

Svace History Server

Version 6.0.0

Содержание

Содержание.....	2
1 Введение.....	5
1.1 Термины.....	5
1.2 Состав дистрибутива.....	5
1.3 Зависимости.....	6
1.4 Используемые порты.....	6
2 Установка.....	7
2.1 Установка и запуск из deb/rpm пакета.....	7
2.1.1 .deb.....	7
2.1.2 .rpm.....	7
2.1.3 Настройка и запуск.....	8
2.2 Запуск в docker.....	9
2.3 Установка и запуск вручную.....	10
2.4 Hooks.....	11
3 Использование.....	13
3.1 Получение справки.....	13
3.2 Процесс импорта.....	13
3.2.1 Промежуточное хранилище.....	13
3.2.2 Пользовательские атрибуты.....	14
3.2.3 Прикрепление файлов.....	15
3.2.4 Загрузка данных на сервер.....	17
3.2.5 Модификация путей в Review (разметке) при импорте.....	19
3.3 Создание PDF отчета для проекта.....	21
4 Миграция.....	25
4.1 Миграция со старого сервера.....	25
4.2 Миграция между версиями Svacer.....	26

4.3 Создание и восстановление резервной копии.....	26
4.3.1 Если PostgreSQL запущен в виде сервиса.....	26
4.3.2 Если PostgreSQL запущен в docker-контейнере.....	27
4.3.3 Создание резервной копии svacer object store через command line API.....	27
4.3.4 Создание резервной копии svacer object store в сервере PostgreSQL.....	28
4.3.5 Создание резервной копии svacer object store вручную.....	29
4.4 Перенос снимков.....	30
4.4.1 Перенос из веб интерфейса.....	30
4.4.2 Перенос из командной строки.....	32
4.5 Перенос Svacer с одного сервера на другой.....	33
5 Работа с сервером.....	34
5.1 Вход в сервер.....	34
5.1.1 Профиль пользователя.....	34
5.1.2 Организации.....	37
5.2 Выбор проекта и снимка.....	40
5.3 Виды интерфейса.....	40
5.4 Элементы управления в режиме разметки.....	40
5.5 Групповая разметка предупреждений.....	47
5.6 Использование регулярных выражений.....	49
5.7 Блокировка разметки.....	50
5.7.1 Экспорт исходных кодов с разметкой (опционально) из снимка.....	51
5.7.2 Импорт разметки из исходного кода на сервер истории.....	53
5.7.3 Изменение инвариантов для распространения разметки между снимками, с несовпадающими путями.....	54
5.8 Подавление предупреждений.....	55
5.9 Публичные REST запросы.....	56
5.10 Администрирование сервера Svace.....	60
5.10.1 Учетные записи и роли пользователей.....	60

5.10.2 Семантика ролей пользователя.....	61
5.10.3 Ведение списка проектов.....	63
5.10.4 Настройка фильтров веток и проектов.....	64
5.10.5 Просмотр информации о сервере.....	64
5.10.6 Копирование разметки.....	65
5.10.7 Создание и изменение шаблонов разметки.....	66
5.11 Поддержка протокола LDAP для аутентификации пользователей.....	68
5.11.1 Общий принцип аутентификации.....	68
5.11.2 Запуск сервера с поддержкой LDAP.....	68
5.11.3 Конфигурационный файл сервера для поддержки LDAP.....	69
5.11.4 Использование CLI сервера Svacer с поддержкой LDAP.....	73
5.11.5 Обновление данных LDAP пользователей.....	73
5.12 Поддержка сквозной аутентификации.....	75
5.13 Поддержка OAuth.....	75
5.13.1 API-вызовы для управления клиентами.....	75
6 Панель отчетов.....	76
7 Работа с SARIF.....	77
7.1 Генерация SARIF.....	77
7.1.1 Генерация из .svres.....	77
7.1.2 Генерация из данных с сервера.....	77
7.2 Загрузка на сервер из SARIF.....	78
8 Интеграция с Visual Studio Code.....	79

1 Введение

Сервер историй **Svacer (Svace Server)** предназначен для хранения и обработки результатов работы статического анализатора Svace. Он разработан с целью заменить старый сервер историй, встроенный в анализатор Svace.

Допускается одновременное использование старого и нового серверов.

1.1 Термины

Project	Проект, подлежащий анализу. По умолчанию именем проекта считается имя директории, в которой был запущен svace build и svace analyze. На сервере проект содержит в себе несколько branch с результатами работы анализатора
Branch (ветка)	Ветка в проекте с результатами работы анализатора. По умолчанию проект имеет одну ветку — master
Snapshot (снимок)	Результат работы svace analyze, импортированный в промежуточное хранилище или на сервер. Содержит: <ul style="list-style-type: none">▪ имя проекта;▪ имя ветки;▪ результаты анализа (предупреждения);▪ исходники, которые были включены в build, на котором производился анализ;▪ вспомогательная информация для навигации по исходникам;▪ дополнительные данные; Допускается импортирование снимков без исходного кода и дополнительной информации
Marker (маркер, предупреждение)	Предупреждение от Svace с информацией о позиции в исходном файле
Svacer	Svace History Server, а также исполняемый файл, содержащий код сервера
Разметка	Проставление статуса маркера с опциональным добавлением комментария

1.2 Состав дистрибутива

bin/svacer bin/svacer.exe	Бинарный файл сервера и утилита для импортирования результатов. Режим работы зависит от аргументов
support/svace-migration/svace-migration.jar	Утилита подготовки данных для импорта из старого сервера историй. Требует JDK для запуска. Может использоваться JDK из состава дистрибутива Svace
extra/docker-	Docker-compose файлы для запуска БД PostgreSQL и Svacer в docker-

compose*.yaml	контейнерах
integrations/vscode	Расширение для Visual Studio Code и документация на него

1.3 Зависимости

PostgreSQL версии не ниже 10-й	<p>Параметры подключения определяются аргументом при запуске сервера или переменной окружения SVACER_PG_URL. По умолчанию, используется следующий connect URL:</p> <p>postgres://svace:svace@0.0.0.0:5432/svace</p> <p>Можно использовать PostgreSQL в docker-контейнере.</p> <p>Настоятельно рекомендуем использовать выделенный сервер для хранения БД, особенно в production</p>
libc.so libpthread.so libdl.so linux-vdso.so	Серверная часть написана на Go с CGO_ENABLED=1 ввиду необходимости читать sqlite-файлы из результатов работы анализатора Svace
Выделенная директория для хранения object storage	Сервер использует встроенную key-value database для хранения информации об исходном коде: контент-файлов и вспомогательной информации для навигации
svace	Для импорта данных на сервер историй требуется наличие Svace на хосте, откуда происходит импорт. Svace используется для получения информации из .svace-dir

1.4 Используемые порты

Приведены значения по умолчанию. Все значения могут быть переопределены.

3002	Используется для предоставления gRPC API. gRPC API является внутренним и предназначен для загрузки данных на сервер
8080	Используется для предоставления REST API и web-сервера для предоставления web-UI

2 Установка

Для работы сервера необходим PostgreSQL. Его можно установить как пакет, либо запустить в докере — docker-compose файл для этого находится в **./extra/docker-compose-postgres.yml**. Svacer сервер состоит из единственного бинарного файла — **svacer**. Он предоставляет как backend-часть, так и веб-сервер, который передаёт статику. По умолчанию запускается на порту 8080.

После запуска по умолчанию создается учётная запись пользователя **admin** с паролем **admin**. Изменить пароль можно в веб-интерфейсе.

Сервер имеет широкие возможности настройки параметров запуска. Узнать о них подробнее можно, прочитав **svacer --help**.

2.1 Установка и запуск из deb/rpm пакета

Процесс установки из .deb и .rpm пакетов несколько различается и будет описан ниже в отдельных главах. Процесс настройки и запуска не различается и будет описан в одной общей главе.

2.1.1 .deb

Скачайте .deb пакет релиза и выполните следующую команду

```
sudo apt install ./svacer_<version>_amd64.deb
```

При этом, если в репозиториях есть PostgreSQL нужной версии, он будет установлен автоматически. Если нет, то установка завершится с ошибкой о зависимости от PostgreSQL. В этом случае вам надо будет установить PostgreSQL версии не ниже 10-й (см. [документацию](#)), после чего повторить установку Svacer.

2.1.2 .rpm

Установите PostgreSQL версии не ниже 10-й для вашей ОС следуя [документации](#). Дополнительно установите пакет postgresql-contrib из тех же репозиториев.

Пример для Centos 7

```
sudo yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
sudo yum install -y postgresql14-server postgresql14-contrib
sudo /usr/pgsql-14/bin/postgresql-14-setup initdb
sudo systemctl enable postgresql-14
sudo systemctl start postgresql-14
```

Для версий PostgreSQL ниже 13-й требуется явно разрешить подключение к локальному хосту базы с логином/паролем. Для этого проверьте следующие строки в конце файла `/var/lib/pgsql/<version>/data/pg_hba.conf`

```
# IPv4 local connections:
host    all             all             127.0.0.1/32          ident
# IPv6 local connections:
host    all             all             ::1/128               ident
```

В случае если в последнем столбце указано значение **ident**, поменяйте его на **md5** и перезапустите PostgreSQL.

```
sudo systemctl restart postgresql-12
```

Скачайте .rpm пакет релиза Svacer и выполните следующую команду

```
sudo yum install ./svacer-<version>.x86_64.rpm
```

2.1.3 Настройка и запуск

Процесс настройки и запуска одинаковый для Svacer установленного из .deb и .rpm пакетов.

В процессе установки создаются следующие файлы и директории

- `/etc/default/svacer` — конфигурационный файл
- `/var/log/svacer` — директория для логов
- `/var/lib/svacer` — директория для object store

После установки Svacer требуется создать пользователя и БД PostgreSQL. Для этого перейдите в пользователя `postgres`, запустите `psql` и выполните соответствующие запросы, после чего выйдите из консоли PostgreSQL и из пользователя `postgres`.

```
sudo su -l postgres
psql
postgres=# create database svace;
postgres=# create user svace with encrypted password 'svace';
postgres=# grant all privileges on database svace to svace;
postgres=# alter user svace superuser;
```

В данном примере создается БД `svace` и права на нее выдаются пользователю `svace` с паролем `svace`, а также этому пользователю выдаются права суперюзера (это необходимо для создания расширений в PostgreSQL). При использовании этих значений дальнейшая конфигурация не требуется и можно переходить к запуску.

При использовании других имен пользователя, БД или пароля потребуется дополнительная конфигурация перед запуском Svacer: в файле `/etc/default/svacer` нужно поменять параметры подключения к БД в строке

```
SVACER_ARGS="- -pg postgres://<user>:<password>@127.0.0.1:5432/<database>"
```

В этой же строке можно указывать прочие аргументы для запуска сервера Svacer.

После создания БД и настройки конфигурации Svacer запустить его можно следующими командами

```
sudo systemctl enable svacer  
sudo systemctl start svacer
```

После чего проверить успешность запуска командой

```
systemctl status svacer
```

В случае успешного запуска сервер будет доступен по адресу <http://localhost:8080>

При установке из `.deb` или `.rpm` пакета Svacer ставится в директорию, прописанную в `$PATH`, поэтому при запуске Svacer для импорта и загрузки результатов полный путь к исполняемому файлу **svacer** указывать не требуется, он будет доступен просто по имени.

2.2 Запуск в docker

Установите `docker` и `docker-compose`. Поскольку образы PostgreSQL и Svacer размещены на докер-хабе, для их скачивания при запуске необходимо наличие интернета.

Используйте для запуска `docker-compose` файл, находящийся в дистрибутиве Svacer: `./extra/docker-compose.yml`. Перейдите в директорию, где находится этот файл и выполните команду

```
docker-compose up -d
```

При этом будет запущено два контейнера: PostgreSQL и Svacer. После запуска контейнеров веб-интерфейс Svacer будет доступен по адресу <http://localhost:8080>

В процессе запуска в текущей директории будут созданы две директории

- `postgres_data` — для хранения БД PostgreSQL
- `svacer_data` — для object store сервера Svacer

Эти директории будут примонтированы в соответствующие контейнеры как volumes, это необходимо для сохранения данных БД и сервера после остановки или перезапуска контейнеров.

Важное уточнение: в докере запускается только сервер Svacer. Для импорта и загрузки результатов на сервер будет нужен исполняемый файл svacer.

Остановить сервер Svacer можно, выполнив следующую команду

```
docker-compose down
```

Также можно запустить в докер-контейнере только PostgreSQL для последующего запуска Svacer вручную. Используйте файл **./extra/docker-compose-postgres.yml**

```
docker-compose -f docker-compose-postgres.yml up -d
```

Чтобы остановить, соответственно

```
docker-compose -f docker-compose-postgres.yml down
```

2.3 Установка и запуск вручную

Для установки Svacer скачайте и распакуйте архив дистрибутива (файл с названием вида **svacer_release-<version>.zip**).

При использовании PostgreSQL, предварительно настроенного на каком-либо хосте, необходимо использовать соответствующие параметры запуска. Допустим, PostgreSQL работает на локальном хосте, в нем создана БД **svacer_db**, доступная от пользователя **svacer_user** с паролем **svacer123**. В этом случае запустите Svacer со следующими параметрами

```
./bin/svacer server --pg  
postgres://svacer_user:svacer123@127.0.0.1:5432/svacer_db
```

Если вы запустили PostgreSQL в докер-контейнере, используя файл **./extra/docker-compose-postgres.yml**, то при запуске Svacer параметр **--pg** для подключения к БД можно не указывать, поскольку был создан пользователь и база, прописанные в Svacer как параметры по умолчанию.

После запуска сервера его веб-интерфейс будет доступен по адресу **http://localhost:8080**

По умолчанию web сервер запускается на интерфейсе **0.0.0.0:<port>**. Для указания отличного сетевого интерфейса пользователь может использовать опцию **--listen <network interface>**. По умолчанию grpc сервер запускается на интерфейсе **0.0.0.0:<port>**. Для указания отличного сетевого интерфейса пользователь может использовать опцию **--listen-grpc <network interface>**.

2.4 Hooks

Для получения данных Hooks при запуске сервера нужно указать опцию **--hooks <path to JSON file>**. Эта опция задаёт файл, содержащий информацию о расширении сервера посредством хуков, которые пользователь может вызывать из web-интерфейса.

Каждый хук соответствует некоторому процессу или скрипту, который сервер запускает в ответ на вызов соответствующей команды из web-интерфейса.

Формат файла Hooks следующий:

```
{
  "hooks": [
    {
      "id": "<id>",
      "label": "<label >",
      "target": "<target>",
      "input": ["<param1>", "<param2>", ...],
      "cmd": "<path to executable>",
      "args": ["<arg1>", "<arg2>", ...]
    },
    ...
  ]
}
```

Табл. 1 — Параметры Hooks

Параметр	Описание
id	Идентификатор хука. Должен быть уникальным в файле
label	Имя команды, которую пользователь видит в web-интерфейсе
target	Место в UI, в которое будет добавлена команда. Сейчас поддерживается только одно значение — default. Оно соответствует вкладке Details на правой панели пользовательского интерфейса
input	Параметры, которые можно передать в запускаемый процесс, исходя из контекста вызова команды в web-интерфейсе. Поддерживаемые значения: <ul style="list-style-type: none"> markerID — UUID идентификатор маркера; branchID — UUID идентификатор ветки; snapshotID — UUID идентификатор снимка;

	<ul style="list-style-type: none"> ▪ projectID — UUID идентификатор проекта; ▪ url — URL маркера в web-интерфейсе, на котором была вызвана команда; ▪ marker — будет заменен на полное имя временного файла, содержащего сериализованное в JSON представление маркера, который включает в себя его трассу и информацию о разметке
cmd	Полный путь к процессу, который будет запущен. Не должен включать аргументы запуска
args	Аргументы, передаваемые в запускаемый процесс. Полный список аргументов запускаемого процесса состоит из списка <args> и списка значений, соответствующих полю <input>

3 Использование

3.1 Получение справки

Запустив Svacer без параметров, можно получить информацию о доступных командах. Дополнительная информация о каждой команде доступна при использовании опции **--help**

```
svacer <command name> --help
```

Общий формат командной строки

```
svacer [global options] command [command options] [arguments...]
```

Глобальные опции относятся ко всем командам.

Опции команд имеют префикс «--». Их следует указывать до списка аргументов.

Для команд с опцией **--password** доступна возможность ввода пароля с клавиатуры.

3.2 Процесс импорта

Процесс импорта результатов работы статического анализатора Svace состоит из двух шагов:

1. Дедупликация данных, предобработка, конвертирование во внутренний формат и сохранение результатов в промежуточном хранилище (svacer import).
2. Загрузка данных из промежуточного хранилища на сервер (svacer upload).

3.2.1 Промежуточное хранилище

Промежуточное хранилище используется для хранения результатов, подготовленных для загрузки на сервер. При импорте данных в промежуточное хранилище снимкам назначается уникальный ID, задается имя проекта и ветки.

Промежуточное хранилище можно использовать для аккумуляции результатов нескольких запусков анализатора, в том числе для различных проектов.

По умолчанию, промежуточное хранилище создается в директории **.svacer-dir** рядом с директорией **.svace-dir**.

Параллельный доступ к хранилищу из разных экземпляров Svacer **не допускается**. Если необходимо использовать разделяемое хранилище и конкурентный доступ, Svacer следует запускать в режиме удаленного промежуточного хранилища.

Процедура импорта с промежуточным хранилищем по умолчанию (размещается в **<path to project>/svacer-dir**):

```
svacer import --svace <path to svace> <path to project>
```

где **<path to svace>** — путь к исполняемому файлу svace из дистрибутива svace, **<path to project>** — путь к проекту (директории, содержащей .svace-dir)

Процедура импорта с явным указанием размещения промежуточного хранилища:

```
svacer import --store <path to store> --svace <path to svace> <path to project>
```

3.2.2 Пользовательские атрибуты

При импорте данных можно ассоциировать пользовательские строковые атрибуты со снимком. Значения атрибутов доступно посредством публичного REST API и интерфейса командной строки. Это позволяет связать дополнительные данные с результатами (например commit ID). Пользовательские атрибуты нельзя модифицировать после импорта данных на сервер.

Пользовательские атрибуты экспортируются и импортируются как часть информации при использовании функциональности по экспорту и импорту снимков.

Пользовательский атрибут может быть передан при использовании команды `import` посредством опции `--field <name>:<value>`. Опция может быть указана несколько раз в командной строке.

```
svacer import ... --field f1:val1 --field f2:val2 ...
```

Если одно и тоже поле указано несколько раз, то значения поля сохраняются как JSON массив строк.

Допустимое имя пользовательского атрибута состоит из латинских букв, цифр, знаков подчеркивания или минуса.

Для получения списка всех пользовательских атрибутов в JSON виде используется следующая команда в командной строке

```
svacer access --project <project name> --branch <branch name> --snapshot <name or id> --user <user name> --password <password> custom_fields
```

Результат операции выдается как JSON объект в stdout. Объект имеет следующую структуру

```
{  
  "field name1": "value",  
  "field name2": ["value", "value"..]  
}
```

Для получения значения конкретного пользовательского атрибута используется следующая команда

```
svacer access --project <project name> --branch <branch name> --snapshot <name or id> --user <user name> --password <password> custom_fields --get <field name>
```

Значение атрибута выводится в stdout как JSON строка или массив.

3.2.3 Прикрепление файлов

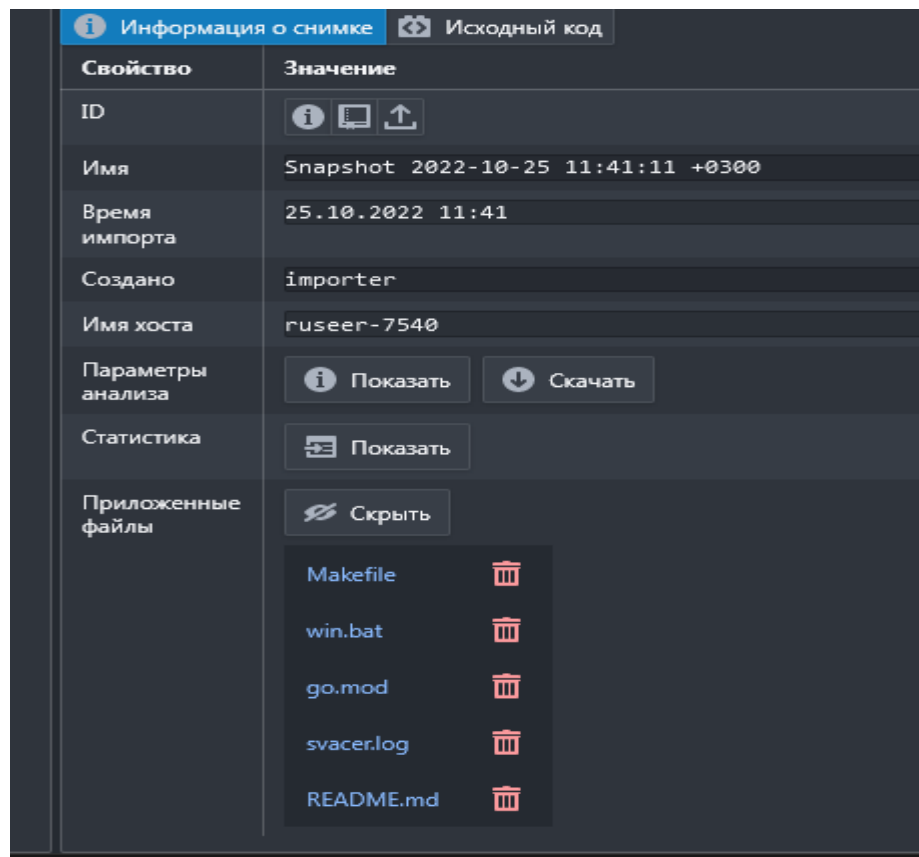
При импорте данных возможно прикрепление файлов размером до 300Мб к снимку. Файлы будут доступны пользователю в веб-интерфейсе и посредством интерфейса командной строки. Прикрепленные файлы фиксируются в момент операции импорта и не могут быть модифицированы после загрузки на сервер. Удалить файлы может администратор посредством веб интерфейса. Основное назначение данной функциональности прикрепление некоторой иммутабельной информации к результатам анализа для использования в интеграциях со сторонними системами.

Прикрепленные файлы являются частью данных, которые экспортируются и импортируются посредством функциональности экспорта и импорта снимков.

Для прикрепления файлов следует использовать опцию **--attach <file name>**

```
svacer import ... --attach file1.txt --attach file2.txt ...
```

Опция может быть указана несколько раз. После загрузки данных на сервер файлы будут видимы в панели информации о снимке



Для получения списка файлов из командной строки следует использовать команду **svacer access** --project <project name> --branch <branch name> --snapshot <name or id> --user <user name> --password <password> **attachments** --list

Результат работы команды представляет собой JSON массив объектов вида

```
[{
  "id": 11,
  "snapshot_id": "d99c23f7-9fb9-4f4e-85ac-2539328947e0",
  "lid": 122885,
  "short_name": "Makefile",
  "description": "/home/ruseer/static-analysis/svacer/Makefile",
  "created_by": "importer",
  "updated_by": null,
  "create_ts": "2022-10-18T06:41:54.532925Z",
  "update_ts": null,
  "checksum":
    "9b1e02a276e810829ac2e38fb775c4ff8d269d0f44fd0fbae40f79a0320389ad"
}]
```

Для получения содержимого файла из командной строки следует использовать команду

```
svacer access --project <project name> --branch <branch name> --snapshot <name or id> --user <user name> --password <password> attachments --get <id> <output file>
```

- Если **<output file>** не указан, то результат будет напечатан в stdout;
- Если **<branch>** не указан, то будет использован ветка master;
- Если **<snapshot>** не указан, то будет использован последний снимок на указанной ветке.

3.2.4 Загрузка данных на сервер

ВНИМАНИЕ: при загрузке данных на сервер, а также при прочих взаимодействиях клиента Svacer с сервером Svacer, крайне рекомендуется, чтобы версия клиента совпадала с версией сервера. В случае несовпадения версий возможны непредсказуемые ошибки.

Для загрузки данных из промежуточного хранилища на сервер используется команда **upload**:

```
svacer upload --user <user> --password <pwd> --host <host> --port <rest_port> --  
grpc <grpc_port> <path_to_store>
```

Если отсутствуют параметры:

- **--user** и **--password**, то используется специальный системный пользователь **importer** — он предназначен для импорта данных на сервер;
- **--host**, то используется **localhost**;
- **--port** и **--grpc**, то используются их значения по умолчанию;

При загрузке данных на сервер уже загруженные снимки повторно не загружаются. Разделяемая информация (такая как исходные файлы, где один и тот же файл может использоваться в различных анализах и т.п.) также повторно не загружается.

При сбоях в ходе загрузки проводится повторная загрузка, но при этом загружаются только недостающие данные.

В случае если Svacer запущен за reverse проху и доступен по протоколу HTTPS, то для загрузки требуется явно указать протокол в параметре **--host**, либо добавить опцию **--ssl**.

```
svacer upload --host https://<url_without_port> <path_to_store>
```

```
svacer upload --ssl --host <url_without_port> <path_to_store>
```

При этом grpc порт (по умолчанию — 3002), по которому идет загрузка данных, на сервере должен быть доступен.

Для информации о загрузке данных на сервер с авторизацией в LDAP см. раздел Использование CLI сервера Svacer с поддержкой LDAP.

При загрузке данных на сервер можно включить печать краткой статистики по загруженным результатам. Данная статистика включает информацию о новых и пропущенных предупреждениях, по сравнению с последним ранее загруженным снимком. Выдача краткой статистики возможно в текстовом или JSON виде. Результаты печатаются в stdout.

```
svacer upload ... --quickStats --quickStatsFormat=text ...
```

Пример вывода

Quick Stats:

```
Previous snapshot : Snapshot 2022-11-16 10:11:37 +0300
Current snapshot  : Snapshot 2022-11-16 10:23:08 +0300
New                : 1
Missing            : 0
Matched            : 1
Same               : 0
```

```
svacer upload ... --quickStats --quickStatsFormat=json ...
```

Пример вывода

```
{"branch": "391f21d1-2173-449c-9d1a-dd846d2a6471", "snapshot": "34a1125d-86a3-45d3-9b41-cb5da24ab96e", "snapshotName": "Snapshot 2022-11-21 10:19:06 +0300", "lastSnapshot": "59657ae8-c3b6-48bc-b586-a6c0580cdb50", "lastSnapshotName": "Snapshot 2022-11-21 10:18:09 +0300", "same": 2}
```

Формат JSON определяется protobuf3 схемой:

```
message ImportSummaryContext {
  string branch = 1;
  string snapshot = 2;
  string snapshotName = 9;
  string lastSnapshot = 3;
  string lastSnapshotName = 4;
  int64 new = 5;
  int64 missing = 6;
  int64 matched = 7;
  int64 same = 8;
}
```

По умолчанию, если опция по указанию формата выдачи отсутствует, выдача идет в текстовом режиме.

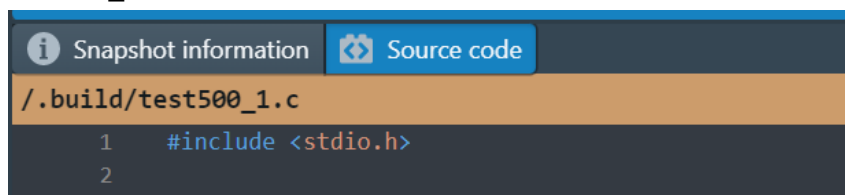
3.2.5 Модификация путей в Review (разметке) при импорте

3.2.5.1 Обычное поведение

По умолчанию пути к файлам не меняются, за исключением пути к папки **.svace-dir**, он будет заменён на **.build**. Эту информацию можно увидеть при импорте проекта:

```
svacer import --clean .svacer-dir
...
info      The path prefix of .svace-dir 'C:\work\generate_files\gen_exp' will be
trimmed when converting file paths
...
```

После этого все маркеры из этой папки и дочерних будут в **Review** показывать путь вида **/.build/test500_1.c** вместо полного пути для файла **C:\work\generate_files\gen_exp\test500_1.c**.



3.2.5.2 Использование --pathPrefix

При необходимости модификацию по умолчанию можно заменить через **--pathPrefix**. Значения опции передаются в виде:

```
--pathPrefix "prefix_for_replace1:value1;prefix_for_replace2:value2..."
```

Поддерживается как передача данных сразу в опцию, так и через указание файла содержащего значения. Разумно делать замены для унификации вида проекта, собранного на разных машинах (или из разных папок). При использовании файла допустимо размещать каждую замену на отдельную строчку.

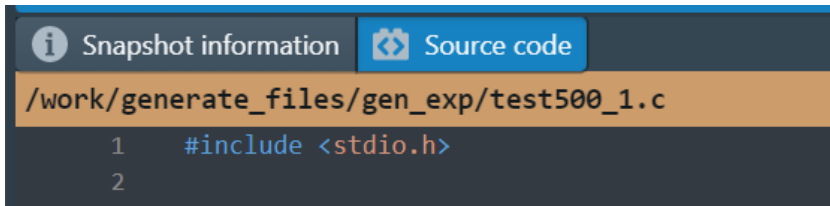
Префиксы путей заменяются только при полном совпадении (включая регистр, даже на Windows).

Для указания двоеточия в префиксе пути используется удвоенное двоеточие, к примеру (для показа преобразований добавлено **--debug**):

```
svacer --debug import --clean --upload --pathPrefix "C::\work:work" .svacer-dir
...
INFO      Creating regular object store
```

```
DEBUG    Creating path mapping rule 'C:\work' => 'work'
DEBUG    Creating path mapping rule 'C:/work' => 'work'
DEBUG    Creating path mapping rule '/C_/work' => 'work'
...
```

После импорта на сервере можно увидеть файл: `/work/generate_files/gen_exp/test500_1.c`



3.2.5.3 Использование `--buildObject @auto`

При импорте конкретного файла `*.svres` исходный код может быть загружен, при указании опции `--buildObject @auto`. В случае, если этот файл лежит в папке `.svace-dir`, исходный код будет взят из нее. При отсутствии папки `.svace-dir` файлы будут браться по абсолютным путям из файла `*.svres`.

Для указания конкретных директорий поиска подходящих файлов из `*.svres` следует использовать опцию `--source-tree`, которую можно указать несколько раз, чтобы задать несколько корневых директорий. Поиск в этих директориях будет происходить только в случае, когда по путям из файла `*.svres` были найдены не все файлы. В случае успешного сопоставления, на сервер будут загружены соответствующие файлы из директории `source tree`.

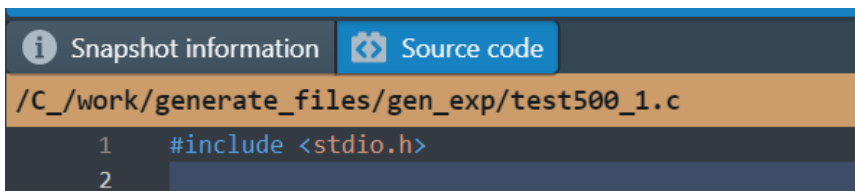
3.2.5.4 Использование `--fullPaths`

Допустимо также полное отключение модификаций путей, для этого импорт надо производить с опцией `--fullPaths`. Опция блокирует как модификацию по умолчанию `.build`, так и явные указания через `--pathPrefix`.

```
svacer --debug import --clean --upload --fullPaths .svacer-dir
```

После загрузки на сервере будет виден только полный путь:

`/C_/work/generate_files/gen_exp/test500_1.c`.



Пути из Windows преобразуются в стиль Linux с небольшим преобразованием символов.

3.3 Создание PDF отчета для проекта

Для создания PDF отчета по проекту из интерфейса командной строки используется команда **pdfgen**. Назначение команды состоит в том, чтобы добиться из консоли результатов (создание PDF файла отчета), аналогичным следующим действиям пользователя в GUI:

1. Выгрузка таблицы маркеров в формате PDF (Рис. 1)
2. Предварительное формирование содержимого таблицы маркеров
 - По режимам сравнения (new, missing, same, matched) с указанием цели и источника сравнения, Рис. 2
 - По используемым фильтрам Custom (ограниченный набор из 4 параметров) Рис. 3

Ниже приводятся фрагменты GUI, которым соответствует cli команда в различных вариантах ее использования:

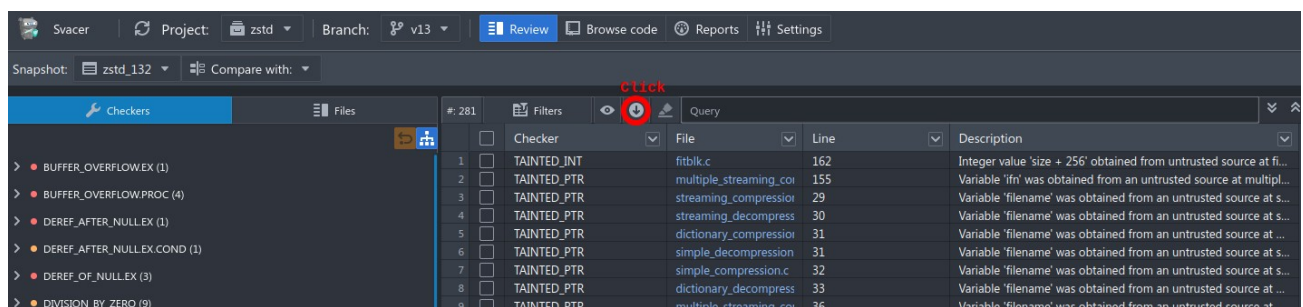


Рис. 1 — Отчет без сравнения

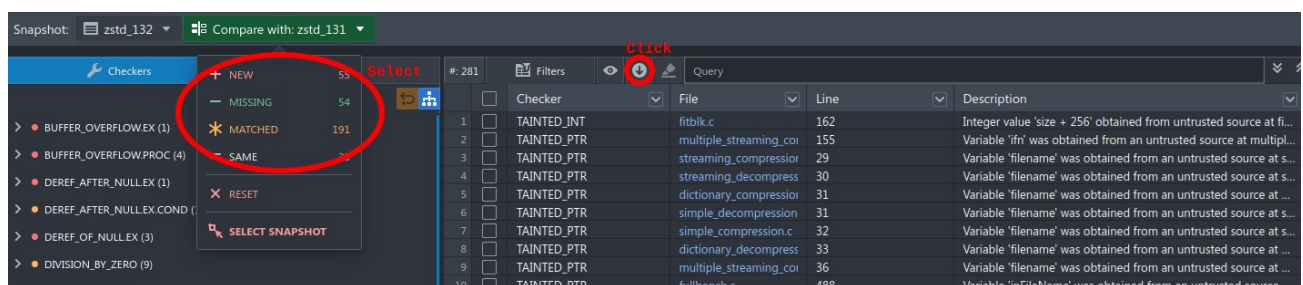


Рис. 2 — Отчет со сравнением

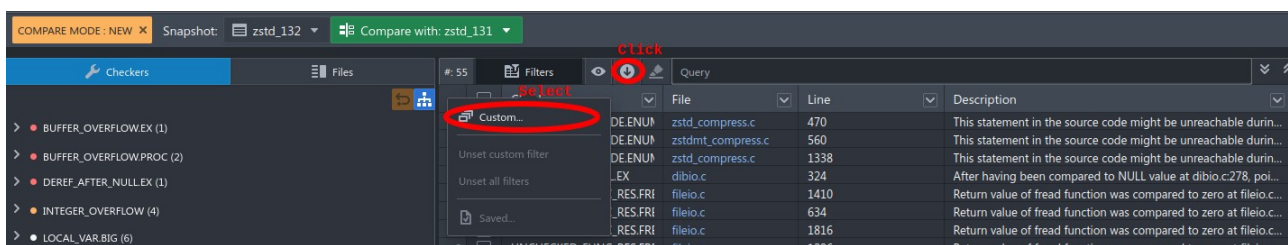


Рис. 3 — Отчет со сравнением и фильтрами

Общий вид команды следующий:

```
svacer pdfgen --cmpMode [none|new|missing|matched|same] --project [name|id] --branch [name|id] --snapshot [name|id] [--targetProject [name|id] --targetBranch [name|id] --targetSnapshot [name|id]] [--file [re_exp] --checker [re_exp] --severity [re_exp] --review [re_exp]] --outFile report_name --tz time_zone --lang report_lang
```

Описание параметров:

- **cmpMode** — параметры сравнения. Возможные значения: **none** — без сравнения, **new** — новые, **missing** — отсутствующие, **matched** — сопоставленные, **same** — одинаковые
- **Project, Branch, Snapshot** — параметры, описывающие снимок, отчет для которого требуется создать. В качестве значений могут использоваться как имена, так и идентификаторы соответствующих сущностей. Например: `--project zstd --branch "7683ed6a-b838-4090-9945-10e148f94be3" --snapshot zstd_130`
- **targetProject, targetBranch, targetSnapshot** — параметры, описывающие снимок, с которым необходимо провести сравнение (обязательные параметры для режимов `cmpMode: new, missing, matched, same`).
- **File, checker, severity, review** — параметры, задающие фильтры, применяемые к списку маркеров. Фильтры задают условия включения маркера в отчет. Значения фильтров задаются в формате регулярных выражений. В случае неверного формата, ошибочный фильтр не будет применяться и операция экспорта будет выполнена, как если бы фильтр не был указан. **File** — фильтр по пути к файлу, где был обнаружен маркер. **Checker** — фильтр по чекеру, предупреждение от которого было создано. **Severity** — серьезность сработавшего чекера. **Review** — статус разметки (`confirmed`, `unclear` и т.д.)
- **outFile** — имя создаваемого файла отчета. Если не будет указано расширение, будет добавлено расширение `pdf`
- **tz** — временная зона в минутах. Используется при формировании записи о комментариях, которые оставил пользователь, производивший разметку
- **lang** — язык, который будет использован в создаваемом отчете (`en` — английский, `ru` — русский)

Для создания отчета без использования режима сравнения, необходимо указать cmpMode в значение none (это значение используется по умолчанию).

Для предварительного формирования таблицы маркеров с учетом сравнения, использовать значения cmpMode [new, missing, same, matched]. Названия этих режимов аналогичны названиями в GUI. При этом требуется указание как минимум одного параметра из списка: targetProject, targetBranch, targetSnapshot. Если какие-то параметры target* не указаны, они будут установлены в значение исходных (project → targetProject, branch → targetBranch, ...). Сформированные на предварительном этапе маркеры, проходят механизм фильтраций (параметры: severity, file, review, checker), после чего происходит создание отчета.

Для создания отчета можно также использовать public API. Необходимо использовать следующие параметры:

URL — /api/public/exportPDF

Метод — POST

Тело запроса — формат JSON, имеет следующий вид:

```
{
  "compare_mode": "new",
  "context": {
    "project": "zstd",
    "branch": "v13",
    "snapshot": "zstd_131"
  },
  "target_context": {
    "project": "zstd",
    "branch": "v13",
    "snapshot": "zstd_132"
  },
  "language": "en",
  "timezone": 180,
  "filters": {
    "checker": "^Z.*$",
    "file": ".*example.*",
    "severity": "Cr.*",
    "review": "Conf.*"
  }
}
```

Данный запрос соответствует запросу на создание отчета, который получается в результате сравнения в рамках проекта zstd и ветки v13 двух снимков zstd_131 и zstd_132. При этом будут выбраны только новые маркеры. Среди выбранных останутся только те, что удовлетворяют условиям фильтров, а именно: чекер начинается с буквы Z, файл содержит слово example, серьезность чекера содержит подстроку Cr (что в силу ограниченного

количества значений для данного фильтра соответствует значению Critical) в названии, а состояние разметки содержит подстроку Conf (что в силу ограниченного количества значений для данного фильтра соответствует значению Confirmed). Язык отчета — английский, часовой пояс — 180 минут (3 часа).

4 Миграция

4.1 Миграция со старого сервера

В данном подразделе описана миграция со старого сервера, поставляющегося в комплекте со Svace. Это не относится к миграции между версиями Svacer. Миграция данных со старого сервера на Svacer состоит из двух этапов:

1. Экспорт информации со старого сервера.
2. Импорт данных в новый.

Для экспорта информации со старого Svace сервера используется утилита **svace-migration.jar**

```
java -jar svace-migration.jar <user> <password> <path to svace> <path to server dir> <path to output dir>
```

Табл. 2 — Параметры, используемые при экспорте данных

Параметр	Описание
<user>	Имя пользователя на старом сервере
<password>	Пароль пользователя на старом сервере
<path to svace>	Путь к директории, где находится Svace
<path server dir>	Директория, из которой запущен старый сервер
<path to output dir>	Директория, в которую будут записаны экспортируемые данные

Экспорт данных возможен при работающем старом сервере. Экспорт инкрементальный, поэтому ранее экспортированные результаты повторно не переносятся.

При экспорте утилита **svace-migration.jar** извлекает:

- все снимки в виде *.svres;
- информацию о размещении исходного кода;
- сопутствующую информацию в директории старого сервера.

ВНИМАНИЕ: Для работы утилиты требуется Java 17+, при запуске возможно потребуется добавить опции JVM:

```
--add-opens java.base/java.lang.reflect=ALL-UNNAMED
--add-opens java.base/java.lang=ALL-UNNAMED
--add-opens java.base/java.io=ALL-UNNAMED
--add-opens java.base/java.util=ALL-UNNAMED
--add-opens java.base/java.util.concurrent=ALL-UNNAMED
--add-opens java.base/java.text=ALL-UNNAMED
--add-opens java.desktop/java.awt.font=ALL-UNNAMED
```

Для импорта экспортированных данных используется следующая команда:

```
svacer migrate [--skip-dxr-errors] --distr <path to svace distr> --temp <path to temp folder> --store <intermediate store> <path to exported data>
```

Табл. 3 — Параметры, используемые при импорте данных

Параметр	Описание
--skip-dxr-errors	Опциональный параметр для игнорирования ошибок в DXR-данных. Рекомендуется к использованию
--distr	Путь к дистрибутиву Svace
--temp	Директория для хранения временных файлов в ходе конверсии
--store	Путь к промежуточному хранилищу
<path to exported data>	Путь к директории, в которую были экспортированы результаты

После конвертации данных в промежуточное хранилище результат можно загрузить на новый сервер с помощью команды upload:

```
svacer upload --user <user> --password <pwd> --host <host> --port <rest port> --grpc <grpc port> <path to store>
```

4.2 Миграция между версиями Svacer

Миграция БД PostgreSQL и object store между версиями Svacer происходит автоматически при обновлении на новую версию. Рекомендуется делать резервные копии перед обновлением (см. раздел Создание и восстановление резервной копии).

Ниже описаны существенные изменения при переходе на определенную версию с более старой

- 4.0.1 и выше — миграция object store (badger) на новую версию. Может занимать существенное время при большом количестве проектов в Svacer. После перехода на версию 4.0.1 и выше object store становится несовместим с более ранними версиями
- 5.0.0 и выше — перенос DXR разметки из PostgreSQL в object store. Версии 5.0.0 и выше не совместимы с более ранними версиями.

4.3 Создание и восстановление резервной копии

4.3.1 Если PostgreSQL запущен в виде сервиса

Если PostgreSQL запущен в виде сервиса (т.е. не в докер-контейнере), то для создания резервной копии БД перейдите в пользователя postgres и запустите создание дампа:

```
sudo su -l postgres
```

```
pg_dump --clean -T tmp_diff* svacer_db > svacer.sql
```

где `svacer_db` — имя БД.

В результате дампа БД будет создан в файле `svacer.sql`

Для восстановления БД из дампа также перейдите в пользователя postgres и запустите восстановление

```
sudo su -l postgres  
psql -f svacer.sql svacer_db
```

4.3.2 Если PostgreSQL запущен в docker-контейнере

Если PostgreSQL запущен в docker-контейнере, то для создания резервной копии БД выполните следующую команду

```
docker exec -t svacer_postgres pg_dump --clean -T tmp_diff* -U postgres  
svacer_db > svacer.sql
```

где:

- **svacer_postgres** — имя docker-контейнера
- **postgres** — имя пользователя, под учётной записью которого создана БД
- **svacer_db** — имя БД

Для восстановления БД используйте команду

```
cat svacer.sql | docker exec -i svacer_postgres psql -U postgres svacer_db
```

4.3.3 Создание резервной копии svacer object store через command line API

Резервную копию object store можно создать, используя следующую команду (доступна только пользователю с ролью admin)

```
svacer server backup --user admin --password <password> --file <filename>.kvbak
```

В результате object store будет сохранен в файл **<filename>.kvbak**

Для восстановления из резервной копии требуется при запуске сервера указать этот файл в опции **--load-from-backup**.

```
svacer server --pg <postgres_url> --store /home/svacer/new-store --load-from-backup  
<filename>.kvbak
```

Восстановление возможно только в пустой object store. То есть в данном примере директория **/home/svacer/new-store** должна быть пустой.

4.3.4 Создание резервной копии svacer object store в сервере PostgreSQL

Резервную копию object store можно сохранить в сервере PostgreSQL, который используется сервером Svacer. Данный механизм использует функциональность large object store из PostgreSQL. Функциональность по бэкапу object store в PostgreSQL является **рекомендуемой стратегией** по бэкапу object store.

Следующие команды подразумевают, что сервер развернут с портами по умолчанию на localhost. При необходимости, порты и адрес сервера могут быть указаны как дополнительные опции.

Создание бэкапа:

```
svacer server backup --user admin --password <password> --pg
```

Бэкап создается в фоновом процессе, пользователь может продолжать работать с сервером. Результат работы команды состоит из ID бэкапа и времени создания. Эти данные печатаются в stdout

```
ID = 23 CREATE_TS = 2022-10-06T06:04:33Z
```

Для печати списка существующих бэкапов используется команда:

```
svacer server backup --user admin --password <password> --list-pg
```

Результат печатается в stdout и включает ID и время создания бэкапа

```
2022-10-06T09:04:36.929+0300      info      Querying server configuration from
http://localhost:8080/api/public/server/info
2022-10-06T09:04:36.930+0300      info      Server configuration grpc = 3002, rest =
8080
2022-10-06T09:04:36.939+0300      info      Logged by user admin
2022-10-06T09:04:36.941+0300      info      Backups:
ID = 23 CREATE_TS = 2022-10-06T06:04:33Z
```

Восстановление object store из бэкапа допустимо только при старте сервера. Для этого следует очистить текущий object store (или указать путь на новую директорию) и использовать следующий флаг при старте:

```
svacer server --load-from-pg-backup=<id>
```

где <id> — ID бэкапа или 0, для использования последнего доступного бэкапа.

Для очистки старых бэкапов используется следующая команда:

```
svacer server backup --user admin --password <password> --drop-  
pg=<lower>:<upper>
```

где <lower> и <upper> нижняя и верхняя граница ID бэкапов.

Настоятельно рекомендуется настроить задачу на хосте, где работает Svacer на периодический запуск бэкапа object store в PostgreSQL.

4.3.5 Создание резервной копии svacer object store вручную

Создать резервную копию object store также возможно вручную. Для этого достаточно скопировать директорию, в которой object store находится. Перед тем как это делать желательно остановить сервер Svacer в целях обеспечения целостности данных.

Если директория, где хранить **object store** не была явно указана при запуске сервера, по умолчанию он будет создан в следующей директории:

- `~/.cache/svacer` — в ОС Linux
- `%LocalAppData%\svacer` — в ОС Windows
(например, `C:\Users\myusername\AppData\Local\svacer`)

При восстановлении из резервной копии следуют поместить файлы object store в директорию по умолчанию (см. выше), либо же в другую директорию и при запуске Svacer указать путь к ней одним из способов:

1. Перед запуском Svacer установить переменную окружения `SVACER_OBJECT_STORE`, например:
`export SVACER_OBJECT_STORE=/home/user/data/object_store`
2. При запуске Svacer передать путь к object store в параметре `--store`, например:
`svacer --store /home/user/data/object_store`

При запуске Svacer в docker-контейнере требуется смонтировать директорию с object store в контейнер как volume и указать путь к ней в переменной **STORE** в файле **docker-compose.yml**. Допустим, директория с object store была скопирована в `/home/user/data/object_store`. В таком случае параметры нужно указать следующим образом

```
volumes:  
  - /home/user/data:/data  
environment:  
  - STORE=/data/object_store
```

4.4 Перенос снимков

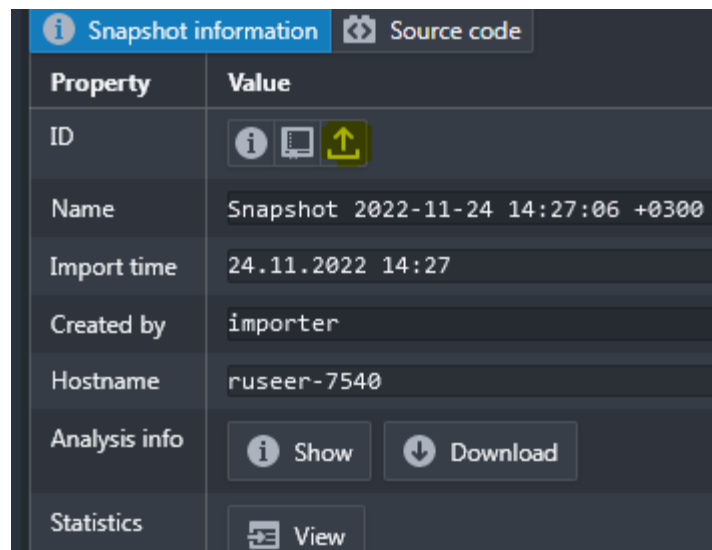
Svacer предоставляет возможность по экспорту и импорту снимков посредством интерфейса командной строки и веб-интерфейса.

Экспорт снимка включает все предупреждения, информацию о снимке, прикрепленные файлы, объект сборки, пользовательские атрибуты, разметку и комментарии. Подавленные предупреждения также включаются в экспортированную информацию и импортируются в статусе подавленных.

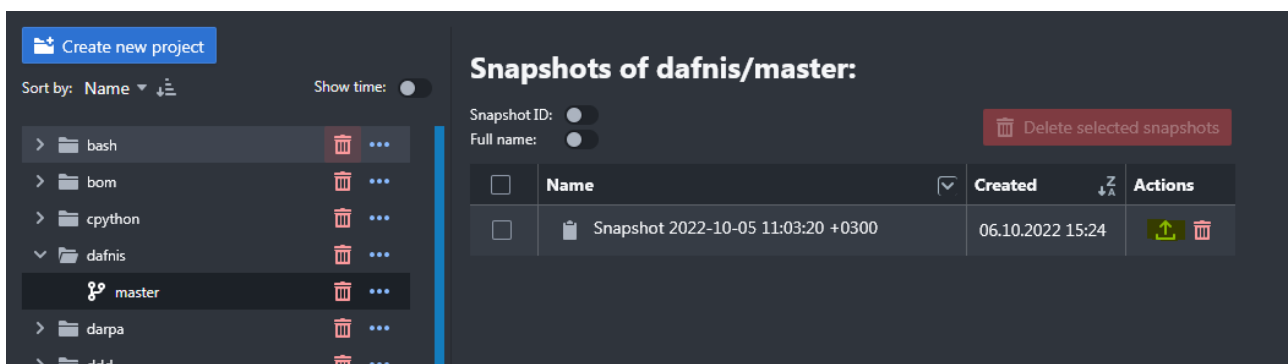
Внимание: в текущей версии экспорт снимка из веб-интерфейса доступен всем пользователям с ролью READ для выбранного проекта/ветки. Импорт снимка из веб-интерфейса доступен только пользователям с ролью admin (обычный пользователь не видит секцию с проектами). Экспорт и импорт снимков из интерфейса командной строки доступен пользователям с ролями READ и WRITE для выбранной ветки.

4.4.1 Перенос из веб интерфейса

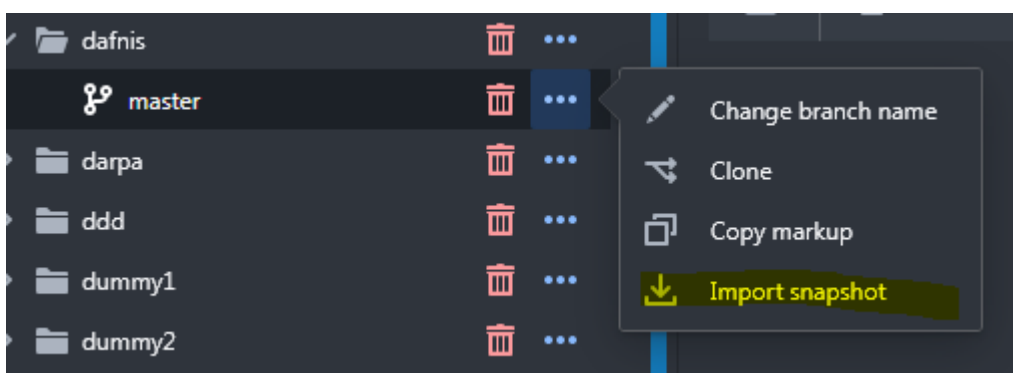
Для экспортирования снимка из пользовательского интерфейса пользователь может использовать кнопку экспорта на панели информации о снимке:



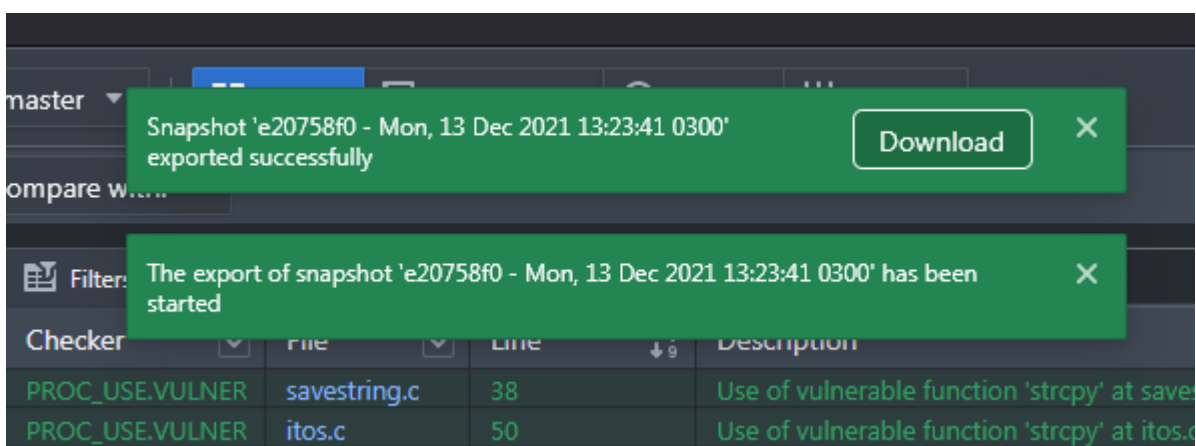
либо использовать соответствующую функциональность из панели настроек проектов (требуется роль admin):

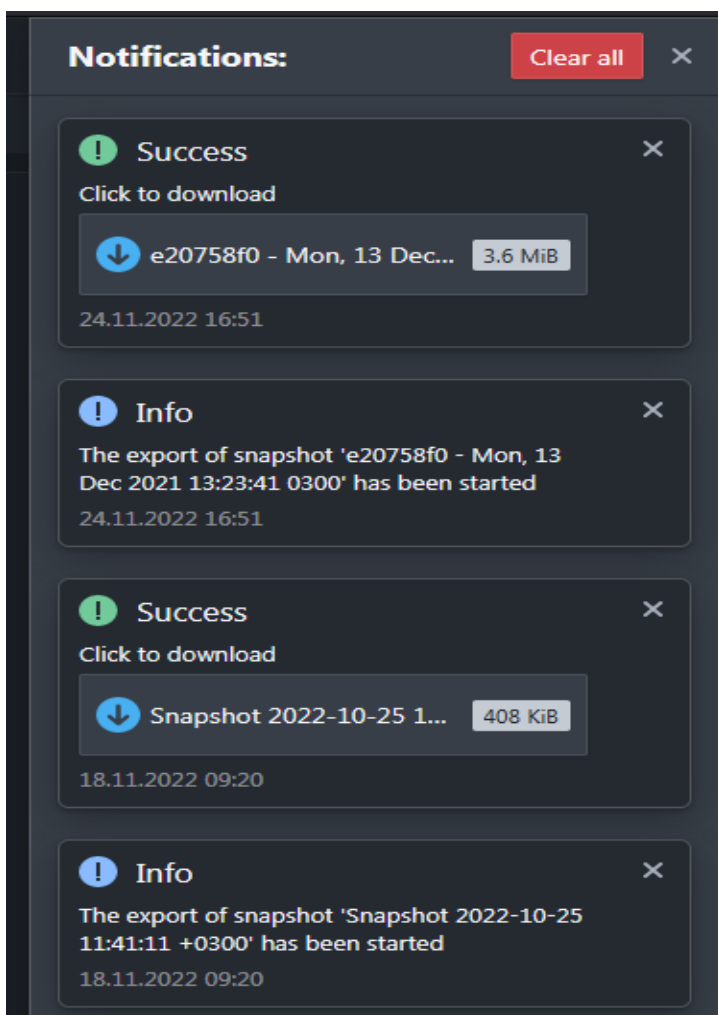


Импортирование снимка через пользовательский интерфейс допустимо только для пользователя с ролью «admin». Команда доступна в меню ветки



Задача по экспорту снимка может занимать значительное время (зависит от размера снимка) и выполняется асинхронным образом. Извещение о завершении задачи по экспорту появится в виде всплывающей зеленой панели а так же будет доступно в панели уведомлений:





4.4.2 Перенос из командной строки

Экспорт и импорт снимков из командной строки доступен пользователям с ролями READ и WRITE на проекте или ветке.

Экспорт снимков:

svacer server export

- user <user>**
- password <pwd>**
- host <host>**
- port <port>**
- grpc <grpc port>**
- project <project name or id>**
- branch <branch name or id>**
- snapshot <snapshot name or id>**
- <output file name>**

Импорт снимков:

svacer server import

- user <user>**

```
--password <pwd>
--host <host>
--port <port>
--grpc <grpc port>
--project <project name or id>
--branch <branch name or id>
--name <new name for snapshot>
[--force]
<input file name>
```

Опция **--force** создаст проект и ветку, если их не было на сервере.

4.5 Перенос Svacer с одного сервера на другой

1. Создайте резервную копию БД и object store в соответствии с тем, как описано в разделе Создание и восстановление резервной копии.
2. Перенесите созданные файлы резервных копий на нужный сервер
3. Восстановите данные из резервной копии на новом сервере

5 Работа с сервером

5.1 Вход в сервер

Для входа в сервер введите логин и пароль на странице ввода учетных данных (Рис. 4).
Учётные данные по умолчанию — admin / admin.

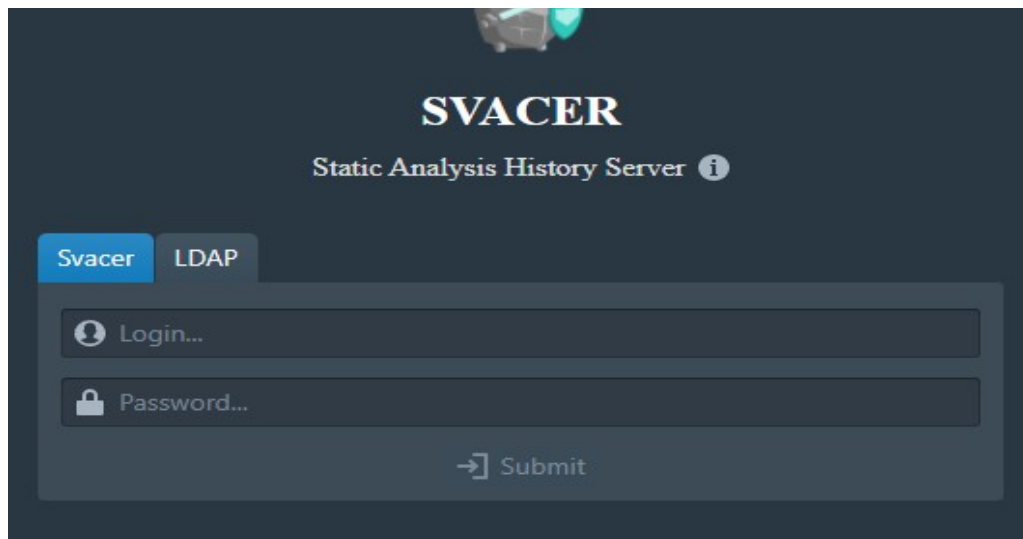


Рис. 4 — Страница ввода учетных данных

В случае конфигурации сервера в режиме поддержки протокола LDAP возможна аутентификация пользователей с помощью внешних серверов. При этом, если настроено более одного LDAP-сервера для авторизации, необходимо выбрать нужный из выпадающего списка (Рис. 5).

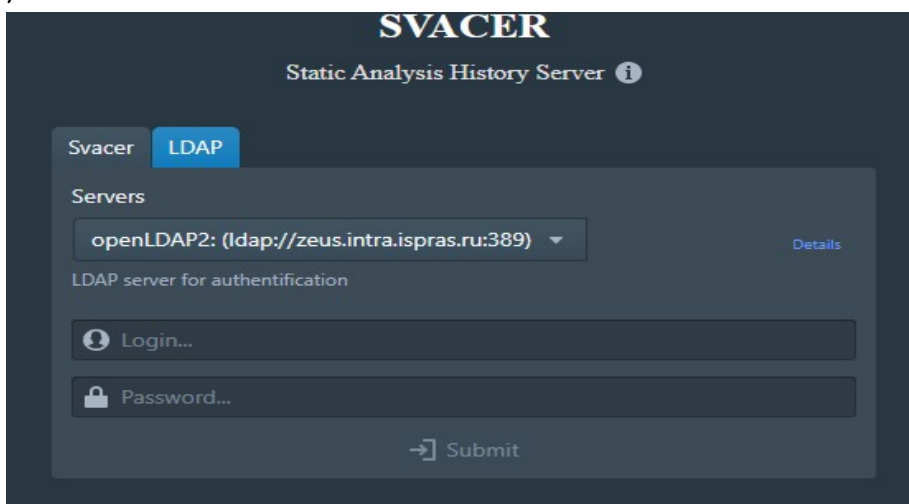


Рис. 5 — Страница входа по протоколу LDAP

5.1.1 Профиль пользователя

В правом верхнем углу, при нажатии на имя пользователя, в выпадающем меню есть возможность выбрать **Profile** для редактирования своего профиля (Рис. 6).

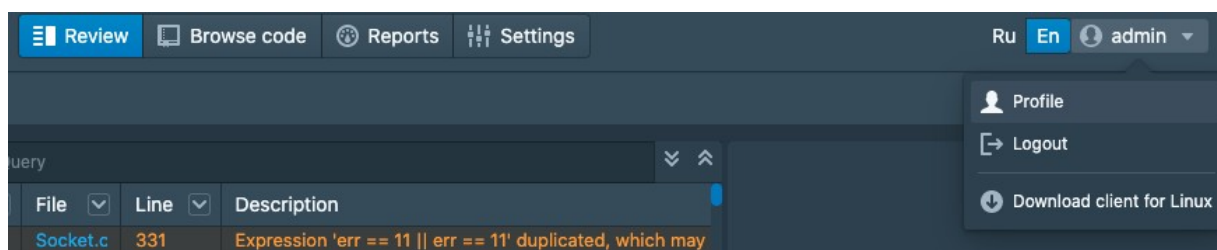


Рис. 6 — Переход в профиль пользователя

При нажатии пользователь имеет возможность изменить свои личные данные. В этот диалог также можно попасть через **Settings > Profile**.

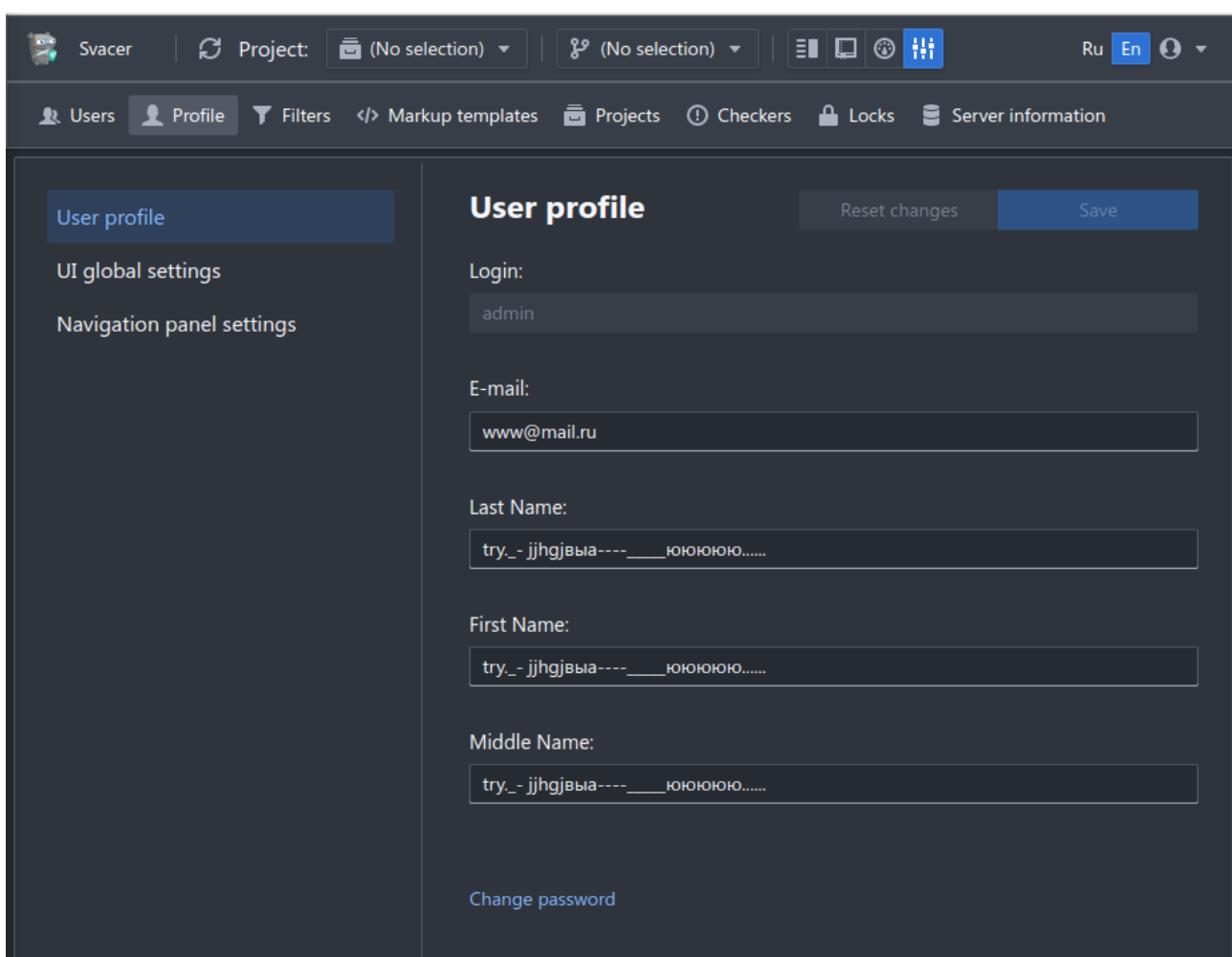


Рис. 7 — Изменение профиля пользователя

Здесь пользователю доступно 3 вкладки: **User profile**, **UI global settings**, и **Navigation panel settings** (Рис. 7).

Во вкладке **User profile** пользователь может изменить свои данные, все кроме логина.

Во вкладке **UI global settings** предоставляется возможность установить пользовательские настройки. Они нужны, чтобы при обновлении страницы или при выходе и повторном входе под одним аккаунтом некоторые параметры не сбрасывались. Например, при установке **Save filters** в положение, как на Рис. 8, пользовательские фильтры не будут сбрасываться при обновлении страницы и при переходе с одних проекта/ветки/снимка на другие.

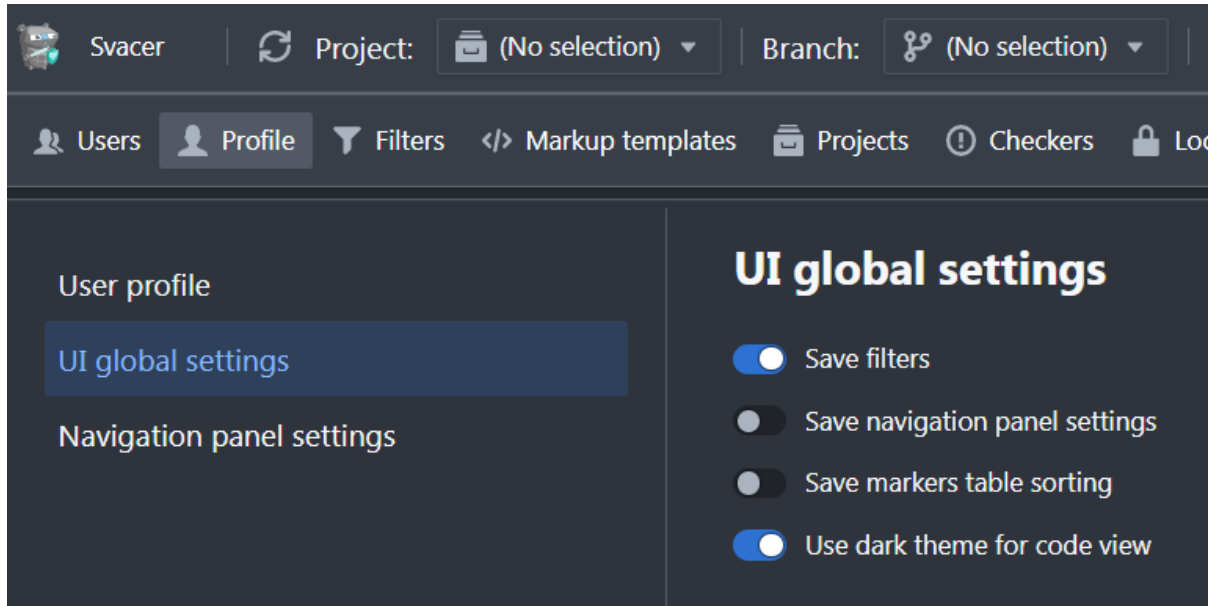


Рис. 8 — Общие настройки интерфейса

Вкладка **Navigation panel settings** позволяет настроить отображение файлов и критерий сортировки полей панели навигации **Review > Files** в левой части экрана. Данная вкладка является аналогом вкладки **Navigation panel settings**, доступной в панели навигации **Review > Files** (Рис. 9). Данные настройки сохраняются для пользователя и не сбрасываются при новой сессии.

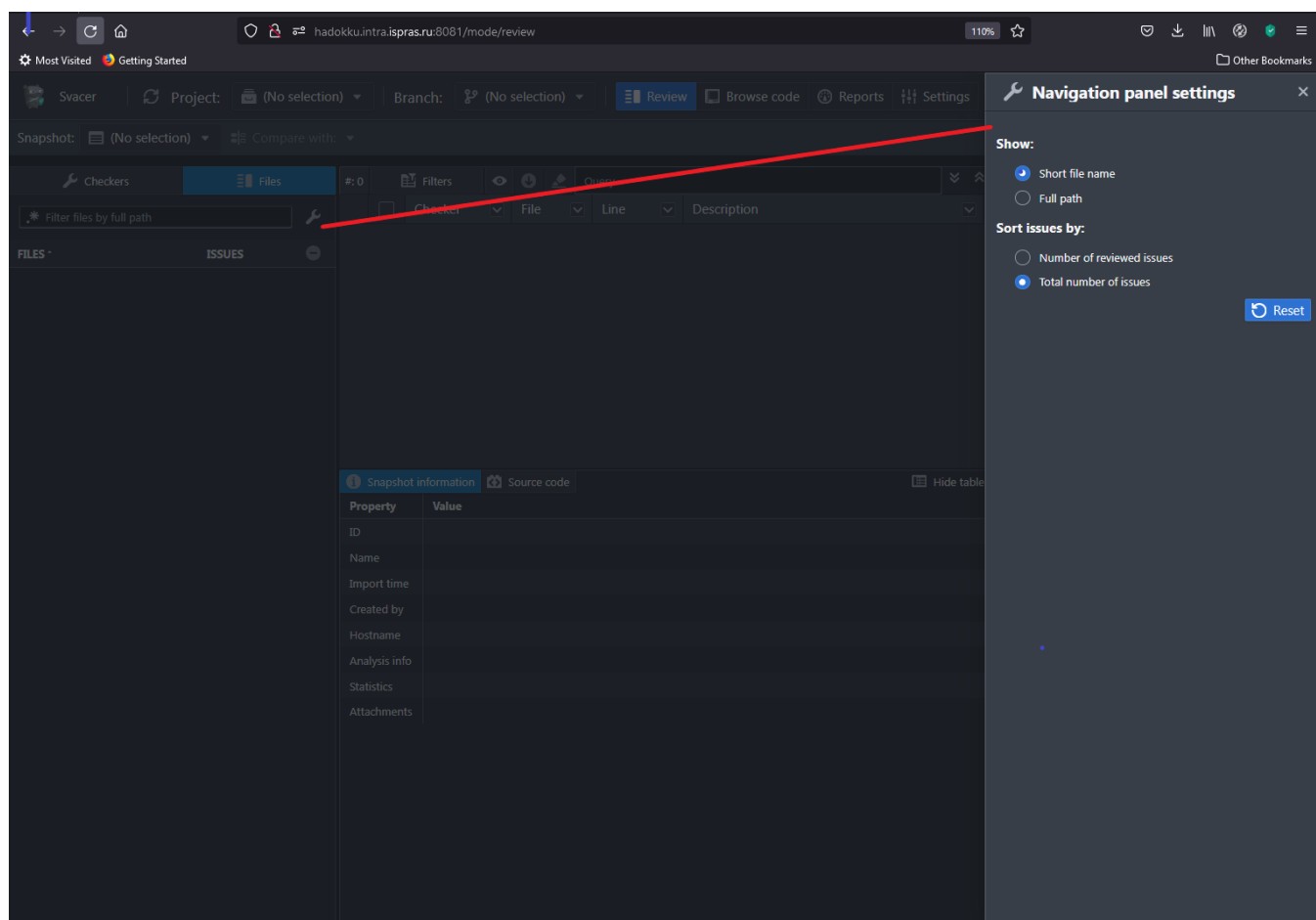


Рис. 9 — Настройки панели навигации

Настройка панели навигации позволяет указать в каком виде показывать пути к файлам и задать порядок сортировки по умолчанию.

5.1.2 Организации


Пользователи могут быть связаны с организациями. Реестром организаций может управлять только администратор сервера. Пользователь может принадлежать нескольким организациям. Организации носят информационный характер и не влияют на права пользователей в системе.

На странице **Settings > Projects** в таблице отображаются все организации, которые были добавлены в систему. Доступен поиск по имени организации и сортировка по колонкам **Name** и **Short name**.

Чтобы добавить новую организацию в реестр

1. Нажмите кнопку **Add organization**

2. Заполните поля появившейся формы
3. Нажмите кнопку **Add**

 **Add organization** ×


Name: *


Short name:

Description:

Cancel Add

Чтобы изменить данные существующей организации

1. Нажмите кнопку  в столбце **Actions**
2. В появившейся форме отредактируйте данные организации
3. Нажмите кнопку **Save**

 **Edit organization** ×

Name: *

Short name:


Description:


Organization description

Cancel

Save

Чтобы удалить организацию из реестра:

1. Нажмите кнопку  в столбце **Actions**
2. В появившемся диалоге подтвердите удаление, нажав на кнопку **Delete**

 **Delete organization** ×

Do you want to delete organization with name "ООО Высокие технологии"?

Cancel

Delete

5.2 Выбор проекта и снимка

После успешной авторизации выберите проект, ветку и снимок для просмотра результатов работы анализатора Svace (Рис. 10).

По умолчанию после выбора проекта автоматически выбирается ветка master и самый свежий снимок. Пример выбранных проекта, ветки и снимка представлен на Рис. 11.

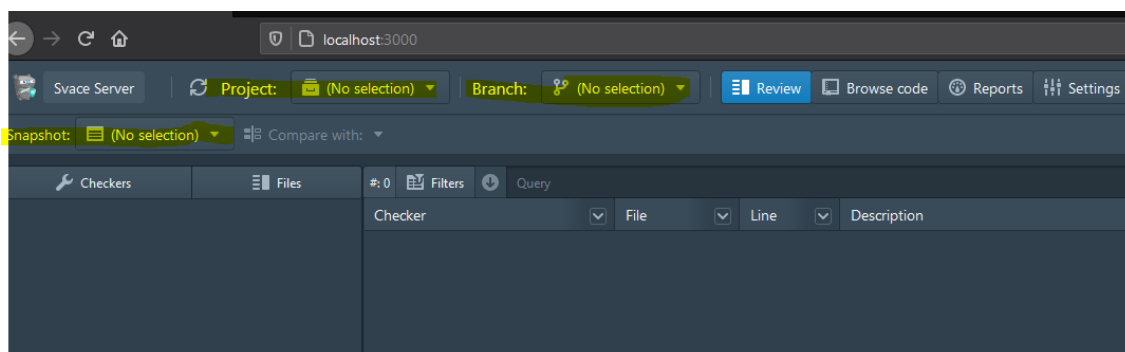


Рис. 10 — Проект, ветка и снимок в интерфейсе

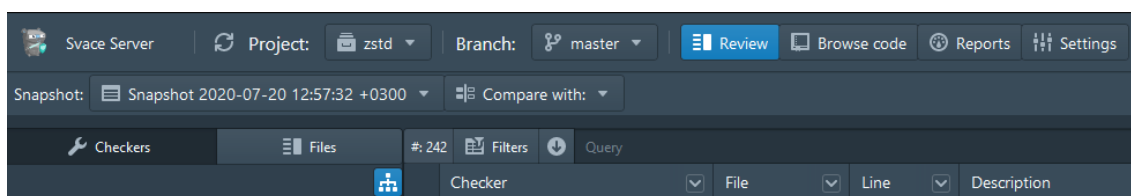


Рис. 11 — Пример выбранных проекта, ветки и снимка

5.3 Виды интерфейса

Сервер историй предоставляет несколько режимов работы интерфейса, каждый из которых предназначен для выполнения определённых действий:

- **Review** — просмотр и разметка найденных предупреждений, а также сравнение снимков или отдельных предупреждений;
- **Browse Code** — просмотр снимков исходного кода, связанного с результатами анализатора Svace;
- **Reports** — формирование отчетов на основе хранимой информации;
- **Settings** — управление сервером и просмотр его состояния;

В каждом режиме интерфейс предоставляет набор панелей, которые позволяют выполнять различные действия.

5.4 Элементы управления в режиме разметки

Режим разметки предоставляет следующий набор элементов управления:

1. Левая группа панелей:

1) Панель **Checkers** — показывает список чекеров Svace, сработавших в выбранном снимке (Рис. 12). При нажатии на один из них, будет установлен фильтр таблицы предупреждений на соответствующий checker/severity. Критичность (severity) чекера обозначается цветом при древовидном отображении и иконкой соответствующего цвета при отображении в виде списка.

Кнопкой **Unset selection** (↶) пользователь может отменить фильтры, примененные из левой группы панелей.

Кнопкой **Group by** (📁) пользователь может выбрать показ списка чекеров в виде дерева, группирующего чекеры по **checker severity** (Рис. 13).

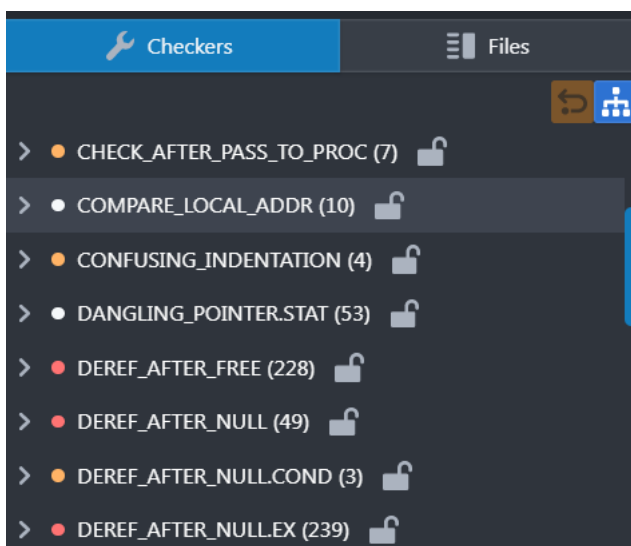


Рис. 12 — Панель Checkers. Данные в виде списка

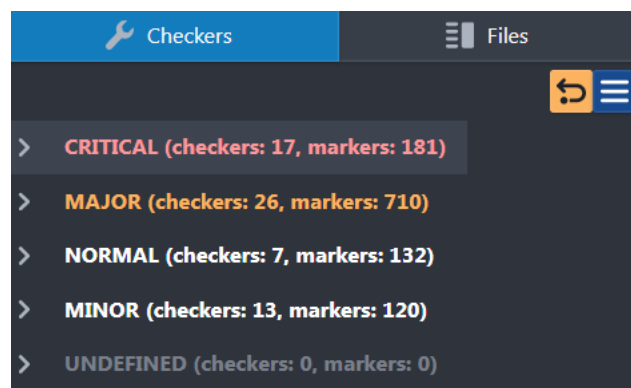


Рис. 13 — Панель Checkers. Данные в виде дерева

2) Панель **Files** — показывает список файлов, для которых есть какие-либо предупреждения (Рис. 14). Столбец **ISSUES** показывает число предупреждений в файле.

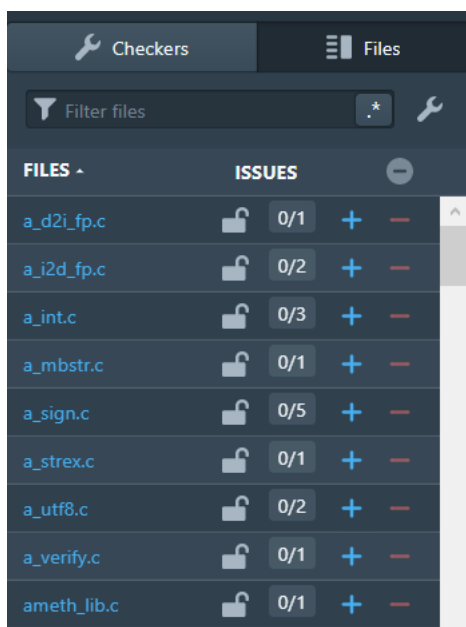


Рис. 14 — Панель Files

Действия, доступные на панели **Files**:

- клик на **FILES** или **ISSUES** — позволяет отсортировать столбцы по значениям;
- нажатие кнопок **+** и **-** позволяют добавить или удалить соответствующий файл в фильтре для просмотра предупреждений в таблице предупреждений;
- ввод в поле **Filter files** имени файла или пути к нему — позволяет задать фильтрацию по имени/пути файла, а также включить фильтрацию по регулярным выражениям, нажав на кнопку **.***, которая находится в правой части поля ввода;

3) Пользовательские фильтры влияют на левую группу панелей.

4. Центральная группа элементов:

1) Таблица предупреждений (Рис. 15, 1):

	<input type="checkbox"/>	Checker	File	Line	Description
1	<input type="checkbox"/>	INVARIANT_RESULT	scrypt.c	426	Expression 'maxmem > (18446744073709551615UL)' is...
2	<input type="checkbox"/>	INVARIANT_RESULT	a_int.c	547	Expression 'r > 9223372036854775807L' is always false...
3	<input type="checkbox"/>	INVARIANT_RESULT	a_int.c	590	Expression 'r > 9223372036854775807L' is always false...
4	<input type="checkbox"/>	DEREF_AFTER_NULL	ssl_cert.c	872	After having been compared to NULL value at ssl_cert.c...
5	<input type="checkbox"/>	DEREF_AFTER_NULL	err_blocks.c	95	After having been compared to NULL value at err_bloc...
6	<input type="checkbox"/>	DEREF_AFTER_NULL	pmeth_gn.c	314	After having been compared to NULL value at pmeth_g...
7	<input type="checkbox"/>	DEREF_AFTER_NULL	digest.c	594	After having been compared to NULL value at digest.c...
8	<input type="checkbox"/>	DEREF_AFTER_NULL	digest.c	640	After having been compared to NULL value at digest.c...
9	<input type="checkbox"/>	DEREF_AFTER_NULL	cipher_aes_ocb.	161	After having been compared to NULL value at cipher_a...
10	<input type="checkbox"/>	DEREF_AFTER_NULL	m_sigver.c	315	After having been compared to NULL value at m_sigver...
11	<input type="checkbox"/>	DEREF_AFTER_NULL	m_sigver.c	343	After having been compared to NULL value at m_sigver...

Рис. 15 — Панель и таблица предупреждений

Эта таблица является основным элементом для работы с предупреждениями.

Действия, доступные в таблице:

- выбор предупреждений для групповой разметки;
- выбор активного предупреждения — двойной клик на строке в любом столбце или одинарный клик на имени файла в столбце **File**;

Также можно использовать клавиши управления курсором и Enter для выбора активного предупреждения.

2) Панель предупреждений (Рис. 15, 2)

Информация и действия, доступные на панели:

- число отображаемых в таблице предупреждений;
- фильтрация предупреждений (Рис. 16):
 - комплексная фильтрация предупреждений — **Filters > Custom**;
 - показ только предупреждений с разметкой — **Show markers with non-default status, severity or action**;
 - Отменить пользовательские фильтры — **Filters > Unset custom filter** (доступно, если применен пользовательский фильтр);
 - Отменить все фильтры — **Filters > Unset all filters** (доступно, если применен любой фильтр);
 - Применить сохраненный фильтр — **Filters > Saved** (выбрать из существующих);
- выгрузка отображаемых предупреждений в формате .csv и выгрузка отчета отображаемых предупреждений в формате .pdf (Рис. 17);
 - групповая разметка выбранных предупреждений (подробнее в разделе 5.5);
 - поиск предупреждений — ввод значения в поле **query** (поддерживается использование метасимволов * и ?).

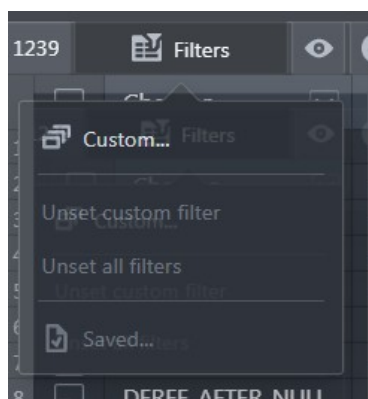


Рис. 16 — Фильтры предупреждений

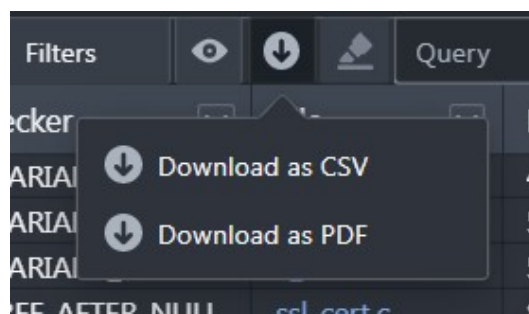


Рис. 17 — Выгрузка предупреждений

3) Панель с кодом и информацией о снимке (Рис. 18):

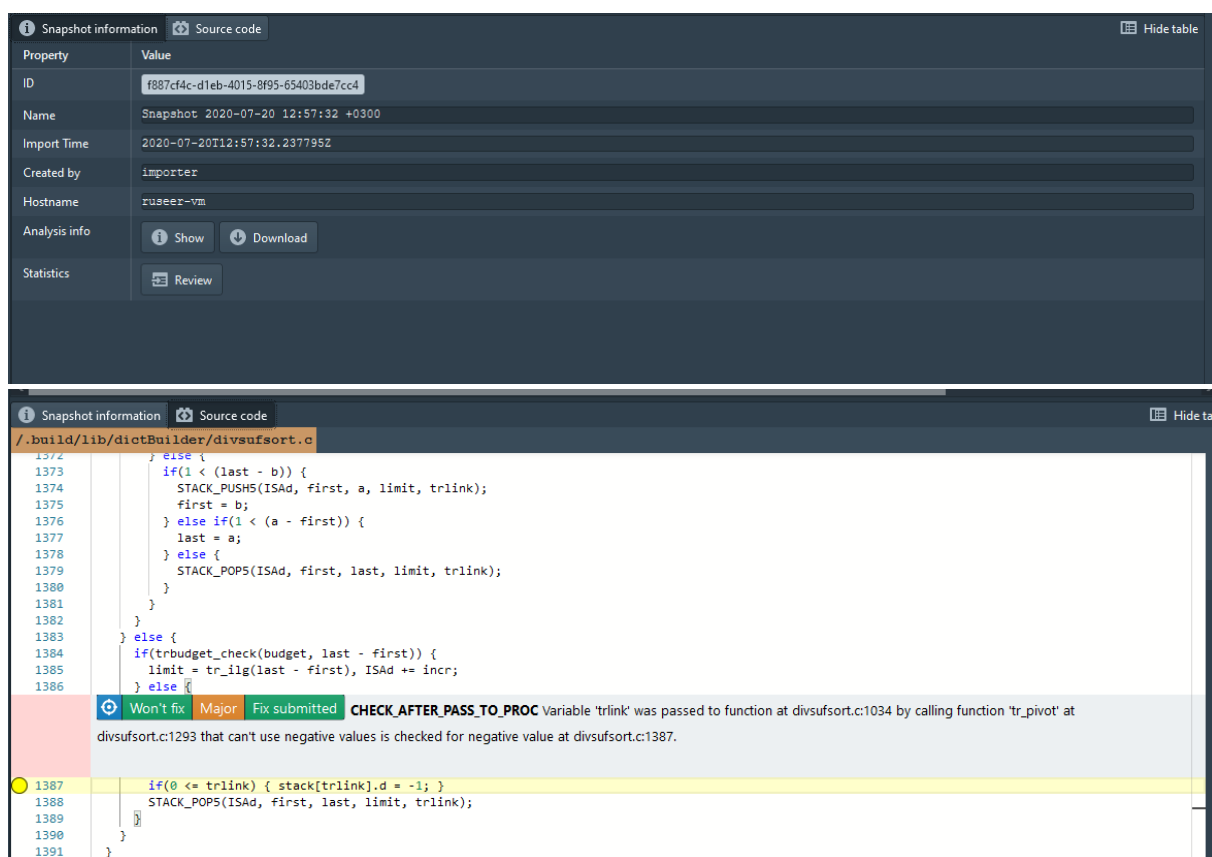


Рис. 18 — Панель с кодом и информацией о снимке

Эта панель предоставляет информацию о снимке и показывает положение предупреждения в исходном коде.

Кнопка **Hide Table** позволяет скрыть таблицу и развернуть панель с кодом на всю среднюю панель.

В части панели с кодом секция с информацией о предупреждении содержит кнопки, которые позволяют разметить открытое предупреждение.

3. Правая группа панелей (Рис. 19).

Эта группа предоставляет дополнительную информацию о предупреждении: статус блокировки предупреждения, статус разметки, кто последний размечал предупреждение и многое другое. Здесь пользователю предоставляется возможность разметить предупреждения, добавить комментарии и просмотреть трассу предупреждения.

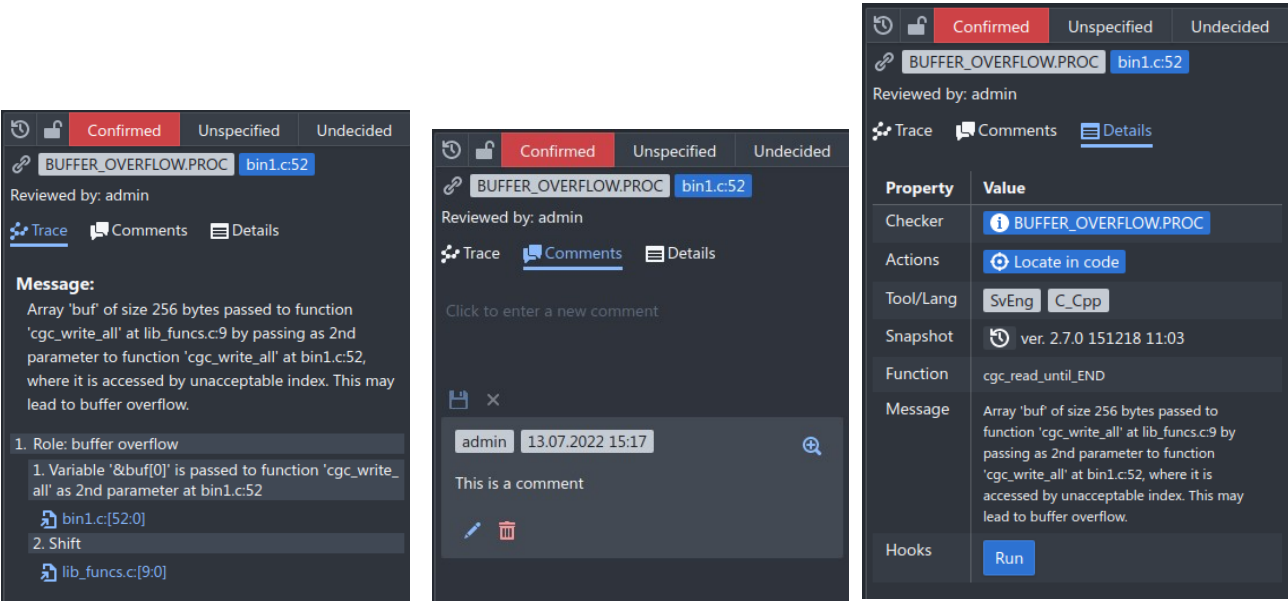


Рис. 19 — Правая группа панелей

Доступные действия:

- Отображение исходного кода в панели **Source code** — доступно по клику на ссылку с именем файла и номером строки; Это позволяет всегда вернуться к нужной точке при навигации по коду;
- Отображение информации (если доступна) о выбранном чекере (Рис. 20) — доступно по клику на имени чекера в закладке **Details**;

Checker information							
ID	Severity	Status	Reliability	Situation	Tools	Languages	Description
INVARIANT_RESULT	Major		Average	Quality	CSA, TagsJavac, Roslyn	C_CPP, JAVA, CSHARP	Expressions whose result doesn't depend on their variable operands.

Рис. 20 — Информация о чекере

- предпросмотр кода, который соответствует строке трассы предупреждения, в отдельном окне (Рис. 21) — доступно по клику на иконку

- разметка предупреждения с помощью кнопок разметки (Рис. 22);

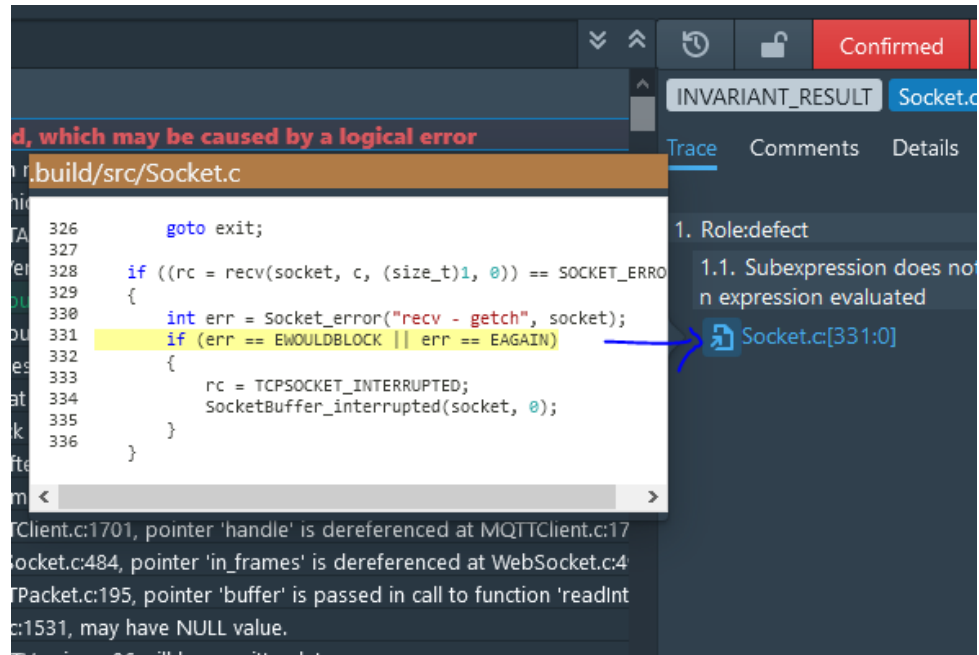


Рис. 21 — Окно предпросмотра кода трассы

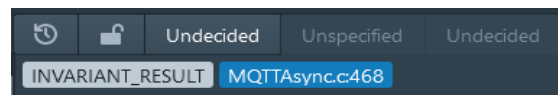



Рис. 22 — Кнопки разметки предупреждения

Разметка предупреждения заключается в установке статуса (**Confirmed**, **Won't Fix**, **False Positive**, **Unclear**) и опциональной установки **severity** и **action** (доступно только после установки **Status**).

- просмотр истории изменения разметки предупреждения (Рис. 23) — доступно по нажатию кнопки ;

Review history					
Status	Severity	Action	Created by	Date	Created From
Confirmed	Critical	Fix submitted	bob	2020-07-14T14:39:58.949301Z	Snapshot 2020-03-25 09:09:41 +0300
Confirmed	Critical	Undecided	bob	2020-07-14T14:39:48.831576Z	Snapshot 2020-03-25 09:09:41 +0300
Confirmed	Unspecified	Undecided	bob	2020-07-14T14:39:46.629581Z	Snapshot 2020-03-25 09:09:41 +0300

Рис. 23 — История изменения разметки предупреждения

5.5 Групповая разметка предупреждений

Для групповой разметки:

1. Выделите одно или несколько предупреждений (Рис. 24).
2. Нажмите кнопку **Review markers group**.

Кнопка становится активной, только если есть выбранные предупреждения.

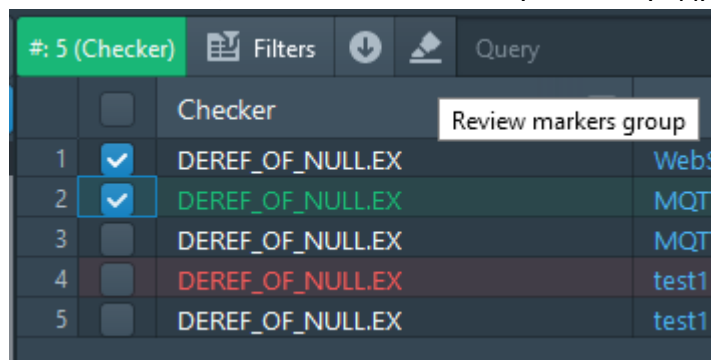


Рис. 24 — Кнопка «Review markers group»

Отобразится окно для групповой разметки (Рис. 25):

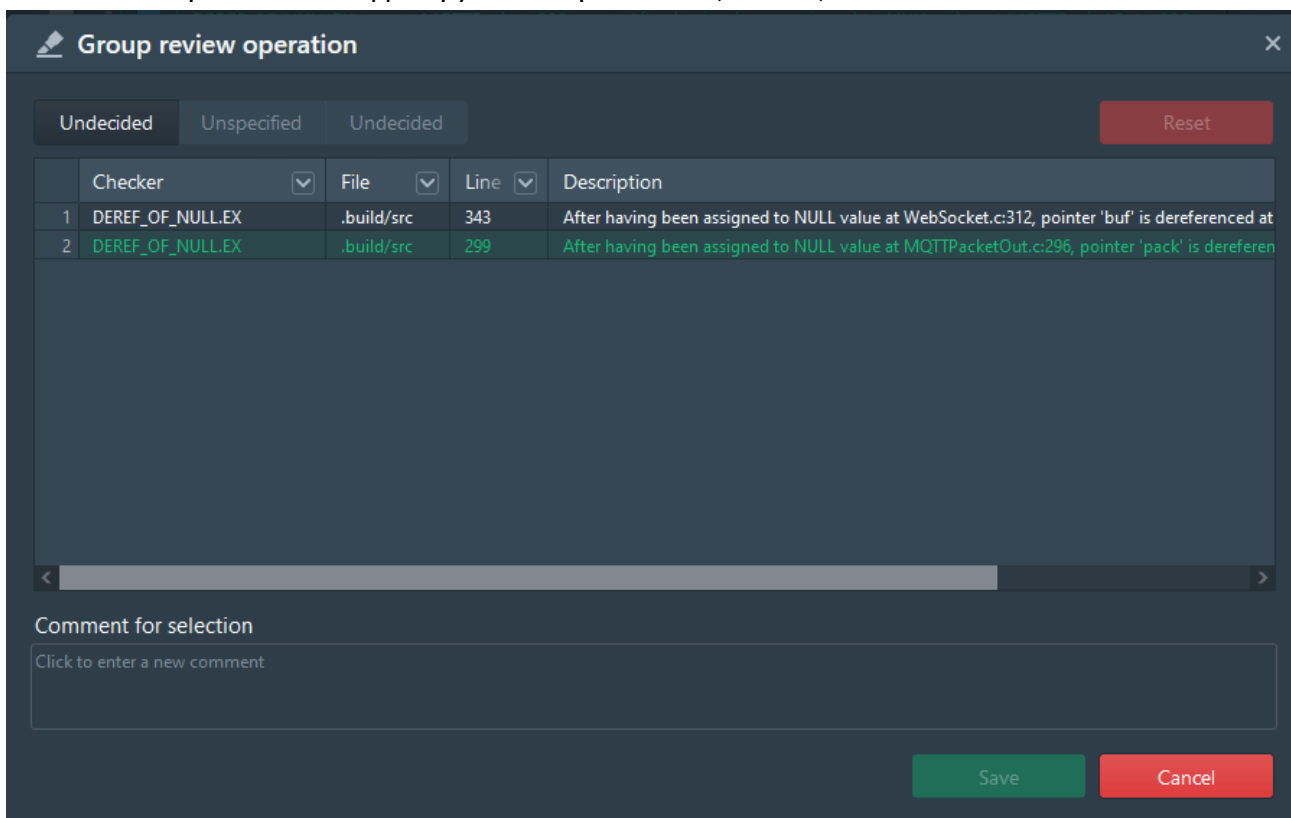


Рис. 25 — Окно для групповой разметки. Начальное состояние

3. Разметьте выбранные предупреждения.

Отображаемые в окне предупреждения окрашиваются в соответствующий цвет. Например, если пользователь поставил статус проверки **Confirmed**, то все предупреждения окрасятся в красный цвет (Рис. 26).

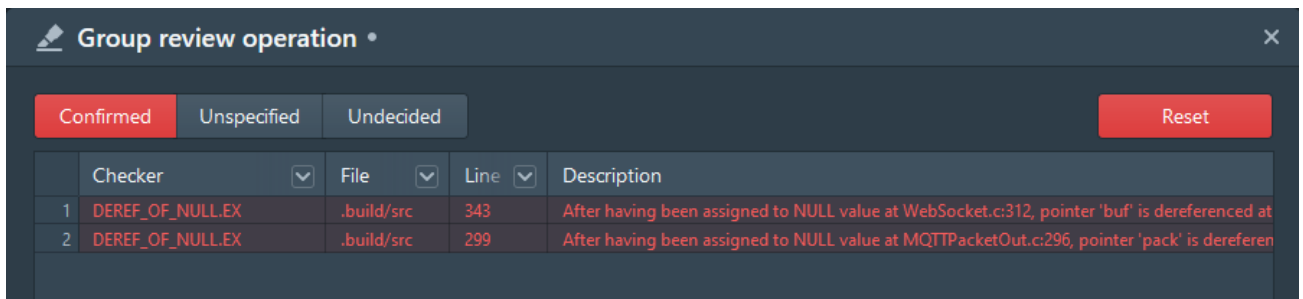


Рис. 26 — Пример размеченных предупреждений.
Установлен статус проверки Confirmed

4. Добавьте общий комментарий для выбранных предупреждений.

После того, как пользователь разметил и (или) написал комментарий, станут доступными кнопки **Reset (Сбросить)** и **Save (Сохранить)**, а справа от заголовка окна появится индикатор — кружок (Рис. 27).

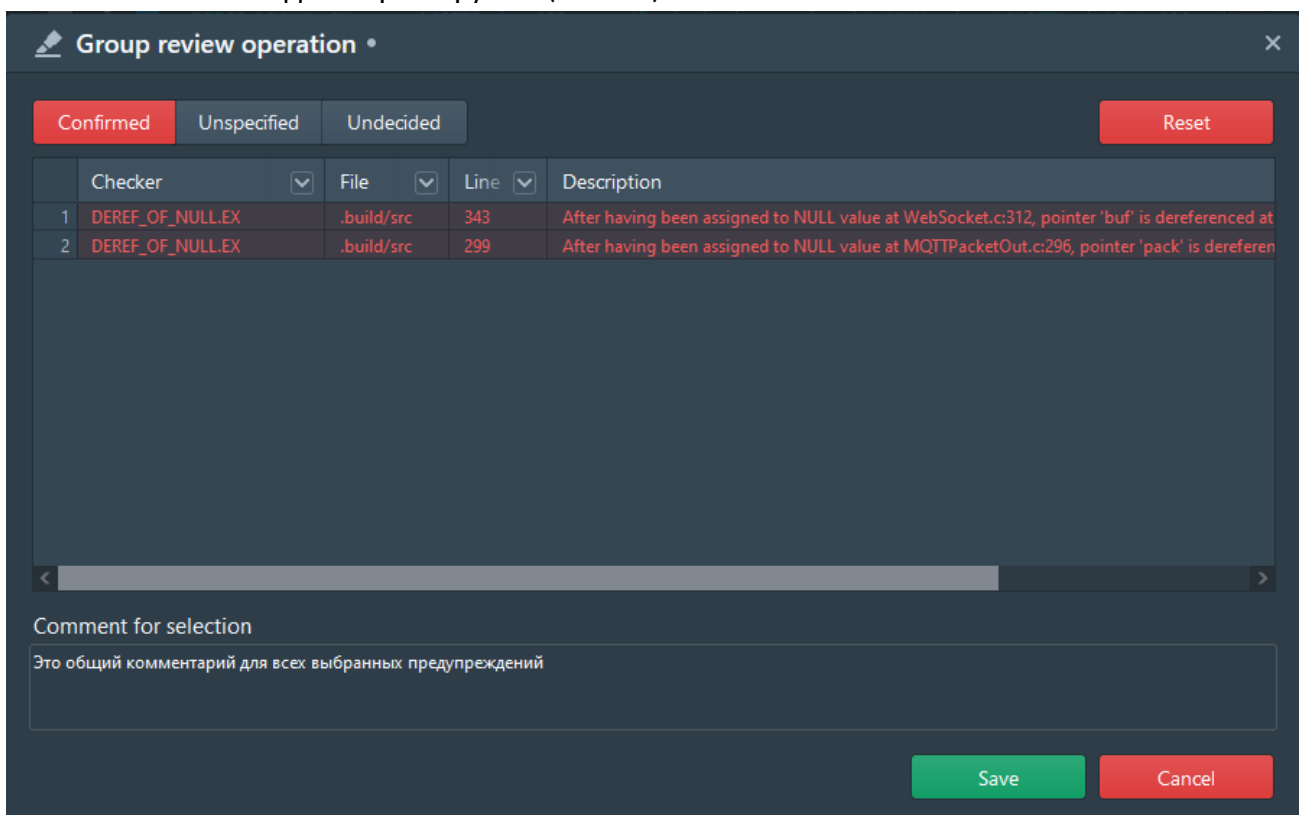
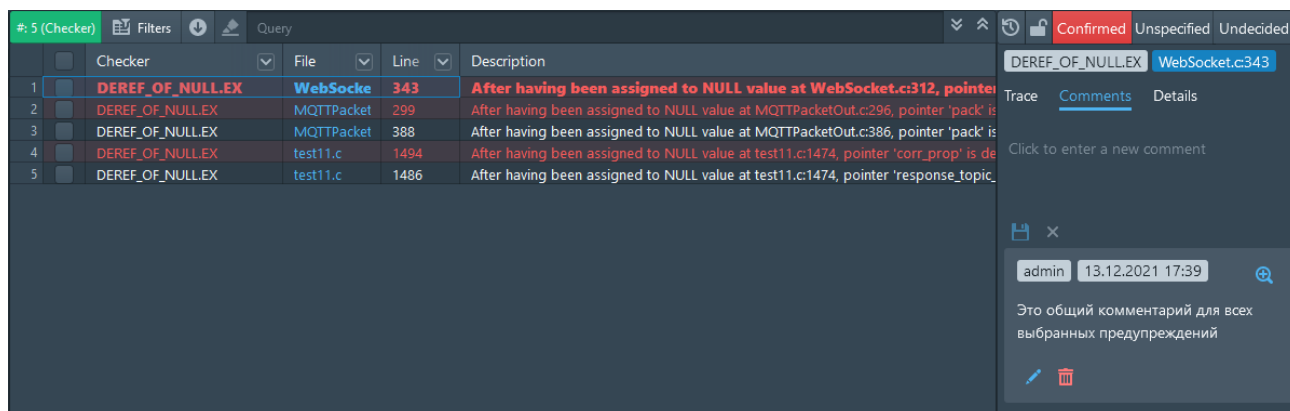


Рис. 27 — Окно для групповой разметки с кнопками

5. Если требуется отменить изменения, нажмите **X** в правом верхнем углу окна или кнопку **Cancel (Отмена)**. Диалоговое окно закроется и изменения не будут применены.

6. Если требуется сбросить все изменения, нажмите кнопку **Reset**. Данные в окне вернутся в первоначальное состояние, кнопки **Reset** и **Save** станут неактивными, а также пропадёт индикатор справа от заголовка окна (Рис. 25).

7. Нажмите кнопку **Save**, чтобы сохранить изменения разметки и сбросить выбор в таблице предупреждений (Рис. 28).



The screenshot shows a software interface with a table of warnings and a sidebar for comments. The table has columns for 'Checker', 'File', 'Line', and 'Description'. The sidebar on the right has tabs for 'Trace', 'Comments', and 'Details', and a text input area for comments.

	Checker	File	Line	Description
1	DEREF_OF_NULL.EX	WebSocke	343	After having been assigned to NULL value at WebSocket.c:312, pointer
2	DEREF_OF_NULL.EX	MQTTPacket	299	After having been assigned to NULL value at MQTTPacketOut.c:296, pointer 'pack' is
3	DEREF_OF_NULL.EX	MQTTPacket	388	After having been assigned to NULL value at MQTTPacketOut.c:386, pointer 'pack' is
4	DEREF_OF_NULL.EX	test11.c	1494	After having been assigned to NULL value at test11.c:1474, pointer 'corr_prop' is de
5	DEREF_OF_NULL.EX	test11.c	1486	After having been assigned to NULL value at test11.c:1474, pointer 'response_topic

Рис. 28 — Таблица предупреждений после сохранения групповой разметки

5.6 Использование регулярных выражений

Использование регулярных выражений для поиска и фильтрации доступно в левой панели на вкладке **Files** и в **Filters > Custom > Files**.

Особенности реализации:

- Поиск происходит только по регулярному выражению. При необходимости использования специальных символов как обычных их надо экранировать (например, точку: «\.»)
- Находятся вхождения подстроки в полном пути к файлу (также, как, к примеру, работает `grep`). При необходимости поиска по полной строке используйте символы начала/конца строки: `^` и `$`
- Поиск регистронезависимый (case insensitive)

В **Filters > Custom > Files** существует возможность применить фильтр как для отображения только предупреждений из файлов, подходящих под паттерн, так и для скрывания таких предупреждений. Для переключения между этими режимами нажмите кнопку **«+»/«-»**, которая расположена рядом с полем ввода.

Примеры регулярных выражений:

1. Найти файлы с «sha» или «md5» где-либо в пути или в имени файла
`sha|md5`

2. Отобразить только предупреждения из файлов с расширением .c

`.*\..c$`

3. Скрыть предупреждения из файлов, имя которых начинается с символа «q» и которые имеют расширения .c или .cc

1) Используйте выражение `/q[^\.]*\..c$|/q[^\.]*\..cc$`

2) Нажмите кнопку «+» рядом с полем ввода, чтобы она отобразилась как «-».

4. Показать только предупреждения из файлов, которые имеют «string» в конце имени файла и с расширением из одного символа

`/.*string\..$`

5. Скрыть предупреждения из файлов в директориях asn1 и pem

1) Используйте выражение `/asn1/|/pem/`

2) Нажмите кнопку «+» рядом с полем ввода, чтобы она отобразилась как «-».

6. Показать только предупреждения из файлов, которые имеют в имени три цифры подряд

`.*/*.*[0-9]{3}[^\.]*$`

Где конструкция `[^\.]*$` означает, что после трех цифр и до конца строки может встречаться любой символ кроме /. Это позволяет исключить директории имеющие три цифры в названии.

5.7 Блокировка разметки

Пользователь может заблокировать другому пользователю возможность размечать предупреждение или группу предупреждений. Это обеспечивает возможность разделения работы нескольких пользователей по разметке результатов без возникновения коллизий.

Для блокировки разметки используйте кнопку :

- на панели чекеров (Рис. 29);

В этом случае блокировка распространяется на все предупреждения чекера в выбранном проекте и ветке (для всех снимков из ветки).

- на панели файлов (Рис. 30);

В этом случае блокировка распространяется на все предупреждения в файле в выбранном проекте и ветке (для всех снимков из ветки).

- на панели разметки (Рис. 31);

В этом случае блокировка распространяется на все эквивалентные предупреждения в выбранном проекте и ветке (для всех снимков из ветки).

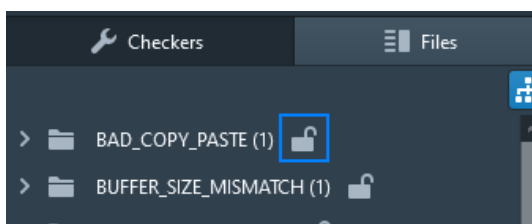


Рис. 29 — Блокировка на панели чекеров

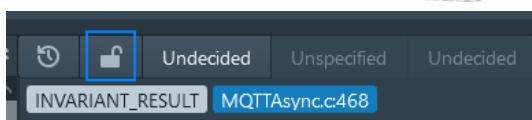


Рис. 31 — Блокировка на панели разметки

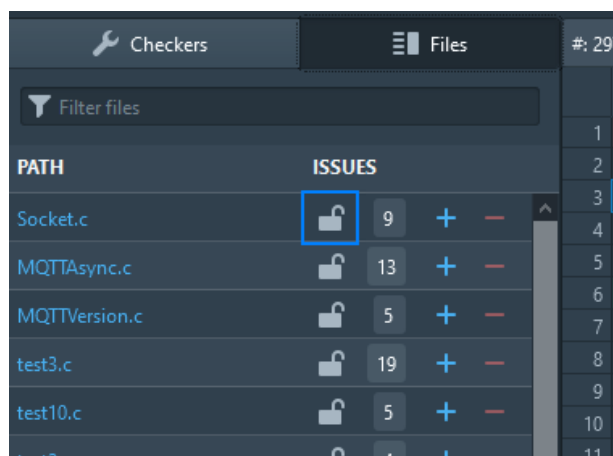


Рис. 30 — Блокировка на панели файлов

Для просмотра всех блокировок используйте панель **Settings > Review Locks** (Рис. 32):

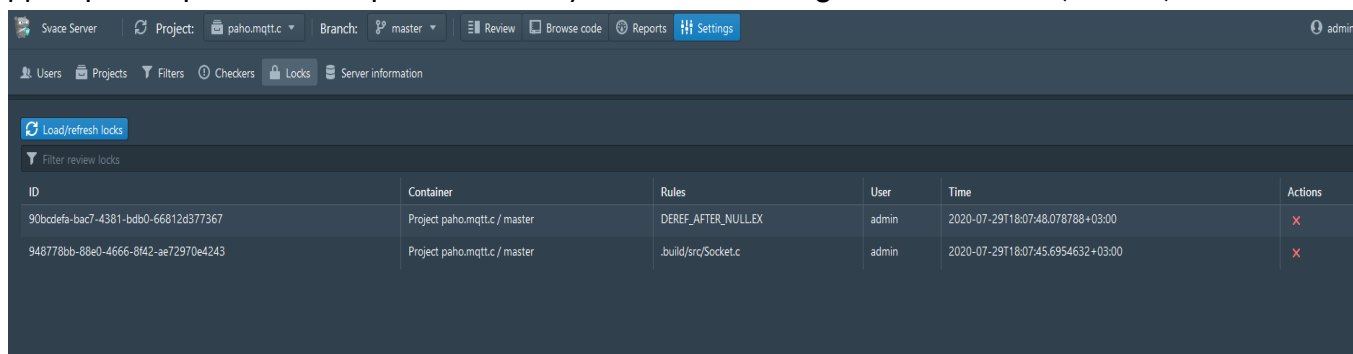


Рис. 32 — Панель Review Locks

Также на этой панели можно удалить свои блокировки.

Пользователь **admin** может удалять любые блокировки.

5.7.1 Экспорт исходных кодов с разметкой (опционально) из снимка

Есть возможность экспортировать с сервера исходный код с разметкой (опционально) из снимка двумя способами:

- 1) Через UI в панели **Snapshot information** в строке **ID** нажать на кнопку **Sources export** (Рис. 33)







Snapshot information		Source code	Hide table
Property	Value		
ID	 		
Name	ver. 2 Sources export :03		
Import time	15.12.2018 11:00		
Created by	importer		
Hostname	seroshki		
Analysis info	 Show  Download		
Statistics	 View		

Рис. 33 — Экспорт кода с разметкой

Откроется диалоговое окно (Рис. 34), где нужно будет выбрать шаблон для экспорта (по умолчанию **None** (т.е. разметка экспортироваться не будет) и опционально заполнить поля для удаления префиксов и исключения путей.


 Sources export

Review export template

None

Strip prefixes (use comma to separate patterns) ?

Exclude paths (use comma to separate patterns) ?

 Reset

Cancel

Ok

Рис. 34 — Диалог экспорта разметки

Пример удаления префиксов путей: `"/.build/"` чтобы убрать папку `.build` при экспорте. При экспорте можно исключить файлы, соответствующие регулярным выражениям в поле **Exclude paths**. В итоге создается архив со всеми исходниками и, если был выбран шаблон разметки, то в исходном коде будет вставлена актуальная разметка, соответствующая выбранному шаблону.

- 2) Через консольную команду **svacer markup export**

Синтаксис команды:

```
svacer markup --user <user> --password <password> --project <project id or name> --branch <branch id or name> --snapshot <snapshot id or name> export [--stripPrefixes <prefix1, prefix2, ... , prefixN> --excludePaths <path1, path2, ... , pathN> --template <name>]
```

Где:

- user, password — пользователь и пароль учетной записи на сервере истории
- project — проект на сервере истории, из которого будет произведен экспорт
- branch — ветка в проекте выбранном выше (по умолчанию master), из которой будет произведен экспорт
- snapshot — снимок в выбранных проекте и ветке (по умолчанию последний загруженный), из которого будет произведен экспорт
- stripPrefixes, excludePaths — аналогичны полям «Удалить префиксы» и «Исключить пути» из формы экспорта разметки в UI, описанной выше
- template — определяет имя шаблона для экспорта разметки. Если имя не указано, разметка не экспортируется

Итогом будет либо создание директорий и файлов, соответствующих выбранному проекту на сервере истории и наполнение их исходным кодом, либо, если директории и файлы уже существуют, они будут перезаписаны. Наличие в файлах разметки в виде комментариев зависит от того, был ли выбран шаблон разметки.

5.7.2 Импорт разметки из исходного кода на сервер истории

Импорт разметки из исходного кода на сервер истории возможен с помощью команды **svacer markup import**.

Синтаксис:

```
svacer markup --user <user> --password <password> --project <project id or name> --branch <branch id or name> --snapshot <snapshot id or name> import --template <name> [--excludePaths <path1, path2, ... , pathN> --from-build-object]
```

Где:

- user, password — пользователь и пароль учетной записи на сервере истории
- project — проект на сервере истории, в который будет произведен импорт разметки
- branch — ветка в проекте выбранном выше (по умолчанию master), в которую будет произведен импорт разметки
- snapshot — снимок в выбранных проекте и ветке (по умолчанию последний загруженный), в который будет произведен импорт разметки
- excludePaths — исключает файлы, соответствующие регулярным выражениям

- `template` — определяет имя шаблона для импорта разметки (**важно:** флаг обязательный, в отличие от экспорта)
- `from-build-object` — при включении этой опции, разметка будет выгружаться из объекта сборки того снимка, который был указан в опциях выше.

После выполнения команды все комментарии оставленные в коде в соответствии с выбранным шаблоном разметки будут разобраны и добавлены на сервер истории в соответствующий проект/ветку/снимок.

Для подавления найденных предупреждений следует использовать следующий формат разметки в исходном коде

```
//svacer_review: -<warn class>[|-<warn class>]+
```

Пример:

```
if (!strcmp("--filelimit",argv[i],11)) {
    j = 11; //svacer_review: -UNUSED_VALUE
    if (*(argv[i]+11) == '=') {
```

Комментарий должен быть расположен в конце строки, где ожидается предупреждение от анализатора Svace.

Для импортирования разметки при использовании команд **import --upload** и **upload** следует указывать флаг **--template <template>**. В таком случае разметка будет импортирована из объекта сборки сразу после загрузки его на сервер.

5.7.3 Изменение инвариантов для распространения разметки между снимками, с несовпадающими путями

Если на сервер был загружен снимок, в котором расположение некоторых (или всех) файлов поменялось, а при импорте не был указан флаг **--pathPrefix**, то разметка с прошлого снимка не распространится на новый снимок. Для решения данной проблемы следует использовать команду **svacer markup update**.

Синтаксис:

```
svacer markup update --user <user> --password <password> --project <project id or name> --branch <branch id or name> --pathPrefix <prefix:replacement; prefix:replacement> [<snapshot id or name>1, <snapshot id or name>2, ..., <snapshot id or name>N]
```

Где:

- `user, password` — пользователь и пароль учетной записи на сервере истории
- `project` — проект на сервере истории, в котором будет обновлена разметка

- `branch` — ветка в проекте выбранном выше (по умолчанию `master`), в которой будет обновлена разметка
- `pathPrefix` — правило отображения для префиксов путей в формате: `<file>` или `line prefix:replacement;prefix:replacement`

В конце команды опционально могут быть указаны имена или идентификаторы снимков, в которых будет обновлена разметка в соответствии с указанными префиксами путей. Если они указаны не будут, изменения будут применены ко всей ветке. После запуска команды на сервер будет отправлен асинхронный запрос на обновление разметки, за дальнейшим прогрессом исполнения можно следить в логе сервера (через UI он доступен пользователям с ролью `admin` в **Settings > Server information**).

5.8 Подавление предупреждений

Для подавление выбранных пользователем предупреждений предоставляется следующий интерфейс командной строки:

Список подавленных предупреждений (печатает JSON)

```
svacer markup [auth/selection] suppress list
```

```
пример: svacer markup --user admin --password admin --project tree-command
suppress list
```

Подавить предупреждение

```
svacer markup [auth/selection] suppress add id1 id2 id3
```

```
пример: svacer markup --user admin --password admin --project tree-command
suppress add 29d298cd-752d-4d19-b535-2089a72a96af
```

Убрать подавление предупреждения

```
svacer markup [auth/selection] suppress remove id1 id2 id3
```

```
пример: svacer markup --user admin --password admin --project tree-command
suppress remove 29d298cd-752d-4d19-b535-2089a72a96af
```

Убрать все подавления

```
svacer markup [auth/selection] suppress remove
```

```
пример: svacer markup --user admin --password admin --project tree-command
suppress remove
```

Список предупреждений с ID доступен посредством public REST API. Функциональность предназначена для интеграции со сторонними инструментами и встраивание в CI/CD pipeline.

На текущий момент информация о подавленных предупреждения недоступна в веб-интерфейсе.

При экспортировании снимка где есть подавленные предупреждения, сами предупреждения будут так же экспортированы как подавленные.

5.9 Публичные REST запросы

Для работы с сервером через различные приложения реализованы публичные запросы, которые не будут меняться при обновлении самого сервера. Если будут происходить значимые изменения, то запрос будет помечаться как deprecated и создаваться новый такой же запрос с пометкой **/api/some/request/v2**.

Для каждого публичного запроса нужен регистрационный токен. Чтобы его получить нужно пройти аутентификацию.

Аутентификация — это POST запрос с basic auth на **/api/login**, который вернёт JWT-токен в body. Полученный токен добавляется в заголовок (Header) всех запросов (Authorization: Bearer <token>).

Основные сущности сервера это:

- проекты (project) — идентифицируются по UUID
- ветка (branch) — идентифицируются по UUID
- снимки (snapshot) — идентифицируются по UUID
- маркеры (markers) — идентифицируются по UUID
- пользователи (users) — идентифицируются по UUID

Публичные запросы:

- **/api/public/projects (GET)** — возвращает список всех проектов с ветками;
- **/api/public/projects/{project_id}/branch/{branch_id}/snapshots (GET)** — возвращает список всех снимков конкретной ветки.
Необходимые ID можно получить из предыдущего запроса.
- **/api/public/projects/{project_id}/branch/{branch_id}/snapshots/{snapshot_id}/fullmarkers (GET)** — возвращает полную информацию обо всех маркерах конкретного снимка.
У запроса есть опция **?traces=true** (по умолчанию false). Она добавит к каждому маркеру его трассу.
Также доступна опция **?checker_info=true** (по умолчанию false). Она добавит к каждому маркеру severity и reliability детектора, к которому относится этот маркер.
- **/api/public/diff?snapshot_v1=<id1>&snapshot_v2=<id2>&level=<number> (GET)** — выполняет сравнение снимков с указанными id.
Если snapshot_v2 отсутствует или пустой, то сравнение будет проведено с предыдущим снимком. Количество информации определяется переменной **level**:

- 0 — только общая информация со статистикой;
- 1 — статистика и id маркеров;
- >1 (default) — полная информация.
- **/api/public/users/current (PUT)** — обновляет текущего пользователя. Нужно передать объект с такими же полями, как в запросе на добавление пользователя.
- **/api/public/snapshots/{snapshot_id}/markers/suppress (POST)** — подавление предупреждений. Тело:

<pre>{ ids: ["id1", "id2", ...] rule: { /* any JSON object */ } }</pre>	<p>Поле <code>ids</code> задает список ID маркеров для подавления.</p> <p>Поле <code>rule</code> задает специфичное для пользователя правило, согласно которому данные маркеры подавляются.</p>
---	---

- **/api/public/snapshots/{snapshot_id}/markers/unsuppress (POST)** — отмена подавления предупреждений. Тело:

<pre>{ ids: ["id1", "id2",...] }</pre>	<p>Поле <code>ids</code> задает список ID маркеров для отмены подавления.</p>
---	---

- **/api/public/snapshots/{snapshot_id}/markers/suppressed (GET)** — отмена подавления предупреждений. Возвращает список подавленных маркеров в виде JSON массива.
- **/api/public/snapshots/{snapshot_id}/custom_fields/{field_name} (GET)** — получение пользовательского атрибута у снимка. Результатом является JSON строка или массив строк
- **/api/public/snapshots/{snapshot_id}/custom_fields (GET)** — получение всех пользовательских атрибутов у снимка в виде одного JSON объекта

Публичные запросы, доступные только пользователям с ролью `admin`:

- **/api/public/users/admin/list (GET)** — возвращает список пользователей. У запроса есть опция `"roles=true"` (по умолчанию `false`). Она добавит к каждому пользователю его роли.
- **/api/public/users/admin/add (POST)** — добавляет нового пользователя. Нужно передать объект с полями `login` (обязательное), `password` (обязательное), `firstname`, `lastname`, `middlename`, `email`, `passChangeRequest`.
- **/api/public/users/admin/{user_id} (PUT)** — обновляет пользователя с переданным ID. Нужно передать объект с такими же полями, как в запросе на добавление пользователя.
- **/api/public/users/admin/{user_id} (DELETE)** — удаляет пользователя с переданным ID.

- **/api/public/users/admin/{user_id}/status (PUT)** — устанавливает пользователю с переданным ID статус переданный в объекте вида {status int}, где 0 — архивный, а 1 — активный.
- **/api/public/roles/admin/list (GET)** — возвращает список ролей.
- **/api/public/roles/admin/add (POST)** — добавляет новую роль. Нужно передать объект с полями name (обязательное), permissions (обязательное), где permissions — это массив объектов с полями target (обязательное) и action (обязательное).
- **/api/public/roles/admin/{role_id} (PUT)** — обновляет роль с переданным ID. Нужно передать объект с такими же полями, как в запросе на добавление роли.
- **/api/public/roles/admin/{role_id} (DELETE)** — удаляет роль с переданным ID.
- **/api/public/users/admin/{user_id}/roles (PUT)** — добавляет или убирает роли пользователю с переданным ID. Нужно передать объект с полями user_id (обязательное), roles (обязательное), где roles — это массив объектов с такими же полями, как в запросе на добавление роли. У запроса есть обязательный параметр "action=add/delete". Опция "add" добавит роль пользователю, а "delete" — уберет.
- **/api/public/users/current (PUT)** — обновление профиля текущего пользователя (того, с чьим токеном авторизации мы обращаемся к API). В body передается JSON объект с полями, аналогичными запросу на добавление нового пользователя
- **/api/public/users/admin/{user_id}/status (PUT)** — установка статуса пользователя. Body должно содержать JSON объект вида {status:<int>} где значение статуса 0 или 1. Значение 0 значит «архивный» пользователь (неактивен, не может зайти в сервер), и 1 — активный пользователь.
- **/api/public/exportPDF (POST)** — создание PDF отчета по проекту. См. 3.3 Создание PDF отчета для проекта .
- **/api/public/generateURLs (POST)** — создание URL для маркера. Принимает JSON объект с полями
 - markers:["id1", "id2", ..] — список ID маркеров для генерации URL
 - location: "<url>" — базовый URL на каком виден Svacer с точки зрения клиента
 - type: "short|full" — опциональный атрибут для генерации короткого или полного URL. По умолчанию создаются полные URL

результат работы команды — JSON объект с полями

 - urls :["url1", "url2", ...] каждый URL соответствует ID маркеру в той же позиции во входных данных
 - type: "short|full" — тип URL
- **/api/health (GET)** — проверка доступности сервера. Возвращает Status 200 OK если сервер работает
- **/api/public/login (POST)** — аутентификация пользователя
Тело запроса имеет формат JSON и выглядит следующим образом:

<pre>{ "auth_type": "ldap", "server": "open_ldap", "login": "user", "password": "user", "role": "admin" }</pre>	<ul style="list-style-type: none"> • auth_type — тип аутентификации ("svacer" или "ldap") • server — имя ldap сервера, как он указан в конфигурационном файле ldap (поле name) при старте сервера. Поле можно опустить в случае использования "svacer" аутентификации • role — роль, с которой будет работать пользователь • login — имя пользователя • password — пароль пользователя
---	---

- **/api/public/help (GET, без авторизации)** — дополнительная информация по REST API

Не публичное API, которое в следующих релизах станет публичным (добавится префикс public):

- **/api/projects/admin/{container_id}/update (PUT)** — Метод переименовывает проект или ветку. Body должно содержать JSON объект с полем "name" и "id", где id должно соответствовать container_id, а container_id соответствовать ID ветки или проекта.

5.10 Администрирование сервера Svace

Функции, описанные в этом разделе, доступны только пользователю с ролью **admin**.

5.10.1 Учетные записи и роли пользователей

В окне **Settings > Users** отображаются все существующие пользователи. В этом окне:

1. Чтобы добавить учётную запись пользователя:
 - 1) Нажмите кнопку **Add user**
 - 2) Заполните поля появившейся формы (Рис. 35)
 - 3) Установите переключатель, если нужно сменить пароль при первой авторизации

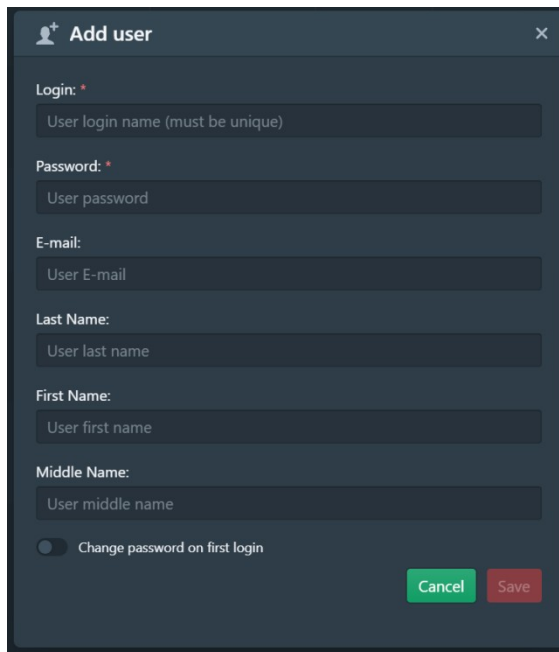



Рис. 35 — Окно создания учётной записи пользователя

2. Чтобы для существующей учетной записи пользователя поменять пароль:
 - 1) Нажмите кнопку  в столбце **Action**
 - 2) В появившейся форме (Рис. 36) введите пароль в оба поля и установите переключатель, если нужно, чтобы пользователь сменил пароль после авторизации

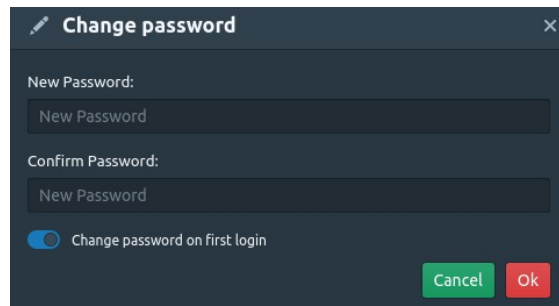




Рис. 36 — Окно смены пароля пользователя

3. Чтобы архивировать пользователя:

- 1) Нажмите кнопку  в столбце **Action**
- 2) При архивации пользователя с него будут сняты все роли и удалены его блокировки
- 3) Для восстановления пользователя нажмите кнопку  в столбце **Action**
- 4) При восстановлении пользователя роли ему присвоены не будут

4. Чтобы добавить роль:

- 1) Нажмите кнопку **Add role**
- 2) Заполните поля появившейся формы (Рис. 37)
- 3) Укажите проект, к которому относится роль (ALL, если ко всем проектам) и права, которые будут доступны пользователю

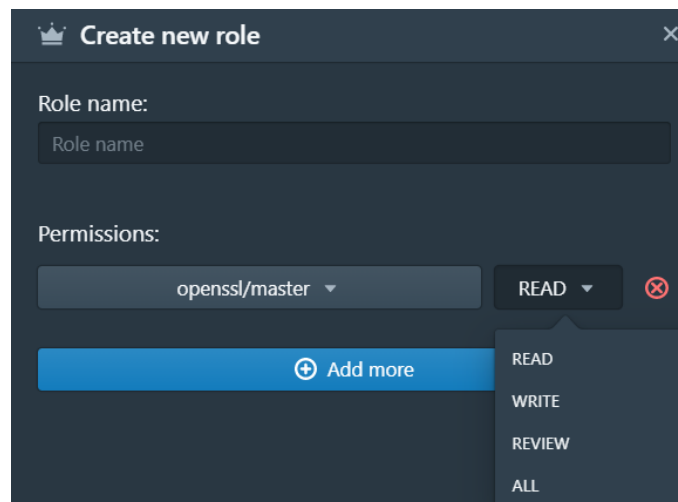


Рис. 37 — Настройка роли и прав доступа

5.10.2 Семантика ролей пользователя

Роль пользователя определяет какие действия допускаются пользователю со следующими объектами сервера: проектами, ветками, маркерами, снимками

Действие	Область	Эффект
READ	Проект	Получение списка веток, получение списка предупреждений из любой ветки проекта, экспорт данных из веток проекта
READ	Ветка	Получение списка предупреждений из ветки проекта, экспорт данных из ветки
WRITE	Ветка/Проект	Импортирование данных на конкретную ветку или на любую ветку проекта
REVIEW	Ветка/Проект	Возможность менять разметку и добавлять/менять комментарии на предупреждениях

Существует специальная область ALL, которая означает все проекты и ветки.

Существуют следующие специальные роли с особым значением, какие не могут быть удалены из системы:

Роль	Эффект
importer	Возможность загрузки данных анализа посредством команды svacer upload
review_master	Разметка, произведенная пользователем с данной ролью не может быть переопределена пользователем без данной роли
admin	Дает полномочия администратора сервера
all_projects	Доступ на все операции и ко всем проектам, но без административных привилегий

5.10.3 Ведение списка проектов

В окне **Settings > Projects** отображаются все проекты с ветками и снимками (Рис. 38). В этом окне доступны:

1. Добавление и удаление проекта (при импорте проект создается автоматически).
2. Клонирование ветки (если при импорте указать флаг **--branch <branch name>** ветка создается автоматически).

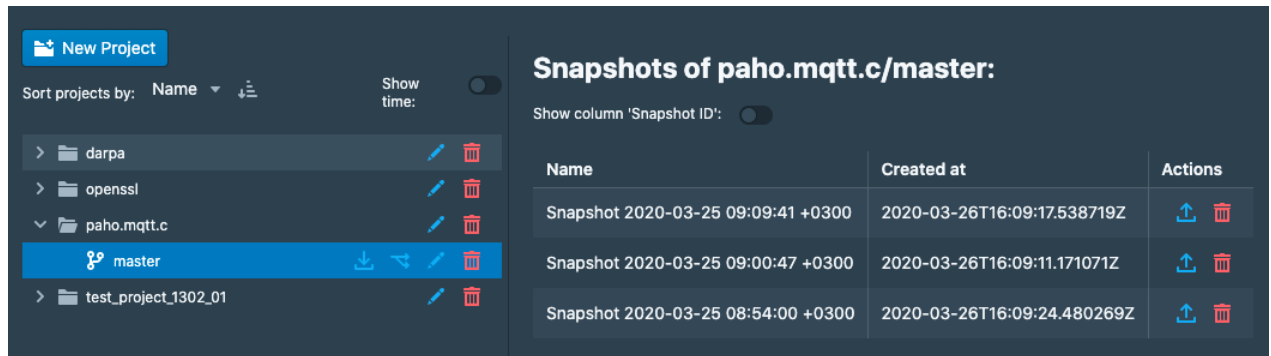


Рис. 38 — Список проектов

3. Экспорт снимков с сервера в файл для дальнейшей загрузки его на другой сервер или в другой проект на этом же сервере. Детали описываются в разделе 4.4.

4. Импорт снимка из файла. Для этого нажмите кнопку **Import snapshot** (Рис. 39) и в появившемся диалоге укажите файл для загрузки и имя нового снимка.

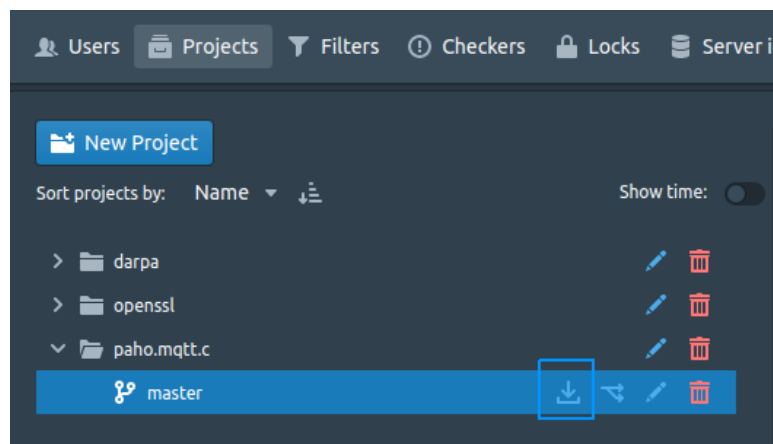


Рис. 39 — Импорт снимка

5.10.4 Настройка фильтров веток и проектов

В окне **Settings > Filters** (Рис. 40) можно создать общие фильтры на ветку или весь проект. Для смены scope с проекта на ветку или обратно следует нажать на кнопку project/branch в столбце Scope.

После применения фильтров в окне **Review** отображаются только маркеры, удовлетворяющие условиям.

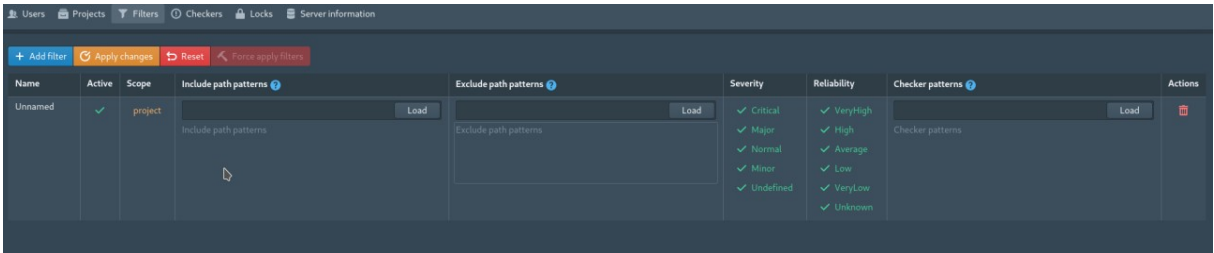


Рис. 40 — Окно фильтров веток и проектов

5.10.5 Просмотр информации о сервере

В окне **Settings > Server information** (Рис. 41) отображаются:

- параметры, с которыми запущен сервер;
- логи сервера;
- журнал базы данных;

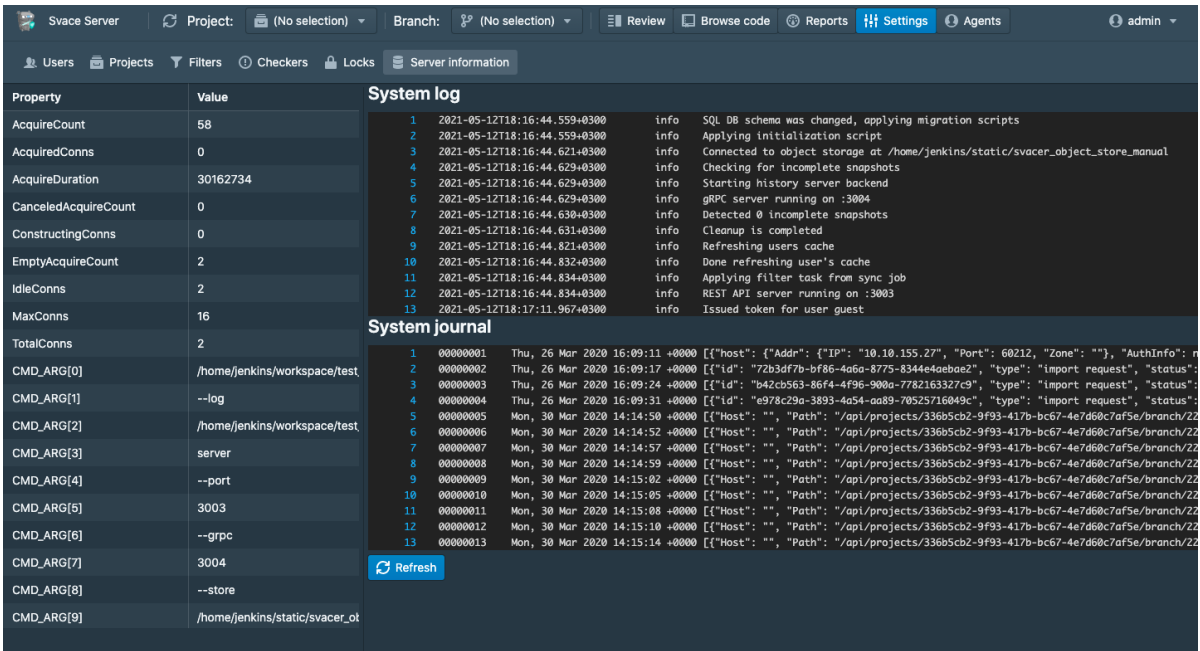


Рис. 41 — Информация о сервере

5.10.6 Копирование разметки

Для копирования разметки между контейнерами необходимо перейти в окно **Settings > Projects** и далее выделить ветку, из которой необходимо скопировать разметку. На Рис. 42 такой веткой будет `zstd_130` проекта `zstd`.

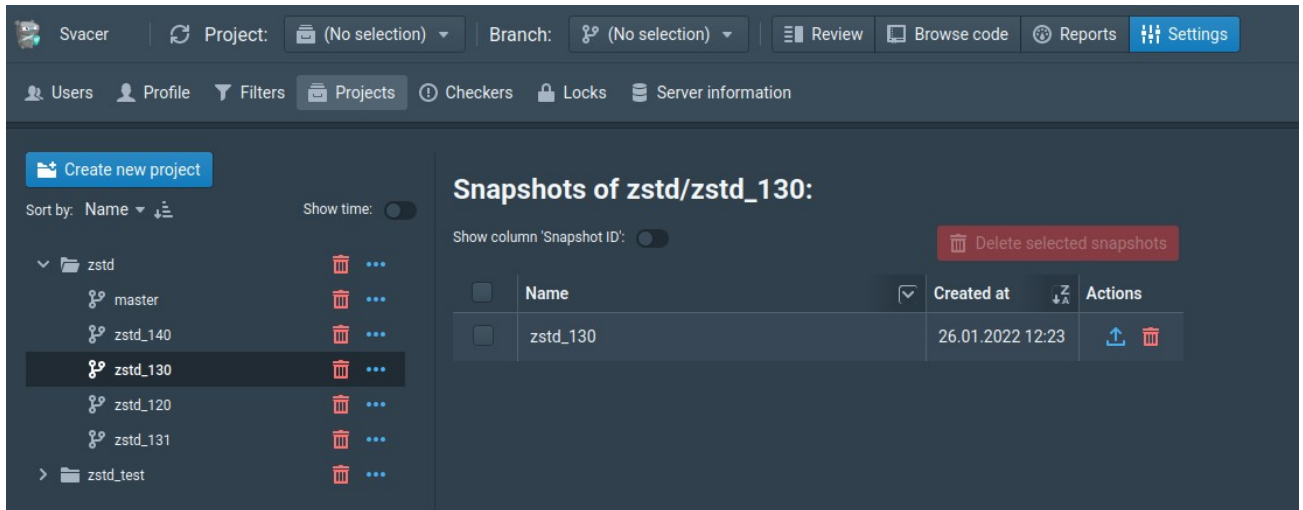


Рис. 42 — Выбор ветки для копирования разметки

Далее необходимо нажать на кнопку «...» и в появившемся меню выбрать **Copy markup**. В результате появляется окно выбора контейнера, в который будет выполнено копирование.

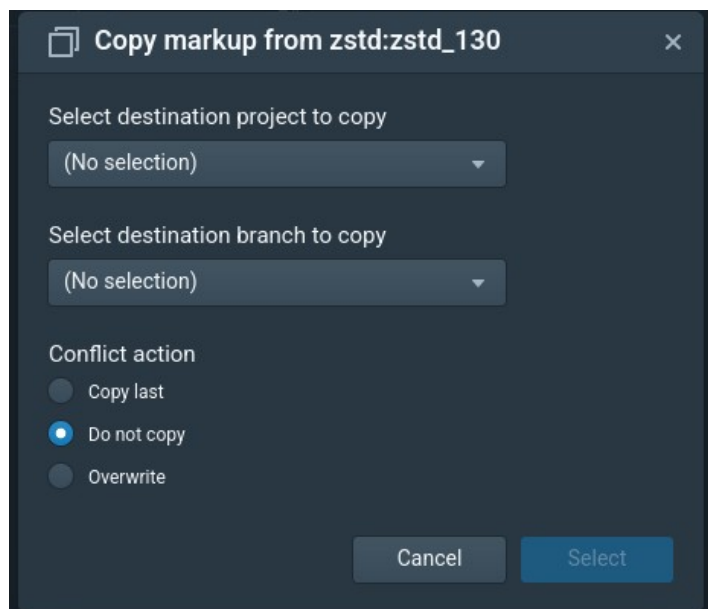


Рис. 43 — Копирование разметки

Разметка в контейнере состоит из множества размеченных маркеров. В целевой ветке уже может находиться разметка. Все множество маркеров в исходной ветке можно разбить на два множества: «Уникальные» и «Общие». «Уникальные» — маркеры, которые есть только в исходной ветке. «Общие» — маркеры, которые есть как в ветке источнике, так и в ветке-приемнике. При выполнении операции копирования разметка на уникальных маркерах копируется всегда в ветку-приемник. Для «Общих» маркеров возможны несколько вариантов продолжения копирования. По умолчанию выбран пункт «Не копировать». В данном случае разметка на «Общих» маркерах в ветке-приемнике не изменяется. В случае выбора пункта «Перезаписать», разметка на «Общих» маркерах в целевом контейнере будет заменена на разметку из ветки-источника. При выборе пункта «Заменить, если новее» разметка из ветки-источника копируется в том случае, если ее метка времени более поздняя, чем та, которая указана в ветке-приемнике на данном маркере.

Приведем пример. Пусть имеется контейнер А с разметкой $\{(M1,D1), (M2,D2), (M3,D3)\}$ и контейнер В с разметкой $\{(M4,D4), (M2,D5), (M3,D6)\}$, где М — это маркер, а D — разметка (включая метку времени). Операцию сравнения на разметке обозначим как $>$. Если метка времени в D1 больше, чем в D2, $D1 > D2$. Пусть $D2 > D5$, а $D3 < D6$. Тогда операция копирования всегда скопирует разметку D1 (но не сам маркер), так как маркер M1 новый для контейнера В. Маркеры M2, M3 — общие для двух контейнеров, поэтому какая разметка будет в контейнере-приемнике будет зависеть от выбранного варианта разрешения конфликта. Если будет выбран пункт «Не копировать», то в контейнере В на маркерах M2, M3 будет разметка D5, D6 соответственно. Если выбран пункт «перезаписать», то на M2, M3 будет разметка D2, D3. Если выбран пункт «Заменить, если новее», то на M2, M3 разметка будет: D2, D6, так как $D2 > D5$ и $D6 > D3$.

При выполнении копирования наличие маркера в ветке приемнике не проверяется.

При успешном копировании выводится всплывающее сообщение вида:

Operation done. Total count: X, Overwritten: Y.

Данное сообщение означает, что всего было скопировано X инвариантов вместе с разметкой и среди скопированных инвариантов Y штук было скопировано с заменой.

5.10.7 Создание и изменение шаблонов разметки

В окне **Settings > Markup templates** (Рис. 44) можно посмотреть все шаблоны, а также создать новый или клонировать существующий.

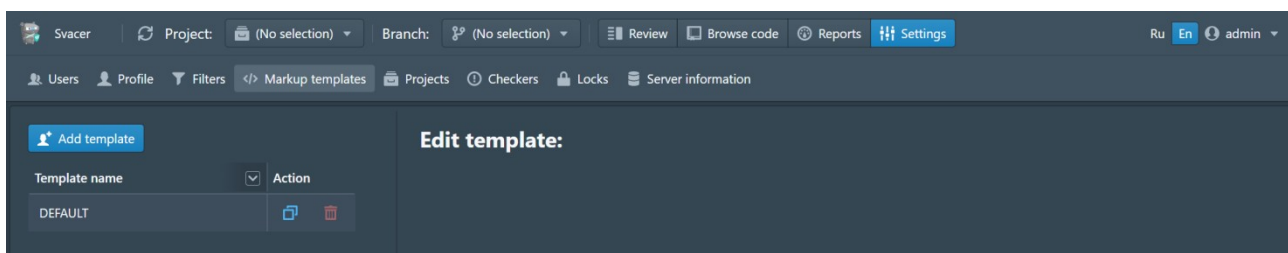


Рис. 44 — Шаблоны разметки

Шаблоны разметки нужны для импорта/экспорта разметки. С их помощью комментарии в исходном коде будут преобразовываться в разметку на сервере истории и обратно. Шаблон с именем **DEFAULT** есть всегда, его нельзя отредактировать или удалить. На Рис. 45 представлены все поля шаблона, доступные для редактирования.

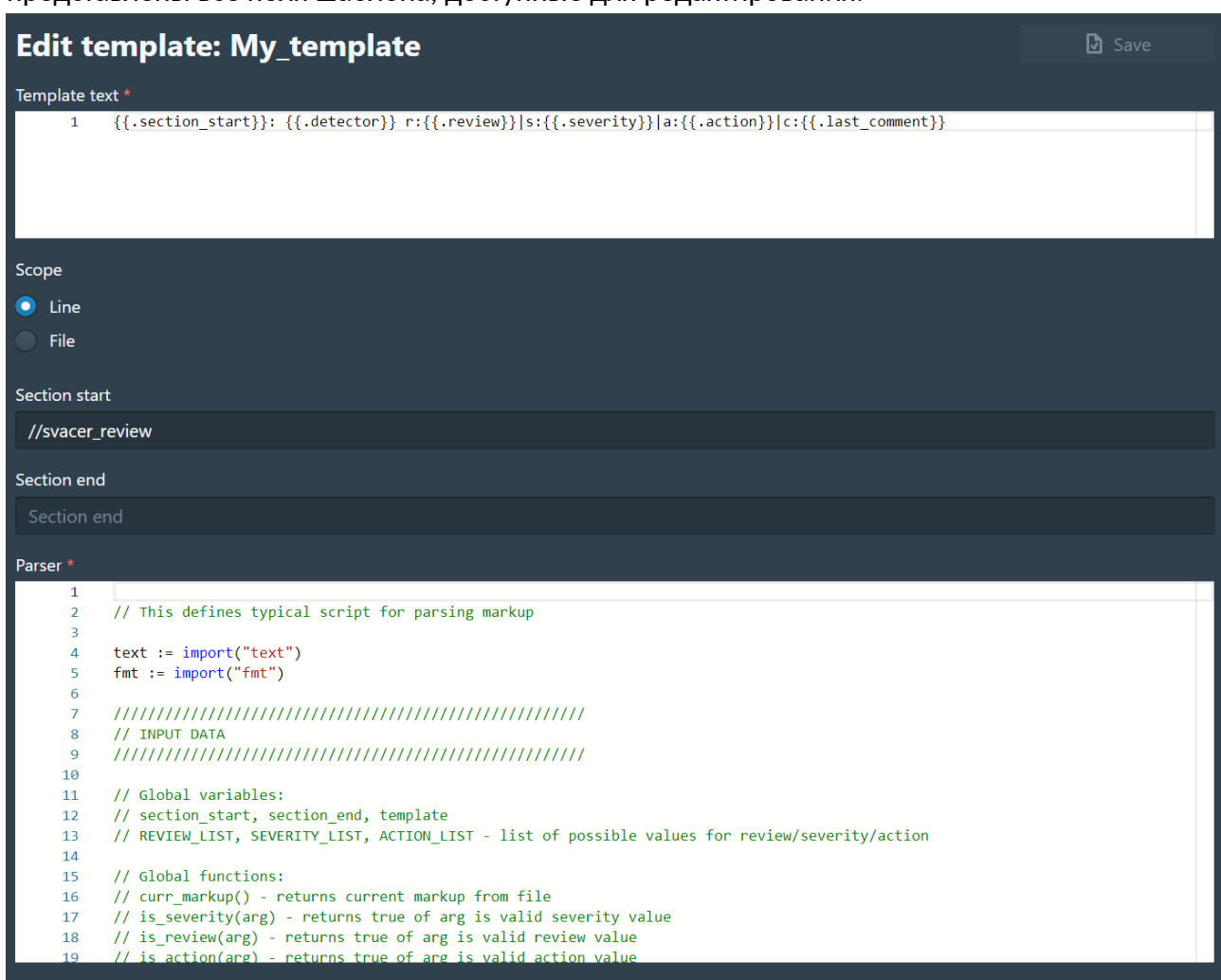


Рис. 45 — Редактирование шаблона разметки

Для создания или изменения шаблонов следует обратиться к разработчикам Svace History Server.

5.11 Поддержка протокола LDAP для аутентификации пользователей

5.11.1 Общий принцип аутентификации

Схема аутентификации с помощью протокола LDAP следующая:

1. Сервер запускается с указанием конфигурационного файла с настройками для LDAP аутентификации
2. В форме аутентификации пользователь выбирает нужный тип аутентификации (svacer или LDAP)
3. При выборе аутентификации LDAP, после заполнения необходимых параметров, запрос на аутентификацию отправляется на сервер svacer
4. Сервер Svacer, используя настройки из конфигурационного файла, обращается к внешнему LDAP серверу и проводит проверку пользователя
5. В случае успешной аутентификации, в БД сервера Svacer создается запись (если ее еще нет) для пользователя, чтобы можно было управлять ролями данного пользователя
6. Сервер возвращает токен безопасности для дальнейшей работы, аналогичный токenu, возвращаемому при аутентификации локальных пользователей

Для каждого пользователя, успешно прошедшего проверку на LDAP сервере, создается локальная запись в БД Svacer. Управлять таким пользователем можно как и обычным пользователем, за исключением:

- Пользователю нельзя сменить пароль
- Пользователь не может сменить пароль самостоятельно
- В случае отсутствия связи между сервером Svacer и сервером LDAP, данный пользователь не сможет зайти в систему

5.11.2 Запуск сервера с поддержкой LDAP

Сервер может быть запущен как с поддержкой аутентификации через LDAP, так и без. Для того, чтобы включить аутентификацию через LDAP необходимо указать конфигурационный файл. Типовой пример конфигурационного файла для подключения к контроллеру домена MS ActiveDirectory или серверу openLDAP можно получить с помощью команды:

svacer ldap print

Запуск сервера с поддержкой LDAP осуществляется командой:

svacer server --ldap ldap.json

При запуске сервер svacer формирует список доступных для LDAP аутентификации серверов и выводит соответствующее сообщение. Ниже приведен лог сервера, запущенного с поддержкой протокола LDAP.

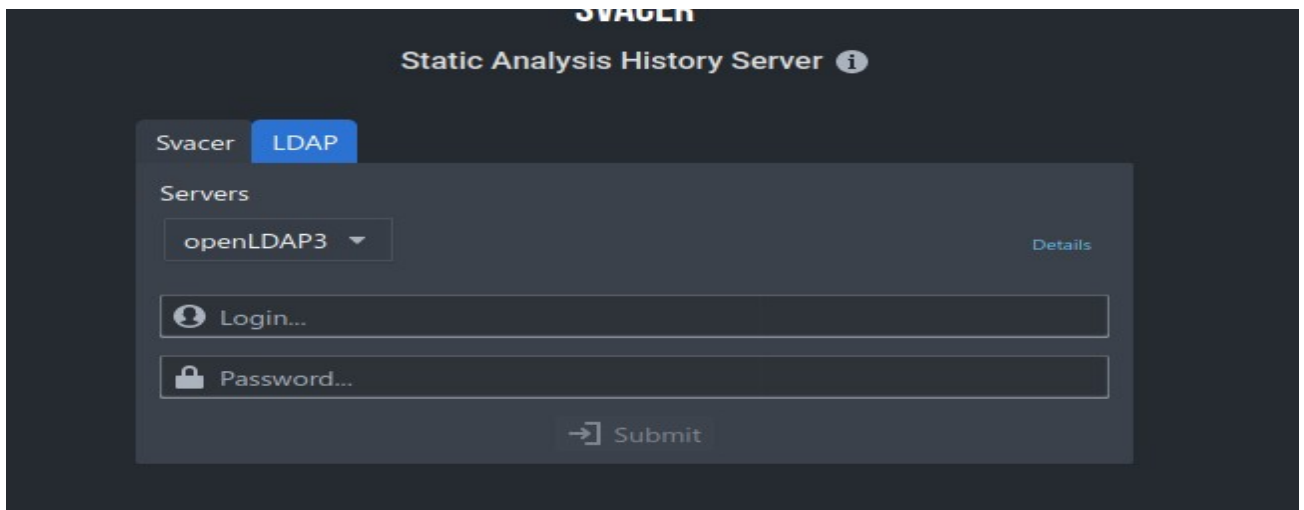
```

2022-07-13T10:46:02.730+0300    info    Object store internal ID e94f4b1c-893a-47ea-b096-60f7e1522288
2022-07-13T10:46:02.780+0300    info    SQL Schema version 124862f7289e83f1fcc7ca52ea62d7da144ea194
2022-07-13T10:46:02.781+0300    info    Connected to object storage at /home/chseva/.cache/svacer
2022-07-13T10:46:02.790+0300    info    Loading ldap configuration from file: ldap.json
2022-07-13T10:46:02.793+0300    info    LDAP server added: openLDAP3
2022-07-13T10:46:02.793+0300    info    Starting history server backend
2022-07-13T10:46:02.793+0300    info    gRPC server running on :3002
2022-07-13T10:46:02.802+0300    info    Performing migration fix-serials

```

После запуска сервера svacer с поддержкой LDAP в GUI становится активна вкладка LDAP.

Для аутентификации с помощью LDAP необходимо выбрать сервер из выпадающего списка.



При нажатии на кнопку **Details** появляется более подробная информация о сервере и его доступности (параметр **Alive**). Доступность сервера LDAP проверяется в момент запуска сервера Svacer, при аутентификации пользователя, а также (в случае наличия соответствующего параметра в конфигурационном файле) периодически.

В случае конфигурации сервера Svacer с поддержкой LDAP можно отключить возможность аутентификации пользователей встроенными средствами сервера Svacer (вкладка Svacer на странице входа). В этом случае вкладка Svacer на странице входа будет неактивной и пользователи смогут заходить в систему только с помощью LDAP аутентификации. Данное поведение распространяется и на работу с сервером с помощью интерфейса командной строки. Для отключения возможности аутентификации встроенными средствами сервера Svacer нужно использовать опцию **disable-svacer-auth**.

Пример: **svacer server --ldap ldap.json --disable-svacer-auth**

5.11.3 Конфигурационный файл сервера для поддержки LDAP

Типовой пример конфигурации представлен ниже.

```
{
```

```
"servers": [  
  {  
    "name": "active_directory",  
    "basedn": "dc=domain,dc=home,dc=org",  
    "useGroup": true,  
    "hideURL": true,  
    "checkAlivePeriod": 60,  
    "connection": {  
      "url": "ldap://192.168.11.71:389",  
      "connectAs": "cn=ldap_service,cn=users,dc=domain,dc=home,dc=org",  
      "password": "12345678",  
      "ignoreCertCheck": true  
    },  
    "group": {  
      "basedn": "cn=users",  
      "filter": "(&(objectCategory=Group)(cn=dep_devops))",  
      "userMember": "member",  
      "display": "cn"  
    },  
    "user": {  
      "basedn": "cn=users",  
      "filter": "(&(objectCategory=Person)(sAMAccountName=*))",  
      "login": "sAMAccountName",  
      "group": "dn",  
      "info": {  
        "display": "cn",  
        "email": "email",  
        "firstName": "givenName",  
        "lastName": "sn"  
      }  
    },  
    "enabled": true  
  },  
  {  
    "name": "open_ldap",  
    "basedn": "dc=example,dc=com",
```

```

    "useGroup": true,
    "connection": {
        "url": "ldap://127.0.0.1:389",
        "connectAs": "cn=admin,dc=example,dc=com",
        "password": "12345678",
        "ignoreCertCheck": true,
        "caCertFiles":["/etc/ssl/certs/a*", "/etc/ssl/certs/my_certs/*"]
    },
    "group": {
        "basedn": "ou=groups",
        "filter": "(&(objectClass=groupOfNames)(cn=svacer))",
        "userMember": "member",
        "display": "cn"
    },
    "user": {
        "basedn": "ou=users",
        "filter": "(&(objectClass=PosixAccount))",
        "login": "uid",
        "group": "dn",
        "info": {
            "display": "dn",
            "email": "mail",
            "firstName": "givenName",
            "lastName": "sn"
        }
    },
    "enabled": true
}
]
}

```

В конфигурации можно указать несколько серверов. Все сервера в совокупности рассматриваются как единый источник проверки пользователей. То есть если на двух серверах будут два пользователя с одинаковым логином, они будут считаться идентичными. Ниже приводится описание параметров конфигурации.

- name — произвольное имя (уникальное для каждого сервера)
- basedn — DN относительно которого будут проводиться все операции в LDAP каталоге
- useGroup — проверять ли принадлежность пользователя к группе
- hideURL — не передавать сетевой адрес LDAP сервера. Конечные пользователи будут видеть в выпадающем списке только имя сервера, указанное в поле name. В поле URL будет пустая строка.
- checkAlivePeriod — Промежуток времени в секундах, который должен проходить между периодическими проверками доступности сервера LDAP. Если параметр равен 0 или не указан, такая проверка не проводится.
- connection — подсекция, описывающая параметры соединения с сервером
 - url — адрес сервера, формата: ldap[s]://domain:port
 - connectAs — имя пользователя в формате DN, из под которого будут производиться операции в LDAP каталоге
 - password — пароль пользователя
 - ignoreCheck — не проверять сертификат в случае ldaps:// соединения. Полезно при самоподписанных сертификатах. Если флаг установлен в значение false, необходимо установить значение поля caCertFiles.
 - CaCertFiles — массив значений, описывающих какие файлы корневых сертификатов добавлять в конфигурацию svacer. Формат файлов сертификатов — PEM
- group — подсекция, описывает параметры проверки принадлежности пользователя к группе
 - basedn — RDN, добавляемая к общей DN. Относительно этого DN производится поиск группы
 - filter — фильтр, используемый при поиске группы. Необходимо, если надо указать несколько групп
 - userMember — атрибут в группе, содержащий пользователей, принадлежащих группе
 - display — атрибут, содержащий имя группы для отображения
- user — подсекция, описывает параметры получения информации о пользователе
 - basedn — RDN, добавляемая к общей DN. Относительно этого DN производится поиск группы
 - filter — фильтр, используемый при поиске пользователей.
 - login — атрибут, в котором находится login пользователя
 - group — атрибут в пользователе, на который ссылается атрибут group.userMember (обычно это «dn» пользователя)
 - info — подсекция описывающая в каких атрибутах находится информация о пользователе
 - display — атрибут, содержащий полное имя пользователя для отображения (ФИО)

- email — атрибут с почтовым адресом
- firstName — атрибут с именем
- lastName — атрибут с фамилией
- enabled — использовать или нет данный сервер

5.11.4 Использование CLI сервера Svacer с поддержкой LDAP

Для выполнения некоторых действий в cli с помощью команды svacer требуется аутентификация пользователя. Для аутентификации с помощью ldap сервера необходимо указать в команде кроме имени пользователя и пароля имя сервера с помощью флага ldap_server (имя сервера регистр-зависимо). Например:

```
./svacer quickdiff --user loginok --password loginok --ldap_server openLDAP2 --project zstd --branch zstd_14 --snapshot_v1 v1.4.1 --snapshot_v2 zstd_140
```

Если флаг ldap_server не указан, аутентификация производится механизмами сервера svacer, как и в предыдущих версиях. Параметр ldap_server можно опустить, указав сервер в переменной окружения SVACER_LDAP_SERVER. Например:

```
SVACER_LDAP_SERVER="openLDAP2" ./svacer quickdiff --user loginok --password loginok --project zstd --branch zstd_14 --snapshot_v1 v1.4.1 --snapshot_v2 zstd_140
```

Для получения списка поддерживаемых свейсером LDAP серверов можно использовать команду:

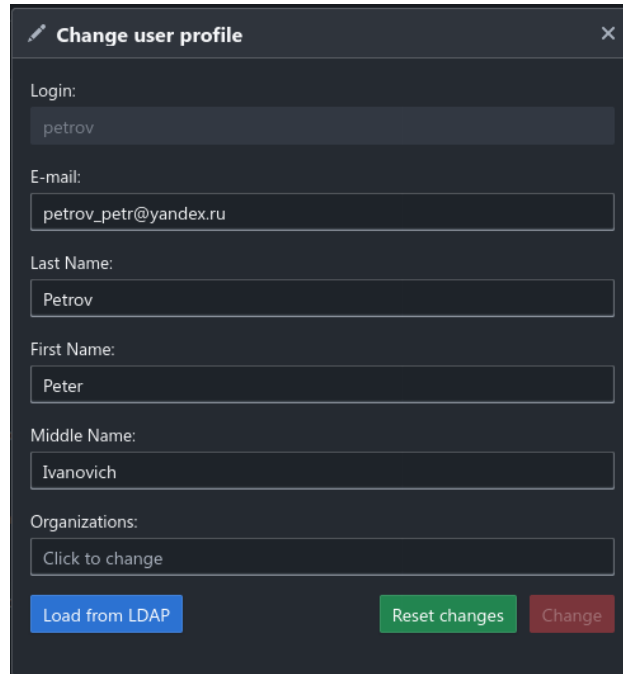
```
./svacer ldap servers --host 127.0.0.1 --port 8080
```

Вывод содержит два столбца (имя и URL сервера). В качестве значения для параметра ldap_server необходимо использовать содержимое первого столбца.

5.11.5 Обновление данных LDAP пользователей

Для каждого LDAP пользователя создается соответствующая ему запись в БД Svacer, в которую включается ФИО из LDAP и адрес его электронной почты. Через какое-то время данные пользователей могут меняться, в этом случае возможно обновить данные LDAP пользователей, хранящиеся в Svacer, синхронизировав их с данными LDAP сервера.

Для обновления данных одного пользователя можно воспользоваться окном редактирования пользователя, в котором можно нажать кнопку **Load from LDAP**. В соответствующие окна будут добавлены данные, загруженные из LDAP сервера.



Change user profile

Login:
petrov

E-mail:
petrov_petr@yandex.ru

Last Name:
Petrov

First Name:
Peter

Middle Name:
Ivanovich

Organizations:
Click to change

Load from LDAP Reset changes Change

Также данные пользователей можно обновить с помощью интерфейса командной строки. Для этого можно использовать подкоманду **updateUsers** команды **ldap**. Общий синтаксис команды выглядит следующим образом:

```
./svacer ldap updateUsers --onlyEmpty
```

Флаг **onlyEmpty** позволяет обновлять только пустые поля ФИО и eMail. Заполненные поля не будут изменены. Команда обновит данные всех пользователей, записи о которых имеются в базе Svacer.

5.12 Поддержка сквозной аутентификации

Svacer поддерживает сквозную аутентификацию через reverse proxy. Для этого используется опция proxy-mode. При включении этой опции есть возможность проходить аутентификацию (используя HTTP Basic Authentication) на reverse proxy сервере и использовать эти данные для аутентификации на сервере Svacer.

При включенном proxy-mode используется header с прокси **Authorization Basic ...**, откуда svacer берет имя пользователя и пароль, проверяет и аутентифицирует пользователя, если аккаунт с таким логином и паролем существует в БД пользователей Svacer.

Пример запуска сервера с поддержкой сквозной аутентификации:

```
./svacer server --proxy-mode
```

5.13 Поддержка OAuth

Svacer поддерживает авторизацию по протоколу OAuth. Для авторизации используются запросы **GET /api/oauth/authorize** и **POST /api/oauth/token** в соответствии со стандартом.

5.13.1 API-вызовы для управления клиентами

- GET /api/oauth/clients — получение всех клиентов для текущего пользователя или всех клиентов текущего сервера для админа;
- GET /api/oauth/client?client_id={client_id} — получение клиента по id;
- POST /api/oauth/client — создание нового клиента;
- PUT /api/oauth/client — изменение данных клиента;
- DELETE /api/oauth/client/{client_id} — удаление клиента.

Клиенты могут быть зарегистрированы на панели Settings > OAuth Clients. Клиент содержит в себе следующие поля:

- ID — генерируется на сервере и не изменяется впоследствии;
- Name — задается пользователем при создании и может впоследствии меняться; отображается на странице подтверждения авторизации;
- Secret — генерируется на сервере, не изменяется впоследствии и возвращается только в ответе на запрос создания;
- Domain — задается пользователем и может впоследствии меняться; используется как redirect_uri, если в запросе authorize отсутствует это поле;
- UserId — извлекается из сессии во время создания и не изменяется впоследствии.

6 Панель отчетов

Веб-интерфейс предоставляет панель с отчетами, которая показывает некоторую статистику по данным. На текущий момент предоставляются два типа отчета:

1. General Report — сравнение двух снимков и показ детальной статистики по результатам сравнения.
2. Trends — показ изменения числа найденных предупреждений на диапазоне между двумя снимками.

При вычислении отчетов используется следующая терминология касательно исправленных предупреждений:

Термин	Семантика
fixed indirectly	Те маркеры которых нет в текущем снимке (т.е. какие в MISSING если делать сравнение снимков).
fixed	Все маркеры, какие fixed indirectly , плюс те, которые имели review = Confirmed и action = Fix Submitted

7 Работа с SARIF

7.1 Генерация SARIF

Поддерживается генерация файла SARIF, который может быть прочитан при помощи расширения Microsoft SARIF Viewer, GitHub или DefectDojo.

Итоговый файл содержит сопоставление внутренних значений критичности маркеров со значениями уровня SARIF по следующему правилу:

- Critical -> error
- Major -> warning
- Minor -> note
- Normal -> none

При импорте результатов анализа в SARIF должны быть предоставлены данные для подключения к серверу, так как значения критичности маркеров берутся оттуда.

Общие параметры для импорта:

- out-file, o — имя выходного файла или «-» для вывода в stdout
- scheme — печать примера схемы выходного файла SARIF
- include-contents — включать контент файлов, в которых обнаружены предупреждения

7.1.1 Генерация из .svres

При импорте из файла svres используются дополнительные параметры:

- svace — путь к исполняемому файлу svace
- svace-dir — интерпретировать путь, как путь к директории .svace-dir

Пример команды для генерации файла SARIF:

```
./svacer sarif -o <path_to_result_file> --svace <path_to_svace>  
<path_to_project_or_svace-dir>
```

7.1.2 Генерация из данных с сервера

При импорте из данных с сервера используются следующие дополнительные параметры:

- from-server — опция для получения данных с сервера
- project — имя или id проекта
- branch — имя или id ветки (master по умолчанию)
- snapshot — имя или id снимка (last по умолчанию)

Также, для корректного получения необходимого проекта, должен быть указан пользователь, у которого есть доступ к этому проекту.

Пример команды для генерации файла SARIF из данных с сервера:

```
./svacer sarif -o <path_to_result_file> --from-server --user <user> --password  
<password> --project <project_name>
```

7.2 Загрузка на сервер из SARIF

При загрузке данных на сервер используются следующие параметры:

- read-file, r — файл SARIF, из которого необходимо читать данные
- project — название проекта на сервере (по умолчанию совпадает с именем файла)
- branch — название ветки на сервере (по умолчанию master)
- snapshot — название снимка на сервере (по умолчанию время загрузки)
- skip-source — пропускать исходный код файлов с предупреждениями
- path-prefix — заменять префиксы в названии файлов (prefix:replacement)

Пример команды для загрузки на сервер:

```
./svacer sarif upload -r <input_file>
```

8 Интеграция с Visual Studio Code

В состав дистрибутива Svacer входит extension для Visual Studio Code, обеспечивающий взаимодействие пользователя из VS Code с сервером Svacer. Подробная информация может быть найдена в документации на данное расширение в дистрибутиве Svacer по пути [integrations/vscode/readme.pdf](#).