DerScanner

Руководство пользователя

DerScanner
Версия 11.2

User Guide

Содержание

1 Введение 5
1.1 Описание возможностей 5
1.2 Технические условия использования DerScanner 6
1.3 Начало работы 6
1.3.1 Уровни критичности 6
1.3.2 Уровень безопасности приложения 7

2 Статический анализ 8
2.1 Особенности запуска анализа 8
2.1.1 Анализ JVM и Android-приложений 8
2.1.2 Анализ iOS-приложений 9
2.1.3 Анализ кода на кода на C, C++, Objective-C и Swift 9
2.1.4 Анализ macOS-приложений 10
2.1.5 Анализ 1C-приложений 10
2.2 Настройки сканирования 15
2.2.1 Исключение файлов из анализа 16
2.3 Пользовательский интерфейс 17
2.3.1 Создание проекта 17
2.3.2 Запуск сканирования 18
2.3.3 Работа с результатами 18
2.3.4 Сканирования 31
2.3.5 Сравнение сканирований 33
2.3.6 Настройки 34
2.3.7 Правила и наборы 35
2.4 API 51
2.4.1 Запуск сканирования из API 51
2.4.2 Экспорт отчёта из API 52
2.5 Инструмент командной строки 52
2.5.1 Запуск сканирования из командной строки 52
2.5.2 Экспорт отчёта из инструмента командной строки 54
2.6 Запуск сканирования из инструментов сборки 57

2.7 ИИ инструменты 58
2.7.1 DerTriage 58
2.7.2 DerCodeFix 58

3 Динамический анализ 59
3.1 Особенности запуска анализа 59
3.2 Настройки сканирования 62
3.3 Пользовательский интерфейс 63
3.3.1 Создание проекта 63
3.3.2 Запуск сканирования в UI 64
3.3.3 Работа с результатами 64
3.3.4 Сканирования 71
3.3.5 Сравнение сканирований 72
3.3.6 Настройки 73
3.4 Корреляция результатов с проектами статического анализа 74
3.4.1 Запуск сканирования 74
3.4.2 Подробные результаты 74
3.4.3 Экспорт отчёта 75
3.5 API 75
3.5.1 Запуск сканирования из API 75
3.5.2 Экспорт отчёта из API 75
3.6 Инстумент командной строки 75
3.6.1 Запуск сканирования из командной строки 75
3.6.2 Экспорт отчёта из инструмента командной строки 77

4 Компонентный анализ 80
4.1 Особенности запуска анализа 80
4.1.1 Инструкция по сборке SBOM файла 81
4.2 Настройки сканирования 83
4.3 Пользовательский интерфейс 83
4.3.1 Создание проекта 83
4.3.2 Запуск сканирования 84
4.3.3 Работа с результатами 84
4.3.4 Сканирования 92
4.3.5 Сравнение сканирований 94
4.3.6 Настройки 95

4.3.7 API 96
4.3.8 Инструмент командной строки 96

5 Интерфейс пользователя: общая часть 101
5.1 Домашняя страница 101
5.2 Проекты 101
5.2.1 Очередь сканирований 102
5.3 Группы проектов 103
5.3.1 Управление группами 103
5.3.2 Аналитика 105
5.4 О продукте 107
5.5 Аккаунт 107
5.5.1 Профиль 107
5.5.2 Доступы 108
5.5.3 Шаблоны 110
5.5.4 Перенос данных 111

6 Интеграции DerScanner 113
6.1 Jira 113
6.2 Gitlab CI 114
6.2.1 Пример настройки с использованием API 115
6.2.2 Пример настройки с использованием CLT 116
6.3 DefectDojo 117
6.4 Jenkins 117
6.4.1 Установка расширения в Jenkins 117
6.4.2 Общие настройки расширения 119
6.4.3 Конфигурация анализа и отчёта 120
6.4.4 Конфигурация Build Failure Conditions 126
6.4.5 Результаты и отчёт 128
6.5 Azure DevOps Server 131
6.5.1 Установка расширения 131
6.5.2 Добавление шага для сборки в Azure DevOps Server 132
6.6 TeamCity 136
6.6.1 Интеграция DerScanner через плагин 136
6.6.2 Интеграция DerScanner с помощью Command Line Runner 137

6.7 IntelliJ IDEA 138
6.7.1 Установка плагина 138
6.7.2 Использование 139
6.7.3 Запуск 141
6.8 Eclipse 141
6.8.1 Установка плагина 141
6.8.2 Использование 142
6.8.3 Запуск 142
6.9 Visual Studio 143
6.9.1 Установка расширения 143
6.9.2 Использование 145
6.10 SonarQube 148
6.10.1 Установка расширения 148
6.10.2 Настройки расширения 149
6.11 Автоматическое сканирование из VCS хостингов 152
7 Соответствие классификации CWE 153
8 Список поддерживаемых расширений файлов 154
9 Список фреймворков и стандартных библиотек, поддерживаемых SAST 156
10 Манифесты, поддерживаемые для генерации SBOM 158
11 Обозначения языков программирования в спецификации API 159
12 Перечень сокращений 161
13 Глоссарий 162

01 Введение

1.1 Описание возможностей

DerScanner — комплексное решение для анализа безопасности приложений на всех этапах жизненного цикла разработки. DerScanner поддерживает статический анализ кода с возможностью применения искусственного интеллекта для триажа и генерации рекомендаций по исправления уязвимостей, оценку качества кода, динамический анализ веб-приложений, анализ состава программного обеспечения и выявление атак на цепочку поставок (Supply Chain атаки).

С помощью API и интеграций DerScanner (с IDE, CI/CD) позволяет гибко встраивать анализ в процесс безопасной разработки (SSDLC), своевременно выявлять уязвимости и дефекты, обеспечивать соотвествие стардартам качества, повышать надёжность разрабатываемого продукта и эффективность процесса его разработки.

Статический анализ в DerScanner предоставляет следующие возможности:

- Анализ кода на языках ABAP, Apex, ASP.NET, Bash, C#, C/C++, COBOL, Dart, Delphi, GO, Groovy, HTML, Java, Java для Android, JavaScript, JSP, Kotlin, LotusScript, Objective-C, Pascal, PHP, PL/SQL, PowerShell, Python, Perl, Ruby, Rust, Scala, Solidity, Swift, T-SQL, TypeScript, VB 6.0, VB.NET, VBA, VBScript, Visual Basic, Vyper, XML, YAML, TOML, 1C. Возможна загрузка исходных или бинарных файлов, анализ кода из архивов или по ссылке на репозиторий Git или Subversion.
- Анализ конфигурационных файлов.
- Анализ приложений для ОС Android (`apk`-файлов) и iOS (`ipa`-файлов).
- Генерация рекомендаций по исправлению уязвимостей (DerCodeFix).
- Триаж уязвимостей с применением искуственного интеллекта (DerTriage).

Вместе со статическим анализом кода DerScanner может выполнить **анализ качества кода** для проектов на JavaScript, TypeScript, Delphi и Pascal. Результаты включают оценки качества кода, которые помогают выявлять потенциальные проблемы и улучшить надежность кода.

Динамический анализ обеспечивает тестирование методом “чёрного ящика”, что позволяет обнаруживать уязвимости и закладки в веб-приложениях в момент их выполнения без доступа к исходному коду. Если исходный код приложения доступен, результаты динамического анализа могут быть сопоставлены с результатами статического анализа.

Компонентный анализ позволяет сканировать библиотеки с открытым исходным кодом на наличие уязвимостей, рисков цепочки поставок и лицензионных рисков. Функциональность доступна для проектов на C/C++, C#, Dart, Erlang, GO, Java, JavaScript, Kotlin, Objective-C, PHP, Python, Ruby, Rust, Scala, Swift, TypeScript, VB.NET. DerScanner использует базы данных NVD, GitHub, GitLab, OSV, а также собственные базы данных для обнаружения уязвимостей.

Для проектов, написанных на JavaScript/TypeScript, Python, C# или Java в рамках анализа состава ПО в DerScanner доступна опция **Гибридного анализа SCA + SAST** (анализа достижимости). Гибрийдный анализ отслеживает выполнение уязвимых функций библиотеки в коде приложения.

Для всех типов анализа DerScanner позволяет просматривать и управлять результатами сканирований, генерировать отчёты.

1.2 Технические условия использования DerScanner

Чтобы начать работу с DerScanner, потребуется компьютер с доступом к внутренней корпоративной сети, где установлен DerScanner, и один из браузеров Mozilla Firefox, Google Chrome, Safari или Microsoft Edge.

1.3 Начало работы

Для входа в систему откройте веб-браузер и введите адрес установки DerScanner, указанный вашим администратором. После загрузки страницы авторизуйтесь, используя ваши учетные данные.

После входа будет открыта **Домашняя страница** системы. Домашняя страница — отправная точка для работы с анализатором.

Новый проект для определённого типа анализа можно создать следующим образом: выберите нужный тип анализа перейдя соответствующую вкладку и укажите ресурс/выюерите код, который хотите проанализировать.

При нажатии на кнопку **Начать сканирование** будет запущено сканирование со стандартными настройками.

1.3.1 Уровни критичности

Уязвимости и риски в DerScanner делятся на четыре уровня критичности: критический, средний, низкий и информационный.

Критические уязвимости обладают высокой степенью риска и, как правило, приводят к компрометации конфиденциальных данных или нарушению целостности системы.

Уязвимости среднего уровня представляют умеренную угрозу: они могут привести к тем же последствиям, что и критические, но с меньшей вероятностью, либо указывать на менее серьёзные нарушения безопасности.

Уязвимости низкого уровня обычно не представляют немедленной угрозы, но потенциально могут быть использованы для атаки при наличии дополнительных условий.

Информационные уязвимости не являются непосредственной угрозой безопасности, но указывают на отклонения от лучших практик программирования и могут служить ориентиром для улучшения качества кода.

Недостатки **Критического уровня** касаются исполнения кода и могут привести к нестабильности работы приложения и сложностям в отладке.

Недостатки

Для анализа качества кода недостатки по аналогии делятся на четыре уровня критичности, которые которые соотвествуют серьезности недостатка.

1.3.2 Уровень безопасности приложения

Уровень безопасности приложения рассчитывается на основе результатов его сканирования отдельно для каждого типа анализа на основе количества критических и уязвимостей среднего уровня. При этом критические уязвимости оказывают большее влияние на итоговую оценку и учитываются независимо от объема кода.

При расчете безопасности по статическому анализу уязвимости среднего уровня оцениваются пропорционально их количеству относительно общего числа строк исходного кода. Для проектов DAST и SCA рейтинг также зависит от количества критических и уязвимостей среднего уровня, однако при его расчёте объём исходного кода не учитывается.

02 Статический анализ

2.1 Особенности запуска анализа

Если источник кода для анализа — ссылка на приватный репозиторий, укажите данные для авторизации в блоке **Настройки приватного репозитория**. Машина, на которой установлен DerScanner, должна иметь доступ к указанному репозиторию.

Чтобы проверить содержимое вложенного архива, включите настройку сканирования **Анализировать библиотеки и вложенные архивы**.

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 Git:

- https://gitlab.example.com/project.git;
- ssh://gitlab.example.com/project.

Примеры ссылок Subversion:

- https://svn.example.com/project/trunk/;
- svn://svn.example.com/project/branches/branch;
- svn+ssh://svn.example.com/project/.

DerScanner поддерживает запуск анализа для таких форматов архивов с исходым или байт кодом: ZIP, 7z, RAR (до версии 4.0), EAR, AAR, tar.bz2, tar.gz, tar, cpio.

Для исполняемых файлов поддерживаются следующие форматы: `.war`, `.jar`, `.apk`, `.aab`, `dll`, `exe`, `.ipa`, `.apk` или `app`-файл.

Полный список форматов файлов, которые анализируются при загрузке архива или по ссылке на репозиторий, представлен в таблице (табл. 8.1).

2.1.1 Анализ JVM и Android-приложений

Для проектов на Java, Scala, Kotlin и Android-приложений DerScanner выполняет анализ не по исходному коду, а по его промежуточному представлению близкому к байт-коду.

Для высокой точности анализа такой проектов рекомендуется (один из двух вариантов):

- передавать на анализ исходный и байт-код проекта: архив, в котором есть исходные файлы (расширения `.java`, `.scala`, `.kt`) и соответствующие им `.class`-файлы на верхнем уровне архива (допустим анализ `.jar` и `.war` при выполнении этих условий). И активировать настройку сканирования.
- передавать на анализ исходный проекта и выбирать опции **Анализ кода со сборкой** (в процессе выполнения анализ проект будет собран с помощью внутреннего сборщика стандартными командами) или **Анализ исходного кода с собственной сборкой** и указать команду сборки (опция будет доступна при подключении инструмента сборки

администратором системы. В командах сборки рекомендуется передавать абсолютные пути до инструментов/скриптов/утилит. Например, для Maven: **/usr/local/bin/mvn clean package**.).

Обратите внимание: команды сборки, переданные через UI DerScanner, вместе с исходным кодом запускаются сервисом с правами пользователя **derscanner** в рабочей директории агента.

Результаты анализа в таких случаях все равно будут транслированы на исходный код проекта.

Для проектов Java доступна опция анализа только исходного кода (**Анализ исходного кода (проект не будет собран, актуально только для Java)**). В этом случае при анализе будет использоваться меньшее количество правил поиска уязвимостей, чем при анализе собранного проекта.

Для анализа JSP-страниц структура их расположения должна соответствовать структуре `.war`-файла.

2.1.2 Анализ iOS-приложений

Для точности анализа `.ipa`-файлов, рекомендуется передавать на анализ файл, который должен содержать `armv8` (`aarch64`) бинарный файл приложения.

Как подготовить `.ipa`-файл для анализа:

1. Соберите проект в Xcode (получится директория `.app`).
2. Заархивируйте её в `.zip`.
3. Переименуйте .zip в `.ipa`.

2.1.3 Анализ кода на кода на C, C++, Objective-C и Swift

Для проектов C, C++, Objective-C и Swift анализ происходит во время сборки приложения. Полноценный анализ исходного кода C/C++ и Objective-C возможен при условии, что проект собирается без ошибок на машине с установленным компонентом анализа C/C++, Objective-C и Swift DerScanner.

Для проектов выполнения сборки на разных ОС (Linux, Windows и macOS) доступны разные версии компонент анализа. В настройках сканирования доступна настройка выбора ОС сборки.

При анализе исходного кода C/C++ возможен анализ без осуществления сборки, однако при этом будет найдено меньше уязвимостей.

Чтобы провести анализ исходного кода (архив или ссылка на репозиторий), сначала убедитесь, что проект успешно собирается выбранными инструментами без ошибок.

Для сборки C/C++ на **Linux/macOS** поддерживается сборка CMake-проектов версий <= 3.17.3) командой `mkdir build && cd build && cmake .. && make`. Поддерживаемые версии компиляторов: **Apple Clang** <= 11.0.3, **Open source Clang** <= 10.0.0, **GCC** <= 10.1. Для **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 (2005-2019) собираемые командой `msbuild <project_name>.sln`.

Проект должен самостоятельно устанавливать свои зависимости или же они должны быть предварительно установлены на компьютере с запущенным анализатором DerScanner.

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

2.1.4 Анализ macOS-приложений

Для анализа macOS-приложений требуется передавать на анализ `.app`-файл, упакованный в ZIP-архив (поддерживаются бинарные файлы архитектур x86 и x64).

Как подготовить `.app`-файл для анализа:

1. Соберите проект в Xcode (получится директория `.app`).
2. Заархивируйте её в `.zip`.

2.1.5 Анализ 1С-приложений

DerScanner подерживает анализ кода из 1С:Предприятие 8 версии. Для корректного отображения результатов анализа в настройках сканирования необходимо выбрать кодировку “IBM866” в настройке кодировки названий файлов и кодировки исходного кода.

Код для анализа можно выгрузить двумя методами: вручную или автоматически.

Для загрузки кода вручную требуется выполнить:

1. Укажите информационную базу и нажмите кнопку **Конфигуратор**.

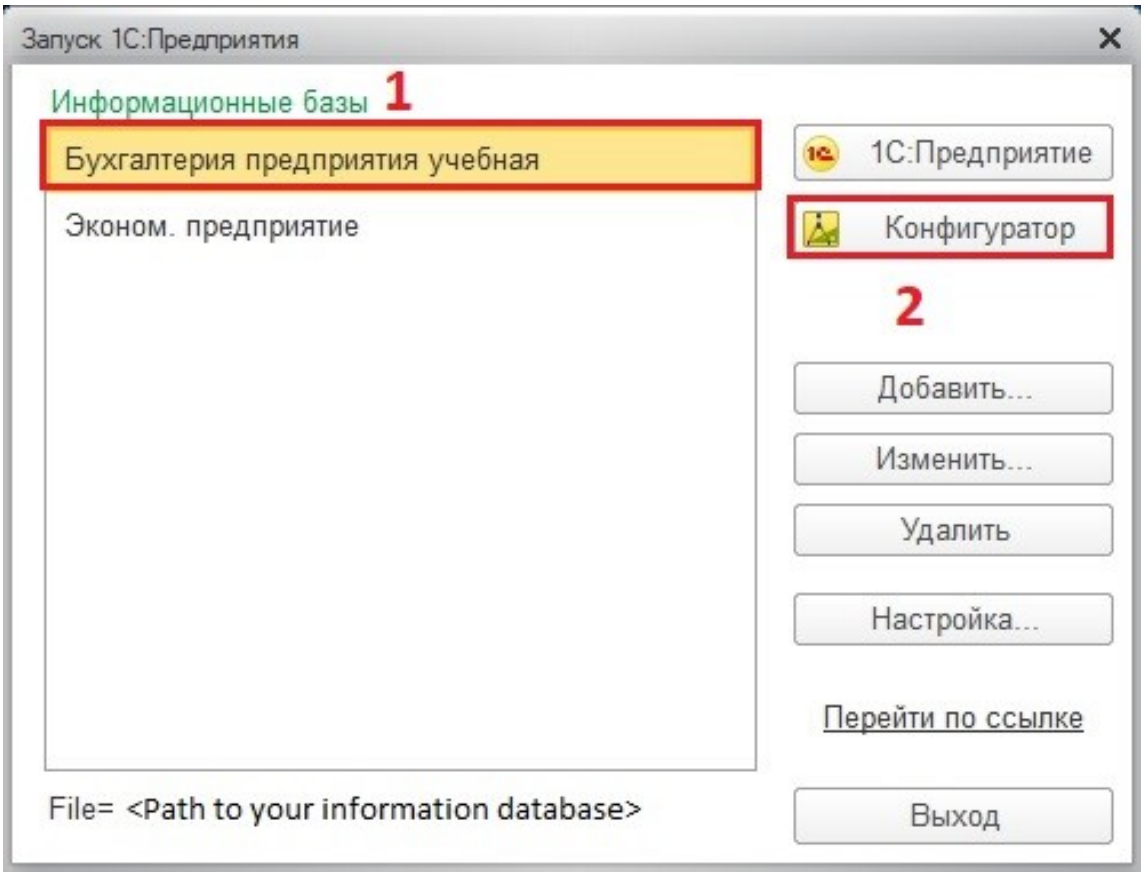


Рис. 2.1: Главное меню

2. В меню **Конфигурация** выберите пункт **Выгрузить конфигурацию в файлы** (возможно предупреждение о длинных именах объектов, его можно проигнорировать).

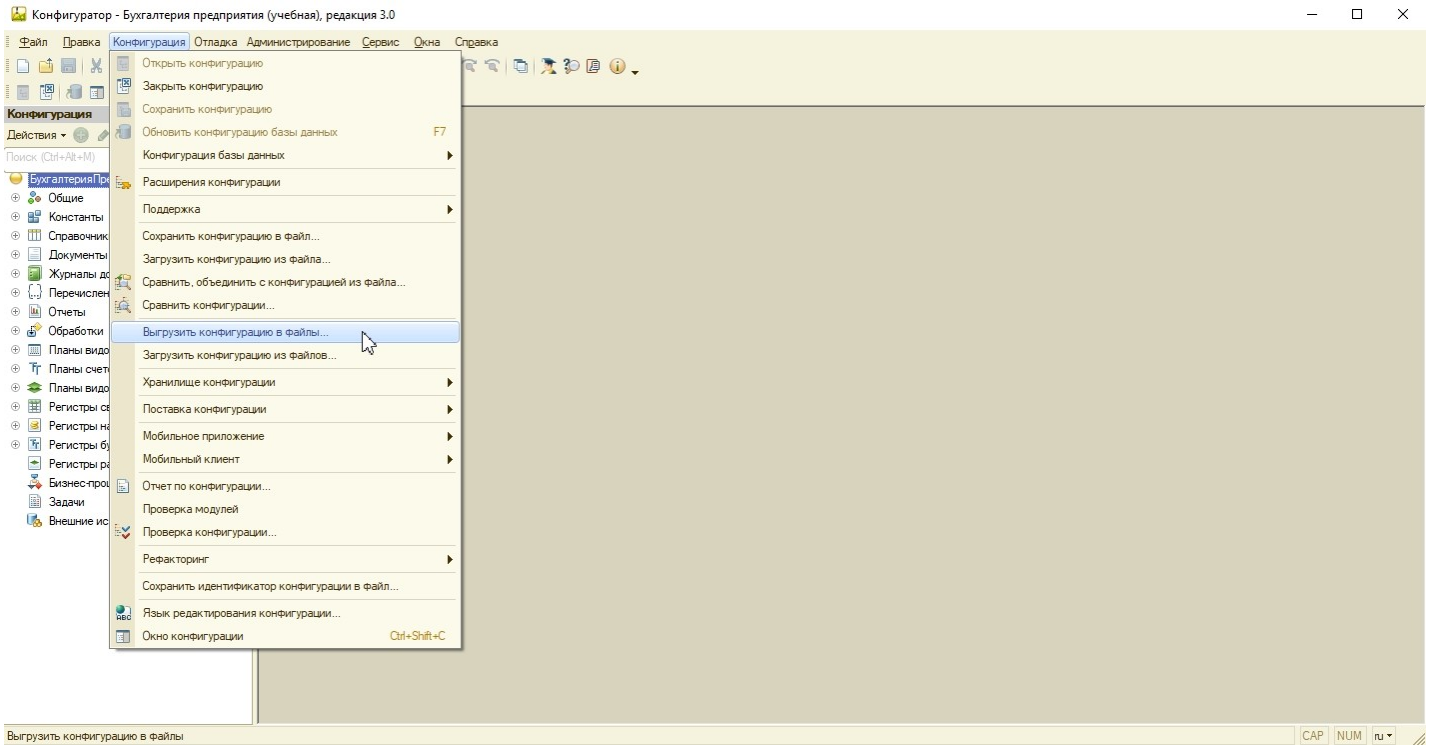


Рис. 2.2: Экспорт

3. Укажите путь к директории выгрузки файлов.

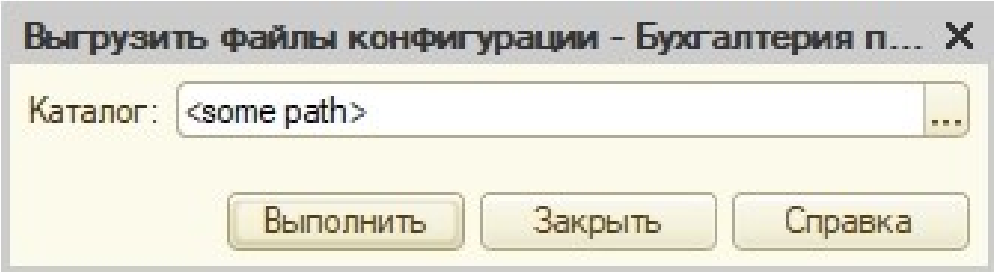


Рис. 2.3: Путь до директории выгрузки файлов конфигурации

4. Добавьте директорию в архив.

Полученный архив готов к анализу в DerScanner (в настройках сканирования необходимо выбрать кодировку “IBM866” в настройке кодировки названий файлов и кодировки исходного кода).

Для автоматический выгрузки на примере «толстого» и «тонкого» клиента необходимо выполнить следующее:

1. Завершите работу с конфигуратором.
2. Откройте для редактирования файл **1c_extractor.bat** и внесите в него значения необходимых переменных.

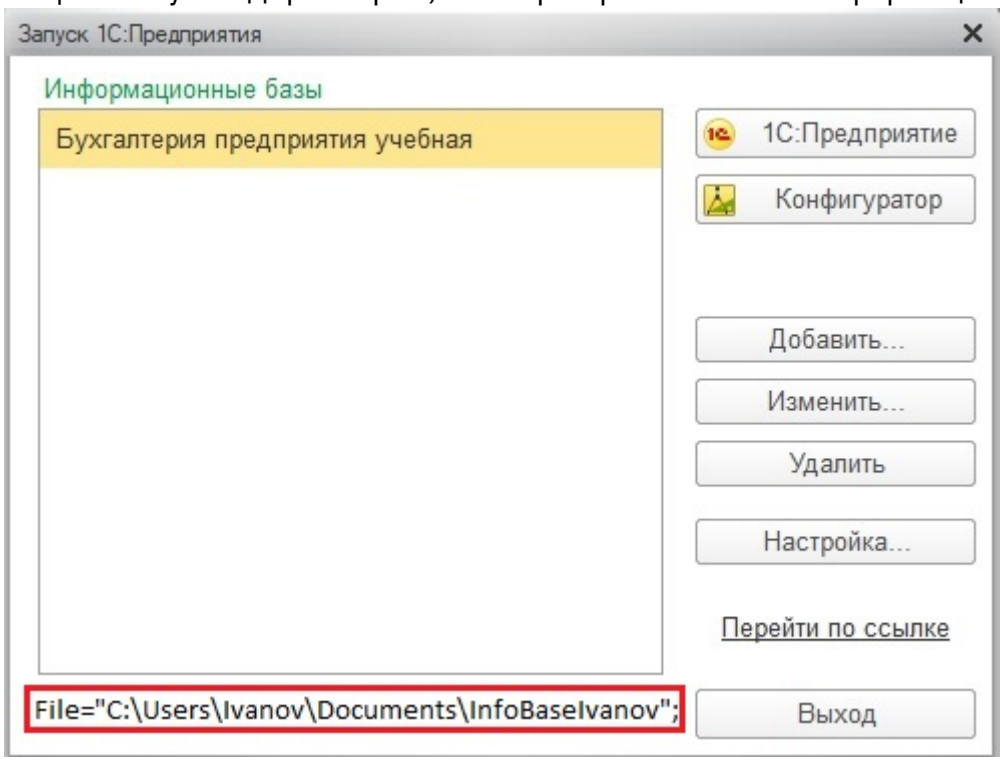
   Для «толстого» клиента:

```
@echo off
```

```
set binpath=<binpath>
set dbpath="<dbpath>"
set username="<username>"
set userpass="<userpass>"
set folder="<folder>"
set logs="<logs>"
start %binpath% CONFIG /N%username% /P%userpass% /F%dbpath%
/DumpConfigToFiles%folder% /OUT%logs%
```

где:

- <binpath> путь до исполняемого файла **1cv8.exe** в директории **bin** подверсии в **C:\Program files\1Cv8**;
- <username> имя пользователя для доступа к базе данных и учётной записи;
- <userpass> пароль доступа к учётной записи и базе данных (если пароль не требуется, параметр можно не добавлять) опустить;
- <dbpath> путь к директории, в которой расположена информационна база:



- <logs> путь к файлу с логами процесса;
- <folder> путь к каталогу объектов метаданных конфигурации.

Пример:

```
@echo off
set binpath=C:\"Program Files (x86)"\1cv8\8.3.13.1690\bin\1cv8.exe
set dbpath="C:\Users\Smirnov\Documents\InfoBaseSmirnov"
set username="Smirnov"
```

```
set userpass="Password"
set folder="C:\Users\Smirnov\Desktop\ExtractModules"
set logs="C:\Users\Smirnov\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=<binpath>
set server="<server>"
set username="<username>"
set userpass="<userpass>"
set folder="<folder>"
set logs="<logs>"
start %binpath% CONFIG /N%username% /P%userpass% /S%server%
/DumpConfigToFiles%folder% /OUT%logs%
```

где:

- <binpath> путь до исполняемого файла **1cv8c.exe** в директории **bin** подверсии в **C:\Program Files (x86)\1Cv8**;
- <username> имя пользователя для доступа к базе данных и учётной записи;
- <userpass> пароль доступа к учётной записи и базе данных (если пароль не требуется, параметр можно не добавлять) опустить;
- <server> адрес информационной базы на сервере, который записан в формате: \<имя компьютера>\<ссылочное имя информационной базы>
- <logs> путь к файлу с логами процесса;
- <folder> путь к каталогу объектов метаданных конфигурации.

Пример:

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

3. Сохраните изменения в файле **1c_extractor.bat**.

4. Запустите **cmd** от имени Администратора в директории **1c_extractor.bat**;
5. Выполните команду `call 1c_extractor.bat` (время выполнения зависит от объема конфигруации и в среднем составляет 5-10 минут, **Конфигуратор** в это время недоступен);
6. Откройте директорию указанную в `<folder>`.

Полученный архив готов к анализу в DerScanner (в настройках сканирования необходимо выбрать кодировку “IBM866” в настройке кодировки названий файлов и кодировки исходного кода).

2.2 Настройки сканирования

Сканирование можно настроить через интерфейс с помощью соответствующих параметров, аналогичные настройки доступны также через API и в инструменте командной строки (CLT).

Настройки выполнения статического анализа в DerScanner можно разделить на такие категории:

- Выбор проекта DAST для корреляции.
- Настройка отправки отчёта с результатами сканирования по его завершении (можно несколько e-mail адресов для получения);
- Выбор приоритета сканирования.
- Выбор агента, который будет выполнять сканирование. Выбранный агент выполнит анализ тех языков проекта, которые он поддерживает.
- Общие настройки:
  - «Предобработка». Настройка сделает код читаемым, но может увеличитьб время сканирования;
  - «Анализировать библиотеки и вложенные архивы». Активация настройки необходима для того, чтобы анализировать архивы, которые могут сожержаться в проекте с исходным кодом;
  - «Использовать пользовательсике паттерны». С активаций кроме системный правил в процессе анализа будут участвовать пользовательские правила.
  - «Инкрементальный анализ». При выполнении инкрементального анализа проверяются лишь изменённые или вновь созданные файлы, что значительно ускоряет повторный анализ. Однако при внесении изменений в настройки проекта или обновления системы аналитики рекомендуется запустить полный анализ для обеспечения точности и полноты проверки. Инкрементальный анализ может быть применим при анализе исходного кода на ABAP, Apex, C#, COBOL, 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, приложений 1С и конфигурационных файлов.
  - «Сохранить загруженный файл». При выборе опции файл останется доступным для администрирования системы.
- Анализ качества кода: выбор языков для выполнения анализа качества кода.
- Статический анализ кода: выбор языков для выполнения статического анализа кода.
- DerTriage. С активной опцией после выполнения статического анализа будет запущен триаж уязвимостей для уязвимостей выбранных уровней критичности.
- DerCodeFix. С активной опцией после выполнения статического анализа будет выполнена генерация исправлений для уязвимостей выбранных уровней критичности.
- Настройки анализа языков:
  - Настройки C/C++:

    - Выбор ОС для сборки проектов C/C++: macOS, Linux, Windows.
    - «Межмодульный анализ». Опция активирует межмодульниый анализ для проекта, что повысит точность анализа, но может увеличить время сканирования.
  - Настройки Java, Scala, Kotlin: выбор опции сборки проекта.
- Настройки приватного репозитория:
  - Поля для ввода учетных данных (логин/пароль, токен, SSH ключ).
  - Поле для указания ветки репозитория Git.
  - «Анализ подмодулей». Опция для включения анализа подмодулей.
- Настройки выбора файлов для анализа.
- Настройки кодировки:
  - Выбор кодировки для названий файлов.
  - Выбор кодировки исходного кода.
- Наборы правил. Выбор наборов правил, правила из которых будут применены при выполнении сканирования.

2.2.1 Исключение файлов из анализа

Исключения файлов могут быть записаны в виде регулярных выражений. При анализе архива необходимо учитывать, что архив является корневой папкой, поэтому в таком случае необходимо в начало добавить `**`.

При записи используются следующие символы:

```
** - 0+ директорий

* - 0+ символов

? - 1 символ
```

Чтобы исключить файл из анализа, указатель на него нужно добавить к этому выражению, например `**/*; !**/*.ts` исключит из анализа все `.ts` файлы из анализа. Для обработки только `.ts`, но исключения `.module.ts` можно указать: `**/*.ts; !**/*.module.ts`.

Каждое отдельное выражение отделяется `;`.

Примеры записи регулярных выражений:

Например, проект имеет такую структуру:

```
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
```

`**/*;!**/{name:.*[tT][eE][sS][tT].*};!**/{name:.*[tT][eE][sS][tT].*}/**/*` - проверять всё, кроме файлов, имя или путь которых содержит слово ‘test’ в любом регистре

`**/*;!**/Archive/**/*` - исключить содержимое библиотеки “Archive”

`!**/*` - не проверять ничего

2.3 Пользовательский интерфейс

2.3.1 Создание проекта

Проект создается на **Домашней странице**. Для создания проекта с запуском сканирования, на вкладке «Статический анализ» загрузите код для анализа в виде архива или укажите ссылку на репозиторий.

Для создания проекта без запуска сканирования, на **Домашней странице** на вкладке “Пустой проект” выберите вариант “Статический анализ”, введите название и нажмите **Создать проект**. При необходимости нажмите **Показать настройки** и установите настройки анализа.

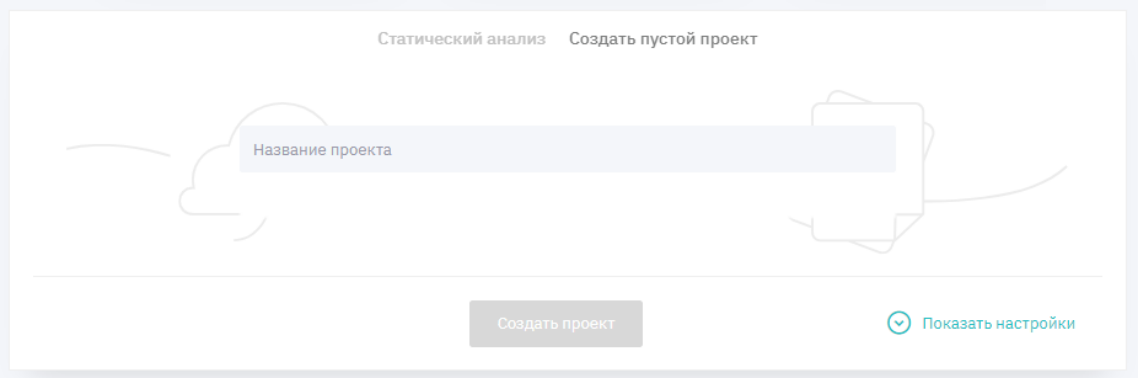


Рис. 2.4: Создание пустого проекта

В созданном проекте можно настроить интеграции и автоматическое сканирование.

2.3.2 Запуск сканирования

Чтобы запустить новое сканирование в UI DerScanner:

1. Загрузите архив с кодом, приложение или укажите ссылку на репозиторий.
2. Укажите настройки сканирования или воспользуйтесь готовым шаблоном.
3. Нажмите **Начать сканирование**.

В зависимости от языка проекта

Если источник кода для аналиа — ссылка на приватный репозиторий, укажите данные для авторизации в блоке **Настройки приватного репозитория**. Машина, на которой установлен DerScanner, должна иметь доступ к указанному репозиторию.

Чтобы проверить содержимое вложенного архива, включите настройку **Анализировать библиотеки и вложенные архивы**.

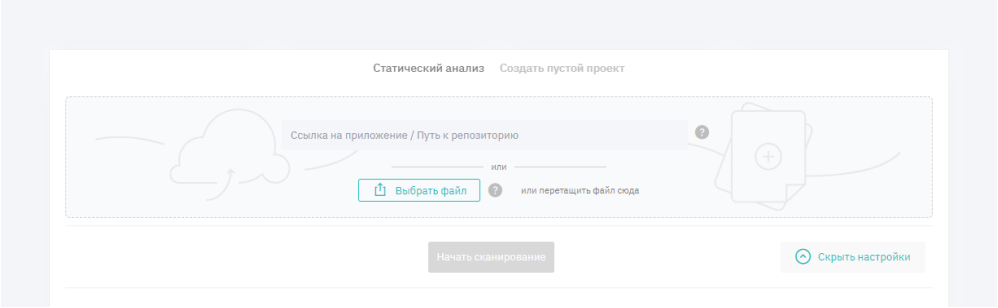


Рис. 2.5: Новое сканирование

2.3.3 Работа с результатами

2.3.3.1 Обзор

Раздел делится на подразделы с результатами статического анализа и анализа качества кода.

В списке правом верхнем углу страницы можно выбрать сканирование, для которого будет отображаться статистика по сканированию.

На странице **Обзор** представлена следующая информация:

- рейтинг;
- статус сканирования;
- продолжительность сканирования;
- количество строк кода;
- количество уязвимостей каждого уровня критичности;
- дашборды со статистикой проекта и сканирования.

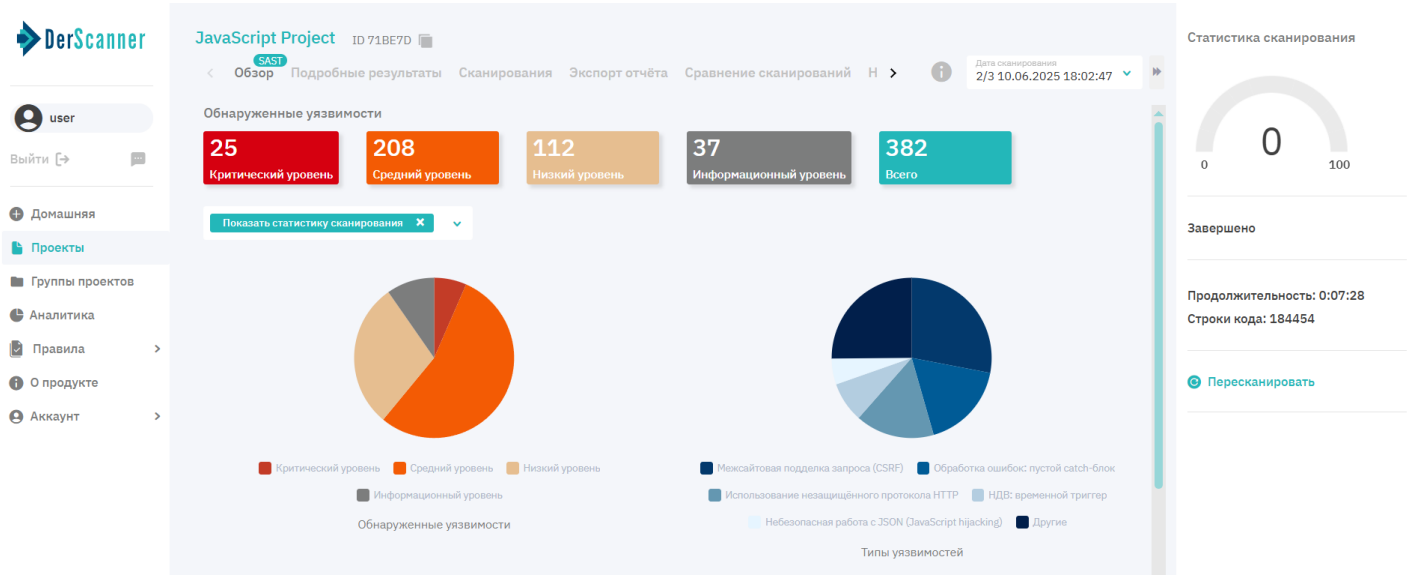


Рис. 2.6: Обзор

Для результатов анализа качества кода DerScanner отображает значения ключевых метрик для каждого из поддерживаемых языков программирования. Среди них:

- цикломатическая сложность;
- сложность восприятия кода;
- процент комментариев;
- процент повторяемости кода;
- длина функций.

Для метрик **Цикломатическая сложность**, **Сложность восприятия кода** и **Длина функций** дополнительно доступны графики, показывающие распределение значений по всему проекту в разрезе выбранного языка.

Значения метрик, превышающие нормальные значения, будут выделены цветом согласно уровню критичности диапазона, в который попало значение.

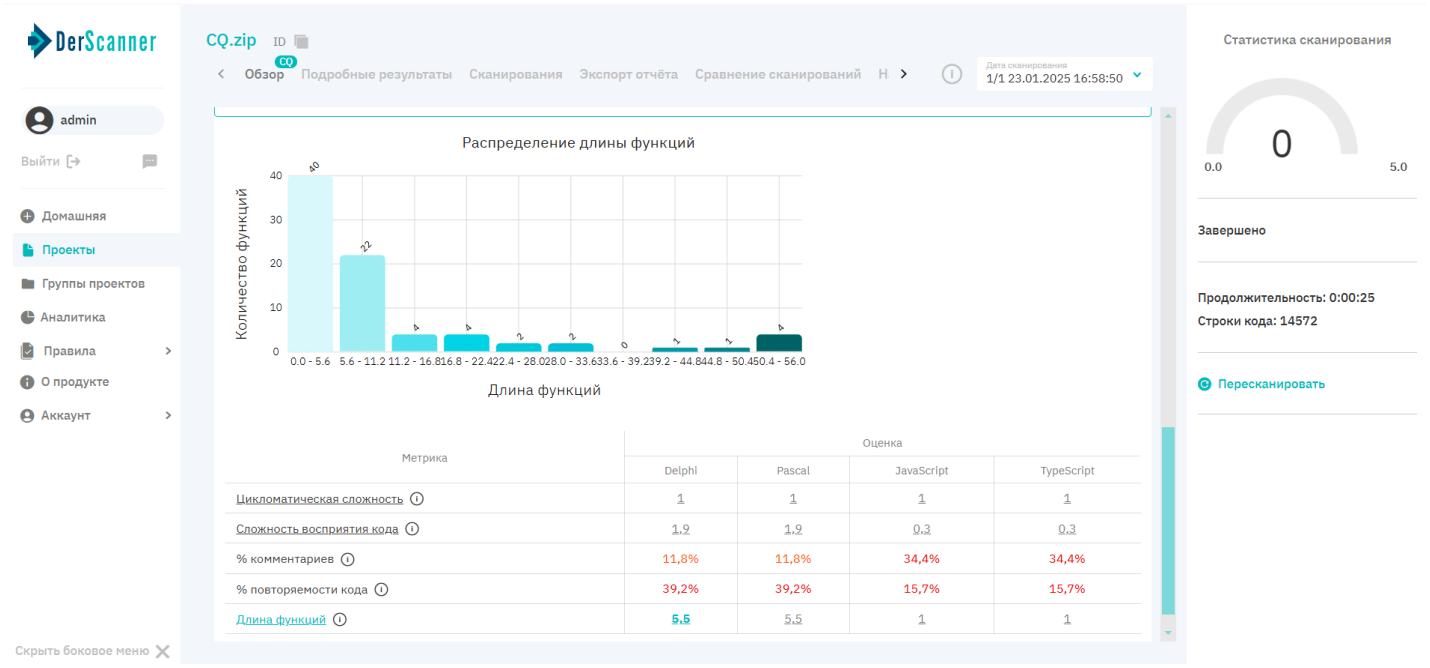


Рис. 2.7: Распределение длины функций

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

2.3.3.2 Подробные результаты

Раздел делится на подразделы с результатами статического анализа и анализа качества кода.

В разделе **Подробные результаты** (рис. 2.8) **для статического анализа** кода отображается информация по каждой из обнаруженных уязвимостей.

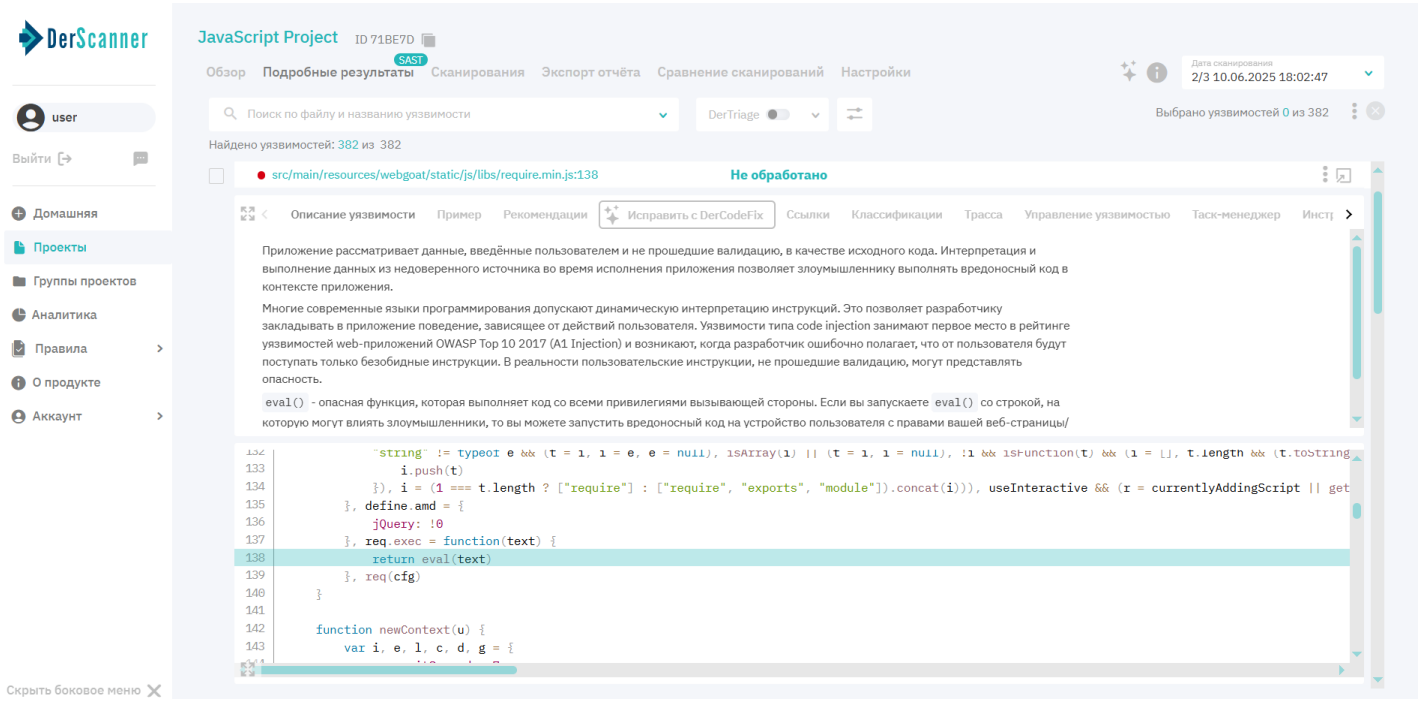


Рис. 2.8: Подробные результаты

По умолчанию уязвимости сгруппированы по языку и названию. По желанию тип группировки можно изменить на:

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

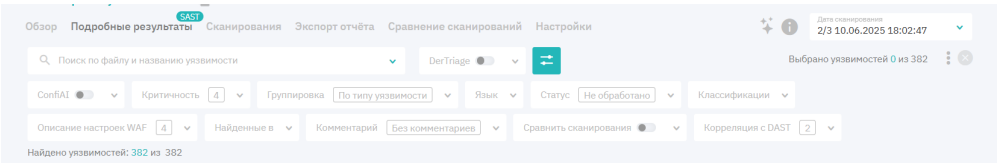


Рис. 2.9: Фильтры результатов

С использованием строки поиска можно найти уязвимости в определенных файлах или уязвимости отпределенного типа.

Фильтровать результаты можно по следующим параметрам:

- по результату триажа уязвимостей — DerTriage (подтвержденные, отклоненные и необработанные компонентом уязвимости);
- по классификации уязвимостей;
- по статусу уязвимостей;
- по типу уязвимостей;
- фильтр ConfiAI 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 — позволяет настроить чувствительность ConfiAI, 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.

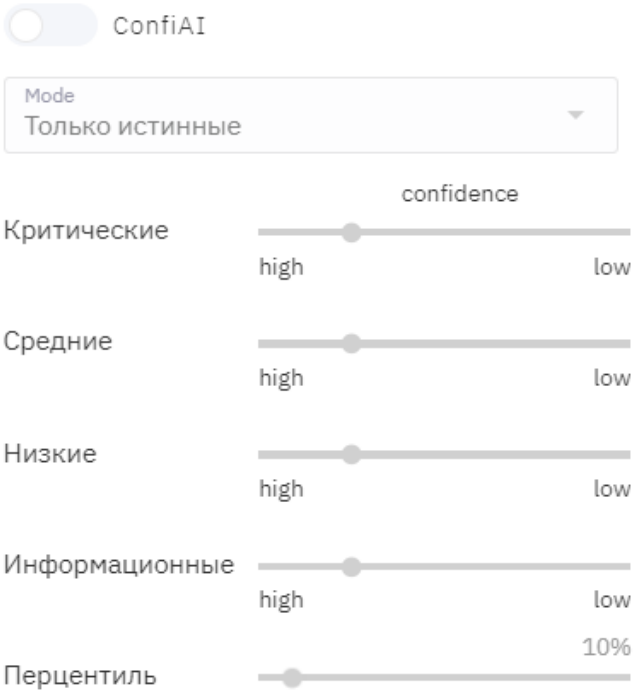


Рис. 2.10: ConfiAI

- по языкам, для которых был произведен анализ;
- по наличию/отсутствию комментариев;
- по наличию/отсутствию связанной задачи в таск-менеджере.

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)

Чтобы изменить статус или критичность уязвимости, воспользуйтесь контекстным меню уязвимости или настройками на вкладке «Управление уязвимостью». Изменить статус/критичность или оставить комментарий для произвольной группы уязвимостей можно установив флаги в чекбоксах рядом с названием уязвимости и использовать контекстное меню над списком уязвимостей.

Изменения статуса и уровня критичности уязвимостей влияют на пересчёт уровня безопасности приложения. Уязвимости, помеченные как **Отклонено**, исключаются из расчёта общего количества уязвимостей и не учитываются при формировании рейтинга

безопасности. Все внесённые изменения сохраняются и остаются актуальными после повторного сканирования. (рис. 2.11).

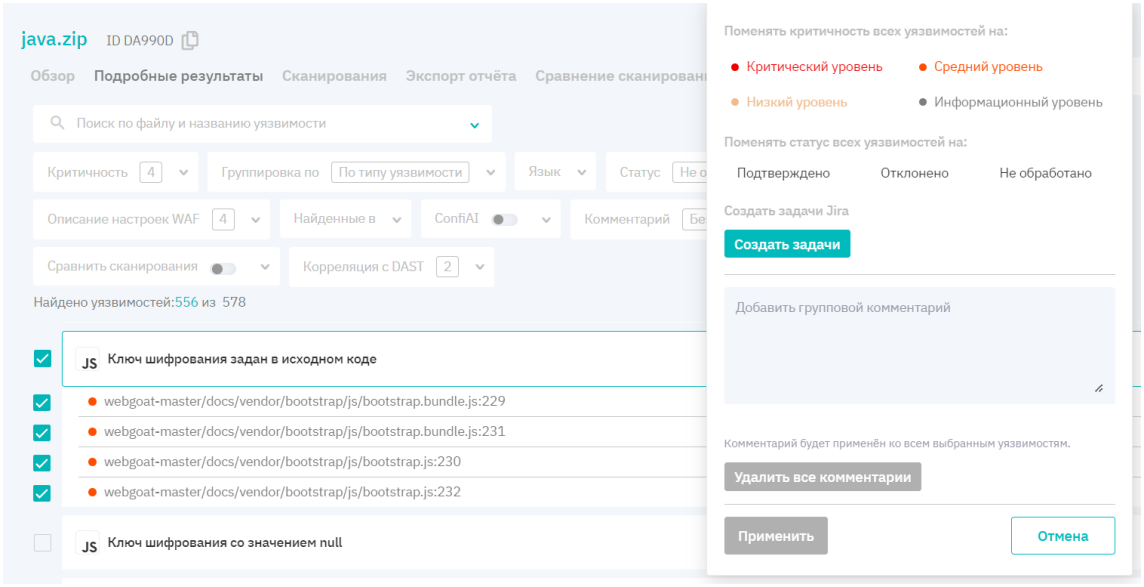


Рис. 2.11: Управление пакетом уязвимостей

После выбора конкретной уязвимости открывается фрагмет с кодом уязвимости и информация о ней. Чтобы раскрыть информацию об уязвимости на полный экран, нажмите на иконку в правом верхнем углу.

В левой части страницы представлена информация по уязвимости (рис. 2.12): **Описание**, **Пример**, **Рекомендации** по устранению, **Ссылки** на ресурсы, связанные с данным типом уязвимостей.

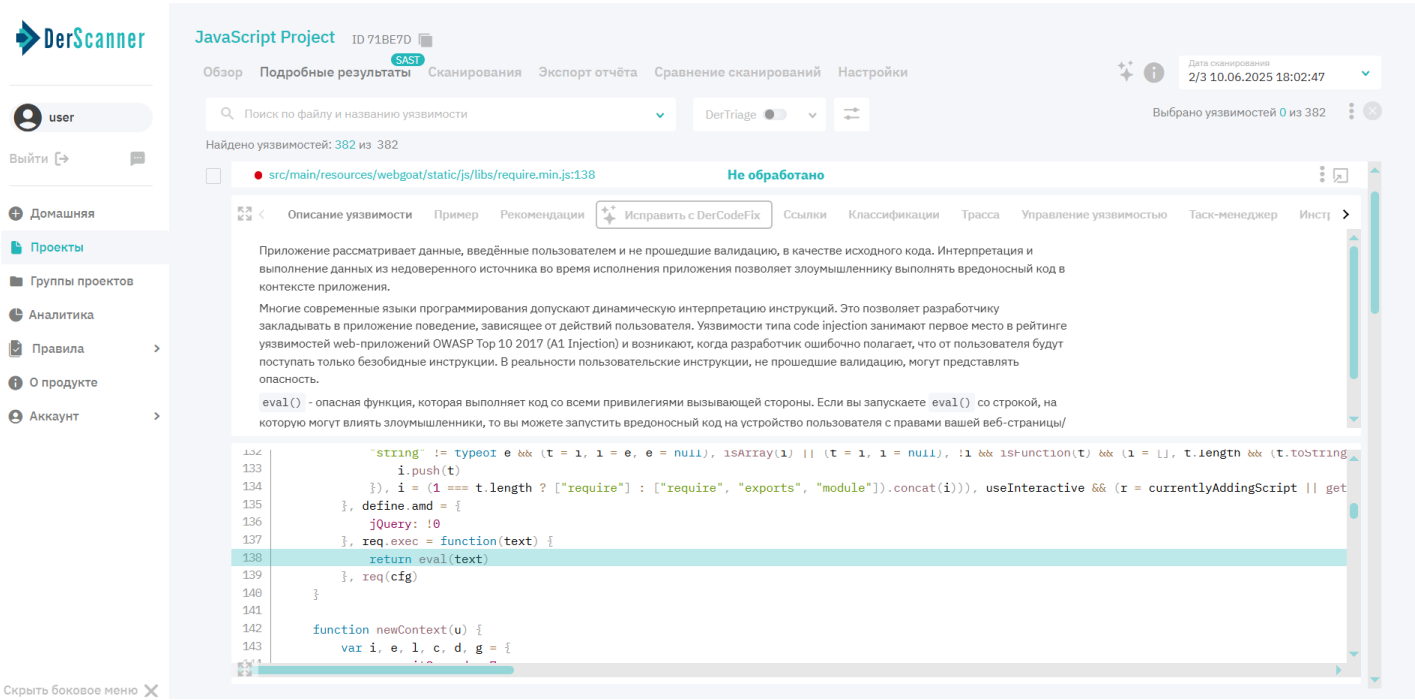


Рис. 2.12: Описание уязвимости

На вкладке **Исправить с DerCodeFix** отображается либо готовое исправление уязвимого фрагмента кода (если генерация исправлений была включена на этапе сканирования), либо кнопка **Сгенерировать исправление**. Чтобы изменить отображение контекста с исправления на оригинальный код и наоборот, используйте переключатель.

На вкладке **Классификации** (рис. 2.13) отображается таблица с перечислением пунктов, которые относятся к данной уязвимости.

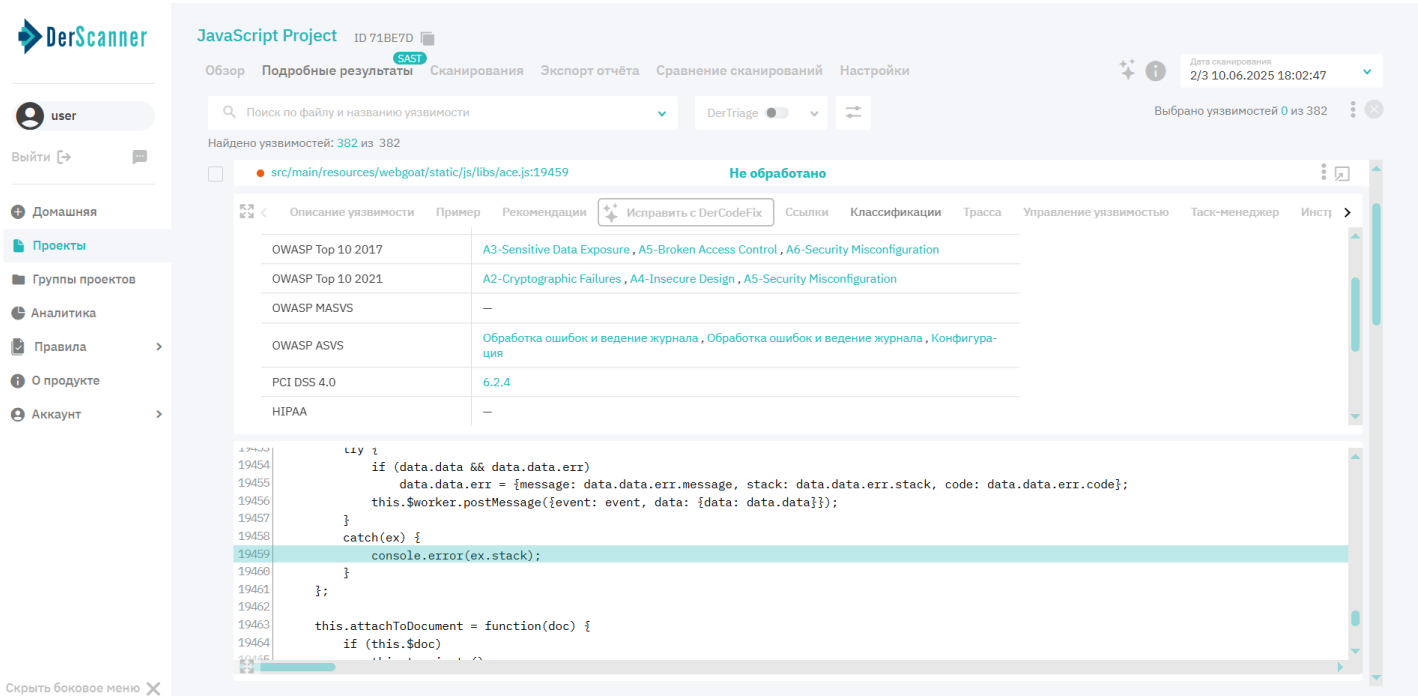


Рис. 2.13: Классификации

Во вкладке **Трасса** (рис. 2.14) отображается путь распространения данных, связанный с уязвимостью. Этот путь представлен в виде диаграммы с интерактивными элементами. При клике на любой из них в центральной части экрана открывается соответствующий фрагмент исходного кода, позволяя детально изучить, как данные проходят через приложение.

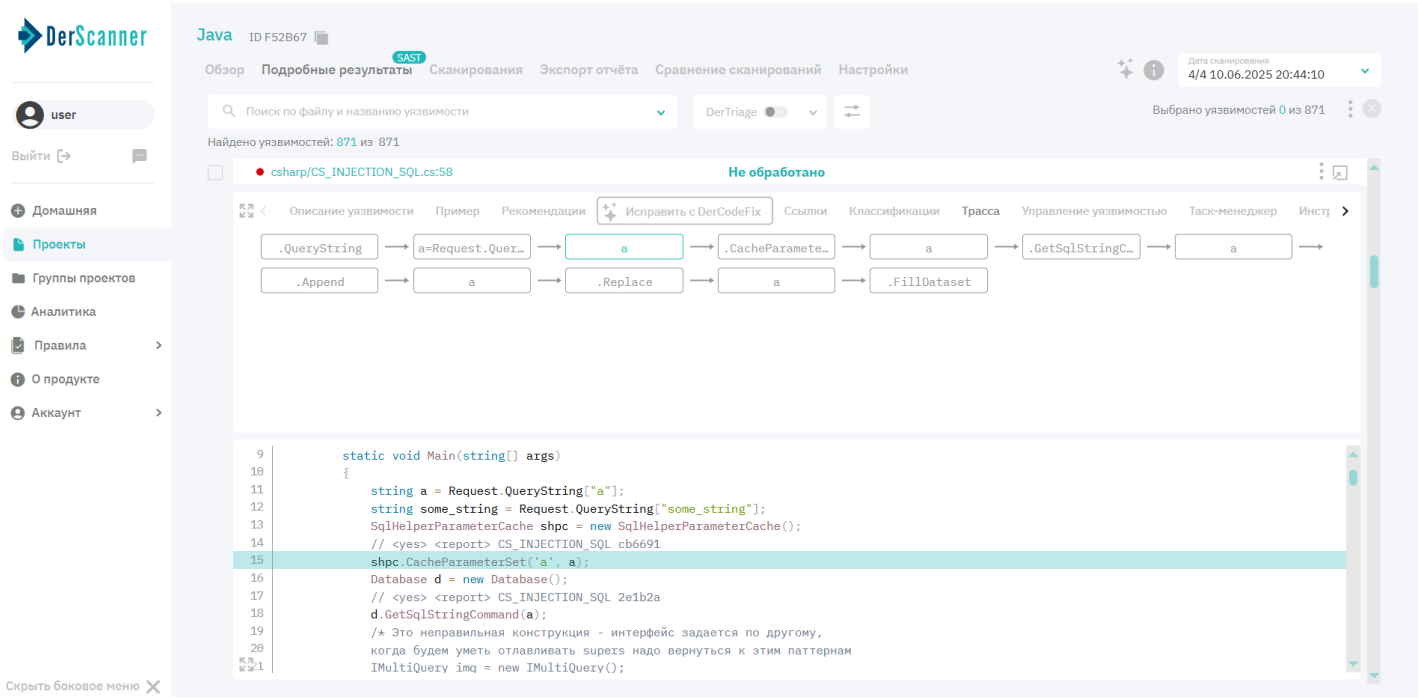


Рис. 2.14: Трасса

На вкладке **Управление уязвимостью** (рис. 2.15) можно изменить уровень критичности и статус, добавить комментарий к вхождению и посмотреть журнал событий, в котором будут отображаться ранее совершённые действия с уязвимостью.

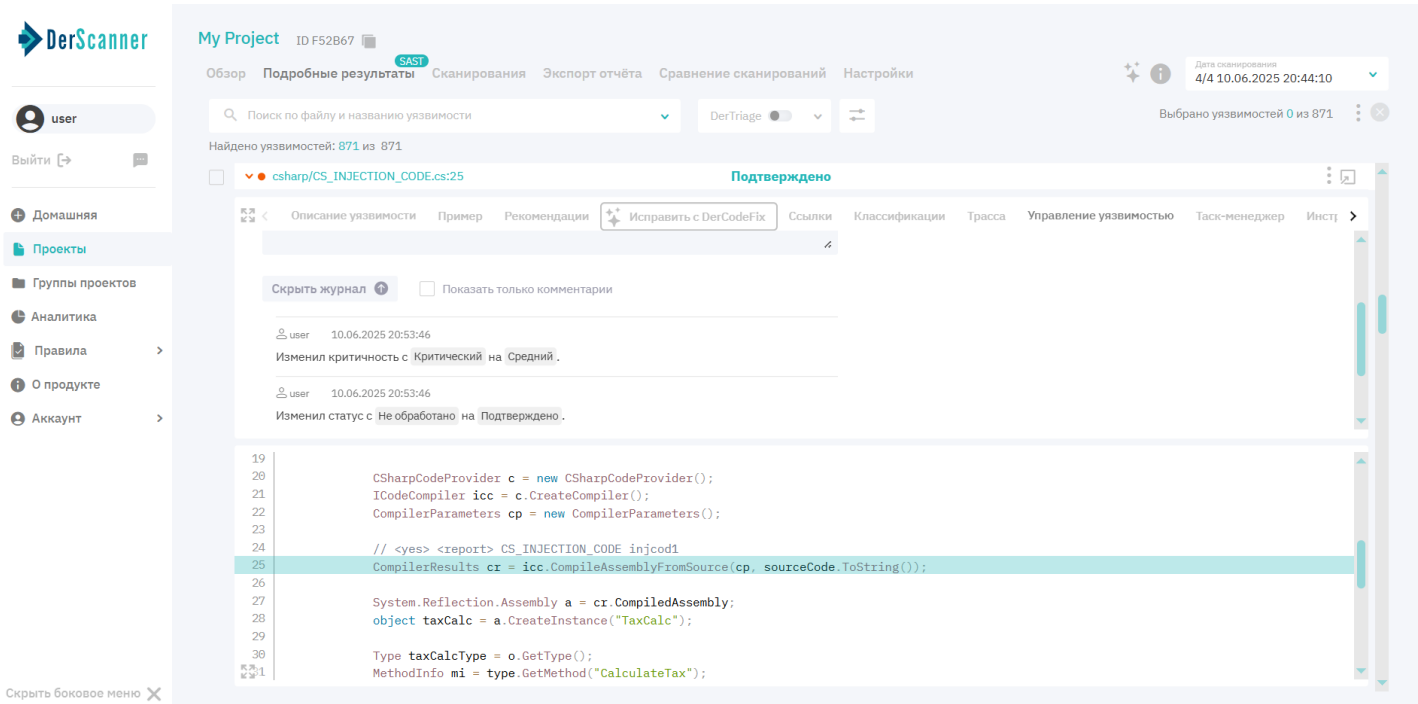


Рис. 2.15: Управление уязвимостью

На вкладке также отображается значение параметра **Confidence** для вхождения. Параметр показывает уверенность ConfiAI, что данная уязвимость не является ложным срабатыванием.

Чем больше значение параметра **Confidence**, тем меньше вероятность, что вхождение ложно-положительное.

Для результатов анализа качества кода на вкладке **Подробные результаты** > **Качество кода** отображается информация по каждому обнаруженному недостатку для выбранного сканирования.

Фильтровать результаты можно по следующим параметрам:

- по статусу недостатка;
- по категории недостатка:
  - дубликаты — вхождения при выявлении дубикатов в коде, сгруппированы по языку и месту в исходном коде;
  - метрики —вхождения при превышении значений метрик качества кода, сгруппированы по названию метрики аналогично вкладке **Обзор**;
  - недостатки — вхождения аналогичные статическому анализу, сгруппированы по языку и типу;
- при наличии двух и более успешных сканирований 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);
- по языкам, для которых был произведен анализ;
- по наличию/отсутствию комментариев.

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

После выбора конкретного вхождения 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: **Описание**, **Пример**, **Рекомендации** по устранению, **Ссылки** на ресурсы, связанные с данным типом недостатка, **Таск-менеджер**.

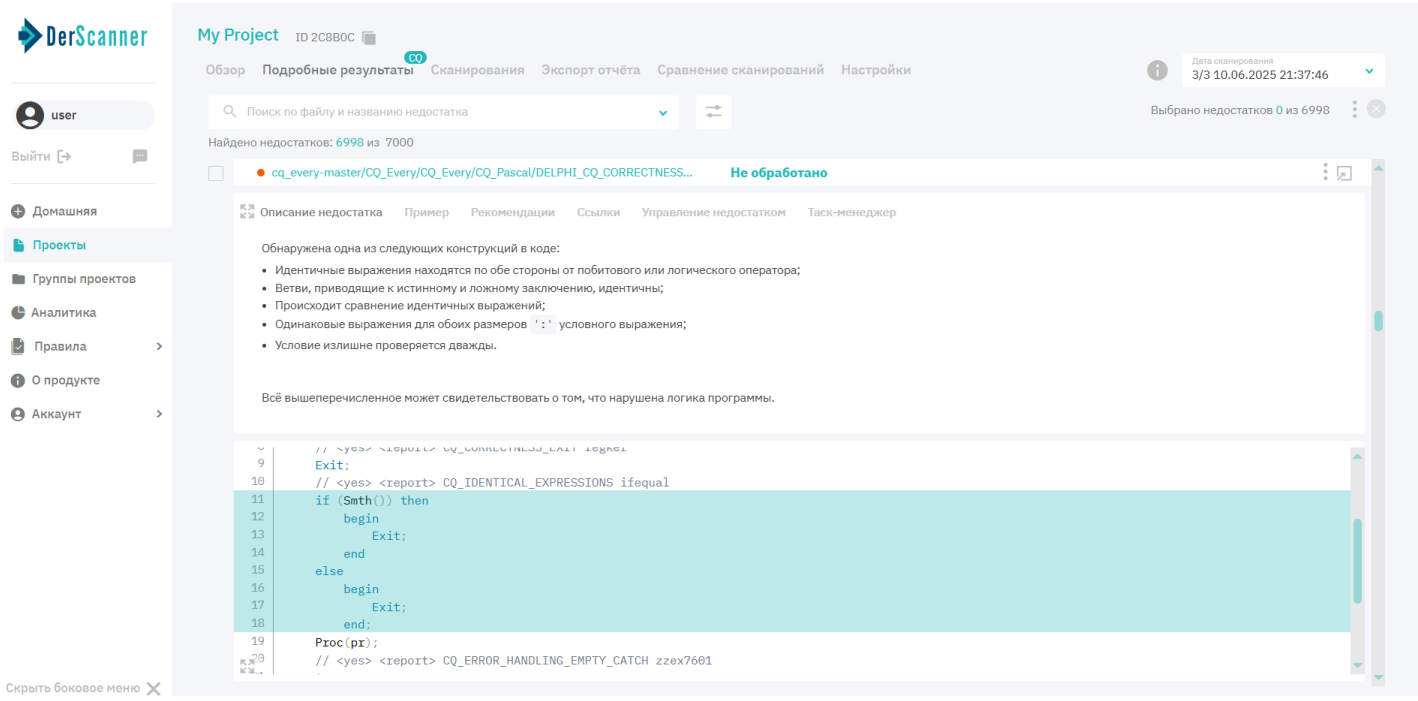


Рис. 2.16: Описание уязвимости

2.3.3.3 Отчёт безопасности

В разделе **Экспорт отчёта** можно выгрузить результаты сканирования в отчёт в формате HTML, PDF, CSV или SARIF и настроить их содержание.

Для формирования отчета с результатами статического анализа и анализа качества кода доступны разные разделы.

Выберите один из готовых шаблонов настроек или задайте информацию для экспорта вручную.

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

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.

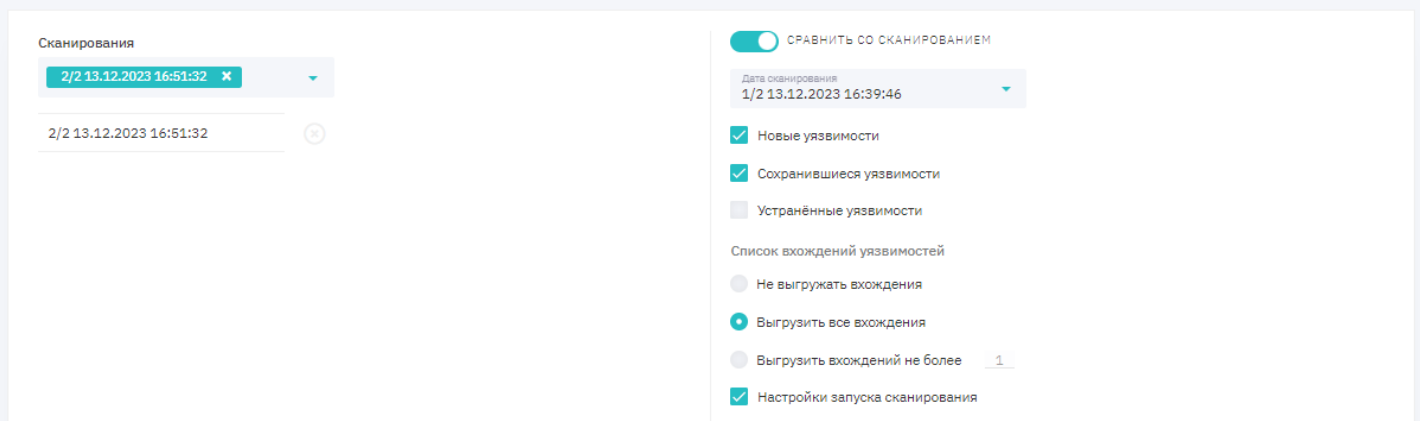


Рис. 2.17: Сравнить со сканированием

2.3.3.3.1 Информация о проекте

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

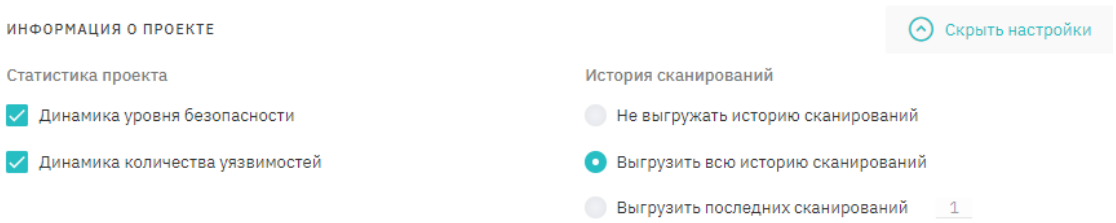


Рис. 2.18: Информация о проекте

2.3.3.3.2 Классификация уязвимостей

Выберите один из способов классификации уязвимостей: По критичности, По секретам, OWASP Top 10 2013, OWASP Mobile Top 2014, OWASP Mobile Top 2014, OWASP Top 10 2017, OWASP Top 10 2021, OWASP ASVS, OWASP MASVS, PCI DSS 4.0, HIPAA, CWE/SANS Top 25 2011, CWE/SANS Top 25 2021 или CWE/SANS Top 25 2023.

При выборе **Сравнить со сканированием** доступна только классификаций уязвимостей **По критичности** и **По секретам**.

При выборе классификации **По секретам** в выборке останутся только уязвимости, связанные с неправильным хранением или утечкой чувствительных данных в коде проекта.

2.3.3.3.3 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.

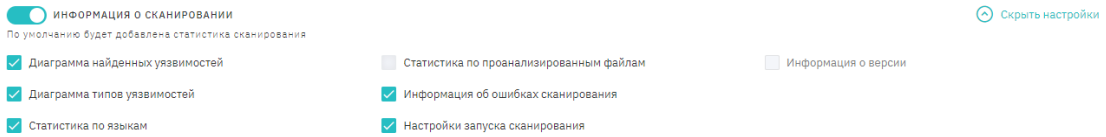


Рис. 2.19: Информация о сканировании

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

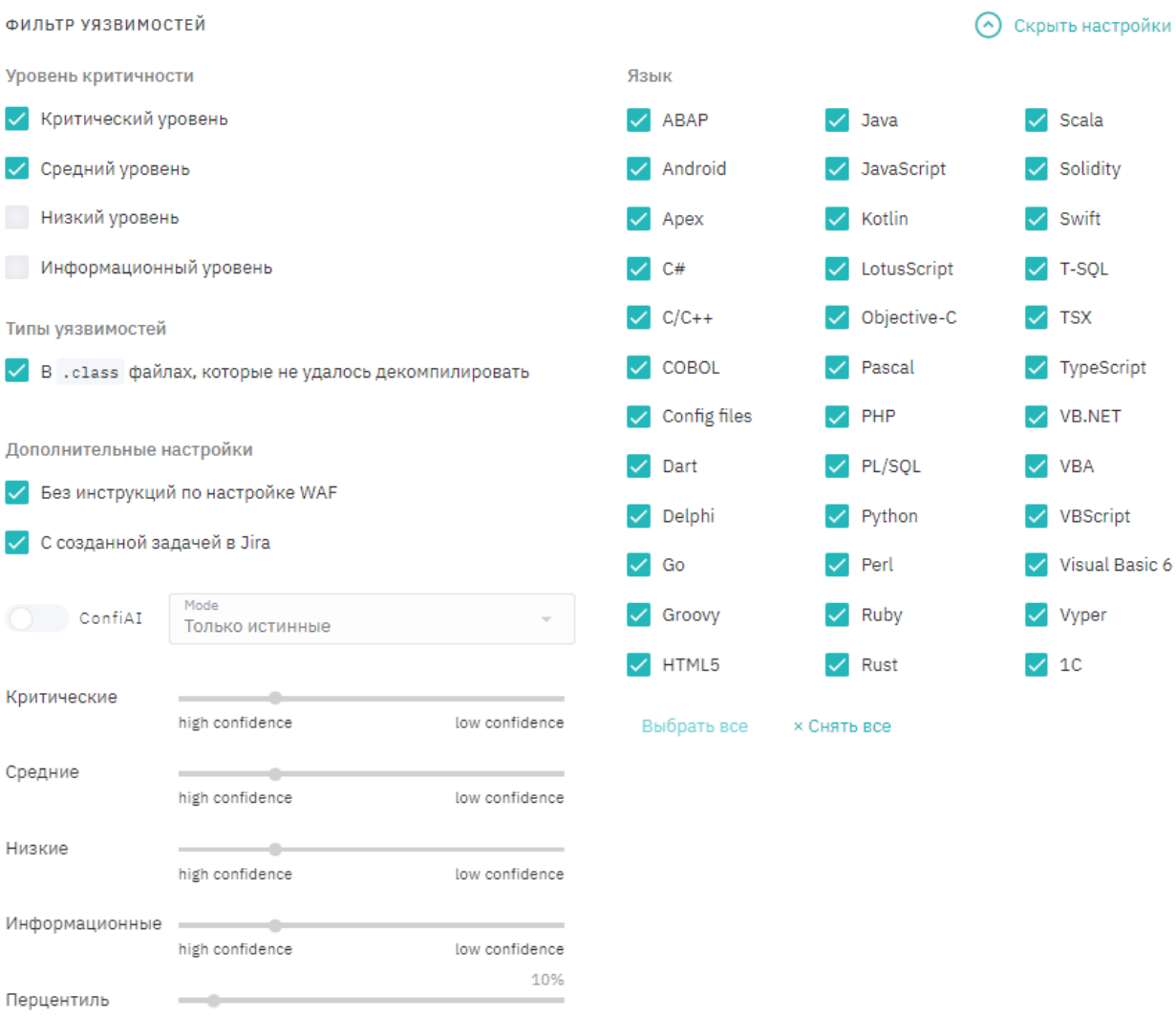


Рис. 2.20: Фильтр уязвимостей

Выберите статусы уязвимостей и задайте количество их вхождений.

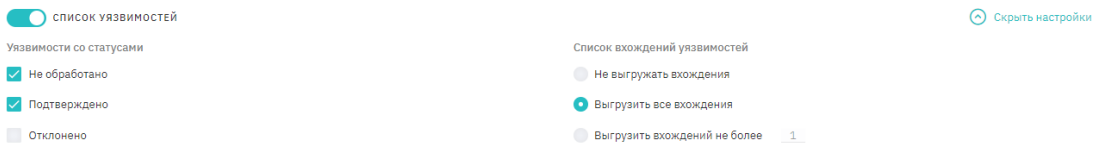


Рис. 2.21: Список уязвимостей

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);
- DerCodeFix: выгрузить исправления (недоступно для SARIF);
- количество вхождений уязвимостей;
- отображение трассы;

- отображение журнала событий:
  - не включать;
  - включить только комментарии;
  - включить только действия;
  - журнал целиком;
- размер контекста исходного кода с уязвимостью.

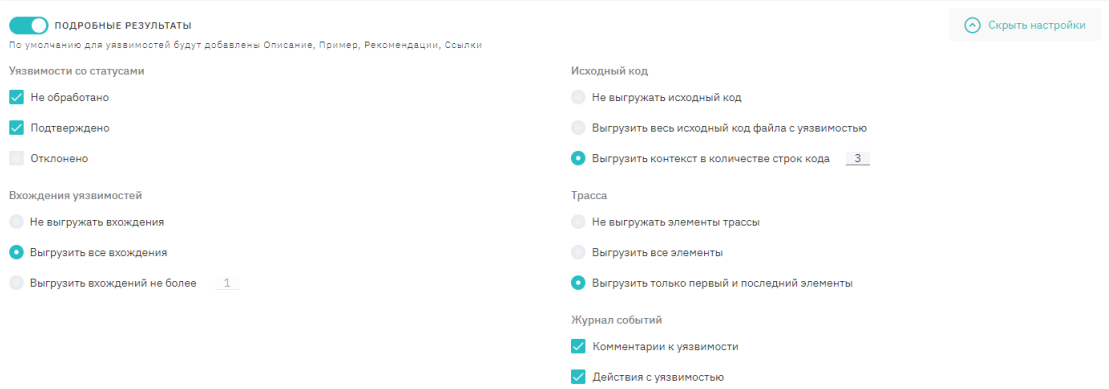


Рис. 2.22: Подробные результаты

2.3.3.3.4 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.

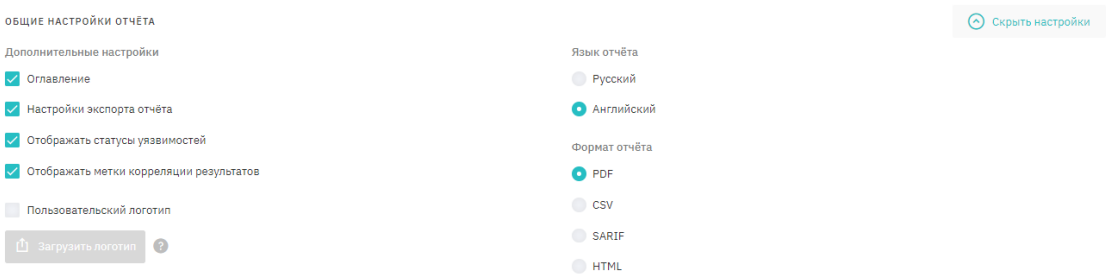


Рис. 2.23: Общие настройки отчёта

Чтобы скачать отчёт, нажмите **Скачать**.

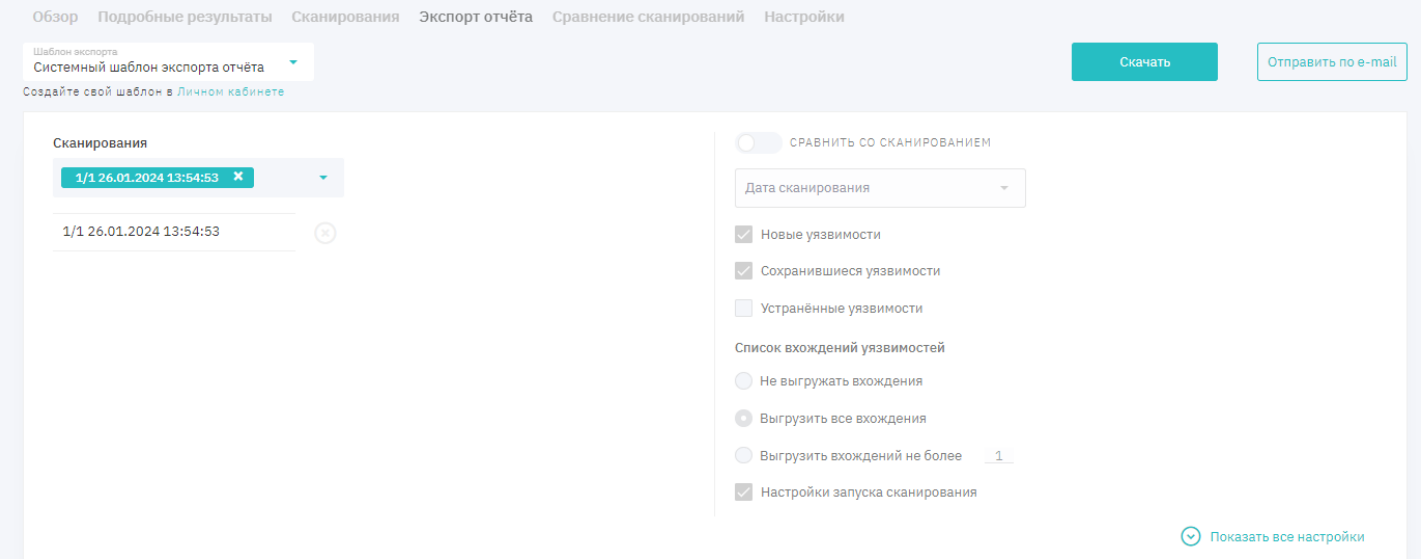


Рис. 2.24: Экспорт

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

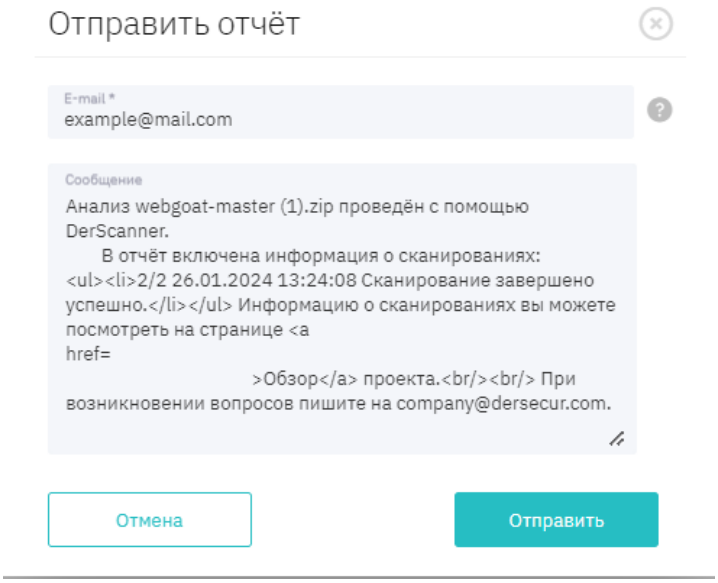


Рис. 2.25: Отправить отчёт по e-mail

2.3.4 Сканирования

В этом разделе (рис. 2.26) можно просматривать и управлять сканированиями, которые сходят в проект. Список поддерживает сортировку по дате запуска, продолжительности выполнения или рейтингу. Чтобы изменить порядок, нажмите на заголовок соответствующего столбца; повторное нажатие инвертирует порядок сортировки.

Для каждого сканирования отображается:

- дата и время его запуска;

- иконка (при нажатии открывается окно «Информация о сканирования» с настроками этого сканирования);
- метка «CQ», если в рамках сканирования был выполнен анализ качества кода;
- контекстное меню сканирования, которое позволяет:
  - выгрузить отчёт по результатам сканирования;
  - архивировать сканирование;
  - удалить сканирование.
- статус сканирования и эпат (при наличии);
- перечень языков, которые были обнаружены в проекте в результате анализа;
- общее время выполнения сканирования (продолжительность);
- количество проанализированный строк кода;
- статистика по уязвимостям (критическим, средним, низким и информационного уровня) или статистика по недостаткам;
- рейтинг приложения.

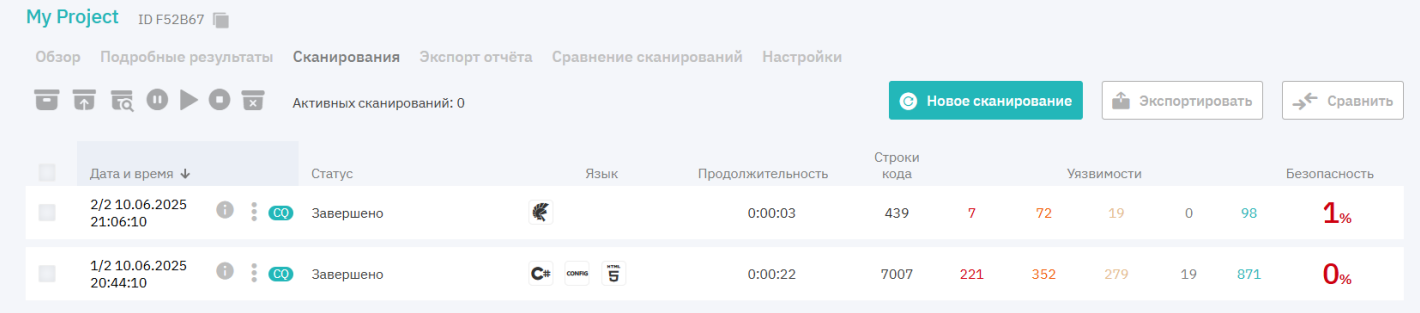


Рис. 2.26: Сканирования

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**.

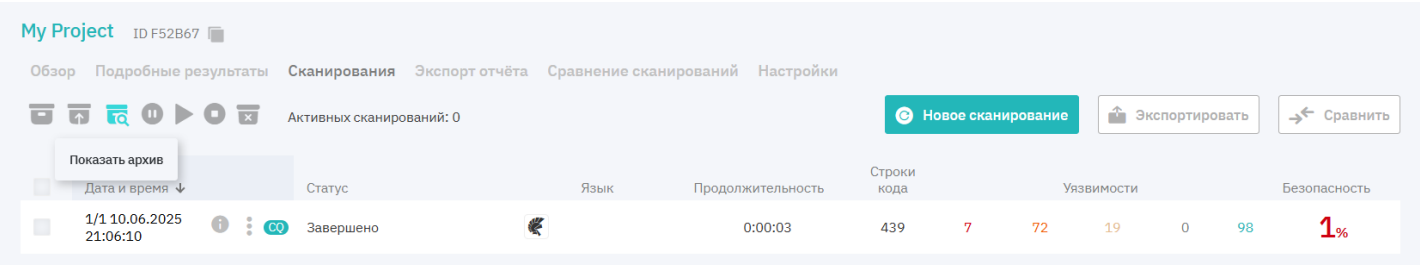


Рис. 2.27: Показать/Скрыть архив

Чтобы повторно проанализировать проект, используйте кнопку **Новое сканирование** (рис. 2.27). Это позволит инициировать дополнительное сканирование с новыми или изменёнными параметрами, не создавая новый проект.

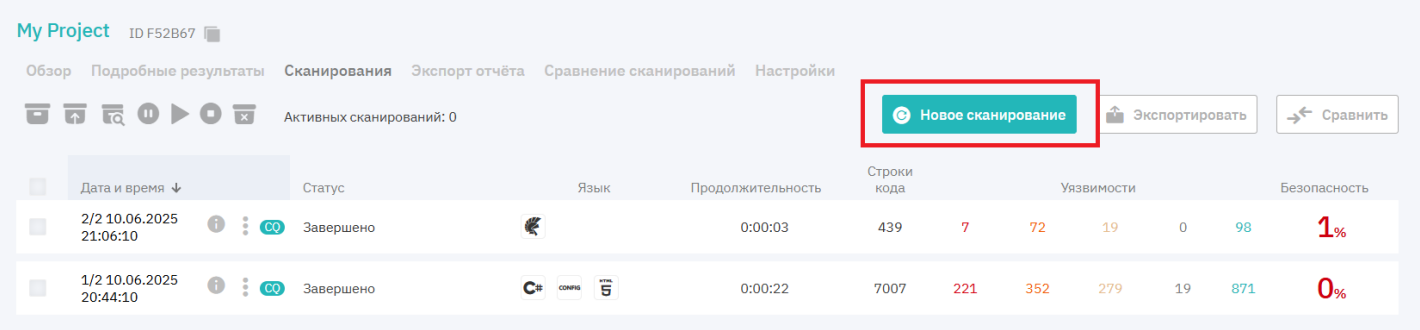


Рис. 2.28: Новое сканирование

В DerScanner предусмотрена возможность одновременного выполнения нескольких сканирований в рамках одного проекта. Текущий статус каждого из сканирований отображается в колонке **Статус** таблицы, а на странице **Очередь сканрований** можно посмотреть информацию по всем активным сканированиям в системе.

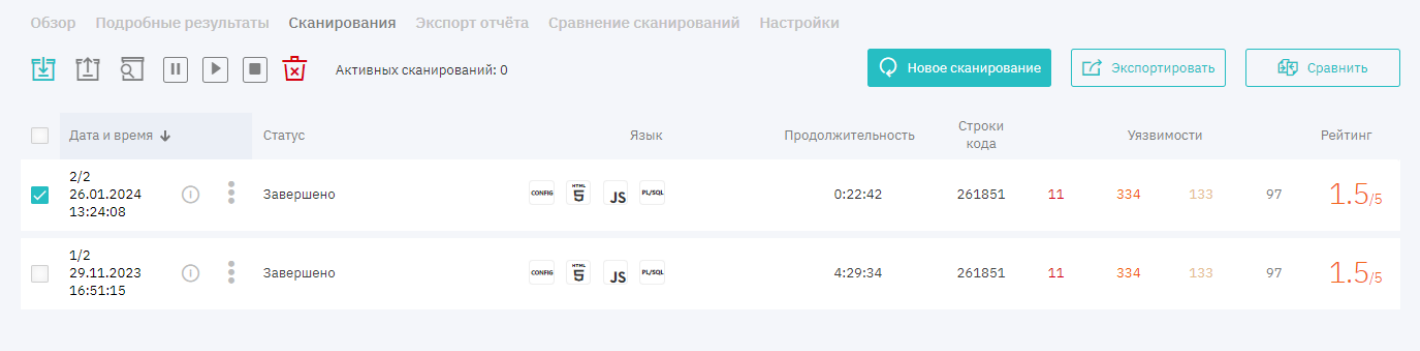


Рис. 2.29: Очередь сканирований

2.3.5 Сравнение сканирований

Раздел **Сравнение сканирований** предназначен для анализа изменений между двумя сканированиями одного проекта (рис. 2.30).

Чтобы сравнить результаты:

- выберите в тип анализа (SAST или CQ, результаты сканирований SAST и CQ нельзя сравнивать между собой);
- выберите 2 завершенных сканирования.

После выбора DerScanner отобразит статистику по уязвимостям, включая новые, устранённые и оставшиеся без изменений. Информация представляется в виде графика и сводной таблицы. Ниже размещается сравнительная таблица, где представлены ключевые параметры каждого сканирования: дата запуска, продолжительность, используемые языки программирования, объём кода, распределение уязвимостей по уровням критичности и итоговый рейтинг проекта.

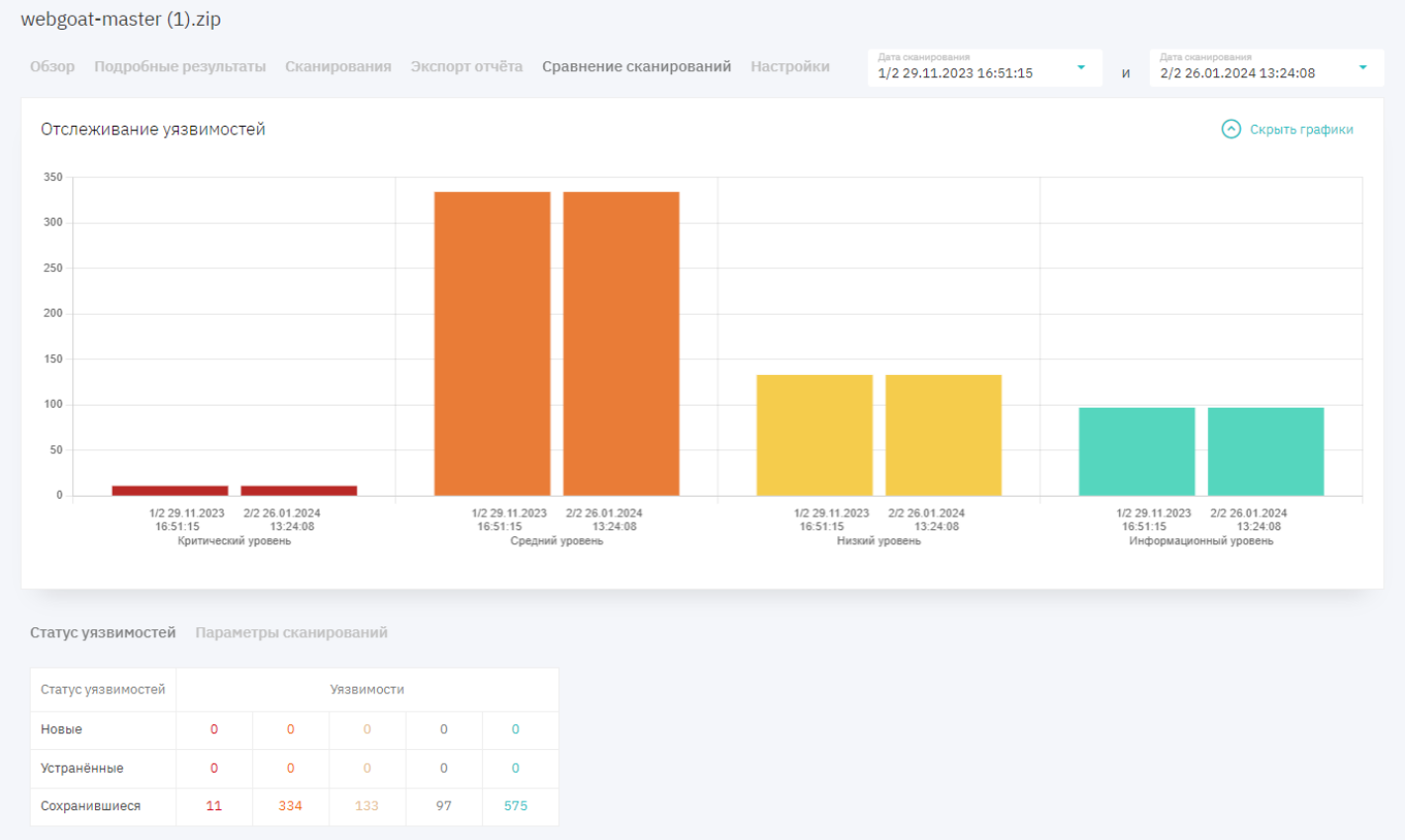


Рис. 2.30: Сравнение сканирований

2.3.6 Настройки

Раздел **Настройки** включает в себя разделы **Общие**, **Права пользователей** (доступен только администратору системы, позволяет давать доступ к проекту другим пользователям DerScanner), **Таск-менеджер**, **Автоматическое сканирование** и **Управление проектом**.

2.3.6.1 Таск-менеджер

На вкладке **Таск-менеджер** можно привязывать проекты **Jira** к проекту DerScanner (подробнее см. раздел Как привязать проект в DerScanner к проекту в Jira).

Проекты привязываются в виде конфигураций. К одному проекту можно привязать более одной конфигурации.

2.3.6.2 Автоматическое сканирование

В проектах с длительным циклом разработки можно задать автоматический запуск анализа, срабатывающий по расписанию или при наступлении определённых событий.

Интеграция с системами контроля версий позволяет инициировать сканирование напрямую из VCS-хостинга. DerScanner поддерживает работу с **GitHub**, **GitLab** и **Bitbucket**, обеспечивая запуск сканирования при событиях **Push**, **Tag**. Интеграция происходит с помощью механизма **webhook** (для корректной работы администратору необходимо предварительно настроить доступность API извне). Чтобы создать интеграцию, нужно настроить её в системе

DerScanner и в самом хостинге. Информацию о **webhook** и их настройке можно найти в документации соответствующих хостингов: GitHub, GitLab и Bitbucket.

DerScanner позволяет запускать автоматическе сканирования по расписанию. Для этого необходимо:

- указать ссылку на репозиторий с кодом, кторый необходимо анализировать;
- указать данные для авторизации, если указана ссылка на приватный;
- указать расписание запуска сканирований в формате Quartz-crone.

2.3.6.3 Управление проектом

В разделе **Управление проектом** доступны базовые действия с проектом: изменить его название, загрузить логотип, переместить проект в архив или вернуть его обратно, полностью удалить проект.

Чтобы временно скрыть проект из общего списка, воспользуйтесь кнопкой **Архивировать проект**.

Для окончательного удаления проекта, без возможности восстановления, нажмите **Удалить проект** и подтвердите своё решение.

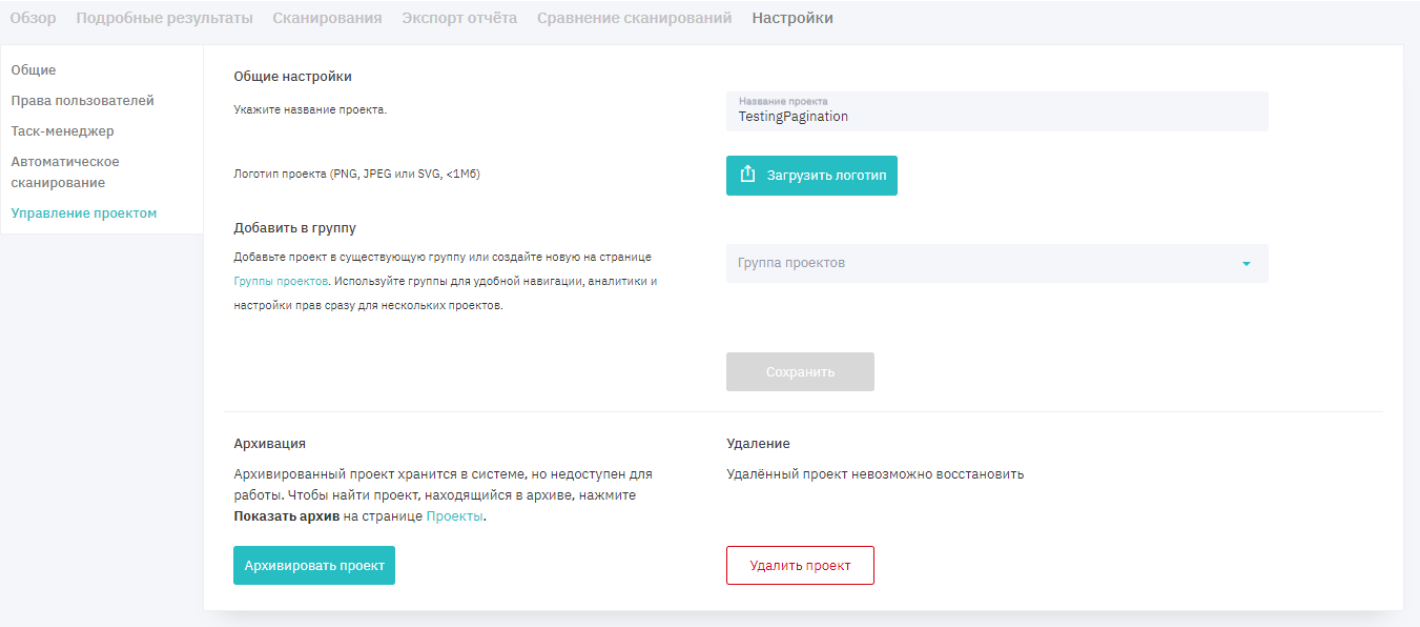


Рис. 2.31: Управление проектом

2.3.7 Правила и наборы

В разделе бокового меню **Правила и наборы** в подразделе **Правила** содержится список всех загруженных правил в систему.

Правила обединены в наборы. Наборы могут быть системные или созданы пользователем, если требуется выполнить сканирование проекта по определенным правилам.

2.3.7.1 Правила

Для поиска, сортировки и группировки правил используйте строку поиска и фильтры, доступные в правой боковой панели.

В строке правила отображаются «метки», которые позволяют отследить главную информацию о правиле не переходя в него. Метки отображают:

- Автора правил (пользовательские правила будут иметь свою метку).
- Классификации, в которые входит уязвимость: Уязвимости сопоставляются с пунктами классификаций: CWE, CWE/SANS Top 25 2011, CWE/SANS Top 25 2021, CWE/SANS Top 25 2023, OWASP Top 10 2021, OWASP Top 10 2017, OWASP Mobile Top 10 2016, OWASP Mobile Top 10 2014, OWASP Top 10 2013, OWASP ASVS, OWASP MASVS, PCI DSS, HIPAA. Каждой классификации соотвествует своя метка.

При нажатии на название правила открывается страница с подробной информацией об этом правиле, которая включает:

- Описание уязвимости — текстовое описание уязвимости;
- Пример — пример с кодом для уязвимости;
- Рекомендации — текстовое описание для исправлния уязвимости;
- Ссылки — ссылки на информационные ресурсы, связанные с уязвимостью;
- Классификации — стаблица со ссылками на соответствующие пункты в CWE 4.0, CWE/SANS Top 25 2011, CWE/SANS Top 25 2021, CWE/SANS Top 25 2023, OWASP Top 10 2021, OWASP Top 10 2017, OWASP Mobile Top 10 2016, OWASP Mobile Top 10 2014, OWASP Top 10 2013, OWASP ASVS, OWASP MASVS, PCI DSS, HIPAA;
- Наборы правил — список наборов правил, в которые водит правило;
- Паттерны — для добавления новых и редактирования существующих паттернов (только для пользовательских правил).

Системные правила недоступны для редактирования.

2.3.7.2 Создание правила

Чтобы создать правило нажмите **Создать правило** в подразделе **Правила**.

Для создания правила с использованием описания информации из существующего правила, нажмите кнопку **Создать копию правила** в карточке правила. Паттерны в таком случае скопированы не будут.

Паттерны описывают условия, при которых определённый участок кода считается уязвимым. Для их хранения используется универсальный формат XML, разработанный специально для этой цели. Чтобы добавить новый паттерн: 1. На вкладке Паттерны у пользовательского правиоа, нажмите **Создать паттерн**. 2. Укажите название, публичность, уровень критичности, уровень confidence и вставьте XML-описание паттерна. 3. Нажмите **Сохранить**.

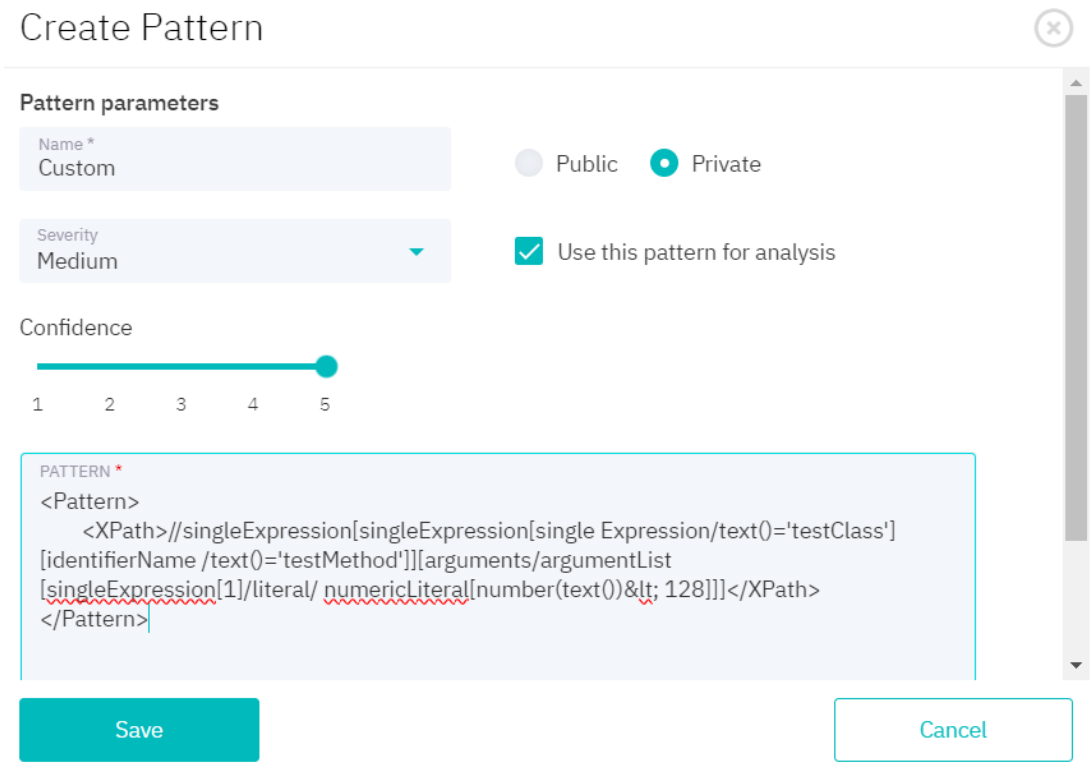


Рис. 2.32: Создание паттерна

2.3.7.2.1 Инструкция по записи паттернов

В DerScanner реализован анализ потока данных, позволяющий отслеживать, как данные перемещаются и трансформируются внутри приложения.

Для приложений на Java, Scala и Kotlin в качестве промежуточного представления используется Java байт-код. В случае других языков применяется абстрактное синтаксическое дерево (AST).

Паттерны анализа формулируются в виде XPath-запросов к соответствующему представлению. Механизм анализа основан на назначении специального флага переменным и отслеживании его распространения по коду. В зависимости от роли в потоке данных, паттерны делятся на три типа: **источники (source)**, **передачи (passthrough)** и **приёмники (sink)**.

- **Source**-паттерны указывают, какие вызовы методов инициируют постановку флагов (то есть с каких точек данные считаются потенциально опасными).
- **Passthrough**-паттерны описывают, как флаг должен передаваться или изменяться при прохождении через методы — например, при очистке данных флаг может быть снят.
- **Sink**-паттерны определяют, какие конструкции являются потенциально небезопасными при наличии флагов и в каких условиях.

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

- 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 принимающей сущности) 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.

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.

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**.

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>
```

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)).

2.3.7.2.1.1 Флаги паттернов

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**. 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**.

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**.

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».

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>
```

2.3.7.2.2 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.

2.3.7.2.3 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.

2.3.7.2.4 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).

2.3.7.2.5 Примеры паттернов

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>
```

2.3.7.2.5.1 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]]]
```

2.3.7.2.5.2 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']]
```

2.3.7.2.5.3 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')]]
```

2.3.7.2.5.4 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]]
```

2.3.7.2.5.5 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')]]]
```

2.3.7.2.5.6 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])&lt;16]]
```

2.3.7.2.5.7 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])]]
```

2.3.7.2.5.8 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]]
```

2.3.7.2.5.9 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]
```

2.3.7.2.5.10 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')]]
```

2.3.7.2.5.11 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']]
```

2.3.7.2.5.12 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>
```

2.3.7.2.5.13 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')]
```

2.3.7.2.5.14 Запись паттерна для языка 1C

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]]
```

2.3.7.2.5.15 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')]]
```

2.3.7.2.5.16 Запись паттерна для языка Go

Вызов функции **testFunc**:

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

2.3.7.2.5.17 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]]]]
```

2.3.7.2.5.18 Запись паттерна для языка Groovy

Вызов метода **testMethod**:

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

2.3.7.2.5.19 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']]
```

2.3.7.2.5.20 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])]]
```

2.3.7.2.5.21 Запись паттерна для языка Pascal

Вызов функции **test_func**:

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

2.3.7.2.5.22 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]]
```

2.3.7.2.5.23 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]
```

2.3.7.2.5.24 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]]
```

2.3.7.2.5.25 Запись паттерна для языка Solidity

Вызов функции **testfunc**:

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

2.3.7.2.5.26 Запись паттерна для языка Vyper

Вызов функции **insecure_function()**:

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

2.3.7.2.5.27 Запись паттерна для языка COBOL

Заданная в исходном коде переменная **sensitiveData**:

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

2.4 API

2.4.1 Запуск сканирования из API

Чтобы запустить сканирование **SAST** из API:

1. Кликните **/scan/start** в разделе **Scan**.
2. Нажмите **Try it out**.

3. В теле запроса выберите файл для анализа или укажите ссылку на проект. Если вы загружаете файл, важно либо запустить анализ из директории с файлом, либо прописать путь к файлу.
4. Укажите UUID проекта, в котором будет запущено новое сканирование.
5. Укажите дополнительные настройки.
6. Нажмите **Execute**.

2.4.2 Экспорт отчёта из API

Чтобы выгрузить отчёт сканирования проекта **SAST** из 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**.

2.5 Инструмент командной строки

2.5.1 Запуск сканирования из командной строки

Запуск сканирования из командной строки доступен только при условии предустановленной Java 11 или новее.

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

Для запуска сканирования из командной строки следует выполнить команду:

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

где:

- `-rest` адрес API (URL).
- `-token` токен авторизации API (можно получить в интерфейсе пользователя в разделе **Аккаунт**).

Аргументы (options):

Обязательные аргументы:

- `-type` возможные значения: FILE, LINK.
- `-path` используется, если type=FILE (путь к каталогу или файлу для анализа).
- `-link` используется, если type=LINK (URL репозитория). Если указана ссылка на приватный репозиторий, укажите данные для авторизации в блоке **Настройки приватного репозитория** .

Необязательные аргументы:

- `-preprocessing` запускать сканирования с предобработкой или без. По умолчанию включено.
- `-disableSsl`.
- `-dertriage` выполнить триаж уязвимостей после завершения основного сканирования.
- `-dercodefix` сгенерировать исправления для уязвимостей после завершения основного сканирования.
- `-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.
- `-analysisOptionsJava` настройки опции анализа для Java (без сборки, анализ собранного проекта с `.class`-файлами, со сборкой DerScanner, с собственной сборкой): `SOURCE_NO_BUILD, PREBUILD, SOURCE_WITH_BUILD, SOURCE_OWN_BUILD`.
- `-dastProjectId` ID проекта динамического анализа для корреляции результатов.
- `-osEnvironment` выбор среды сборки проектов для C/C++: `MAC - MacOS, LINUX - Linux, WIN - Windows`.
- `-intermodular` использовать межмодульный анализ для C/C++ проектов.
- `-javaCustomBuildCommand` параметры сборки собственными инструментами Java.
- `-priority` приоритет выполнения сканирования.
- `-sendReportAutoscan` отправить отчёт на почту для автоматических сканирований.
- `-sendReportManual` отправить отчёт на почту для сканирований, запущенных вручную.
- `-sendReportRecipients` отправить отчёт на почту получателям. Если получателей несколько, указать e-mail адреса через запятую.
- `-sendReportTemplate` использовать шаблон отчёта при отправке на почту. Укажите UUID шаблона.
- `-sendReportToAdmins` отправить отчёт на почту администраторам.
- `-vcs.branch` ветка репозитория, если не master (при анализе приложений по ссылке на репозиторий).
- `-vcs.account` UUID репозитория.
- `-vcs.login` логин репозитория.
- `-vcs.password` пароль репозитория.
- `-vcs.token` токен аутентификации репозитория.
- `-vcs.tokenId` UUID токена аутентификации.
- `-vcs.sshKey` путь до ключа SSH репозитория.
- `-vcs.sshKeyId` UUID ключа SSH.
- `-vcs.sshDefault` использовать стандартную конфигурацию SSH-клиента. Если значение false, то указать конфигурацию SSH-клиента.
- `-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.
- `-useUserPatterns` использовать пользовательские правила.
- `-analyzelibs` анализировать ли библиотеки или вложенные архивы.
- `-isVisualStudio` проект Visual studio.

- `-saveFile` полностью сохранить загружаемый файл.
- `-incremental` использовать инкрементальный анализ.
- `-uuid` UUID проекта, в котором будет запущено сканирование. UID 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 sastStart -type LINK -link
https://github.com/your_repo
```

Для получения статуса сканирования:

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

- `-rest` адрес API (URL).
- `-token` токен авторизации API; можно получить в интерфейсе пользователя в разделе `Аккаунт` (см. раздел Аккаунт).

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 sastStatus -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] sastScore [options]
```

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

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 sastScore -scanid b001eab1-ba6c-4b05-9066-e84d594210e2
```

2.5.2 Экспорт отчёта из инструмента командной строки

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

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

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

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

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

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.
- `-dercodefix` включить в отчёт информацию о сгенерированных исправлениях.
- `-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.low` включить в отчёт уязвимости низкого уровня.
- `-filter.medium` включить в отчёт уязвимости среднего уровня (включается по умолчанию).
- `-filter.tasks` включить в отчёт уязвимости с созданными задачами в таск-менеджере (включается по умолчанию).
- `-confidence.included` включить в отчёт данные ConfiAI. При включении укажите:
  - `-confidence.critical` настройка значения ConfiAI для критических уязвимостей (по умолчанию 0).
  - `-confidence.info` настройка значения ConfiAI для уязвимостей информационного уровня (по умолчанию 0).
  - `-confidence.low` настройка значения ConfiAI для уязвимостей низкого уровня (по умолчанию 0).
  - `-confidence.medium` настройка значения ConfiAI для уязвимостей среднего уровня (по умолчанию 0).
- `-general.contents` включить оглавление (включается по умолчанию).
- `-general.correlationTags` отображать метки корреляции результатов (включается по умолчанию).
- `-general.format` формат отчёта. Принимает значения: PDF, CSV, SARIF, HTML. По умолчанию принимает значение PDF.
- `-general.included` включить настройки экспорта (включается по умолчанию).
- `-general.locale` настроить язык отчёта (по умолчанию английский).

- `-general.logo` использовать пользовательский логотип. По умолчанию не используется. Если пользователь хочет использовать собственный логотип, необходимо указать путь до логотипа.
- `-general.statuses` отображать статусы уязвимостей (по умолчанию выключено).
- `-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.action` включить действия с уязвимостями (включается по умолчанию).
  - `-results.comment` включить комментарии к уязвимостям (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.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.statuses` включить уязвимости со статусами (указать статусы). По умолчанию все уязвимости, кроме отклоненных.
  - `-results.taskInfo` включить информацию о задачах в таск-менеджере (включается по умолчанию).
  - `-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.fileStat` включить статистику по проанализированным файлам (по умолчанию выключено).
  - `-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, ASVS_L1, ASVS_L2, ASVS_L3, MASVS_L1, MASVS_L2, MASVS_L1_R, MASVS_L2_R, PCI_DSS, HIPAA, CWE_SANS_11, CWE_SANS_21, CWE_SANS_23. Значение по умолчанию: CR.
- `-table.included` включить список уязвимостей (включается по умолчанию). При включении в отчёт укажите:
  - `-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.statuses` включить уязвимости со статусами (указать статусы). По умолчанию все уязвимости, кроме отклоненных.
- `-comparison.included` включать ли в отчёт сравнение с предшествующим сканированием. При включении в отчёт укажите:
  - `-comparison.ScanId` ID 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 sastExport -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.

2.6 Запуск сканирования из инструментов сборки

Сканирование исходного кода для проектов на языках Java, Scala и Kotlin можно инициировать прямо из инструментов сборки Maven, Gradle и SBT.

Запуск выполняется с помощью инструмента командной строки **clt.jar**. Во время запуска необходимо передать следующие параметры в настройках:

- Токен авторизации (`-token`) — генерируется в личном кабинете пользователя в разделе

«Аккаунт».
- Адрес API сервера (`-rest`) — указывается URL адреса API.
- Путь к файлу **clt.jar**, используемый для запуска сканирования.

2.7 ИИ инструменты

DerScanner использует функции искусственного интеллекта, чтобы ускорить процессы обеспечения безопасности в конвейере разработки. В рамках нашей приверженности принципам управления данными и прозрачности, эти функции доступны локально — без необходимости подключения к Интернету. Их также можно навсегда отключить.

2.7.1 DerTriage

DerTriage — это интеллектуальный инструмент DerScanner для автоматической проверки результатов SAST-анализа. Технология SAST неизбежно порождает некоторое количество ложных срабатываний. DerTriage анализирует необработанные данные, полученные после SAST-анализа, и учитывает более широкий контекст, чтобы определить, является ли конкретное обнаружение действительным.

DerTriage можно запускать как на этапе сканирования в настройках сканирования, так и после его завершения — в разделе **Подробные результаты**.

Запуск до начала сканирования позволяет вносить изменения массово, но отрицательно сказывается на общем времени анализа. Чтобы активировать DerTriage, выберите уровни критичности уязвимостей для оценки в настройках сканирования. Кроме того, при выборе опции **Применить результат триажа** уязвимостям в **Подробных результатах** будут автоматически назначены статусы: **Подтверждено** для достоверных срабатываний и **Отклонено** для ложных. Обоснование будет доступно в информации об уязвимости.

Для точечной проверки DerTriage можно запускать вручную в **Подробных результатах**. Перейдите к интересующей уязвимости и нажмите на три точки справа. Выбрав DerTriage, вы получите рекомендацию о достоверности обнаружения и обоснование этого вывода.

2.7.2 DerCodeFix

DerCodeFix направлен на устранение уязвимостей в коде. Вместо обобщённой рекомендации он вносит конкретные изменения в уязвимый код для устранения потенциальной угрозы.

Как и DerTriage, DerCodeFix можно запускать как на этапе сканирования, так и при анализе результатов.

Чтобы активировать DerCodeFix во время сканирования, выберите нужные уровни критичности уязвимостей в настройках сканирования. Кроме того, при выборе опции **Для подтвержденных триажем** будут сгенерированы исправления только для подтверждённых DerTriage уязвимостей. Исправленный фрагмент кода будет доступен в разделе **Подробные результаты** > ваша уязвимость > вкладка **DerCodeFix**.

Для точечной генерации исправления перейдите к уязвимости в **Подробных результатах** и нажмите **Исправить с DerCodeFix**. Нажатие на строку кода предоставит краткое объяснение изменений, а сами изменения будут выделены фиолетовым ниже. Чтобы просмотреть исходный код, используйте переключатель.

03 Динамический анализ

3.1 Особенности запуска анализа

DerScanner позволяет сканированиять работающие веб-приложения методом динамического анализа. На вход случайные или заведомо неверные данные и анализирует реакцию приложения на них.

Перед запуском динамического анализа необходимо разобраться в архитектуре тестируемого приложения и используемых в нём механизмах авторизации.

Подготовка к анализу

Перед запуском сканирования необходимо обеспечить доступ анализатора DerScanner к тестируемому приложению через сеть. Если при старте анализа появляется сообщение об ошибке соединения, проверьте сетевую доступность приложения с сервера, где запущен DerScanner, по указанному порту (по умолчанию — 80 для HTTP и 443 для HTTPS).

После настройки сетевого соединения важно проверить, не мешают ли сканированию средства веб-защиты (WAF), установленные между DerScanner и целевым приложением. Убедитесь, что анализатору разрешён весь необходимый трафик — для этого либо отключите WAF на время проверки, либо добавьте все используемые типы запросов в список исключений (белый список).

Если в приложении есть авторизация, чтобы анализатор мог получить к нему доступ, его необходимо авторизовать. В DerScanner предусмотрено 5 основных способов авторизации:

- **Авторизация по логину/паролю**: базовый метод авторизации, при котором анализатор попытается подставить логин и пароль в соответствующие поля, обнаруженные на сервере;

- **Авторизация по токену**: в случае, если в приложении используется Bearer-аутентификация, в заголовки запросов вставляется `bearer: TOKEN`. Для выполнения авторизации по токену:

  - На сканируемом веб-ресурсе:

    1. Перейдите по URL, который вы ввели в поле **URL для атаки**, и перейдите на страницу авторизации.
    2. Перейдите в консоль разработчика в браузере (в Chromium браузерах нажмите **F12**) и введите логин и пароль, который будет использоваться в динамическом анализе (например, **user/password**).
    3. В консоли разработчика перейдите на вкладку **Network**.
    4. В окне **Name** выберите строку с запросом на логин (*token* или *login* в зависимости от браузера).
    5. В правом окне перейдите на вкладку **Preview** и скопируйте значение, содержащее токен авторизации (в Chromium *access_token*).

- В DerScanner:

  6. В настройках сканирования, раздел **Авторизация** выберите опцию **Токен аутентификации**.
  7. Вставьте скопированное значение в поле **Токен**.
  8. В разделе **Ajax Spider** выберите чекбокс **Использовать ajax spider**.
  9. Запустите сканирование (Для получения корректных результатов сканирования токен авторизации должен быть активным долгое время. В ином случае существует риск получения некорректных или неполных результатов анализа).

- **Авторизация по заголовкам**:наиболее универсальный метод анализа. При выборе этого метода динамический анализатор добавляет указанные в форме данные в заголовки всех запросов. Это позволяет пройти авторизацию в любом приложении, где аутентификационные данные передаются через заголовки (токены, session ID, user ID и т.д.). Метод также подходит для приложений со сложными механизмами авторизации на нескольких ресурсах. Для этого необходимо вручную войти в приложение, скопировать все нужные заголовки для авторизации и вставить их в DerScanner для сканирования. Важно помнить, что заголовки остаются неизменными в процессе сканирования, поэтому при истечении срока действия токена авторизация может быть прервана. Рекомендуется увеличить срок действия всех необходимых токенов на уровне приложения для стабильной работы при использовании этого метода. Для авторизации по заголовкам:

  - На сканируемом веб-ресурсе:

    1. Перейдите по URL, который вы ввели в поле **URL для атаки**, и авторизуйтесь.
    2. Выполните любой запрос, для которого требуется авторизация (например, запрос cookies).
    3. Перейдите в консоль разработчика в браузере (в Chromium браузерах нажмите **F12**) и введите логин и пароль, который будет использоваться в динамическом анализе (например, **user/password**).
    4. Перейдите на вкладку **Network**.
    5. Откройте запрос, выполненный в пункте **2**.
    6. Скопируйте требуемые заголовки из раздела **Request Headers**.

  - В DerScanner:

    7. В настройках сканирования, раздел **Авторизация** выберите опцию **Заголовки**.
    8. Вставьте скопированные значения в поле **Заголовки**.
    9. Запустите сканирование.

- **Авторизация с заполнением формы**: позволяет авторизоваться в любом веб-приложении, у которого есть форма авторизации, в которой необходимо заполнить только логин и пароль. Не поддерживает формы, состоящие из нескольких форм, а также многофакторную авторизацию. Для авторизации через форму:

  - На сканируемом веб-ресурсе:

    1. Перейдите по URL, который вы ввели в поле **URL для атаки**, и перейдите на страницу

авторизации.

2. Перейдите в консоль разработчика в браузере (в Chromium браузерах нажмите **F12**) и введите логин и пароль, который будет использоваться в динамическом анализе (например, **user/password**).
3. В консоли разработчика перейдите на вкладку **Network**.
4. В окне **Name** выберите строку с запросом на логин (*token* или *login* в зависимости от браузера).
5. В правом окне перейдите на вкладку **Payload** и скопируйте значения, содержащие логин, пароль (в Chromium *username/password*) и возможные дополнительные поля.
6. В интерфейсе сканируемого приложения необходимо получить сообщение о выходе. Для этого нажмите правой кнопкой мыши на кнопку **Log Out** (или ее аналог) в интерфейсе, после чего выберите **Inspect element** в контекстном меню.
7. В панели разработчика нажмите правой кнопкой мыши по выделенному участку HTML-кода и нажмите **Copy > Copy element**.
8. Если вы планируете авторизацию по сообщению о входе, скопируйте HTML-код элемента **Log In** вместо **Log Out** или его аналога.

- В DerScanner:

9. В настройках сканирования, раздел **Авторизация** выберите опцию **Форма авторизации**.
10. В поле **POST запрос авторизации** вставьте значения из пункта **5** в формате `ключ={значение}`. Ключи между собой соединяются через `&`. Например: `user={uservalue}&password={passwordvalue}&test={false}`.
11. В поля **Название поля ввода логина/пароля** вставьте названия ключей из пункта **10**. Т.е. если `user={uservalue}&password={passwordvalue}&test={false}`, то в поле ввода логина должно быть `user`, а в поле ввода пароля `password`.
12. В поля **URL страницы входа/выхода** вставьте скопированные значения URL страниц входа и выхода.
13. В поля **Регулярное сообщение о входе/выходе** вставьте значение из пункта **7** или **8** соответственно. Для запуска сканирования требуется заполнить одно из этих полей.
14. Запустите сканирование.

- **NTLM-авторизация**: стоит выбирать эту опцию, если авторизация в приложении реализована через NTLM. Позволяет пройти авторизацию как на одном сервере вместе с приложением, так и на стороннем сервере авторизации. Для авторизации через NTLM:

1. В настройках сканирования в DerScanner, раздел **Авторизация** выберите опцию **NTLM**.
2. Заполните поля **Хост**, **Порт** и **Атрибут области** (**realm**) в случае, если сервис авторизации в анализируемом приложении находится на отдельном от самого приложения хосте. Для большинства случаев эти поля необязательные.

3. В поле **Имя пользователя** укажите логин учётной записи для авторизации.
4. В поле **Имя поля для ввода имени пользователя** укажите значение, если отличается от username.
5. В поле **Пароль** укажите пароль учётной записи для авторизации.
6. В поле **Имя поля для ввода пароля** укажите значение, если отличается от password.
7. Запустите сканирование.

Рекомендуется приложить к сканированию схему API приложения в формате OpenAPI.json. Это позволит анализатору использовать запросы, специфичные для конкретного приложения,что значительно улучшит глубину анализа, а соответственно полученные результаты.

3.2 Настройки сканирования

Сканирование можно настроить через интерфейс с помощью соответствующих параметров, аналогичные настройки доступны также через API и инструмент командной строки (CLT).

- Выбор приоритет сканирования;
- Выбор агента сканирования, который будет выполнять анализ;
- Исключение URL из анализа;
- Выбор режима сканирования (режим может зависеть от целей анализа и ресурсов):
  - Стандартный: базовый режим;
  - Агрессивный: агрессивный режим, позволяющий помимо сканирования выполнять атаки на приложение;
  - Активная атака: режим активной атаки, в котором атаки начинаются сразу с начала сканирования;
- Настройки авторизации: выбор способа и заполнение данных для выполнения авторизации;
- Ajax-паук:
  - Включение режима ajax-паука (дополнительная опция для более глубокого анализа);
  - установка таймаута для ajax-паука;
- OpenAPI схема:
  - загрузка `.json`-файла с OpenAPI схемой для веб-приложения;
  - настройка замены параметров пути.

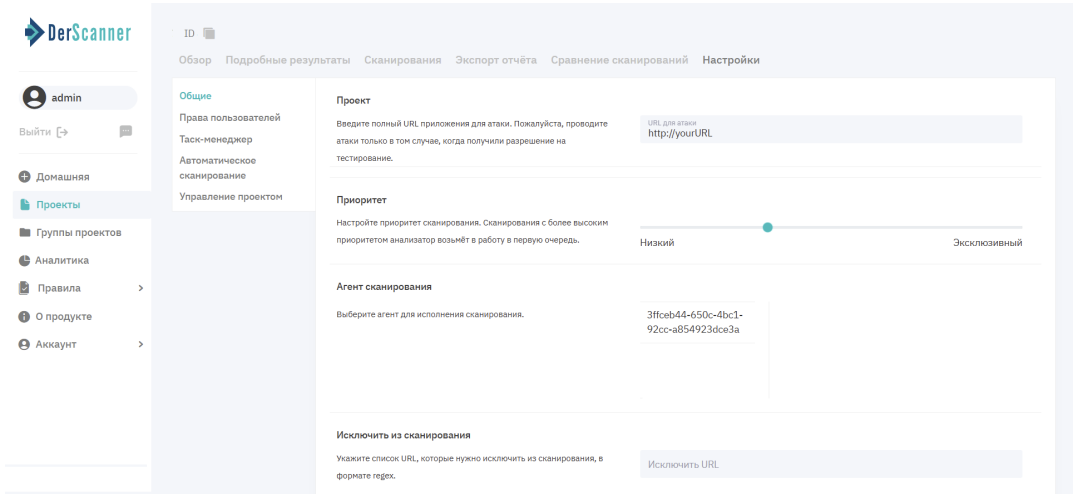


Рис. 3.1: Настройки анализа

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.

3.3 Пользовательский интерфейс

3.3.1 Создание проекта

В интерфейсе DerScanner реализованы следующие способы создания проекта динамического анализа:

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

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.

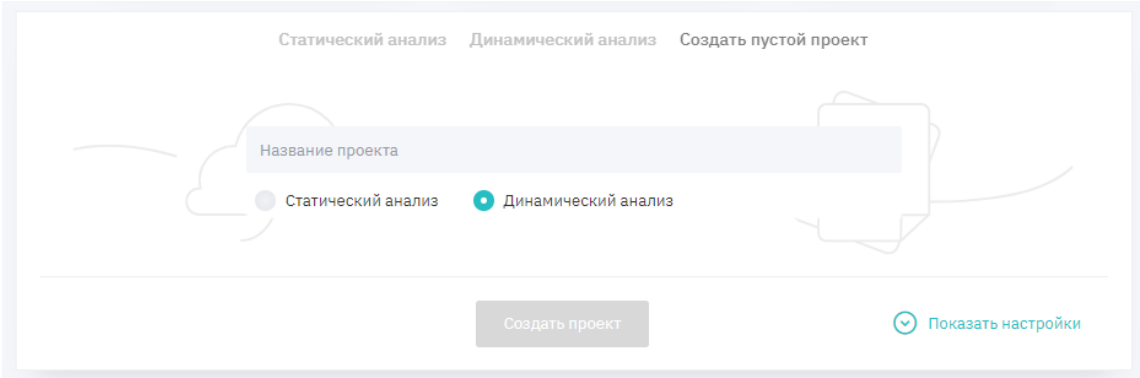


Рис. 3.2: Создание пустого проекта

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

3.3.2 Запуск сканирования в UI

Чтобы запустить новое сканирование в UI:

1. Перейдите на **Домашнюю страницу**.
2. Укажите URL ресурса для анализа.
3. Настройте анализ.
4. Нажмите **Начать сканирование**.

3.3.3 Работа с результатами

3.3.3.1 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.

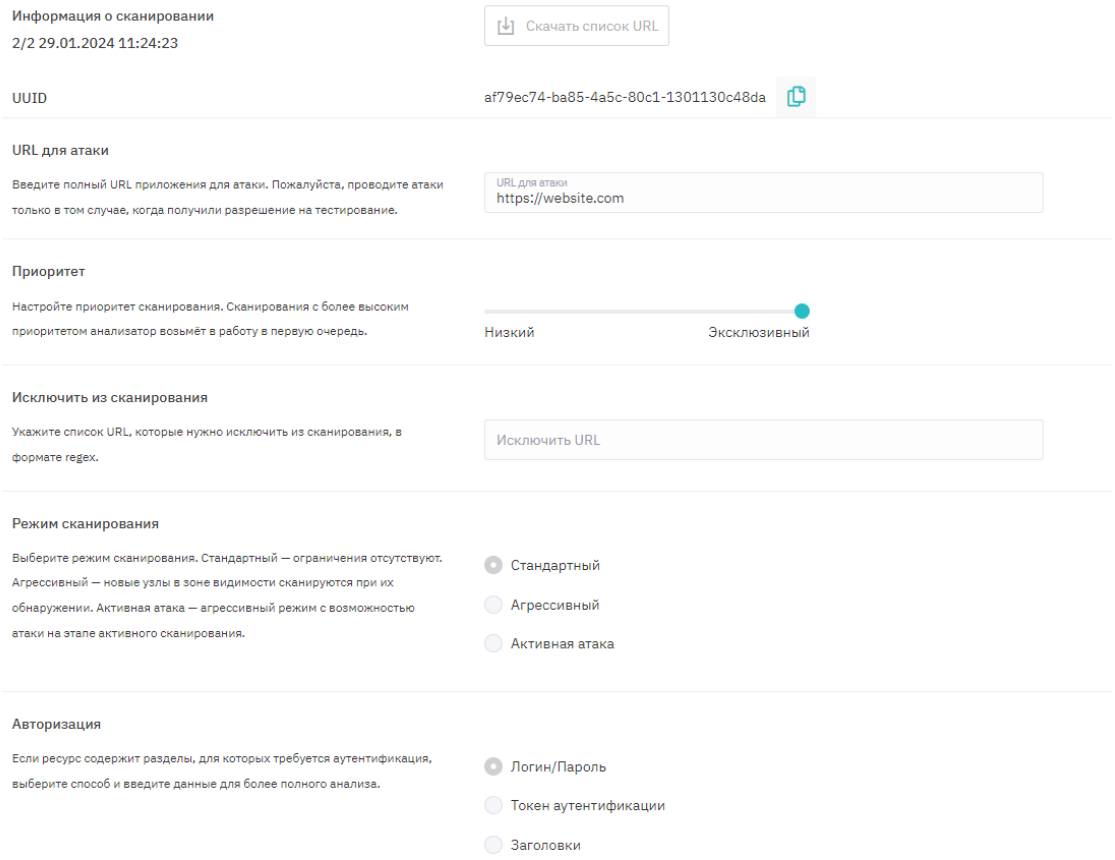


Рис. 3.3: Параметры запуска анализа

На странице **Обзор** представлена следующая информация:

- рейтинг;
- статус сканирования;
- продолжительность сканирования;

- количество уязвимостей каждого уровня критичности
- дашборды по проекту и сканированию.

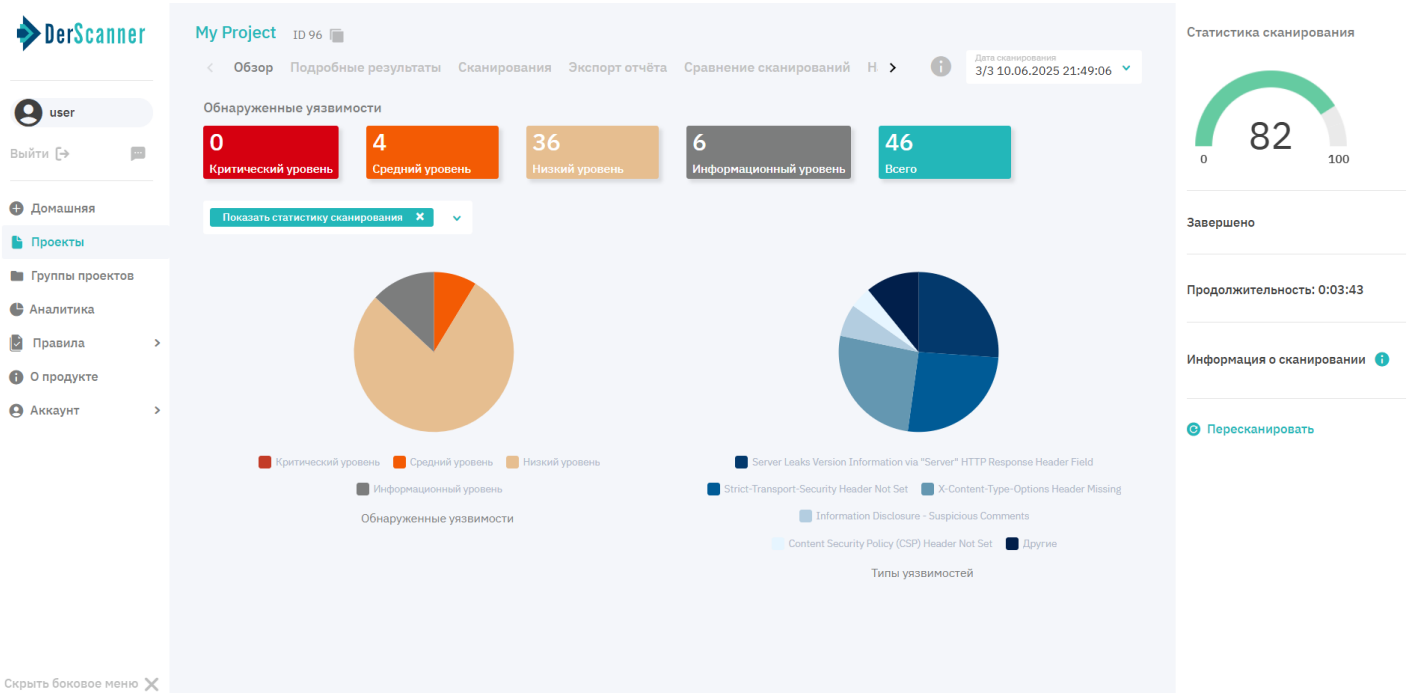


Рис. 3.4: Обзор

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

3.3.3.2 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.

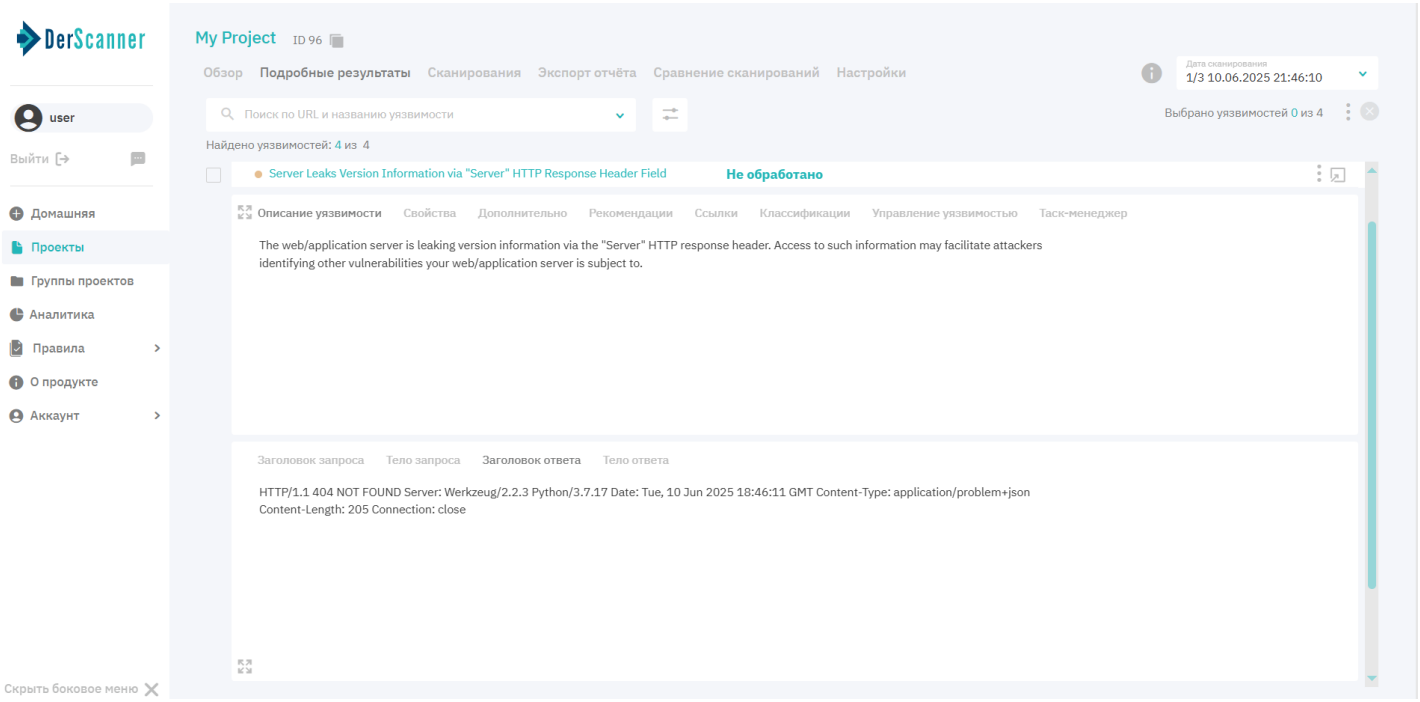


Рис. 3.5: Подробные результаты

На странице 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. 3.6).

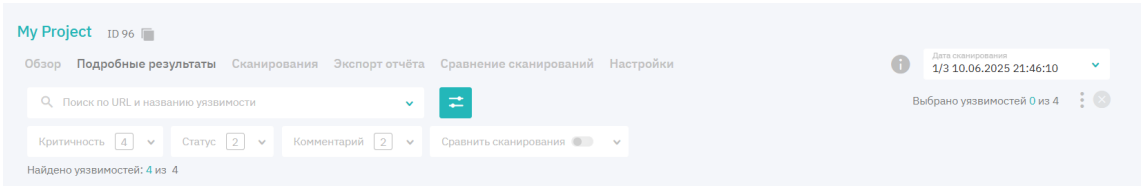


Рис. 3.6: Фильтры результатов

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 справа от названия 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 повторном сканировании изменения сохраняются.

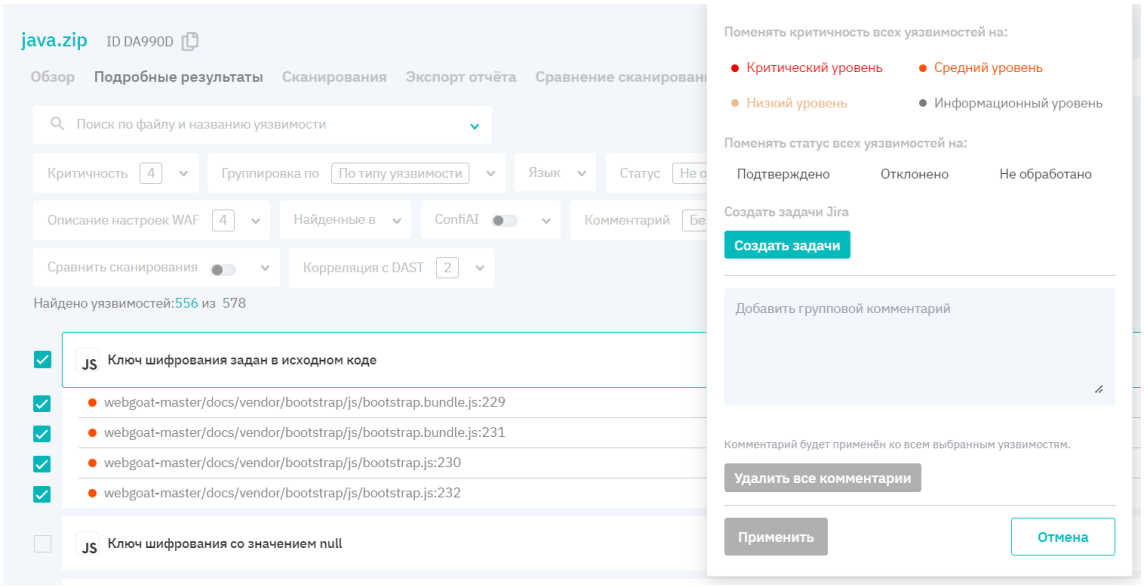


Рис. 3.7: Управление пакетом уязвимостей

После выбора конкретного вхождения уязвимости в правой части страницы отображается соответствующие ему заголовок и тело запроса и заголовок и тело ответа. Чтобы раскрыть информацию об уязвимости на полный экран, нажмите на иконку в правом верхнем углу. В левой части страницы представлена информация об уязвимости (рис. 3.8): **Описание**, **Свойства**, **Дополнительно**, **Рекомендации**, **Ссылки**, **Классификации**, **Управление уязвимостью** и **Таск-менеджер**.

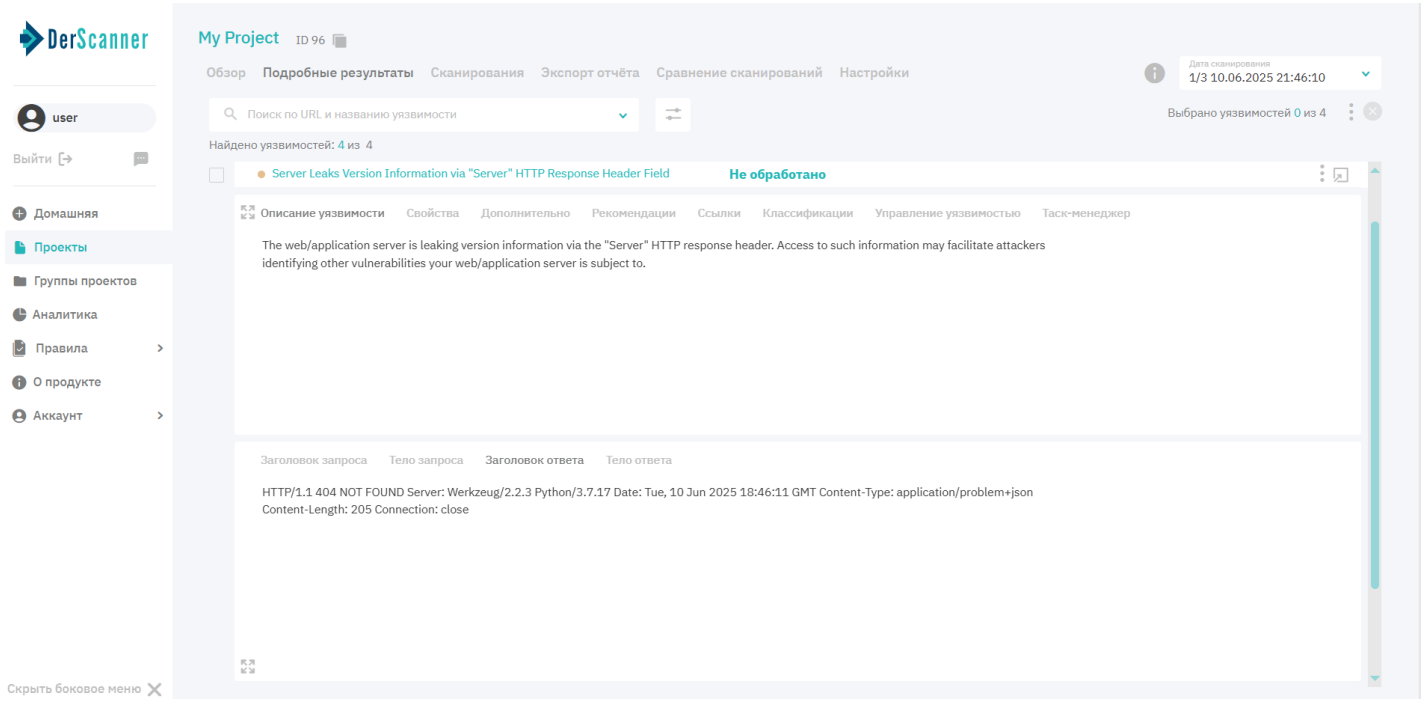


Рис. 3.8: Свойства уязвимости

На вкладке **Классификации** (рис. 3.9) отображаются соответствующие пункты CWE, WASC, OWASP Top 10 2017, OWASP Top 10 2021 и WSTG-v4.2.

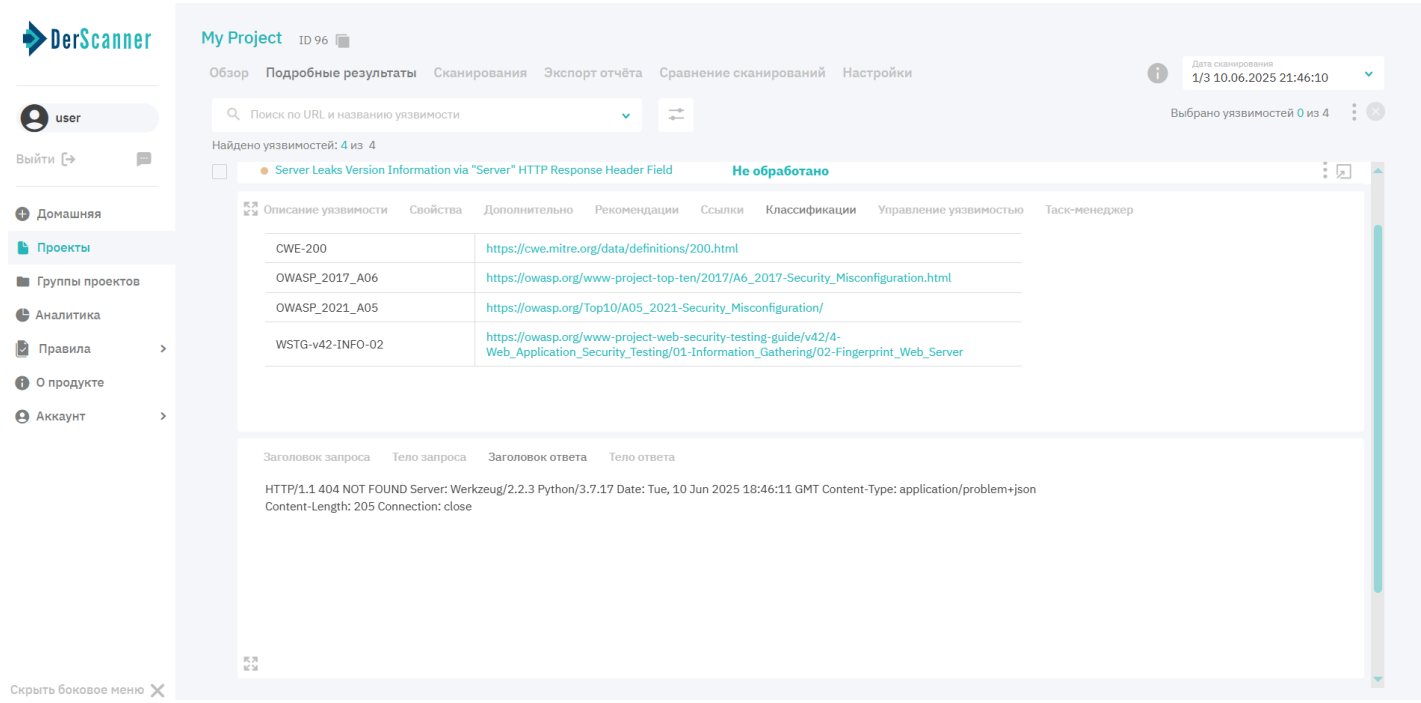


Рис. 3.9: Классификации

На вкладке **Управление уязвимостью** (рис. 3.10) можно изменить уровень критичности и статус, добавить комментарий к вхождению и посмотреть журнал событий с оставленными ранее комментариями и действиями.

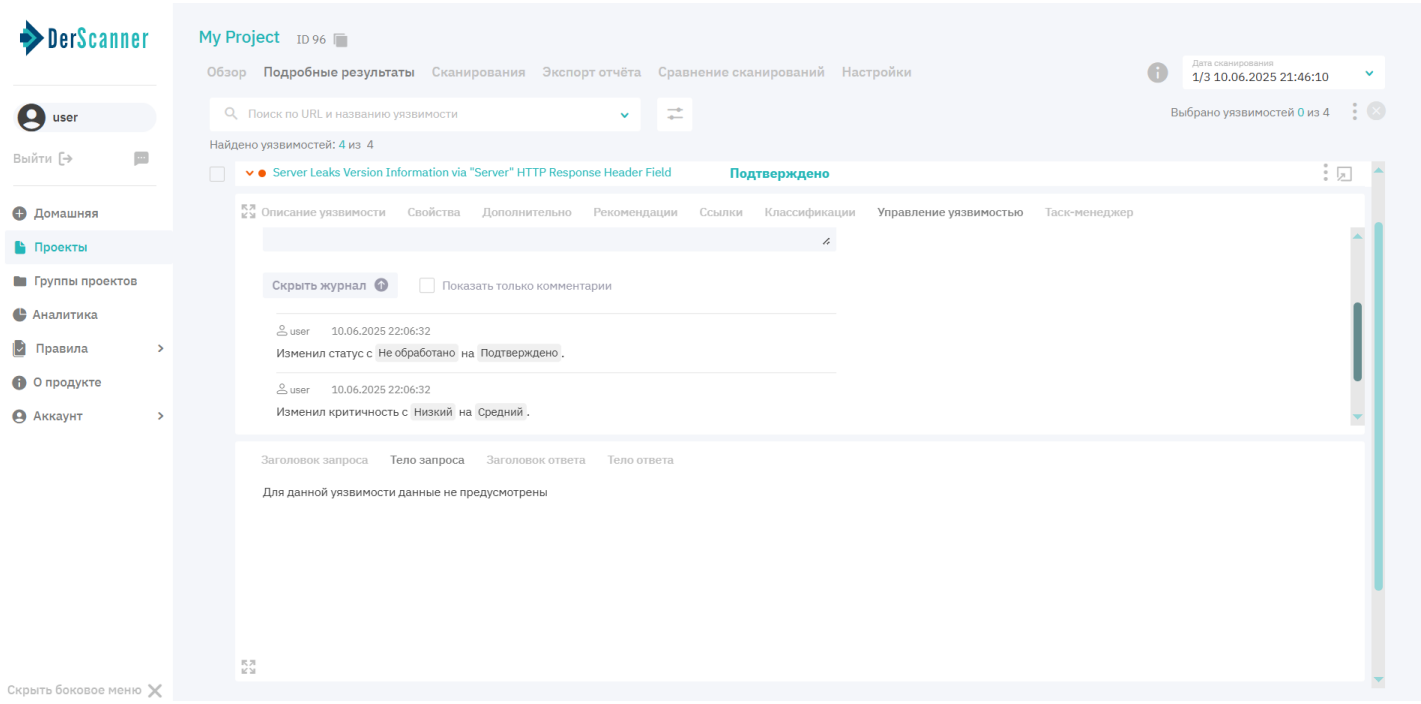


Рис. 3.10: Управление уязвимостью

3.3.3.3 Отчёт безопасности

В разделе **Экспорт отчёта** можно выгрузить результаты сканирования в отчёт в формате PDF, HTML, CSV или JSON. 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.

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

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.

3.3.3.3.1 Информация о проекте

В отчёт можно включить динамику уровня безопасности и историю сканирований.

3.3.3.3.2 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.

Выберите уязвимости уровню критичности и типу.

Выберите статусы уязвимостей и задайте количество их вхождений.

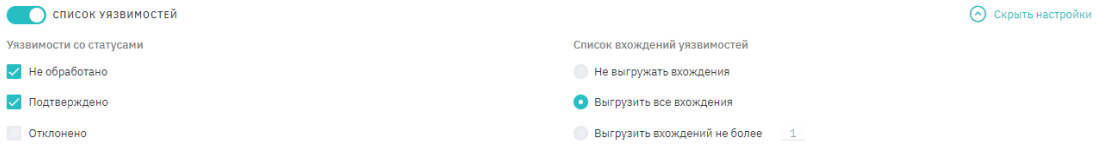


Рис. 3.11: Список уязвимостей

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;
- отображение журнала событий:
  - не включать;
  - включить только комментарии;
  - включить только действия;
  - журнал целиком;
- включение запроса/ответа.

3.3.3.3.3 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.

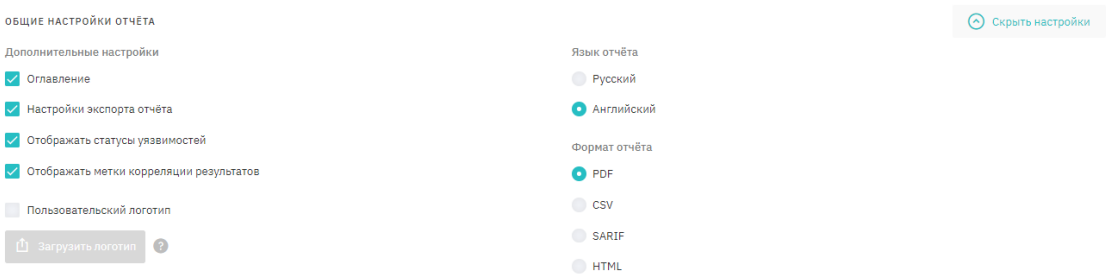


Рис. 3.12: Общие настройки отчёта

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.

3.3.4 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.

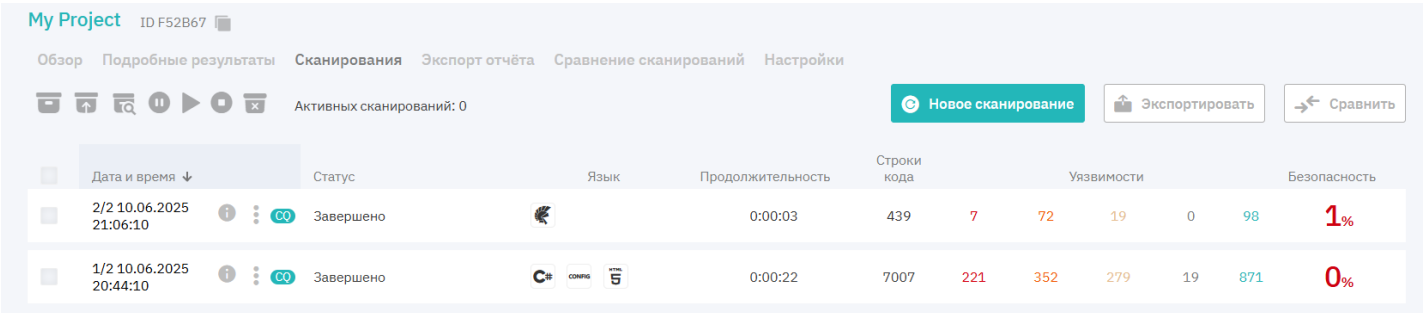


Рис. 3.13: Сканирования

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**.

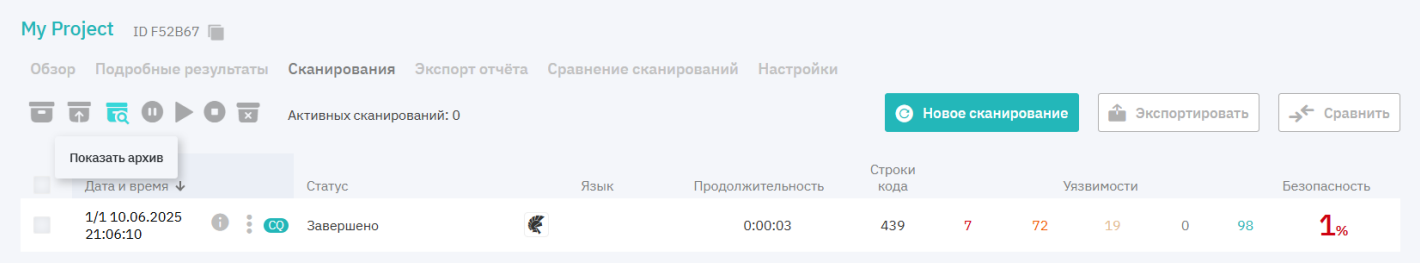


Рис. 3.14: Показать/Скрыть архив

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

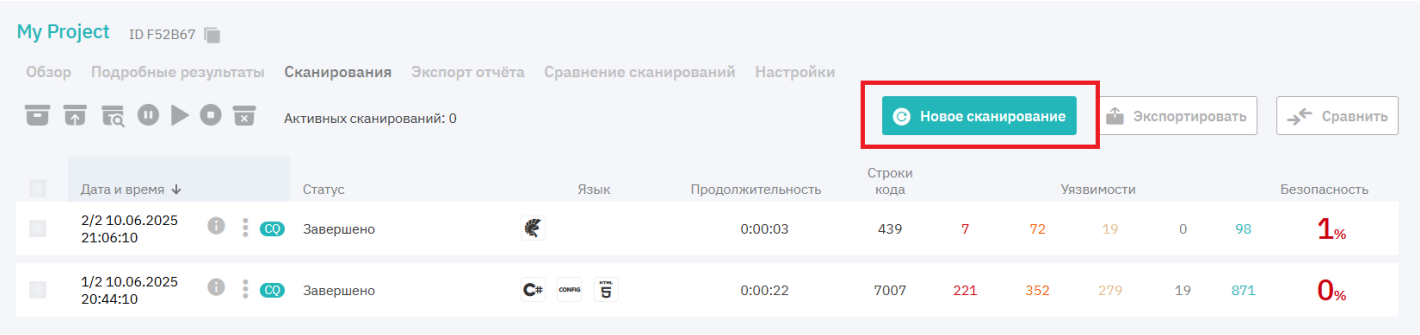
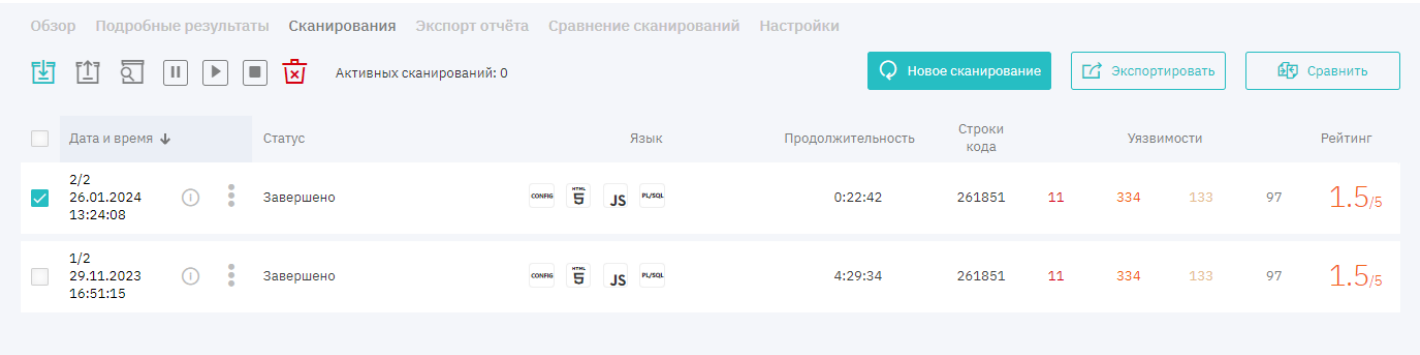


Рис. 3.15: Новое сканирование

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



Сравн
сканирований

3.3.5 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.

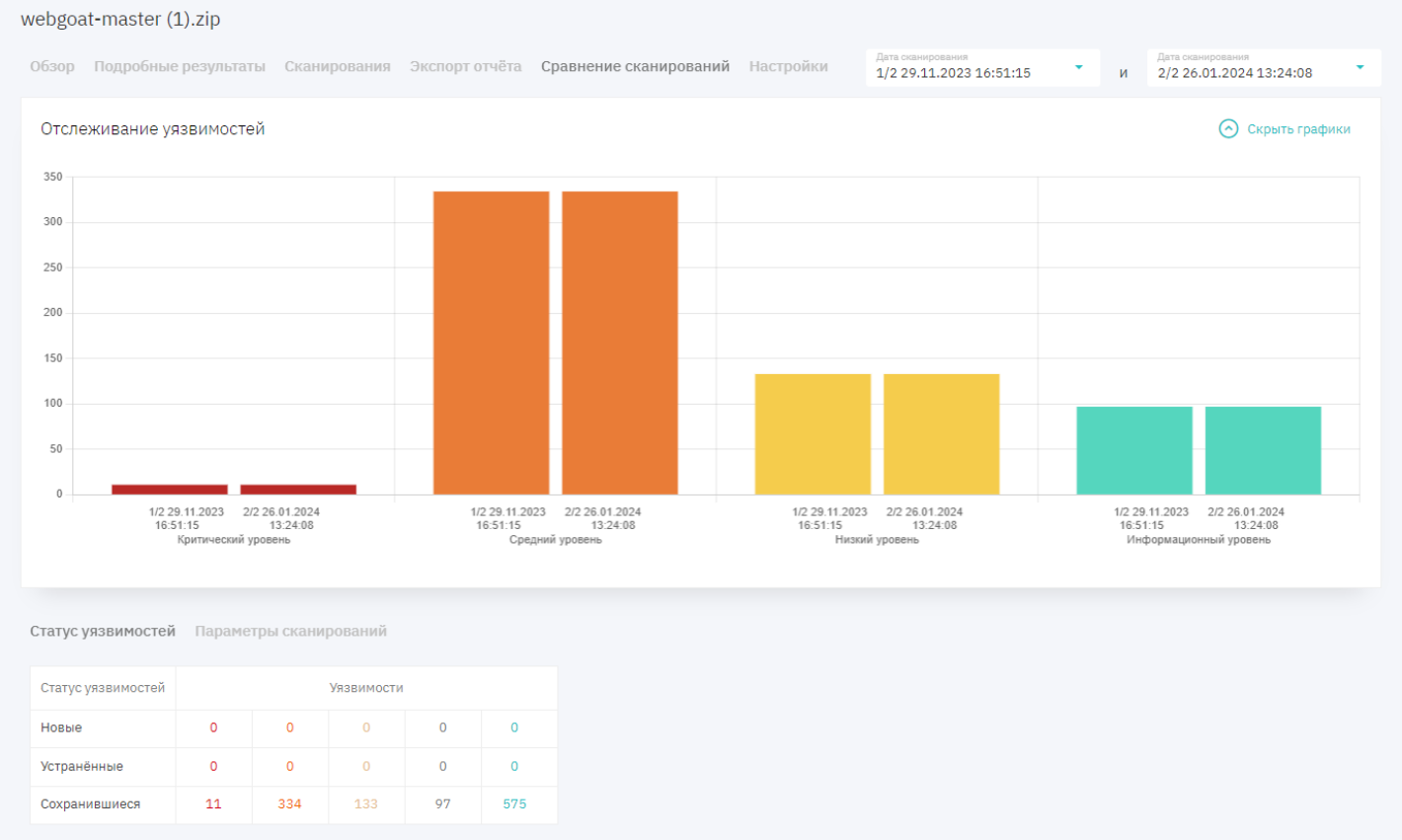


Рис. 3.16: Сравнение сканирований

3.3.6 Настройки

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

На вкладке **Таск-менеджер** можно привязать проект **Jira** к проекту DerScanner (подробнее см. раздел Как привязать проект в DerScanner к проекту в Jira).

3.3.6.1 Автоматическое сканирование

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

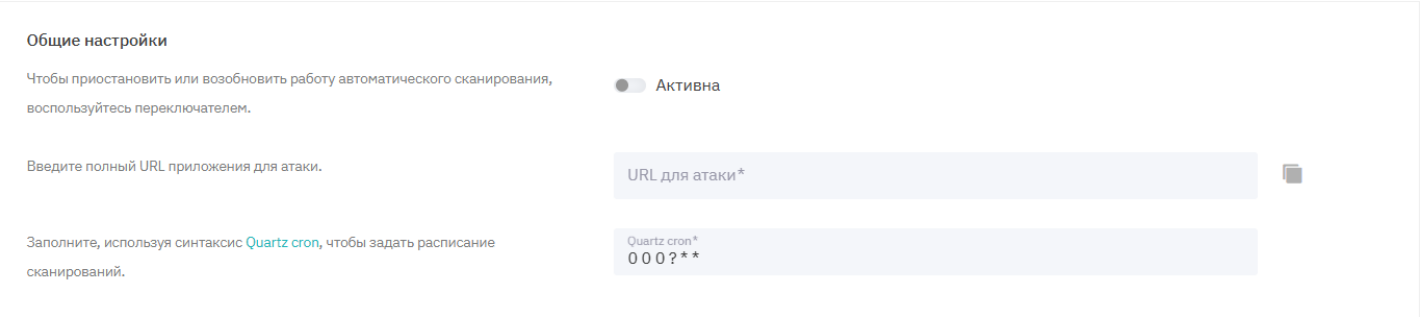


Рис. 3.17: Сканирование по расписанию

3.3.6.2 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.

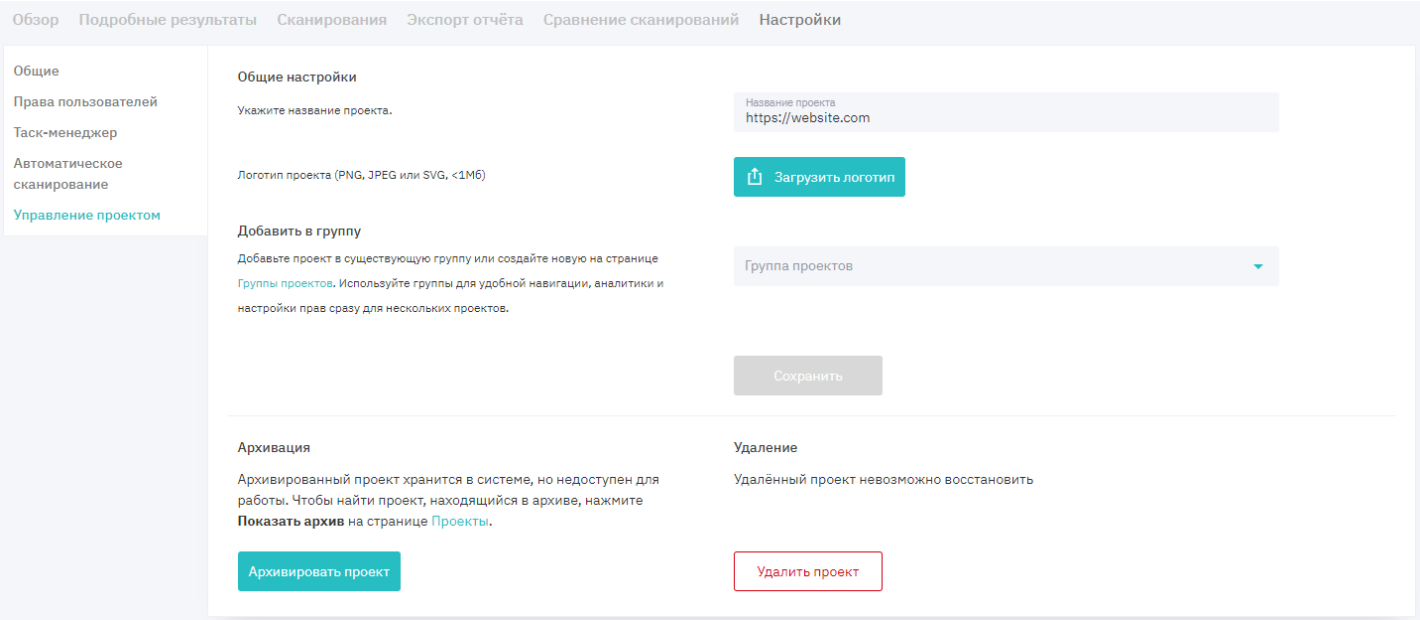


Рис. 3.18: Управление проектом

3.4 Корреляция результатов с проектами статического анализа

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

3.4.1 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**.

3.4.2 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.

3.4.3 Экспорт отчёта

Чтобы добавить информацию о корреляции в отчёт сканирования, нужно активировать опцию **Отображать метки корреляции результатов** в блоке **Общие настройки отчёта**.

3.5 API

3.5.1 Запуск сканирования из API

Чтобы запустить сканирование **DAST** из API:

1. Кликните **/dastProjects/{id}/scans** в разделе **dast**.
2. Нажмите **Try it out**.
3. В параметрах запроса укажите ID проекта DAST, в котором будет запущено новое сканирование. Если в настройках проекта уже добавлены аутентификационные данные, их также можно переиспользовать.
4. В теле запроса укажите настройки анализа в формате JSON.
5. Нажмите **Execute**.

3.5.2 Экспорт отчёта из API

Чтобы выгрузить отчёт сканирования проекта **DAST** из API:

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

3.6 Инстумент командной строки

3.6.1 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 создания проекта DAST из командной строки следует выполнить команду:

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

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

Аргументы (options):

Обязательные аргументы:

- `-name` название проекта DAST.

Для запуска сканирования из командной строки следует выполнить команду:

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

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

Аргументы (options):

Обязательные аргументы:

- `-project` ID проекта.
- `-url` URL для атаки.

Необязательные аргументы:

- `-ajax` использовать AJAX Spider (по умолчанию выключен).
- `-ajaxTimeout` значение таймаута для AJAX в минутах (по умолчанию 5).
- `-auth` способ аутентификации для разделов ресурса: BASIC, TOKEN, HEADERS, FORM.
- `-authFormData` форма авторизации. POST запрос.
- `-authToken` токен авторизации.
- `-excludeUrl` исключение URL из сканирования. Если несколько, то перечислить через запятую.
- `-headers` авторизация по заголовкам.
- `-loginFormName` логин формы авторизации.
- `-openapiPath` путь к файлу OpenAPI схемы.
- `-openapiUrl` URL OpenAPI схемы.
- `-password` пароль для стандартной авторизации (BASIC).
- `-passwordFormName` пароль формы авторизации.
- `-priority` приоритет выполнения сканирования.
- `-regexLoggedIn` регулярное выражение в сообщении о входе в форме авторизации.
- `-regexLoggedOut` регулярное выражение в сообщении о выходе в форме авторизации.
- `-reuseAuth` использовать данные в следующих запусках анализа проекта для DAST сканирований (по умолчанию выключено).
- `-scanMode` выбор режима сканирования (стандартный, агрессивный, активная атака).
- `-urlLoginPage` URL страницы входа для формы авторизации.
- `-urlLogoutPage` URL страницы выхода для формы авторизации.

- `-username` имя пользователя для стандартной авторизации (BASIC).

Пример:

```
java -jar clt.jar -rest [rest_url] -token [token] dastScanCreate -url URL -project PROJECT
```

Для проверки статуса сканирования DAST из командной строки следует выполнить команду:

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

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

Аргументы (options):

- `-scan` 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 [rest_url] -token [token] dastScanInfo -scan SCAN
```

3.6.2 Экспорт отчёта из инструмента командной строки

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

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

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

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

- `-rest` адрес API (URL).
- `-token` токен авторизации API; можно получить в интерфейсе пользователя в разделе **Аккаунт** (см. раздел Аккаунт). 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.critical` включить в отчёт уязвимости критического уровня (включается по умолчанию).
- `-filter.info` включить в отчёт уязвимости информационного уровня.
- `-filter.low` включить в отчёт уязвимости низкого уровня.
- `-filter.medium` включить в отчёт уязвимости среднего уровня (включается по умолчанию).
- `-filter.tasks` включить в отчёт уязвимости с созданными задачами в таск-менеджере (включается по умолчанию).
- `-general.contents` включить оглавление (включается по умолчанию).
- `-general.format` формат отчёта. Принимает значения: PDF, CSV, JSON, HTML. По умолчанию принимает значение PDF.
- `-general.included` включить настройки экспорта (включается по умолчанию).
- `-general.locale` настроить язык отчёта (по умолчанию английский).
- `-general.logo` использовать пользовательский логотип. По умолчанию не используется. Если пользователь хочет использовать собственный логотип, необходимо указать путь до логотипа.
- `-general.statuses` отображать статусы уязвимостей (по умолчанию выключено).
- `-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.action` включить действия с уязвимостями (включается по умолчанию).
  - `-results.comment` включить комментарии к уязвимостям (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.request` включить заголовки и тела запросов и ответов (включается по умолчанию).
  - `-results.statuses` включить уязвимости со статусами (указать статусы). По умолчанию все уязвимости, кроме отклоненных.
  - `-results.taskInfo` включить информацию о задачах в таск-менеджере (включается по умолчанию).
- `-scanInfo.included` включить настройки сканирования (включается по умолчанию). При включении укажите:
  - `-scanInfo.foundChart` включить диаграмму найденных уязвимостей (включается по умолчанию).
  - `-scanInfo.settings` включить настройки сканирования (включается по умолчанию).
  - `-scanInfo.typeChart` включить диаграмму найденных уязвимостей (включается по

умолчанию).
- `-table.included` включить список уязвимостей (включается по умолчанию). При включении в отчёт укажите:
  - `-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.statuses` включить уязвимости со статусами (указать статусы). По умолчанию все уязвимости, кроме отклоненных.
- `-comparison.included` включать ли в отчёт сравнение с предшествующим сканированием. При включении в отчёт укажите:
  - `-comparison.Scan` UUID предшествующего сканирования для сравнения (обязательный параметр).
  - `-comparison.fixed` включить в отчёт устранённые уязвимости.
  - `-comparison.newIssue` включить 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 [rest_url] -token [token] dastReport -path PATH -project PROJECT
```

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.

04 Компонентный анализ

В DerScanner реализована возможность сканировать приложения с целью выявления уязвимых компонент и зависимостей в open-source библиотеках. Сторонние библиотеки можно проверить на наличие уязвимостей, а также наличие лицензионных и Supply chain рисков.

4.1 Особенности запуска анализа

Анализ на уязвимости, лицензионные риски и атаки Supply Chain можно запускать по:

- заранее собранному SBOM-файлу проекта;
- исходному коду проекта (для проектов на JavaScript/TypeScript, PHP, Python, Ruby, C#/VB.NET, C/C++/Objective-C, GO, Java/Kotlin/Scala, Rust, Swift, Erlang. Анализ в этом случае все равно пройдет и использованием SBOM, но он будет собран средствами DerScanner.

Для запуска гибридного анализа SCA + SAST (он доступен для проектов на JavaScript/TypeScript, Python с анализом транзитивных и директивных зависимостей и C#, Java с анализом только директивных зависимостей) необходима загрузка исходного кода проекта с исходным кодом зависимостей. Допустима загрузка на анализ и исходного кода, и SBOM-файла. В таком случае для анализа уязвимостей будет использован загруженный SBOM и не будет собираться новый.

Для того, чтобы гибридный анализ прошел корректно, необходимо:

- Для JS/TS проектов для анализа необходима директория со сторонними зависимостями проекта. По умолчанию директория называется «node_modules» и с таким названием анализ будет проходить по умолчанию. Если название директории в проекте отличается, если надо указать в настройках сканирования.
- Для Python проектов для анализа необходима директория со сторонними зависимостями проекта. По умолчанию директория называется «venv» и с таким названием анализ будет проходить по умолчанию. Если название директории в проекте отличается, если надо указать в настройках сканирования.
- Для Java проектов для анализа необходима директория со сторонними зависимостями проекта, включая `.jar`-файлы. Для этого можно использовать Maven-плагин **dependency:copy-dependencies**, который копирует все зависимости в указанный каталог. По умолчанию директория называется «.m2/repository/» и с таким названием анализ будет проходить по умолчанию. Если название директории в проекте отличается, если надо указать в настройках сканирования.
- Для C# проектов для анализа необходим только исходный код проекта.

4.1.1 Инструкция по сборке SBOM файла

4.1.1.1 С помощью DerScanner

Для успешной сборки SBOM необходимо, чтобы архив или репозиторий (Git, Subversion) содержали как исходный код, так и соответствующие манифесты. В DerScanner реализованы следующие способы сборки SBOM по исходному коду:

- **Онлайн** — если DerScanner имеет доступ в Интернет. Обратите внимание, что при онлайн-генерации SBOM-файла все необходимые компоненты загружаются из Интернета, что несёт в себе потенциальные риски для безопасности вашей системы. Чтобы собрать SBOM-файл из исходного кода, необходимо открыть доступ к следующим источникам:
  - https://www.google.com/
  - https://www.npmjs.com
  - https://github.com/
  - https://maven.pkg.github.com/
  - https://pkg.go.dev/
  - https://cocoapods.org/
  - https://packagist.org/
  - https://crates.io/
  - https://rubygems.org/
  - https://rebar3.org/

В зависимости от проекта DerScanner посетит один или несколько источников.

- **Автономный режим** — если DerScanner не имеет доступа к Интернету. В этом случае генератор собирает SBOM-файл только по тем компонентам, данные о которых уже доступны в его локальной базе данных.

4.1.1.2 Самостоятельно

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.

4.2 Настройки сканирования

Сканирование можно настроить через интерфейс с помощью соответствующих параметров, аналогичные настройки доступны также через API и инструмент командной строки (CLT):

- Выбор типа анализа SCA, Supply chain и анализ лицензионных рисков;
- Выбор приоритета сканирования;
- Выбор агента сканирования, который будет выполнять анализ;
- Общие настройки:
  - «Сохранить загруженный файл». При выборе опции файл останется доступным для администрирования системы;
  - «Гибридный анализ». Позволяет включить гибридный анализ SCA + SAST для проекта;
  - «Собрать SBOM только с прямыми зависимостями». Для проектов с исходным кодом (Gradle, Maven), опция позволяет собрать SBOM и выполнить анализ только прямых зависимостей.
- Настройки приватного репозитория:
  - Поля для ввода учетных данных (логин/пароль, токен, SSH ключ).
  - Поле для указания ветки репозитория Git.
  - «Анализ подмодулей». Опция для включения анализа подмодулей.

4.3 Пользовательский интерфейс

4.3.1 Создание проекта

В интерфейсе DerScanner реализованы следующие способы создания проекта:

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

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.

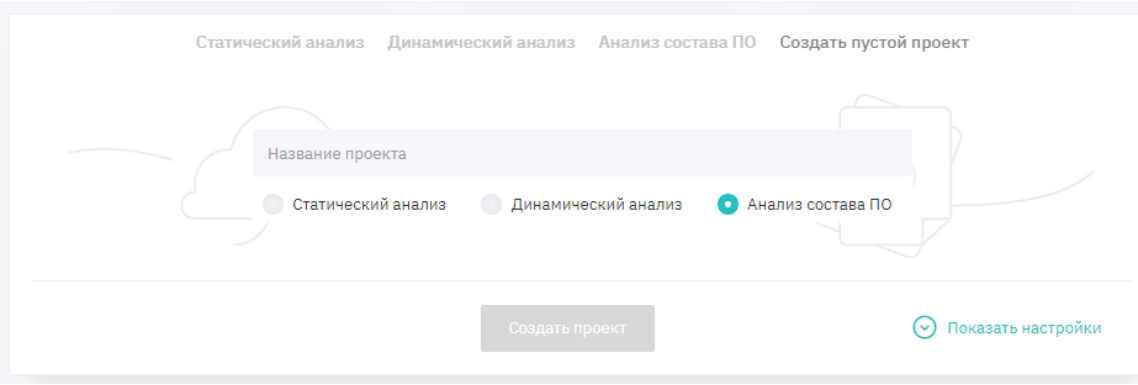


Рис. 4.1: Создание пустого проекта

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

4.3.2 Запуск сканирования

Чтобы запустить новое сканирование в UI:

1. На Домашней странице, перейдите на вкладку **Анализ состава ПО**.
2. В блоке **SBOM файл** загрузите проект в виде SBOM файла в формате Cyclone DX (архив с SBOM файлом или ссылка на SBOM файл в репозитории приведет к ошибке сканирования).

   DerScanner может автоматически собрать SBOM файл из исходного кода для проектов, написанных на JavaScript/TypeScript, PHP, Python, Ruby, C#/VB.NET, C/C++/Objective-C, GO, Java/Kotlin/Scala, Rust, Swift, Erlang.
3. Настройте анализ.
4. Нажмите **Начать сканирование**.

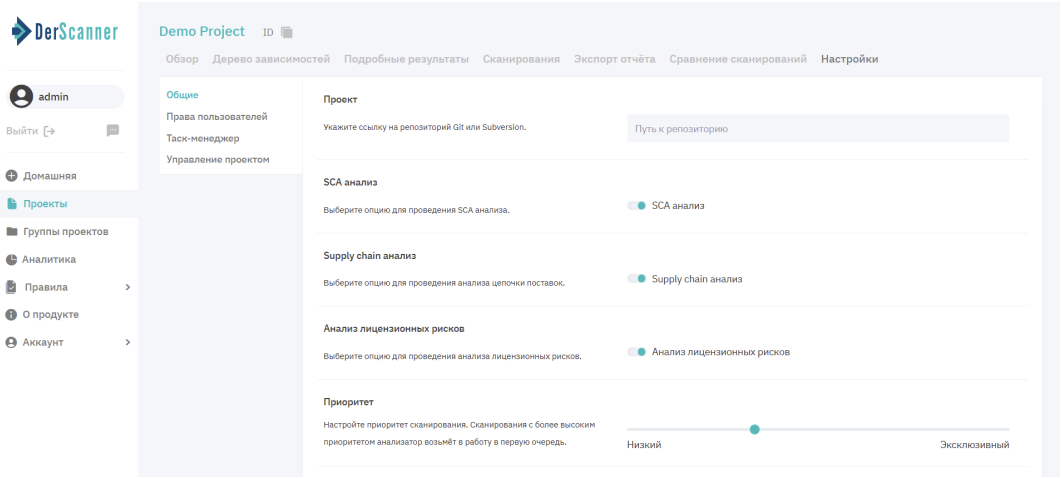


Рис. 4.2: Настройки анализа

4.3.3 Работа с результатами

4.3.3.1 Обзор

В разделе **Обзор** содержится статистика сканирования. Сканирование, для которого будут отображаться данные, можно выбрать в правом верхнем углу страницы. Нажмите на иконку , чтобы отобразились параметры запуска анализа для выбранного сканирования.

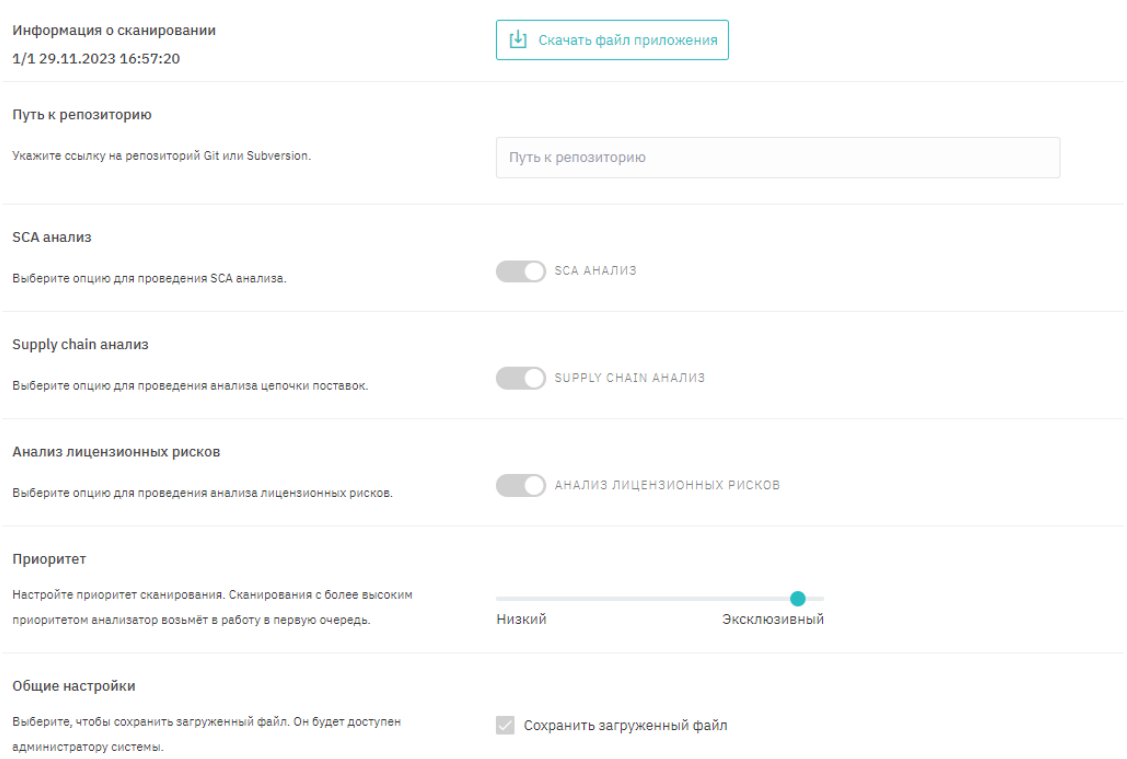


Рис. 4.3: Параметры запуска анализа

В разделе представлена следующая информация:

- рейтинг;
- статус сканирования;
- продолжительность сканирования;
- общее количество компонент;
- количество уязвимых компонент;
- дашборды по проекту и сканированию.

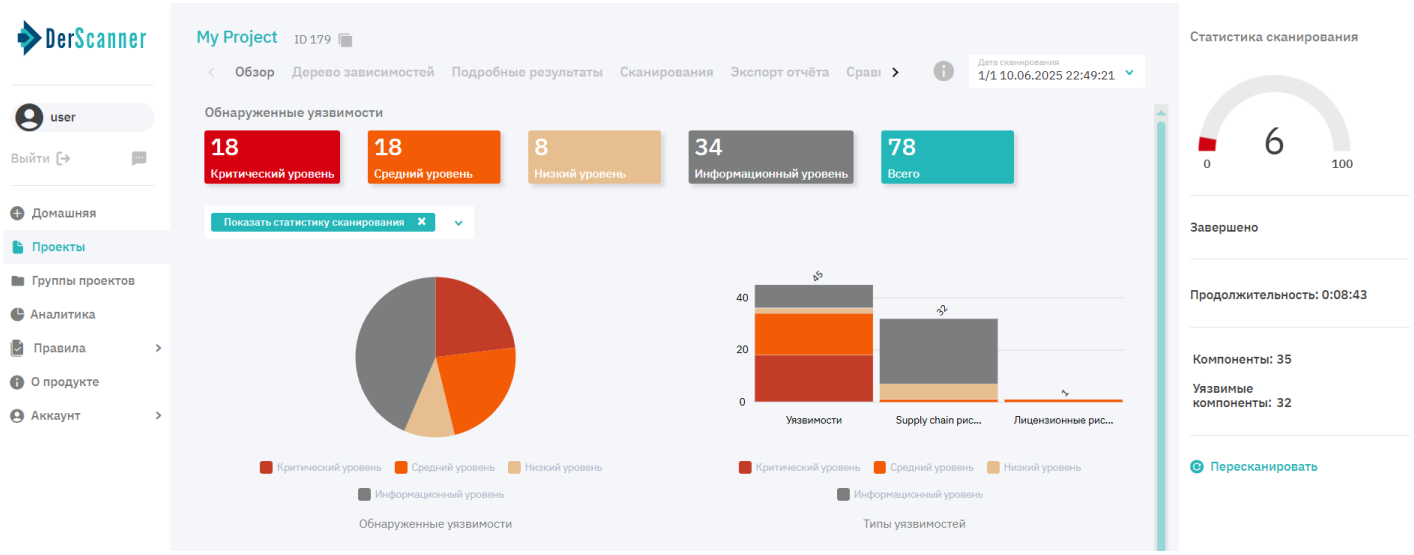


Рис. 4.4: Сводные результаты

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

4.3.3.2 Дерево зависимостей

В **Дерево зависимостей** представлен интерактивный граф зависимостей всех компонент в проекте. Сканирование, для которого будут отображаться данные, можно выбрать в правом верхнем углу страницы. Нажмите на иконку , чтобы отобразились параметры запуска анализа для выбранного сканирования.

Чтобы скрыть ветку графа, нажмите на **-** на ребре. Чтобы отменить действие, нажмите на **+**.

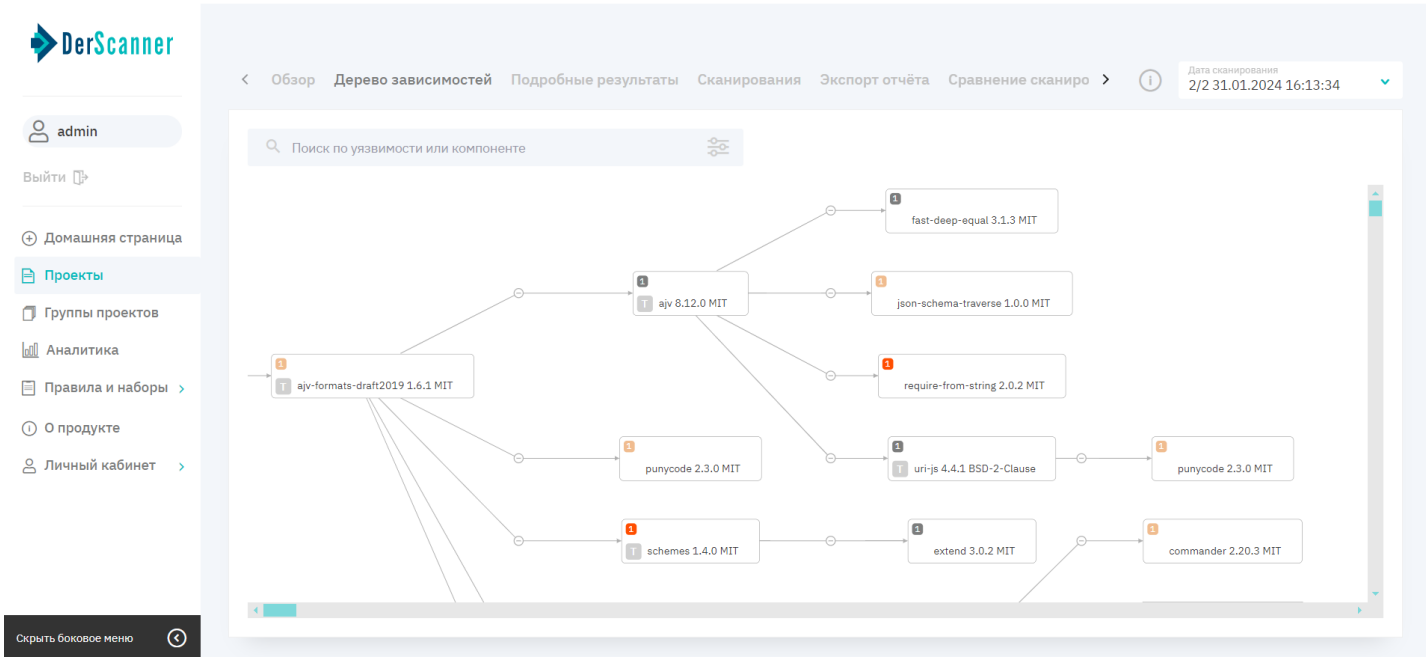


Рис. 4.5: Дерево зависимостей

Для каждой компоненты указывается название, версия и лицензия, а также уязвимости при наличии. Список уязвимостей можно развернуть по клику на компоненту. Клик по уязвимости осуществляет переход к данной уязвимости на странице **Подробные результаты**.

Для удобной навигации предусмотрен поиск по названию уязвимости или компоненте, а также фильтры.

4.3.3.3 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.

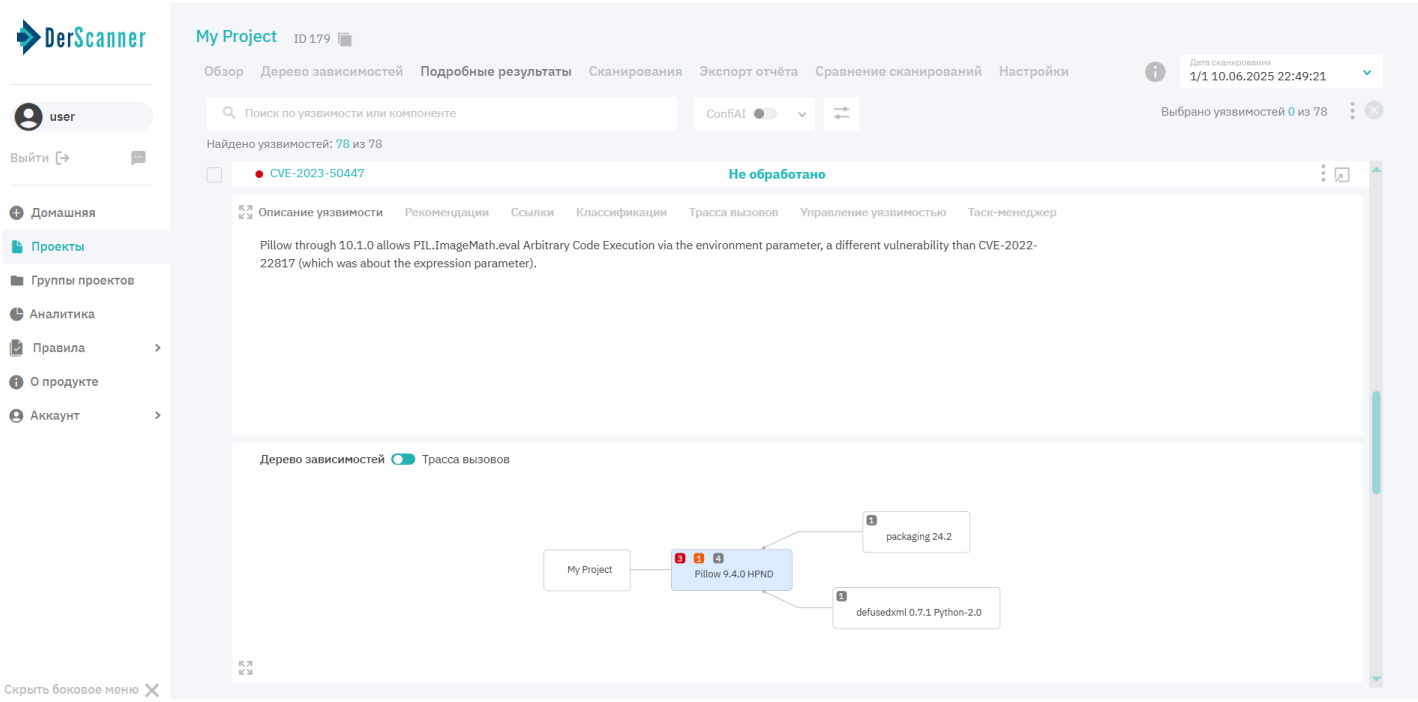


Рис. 4.6: Подробные результаты

На странице 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.

Если компонента содержит **Supply chain** или **лицензионные** риски, они будут отмечены соответствующими тэгами: **S** для Supply chain, **L** для лицензионных рисков. Чтобы ознакомиться с подробной информацией о рисках, нажмите на название библиотеки в списке компонент и на название риска в открывшемся списке вхождений.

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. 4.7).

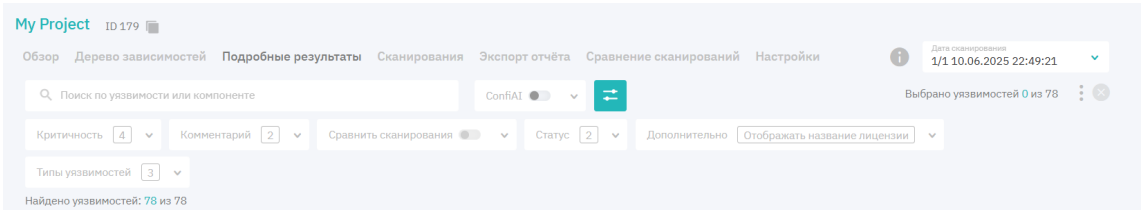


Рис. 4.7: Фильтры результатов

Фильтровать результаты можно по следующим параметрам:

- типы уязвимостей:
  - уязвимости;
  - supply chain риски;
  - лицензионные риски.

- статусы уязвимостей для отображения:
  - подтверждено;
  - не обработано;
  - отклонено.
- включить ConfiAI 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 — позволяет настроить чувствительность ConfiAI, 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 — позволяет настроить значение EPSS, перемещая ползунок в разные положения. Крайнее левое положение ползунка (1) характеризует вхождения с самой высокой вероятностью эксплуатации, крайнее правое (0) отобразит уязвимости для любой вероятности.
- 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.

Нажмите на три точки справа от названия 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 повторном сканировании изменения сохраняются.

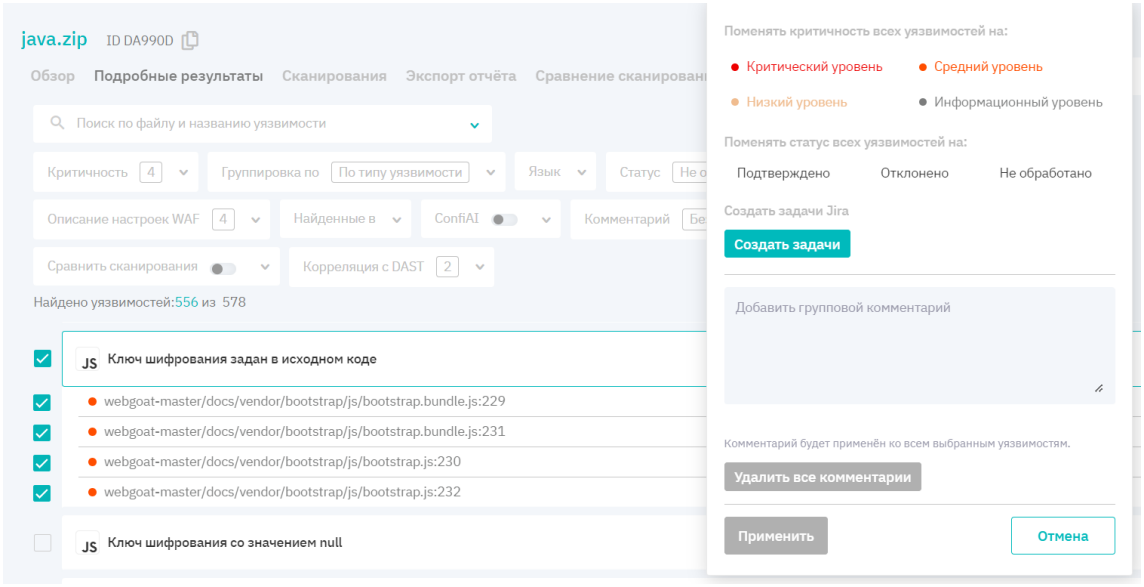


Рис. 4.8: Управление пакетом уязвимостей

После выбора конкретной уязвимости в правой части страницы отображается соответствующее ей дерево зависимостей компоненты. Чтобы раскрыть информацию об уязвимости на полный экран, нажмите на иконку в правом верхнем углу. В левой части представлена следующая информация (рис. 4.9): **Описание уязвимости**, **Рекомендации**, **Ссылки**, **Классификации**, **Управление уязвимостью**, **Таск-менеджер**.

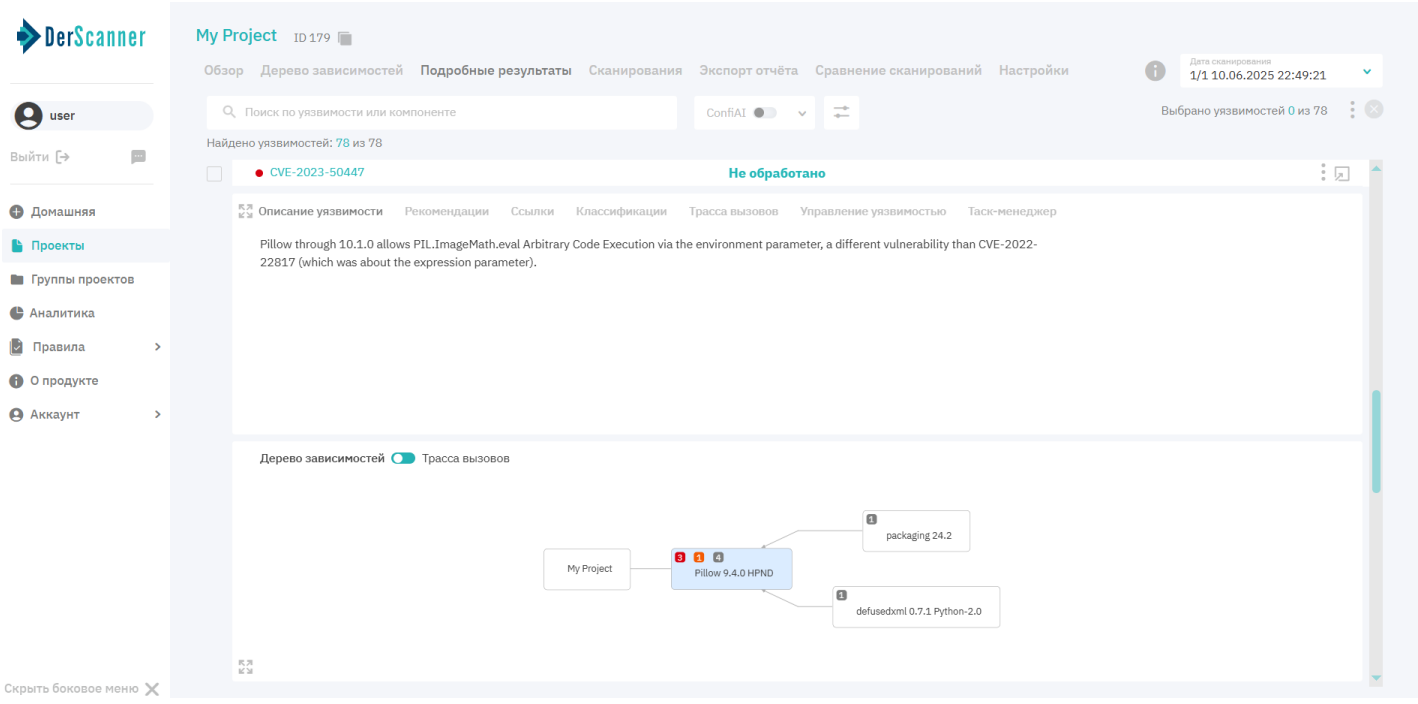


Рис. 4.9: Свойства уязвимости

В случае, когда вместе с SBOM файлом был просканирован исходный код проекта, список вызовов и импортов компоненты будет доступен на вкладке **Список вызовов**. Для просмотра необходимо выбрать элемент на трассе вызовов. Переключение между режимами просмотра

частичного дерева зависимостей и трассы вызовов происходит с помощью переключателя над областью с деревом.

Для рисков Supply chain предоставляется **Описание** с оценкой по метрикам, **Управление уязвимостью**, **Таск-менеджер**.

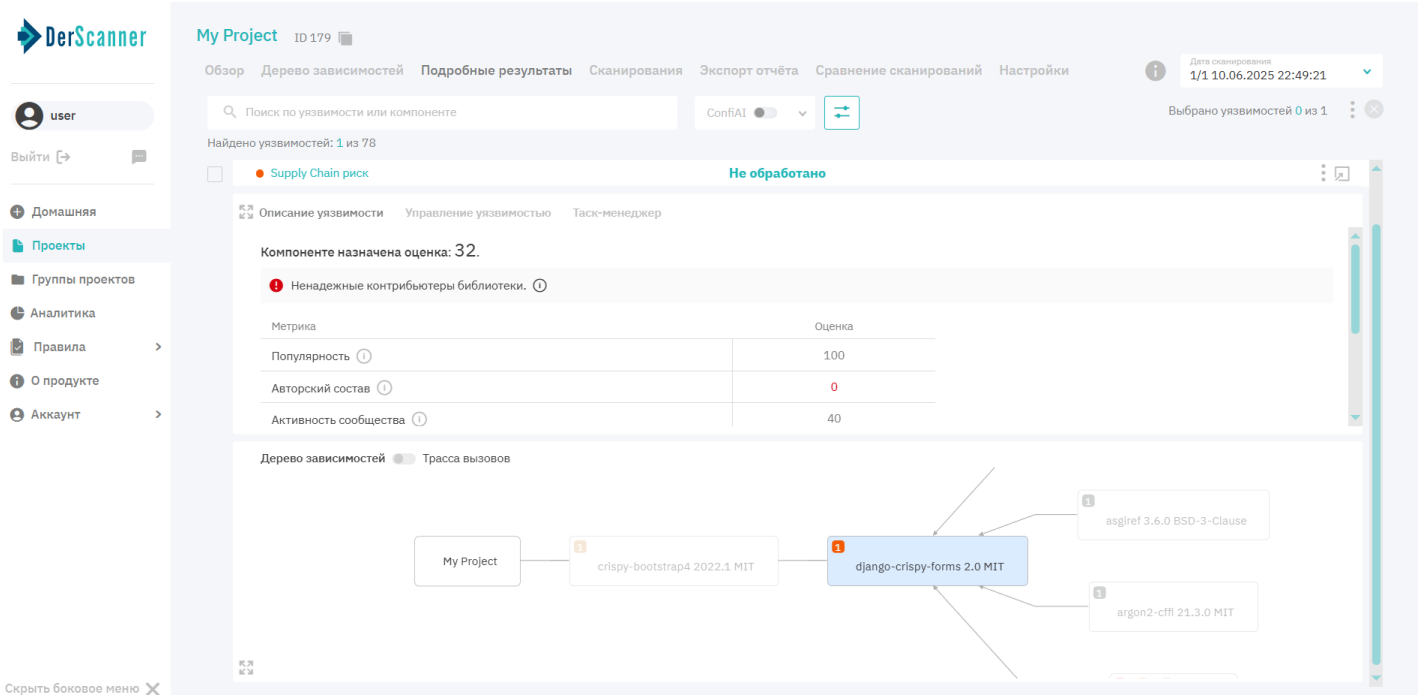


Рис. 4.10: Оценка компоненты

Для лицензионных рисков предоставляется **Описание**, **Управление уязвимостью**, **Таск-менеджер**.

На вкладке **Управление уязвимостью** (рис. 4.11) можно изменить уровень критичности и статус, добавить комментарий к уязвимости и посмотреть журнал событий с оставленными ранее комментариями и действиями.

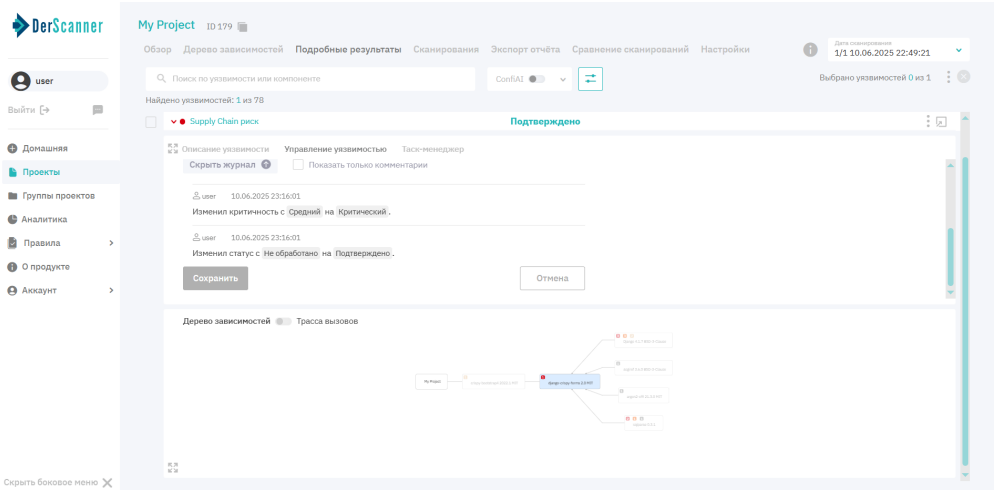


Рис. 4.11: Управление уязвимостью

На вкладке также отображаются EPSS и CVSS оценки для вхождения. Они показывают уверенность в том, что данная уязвимость может быть проэксплуатирована: чем выше оценка, тем меньше вероятность, что вхождение ложно-положительное.

4.3.3.4 Отчёт безопасности

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

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

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.

4.3.3.4.1 Информация о проекте

В отчёт можно включить динамику уровня безопасности и историю сканирований.

4.3.3.4.2 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.

Выберите уязвимости по уровню критичности и типу, а также компоненты для отображения. Воспользуйтесь фильтром ConfiAI в одном из режимов: **Только истинные**, **Только важные**, **Пользовательский** или **Динамический** режимы.

Выберите статусы уязвимостей и задайте количество их вхождений, а также настройте добавление полного дерева зависимостей.

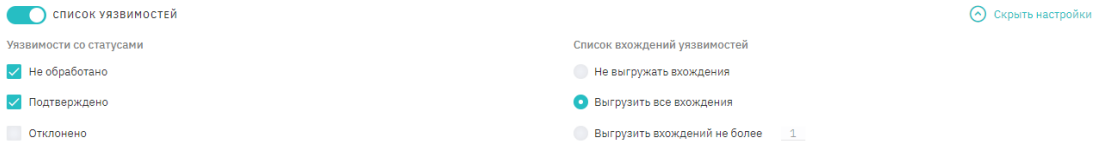


Рис. 4.12: Список уязвимостей

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 журнала событий:
  - не включать;
  - включить только комментарии;
  - включить только действия;
  - журнал целиком;
- отображение дополнительной информации.

4.3.3.4.3 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.

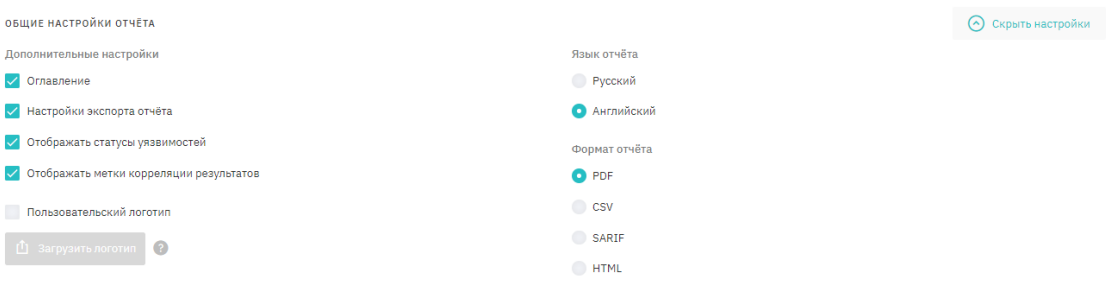


Рис. 4.13: Общие настройки отчёта

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.

4.3.4 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.

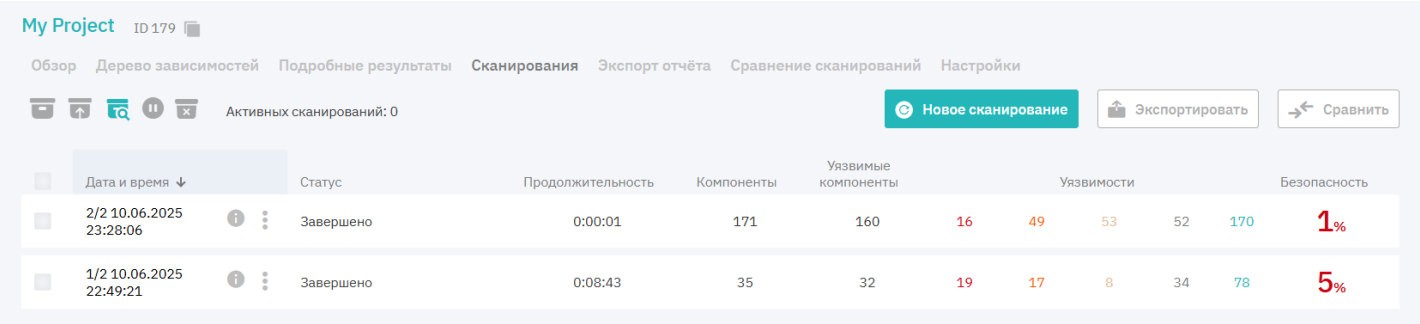


Рис. 4.14: Сканирования

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**.

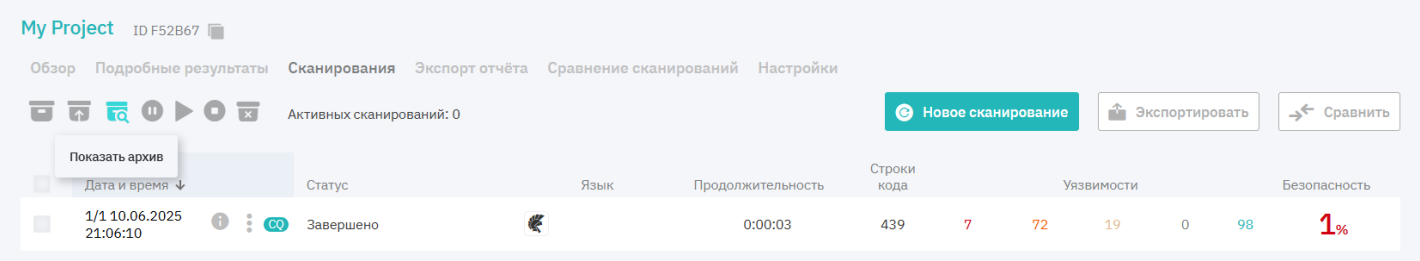


Рис. 4.15: Показать/Скрыть архив

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

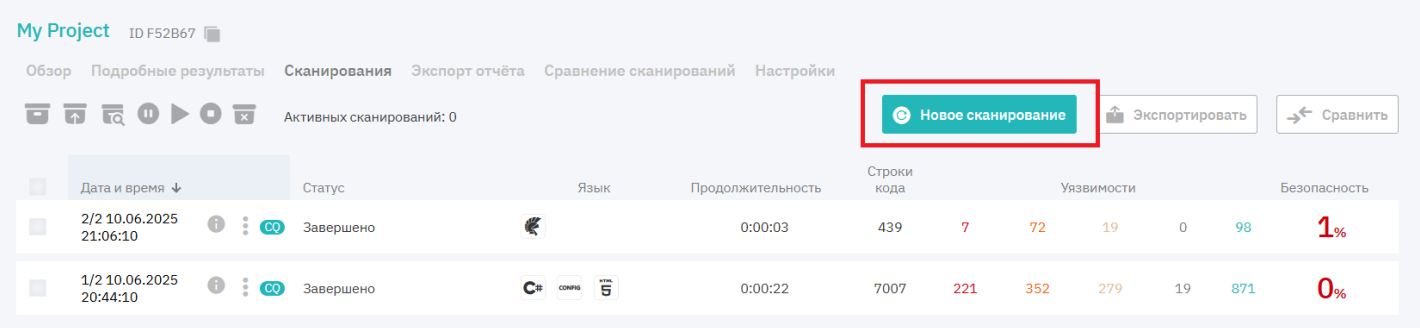


Рис. 4.16: Новое сканирование

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

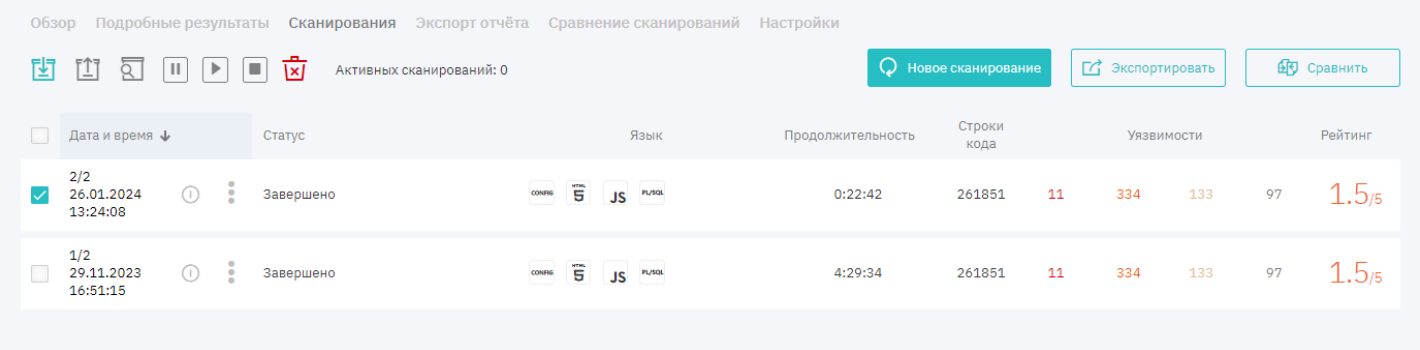


Рис. 4.17: Очередь сканирований

4.3.5 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.

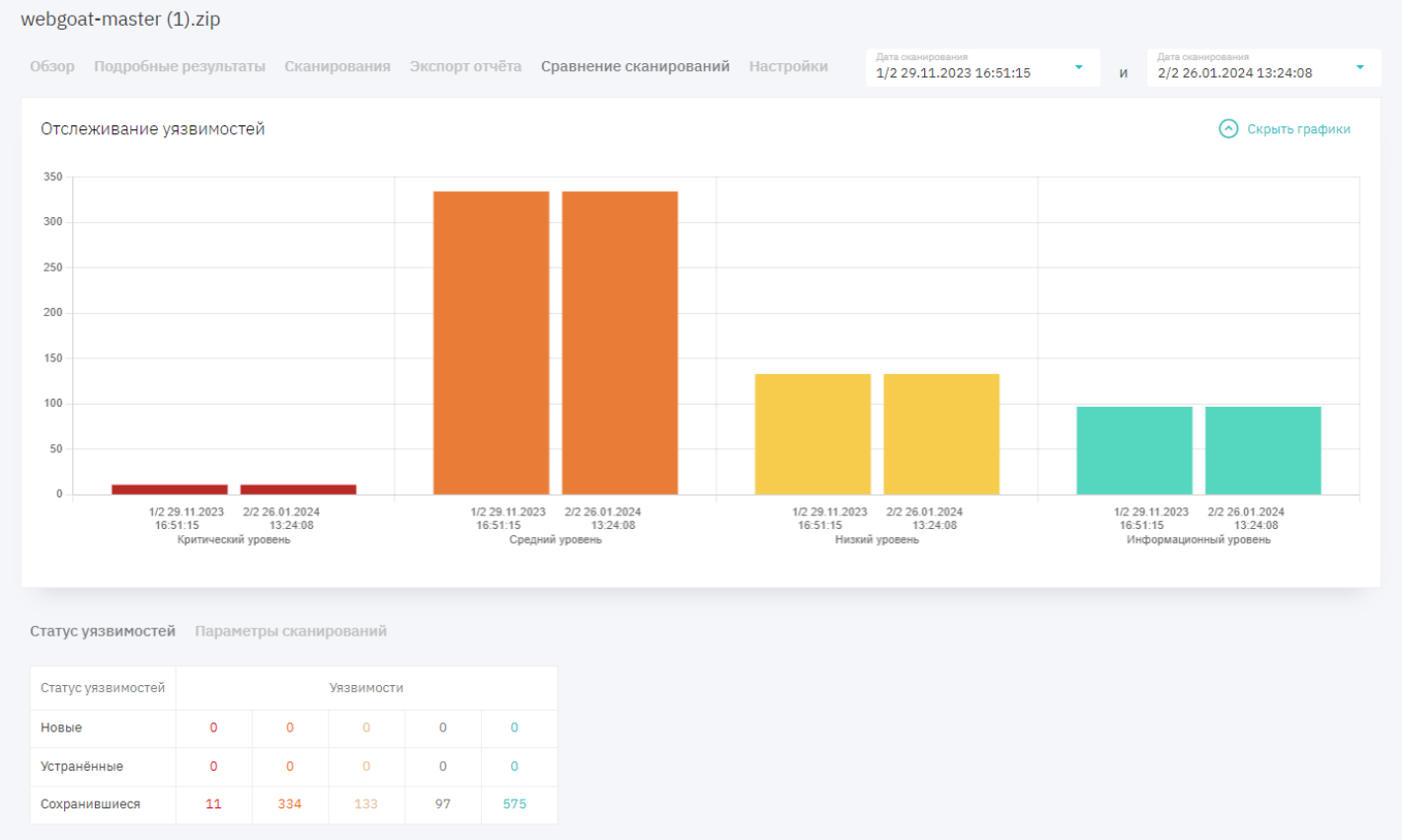


Рис. 4.18: Сравнение сканирований

4.3.6 Настройки

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

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

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.

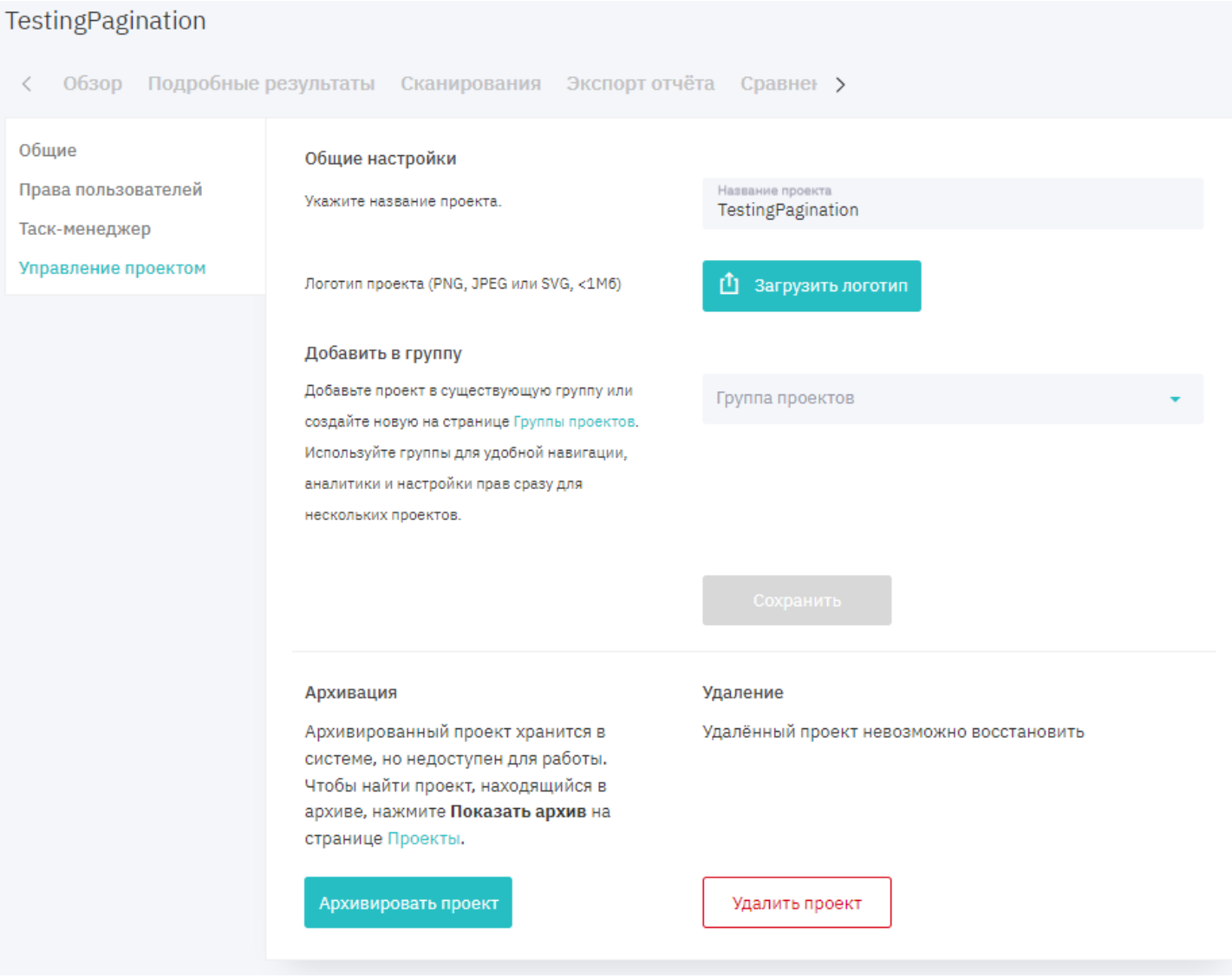


Рис. 4.19: Управление проектом

4.3.7 API

4.3.7.1 Запуск сканирования из API

Чтобы запустить сканирование **SCA** из API:

1. Перейдите в раздел **sca**.
2. Выберите нужный запрос в зависимости от вида исходных данных для анализа:
   - **/sca_projects/{id}/scans/archive** — для анализа по архиву с исходным кодом + SBOM-файлу проекта;
   - **/sca_projects/{id}/scans/url** — для анализа по ссылке на архив с исходным кодом + SBOM-файлу проекта;
   - **/sca_projects/{id}/scans/vcs** — для анализа по ссылке на репозиторий + SBOM-файлу проекта;
   - **/sca_projects/{id}/scans/url** — для анализа по SBOM-файлу проекта.
3. Нажмите **Try it out**.
4. В параметрах запроса укажите ID проекта SCA, в котором будет запущено новое сканирование.
5. В зависимости от выбранного запроса, в теле запроса выберите файлы для анализа или укажите ссылку на проект. Если вы загружаете файл, важно либо запустить анализ из директории с файлом, либо прописать путь к файлу.
6. Укажите дополнительные настройки.
7. Нажмите **Execute**.

4.3.7.2 Экспорт отчёта из API

Чтобы выгрузить отчёт сканирования проекта **SCA** из API:

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

4.3.8 Инструмент командной строки

4.3.8.1 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 создания проекта из командной строки следует выполнить команду:

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

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

Аргументы (options):

Обязательные аргументы:

- `-name` название проекта.

Для запуска сканирования из командной строки следует выполнить команду:

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

- `-rest` адрес API (URL).
- `-token` токен авторизации API; можно получить в интерфейсе пользователя в разделе `Аккаунт` (см. раздел Аккаунт).

Аргументы (options):

Обязательные аргументы:

- `-project` ID проекта.
- `-path` путь к каталогу или файлу для анализа (не должен использоваться, если указан параметр link).
- `-link` URL репозитория (не должен использоваться, если указан параметр path).

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

Необязательные аргументы:

- `-sca` выполнить SCA анализ (по умолчанию включено).
- `-licenseRisks` выполнить анализ лицензионных рисков (по умолчанию включено).
- `-supplyChain` выполнить Supply Chain анализ (по умолчанию включено).
- `-priority` приоритет выполнения сканирования.
- `-saveFile` полностью сохранить загружаемый файл.
- `-vcs.branch` ветка репозитория, если не master (при анализе приложений по ссылке на репозиторий).
- `-vcs.account` UUID репозитория.
- `-vcs.login` логин репозитория.
- `-vcs.password` пароль репозитория.
- `-vcs.token` токен аутентификации репозитория.
- `-vcs.tokenId` UUID токена аутентификации.
- `-vcs.sshKey` путь до ключа SSH репозитория.
- `-vcs.sshKeyId` UUID ключа SSH.
- `-vcs.sshDefault` использовать стандартную конфигурацию SSH-клиента. Если значение false, то указать конфигурацию SSH-клиента.
- `-vcs.saveCredentials` использовать учётные данные при пересканировании.
- `-iconPath` путь к логотипу проекта.

Пример:

```
java -jar clt.jar -rest [rest_url] -token [token] scaScanCreate -link [link] -project PROJECT
```

Для проверки статуса сканирования из командной строки следует выполнить команду:

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

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

Аргументы (options):

- `-scan` ID сканирования.

Пример:

```
java -jar clt.jar -rest [rest_url] -token [token] scaScanInfo -scan SCAN
```

4.3.8.2 Экспорт отчета из инструмента командной строки

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

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

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

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

- `-rest` адрес API (URL).
- `-token` токен авторизации API; можно получить в интерфейсе пользователя в разделе `Аккаунт` (см. раздел Аккаунт).

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` ID проекта.

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

Опциональные аргументы:

- `-scans` ID сканирований, которые следует включить в отчёт. Если сканирований несколько, то перечислять их следует через запятую.
- `-filter.vulnerabilities` включить в отчёт уязвимости (включается по умолчанию).
- `-filter.critical` включить в отчёт уязвимости критического уровня (включается по умолчанию).
- `-filter.medium` включить в отчёт уязвимости среднего уровня (включается по умолчанию).
- `-filter.low` включить в отчёт уязвимости низкого уровня.
- `-filter.info` включить в отчёт уязвимости информационного уровня.

- `-filter.licenseRisks` включить в отчёт лицензионные риски (включается по умолчанию).
- `-filter.supplyChain` включить в отчёт Supply chain риски (включается по умолчанию).
- `-filter.vulnerableDependencies` включить в отчёт компоненты с уязвимостями (включается по умолчанию).
- `-filter.safeDependencies` включить в отчёт компоненты без уязвимостей.
- `-filter.tasks` включить в отчёт уязвимости с созданными задачами в таск-менеджере (включается по умолчанию).
- `-confidence.included` включить в отчёт данные Confi AI. При включении укажите:
  - `-confidence.critical` настройка значения Confi AI для критических уязвимостей (по умолчанию 0).
  - `-confidence.medium` настройка значения Confi AI для уязвимостей среднего уровня (по умолчанию 0).
  - `-confidence.low` настройка значения Confi AI для уязвимостей низкого уровня (по умолчанию 0).
  - `-confidence.info` настройка значения Confi AI для уязвимостей информационного уровня (по умолчанию 0).
- `-general.contents` включить оглавление (включается по умолчанию).
- `-general.format` формат отчёта. Принимает значения: PDF, CSV, JSON, HTML. По умолчанию принимает значение PDF.
- `-general.included` включить настройки экспорта (включается по умолчанию).
- `-general.locale` настроить язык отчёта (по умолчанию английский).
- `-general.logo` использовать пользовательский логотип. По умолчанию не используется. Если пользователь хочет использовать собственный логотип, необходимо указать путь до логотипа.
- `-general.statuses` отображать статусы уязвимостей (по умолчанию выключено).
- `-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.action` включить действия с уязвимостями (включается по умолчанию).
  - `-results.comment` включить комментарии к уязвимостям (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.statuses` включить уязвимости со статусами (указать статусы). По умолчанию все уязвимости, кроме отклоненных.
  - `-results.taskInfo` включить информацию о задачах в таск-менеджере (включается по умолчанию).
  - `-results.dependencies` включить метки зависимостей (включается по умолчанию).
  - `-results.dependencyTree` включить дерево зависимостей компоненты.
- `-scanInfo.included` включить настройки сканирования (включается по умолчанию). При включении укажите:

  - `-scanInfo.foundChart` включить диаграмму найденных уязвимостей (включается по умолчанию).
  - `-scanInfo.settings` включить настройки сканирования (включается по умолчанию).
  - `-scanInfo.dendenciesChart` включить диаграмму найденных зависимостей (включается по умолчанию).
- `-table.included` включить список уязвимостей (включается по умолчанию). При включении в отчёт укажите:
  - `-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.statuses` включить уязвимости со статусами (указать статусы). По умолчанию все уязвимости, кроме отклоненных.
  - `-table.dependencyTree` включить полное дерево зависимостей.
- `-comparison.included` включать ли в отчёт сравнение с предшествующим сканированием. При включении в отчёт укажите:
  - `-comparison.ScanId` ID сканирования для сравнения (обязательный параметр).
  - `-comparison.fixed` включить в отчёт устранённые уязвимости.
  - `-comparison.newIssue` включить 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 scaExport -scan 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 ID необходимого сканирования (аргумент **-scans**). ID сканирования можно получить в информации о сканировании. Чтобы скопировать в буфер ID сканирования, нажмите на иконку .

Описанная выше функциональность также доступна через REST.

05 Интерфейс пользователя: общая часть

5.1 Домашняя страница

Домашняя страница стартовая страница системы, которая предназначена для загрузки и запуска сканирования новых проектов.

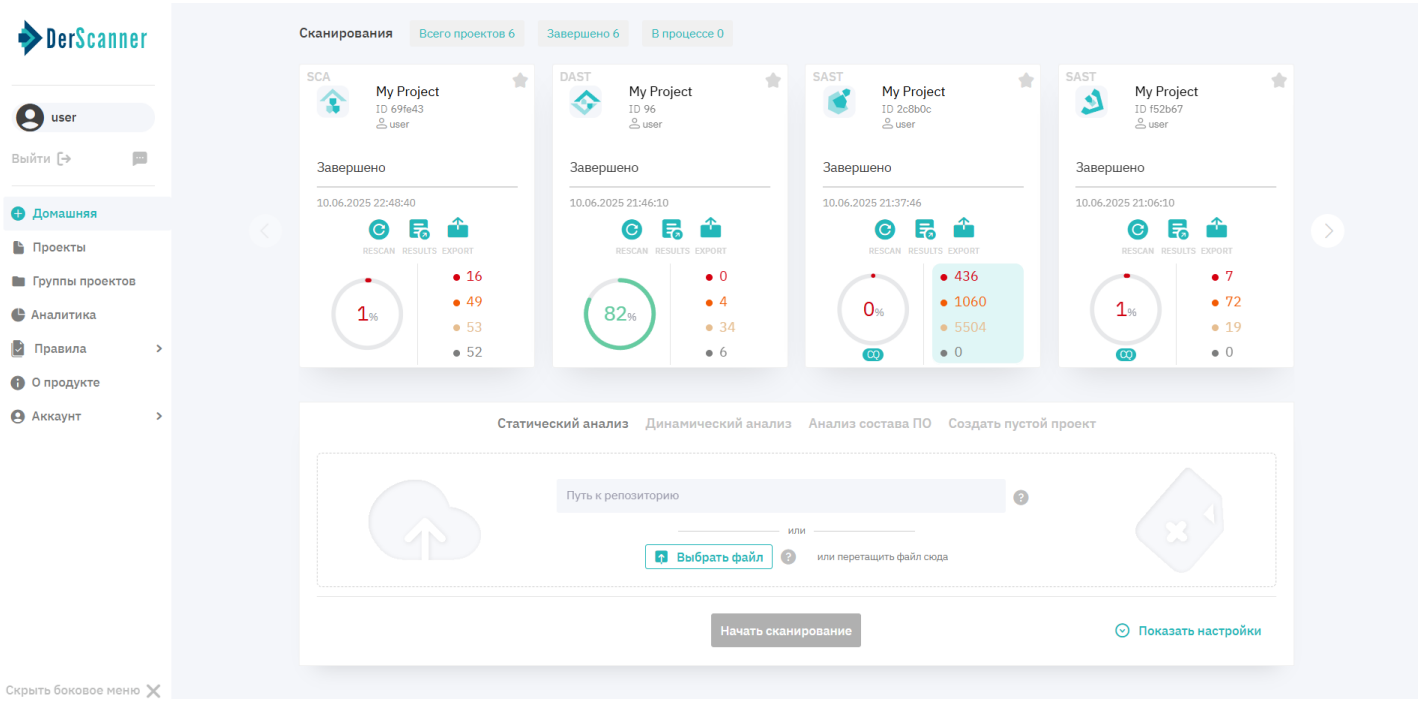


Рис. 5.1: Домашняя страница

5.2 Проекты

Страница отображает таблицу с проектами в системе и основую иноформацию по последнему сканированию в них. Используя фильтры SAST, DAST, SCA можно переключаться между типами анализа.

Для каждого проекта отображаются следующие данные:

- логотип, название проекта, автор проекта, ID проекта;
- статус последнего сканирования;
- контекстное меню действий с проектом, которое включает:
  - копировать UUID проекта;

  - посмотреть подробные результаты последнего сканирования;
  - запустить сканирование;
  - выгрузить отчёт;
  - настроить проект;
  - добавить в группу;
  - архивировать проект;

- кнопка для добавления проекта в группу «Избранное»;
- дата и время последнего сканирования;
- языки, для которых был произведен анализ (для SAST, CQ);
- количество проанализированных строк кода в сканировании (для SAST, CQ);
- количество зависимостей, найденных в проекте в результате анализа (для проектов SCA);
- количество уязвимых зависимостей, найденных в результате анализа;
- количество уязвимостей критического, среднего, низкого и информационного уровней, а также общее количество уязвимостей;
- процент безопасности приложения.

5.2.1 Очередь сканирований

На вкладке **Очередь сканирований** отображается список всех активных сканирований в системе. Используя фильтры SAST, DAST, SCA можно переключаться между типами анализа.

Система поддерживает 4 уровня приоритета сканирований:

- низкий;
- средний;
- высокий;
- эксклюзивный.

Изменение приоритета затрагивает только сканирования в очереди.

В основной части страницы отображается таблица со списком всех активных сканирований в системе. Для каждого сканирования представлена ключевая информация: имя проекта, ID сканирования и кнопка для его копирования, автор сканирования. Далее следуют столбцы с датой создания, текущим статусом сканирования, приоритет и агент, который выполняется или в очереди к которому находится сканирование.

Среди сканирований с одинаковым приоритетом первым будет просканирован проект, запущенный раньше.

5.3 Группы проектов

5.3.1 Управление группами

Проекты SAST, DAST, SCA в DerScanner можно объединять в группы. Используя группы, можно выполнять действия с несколькими логически связанными проектами одновременно.

Для создания новой группы проектов нажмите кнопку **Создать группу**, введите её имя и выберите проекты, которые необходимо включить.

Если вы хотите, чтобы созданная группа была доступна всем пользователям, установите опцию **Публичная**. Приватные группы доступны только ее автору и администратору DerScanner.

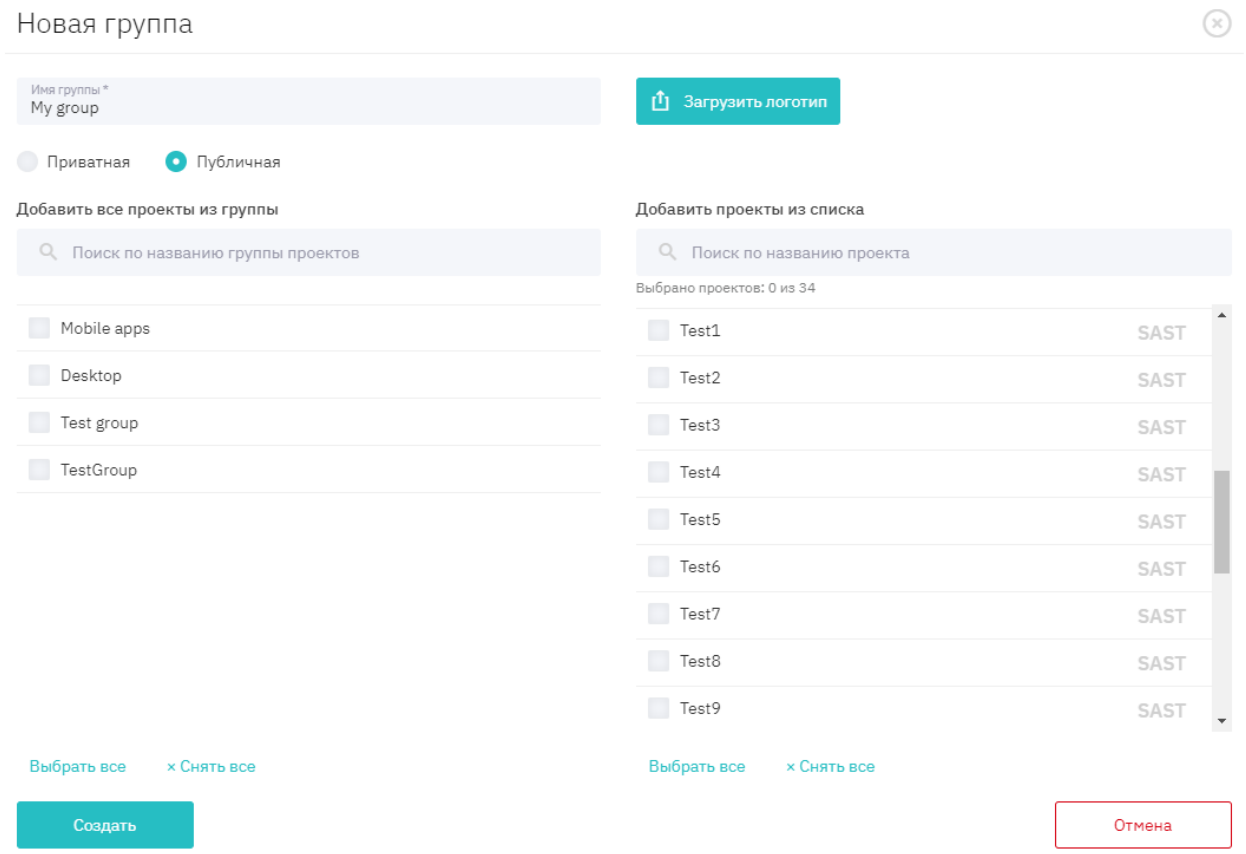


Рис. 5.2: Создание группы проектов

Для групп доступна сводная информация и аналитика. Работать с группами можно в разделе **Группы проектов**.

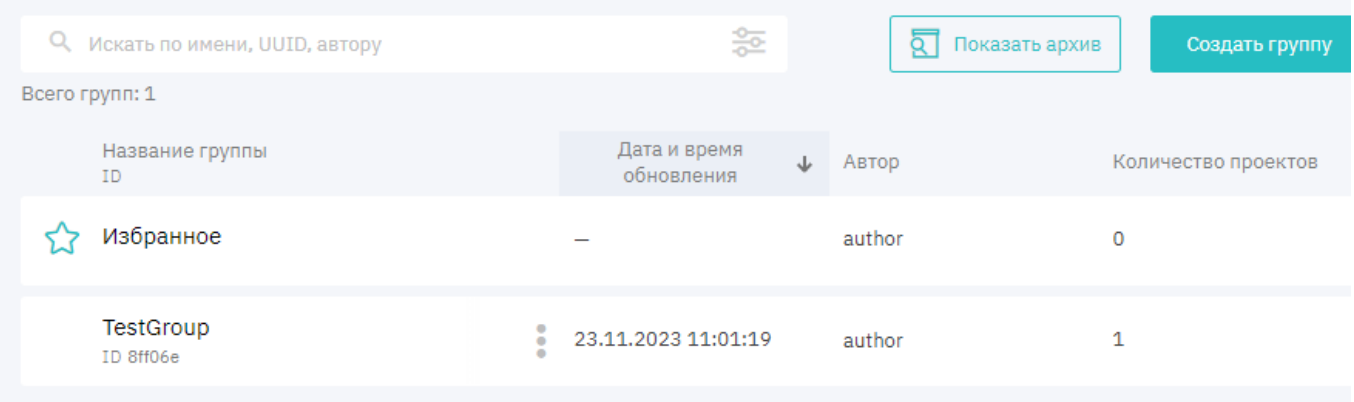


Рис. 5.3: Группы проектов

Список групп представлен в виде таблицы. При клике на название группы произойдет переход к ее странице.

На вкладке **Обзор** представлена агрегированная статистика по сканированиям всех проектов, входящих в группу. Графики позволяют анализировать динамику показателей в разрезе выбранного временного периода. Настроить отображение можно по суммарным значением или средним. В таблице **Статистика группы** отражены ключевые метрики по результатам сканирований.

Управление составом группы осуществляется во вкладке **Проекты**. Здесь отображается перечень всех проектов, включённых в группу.

При удалении проектам из группы он остается доступен в системе.

В разделе **Управление группой** на вкладке **Настройки** доступны операции редактирования параметров группы, её архивирования и восстановления и полного удаления.

В разделе **Jira** можно настроить конфигурацию **Jira**, которая может быть привязана сразу ко всем проектам, которые входят в состав группы.

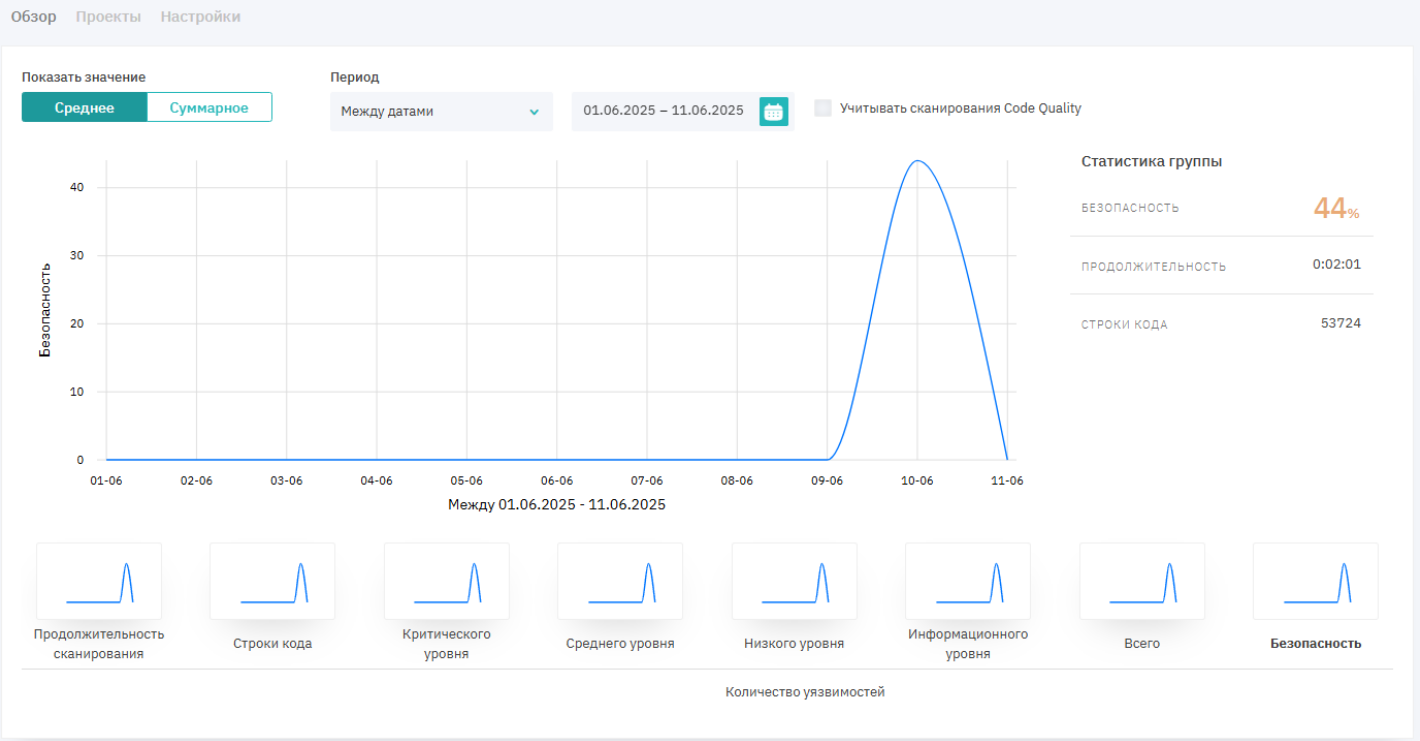


Рис. 5.4: Группа проектов: Обзор

5.3.2 Аналитика

Во вкладке **Результаты** отображаются графики, позволяющие оценить динамику ключевых метрик по группам проектов за определенные промежутки времени. По умолчанию в списке выбрана только группа «Все проекты», дополнительные группы можно добавить для отображения в списке над таблицей. Для каждой из выбранных групп отображаются следующие показатели:

- общее количество сканирований (в режиме суммирования)
- средняя оценка безопасности за период (в режиме средних значений);
- средняя продолжительность сканирований;
- объём исходного кода в строках;
- количество выявленных уязвимостей с разбивкой по уровням критичности.
- количество сканирований (при отображении суммарного значения) или рейтинг (при отображении среднего значения);
- продолжительность сканирований;
- количество строк кода;
- количество уязвимостей (с учётом уровня критичности).

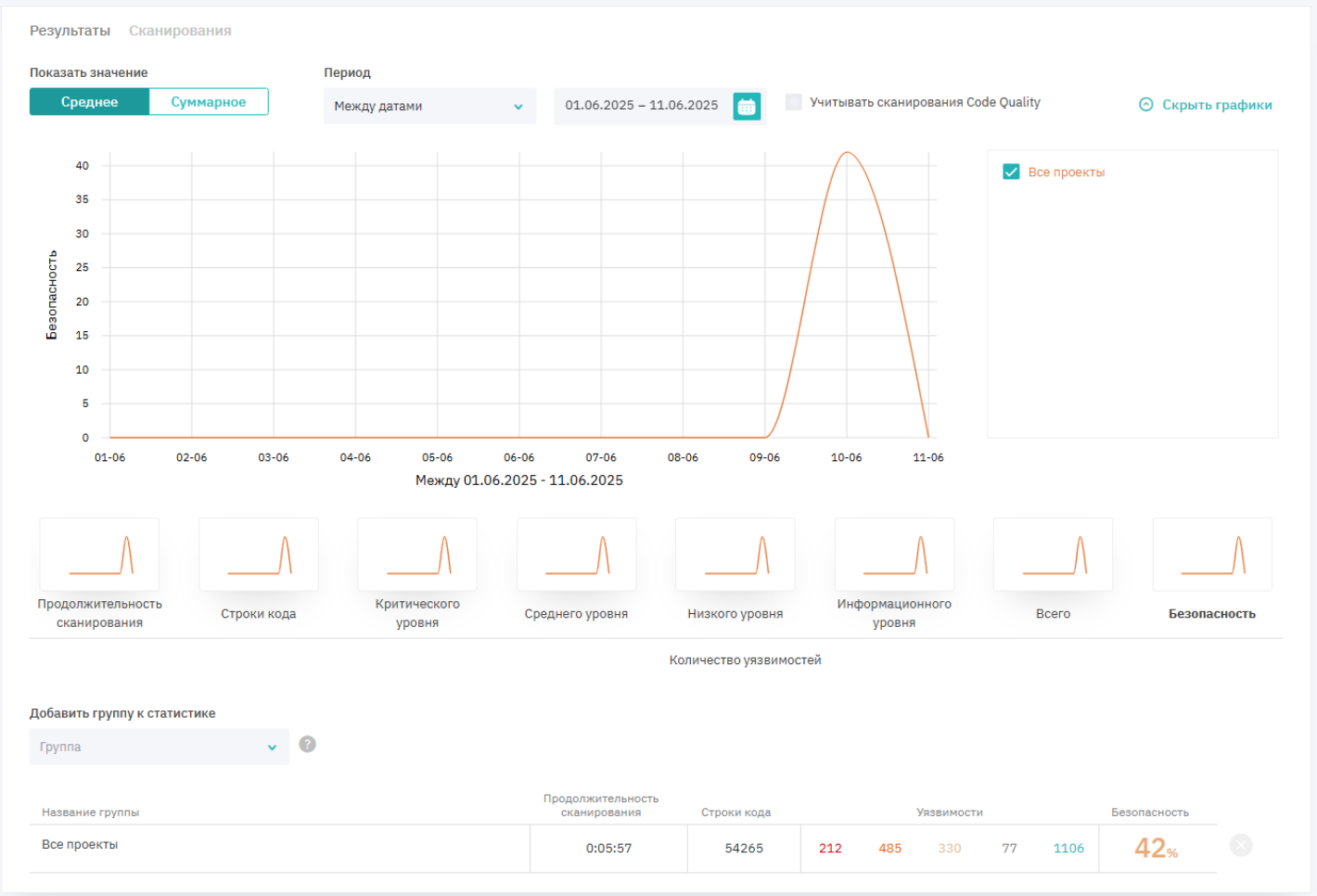


Рис. 5.5: Результаты сканирований по группам проектов

Вкладка **Сканирования** предоставляет таблицу с агрегированными данными по группам: общее количество проектов, число выполненных сканирований и распределение по статусам. Нажатие на крестик уберет группу из таблицы.

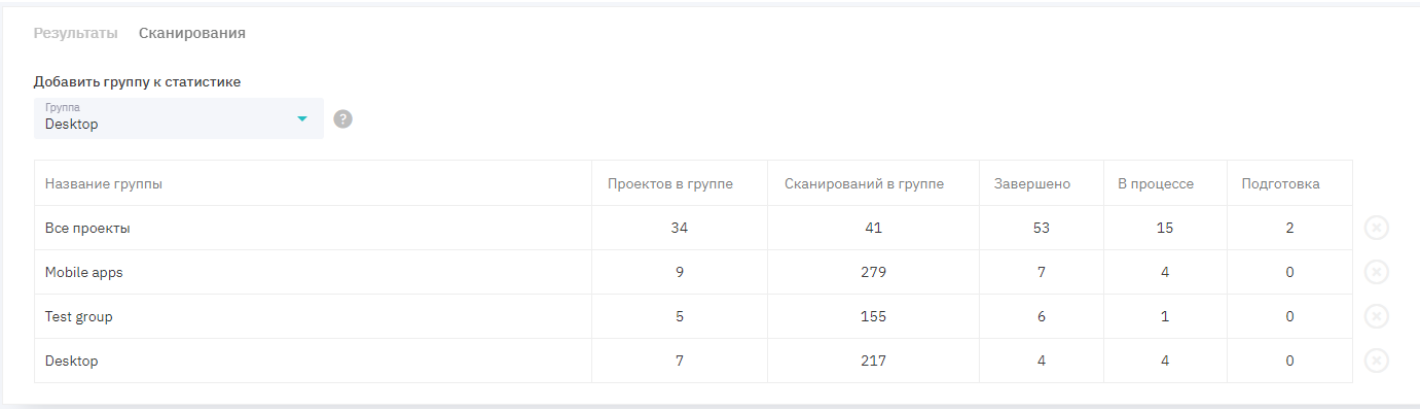


Рис. 5.6: Просмотр статистики по группам проектов

5.4 О продукте

В разделе меню **О продукте** можно получить основную информацию о возможностях DerScanner, скачать руководство пользователя, отключить или возобновить отображение подсказок по работе с DerScanner в интерфейсе.

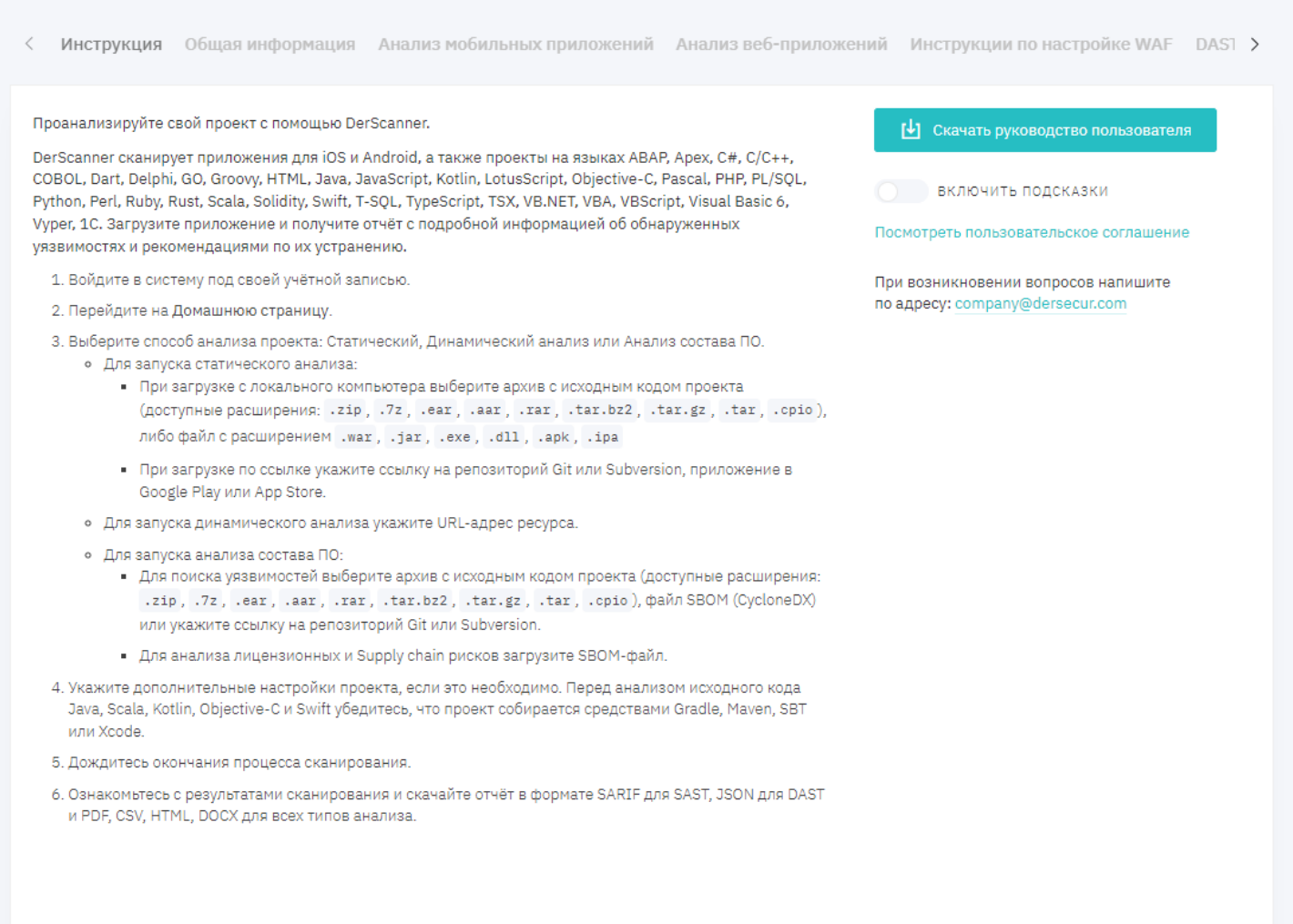


Рис. 5.7: О продукте

5.5 Аккаунт

Раздел предназначен для настройки учетной записи пользователя DerScanner.

5.5.1 Профиль

В разделе отображается основная информация о состоянии учетной записи (ограничения лицензии или ограничения, полученные от администратора) и распоожены настройки оповещений.

Если администратор DerScanner указал e-mail в профиле, тогда можно настроить получение писем по результатам сканирований, запущенных вами.

При необходимости в состав оповещения можно включить краткую сводку с результатами

сканирования, статистику по используемым языкам программирования (для результатов SAST и CQ), а также текст ошибки в случае завершения сканирования с ошибкой.

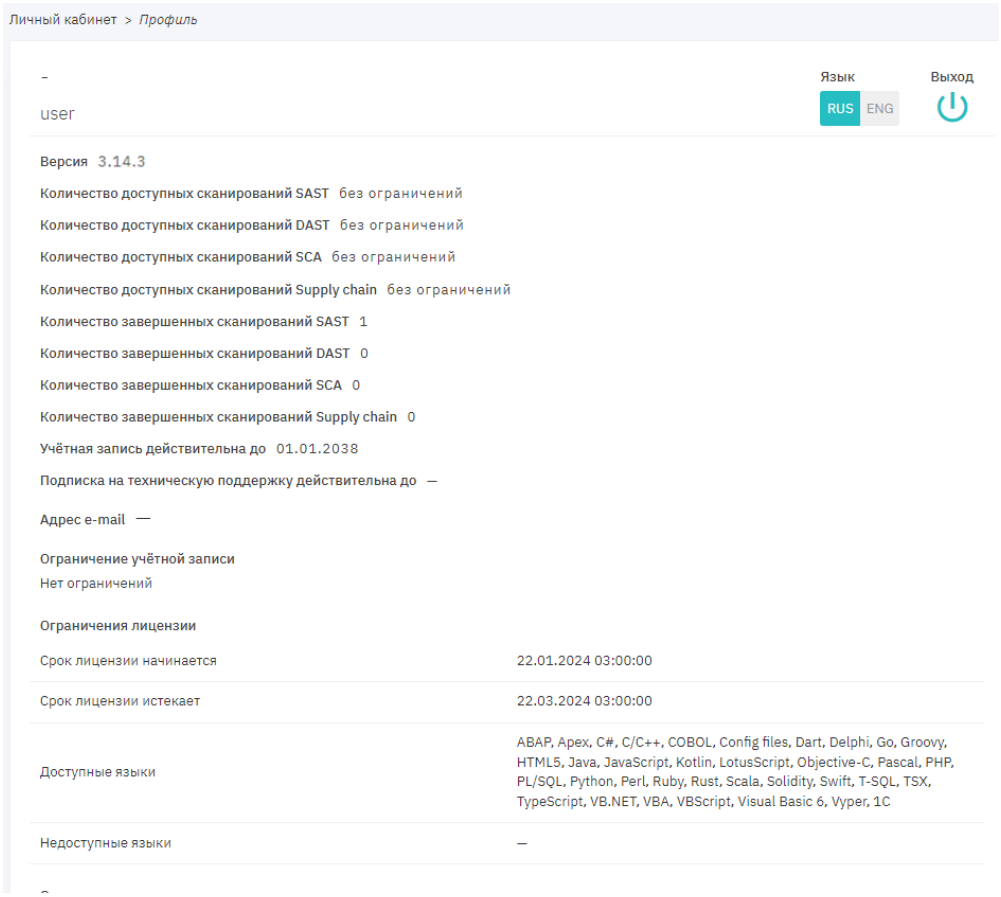


Рис. 5.8: Профиль

5.5.2 Доступы

5.5.2.1 Токен и пароль

Токен авторизации используется для аутентификации пользователя при выполнении операций в DerScanner в CLT, API и других интеграциях.

Для получения токена авторизации API:

1. Нажмите кнопку **Создать токен**.
2. Введите пароль чётной записи в DerScanner.
3. Укажите время действия токена.
4. Нажмите **Получить активный токен**.
5. Скопируйте и сохраните полученный токен.

Активные токены отображаются в таблице на странице.

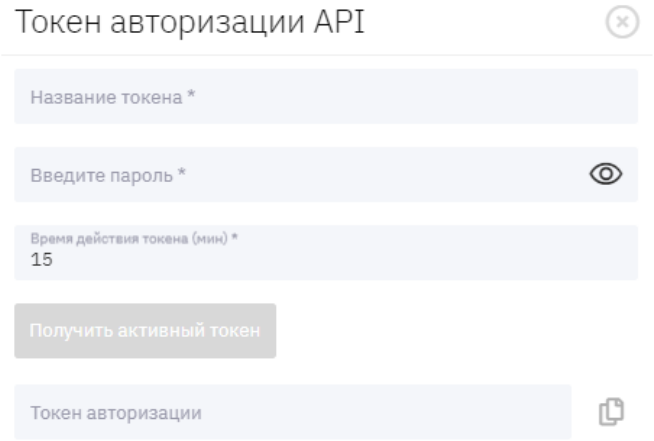


Рис. 5.9: Токен авторизации API

Для смены пароля от учетной записи на вкладке **Пароль** выполните следующее:

1. Укажите текущий пароль.
2. Укажите новый пароль в соотвествующих полях.
3. Нажмите **Сохранить**.

5.5.2.2 Таск-менеджер

Для интеграции в DerScanner требуется указать данные учетной записи Jira. Задачи будут создаваться от имени привязанного аккаунта.

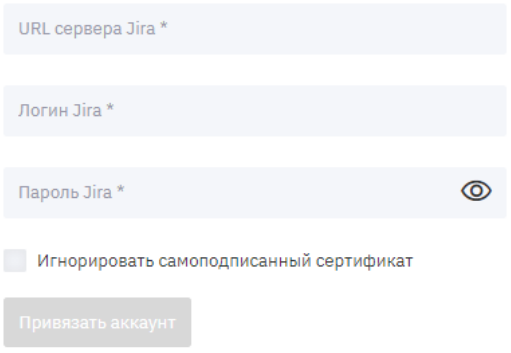


Рис. 5.10: Привязка аккаунта Jira

5.5.2.3 Репозитории

В DerScanner можно сохранить учетные данные для репозиториев или данные авторизации для анализа DAST для того, чтобы в дальнейшем использовать их в сканированиях.

Сохранять можно:

- логин и пароль (сохраненные данные можно использовать для анализа из репозиториев или для авторизации при анализе DAST);
- токен;
- SSH-ключ (может быть загружен в виде файла или введен вручную);
- заголовки;
- форма авторизации;
- данные для авторизации NTLM.

5.5.3 Шаблоны

Набор настроек запуска сканирования для анализа SAST и настройки экспорта отчетов сканирований SAST, CQ, DAST и SCA можно сохранять для того, чтобы переиспользовать в дальнейшем.

5.5.3.1 Сканирование

Шаблоны настроек анализа позволяют не настраивать конфигурацию сканирования вручную перед каждым запуском, а заранее создавать комбинации настроек под определенные проекты, чтобы в дальнейшем их переиспользовать. 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.

Чтобы создать шаблон настроек:

1. Нажмите на кнопку **Создать шаблон**.
2. Укажите название шаблона и его описание.
3. При необходимости выберите **Шаблон настроек по умолчанию**, чтобы применять выбранный шаблон по умолчанию при создании нового проекта.
4. Выберите **Публичный шаблон** (доступен для использования всем пользователям системы) или **Приватный шаблон** (будет доступен только автору шаблона и администратору).
5. Укажите настройки, которые должен содержать шаблон.
6. Нажмите **Сохранить**.

Для редактирования шаблона настроек:

1. Нажмите на название шаблона в таблице.
2. Внесите изменения.
3. Нажмите **Сохранить**.

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.

5.5.3.2 Отчёт

Шаблоны экспорта отчета позволяют не настраивать конфигурацию экспорта каждый раз вручную.

Чтобы создать шаблон:

1. Нажмите на кнопку **Создать шаблон**.
2. Задайте название шаблона и его описание.
3. При необходимости выберите **Шаблон настроек по умолчанию**, чтобы использовать выбранный шаблон каждый раз при экспорте.
4. Выберите **Публичный шаблон** (доступен для использования всем пользователям системы) или **Приватный шаблон** (будет доступен только автору шаблона и администратору).
5. Выберите настройки, которые должен содержать шаблон.
6. Нажмите **Сохранить**.

Для редактирования шаблона экспорта:

1. Нажмите на название шаблона в списке.
2. Внесите изменения.
3. Нажмите **Сохранить**.

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

5.5.4 Перенос данных

В разделе **Перенос данных** можно выбирать сканирования, проекты, группы проектов, которые необходимо экспортировать на другую установку DerScanner.

Для экспорта результатов:

1. Установите положение переключателя на **Проекты** или **Сканирования** в зависимости от требований.
2. Выберите сущности, которые необходимо экспортировать.
3. Укажите пароль для архива с сущностями и нажмите **Экспорт**.

Для **проектов** в архив будет добавлена следующая информация:

- проект;
- настройки проекта (настройки приватного репозитория экспортируются в настройках проекта (ключи в разделе **Аккаунт** > **Доступы** не сохраняются), права пользователей не экспортируются);
- исходный код (для проектов SAST) и подробные результаты (включая журнал событий);
- сканирования (архивированные сканирования не экспортируются);
- информация о сканированиях с файлом приложения, если при запуске была выбрана настройка **Сохранить загруженный файл**, и журналами событий (автор сканирования не сохраняется);
- правила (включая пользовательские), которые были нарушены уязвимостями данных проектов (только для проектов SAST);
- базы SCA, которые были нарушены уязвимостями данных проектов (только для проектов SCA).

Для **сканирований** в архив будет добавлена следующая информация:

- сканирование;
- исходный код (для проектов SAST) и подробные результаты (включая журнал событий);
- информация о сканировании с файлом приложения, если при запуске была выбрана настройка **Сохранить загруженный файл**, и журналами событий (автор сканирования не сохраняется);
- правила (включая пользовательские), которые были нарушены уязвимостями данных проектов (только для проектов SAST);
- базы SCA, которые были нарушены уязвимостями данных проектов (только для проектов SCA).

Для **групп проектов** будет перенесена группа и все проекты, которые в нее входят.

Обратите внимание: Правила SAST и базы SCA переносятся таким образом, что их результаты отображаются только в импортированных проектах. Выбрать их для сканирования новых проектов не получится.

Чтобы импортировать проекты на новую установку DerScanner:

1. В разделе **Перенос данных** > **Импорт** загрузите архив, экспортированный с другой установки DerScanner.
2. Введите пароль от архива.
3. Выберите нужные файлы для импорта на новую установку.
4. Если в импортируемом архиве находятся отдельные сканирования, выберите проект на новой установке, в который будут помещены импортируемые сканирования.
5. Нажмите **Импорт**.

06 Интеграции DerScanner

Плагины DerScanner и инструмент командной строки (clt.jar) находятся на сервере с установленным DerScanner в директории /opt/DerScanner/app/plugins для Linux и C:\DerScanner\plugins для Windows.

6.1 Jira

DerScanner использует для интеграции **Jira REST API v2**.

Для начала работы с Jira выполните слудующие шаги:

1. Привяжите профиль Jira к профилю в системе на странице **Аккаунт** > **Доступы** > **Таск-менеджер**;
2. Создайте конфигурацию проекта:
   1. Откройте проект и перейдите в раздел **Таск-менеджер** на вкладке **Настройки**.
   2. Нажмите **Добавить конфигурацию**, чтобы привязать проект к проекту в Jira.
   3. Выберите проект Jira и заполните остальные требуемые поля.
3. Создайте задачу:
   1. Перейдите в раздел **Подробные результаты** внутри проекта.
   2. Выберите уязвимость.
   3. Откройте вкладку **Таск-менеджер**.
   4. Нажмите **Создать задачу**.
   5. В появившейся форме укажите необходимые параметры: проект Jira, описание, приоритет, исполнителя и другие поля (по необходимости). Можно также задать родительскую задачу и компоненты.
   6. Нажмите **Создать**.

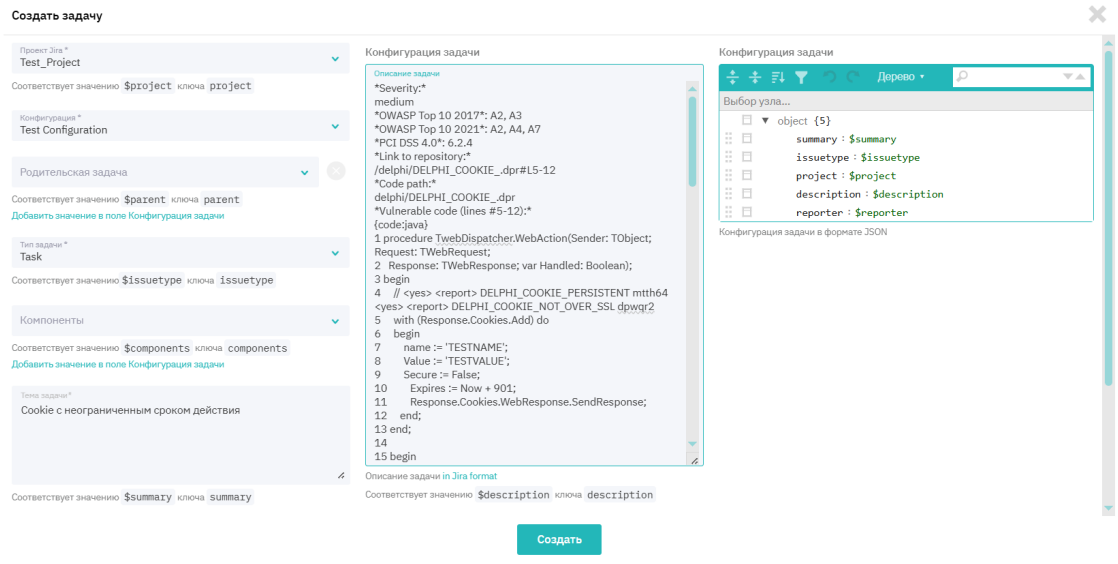


Рис. 6.1: Параметры задачи в Jira

Чтобы открыть задачу в Jira, просто кликните по её названию в списке. Для удаления задачи из интерфейса DerScanner используйте кнопку с иконкой удаления.

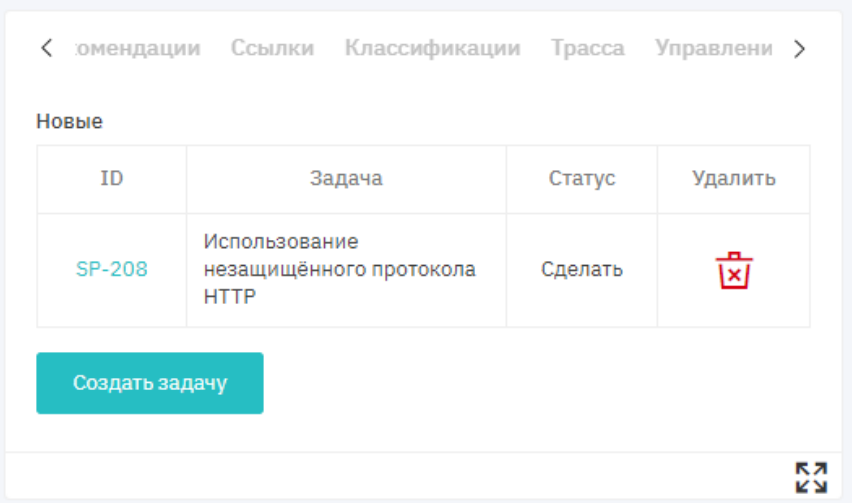


Рис. 6.2: Список задач в Jira

Если необходимо создать задачи сразу для нескольких уязвимостей, отметьте нужные элементы и выберите **Создать задачи** в меню управления уязвимостью.

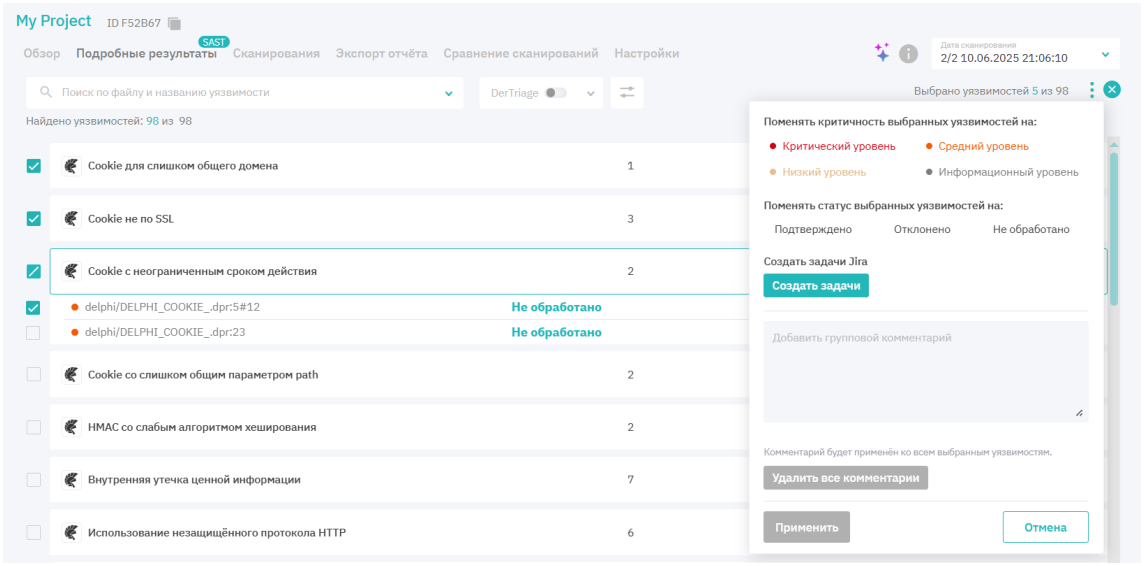


Рис. 6.3: Создать задачи для нескольких уязвимостей

6.2 Gitlab CI

Из коробки DerScanner поддерживает интеграцию с GitLab через webhook, чего будет достаточно для автоматического запуска сканирований при определённых условиях (подробнее в разделе VCS хостинги).

Если вы хотите встроить DerScanner в ваш GitLab CI в качестве Quality Gate, потребуется более детальная настройка с использованием скриптов. Примеры скриптов поставляются вместе с дистрибутивом системы.

6.2.1 Пример настройки с использованием API

В рамках тестового пайплайна (gitlab-api.zip) проект анализируется компонентами SAST, DAST и SCA.

Stages:

- **build** — сборка проекта;
- **test** — запуск unit-тестов;
- **deploy** — развертывание сервиса в каком-либо окружении;
- **run** — развертывание и запуск сервиса для DAST анализа;
- **SAST** — SAST анализ;
- **DAST** — DAST анализ;
- **OSA** — SCA анализ.

Module's jobs:

- **SAST:start_scan** — запуск SAST сканирования;
- **DAST:create_project** — создание DAST проекта;
- **DAST:start_scan** — запуск DAST сканирования;
- **OSA:create_project** — создание SCA проекта;
- **OSA:create_sbom** — генерация SBOM-файла;
- **OSA:start_scan_sbom** — запуск SCA сканирования SBOM-файла;
- **OSA:start_scan_archive** — запуск SCA сканирования архива.

Common jobs:

- **status_scan** — проверка статуса сканирования с ограничением в виде job's timeout;
- **check_rate_scan** — проверка результатов сканирования на соответствие определенным условиям;
- **get_summary** — генерация кратких результатов сканирования;
- **report** — генерация отчетов в форматах PDF и CSV.

Список переменных, которые редактируются пользователем:

- **API_V1_URL** — REST URL приложения, например: http://10.0.2.6/app/api/v1;
- **TOKEN** — токен аутентификации;
- **LANGUAGES** — языки сканирования для SAST анализа;
- **TARGET_URL** — URL приложения для DAST анализа;
- **OSA_SCAN_TYPE** — выбор типа сканирования для SCA: архив (`archive`) или SBOM-файл (`sbom`);
- **MIN_RATING_SAST** — минимальная оценка рейтинга безопасности для SAST;
- **MIN_RATING_DAST** — минимальная оценка рейтинга безопасности для DAST;
- **MIN_RATING_OSA** — минимальная оценка рейтинга безопасности для SCA;
- **MAX_CRIT_SAST** — максимально допустимое количество критических уязвимостей в коде для SAST;
- **MAX_CRIT_DAST** — максимально допустимое количество критических уязвимостей в коде для DAST;
- **MAX_CRIT_OSA** — максимально допустимое количество критических уязвимостей для SCA.

Обратите внимание: чувствительные переменные необходимо хранить либо в специализированных сервисах, либо в скрытых переменных проекта/группы.

Замечания:

Для запуска сканирования SCA с использованием архива или SBOM-файла необходимо указать значение переменной **OSA_SCAN_TYPE** `sbom` или `archive` соотвественно.

Для запуска всех DAST сканирований (для SCA аналогично) в одном проекте удалите **DAST:create_project** и задайте следующие глобальные переменные:

- **DAST_PROJECT_ID**;
- **DAST_PROJECT_UUID**.

Для SAST необходимо задать только **SAST_PROJECT_UUID**. Значения **PROJECT_ID** и **PROJECT_UUID** для каждого типа анализа можно получить из API или интерфейса после первого запуска сканирования либо после создания пустого проекта.

6.2.2 Пример настройки с использованием CLT

В рамках тестового пайплайна (gitlab-clt.zip) проект анализируется компонентами SAST, DAST и SCA.

Stages:

- **build** — сборка проекта;
- **test** — запуск unit-тестов;
- **deploy** — развертывание сервиса в каком-либо окружении;
- **run** — развертывание и запуск сервиса для DAST анализа;
- **SAST** — SAST анализ;
- **DAST** — DAST анализ;
- **OSA** — SCA анализ.

Module's jobs:

- **SAST:start_scan** — запуск SAST сканирования;
- **DAST:create_project** — создание DAST проекта;
- **DAST:start_scan** — запуск DAST сканирования;
- **OSA:create_project** — создание SCA проекта;
- **OSA:create_sbom** — генерация SBOM-файла;
- **OSA:start_scan_sbom** — запуск SCA сканирования SBOM-файла;
- **OSA:start_scan_archive** — запуск SCA сканирования архива.

Common jobs:

- **status_scan** — проверка статуса сканирования с ограничением в виде job's timeout;
- **check_rate_scan** — проверка результатов сканирования на соответствие определенным условиям;
- **report** — генерация отчетов в форматах PDF и CSV.

Список переменных, которые редактируются пользователем:

- **API_V1_URL** — REST URL приложения, например: http://10.0.2.6/app/api/v1;
- **TOKEN** — токен аутентификации;
- **CLT_URL** — ссылка для загрузки clt.jar;
- **LANGUAGES** — языки сканирования для SAST анализа;
- **TARGET_URL** — URL приложения для DAST анализа;
- **OSA_SCAN_TYPE** — выбор типа сканирования для SCA: архив (`archive`) или SBOM-файл (`sbom`);

- **MIN_RATING_SAST** — минимальная оценка рейтинга безопасности для SAST;
- **MIN_RATING_DAST** — минимальная оценка рейтинга безопасности для DAST;
- **MIN_RATING_OSA** — минимальная оценка рейтинга безопасности для SCA.

Обратите внимание: чувствительные переменные необходимо хранить либо в специализированных сервисах, либо в скрытых переменных проекта/группы.

Замечания:

Для запуска сканирования SCA с использованием архива или SBOM-файла необходимо указать значение переменной **OSA_SCAN_TYPE** `sbom` или `archive` соотвественно.

Для запуска всех DAST сканирований (для SCA аналогично) в одном проекте удалите **DAST:create_project** и задайте следующие глобальные переменные:

- **DAST_PROJECT_ID**;
- **DAST_PROJECT_UUID**.

Для SAST сканирования необходимо в **SAST:start_scan** добавить аргумент **-uuid ${SAST_PROJECT_UUID}** для CLT. Значения **PROJECT_ID** и **PROJECT_UUID** для каждого типа анализа можно получить из CLT, API или интерфейса после первого запуска сканирования либо после создания пустого проекта.

6.3 DefectDojo

Для интеграции с DefectDojo плагин не требуется. В DefectDojo можно загрузить отчёт из DerScanner в формате SARIF.

Для загрузки отчёта в DefectDojo:

1. Перейдите на страницу проекта в DefectDojo.
2. На вкладке **Findings** нажмите **Import Scan Results**.
3. Заполните все необходимые поля. В поле **Scan Type** выберите **SARIF**.
4. В DerScanner скачайте отчёт в формате SARIF.
5. В DefectDojo нажмите **Import**.

6.4 Jenkins

DerScanner поддерживает **Jenkins 2.164** и более поздние версии.

6.4.1 Установка расширения в Jenkins

1. Откройте страницу **Manage Jenkins** (**http://<адрес машины, на которой установлен Jenkins>:8080/jenkins/manage**).

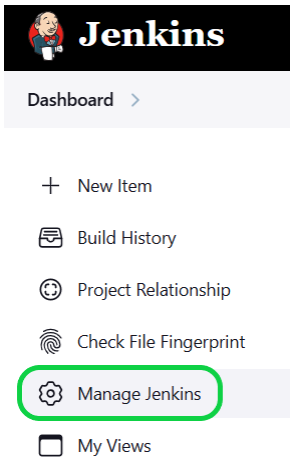


Рис. 6.4: Jenkins: Настройки Jenkins

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

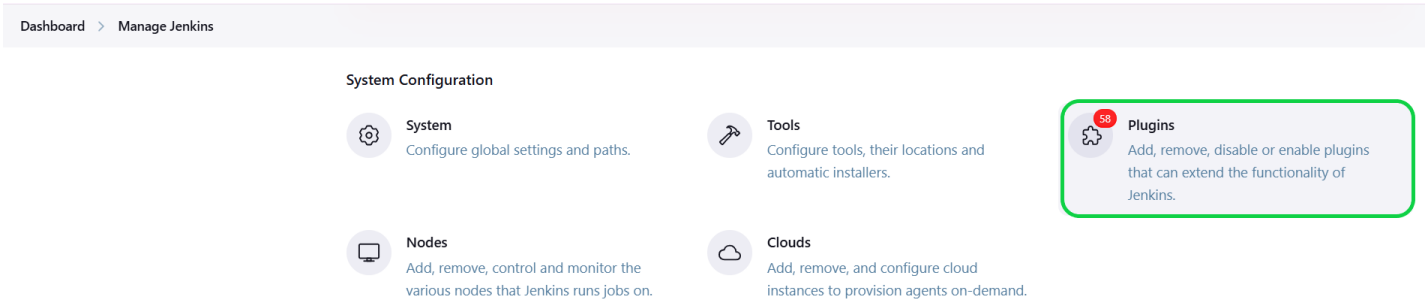


Рис. 6.5: Jenkins: Управление плагинами

3. Выберите вкладку **Advanced settings**.

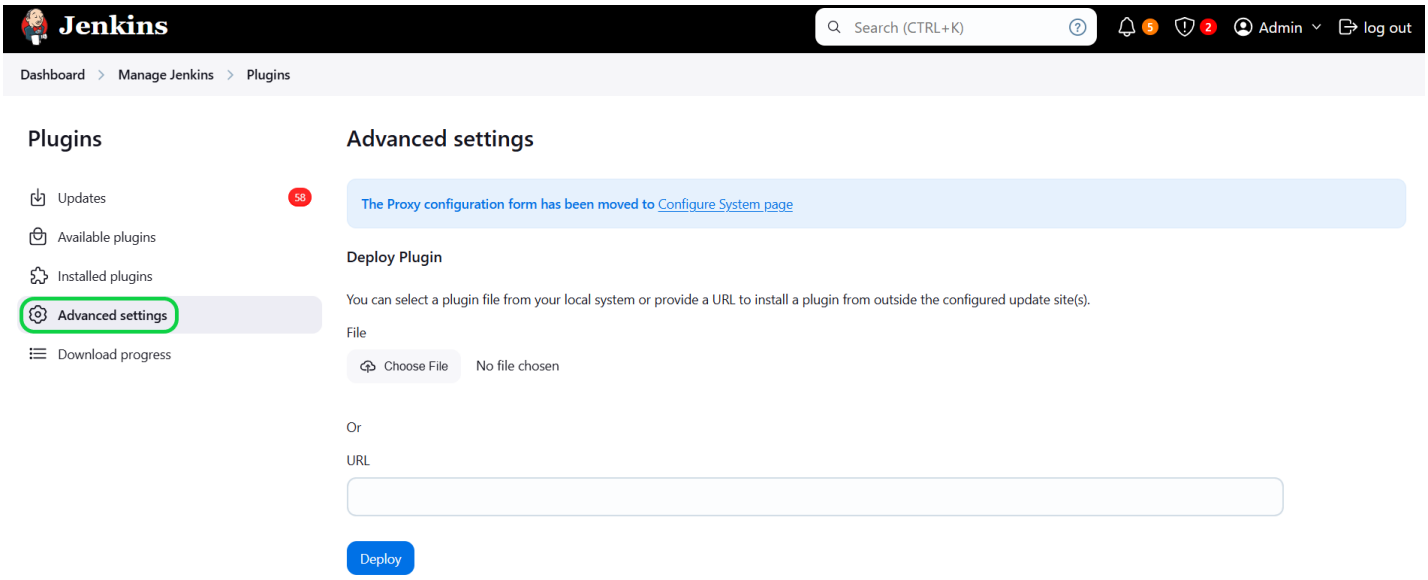


Рис. 6.6: Jenkins: Дополнительные настройки

4. В пункте **Deploy Plugin** (**Загрузить плагин**) нажмите кнопку **Choose File** (**Выберите файл**).

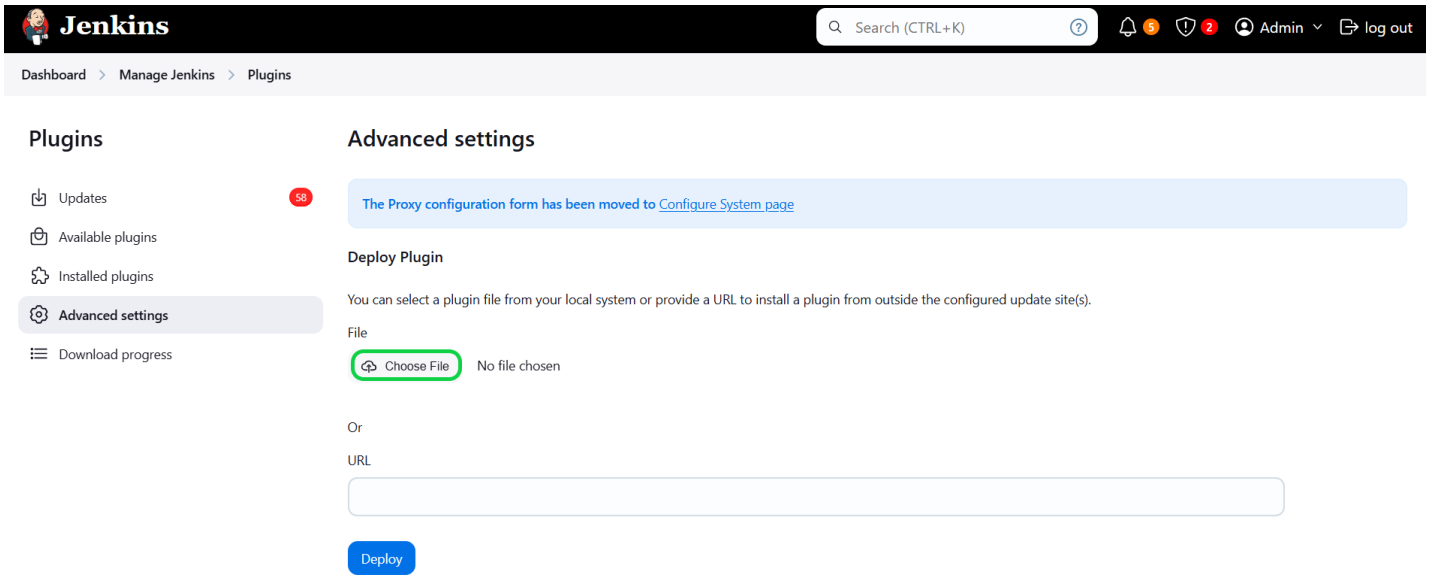


Рис. 6.7: Jenkins: Загрузить плагин

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

6.4.2 Общие настройки расширения

1. Перейдите на страницу настроек Jenkinks **Manage Jenkins** (**http://<адрес машины, на которой установлен Jenkins>:8080/jenkins/manage**).
2. Выберите **System** в разделе **System Configuration** (**Конфигурация системы**).
3. Найдите пункт **DerScanner Analysis** (**Анализ DerScanner**) и заполните поля:
   - введите DerScanner URL, например **http://<адрес машины, на которой установлен Jenkins>** адрес машины, на которой установлен DerScanner;
   - введите адрес REST API для DerScanner, например **http://<адрес машины, на которой установлен DerScanner>/app/api/v1/**;
   - введите токен пользователя, полученный в разделе **Токен и пароль** в интерфейсе DerScanner;
   - 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.

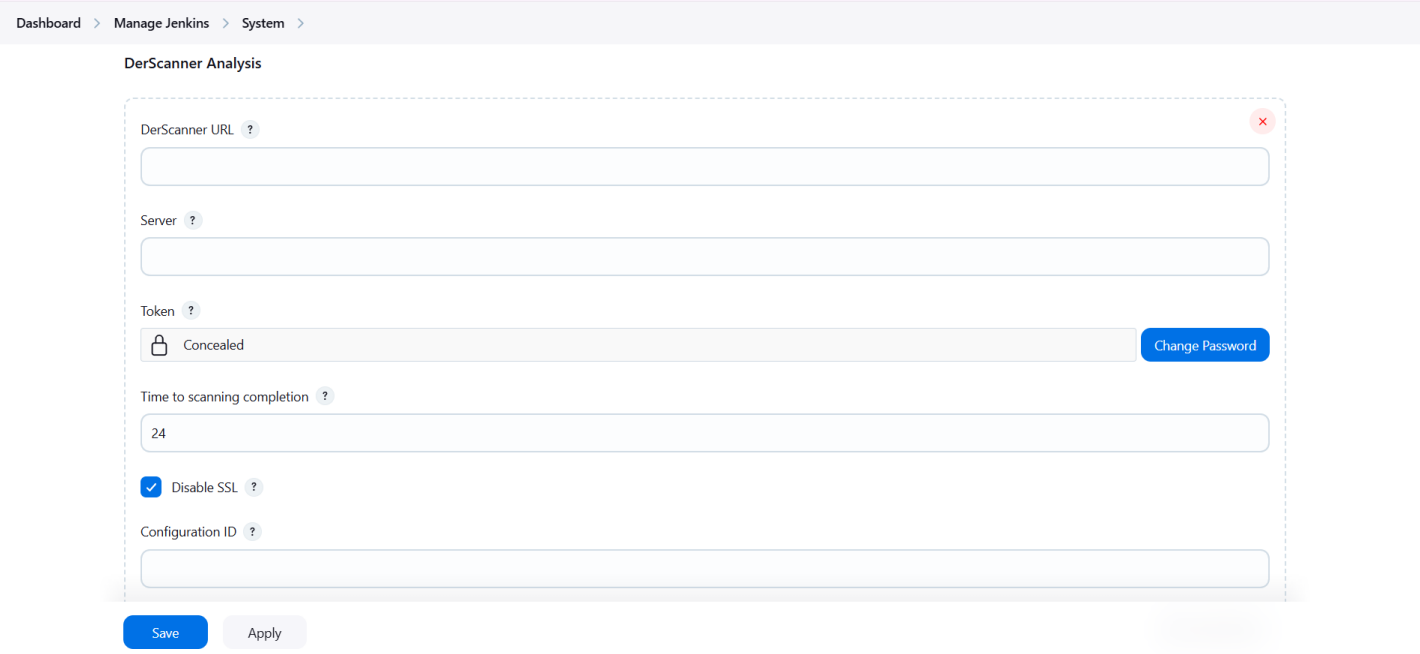


Рис. 6.8: Jenkins: Статический анализ DerScanner

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

6.4.3 Конфигурация анализа и отчёта

Конфигурацию анализа и отчёта можно осуществить двумя способами:

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

6.4.3.1 Задача со свободной конфигурацией

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

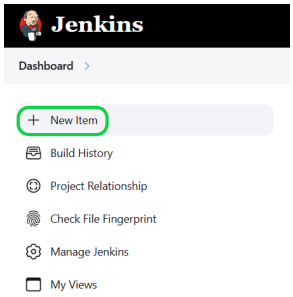


Рис. 6.9: Jenkins: Новый Item

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

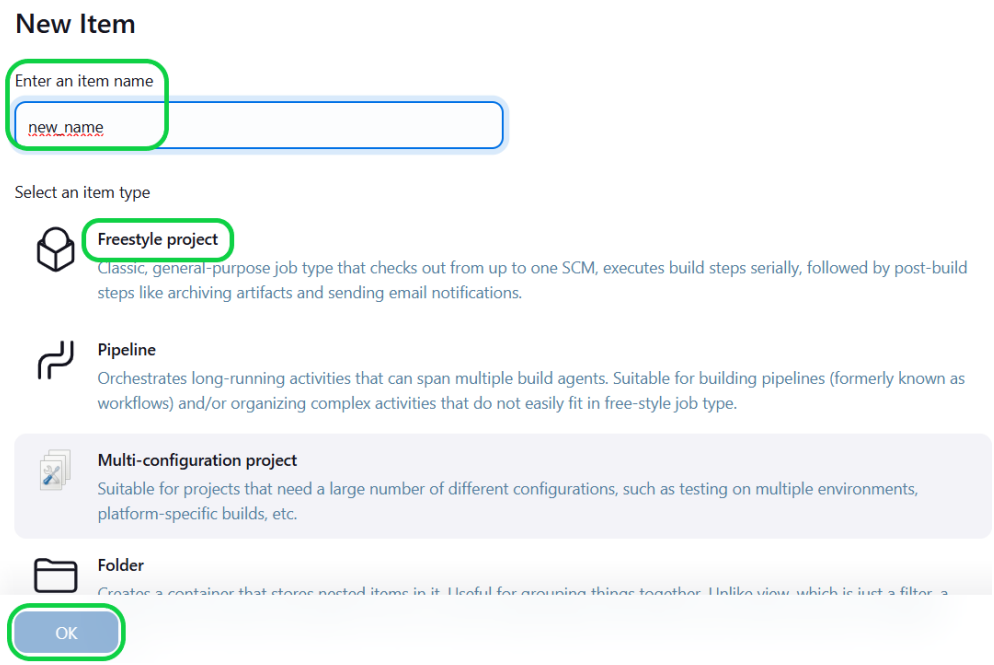


Рис. 6.10: Jenkins: Задача со свободной конфигурацией

4. Нажмите **OK**.
5. В пункте **Build Steps** (**Шаги сборки**) в поле **Add build step** (**Добавить шаг сборки**) выберите **Execute DerScanner Analysis** (для примера используется статический анализ).
6. В раскрывшемся меню:
   1. Введите UUID проекта.

      Project UUID это идентификатор существующего проекта DerScanner. Скопировать **Project UUID** можно в боковом меню проекта. Пример: 9feefaf0-4c17-47fe-b1f5-f7f64d4da722. Настройка конфигураций ниже перенастроит проект DerScanner, если текстовое поле **Project UUID** не пустое. Если текстовое поле **Project UUID** пустое, плагин Jenkins создаст новый проект в DerScanner.
   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 **DerScanner Static Analysis report export in PDF** (**Экспорт pdf-отчёта статического анализа DerScanner**) и сделайте настройку (см. раздел Экспорт отчёта).
8. Нажмите **Save** (**Сохранить**).

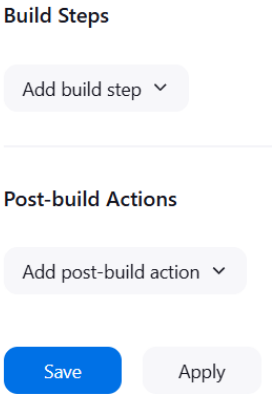


Рис. 6.11: Jenkins: Послесборочные операции

6.4.3.2 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**.

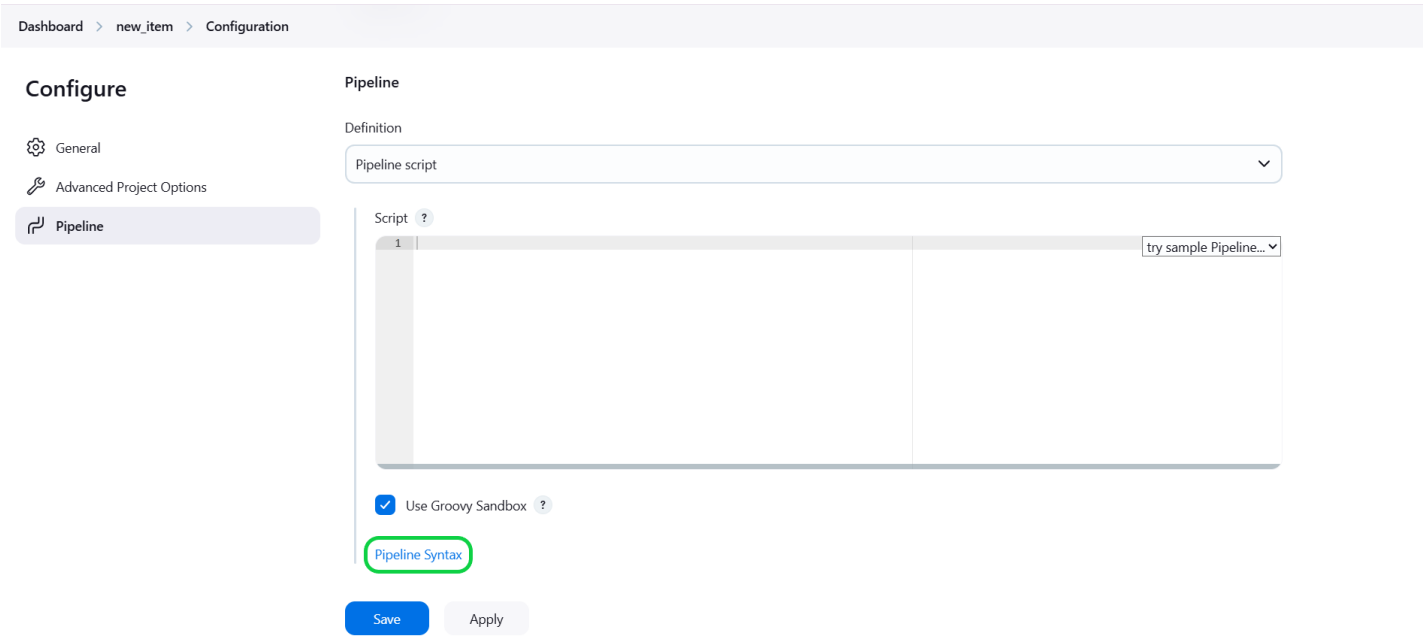


Рис. 6.12: Jenkins: Pipeline Syntax

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 DerScanner Static Analysis** (**Выполнить статический анализ DerScanner**).

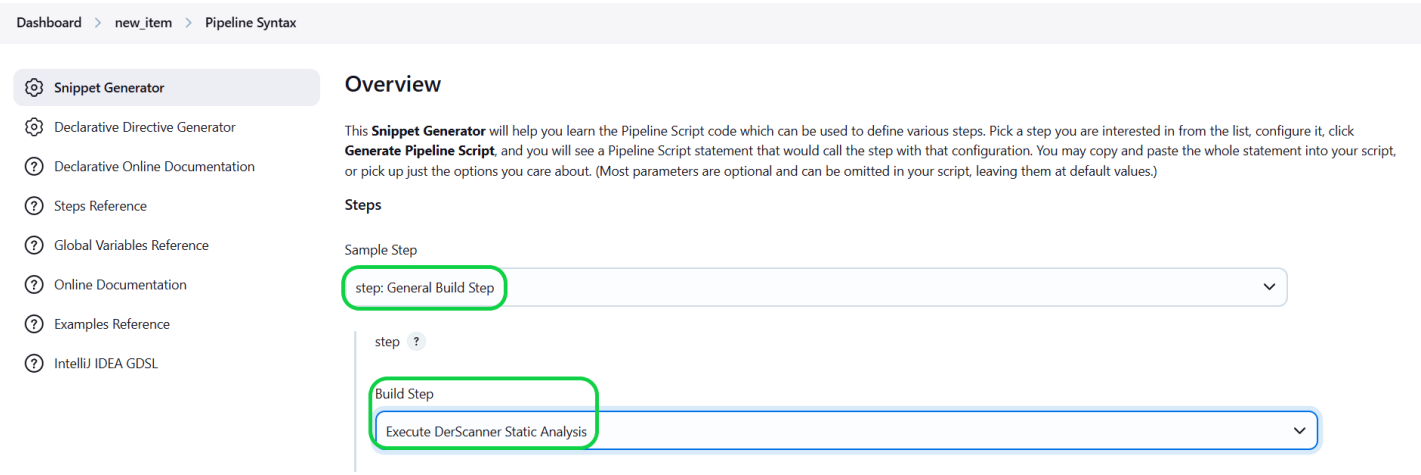


Рис. 6.13: Jenkins: Сборка

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

      Project UUID это идентификатор существующего проекта DerScanner. Скопировать **Project UUID** можно в боковом меню проекта. Настройка конфигураций ниже перенастроит проект DerScanner, если текстовое поле **Project UUID** не пустое. Если текстовое поле **Project UUID** пустое, плагин Jenkins создаст новый проект в DerScanner.
   2. Выберите сервер.
   3. Настройте остальные конфигурации сканирования (если потребуется). Подробнее о конфигурациях сканирования см. Настройки.
9. Нажмите **Generate Pipeline Script**.

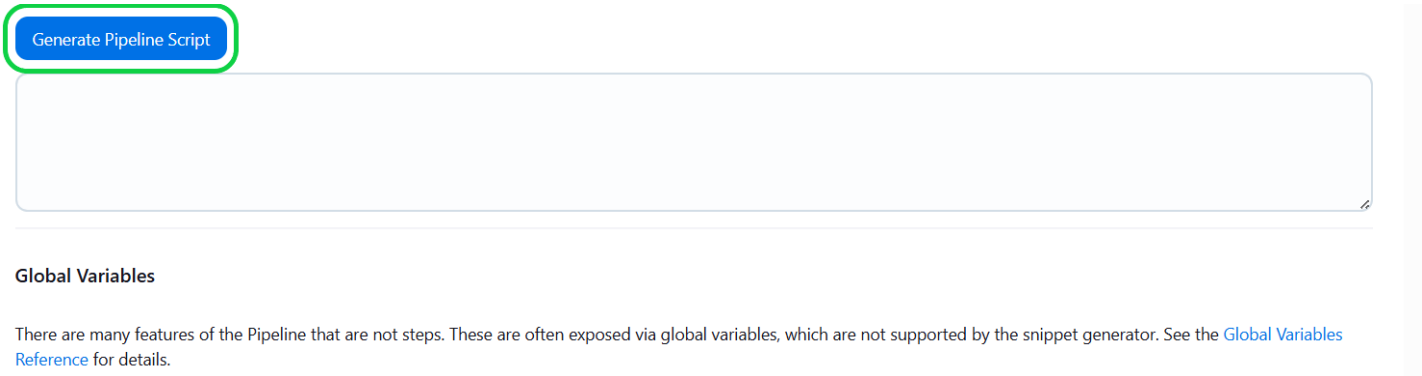


Рис. 6.14: Jenkins: Генерация скрипта Pipeline

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 **DerScanner Static Analysis report export** (**Экспорт отчёта статического анализа DerScanner**).

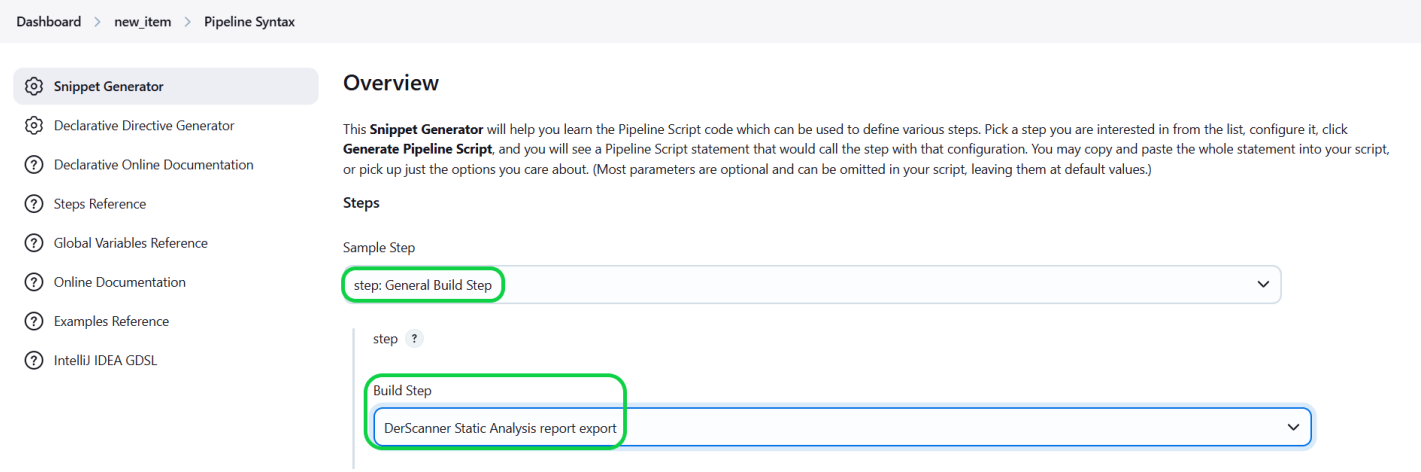


Рис. 6.15: Jenkins: Экспорт отчёта

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**).

6.4.4 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.

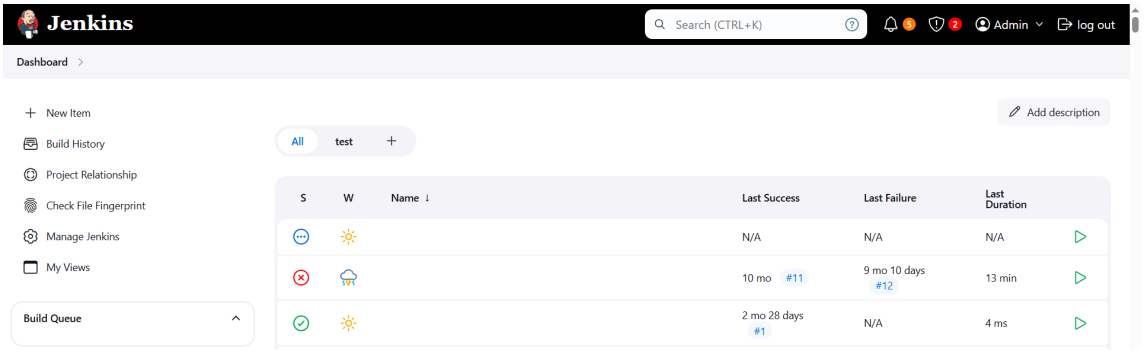


Рис. 6.16: Jenkins: Выбор Item'а

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

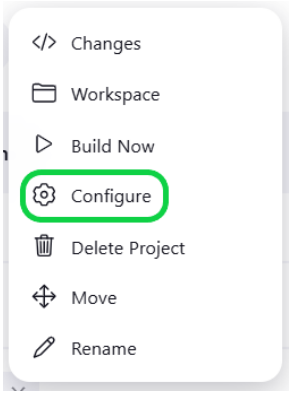


Рис. 6.17: Jenkins: Выбор настроек

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

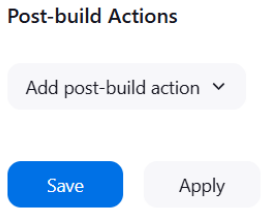


Рис. 6.18: Jenkins: Выбор послесборочных операций

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 DerScanner;
  - **SCAN_ID** UUID сканирования в DerScanner;
  - **SCAN_URL** ссылка на результаты сканирования в DerScanner;
  - **SERVER** адрес REST API;
  - **SERVER_UUID** идентификатор сервера, присваиваемый при настройке глобальной конфигурации;
  - **PDF_URL** адрес PDF-отчёта;
  - **SCORE** рейтинг проекта в DerScanner;
  - **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**.

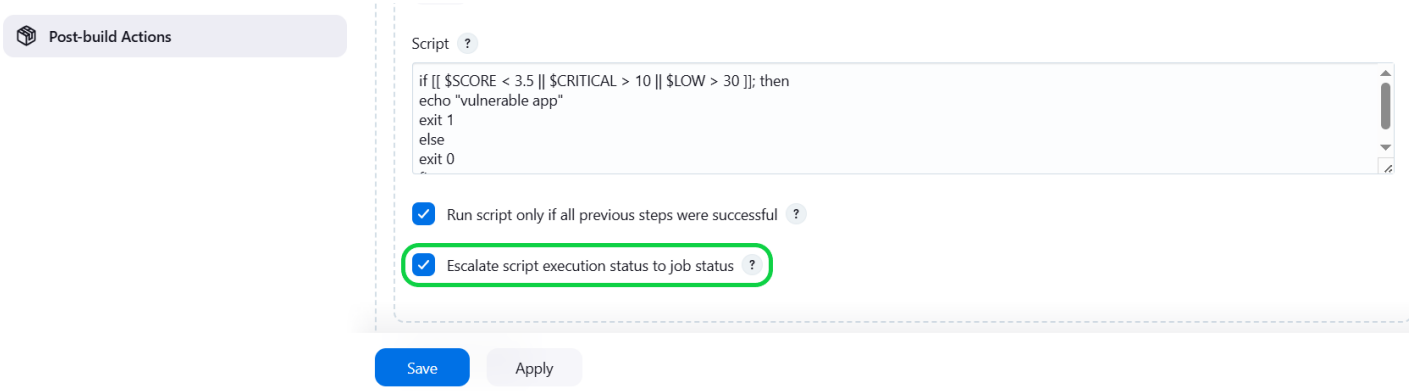


Рис. 6.19: Jenkins: Сохранение послесборочных операций

8. Нажмите **Save** (**Сохранить**).

6.4.5 Результаты и отчёт

Для того чтобы просмотреть результаты сканирования:

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

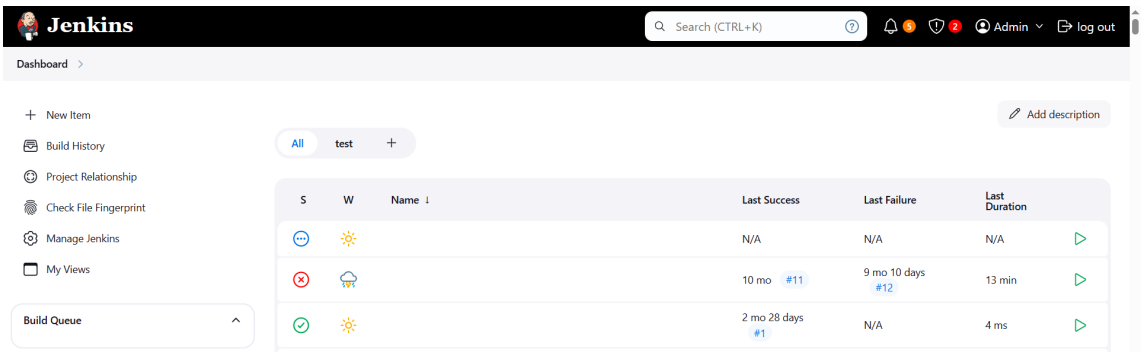


Рис. 6.20: Jenkins: Выбор Item’а

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

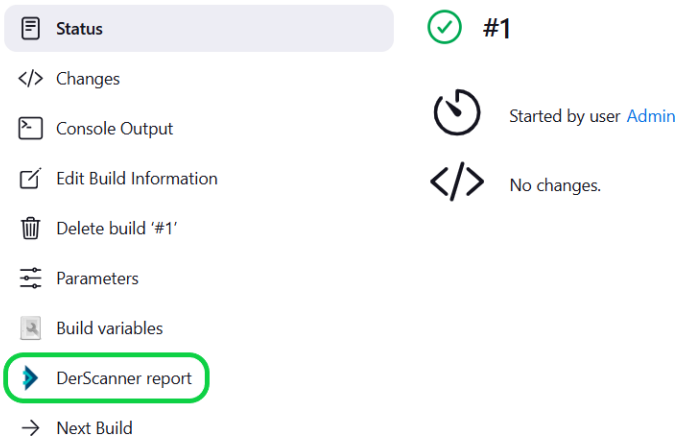


Рис. 6.21: Jenkins: отчёт

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

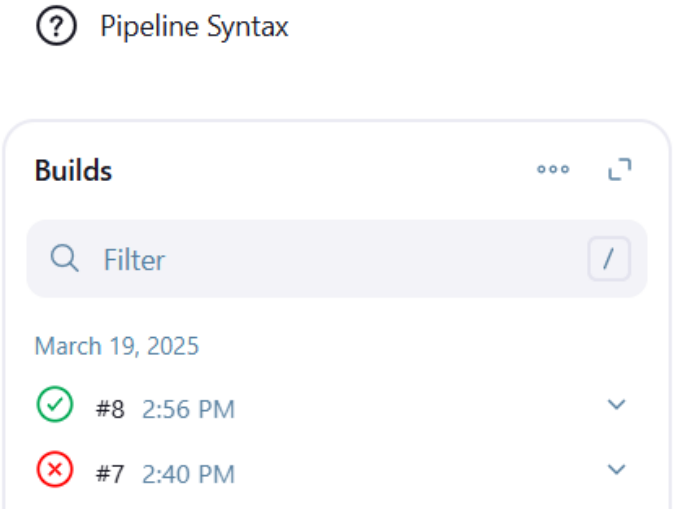


Рис. 6.22: Jenkins: Номер сборки

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

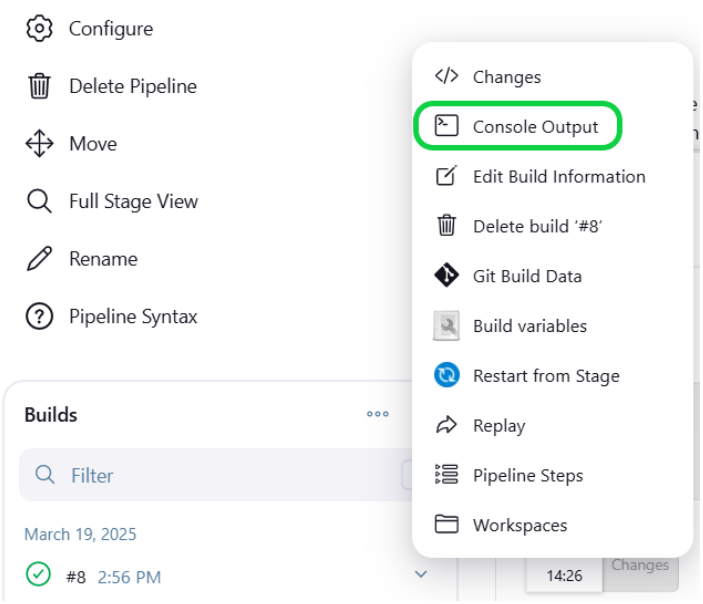


Рис. 6.23: Jenkins: Консоль

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

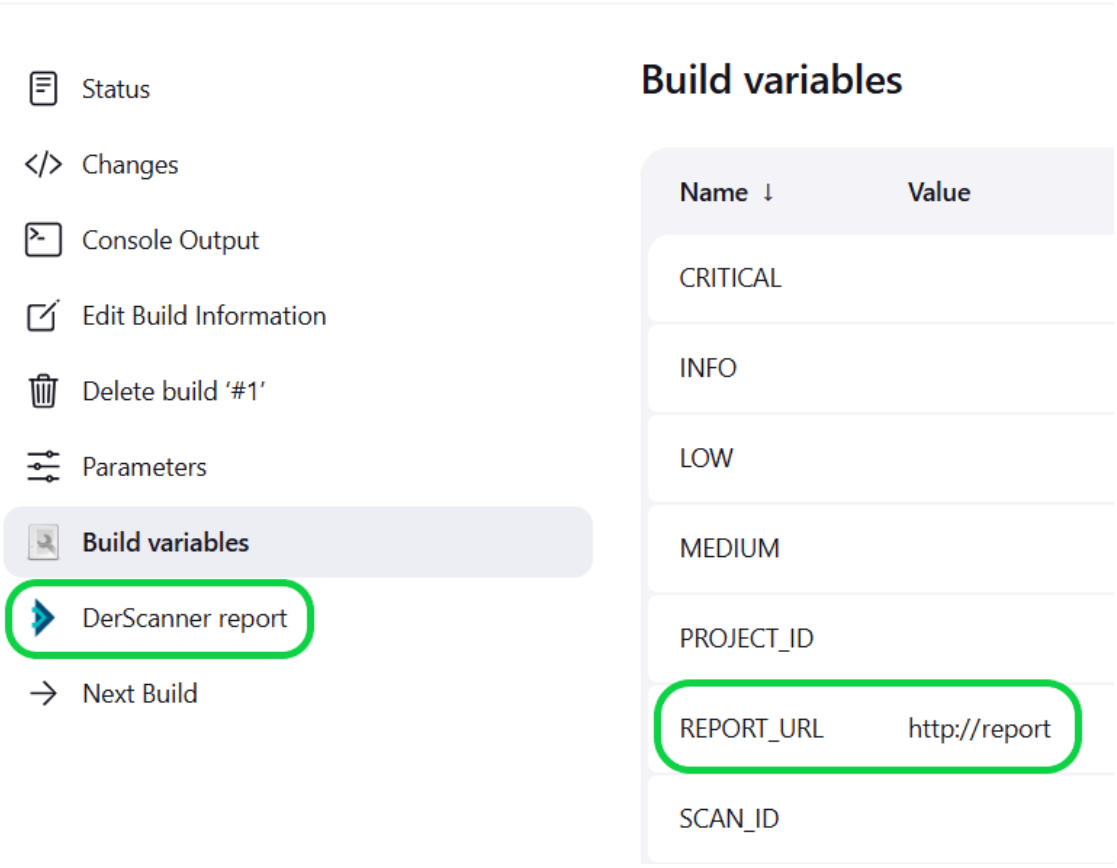


Рис. 6.24: Jenkins: Результаты сканирования

6.5 Azure DevOps Server

DerScanner поддерживает интеграцию с **Team Foundation Server 2018**, **Azure Devops Server 2019**, **Azure Devops Server 2020** и **Azure DevOps Services**.

6.5.1 Установка расширения

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

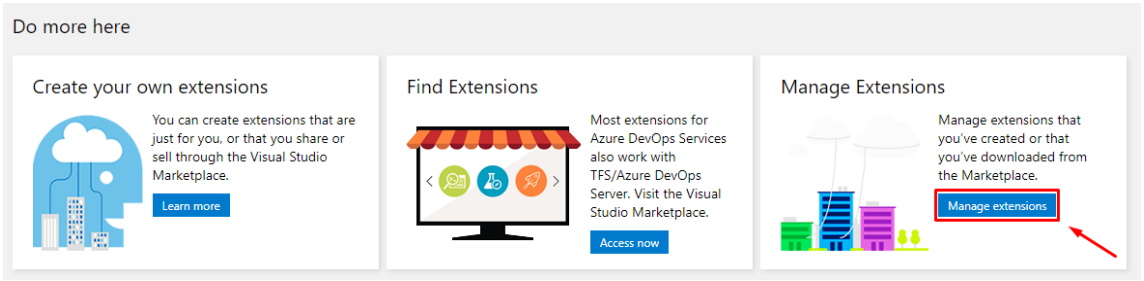


Рис. 6.25: Azure DevOps Server: Управление расширениями

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



Рис. 6.26: Azure DevOps Server: Отправить новое расширение

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

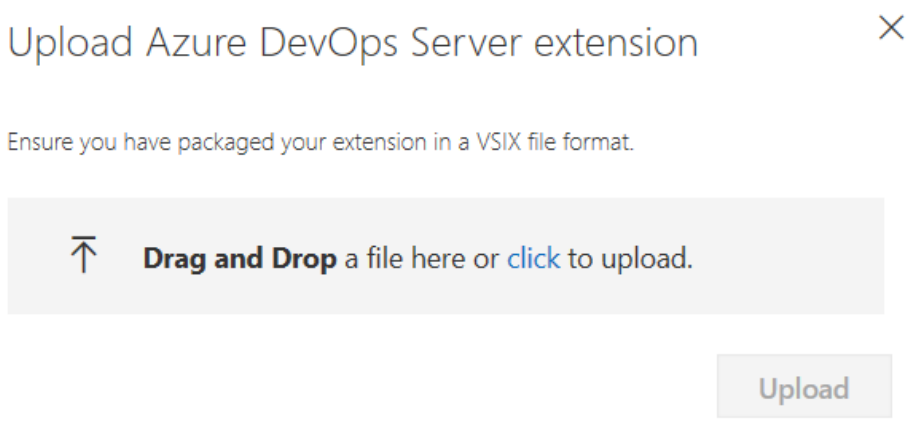


Рис. 6.27: Azure DevOps Server: Отправка

6. Выберите из списка расширение DerScanner.

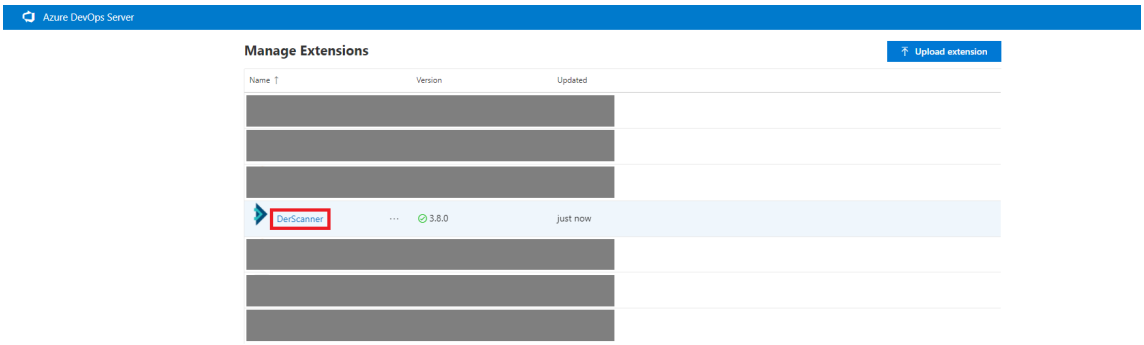


Рис. 6.28: Azure DevOps Server: Выбор расширения

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

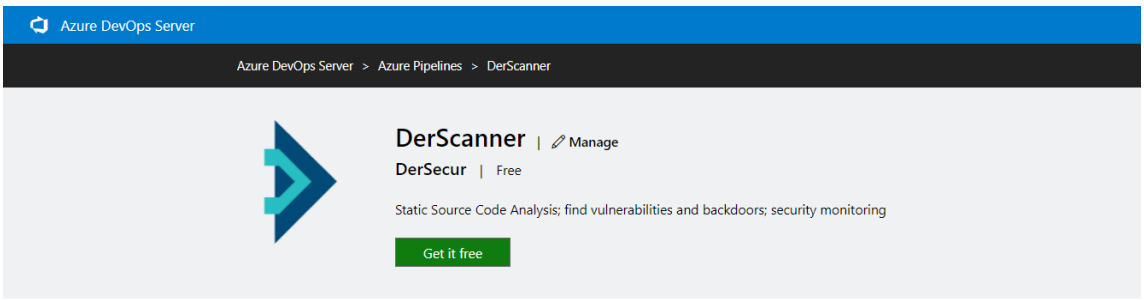


Рис. 6.29: Azure DevOps Server: Установка

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.

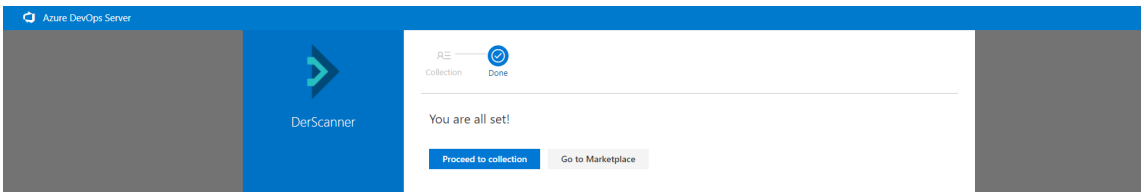


Рис. 6.30: Azure DevOps Server: Подтверждение

6.5.2 Добавление шага для сборки в Azure DevOps Server

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

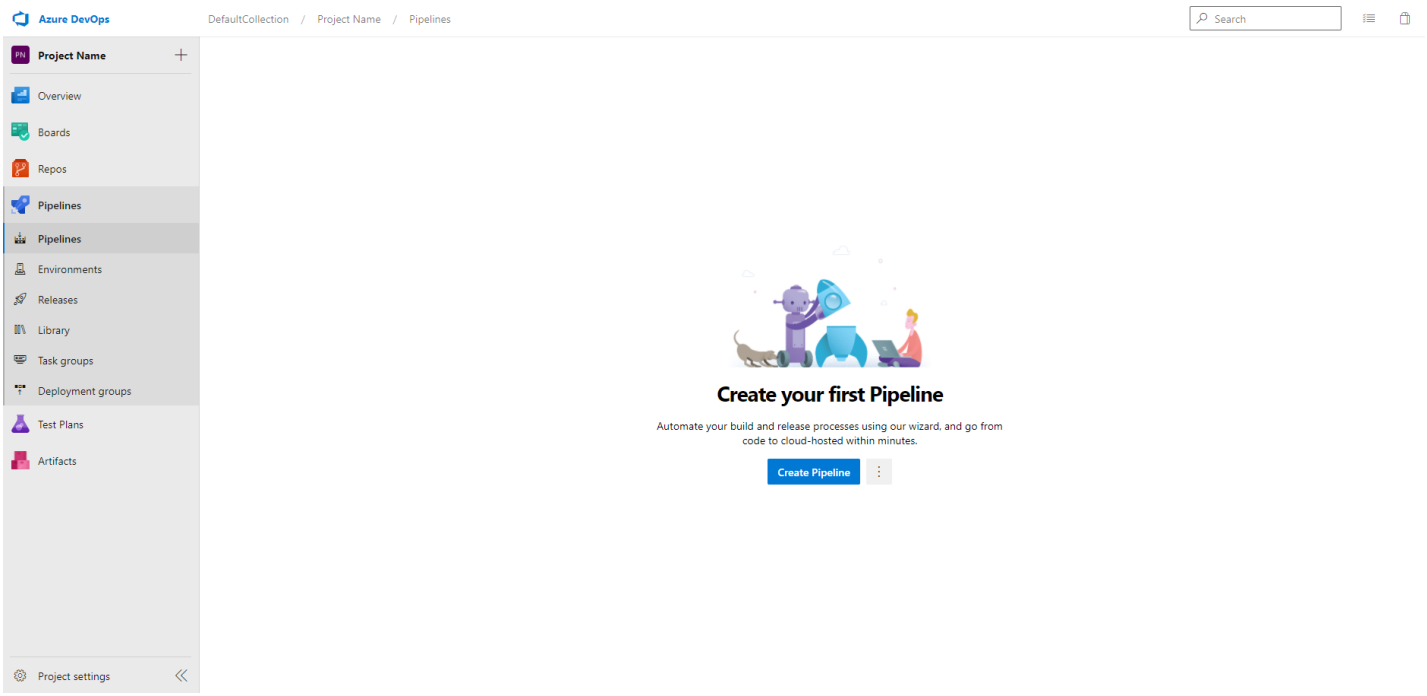


Рис. 6.31: Azure DevOps Server: Сборка и выпуск

3. Нажмите на три точки и нажмите **Edit** (**Изменить**) для имеющегося определения сборки или создайте новое, нажав **New Pipeline** (**Создать сборку**). Если вы выбрали новую сборку, выберите **Classic Redactor** (**классический редактор**) в расположении кода. Далее нажмите на **Продолжить** > в выборе шаблона укажите **Empty Job** (**пустое задание**).
4. Нажмите **Add Task** (**Добавить задачу**).

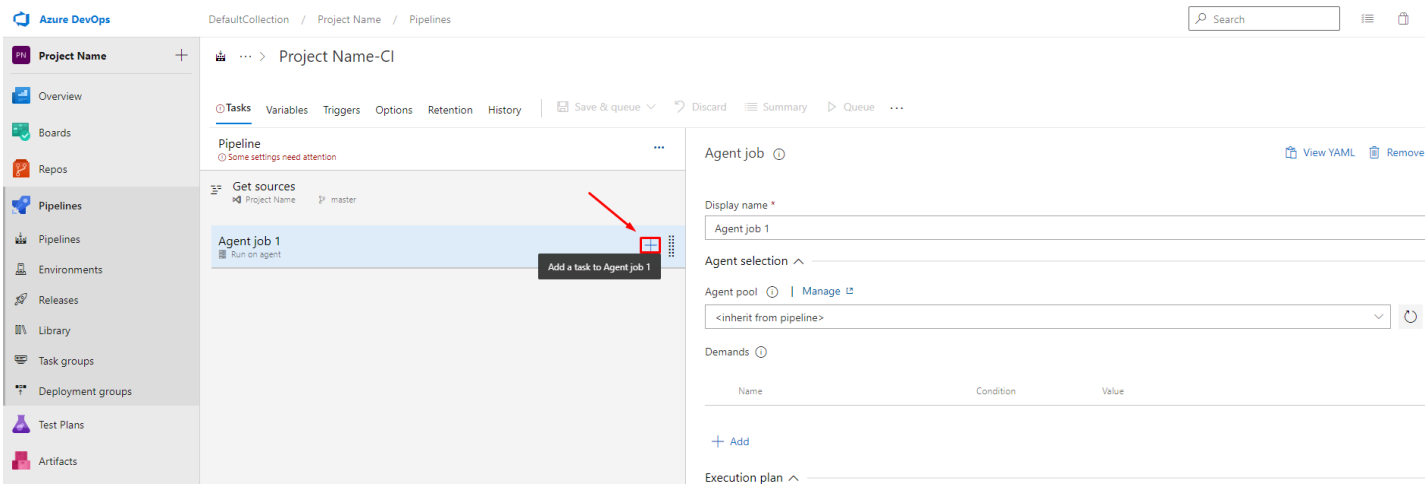


Рис. 6.32: Azure DevOps Server: Добавление задачи

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

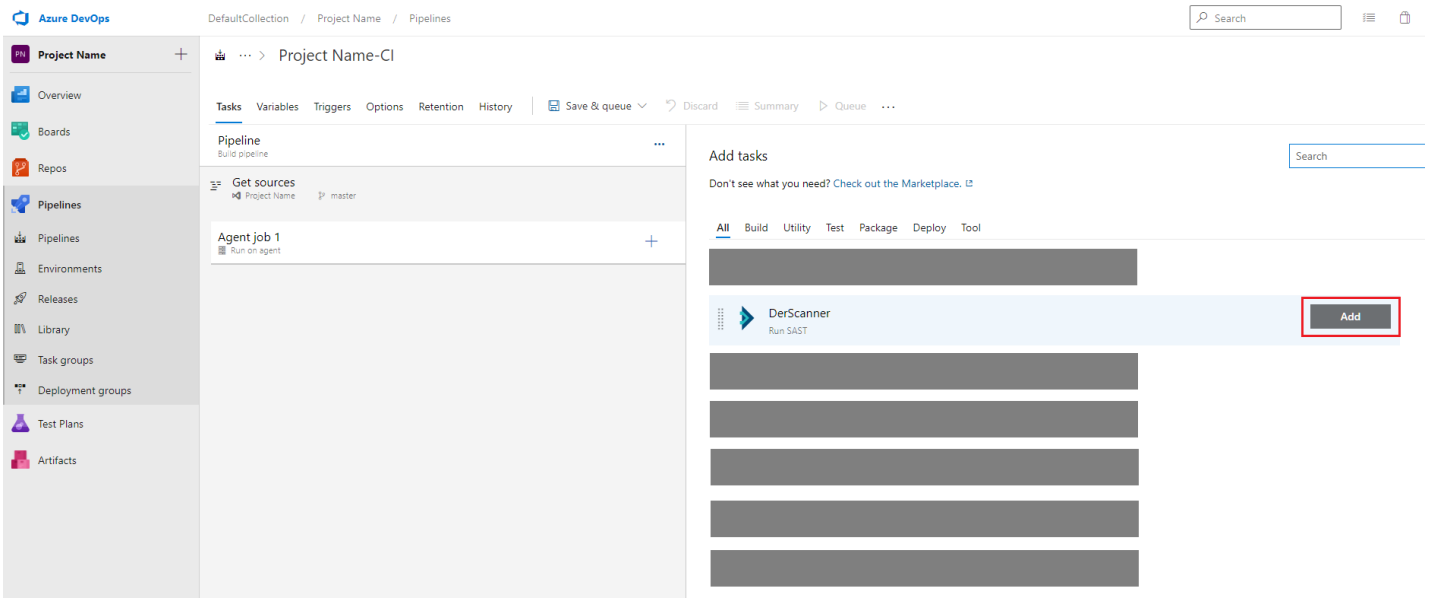


Рис. 6.33: Azure DevOps Server: Выбор задачи

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

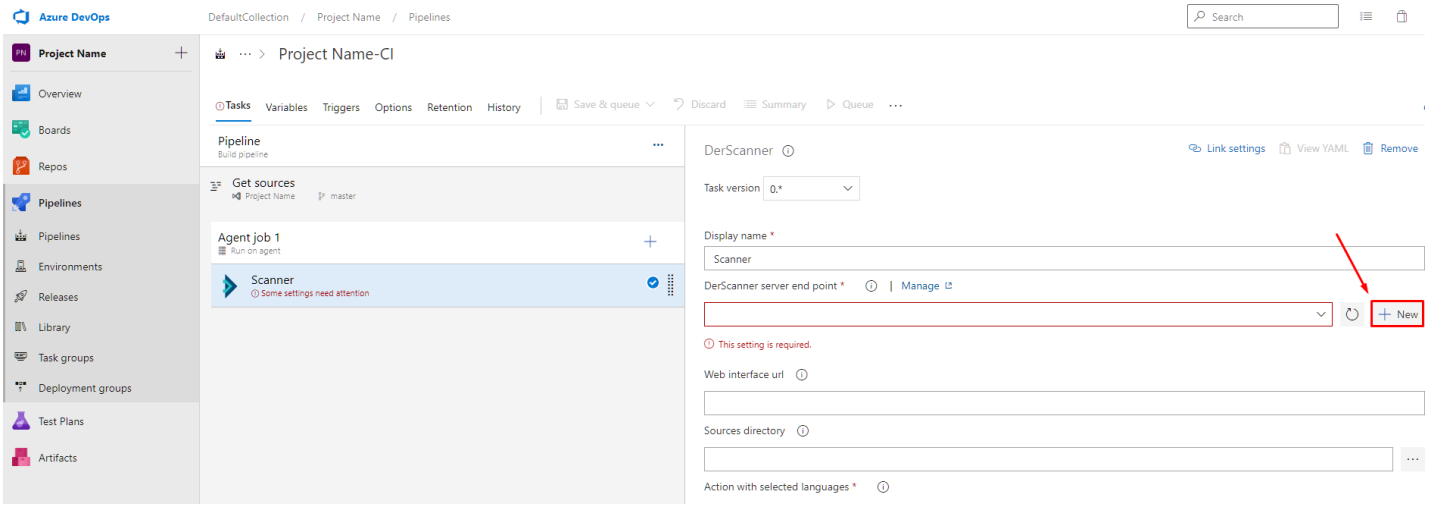


Рис. 6.34: Azure DevOps Server: Редактировать задачу

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

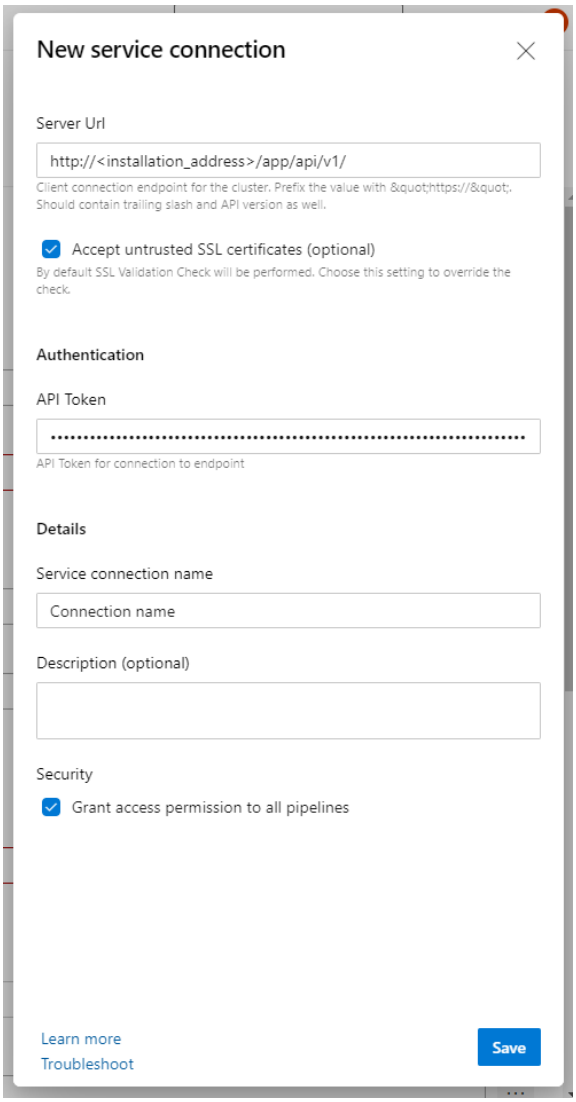


Рис. 6.35: Azure DevOps Server: Добавление подключения

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).

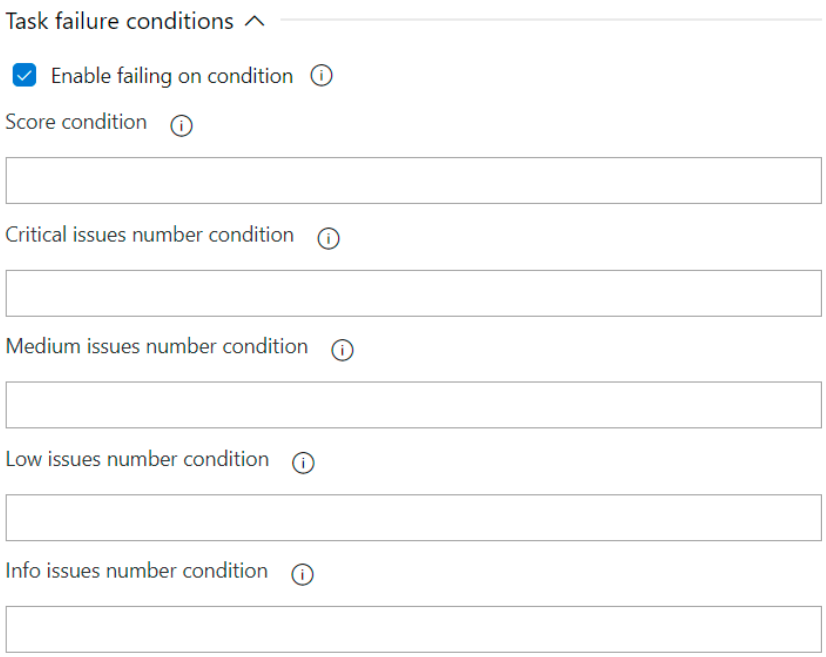


Рис. 6.36: Azure DevOps Server: Failure conditions

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**).

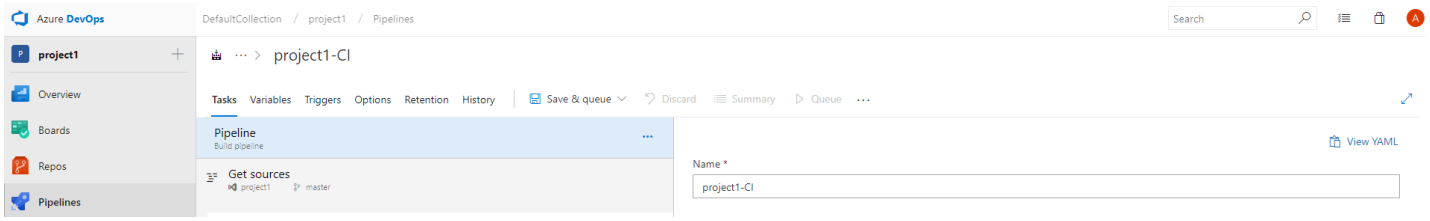


Рис. 6.37: Azure DevOps Server: Сохранение

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

6.6 TeamCity

DerScanner поддерживает интеграцию с **TeamCity 2017.2.2.** и более поздними версиями.

6.6.1 Интеграция DerScanner через плагин

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->DerScanner**.
    2. В разделе **DerScanner Static Analysis** введите адрес API (должен заканчиваться слешем, например: **http://<адрес машины, на которой установлен DerScanner>/app/api/v1/** и токен, который можно получить в веб-интерфейсе DerScanner в разделе **Токен и пароль**.
    3. Нажмите на кнопку **Test connection** (при успешной проверке подключения появится надпись **Successful**).
    4. Нажмите **Save**.
4. Добавьте **DerScanner SAST Build Step** (для выполнения статического анализа или DAST, SCA Build Step для выполнения других типов анализа) в настройках сборки и укажите для него необходимые настройки.
5. Добавьте **Build Features** для получения результатов анализа или отчета:
    - **DerScanner report** настройте экспорт отчёта с результатами сканирования;
    - **DerScanner 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 **DerScanner statistics** (DerScanner LOC; info, low, medium, critical vulnerabilities; scan duration; score).

6.6.2 Интеграция DerScanner с помощью Command Line Runner

Для интеграции DerScanner 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 DerScanner:
    - 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;
```

6.7 IntelliJ IDEA

DerScanner поддерживает интеграцию с **IntelliJ IDEA 2022.3.2 Runtime version: 17.0.5+1-b653.25 amd64**.

6.7.1 Установка плагина

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

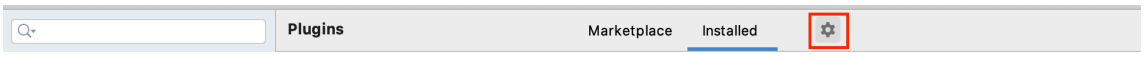


Рис. 6.38: Настройка конфигурации

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

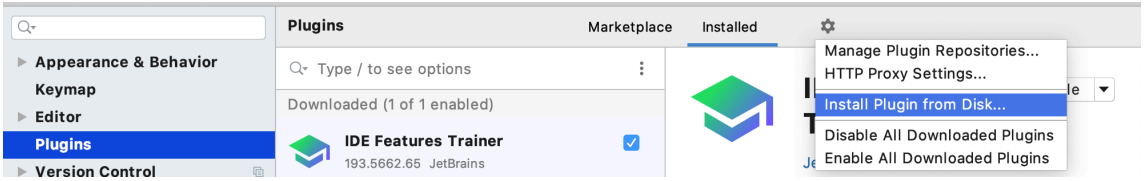


Рис. 6.39: Выбор плагина

4. Укажите путь до архива.

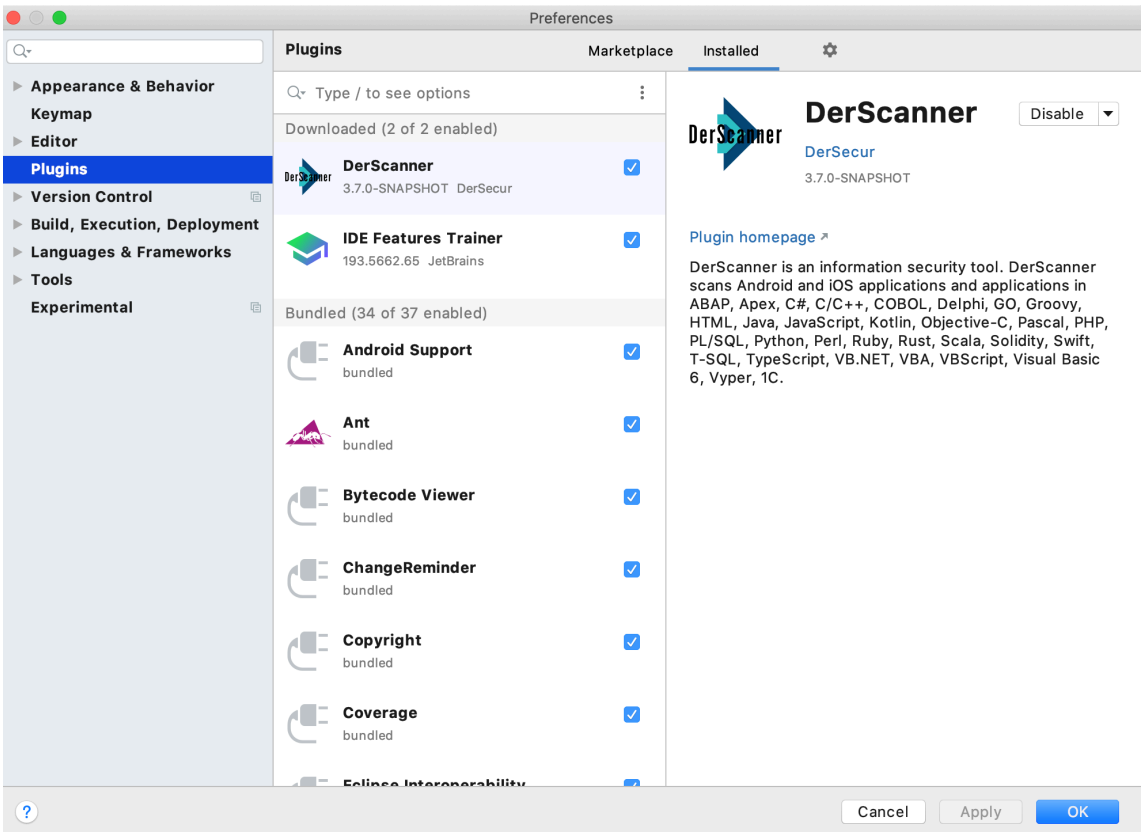


Рис. 6.40: Установленный плагин

6.7.2 Использование

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

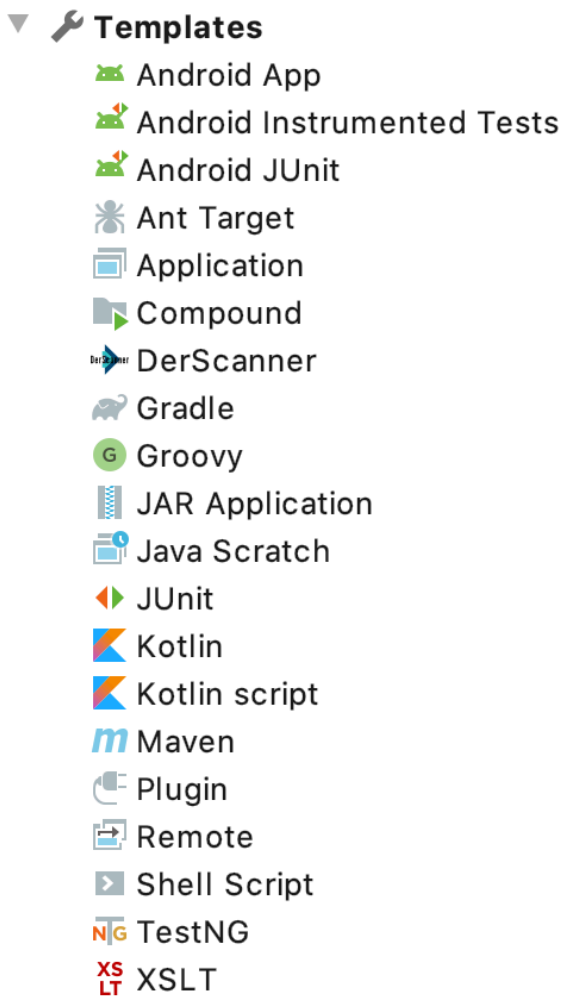


Рис. 6.41: Добавить конфигурацию

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, полученный в разделе **Токен и пароль** в интерфейсе DerScanner;
- **URL** путь до установки: **http://<installation_address>/app/api/v1/**;
- Нажмите **OK**.

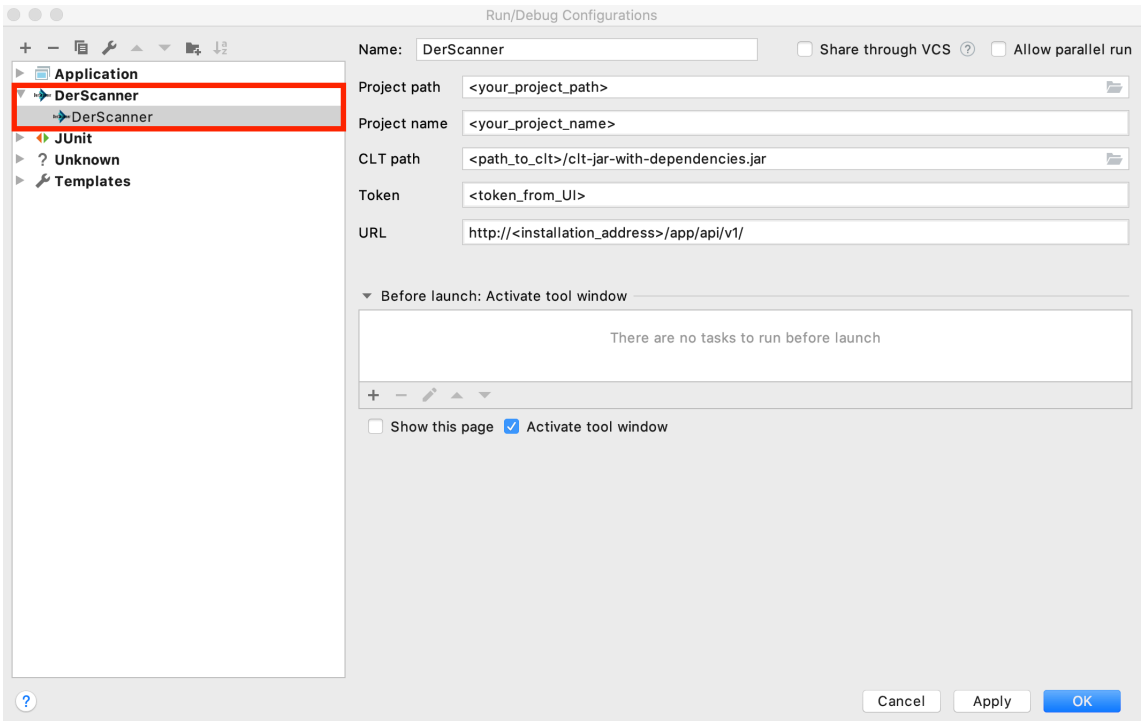


Рис. 6.42: Значения конфигурации

6.7.3 Запуск

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

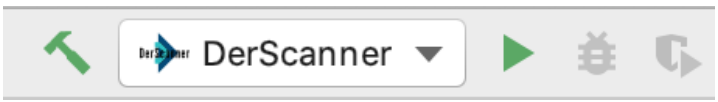


Рис. 6.43: Выбор конфигурации

2. Нажмите **Пуск**.

6.8 Eclipse

DerScanner поддерживает интеграцию с **Eclipse 2024-09 4.33.0** и **2022-12 (4.26.0)**.

6.8.1 Установка плагина

1. Для установки плагина завершите работу среды разработки Eclipse.
2. Поместите файл плагина в директорию **путь, по которому установлен Eclipse/dropins/plugins**. 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 **DerScanner**.

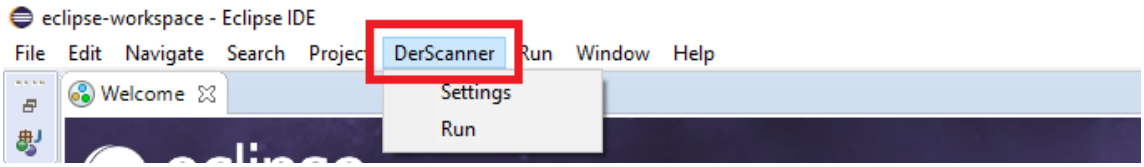


Рис. 6.44: Eclipse: установка

6.8.2 Использование

1. В тулбаре выберите **DerScanner**.
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 DerScanner в разделе **Аккаунт** (**Account**).
   4. Укажите название проекта (опционально).

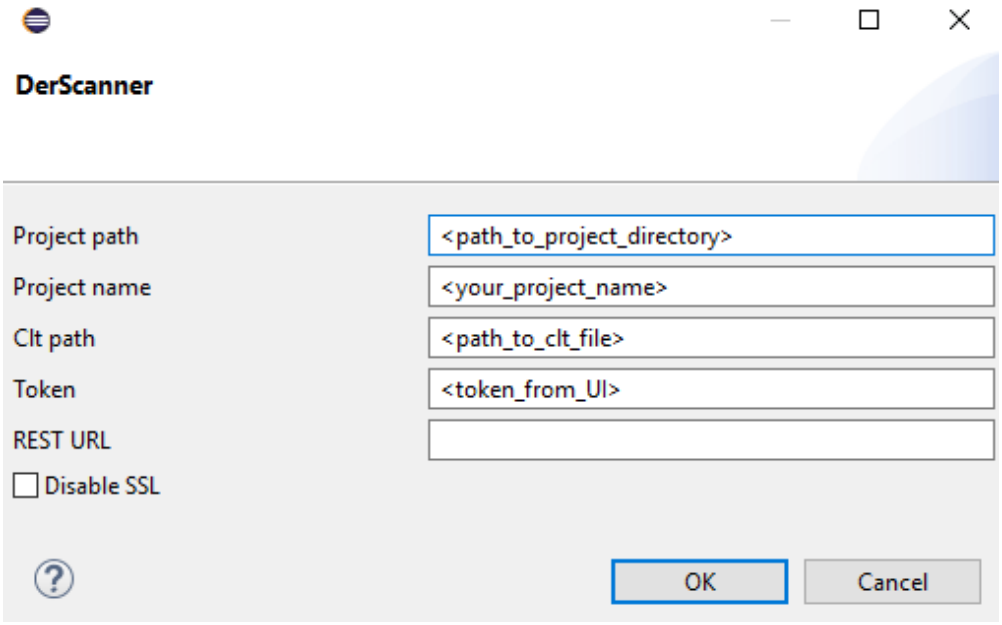


Рис. 6.45: Eclipse: настройки

6.8.3 Запуск

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

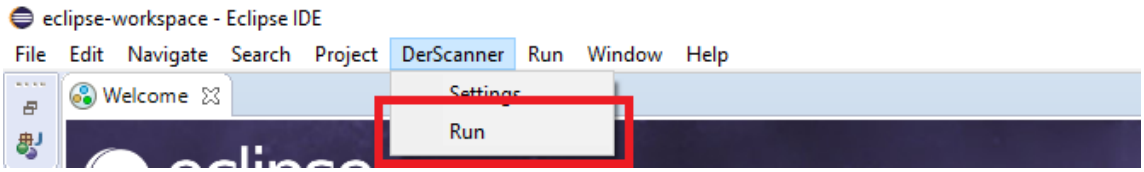


Рис. 6.46: Eclipse: запуск

6.9 Visual Studio

DerScanner поддерживает интеграцию с **Visual Studio 2019 16.11.41**.

6.9.1 Установка расширения

1. Откройте расширение с помощью Visual Studio.

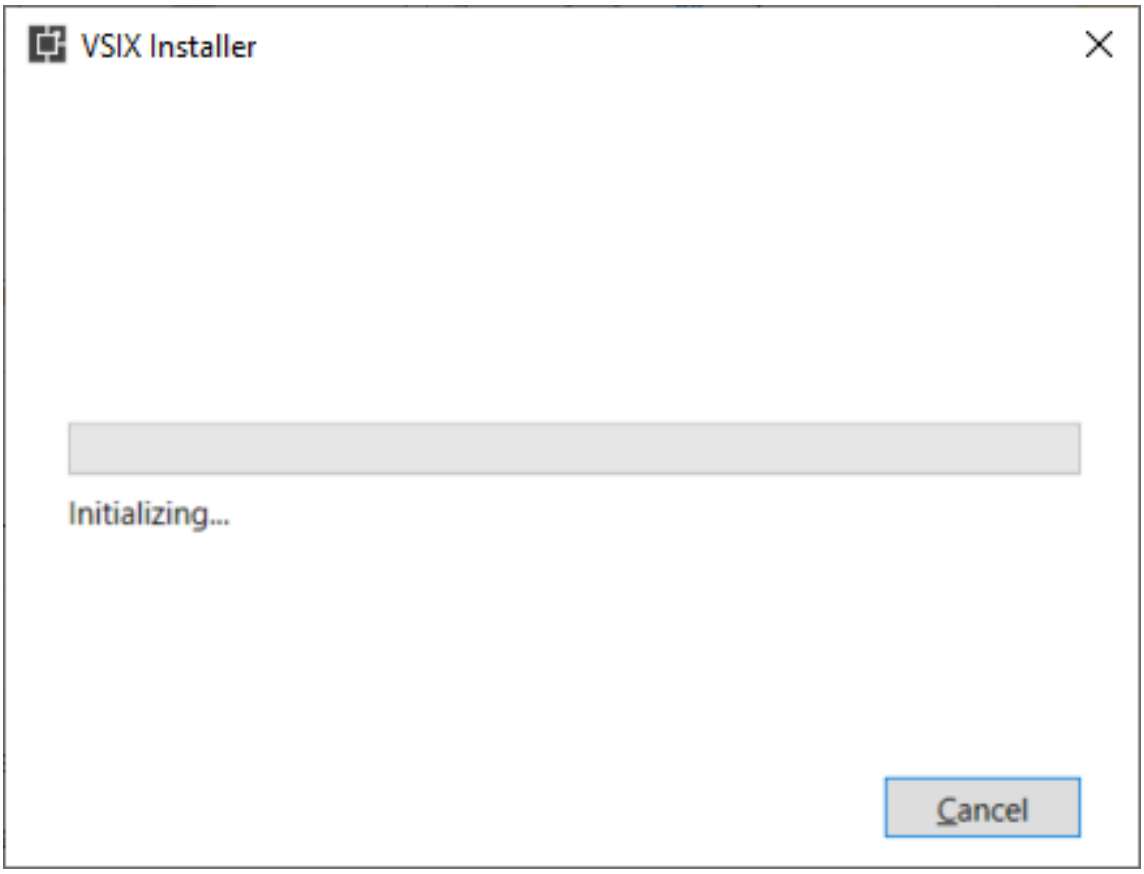


Рис. 6.47: Visual Studio: Запуск

2. Выберите IDE установки.

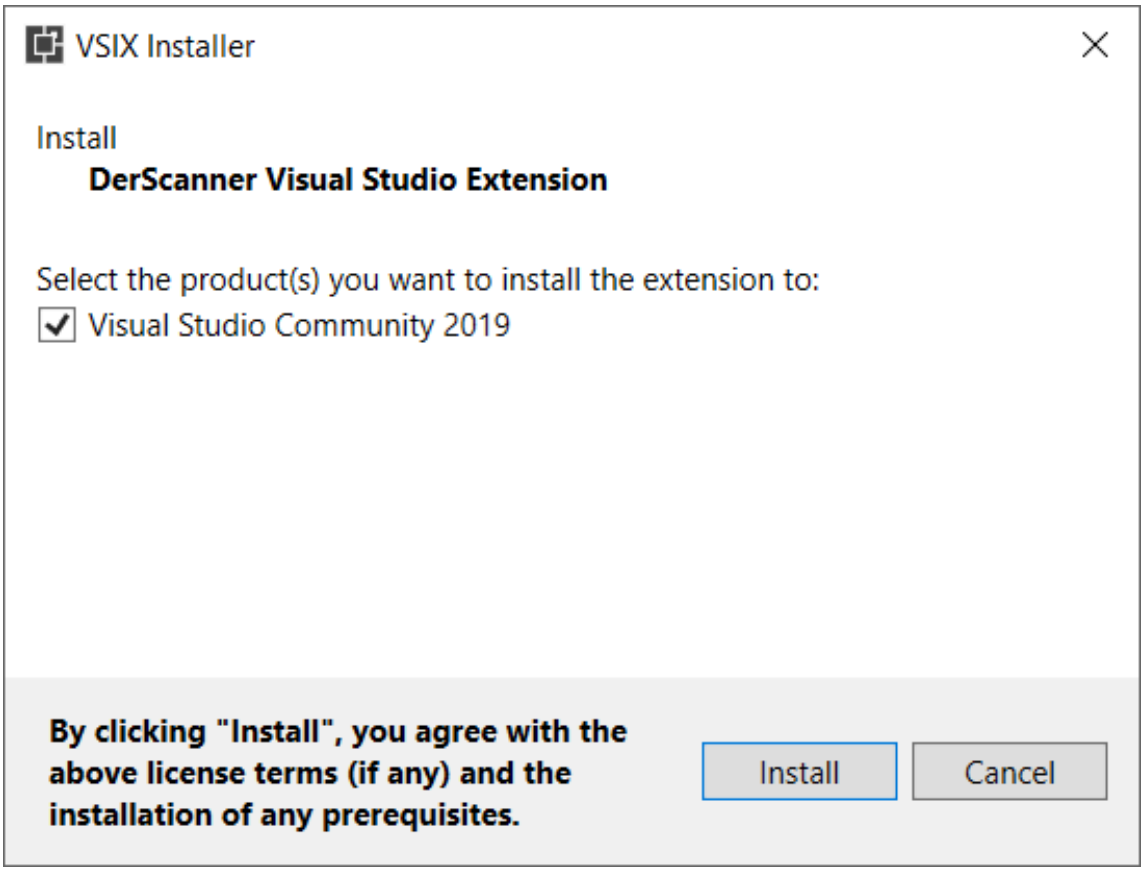


Рис. 6.48: Visual Studio: Выбор IDE

3. Дождитесь завершения.

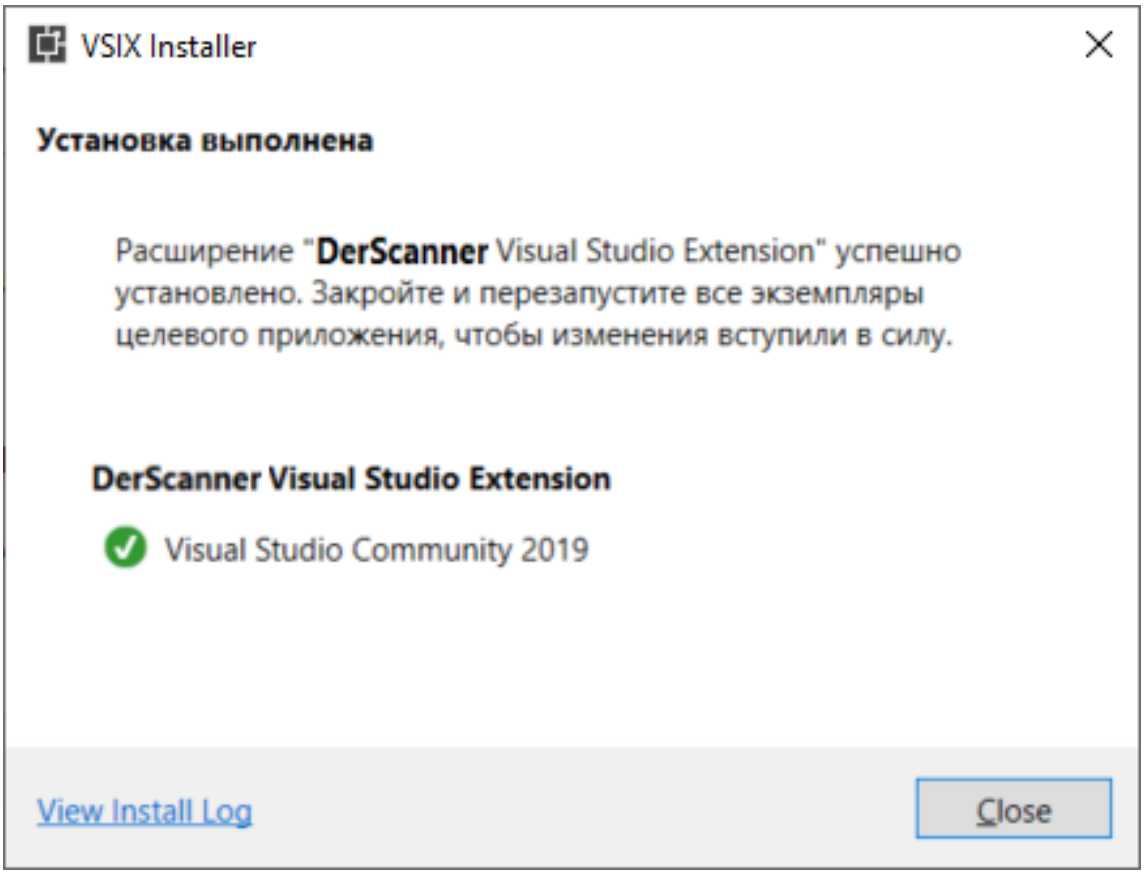


Рис. 6.49: Visual Studio: Установка расширения

6.9.2 Использование

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

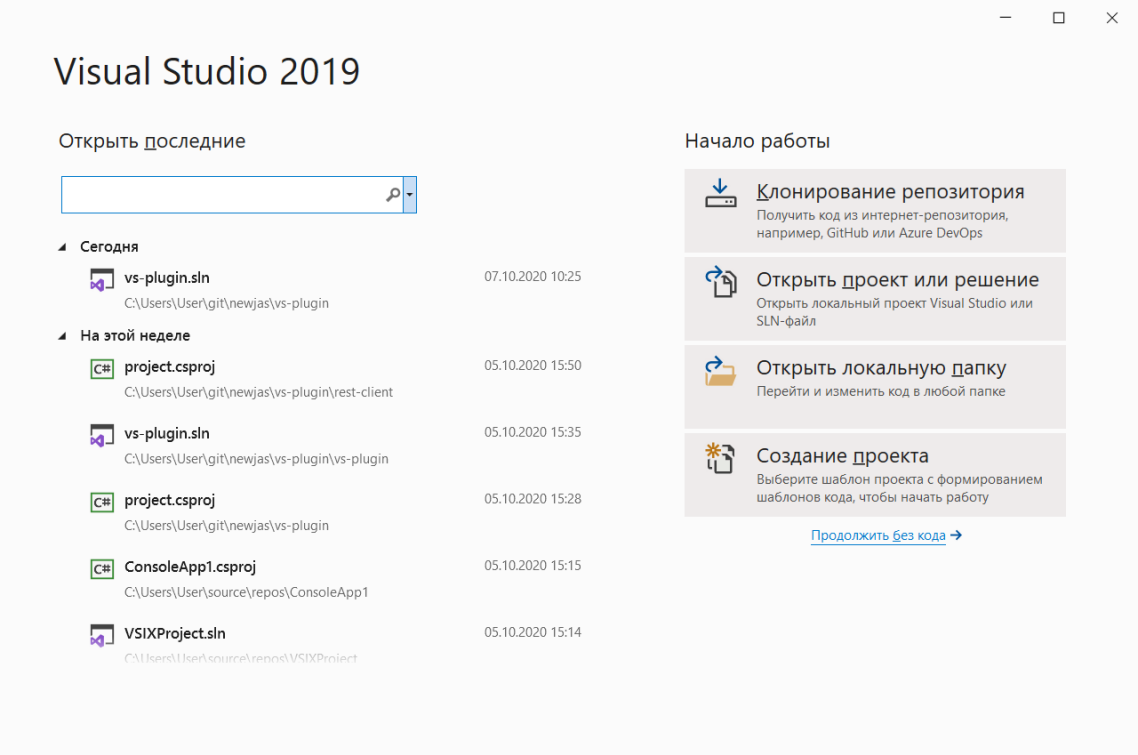


Рис. 6.50: Visual Studio: Открытие проекта

2. Перейдите в параметры.

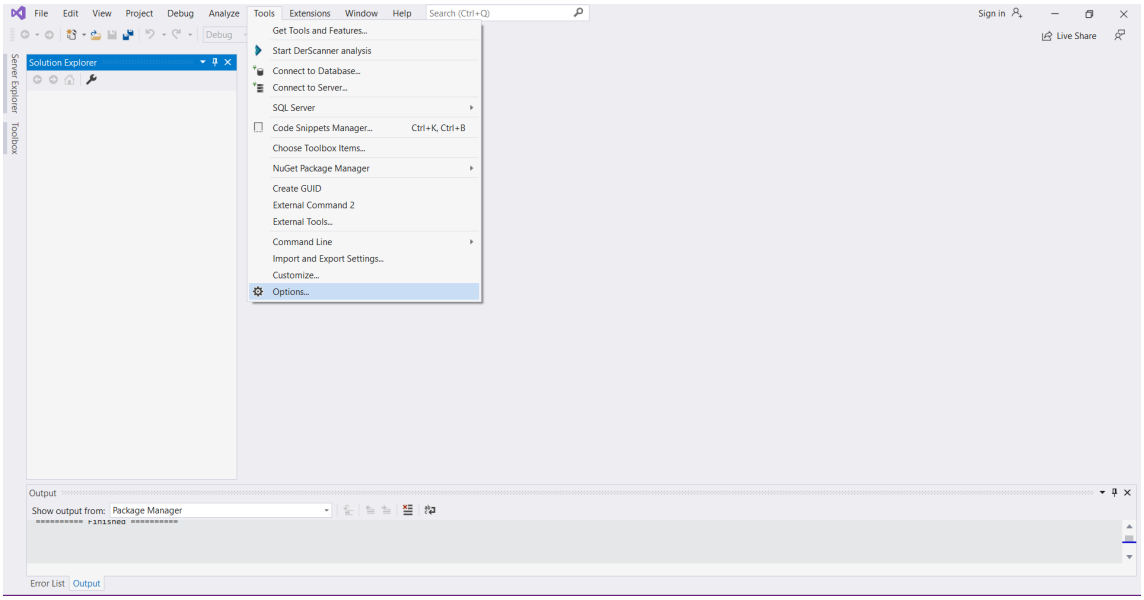


Рис. 6.51: Visual Studio: Параметры

3. Выберите **DerScanner Settings**.

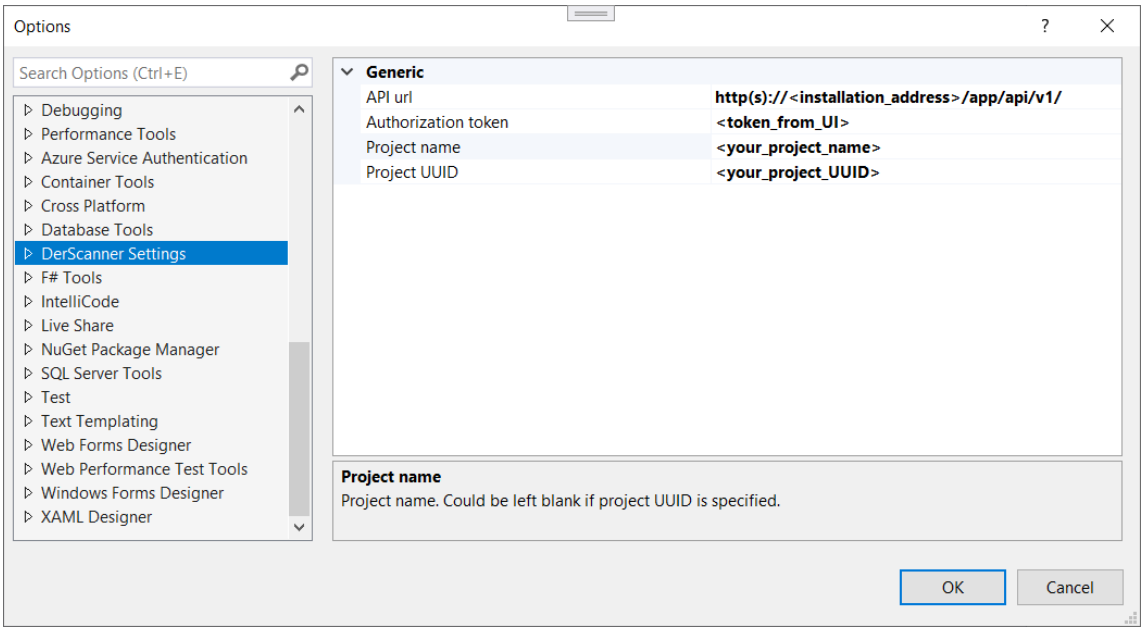


Рис. 6.52: Visual Studio: Выбор настроек

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

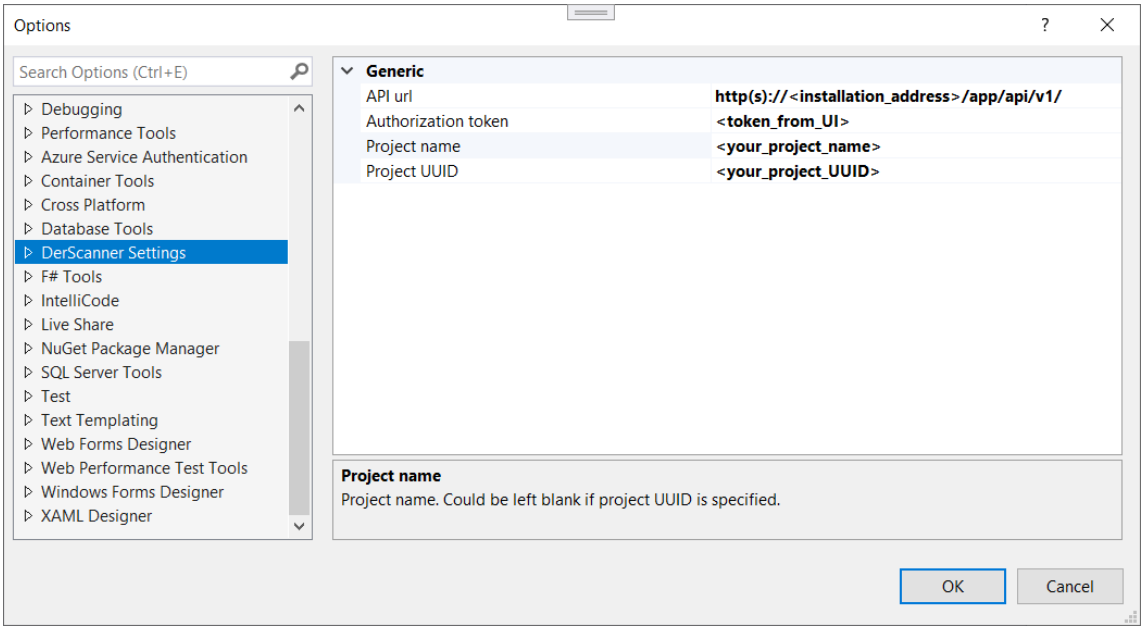


Рис. 6.53: Visual Studio: Параметры

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

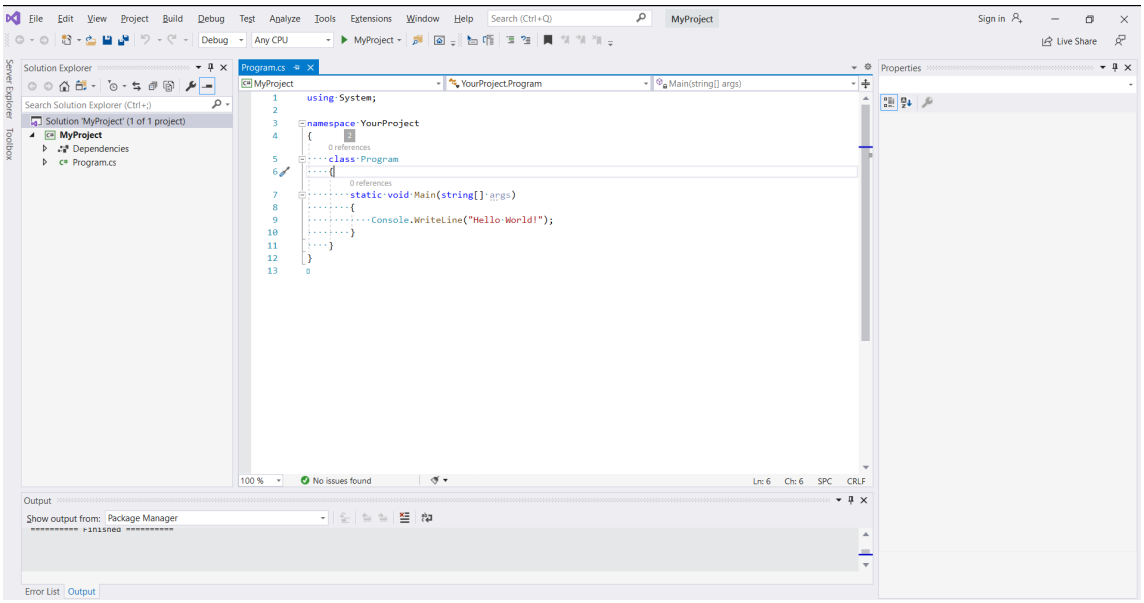


Рис. 6.54: Visual Studio: Выбор проекта

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

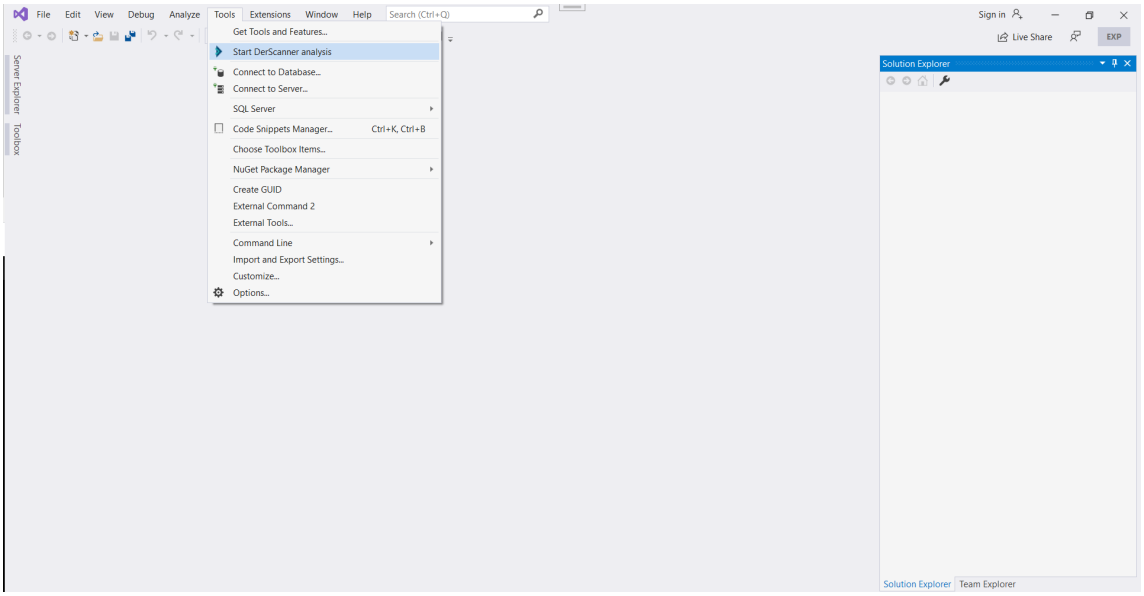


Рис. 6.55: Visual Studio: Запуск анализа

6.10 SonarQube

DerScanner поддерживает интеграцию с **SonarQube 7.9.** или более поздними версиями.

6.10.1 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.

6.10.2 Настройки расширения

Расширение конфигурируется тремя параметрами:

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

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

6.10.2.1 Глобальные настройки

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

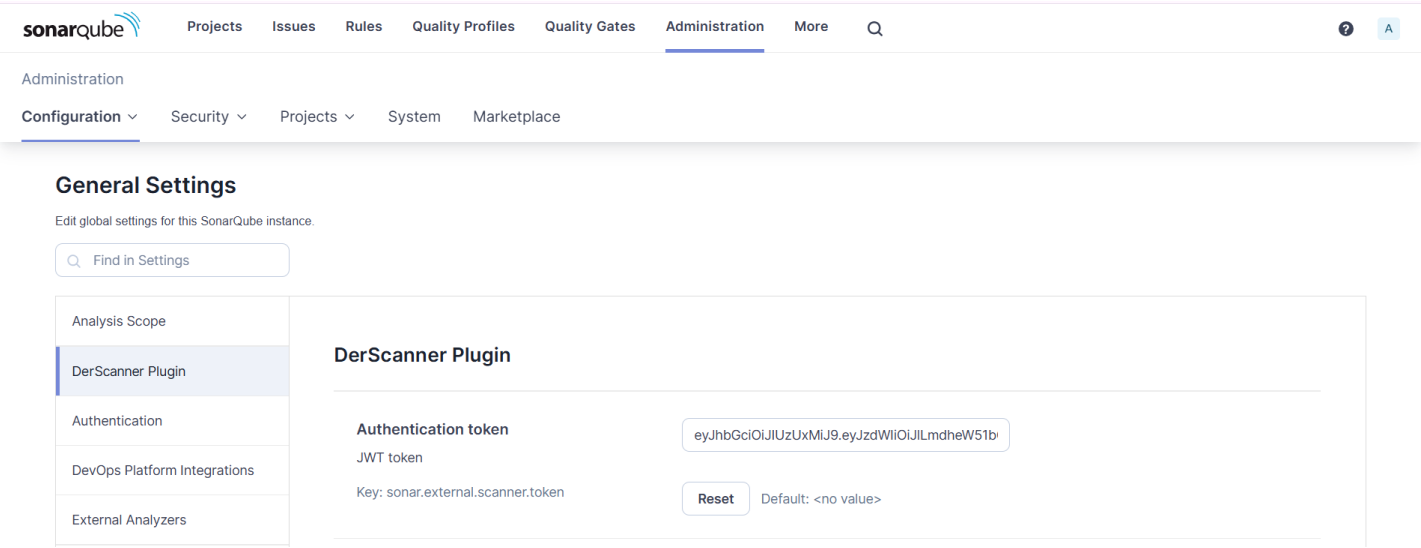


Рис. 6.56: SonarQube: Глобальные настройки

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

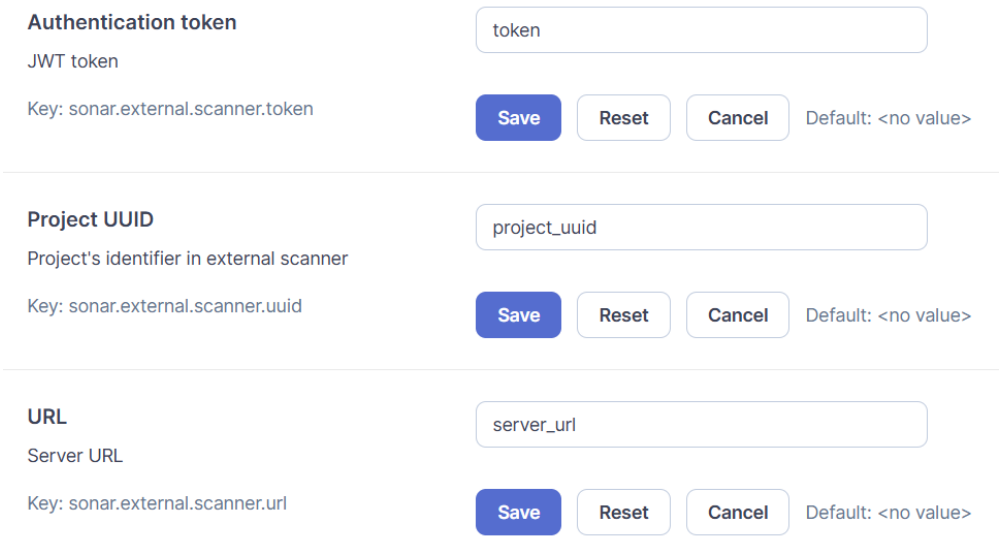


Рис. 6.57: SonarQube: Изменённые поля

6.10.2.2 Настройки проекта

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

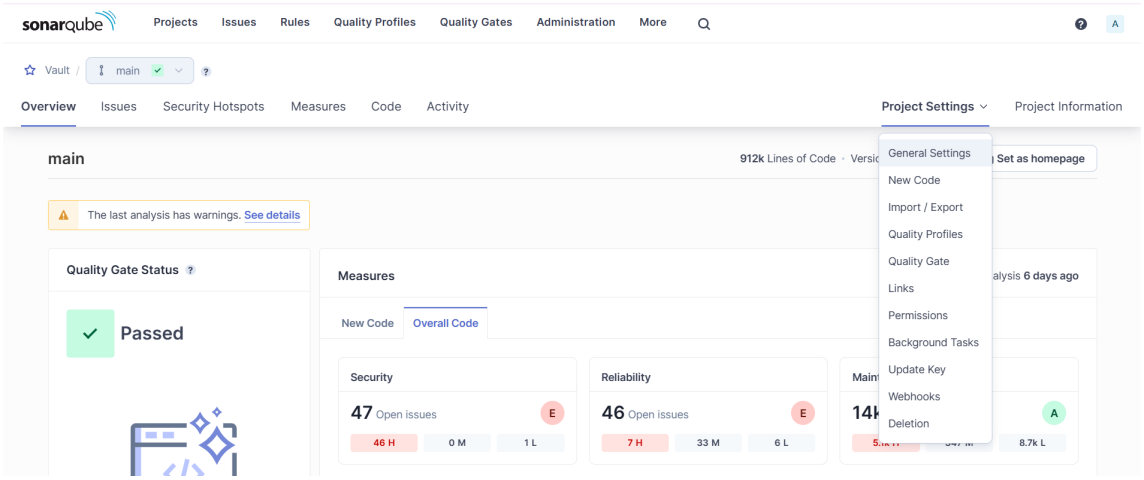


Рис. 6.58: SonarQube: Страница проекта

3. Перейдите на вкладку **DerScanner Plugin**.

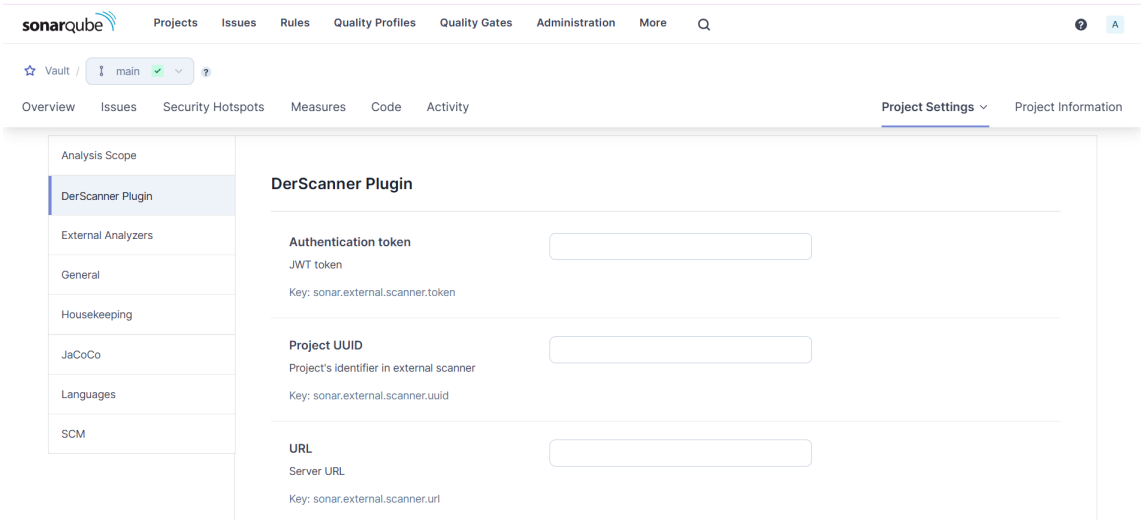


Рис. 6.59: SonarQube: Поля проекта

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

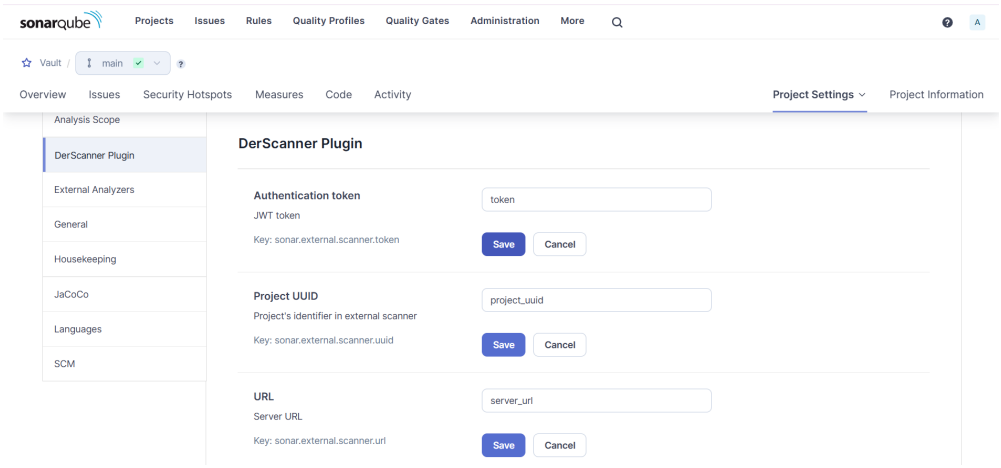


Рис. 6.60: SonarQube: Поле проекта

6.10.2.3 Запуск сканирования

При запуске сканирования в SonarQube, расширение загружает результаты последнего сканирования DerScanner для указанного проекта (поле **Project UUID** в настройках) и не запускает новое сканирование в DerScanner. Плагин сохраняет найденные в DerScanner 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 DerScanner.

6.11 Автоматическое сканирование из VCS хостингов

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**.

Обратите внимание, что для корректной работы интеграции необходимо предварительно настроить доступность API извне. Инструкции по настройке доступности API извне приведены в Руководстве Администратора.

Чтобы создать интеграцию, нужно настроить её в системе DerScanner и в самом хостинге.
Настройка

Чтобы создать интеграцию с VCS хостингом:

1. В настройках 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 DerScanner 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 DerScanner. 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 Quartz cron-выражения.
7. Для сохранения интеграции нажмите кнопку **Сохранить**.

Важно обратить внимание:

- если пользователь, создавший интеграцию, будет удалён из системы DerScanner, интеграция будет нерабочей. Чтобы продолжить работоспособность интеграции, нужно создать её повторно;
- машина, на которой установлен DerScanner, 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).

07 Соответствие классификации CWE

DerScanner предоставляет функциональность для работы с уязвимостями в соответствии с классификацией уязвимостей 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 DerScanner 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, HTML, CSV в соответствии с классификацией уязвимостей CWE/SANS Top 25 (подробнее в разделе Экспорт отчёта).

08 Список поддерживаемых расширений файлов

Таблица 8.1: Список поддерживаемых расширений файлов

Язык	Расширение
1C	`.bsl`, `.os`
ABAP	`.abap`
Apex	`.cls`
ASP.NET	`.aspx`
Bash	`.sh`
C#	`.cs`
C/C++	`.cpp`, `.cc`, `.c++`, `.c`
COBOL	`.cbl`, `.cob`, `.cpy`
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`, `.tsx`, `.vbhtml`, `.vue`, `.xht`, `.xhtml`
Java, Scala, Kotlin	`.java`, `.scala`, `.kt`, `.class`
JavaScript	`.cshtml`, `.js`, `.jsf`, `.jsm`, `.jsp`, `.jspx`, `.jsx`, `.htm`, `.html`, `.phtml`, `.php`, `.vbhtml`, `.xht`, `.xhtml`, `.xsjs`
JSP	`.jsp`
LotusScript	`.lsl`, `.lss`
Objective-C, Swift	`.m`, `.mm`, `.swift`

Язык	Расширение
Pascal	`.pas`, `.p`, `.pp`
PHP	`.php`, `.php3`, `.php4`, `.php5`, `.php6`, `.phps`, `.phpt`, `.phtml`
PL/SQL	`.fnc`,`.pck`, `.pkb`, `.pkh`, `.pks`, `.plsql`, `.prc`, `.sf`, `.spb`, `.spp`, `.sps`, `.sql`, `.trg`
PowerShell	`.ps1`
Python	`.py`, `.python`
Perl	`.pl`, `.pm`, `.cgi`, `.plx`
Ruby	`.rb`
Rust	`.rs`
Solidity	`.sol`
T-SQL	`.sp`, `.sql`, `.tsql`
TypeScript	`.ts`
VB 6.0	`.vbp`
VB.NET	`.vb`
VBA	`.vba`
VBScript	`.asp`, `.hta`, `.htm`, `.html`, `.vbe`, `.vbs`, `.wsc`, `.wsf`
Visual Basic	`.vb`, `.bas`, `.cls`, `.frm`
Vyper	`.vy`
XML	`.xml`
YAML	`.yaml`, `.yml`
TOML	`.toml`

09 Список фреймворков и стандартных библиотек, поддерживаемых SAST

Таблица 9.1: Список фреймворков и стандартных библиотек, поддерживаемых SAST

Язык	Библиотеки
Java, Scala, Kotlin	Android, Android Support Library, android (.accessibilityservice, .accounts, .animation, .annotation, .app, .bluetooth, .companion, .content, .database, .drm, .gesture, .graphics, .hardware, .icu, .inputmethodservice, .location, .media, .mtp, .net, .nfc, .opengl, .os, .preference, .print, .printservice, .provider, .renderscript, .sax, .security, .service, .speech, .system, .telecom, .telephony, .test, .text, .transaction, .util, .view, .webkit, .widget), com.android.internal.util 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, Apache Xalan Java, org.apache.xml, org.apache.xpath, Apache Xerces, Sun Org Apache XML Clojure, ClojureScript, cljs dalvik.annotation, dalvik.bytecode, dalvik.system 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 ScalaCheck, Scala JS, Scala Library, Scalac SCoverage Plugin, Scala Test, scala (.annotation, .beans, .collection, .compat, .concurrent, .io, .math, .ref, .reflect, .runtime, .sys, .text, .util)

Язык	Библиотеки
	Akka, ASM, AssertJ, Bean Validation API, CDI (Contexts and Dependency Injection for Java) APIs, javax.decorator, cognitect, Codehaus, Jackson, Crashlytics, Dom4J, EasyMock, EclipseLink, FindBugs, Firebase Messaging, H2 Database Engine, Hamcrest, Hazelcast, Hibernate, Hibernate API, HttpClient API, HyperSQL Database, Jackson, Java Servlet, JavaMail API, com.sun.mail, 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, Simple XML Framework, SLF4J, Spock Framework API, org.spockframework, SpringFramework, Spring Boot, TestNG, com.beust.testng, Vaadin, W3C DOM
JavaScript	Bootstrap
	jQuery mobile
	Node.js, Node Express
	React, React Native, Preact
	vue.js
C#	.NET Core
	ASP.NET, ASP.NET Core
	Entity Framework
	NLog
GO	Echo
	Gin
Swift	Network Framework
PHP	Laravel
	Symphony
	Yii
Python	Django
	Flask
	SQLAlchemy platform

10 Манифесты, поддерживаемые для генерации SBOM

Таблица 10.1: Манифесты, поддерживаемые для генерации SBOM

Язык	Манифесты
Java/Kotlin/Scala	`pom.xml`, `ivy.xml`, `*.gradle`, `gradle.lockfile`
JavaScript/TypeScript	`package.json`, `package-lock.json`, `yarn.lock`, `pnpm-lock.yaml`
Python	`setup.py`, `Pipfile`, `Pipfile.lock`, `pyproject.toml`, `poetry.lock`, `requirements.txt`, `requirements.pip`, `requires.txt`
C/C++	`conanfile.txt`, `conan.lock`
Go	`go.mod`, `go.sum`
PHP	`composer.json`, `composer.lock`
Ruby	`Gemfile`, `Gemfile.lock`, `*.gemspec`
C#/VB.NET	`*.nuspec`, `packages.lock.json`, `Project.json`, `Project.lock.json`, `packages.config`, `paket.dependencies`, `paket.lock`, `*.csproj`, `*.fsproj`, `*.vbproj`, `project.assets.json`, `sln`
Objective-C/Swift	`Podfile`, `Podfile.lock`, `*.podspec`
Rust	`Cargo.lock`, `Cargo.toml`
Dart	`pubspec.yaml`, `pub.lock`
Erlang	`rebar.config`, `rebar`, `rebar.lock`

11 Обозначения языков программирования в спецификации API

Таблица 11.1: 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

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

Для удобного копирования в запрос:

ABAP,ANDROID,APEX,CS,CCPP,COBOL,CONFIG,DART,DELPHI,GO,GROOVY,HTML5,
JAVA,JAVASCRIPT,KOTLIN,LOTUS,OBJC,PASCAL,PHP,PLSQL,PYTHON,PERL,RUBY,RUST,
SCALA,SOLIDITY,SWIFT,TSQL,TSX,TYPESCRIPT,VBNET,VBA,VBSCRIPT,VB,VYPER,ONES

12 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
SBOM	Software Bill of Materials – документ, содержащий подробный перечень компонент и зависимостей
SDLC	System Development Life Cycle – жизненный цикл разработки системы
VCS	Version Control System – система управления версиями

13 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.

LDAP (Lightweight Directory Access Protocol) — протокол прикладного уровня для доступа к службе каталогов. Протокол использует TCP/IP и позволяет производить операции аутентификации (bind), поиска (search) и сравнения (compare), а также добавления, изменения или удаления записей.

Software Bill of Materials (SBOM) — документ, содержащий подробный перечень компонент и зависимостей, используемых в проекте программного обеспечения. В нем также перечислены все библиотеки, фреймворки и их соответствующие версии, которые используются в программном обеспечении.

Software Development Lifecycle (SDLC) — методика разработки, которая обеспечивает качество и правильность работы программного обеспечения. Методика SDLC состоит из 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**.

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.

Инструменты командной строки (CLT) — программы без графического интерфейса, взаимодействие с которыми осуществляется через командную строку.

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 компонент, 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 DerScanner трасса для вхождения изображается в виде диаграммы, отображающей путь распространения данных уязвимости, который выявлен с помощью анализа потока данных.