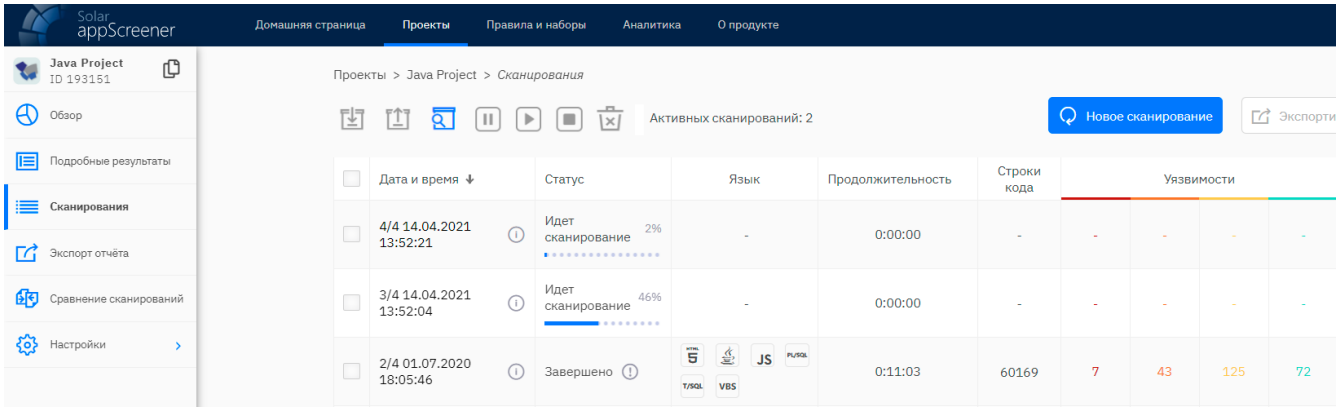


Рис. 6.144: Очередь сканирований

6.7.3.4. Экспорт отчёта

В разделе **Экспорт отчёта** можно выгрузить результаты сканирования в отчёт в формате PDF, CSV или DOCX. Выберите один из готовых шаблонов настроек или задайте информацию для экспорта вручную.

Настройки отчёта включают следующие блоки:

- сканирования;
- сравнить со сканированием;
- информация о проекте;
- информация о сканировании;
- фильтр уязвимостей;
- список уязвимостей;
- подробные результаты;
- общие настройки отчёта.

Сканирования

Для экспорта отчёта выберите одно или несколько сканирований. Чтобы получить только сводную информацию по проекту, удалите все сканирования из списка.

Сравнить со сканированием

Выберите одно сканирование, чтобы опция **Сравнить со сканированием** стала доступна. В отчёт будут включены таблица сравнения, график и статистика по новым, сохранившимся и устранённым уязвимостям.

Выберите статусы уязвимостей (новые, сохранившиеся и/или устранённые) и укажите количество вхождений каждой уязвимости.

Информация о проекте

В отчёт можно включить динамику уровня безопасности и историю сканирований.

Информация о сканировании

По умолчанию будет добавлена статистика сканирования: статус, рейтинг, продолжительность, количество компонент и уязвимостей.

Выберите дополнительную информацию о сканировании:

- диаграмма найденных уязвимостей;
- диаграмма уязвимых компонент;
- настройки запуска сканирования.

Фильтр уязвимостей

Выберите уязвимости по уровню критичности и типу, а также компоненты для отображения.

Список уязвимостей

Выберите статусы уязвимостей и задайте количество их вхождений.

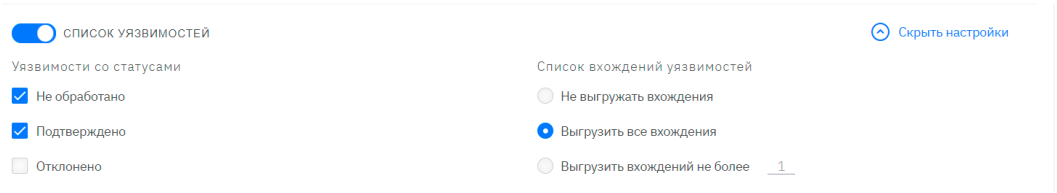


Рис. 6.145: Список уязвимостей

Подробные результаты

По умолчанию для уязвимостей будут добавлены описание, рекомендации по устранению, ссылки. Также можно настроить:

- статусы уязвимостей: **Не обработано**, **Подтверждено**, **Отклонено** (подробнее в разделе Подробные результаты);
- количество уязвимостей компоненты;
- отображение комментариев.

Общие настройки отчёта

Выберите язык, формат отчёта и при необходимости включите в него настройки экспорта и оглавление. Также можно настроить отображение статусов уязвимостей в отчёте и установить пользовательский логотип.

Обратите внимание:

Для корректного отображения данных CSV-отчёта в **Microsoft Excel** необходимо вручную выбрать в выпадающем списке **Обнаружение типов данных** опцию **Не обнаруживать типы данных** во время импорта файла. Настройка отображения статусов уязвимостей недоступна для этого формата.

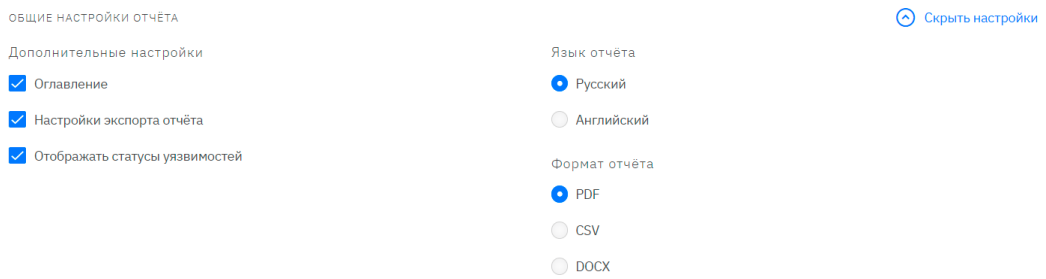


Рис. 6.146: Общие настройки отчёта

Чтобы скачать отчёт, нажмите **Скачать**.

Чтобы отправить отчёт по почте, нажмите **Отправить по e-mail**. В открывшейся форме укажите список адресов получателей и при необходимости отредактируйте текст письма.

6.7.3.5. Сравнение сканирований

В разделе **Сравнение сканирований** можно производить сравнение результатов сканирований. Чтобы сравнить результаты, выберите два сканирования в верхней части страницы. На странице отобразится количество устраненных, новых и сохранившихся уязвимостей на графике и в таблице. Также будет представлена таблица со сравнением

по дате сканирования, продолжительности, общему количеству и количеству уязвимых компонент, количеству уязвимостей с учётом уровня критичности и рейтингу.

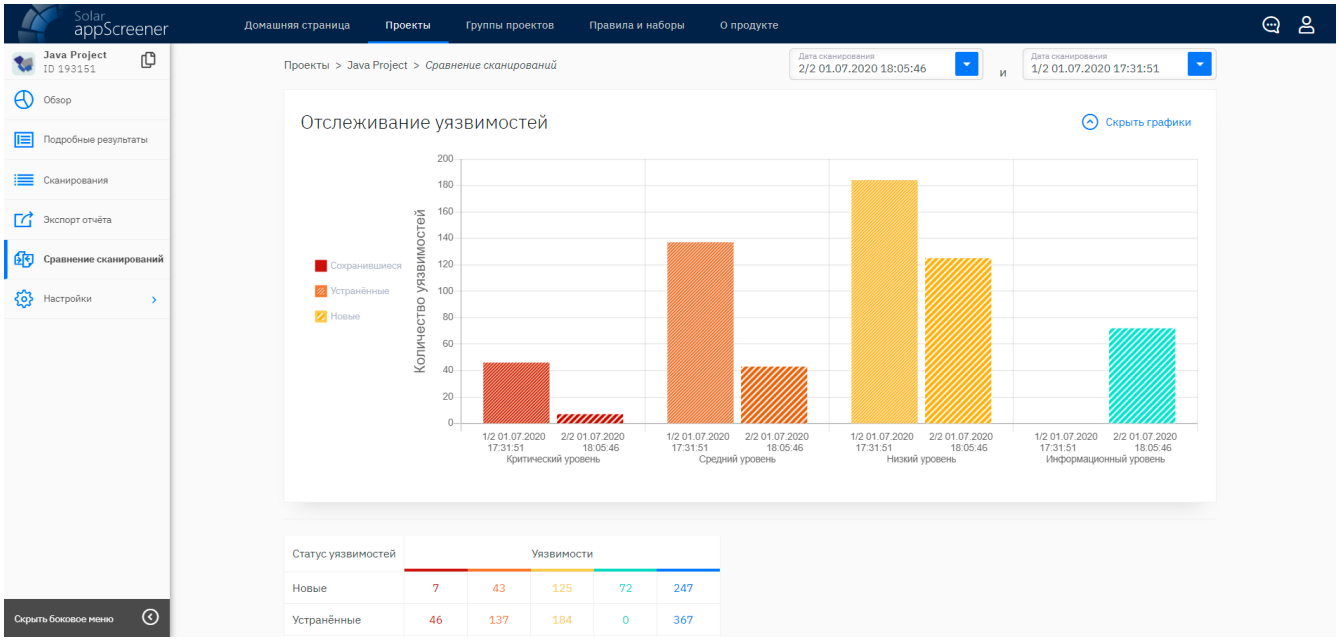


Рис. 6.147: Сравнение сканирований

6.7.3.6. Настройки

В разделе **Настройки** отображаются настройки проекта. В этом разделе можно работать с сущностями: **Общие**, **Права пользователей**, **Автоматическое сканирование** и **Управление проектом**.

6.7.3.6.1. Общие

В подразделе **Общие** (рис. 6.148) можно задать настройки для последующих сканирований:

- указать ссылку на репозиторий Git или Subversion;
- задать приоритет сканирования;
- выбрать способ авторизации и заполнить необходимые данные для ресурсов, требующих аутентификации;
- указать ветку для сканирования в Git-репозитории.

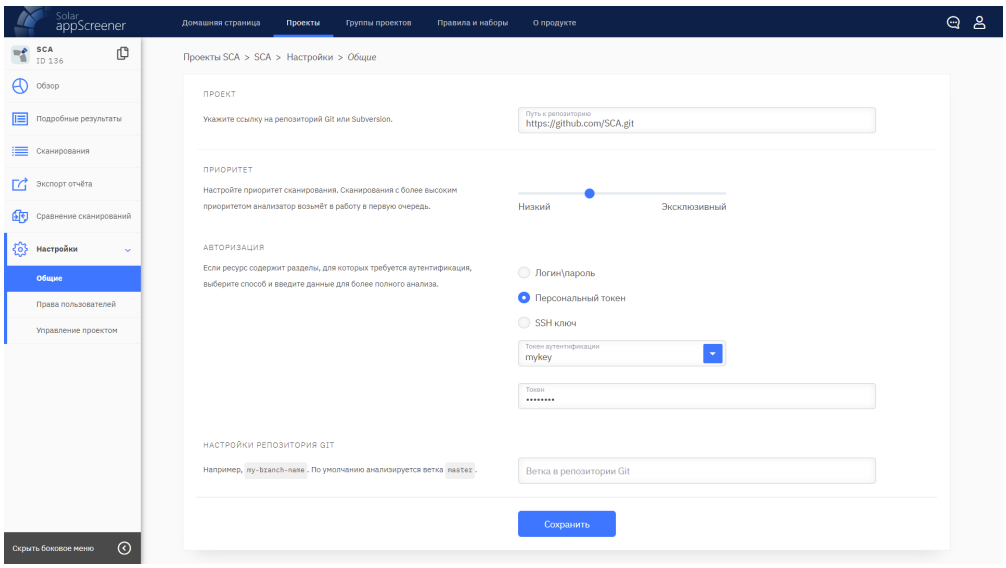


Рис. 6.148: Общие

В подразделе **Права пользователей** можно быстро выдать доступ к проекту другим пользователям системы и настроить их права в проекте.

6.7.3.6.2. Управление проектом

В подразделе **Управление проектом** можно редактировать данные проекта, а также архивировать или удалить проект. Архивированные проекты продолжают храниться в системе. Чтобы удалить проект без возможности восстановления, нажмите **Удалить проект** и подтвердите действие.

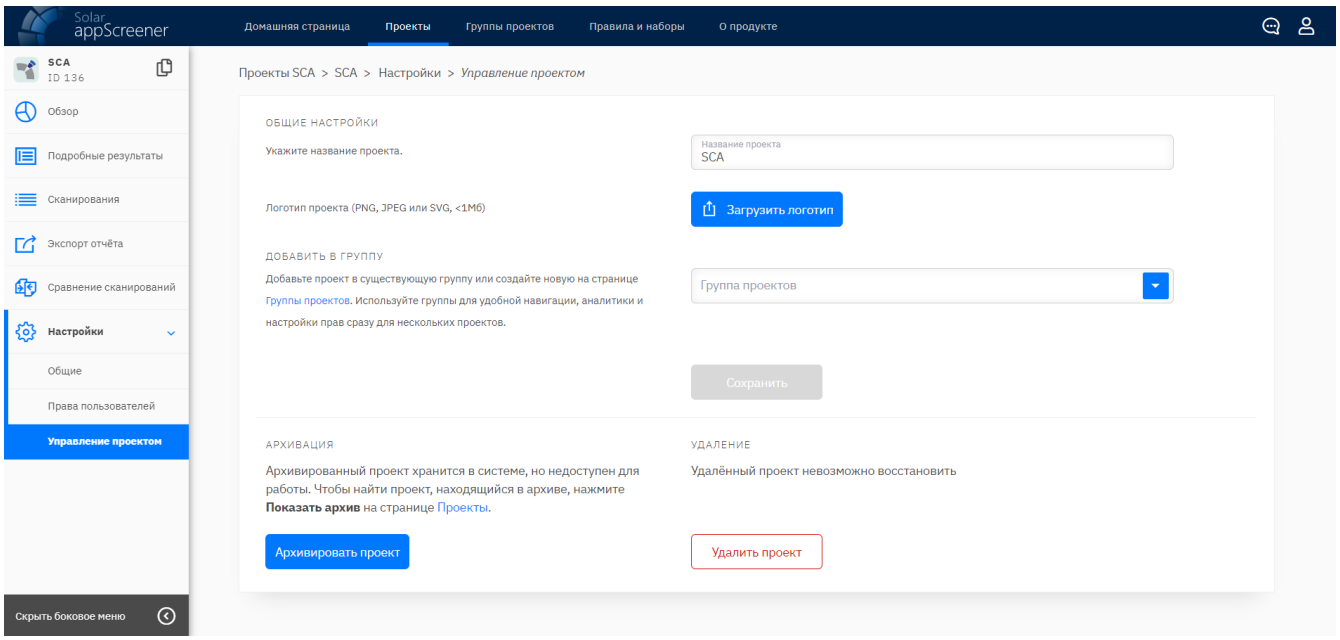


Рис. 6.149: Управление проектом

7. СПИСОК СТАНДАРТНЫХ БИБЛИОТЕК И ФРЕЙМВОРКОВ

Таблица 7.1: Список поддерживаемых стандартных библиотек и фреймворков

Язык	Библиотеки
Java, **Scala**, **Kotlin**	Android android.accessibilityservice android.accounts android.animation android.annotation android.app android.bluetooth android.companion android.content android.database android.drm android.gesture android.graphics android.hardware android.icu android.inputmethodservice android.location android.media android.mtp android.net android.nfc android.opengl android.os android.preference android.print android.printservice android.provider android.renderscript android.sax android.security android.service android.speech android.system

Язык	Библиотеки
	android.telecom
	android.telephony
	android.test
	android.text
	android.transaction
	android.util
	android.view
	android.webkit
	android.widget
	com.android.internal.util
	dalvik.annotation
	dalvik.bytecode
	dalvik.system
	Android Support Library
	Apache Ant
	org.apache.tools.mail
	org.apache.tools.bzip2
	org.apache.tools.tar
	org.apache.tools.zip
	Apache Camel
	Apache Commons
	Apache Groovy
	org.apache.groovy
	groovy
	groovyjarjarantlr
	groovyjarjarantlr4
	groovyjarjarasm.asm
	groovyjarjarcommonscli
	Apache HttpComponents
	Apache Log4j
	org.apache.logging.log4j
	Apache Struts
	Apache Tomcat API
	org.apache.jasper
	org.apache.tomcat
	org.apache.juli
	org.apache.naming
	org.apache.el
	org.apache.coyote

Язык	Библиотеки
	Apache Xalan Java
	org.apache.xml
	org.apache.xpath
	Apache Xerces
	ASM
	AssertJ
	Bean Validation API
	CDI(Contexts and Dependency Injection for Java) APIs
	javax.decorator
	Clojure
	ClojureScript
	cljs
	cognitect
	Codehaus Jackson
	Crashlytics
	Dom4J
	EasyMock
	EclipseLink
	FindBugs
	Firebase Messaging
	Google
	Google APIs for Android
	com.google.android.vending
	Google Cloud Messaging
	Google Guice
	Gson
	Guava: Google Core Libraries for Java
	com.google.thirdparty
	GWT
	ccom.google.web
	H2 Database Engine
	Hamcrest
	Hibernate API
	HttpClient API
	HyperSQL Database
	Jackson
	Java Servlet
	JavaMail API
	com.sun.mail

Язык	Библиотеки
	Javassist
	Javax Annotation API
	Javax Inject
	Jetty
	Joda Time
	JSOUP
	OpenStreetMap
	JUnit
	org.junit
	Logback
	Maven Plugin API
	Mockito
	MySQL java connector API
	com.mysql.cj
	OpenStreetMap
	OOXML Schemas
	OSGi Core
	Play Framework
	Plexus Common Utilities
	PowerMock
	Project Lombok
	Reflections API
	SAX API
	Vaadin
	ScalaCheck
	Scala JS
	Scala Library
	scala.annotation
	scala.beans
	scala.collection
	scala.compat
	scala.concurrent
	scala.io
	scala.math
	scala.ref
	scala.reflect
	scala.runtime
	scale.sys
	scala.text

Язык	Библиотеки
	scala.util
	Scalac SCoverage Plugin
	Scala Test
	Simple XML Framework
	SLF4J
	Spock Framework API
	org.spockframework
	SpringFramework
	Sun Org Apache XML
	TestNG
	com.beust.testng
	W3C DOM
JS	Bootstrap
	jQuery mobile
	Node Express
	Node.js
	Preact
	React
	React Native
	vue.js
C#	ASP.NET
	ASP.NET Core
	Entity Framework
Go	Gin
Swift	Network Framework
PHP	Symphony
Python	Flask
	Django
	SQLAlchemy platform

8. СПИСОК ПОДДЕРЖИВАЕМЫХ РАСШИРЕНИЙ ФАЙЛОВ

Таблица 8.1: Список поддерживаемых расширений файлов

Язык	Расширение
1C	.bsl, .os
ABAP	.abap
Apex	.cls
C#	.cs
C/C++	.cpp, .cc, .c++, .c
COBOL	.cbl, .cob, .cpy
Config files	.config, .xml, .properties, .policy, .aspx, .ini, .plist
Dart	.dart
Delphi	.pas, .dpr, .dpk, .pp
GO	.go
Groovy	.groovy, .gsh, .gvy, .gy
HTML	.aspx, .cshtml, .js, .jsf, .jsm, .jsp, .jspx, .htm, .html, .phtml, .php, .ts, .vbhtml, .vue, .xht, .xhtml
Java, Scala, Kotlin	.java, .scala, .kt, .class
JavaScript	.cshtml, .js, .jsf, .jsm, .jsp, .jspx, .htm, .html, .phtml, .php, .ts, .vbhtml, .xht, .xhtml
LotusScript	.lsl, .lss
Objective-C, Swift	.m, .mm
Pascal	.pas
PHP	.php, .php3, .php4, .php5, .phps, .phpt, .phtml
PL/SQL	.pck, .pkb, .pkh, .pks, .plsql, .prc, .sp, .spb, .spp, .sps, .sql, .st, .trg
Python	.py
Perl	.pl, .pm, .cgi, .plx
Ruby	.rb
Rust	.rs
Solidity	.sol
T-SQL	.sp, .sql, .tsql
TypeScript	.ts, .tsx
VB.NET	.vb
VBA	.vba
VBScript	.asp, .hta, .htm, .html, .vbe, .vbs, .wsc, .wsf
Visual Basic 6	.vb, .bas, .cls, .frm
Vyper	.vy
Config	.config, .plist, .wsdl, .xml, .xsd, .properties, .policy

9. ОБОЗНАЧЕНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ В СПЕЦИФИКАЦИИ API

Таблица 9.1: Обозначения языков программирования в спецификации API

Оригинальное название языка	Обозначение в спецификации API
1C	ONES
ABAP	ABAP
Java for Android	ANDROID
Apex	APEX
C#	CS
C/C++	CCPP
COBOL	COBOL
Config	CONFIG
Dart	DART
Delphi	DELPHI
GO	GO
Groovy	GROOVY
HTML5	HTML5
Java	JAVA
JavaScript	JAVASCRIPT
Kotlin	KOTLIN
LotusScript	LOTUS
Objective-C	OBJC
Pascal	PASCAL
PHP	PHP
PL/SQL	PLSQL
Python	PYTHON
Perl	PERL
Ruby	RUBY
Rust	RUST
Scala	SCALA
Solidity	SOLIDITY
Swift	SWIFT
T-SQL	TSQL
TypeScript	TYPESCRIPT
Visual Basic.NET	VBNET
Visual Basic for Applications	VBA
Visual Basic Script	VBSCRIPT

Оригинальное название языка	Обозначение в спецификации API
Visual Basic	VB
Vyper	VYPER