АКЦИОНЕРНОЕ ОБЩЕСТВО
«ЭШЕЛОН ТЕХНОЛОГИИ»

УТВЕРЖДЕН
АПДГ.00101-01 34-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «АНАЛИЗАТОР ИСХОДНЫХ ТЕКСТОВ ПРОГРАММ «АК-ВС 3»

Руководство оператора

АПДГ.00101-01 34

Листов 98

2022

Подп. и дата
Инв. №
Взам. инв.
Подп. и дата
Инв. №
дубл.
№
подл.

АННОТАЦИЯ

В документе содержатся сведения о назначении, функциях и особенностях эксплуатации изделия «Анализатор исходных текстов программ «АК-ВС 3» (далее - «АК-ВС 3»).

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

«АК-ВС 3» применяется для автоматизации процесса проведения статического и динамического анализов исходных кодов программ и построения основных отчетов согласно требованиям руководящего документа «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей» (Гостехкомиссии, 1999 год) до первого уровня контроля включительно, а также согласно требованиям «Методики выявления уязвимостей и недекларированных возможностей в программном обеспечении» (ФСТЭК, 2020 год).

«АК-ВС 3» анализирует исходные тексты программ, написанные на языках программирования: C (ANSI C, C90, C99, C11, C17), C++ (C++98, C++2003, C++11, C++14, C++17, частично C++20), Java (SE 8, частично SE 11), C# (5.0 полностью, частично до 9.0).

Количественных ограничений на объем и количество файлов исходных текстов, а также на размер отдельного файла нет. Допустимые кодировки файлов: ASCII, windows-1251, UTF-8, ISO-2022-JP, ISO-2022-CN, ISO-2022-KR, ISO-8859-5, ISO-8859-7, ISO-8859-8, Big5, GB18030, EUC-JP, EUC-KR, EUC-TW, Shift_JIS, IBM855, IBM866, KOI8-R, x-MacCyrillic, windows-1252, windows-1253, windows-1255, UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, GB2312, ISO-10646-UCS-4-3412, ISO-10646-UCS-4-2143.

1.1. Справочные материалы и вспомогательные программы

В главном меню веб-интерфейса программы доступны материалы по методам исследования программ, в том числе фаззеру Afl и средству полносистемного динамического анализа Код2. В материалах содержатся примеры исследований и имеется возможность загрузить инструменты.

1.2. Статический анализ исходных текстов программ

В рамках статического анализа «АК-ВС 3» выполняет автоматическое построение отчетов в форме:

- Таблицы информационных объектов (ИО);
- Таблицы функциональных объектов (ФО);
- Список невызываемых ФО;
- Список неопределенных ФО;
- Таблица связей ФО по управлению;
- Маршруты выполнения ФО;

- Таблица связей ФО по информации;
- Критические маршруты выполнения ФО;
- Блок-схемы ФО;
- Список базовых блоков (ББ);
- Таблицы наличия заданных конструкций в исходных текстах;
- Список предполагаемых дефектов.

Статический анализ производится сигнатурным и межпроцедурным контекстно-чувствительным методом, чувствительным к путям выполнения. С помощью мониторов сборки производится перехват процесса сборки приложения, в результате чего «АК-ВС 3» способен проводить анализ путей выполнения, проходящих через несколько единиц компиляции в рамках одной исполняемой программы.

1.3. Динамический анализ программ с помощью датчиков

В рамках динамического анализа «АК-ВС 3» выполняет:

- внедрение датчиков в исходные тексты программ;
- анализ логов (журналов) отработки внедренных датчиков;
- генерацию отчета проведения динамического анализа.

1.4. Полносистемный динамический анализ с помощью Код2

Инструмент Код2 является эмулятором компьютера с архитектурой x86 или AMD64. Он позволяет:

- запускать различные ОС, например, Linux и Windows, с существенным замедлением функционирования;
- записывать сценарии выполнения программы, выполняющейся в эмуляторе;
- воспроизводить записанные сценарии;
- при воспроизведении производить доп. исследования, например, изучать распространение помеченных данных по памяти, передачу в сетевой интерфейс, запись в файловую систему;
- осуществлять удалённую отладку во время воспроизведения сценария, в том числе остановку по достижении данного момента и движение по времени в прошлое от любой точки останова в пределах записанного сценария;
- выводить выполнявшиеся ассемблерные команды.

1.5. Основные подсистемы и компоненты

«АК-ВС 3» состоит из следующих подсистем и компонентов:

Подсистема стенда для сборки исследуемого ПО

- мониторы сборки;

Подсистема сервера:

- модуль REST-сервиса;
- менеджер задач;
- загрузчик исходных текстов;
- конвертор исходных текстов в кодировку UTF-8;
- синтаксический анализатор исходных текстов;
- сигнатурный статический анализатор;
- Dataflow-анализатор;
- статический анализатор с межпроцедурным/межмодульным анализом, чувствительным к путям выполнения;
- модуль вставки датчиков;
- модуль построения отчетов анализа исходных текстов программ;

Подсистема клиента:

- средство просмотра исходных текстов «Эшелониум»;

Подсистема эмуляции Код2:

- инструмент Код2 для полносистемного динамического анализа.

Модуль REST-сервиса предназначен для получения задач от пользователя и запуска менеджера задач.

Менеджер задач предназначен для управления задачами, поступающими от пользователя и управления запуском всех компонентов «АК-ВС 3».

Загрузчик исходных текстов предназначен для загрузки файлов с исходными текстами в проект для дальнейшего анализа. Исходные тексты загружаются в архиве в формате ZIP или 7z.

Конвертор исходных текстов в кодировку UTF-8 предназначен для преобразования входных данных в кодировку UTF-8.

Синтаксический анализатор исходных текстов предназначен для выделения из исходных текстов деревьев разбора и размещения их в заданном виде в таблицы в формате XML. Синтаксический анализатор преобразует исходные тексты следующих языков программирования: C, C++, C#, Java.

Сигнатурный статический анализатор ищет ошибки в деревьях разбора путём сопоставления их с образцами (сигнатурами).

Статический анализатор с межпроцедурным/межмодульным анализом, чувствительны путям выполнения, проводит лексический, синтаксический анализ (независимо от вышеупомянутого модуля синтаксического анализатора), символьное выполнение фрагментов программного кода и выявление путей исполнения, ведущих к ошибкам.

Модуль вставки датчиков предназначен для подготовки к проведению динамического анализа. Модуль запускается менеджером задач после завершения работы синтаксического анализатора и вставляет датчики в исходные тексты. Логи отработки датчиков используются для построения отчетов по динамическому анализу.

Модуль построения отчетов анализа исходных текстов программ предназначен для создания файлов отчетов. Данный модуль запускается менеджером задач после завершения работы синтаксического анализатора и модуля вставки датчиков.

Средство просмотра исходных текстов позволяет просматривать исходные тексты программ, при этом для поддерживаемых анализатором языков доступны функции поиска строки и перехода к определению.

Инструмент Код2 служит для эмуляции работы компьютера и записи сценария его выполнения и последующим изучением записанного сценария. Записанный сценарий представлен в виде описания последовательности исхода всех происходящих в машине недетерминированных событий. С помощью разработанных АО «Эшелон Технологии» подключаемых модулей Код2 позволяет проводить исследования тестовых сценариев методом помеченных данных, а также распечатывать полносистемные ассемблерные трассы.

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1. ЭВМ, необходимые для работы АК-ВС, и их роли

При работе «АК-ВС 3» требуются до 4 различных физических или виртуальных ЭВМ, выполняющих следующие роли (причём одна ЭВМ может исполнять сразу несколько ролей):

2.1.1. Стенд для сборки исследуемого ПО

На этом стенде собирается исследуемое ПО.

2.1.2. Стенд для запуска исследуемого ПО

На этом стенде проводятся динамические исследования.

2.1.3. Сервер АК-ВС

Сервер отвечает за структурный и статический анализ, построение отчётов и обслуживает веб-интерфейс.

2.1.4. Клиент АК-ВС

На клиенте пользователь управляет анализом, просматривает отчёты и исходные тексты в среде «Эшелониум».

2.2. Требования к ЭВМ

Компоненты «АК-ВС 3» устанавливается на ЭВМ, удовлетворяющие минимальным аппаратным и программным требованиям, представленным в табл. 1 – 4.

Таблица 1 — Требования к среде функционирования «АК-ВС 3», стенд для сборки ПО

Параметр	Значение
Процессор	64-битный Intel-совместимый
Объем оперативной памяти	4 Гбайт (рекомендуется 16 Гбайт)
Объем свободного места на жестком диске	20 Гбайт (рекомендуется 100 Гбайт)
Операционная система на стенде для сборки и исследования ПО	Linux Ubuntu 18.04 Астра Смоленск 1.6 Астра Смоленск 1.5 (по запросу)

Таблица 2 — Требования к среде функционирования «АК-ВС 3», сервер анализа

Параметр	Значение
Процессор	64-битный Intel-совместимый
Объем оперативной памяти	8 Гбайт (рекомендуется 16 Гбайт)
Объем свободного места на жестком диске	50 Гбайт (рекомендуется 100 Гбайт)
Порты	USB-порт для ключа защиты
Операционная система на стенде для сборки и исследования ПО	Linux Ubuntu 18.04

Таблица 3 — Требования к среде функционирования «АК-ВС 3», клиент

Параметр	Значение
Процессор	64-битный Intel-совместимый
Объем оперативной памяти	8 Гбайт (рекомендуется 16 Гбайт)
Объем свободного места на жестком диске	10 Гбайт (рекомендуется 100 Гбайт)
Операционная система на стенде для сборки и исследования ПО	Linux Ubuntu 18.04
Дополнительное ПО	Веб-браузер (Google Chrome версии 88 и выше, Mozilla Firefox версии 85 и выше)

Таблица 4 — Требования к среде функционирования «АК-ВС 3», стенд для проведения полносистемного динамического анализа Код2

Параметр	Значение
Процессор	64-битный Intel-совместимый
Оперативная память	16 Гбайт (рекомендуется 64 Гбайт)
Жесткий диск (свободное пространство)	200 Гбайт (рекомендуется 500 Гбайт)
Операционная система	Linux Ubuntu 18.04

Общих требований к стенду для запуска программ не предъявляется. Для проведения исследования методом помеченных данных необходимо создать виртуальную машину входящим в состав инструментом Код2, который основан на QEMU. Поддерживаются ОС семейства Linux с архитектурой x86 и x64, но набор функциональных возможностей зависит от конкретной версии ОС. Больше деталей см. в методических материалах по методу отслеживания помеченных данных.

«АК-ВС 3» обеспечивает выполнение функциональных возможностей при реализации следующих предварительных организационно-распорядительных мер:

- обеспечение сохранности оборудования и физической целостности системных блоков компьютеров;
- обеспечение свободной от вирусов программной среды компьютеров;
- обеспечение контроля изменения прикладной программной среды, исключение ввода в компьютер программных средств без гарантированной проверки.

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Установка и удаление программы

3.1.1. Установка программы с помощью консольных инсталляторов

Для установки понадобится подключение к репозиторию стандартных пакетов для ОС, на которой происходит установка. Если нет особых противопоказаний (например, они могут быть на стенде для сборки), нужно перед установкой выполнить команды:

```
sudo apt-get update
sudo apt-get upgrade
```

Дистрибутив «АК-ВС 3» для операционных систем семейства Linux представляет собой архив, который нужно распаковать на жёсткий диск:

```
cd ~
tar zxf /media/cdrom/AKVS-3.2.0-*.tgz
cd AKVS-3.2.0
```

В текущей директории будут находиться файлы с расширением .run, которые нужно выполнить, соответственно:

- на стенде для сборки исследуемого ПО:

  для Astra Linux 1.5:

  ```
  ./install_akvs_bench_3.2.0.astra-1.5.*.run
  ```

  Для остальных ОС:

  ```
  ./install_akvs_bench_3.2.0.run
  ```

- на стенде для запуска исследуемого ПО ничего устанавливать не нужно;
- на сервере:

  ```
  sudo ./install_akvs_server_3.2.0.run
  ```

- на клиенте:

  ```
  ./install_akvs_client_3.2.0.run
  ```

- на стенде для проведения полносистемного анализа Код2:

  ```
  ./install_akvs_kod2_3.2.0.run
  ```

Если одна ЭВМ совмещает несколько ролей, на ней нужно запустить все соответствующие инсталляторы. Пользователь, от имени которого запускаются инсталляторы, должен иметь права

на sudo. В ходе установки программа-установщик может задавать ряд вопросов, например, о пути установки и о добавлении программ в $PATH. Если что-то добавлено в $PATH, нужно перед использованием программы закрыть терминальную сессию и запустить её снова.

3.1.2. Установка программы с помощью графического инсталлятора

Инсталлятор запускается из исполняемого файла akvs_installer_3.2.0.AppImage. После запуска инсталлятор попросит ввести пароль sudo. Это необходимо для работы с обновлениями и возможности менять пользователя - владельца файлов сервера «АК-ВС 3».

В зависимости от того, установлен ли уже модуль сервера, главное окно может содержать (рис. Рисунок 2) или не содержать (рис. Рисунок 1) дополнительные пункты, позволяющие запустить/перезапустить сервер и сменить владельца файлов сервера.

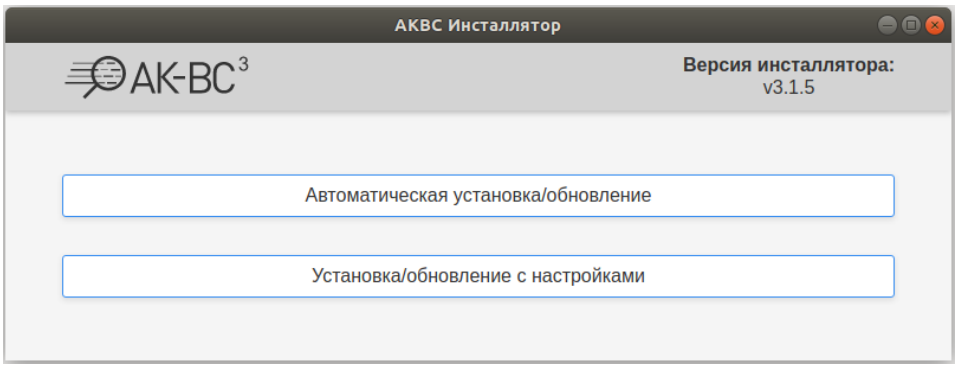

Рисунок 1 — Главная страница инсталлятора

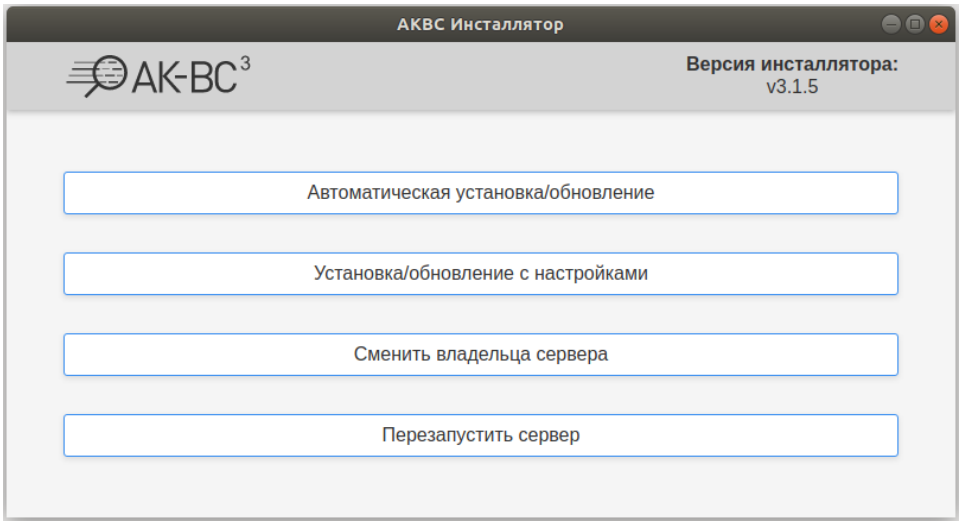

Рисунок 2 — Главная страница инсталлятора при установленном сервере

При выборе пункта «Автоматическая установка/обновление» будут установлены все модули «АК-ВС 3» с настройками по умолчанию, либо обновлены с сохранением текущих настроек, если какие-то модули уже установлены.

При выборе пункта «Установка/обновление с настройками» откроется окно с информацией об установленных модулях и выбором что нужно установить/обновить (рис. Рисунок 3).

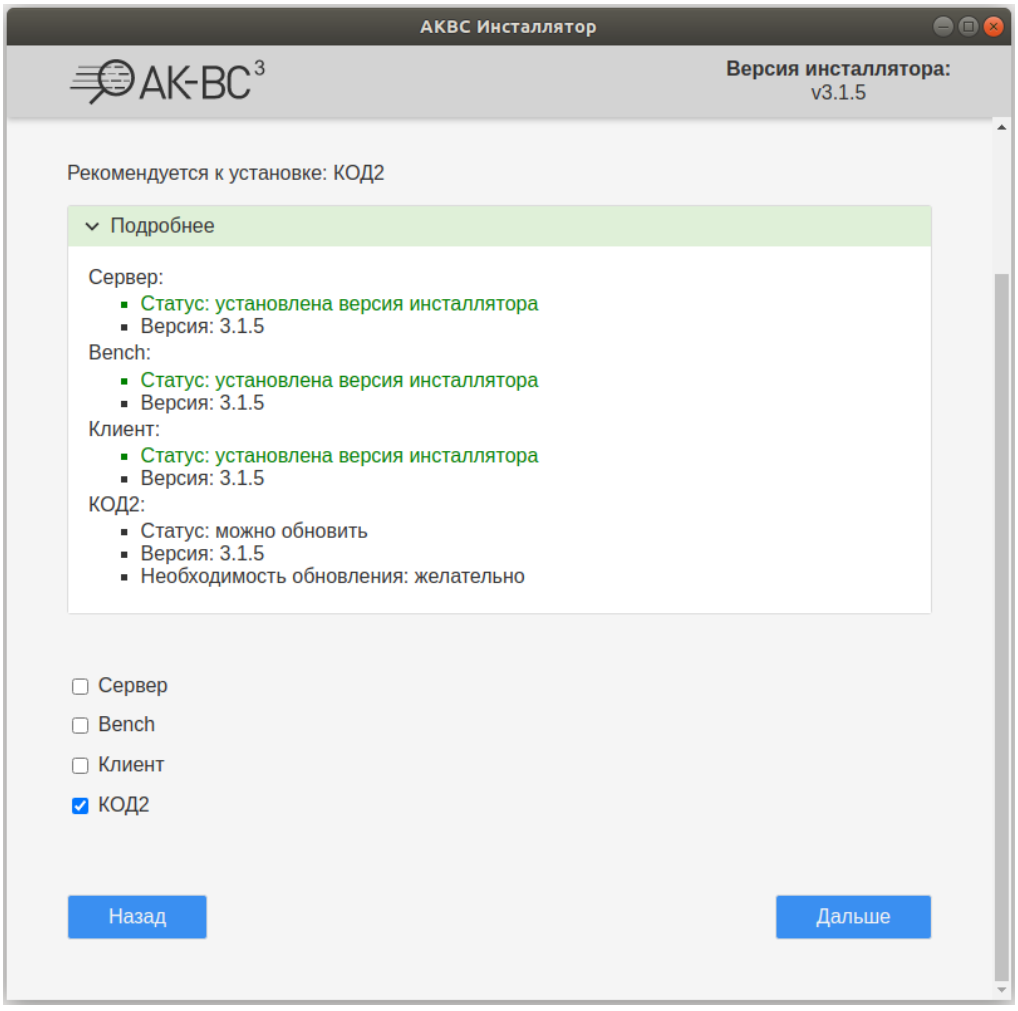

Рисунок 3 — Страница установка/обновление с настройками

3.1.3. Вопросы безопасности

Инсталлятор сервера позволяет выбрать пользователя, которому будут принадлежать файлы «АК-ВС 3», по умолчанию предлагается создать нового пользователя akvs, что дополнительно защищает от несанкционированного доступа.

При необходимости обеспечить отсутствие утечек исходных текстов изучаемых программ, а также найденных дефектов и их оценок экспертом, все машины, участвующие в анализе, должны быть отключены от сети интернет, а круг лиц, имеющих к ним доступ, ограничен.

После установки сразу же смените пароль администратора.

3.1.4. Удаление программы

- удалите директории, созданные при инсталляции, по умолчанию это:
  - на любом стенде - /opt/akvs_update_info;
  - на сервере - /opt/akvs_server;
  - на клиенте - ~/akvs_client, ~/.vscode-oss, ~/.config/Echelonium, ~/.echelonium-projects;
  - на стенде для сборки исследуемых программ - ~/akvs_bench;
  - на стенде для проведения полносистемного анализа Код2 - /opt/kod2.
- на сервере удалите базу данных «АК-ВС 3» с помощью команды mongo:

```
$ mongo
MongoDB shell version v3.6.3
> use akvs3
switched to db akvs3
>db.dropDatabase();
{ "dropped" : "akvs3", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

В выводе команды `showdbs` не должно выводиться имя базы, по умолчанию это akvs3.

Указанные процедуры также необходимо произвести для «чистой» переустановки подсистем «АК-ВС 3». При этом существующие проекты будут потеряны.

3.2. Запуск сервера

3.2.1. Подготовительный этап

Для работы с «АК-ВС 3» необходим ключ лицензии на сервере.

Активация программного ключа лицензии описана в п. 3.2.4..

3.2.2. Настройка веб-сервера

Чтобы пользователи могли обращаться к серверу «АК-ВС 3» с другого компьютера, нужно в директории установки сервера отредактировать файл конфигурации, по умолчанию это /opt/akvs_server/config/default.ini, задав в секции «[rest]» параметр «host=доменное-имя-сервера» (приложение Приложение 1).

3.2.3. Запуск сервера «АК-ВС 3»

Если в комплект поставки входит аппаратный ключ лицензии, то его необходимо подключить к серверной машине.

Для запуска «АК-ВС 3» в среде под управлением операционной системы семейства Linux необходимо выполнить команду:

```
# systemctl start akvs_server
```

Состояние и адрес сервера можно получить, выполнив команду:

```
# systemctl status akvs_server
```

Рисунок 4 — Адрес веб-интерфейс

После того как сервер будет успешно запущен, для получения доступа к веб-интерфейсу из другого браузера необходимо перейти по адресу сервера и порту 11000, например, http://127.0.0.1:11000 (рис. Рисунок 4).

Примечание. Настройка сервера «АК-ВС 3» через конфигурационный файл «default.ini» представлена в приложении 1.

Примечание. Дополнительную информацию об ошибках, возникших в процессе запуска веб-интерфейса, можно посмотреть в логе «data/logs/console.log».

В веб-интерфейсе откроется страница авторизации (рис 5). Для авторизации при первом входе используйте пару логин-пароль — «admin-admin». ВНИМАНИЕ! Обязательно смените пароль администратора.

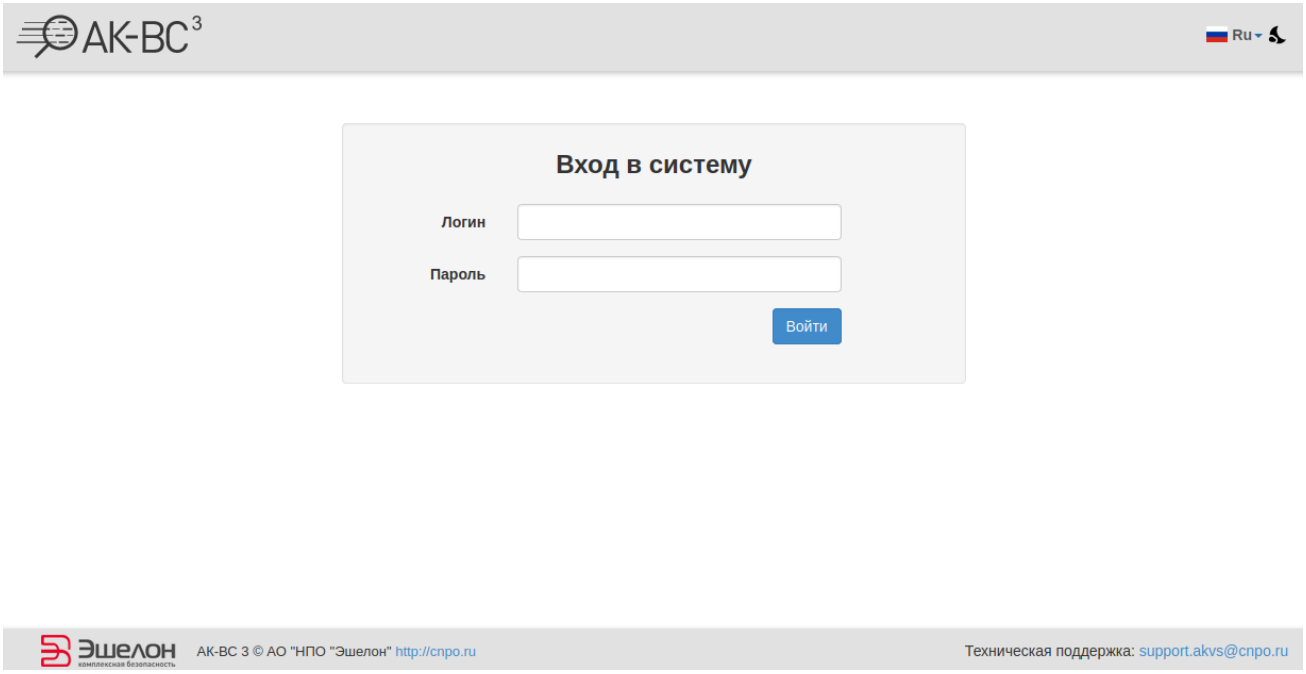

Рисунок 5 — Страница авторизации

3.2.4. Активация программного ключа лицензии

Если используется программный ключ лицензии (файл с расширением «.wbb»), то после авторизации будет предложено загрузить его, как показано на рисунке Рисунок 6.

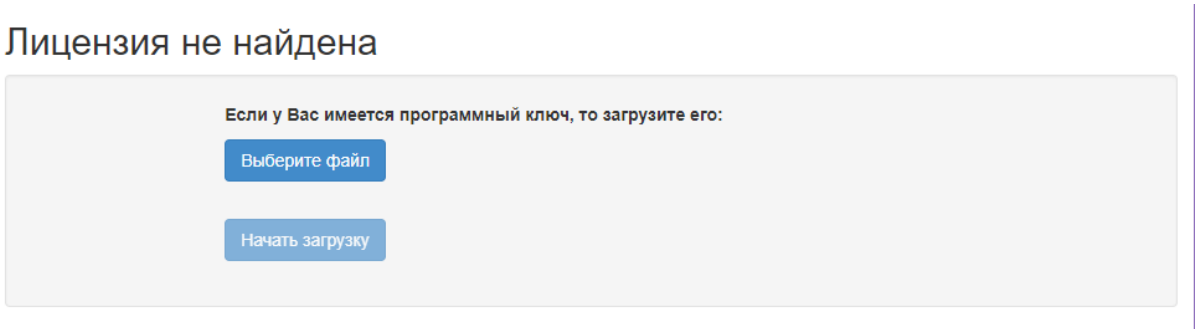

Рисунок 6 — Загрузка программного ключа

Для загрузки программного ключа нажмите кнопку «Выберите файл», укажите файл программного ключа и нажмите кнопку «Открыть». После загрузки ключа либо отобразится ключ

инсталляции (последовательность букв и цифр) (рис. Рисунок 7), либо будет предложено воспользоваться стандартной процедурой создания файла запроса лицензии Codemeter. Полученный артефакт следует отправить на электронную почту технической поддержки «АК-ВС 3» support.akvs@cnpo.ru.

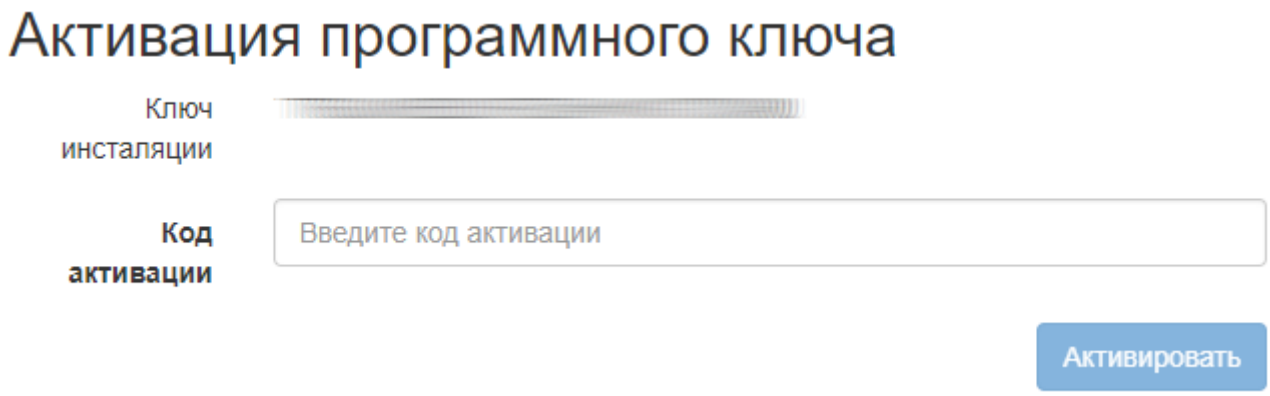

Рисунок 7 — Активация программного ключа

В ответном письме для первого случая будет направлен код активации, который необходимо ввести в соответствующее поле формы, и нажать кнопку «Активировать» (рис. 7). Во втором случае будет прислан файл активации, который нужно импортировать в систему лицензий Codemeter.

3.3. Работа с программой – главная страница веб-приложения

После авторизации откроется главная страница (рис. 8).

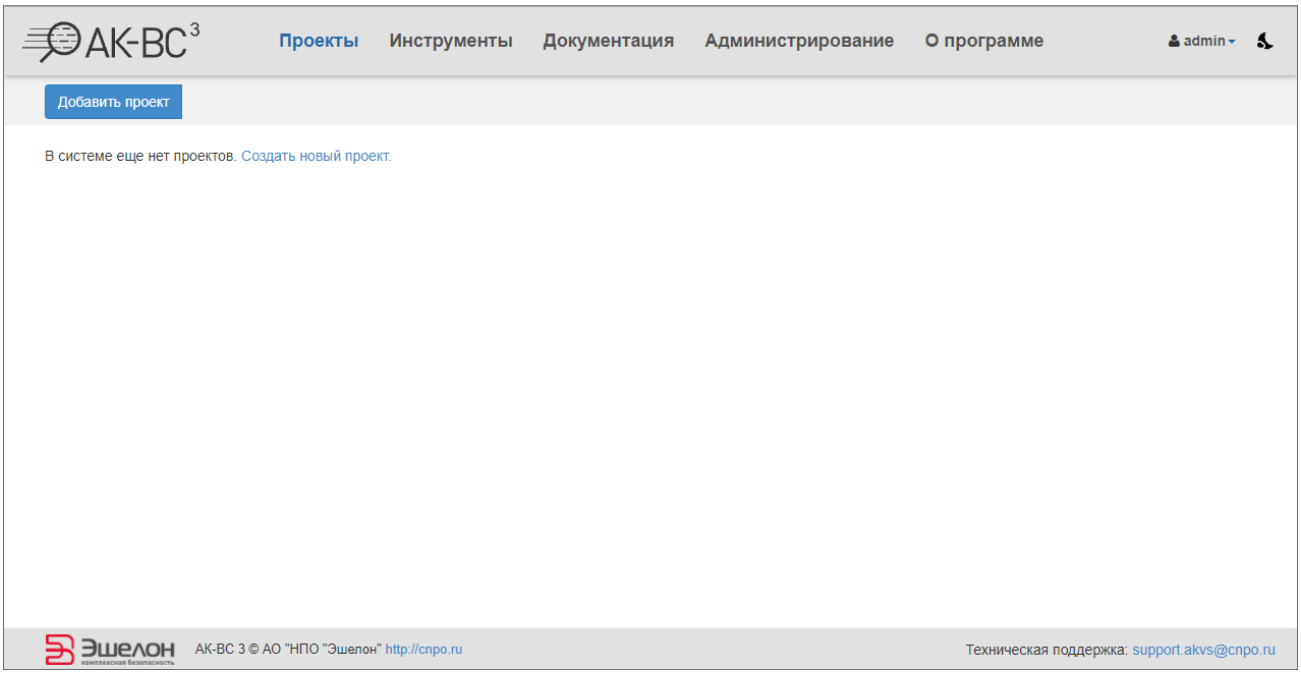

Рисунок 8 — Главная страница веб-приложения

На главной странице расположен список проектов, а также содержатся вкладки: «Инструменты», «Документация», «Администрирование» и «О программе».

3.3.1. Подготовка исходных текстов к анализу

Понять, что является исходными текстами программы, можно, только анализируя сборочные скрипты. Сборка является сложным процессом, выполняемым специализированными программами, такими, как automake, conf, cmake, maven и т.п., на основании файлов проекта, или скриптами произвольной формы. Поэтому «АК-ВС 3» собирает информацию о составе исходных текстов проекта, наблюдая за реально выполняемым процессом сборки. Основное требование для успешного перехвата сборки – это должна быть сборка приложения «с нуля». Повторная сборка зачастую пропускает те файлы, которые уже были созданы. Поскольку штатные средства очистки сборки (make clean) пишут разработчики, они могут содержать ошибки и не всегда срабатывать правильно. Это нужно учитывать при подготовке исходных текстов. В некоторых случаях, например, при генерации, перемещении, удалении исходных текстов в ходе сборки, «АК-ВС 3» не может правильно определить состав исходных текстов, это проявляется в ошибках на этапе анализа. В этом случае может понадобиться вручную составить или поправить файлы.

3.3.2. Файлы конфигурации синтаксических анализаторов C/C++

Каждый файл конфигурации сигнатурного анализа имеет название «parser-XXX», где XXX — сокращенное название языка (например, cpp, cs) и содержится в директории «config». Для C/C++ также существует файл relations-cpp, который содержит информацию о линковке (указывает, какие исходные тексты вошли в какие исполняемые модули). Для АЧКПВ и МКЧА в Java и C# после утилиты подготовки конфигурации архив содержит дополнительную машиночитаемую базу данных, не предназначенную для ручной обработки пользователем.

3.3.3. Конфигурация синтаксического анализатора C/C++

Языки программирования C/C++ используют директивы препроцессора. Препроцессор - это гибкий инструмент, который часто применяется в проектах. Однако в задачах анализа исходных текстов, написанных на C/C++, препроцессор становится серьёзным препятствием, так как он меняет вид и структуру кода.

В связи с этим необходимо настроить проект, чтобы повысить качество анализа.

Настройка проекта включает в себя:

- указание путей до заголовочных файлов;
- загрузку сторонних заголовочных файлов (например, используемых библиотек) и указание путей до них;
- указание значений макросов;
- указание прочих параметров компиляции (например, используемых стандартов).

Рассмотрим пример того, что произойдет, если не выполнить настройку:

Файл «a.h» (рис. 9):

```
1   #define MY_IF(a) if(a) {
```

Рисунок 9 — Пример заголовочного файла проекта

Файл «a.cpp» (рис. Рисунок 10):

```
1   #include "a.h"
2
3   int main() {
4       int b = 1;
5       MY_IF(b)
6           b++;
7       }
8
9       my_function();
10  }
```

Рисунок 10 — Пример исходного файла проекта

Если в процессе анализа файл «a.h» не будет найден, то синтаксический анализатор примет за конец функции «main» фигурную скобку на седьмой строчке вместо десятой.

В «АК-ВС 3» имеется утилита, которая поможет сконфигурировать проекта в автоматическом режиме.

3.3.3.1. Утилита автоматической конфигурации для проектов на C/C++в среде Linux

В зависимости от выбора, сделанного при установке, утилита cppconf_linux.py может быть доступна по пути поиска на сборочном стенде. Если она не доступна, её можно найти в директории развёртывания программы установки для сборочного стенда.

Скрипт выполняет команду для сборки через утилиту strace, которая отслеживает системные вызовы. Лог strace записывается в «/tmp/_cppconfig_log», на основе которого скрипт создает архив «src_config.zip» в директории запуска.

Примечание. Для работы утилиты требуются установленные программы strace 5.7 или выше и Python версии 2.7 или 3. Python и strace устанавливаются из пакетов во время развёртывания дистрибутива для сборочного стенда.

Алгоритм работы, если проект можно собрать с использованием strace, представлен ниже.

Запустите скрипт командой:

```
cppconf_linux.py -s команда_сборки
```

Пример:

```
cppconf_linux.py -s make
```

Результат работы утилиты - архив «src_config.zip» с исходным текстом проекта, конфигурационным файлом и используемыми сторонними заголовочными файлами.

1. Анализатор АЧКПВ и МКЧА не полностью совместим с gcc < 5.2. При использовании такого компилятора будет выдано предупреждение.
2. Скрипты сборки зачастую генерируют, удаляют, перемещают или переименовывают файлы. В них может применяться libtool, в результате чего имя объектного модуля может измениться, производиться конкатенация объектных модулей и т.п. Во всех этих случаях монитор не сможет правильно составить файлы конфигурации. Это приведёт к ошибкам при анализе.
3. Анализатор ограничен по масштабу проектов, которые он способен обработать. Например, при анализе ОС Linux не стоит пытаться проанализировать все исходные тексты ОС целиком. Анализируйте отдельные пакеты.
4. Отслеживание линковки происходит только для статической линковки. Линковка разделяемых библиотек не перехватывается и межмодульный анализ через границы разделяемых библиотек не проводится.

3.3.3.2. Утилита автоматической конфигурации для проектов на C/C++в среде Windows

Для создания архива с конфигурацией проекта на языках C/C++ в среде Windows используется утилита CppConfMonitor, которая поставляется в виде архива monitor.zip. Архив содержит x64 и x86 версии, использовать x86 на 64-битных системах не рекомендуется.

Для функционирования программа требует права администратора, так как использует системные средства ОС Windows.

Для перехвата процессов и обнаружения прочтённых файлов CppConfMonitor использует вспомогательные исполняемые файлы EtwProcessor и ProcessCatcher. При копировании или перемещении основной утилиты они также должны быть скопированы/перемещены с сохранением относительности путей.

CppConfMonitor может быть запущен отдельно, и тогда необходимо его вручную остановить после завершения сборки, отправив «s» в консоль. Либо можно по аналогии с cppconf_linux.py указать команду сборки, тогда мониторинг будет остановлен автоматически.

Например:

```
CppConfMonitor.exe --directory C:\MyExe\ -e make
```

Если утилите не указывать папку проекта, то для анализа будут собраны только файлы, для которых были вызваны процессы компиляции. Таким образом, большинство template, inline-функций и включённых файлов исходных текстов не будут проанализированы сигнатурным анализатором.

Если указать папку проекта (флаг -p или --project), то все прочтённые файлы, имеющие расширения C/C++ файлов исходных текстов или заголовочных файлов и находящиеся внутри папки с проектом, будут добавлены для анализа.

Например, в данном случае условные C:\MyExe\1.h и C:\MyExe\2.h тоже будут проанализированы, хотя компилировался только 1.cpp:

```
CppConfMonitor.exe -p C:\MyExe\ --directory C:\MyExe\ -e make
```

В программе присутствует флаг --old_monitoring, который позволяет использовать другой механизм обнаружения процессов компиляции, но пользоваться им рекомендуется только в крайнем случае, так как он имеет небольшую вероятность пропуска или помехи выполнению некоторых из обнаруживаемых процессов. В первом случае будет выведена ошибка в консоли монитора, во втором случае прервётся процедура сборки. Особенно свойственно такое поведение для проектов с большим количеством быстрых процессов компиляции.

3.3.3.3. Ручная настройка

Для проведения конфигурации необходимо понимать структуру конфигурационного файла «АК-ВС 3» и знать о некоторых параметрах, передаваемых компиляторам.

Конфигурационный файл – это архив в формате ZIP, где:

- «parser-cpp» (без расширения) — файл с перечнем команд компиляции;

- «relations-cpp» (без расширения) – файл с описанием связей между исходными текстами и выполняемыми модулями, нужен только для МКЧА и АЧКПВ;
- заголовочные файлы.

Файл «parser-cpp» это файл в формате XML.

Корневой элемент -«config», дочерние элементы:

- «settings»— хранит общие настройки парсера в виде линейного списка из «param»;
- «compilerConfigs» — хранит опции парсера для компиляторов и стандартов языка C++ в виде линейного списка из «macro» и «idir»;
- «arguments» — хранит аргументы парсера для отдельных файлов в виде линейного списка из «fileArgs».

3.3.3.4. Общие настройки парсера

Каждая настройка хранится в теге «param», значение атрибута «name» задает название параметра, «value» - значение.

Пример:

```
<settings>
    <param name= “threadCount” value= “0” />
    <param name= “usePredefines” value= “0” />
</settings>
```

Перечень доступных параметров представлен в таблице 2.

Таблица 2 — Доступные параметры

Название	Описание	Возможные значения	Значение по умолчанию
threadCount	Количество потоков	-1 — программа сама вычисляет необходимое количество; 0 — не использовать потоки; n (n>0) — количество потоков.	-1
errorsLimit	Количество обрабатываемых	-1 — все ошибки; 0 — не допускать	5

Название	Описание	Возможные значения	Значение по умолчанию
	ошибок, при превышении - весь файл считается ошибочным	ошибок; n — допускать n ошибок(n>0).	
skipBuiltinErrors	Игнорировать ошибки, содержащие текст "use of undeclared identifier '__builtin_"	0 – нет; 1 – да.	1
useAutoTypeForReturn	Использование типа «auto» для временной переменной выражения под return	0 — не использовать (вычисляется возвращаемый тип функции); 1 — использовать.	0
usePredefines	Загрузка предустановленных макросов ОС и компилятора	0 — не загружать; 1 — загружать.	1
insertProbeInUnknown Encoding	Вставлять датчики в файлы, кодировку которых не удалось определить	0 — не вставлять 1 — вставлять	0
insertProbesIfError	Вставлять датчики в файлы, при анализе которых возникли ошибки разбора	0 — не вставлять 1 — вставлять, игнорируя ошибки заголовочных файлов 2 — вставлять	0

3.3.3.5. Опции парсера для компиляторов и стандартов языка C++

С помощью опций «macro» и «idir» можно задать макросы препроцессора и дополнительные пути заголовочных файлов

- macro:
  - compiler — путь до компилятора;
  - std — используемый стандарт, если не указать будет использован стандарт по умолчанию;
  - value — значение, формата “-DNAME=VALUE”.

Пример:

```
<macrocompiler="/usr/bin/c++" [std="c++11"] value="-DFOO=BAR" />
```

- idir:
  - compiler — путь до компилятора;
  - std — используемы стандарт, если не указать будет использован стандарт по умолчанию;
  - value — значение, формата “-I../prj/config/include/path”.

Пример:

```
<idircompiler="/usr/bin/c++" [std="c++11"]
value="-I../prj/config/usr/include" />
```

3.3.3.6. Аргументы парсера для отдельных файлов

Состоит из линейного списка fileArgs, представляет собой список компиляций исходного текста с одним путём.

Через аргументы можно задать include-path, define, язык, компилятор, порядок обработки ошибок и т.п. Список аргументов схож со списком аргументов clang и gcc.

Аргументы задаются с помощью тегов следующим образом:

- fileArgs:
  - path — путь к файлу с исходными текстами.

  Тело тега состоит из линейного списка “<args>”.
- args:
  - compiler — путь до компилятора;
  - std — используемый стандарт, если не указать, будет использован стандарт по умолчанию.

  Тело элемента состоит из линейного списка аргументов “<argument>”.
- argument:
  - value — аргумент компилятора

    Часто используются следующие:
    - -I — путь до заголовочного файла;
    - -D — макрос.

Пример:

```
<fileArgs path="home/user/foo.c">
    <args compiler="/usr/bin/c++">
        <argument value="-I../prj/config/usr/include/a" />
```

```
            <argument value="-DFOO=BAR" />
        </args>
        <args compiler="/usr/bin/c++" std="c++11">
            <argument value="-I../prj/config/usr/include/b" />
            <argument value="-DBAR=FOO" />
            <argument value="-ferror-limit=0"/>
        </args>
    </fileArgs>
```

Заголовочные файлы могут как принадлежать проекту, так и не принадлежать (например, системные заголовочные файлы, заголовочные файлы от сторонних библиотек).

Если заголовочный файл есть в архиве с исходными текстами и конфигурацией «src_config.zip», то достаточно просто указать путь до него — путь задается относительно корня архива «src_config.zip».

Если заголовочного файла нет в архиве с исходными текстами и конфигурацией «src_config.zip», то его надо добавить в «src_config.zip», например, в папку «sysinclude», а так же прописать путь с префиксом «../prj», т.е: «value=”-I../prj/sysinclude/”».

3.3.3.7. Синтетический пример

Допустим, есть директория с проектом «project» (рис. 11).

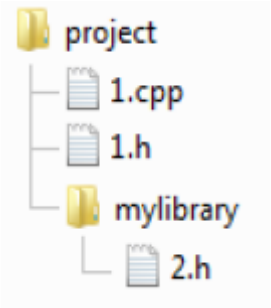

Рисунок 11 — Пример директории с проектом

Также существует директория с заголовочными файлами от сторонней библиотеки (рис. 12):

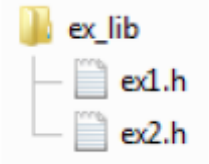

Рисунок 12 — Пример директории с заголовочным файлом

Файл «1.cpp» имеет вид, представленный на рис. 13.

```
1  #include "1.h"
2  #include "2.h"
3  #include "ex1.h"
4  #include "ex2.h"
```

Рисунок 13 — Пример библиотеки проекта

Соберем проект через командную строку с помощью команды:

```
g++ -I. -Imylibary -I/home/akvs/ex_lib/ -DSOMEDEFINE
```

В данном случае мы создаем архив «src_config.zip», содержащий два архива, «src.zip» и «prj.zip».

Архив «src.zip» с содержимым директории «project» выглядит следующим образом (рис. 14).

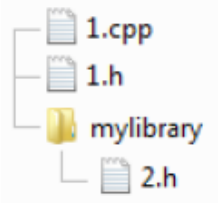

Рисунок 14 — Пример архив src.zip

Архив «prj.zip» с содержимым представлен на рис. 15.

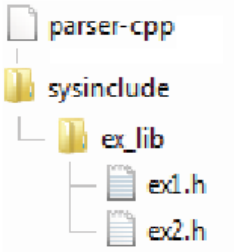

Рисунок 15— Пример архив prj.zip

Файл с настройкой компилятора «parser-cpp» будет иметь такой вид:

```
<?xml version="1.0"?>
<config>
<settings>
</settings>
<compilerConfigs>
    <macro compiler="g++" value="-ferror-limit=0" />
```

```
</compilerConfigs>
<arguments>
     <fileArgs path="1.cpp ">
          <args compiler="g++">
               <argument value="-DSOMEDEFINE" />
               <argument value="-I." />
               <argument value="-Imylibary" />
               <!—файлы ex_lib лежат не в src.zip, а в prj.zip, поэтому
          добавляем префикс ../prj/ -->
               <argument value="-I../prj/sysinclude/ex_lib" />
          </args>
     </fileArgs>
</arguments>
</config>
```

3.3.4. Конфигурация синтаксического анализатора C#

Для сигнатурного анализа ручная конфигурация не требуется – анализируются все файлы с расширением «.cs». Сигнатурный анализатор поддерживает C# до 5-ой версии включительно, анализатор МКЧА и АЧКПВ поддерживает .NET 5 и более ранние версии. Для анализа МКЧА и АЧКПВ необходимо выполнить сборку проекта и его упаковку с помощью инструмента akvs_sharp. Перед началом рекомендуется осуществить очистку прошлых результатов сборки, затем запустить сборку проекта командой, указанной ниже, где -d – директория проекта и -c – скрипт или команда для сборки.

```
akvs_sharp -d ./project_dir -c dotnet build
```

В случае успешной сборки проекта будет выполнена автоматическая упаковка директории проекта в архив «akvs_sharp.zip», который будет помещен в текущую рабочую директорию. Полученный архив нужно отправить на сервер для анализа.

При необходимости можно изменить конфигурацию парсера сигнатурного анализатора. Файл конфигурации парсера хранится в файле формата XML и имеет следующую структуру: корневой элемент - «config», дочерний элемент - «settings». «Settings» хранит параметры анализатора в виде линейного списка из «param». Каждая настройка хранится в теге «param»,

значение атрибута «name» задает название параметра, «value»-значение. Список параметров представлен в таблицеТаблица 3.

Таблица 3 — Список параметров

Параметр	Описание	Возможные значения	Значение по умолчанию
version	Версия стандарта C#	1.0, 2.0, 3.0, 4.0, 5.0	5.0
astCountInMemory	Количество хранимых AST в памяти. Можно увеличить это значение при достаточном объеме оперативной памяти	Целое число, отрицательное число обозначает хранить все AST в памяти	5000
maxFileSize	Максимальный размер файла для анализа (в байтах)	Целое число	5242880
warningDisable	Номера предупреждений, которые надо подавить с помощью директивы #pragma warning disable при вставке датчиков	Номера предупреждений через запятую	0162,0436

Пример конфигурационного файла:

```
<?xml version=”1.0”?>
<config>
<settings>
<param name=”version” value=”4.0” />
</settings>
</config>
```

3.3.5. Конфигурация синтаксического анализатора Java

Конфигурация для сигнатурного анализа и для анализа МКЧА и АЧКПВ делается отдельными механизмами. Для сигнатурного анализа конфигурирование не производится – анализируются все файлы с расширением .java. Для подготовки к анализу МКЧА и АЧКПВ необходимо провести контролируемую сборку с помощью инструмента akvs_jam, который устанавливается инсталлятором для сборочного стенда.

Нужно осуществить очистку результатов сборки, а затем – полную сборку с помощью команды:

```
akvs_jam capture -- mvn compile
```

В результате в директории, где производилась сборка, возникает поддиректория jam-engine-out. Для отправки на сервер анализа директорию проекта вместе с jam-engine-out нужно самостоятельно запаковать в архив 7z с помощью команды 7z.

Есть возможность использовать упрощённую команду с автоматическим созданием архива.

```
akvs_jam -c mvn compile
```

Так же можно указать папку проекта, которая будет использоваться для сборки и в результате будет упакована.

3.4. Статический анализ и динамический анализ

Главной страницей веб-интерфейса является вкладка «Проекты», которая содержит список проектов с базовой информацией о статусе каждого проекта (рис. 16).

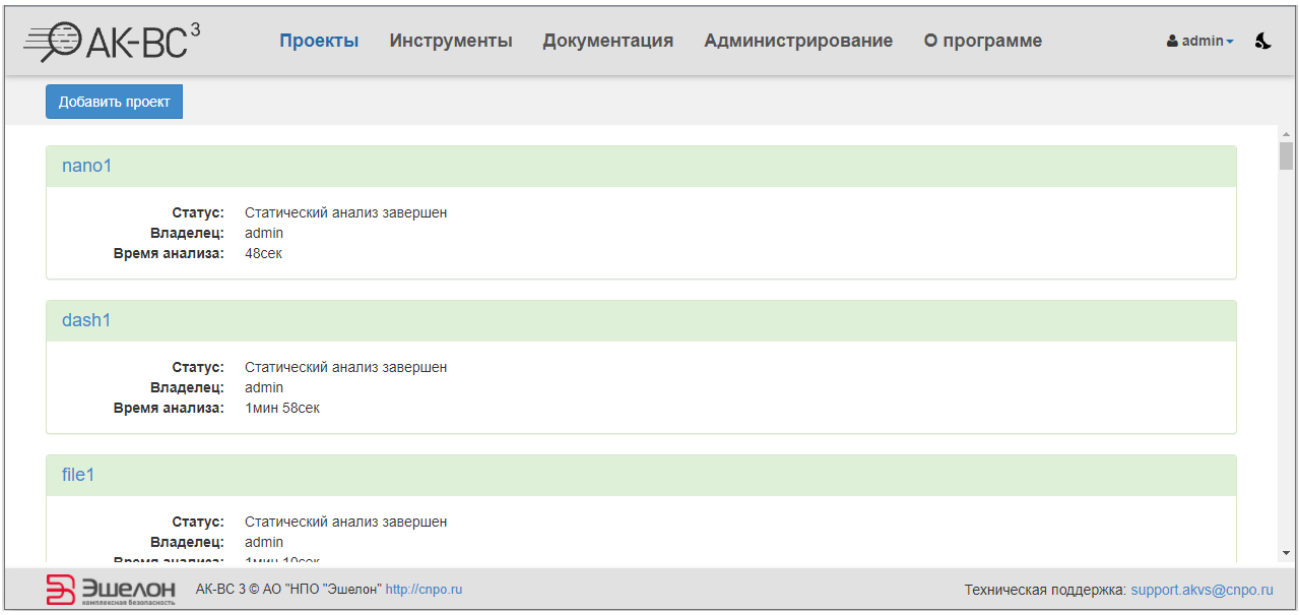

Рисунок 16 — Проекты

Веб-интерфейс анализатора также содержит вкладку «Инструменты», которая содержит ссылки для загрузки датчиков для динамического анализа, вспомогательных скриптов, утилит и инструкций (рис.17).

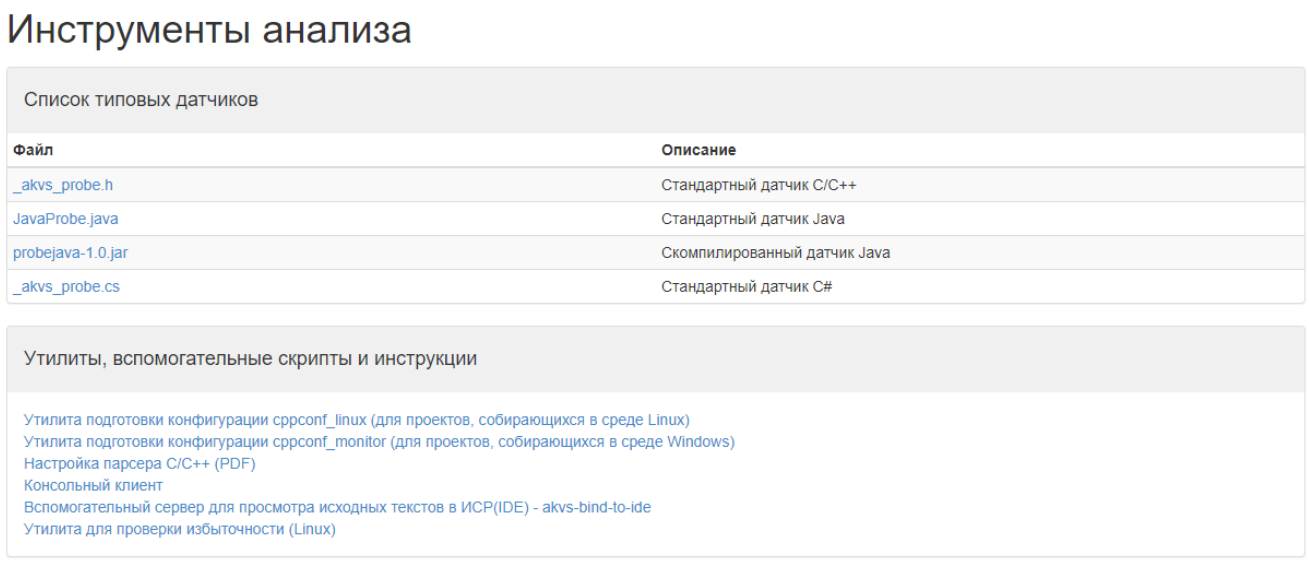

Рисунок17 — Вкладка «Инструменты»

Вкладка «Администрирование» содержит инструменты для администрирования «АК-ВС 3» (рис. 18). Подробнее см. п. 3.4.5..

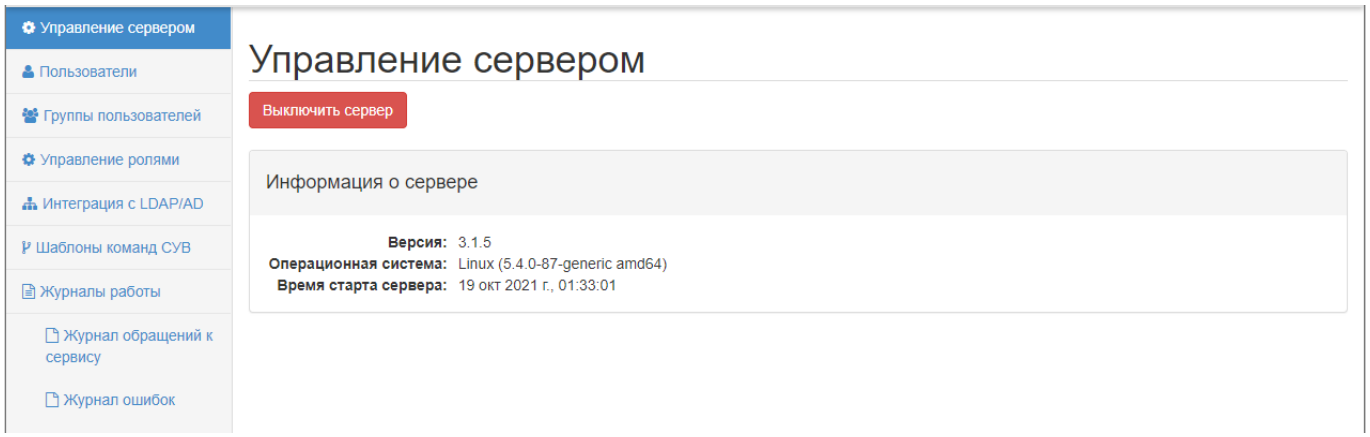

Рисунок 18 — Вкладка «Администрирование»

Вкладка «О программе» содержит информацию о лицензии, версии программы, контактные данные технической поддержки, а также справочную информацию.

Примечание. Инструкция по обновлению лицензии представлена в приложении Приложение 2.

3.4.1. Создание проекта

Для создания нового проекта на вкладке «Проекты» нажмите кнопку «Добавить проект» (рис. 19).

В появившемся окне нужно ввести название проекта и метод загрузки. Если выбран метод загрузки «git», то также нужно указать адрес репозитория и ветку. Метод загрузки из СУВ не осуществляет конфигурацию проектов, поэтому его использование допустимо только для анализа Java и C# без применения метода МКЧА. Название проекта может содержать только латинские буквы, цифры, дефис, нижнее подчеркивание и точку и не должно начинаться с точки. После этого нажмите кнопку «Добавить».

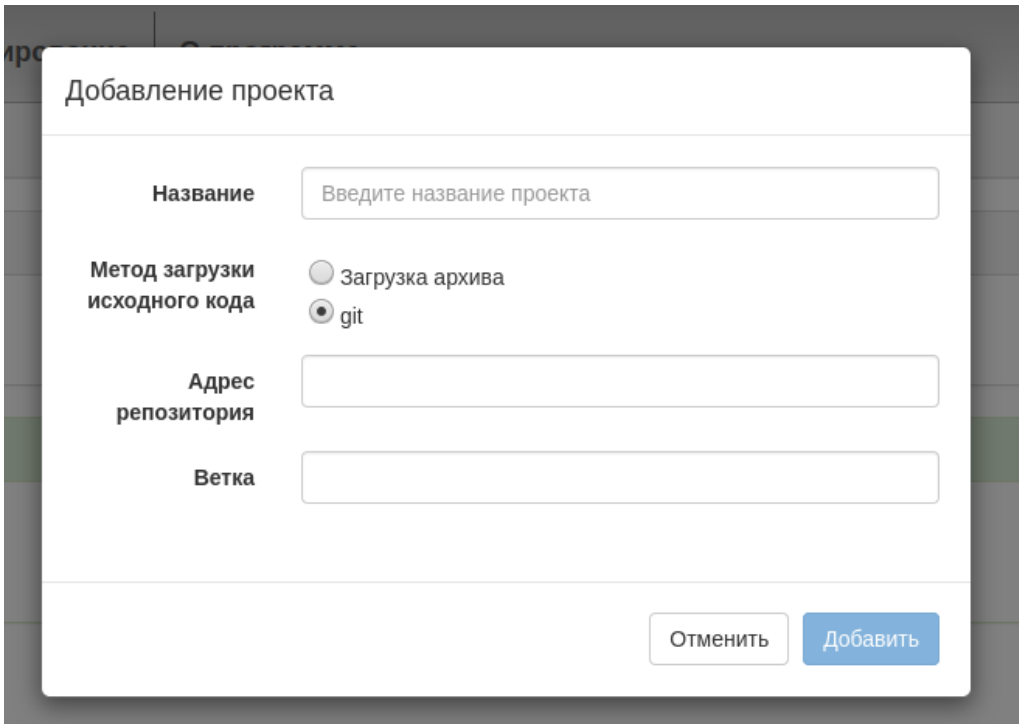

Рисунок 19 — Добавление проекта

Страница созданного проекта показана на рисунке 20. При необходимости можно переименовать или удалить проект. Для этого нажмите соответствующую кнопку «Переименовать проект» или «Удалить проект» (рис. 20).

Примечание. ПРИ УДАЛЕНИИ ПРОЕКТА УДАЛЯЮТСЯ ВСЕ ДАННЫЕ ПРОЕКТА БЕЗ ВОЗМОЖНОСТИ ВОССТАНОВЛЕНИЯ!

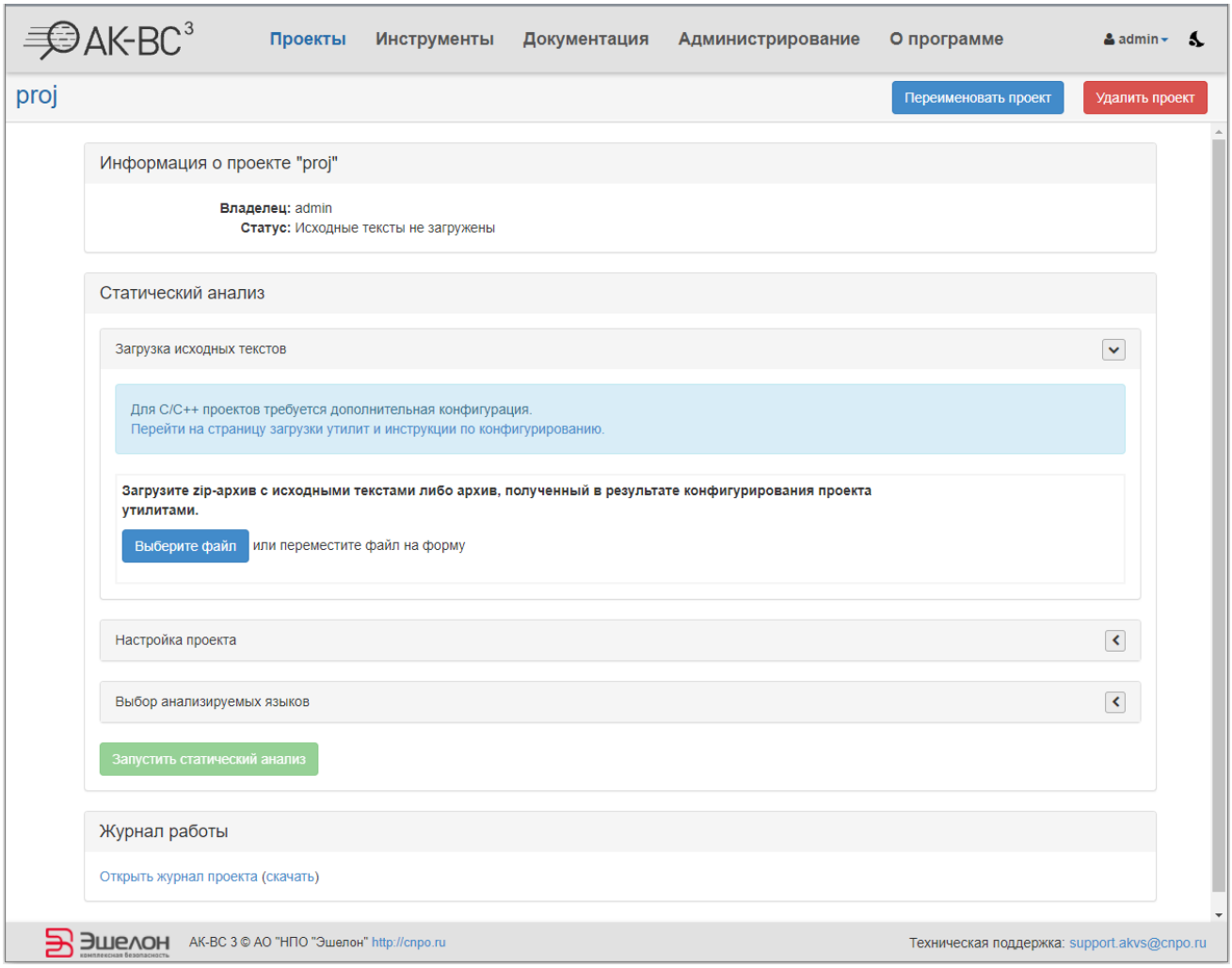

Рисунок 20 — Страница проекта

3.4.2. Запуск статического анализа

Загрузите исходные тексты программы в архиве формата ZIP или 7z (при наличии файлов с кириллицей предпочтителен формат 7z), подготовленный согласно п. 3.3.1..

Для того, чтобы загрузить архив с исходными кодами, нажмите на вкладку «Загрузка исходных текстов». Затем в открывшемся подменю нажмите кнопку «Выберите файл» или переместите архив на форму.

Для настройки проекта необходимо нажать на вкладку «Настройка проекта», при этом развернётся группа элементов управления (рис. 21). В поле «Уровень контроля» необходимо выбрать уровень контроля в процессе анализа. Для уровней контроля 3 и 4 по умолчанию МКЧА и АЧКПВ отключены, но их можно включить галочкой. В поле «Типы анализа» необходимо выбрать типы проводимого анализа. При анализе проектов на C/C++ можно расширить набор выполняемых проверок за счёт замедления анализа и повышения вероятности возникновения

ошибок при работе анализатора. Для этого нужно установить галочку «Включить экспериментальные проверки». Для того, чтобы исходный код был вставлен в блок-схемы, необходимо поставить галочку напротив поля «Вставлять исходный код в блок-схемы» (данная функция используется при уровне контроля 2 и выше).

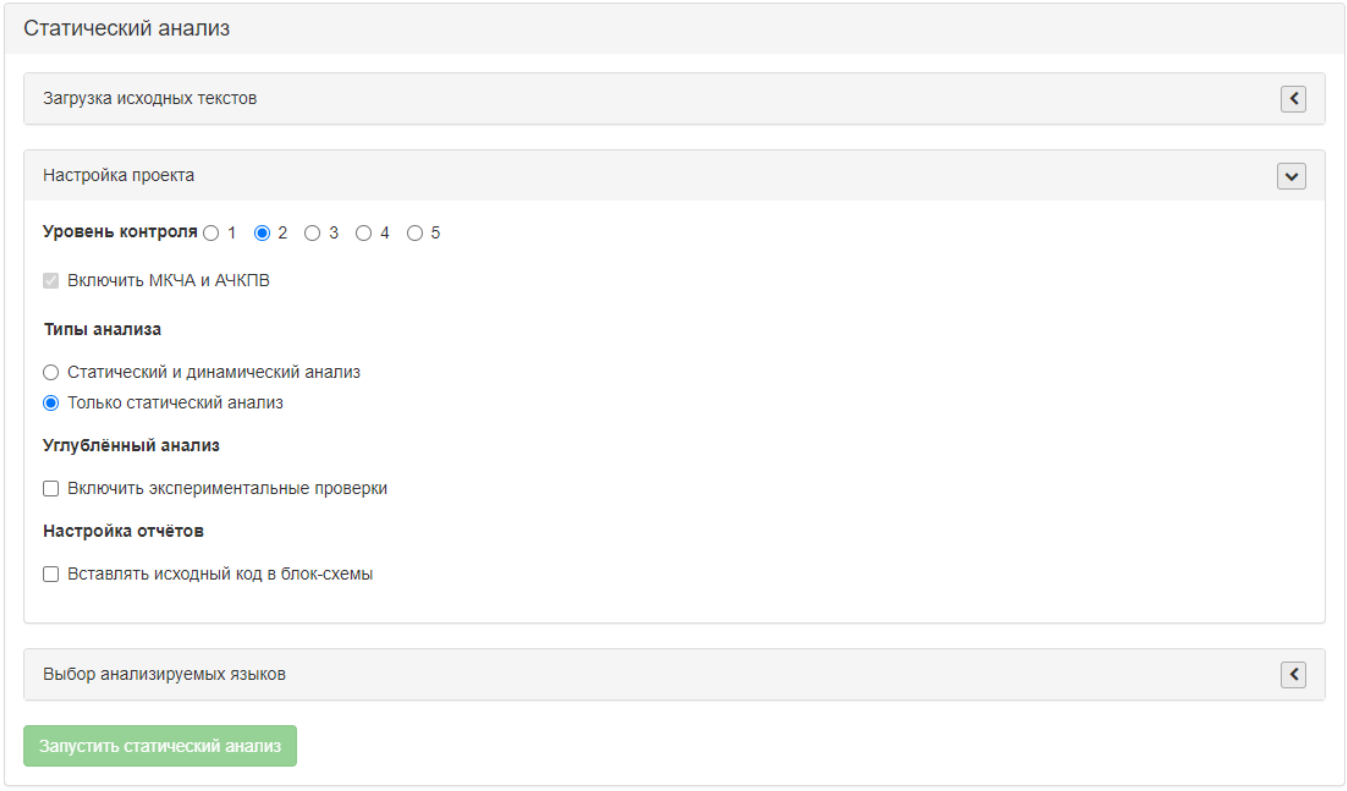

Рисунок 21 — Настройка проекта стат. анализатора

Для выбора анализируемых языков необходимо нажать на вкладку «Выбор анализируемых языков» и поставить галочку напротив нужного языка. По умолчанию, все поддерживаемые языки включены для проведения анализа. Для запуска анализа необходимо нажать кнопку «Запустить статический анализ». После этого статус проекта изменится на «Выполнение статического анализа». По завершению статического анализа статус проекта изменится на «Статический анализ завершен» (это можно видеть в списке проектов) и откроется страница отчётов (рис. 22).

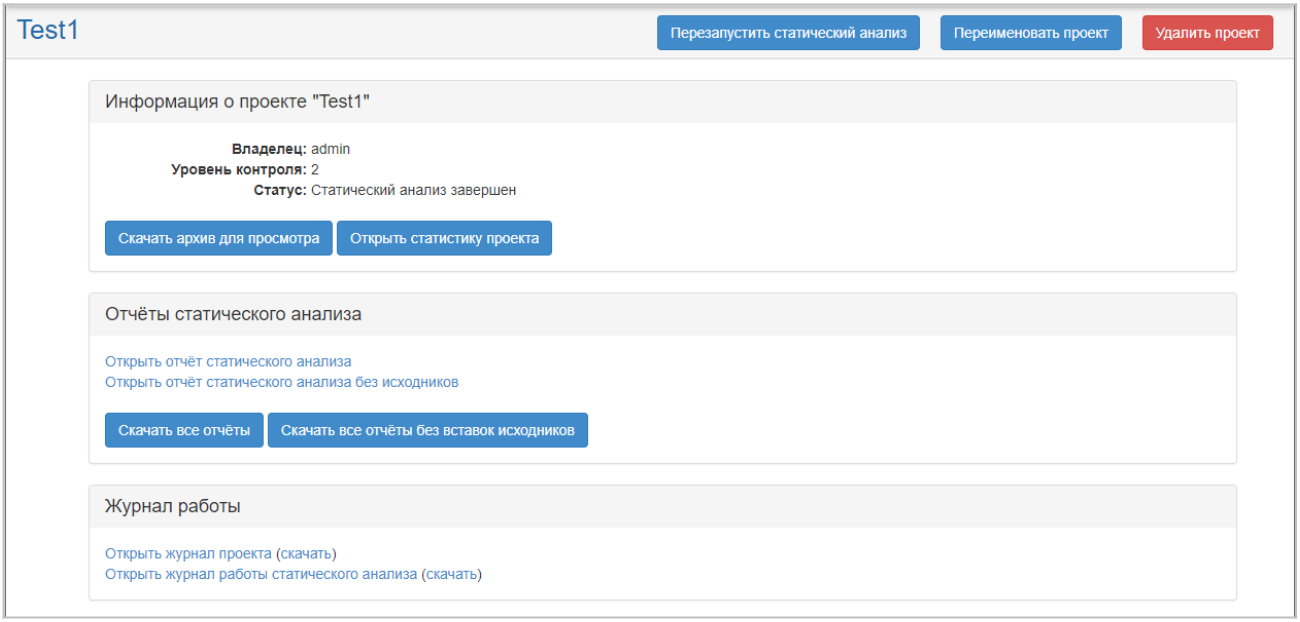

Рисунок 22 — Статический анализ завершен

3.4.2.1. Просмотр журнала статического анализа

Процесс анализа несовершенен и рекомендуется всегда смотреть журнал работы статического анализатора на предмет наличия в нём ошибок.

3.4.2.2. Просмотр отчётов статического анализа

Отчёт статического анализа может быть сделан с включением фрагментов исходников или без включения. Есть два способа просмотра отчётов (рис. 22).

- через веб-браузер, отчёты выдаются с сервера «АК-ВС 3» – две гиперссылки «Открыть отчёт статического анализа…» (рис. 24).

- через веб-браузер, отчёты скачиваются на локальный компьютер – кнопки «Скачать все отчёты…»

В любом случае, для просмотра ссылок на исходные тексты, нужно установить на машине, где производится просмотр, дистрибутив клиентской части «АК-ВС 3» и скачать архив для просмотра исходников данного анализа, кнопка «Скачать архив для просмотра». В этом архиве находятся все исходные тексты, которые были проанализированы, а также вспомогательные скрипты. Скачанный архив для просмотра нужно раскрыть, и в поддиректории src выполнить команду:

```
sh run.sh
```

После этого гиперссылки на места в исходном тексте будут открываться в ИСР Эшелониум. Если же скрипт не запущен, то при попытке перейти по ссылке появится сообщение (рис. 23).

Чтобы не было необходимости запускать run.sh вручную, можно сразу просматривать результаты анализа через Эшелониум.

Рисунок 23 — не запущен akvs-bind-to-ide

Список отчётов по статическому анализу проекта "Project"

- Отчёт по метрикам
- Список файлов проекта
- Список информационных объектов
- Перечень функциональных объектов (функций и процедур)
- Перечень функциональных объектов (ветвей)
- Перечень базовых блоков
- Перечень невызываемых ФО
- Перечень неопределенных ФО
- Таблица связей ФО по управлению
- Маршруты выполнения ФО
- Таблица связей ФО по информации
- Критические маршруты для выбранного ИО
- Таблица связей функций и ветвей
- Маршруты выполнения ФО с ветвями
- Таблица связей базовых блоков
- Блок-схемы ФО
- Отчет о выявленных вставках кода
- Отчёт о выявленных программных ошибках

Рисунок 24 — Список отчетов по статическому анализу

На рис. 25 представлен отчет по метрикам статического анализа.

Project: отчёт по метрикам статического анализа	
Список кодировок в проекте	UTF-8
Список языков в проекте	C/C++
Общее количество файлов	37
Суммарный размер файлов	581 KB (595 065 байт)
Средний размер файла	16 KB (16 082 байт)
Количество строк кода	17 891
Среднее количество строк в файле	483
Количество информационных объектов	2 550
Количество функциональных объектов	427
Количество ветвей	2 172
Количество невызываемых функциональных объектов	8
Количество неопределенных функциональных объектов	141
Количество связей по управлению	1 415
Количество связей по информации	576
Количество связей функций и ветвей	2 995

Рисунок 25 — Отчет по метрикам статического анализа

На рис. 26 представлен список файлов проекта.

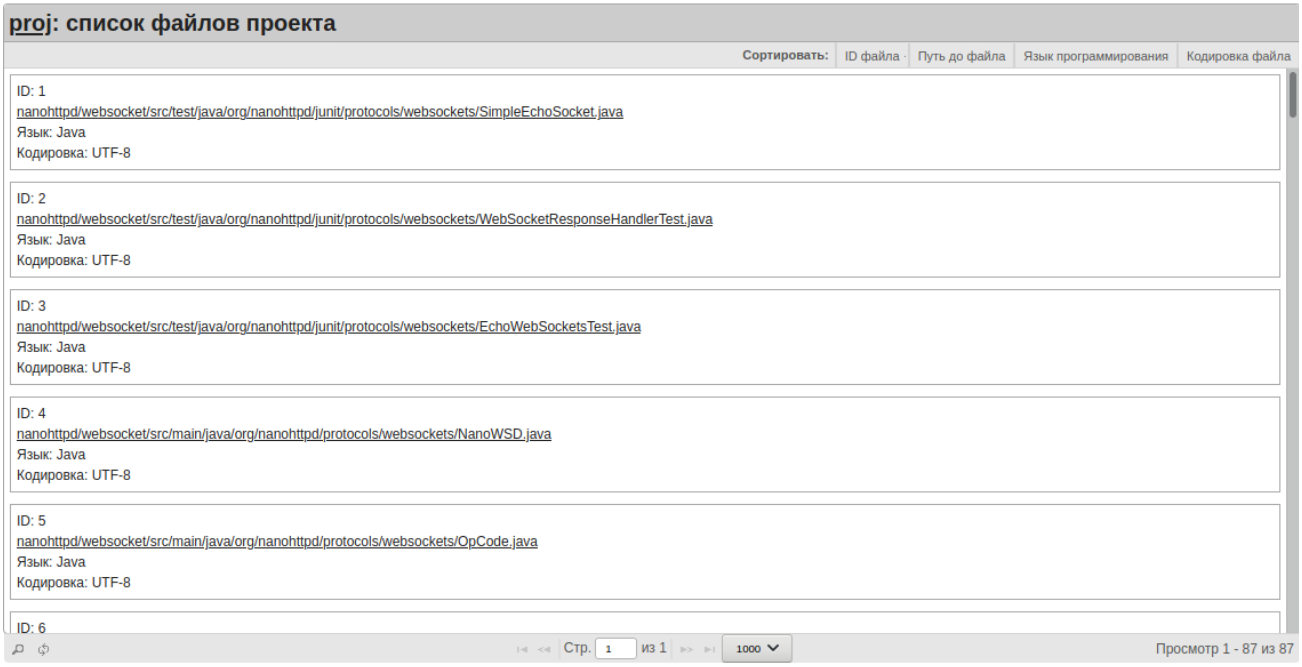

Рисунок 26 — Список файлов проекта

Каждая запись в отчёте изображается рамкой. Одиночный щелчок мышью окрашивает ячейку (тем самым можно запоминать текущую позицию при просмотре больших списков) В верхней части отчёта имеются кнопки для сортировки по полям отчёта. В нижней – кнопки для фильтрации, обновления списка, номер страницы и настройки пагинации. Окно фильтра можно видеть на рис. 27.

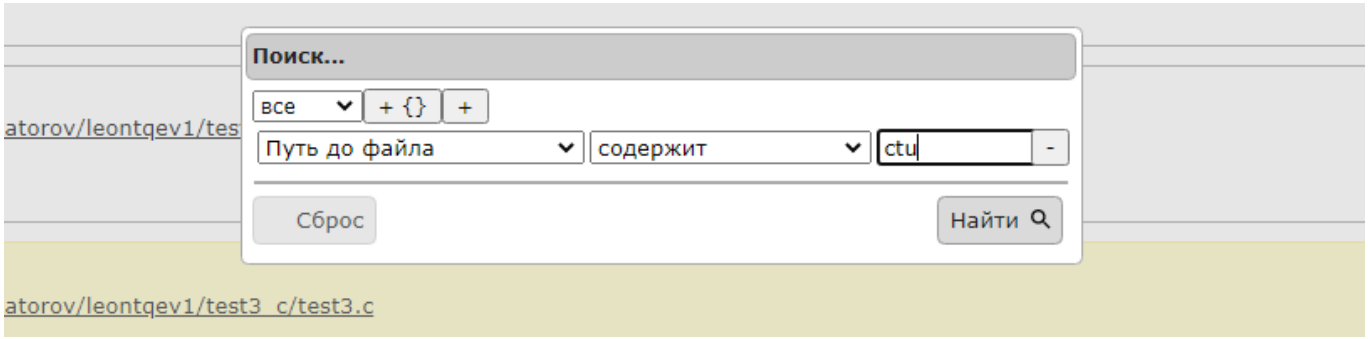

Рисунок 27 — Окно определения условия фильтрации результатов

На рис. 28 представлены маршруты выполнения ФО. Для просмотра маршрутов в интерактивном режиме необходимо нажать на соответствующую ссылку. После этого откроется страница, изображенная на рис. 29. Для того, чтобы продолжить маршрут вперед или назад, необходимо нажать соответствующую кнопку. Полученные маршруты можно сохранить в формате SVG (рис. 30).

Рисунок 28 — Отчет о маршрутах выполнения ФО

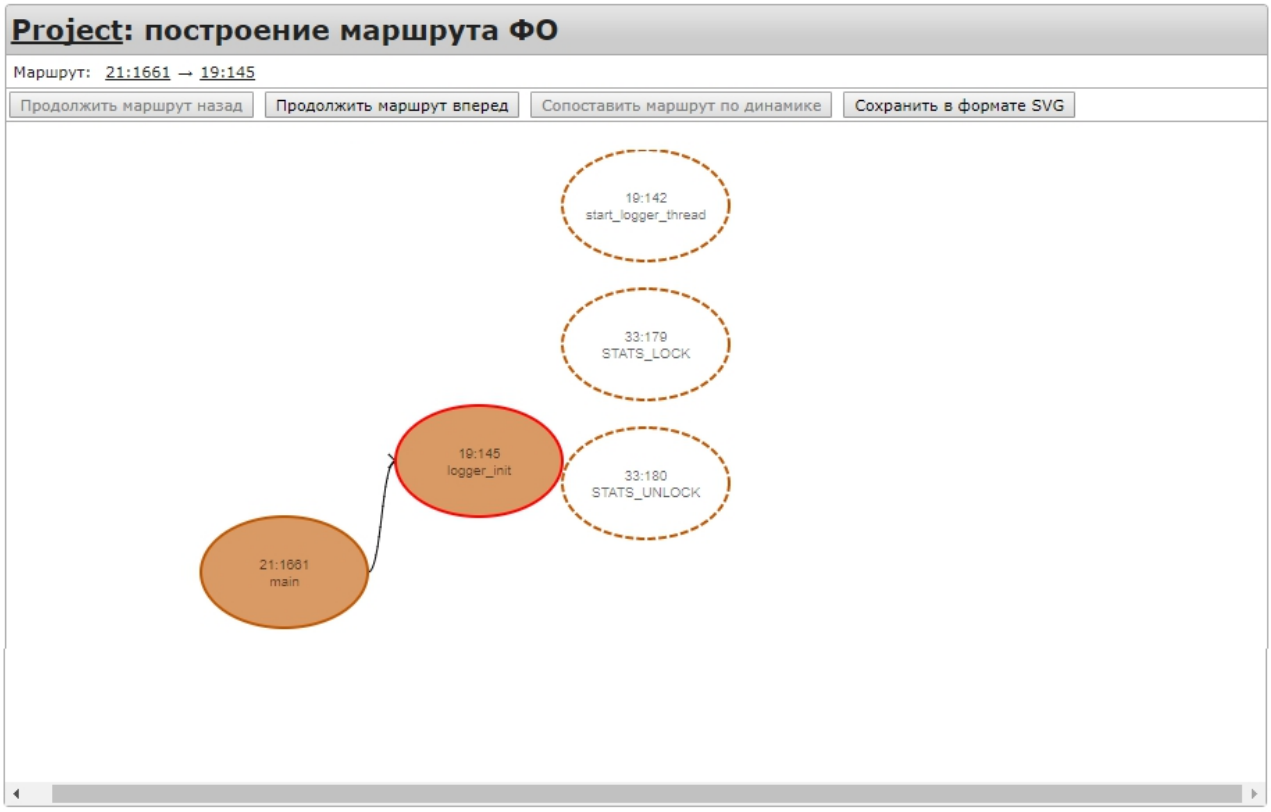

Рисунок 29 — Пример интерактивного построения маршрута ФО

Рисунок 30 — Пример построенного маршрута ФО

Чтобы просмотреть все маршруты от точки входа до выбранного ФО необходимо нажать на соответствующую ссылку. После этого откроется страница, показанная на рис. 31. Маршруты можно сохранить в формате SVG.

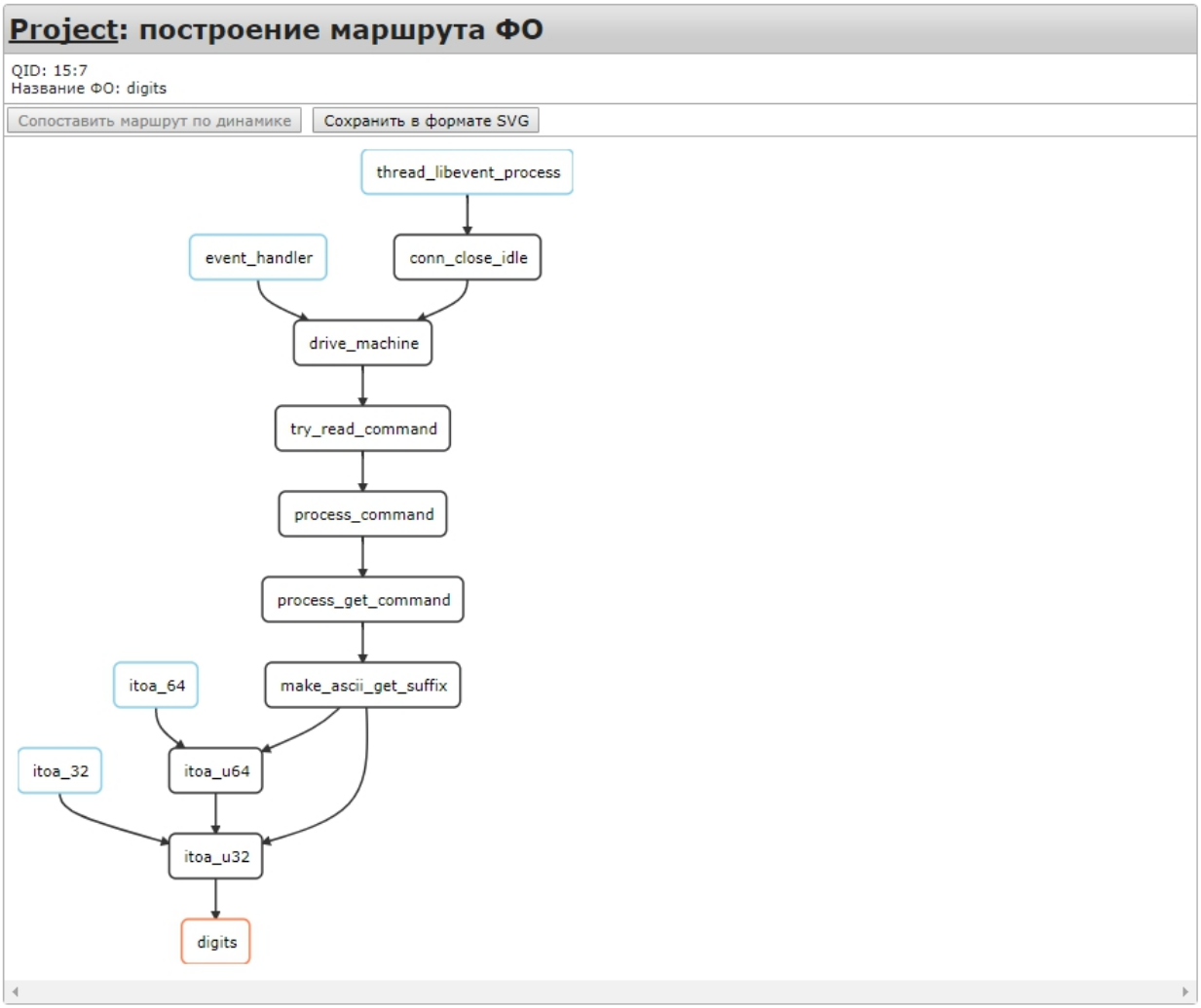

Рисунок 31 — Построение маршрута ФО

На рис. 32 представлена таблица связей ФО по информации.

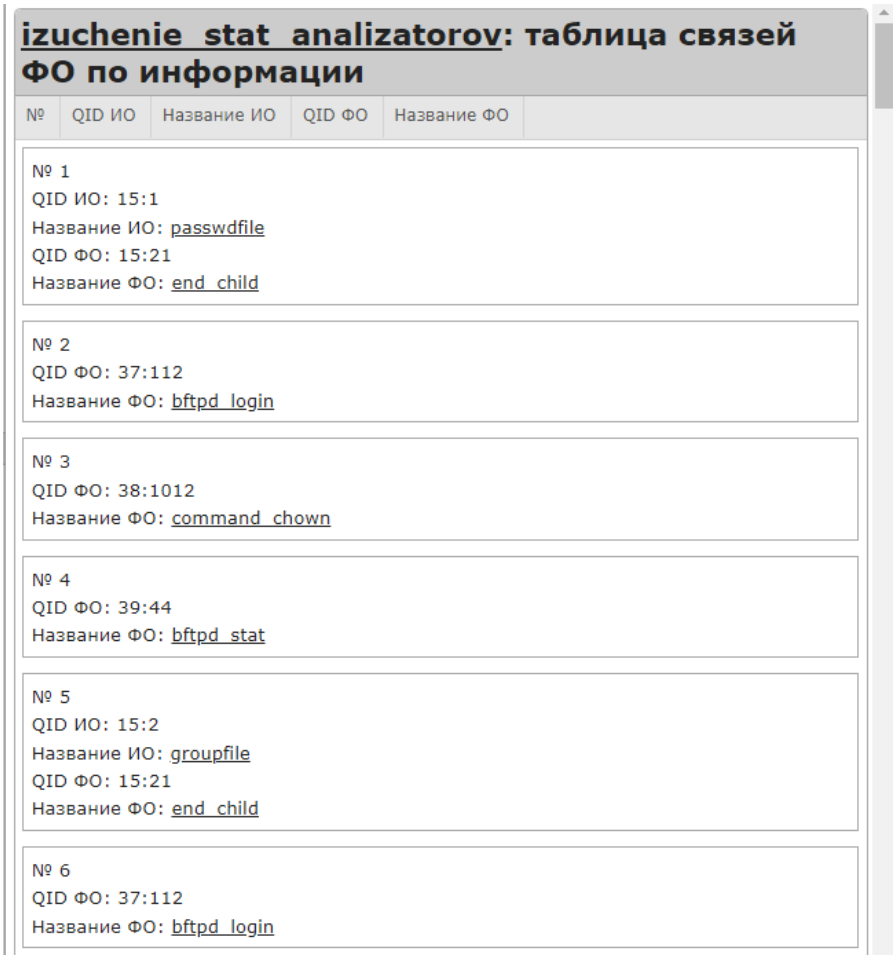

Рисунок 32 — Таблица связей ФО по информации

На рис. 33 представлена таблица ИО. Для просмотра критических маршрутов необходимо выбрать один из ИО, представленных в таблице. После этого откроется страница, показанная на рис. 34. Для просмотра маршрута необходимо нажать «Проложить маршрут».

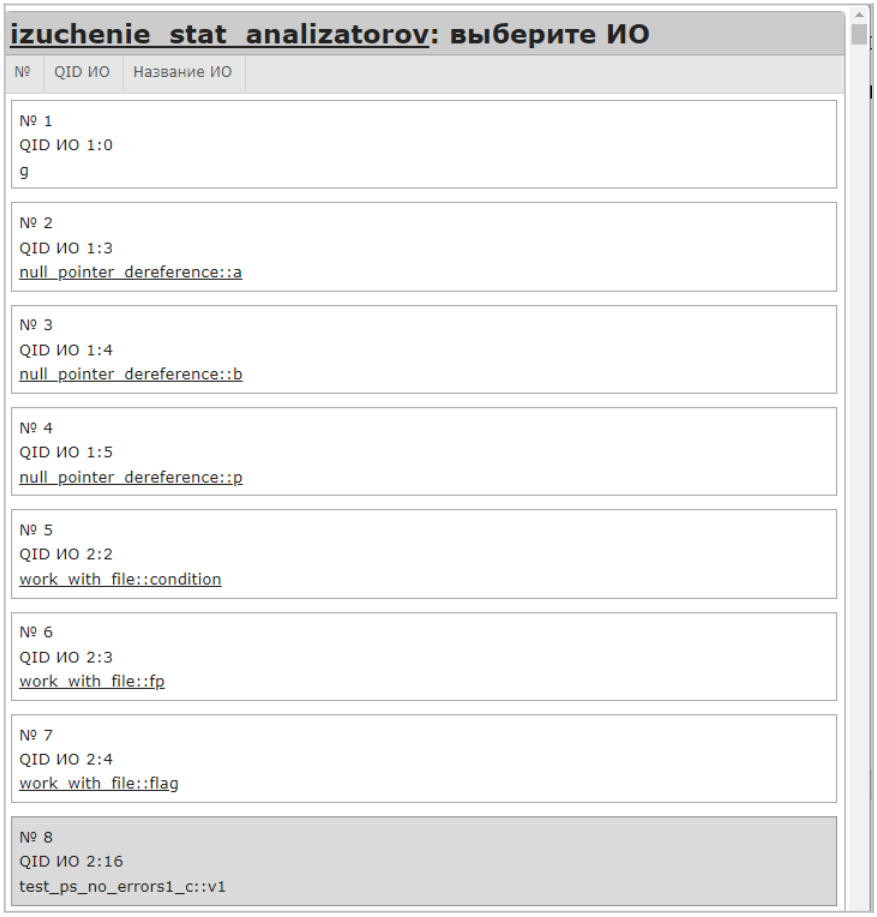

Рисунок 33 — Выбор ИО для построения критического маршрута

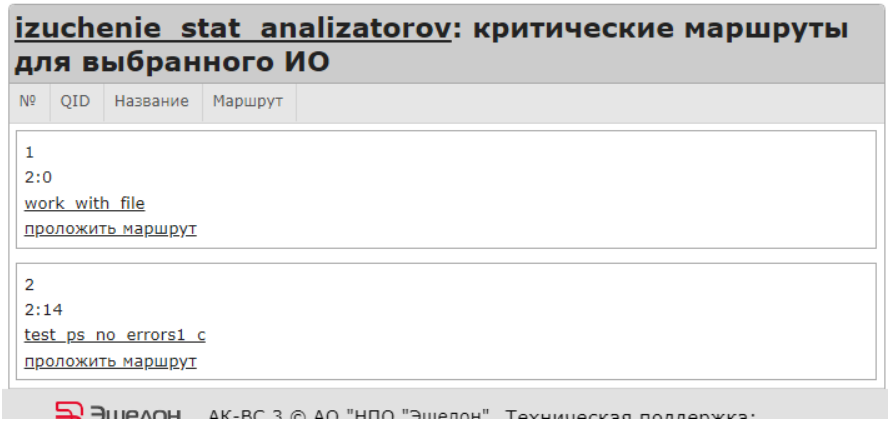

Рисунок 34 — Критические маршруты для выбранного ИО

На рис. 35 представлена таблица связей функций и ветвей.

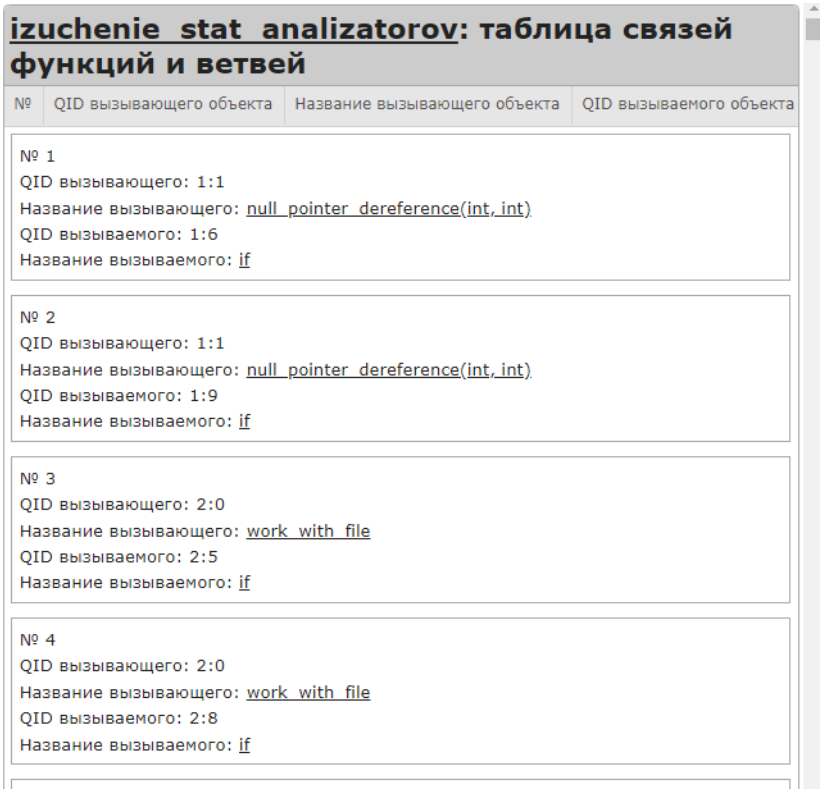

Рисунок 35 — Таблица связей функций и ветвей

На рис. 36 представлены маршруты выполнения ФО с ветвями. Для просмотра интересующих маршрутов необходимо нажать на соответствующую ссылку (аналогично построению критических маршрутов).

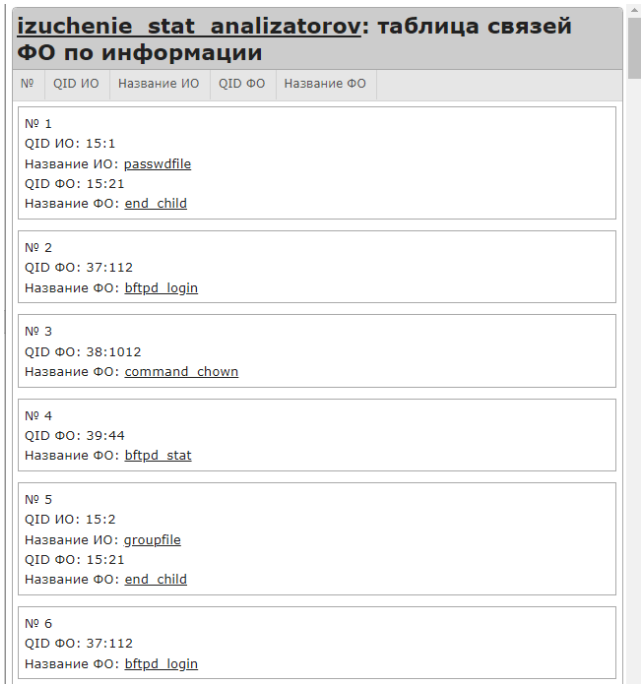

Рисунок 36 — Таблица исходных ФО для построения маршрута

На рис. 37 представлена таблица для построения блок-схемы ФО. В отчете имеются поля, аналогичные представленным в маршрутах выполнения ФО. Для просмотра блок-схемы необходимо нажать на один из представленных ФО. После этого откроется страница, показанная на рис. 38. При наведении курсора мыши на строку исходного кода будет выделен узел блок-схемы, соответствующий этой строке. Аналогично при наведении курсора мыши на узел блок-схемы будет выделена строка исходного кода, соответствующая этому узлу. Представленную блок-схему можно сохранить в формате SVG.

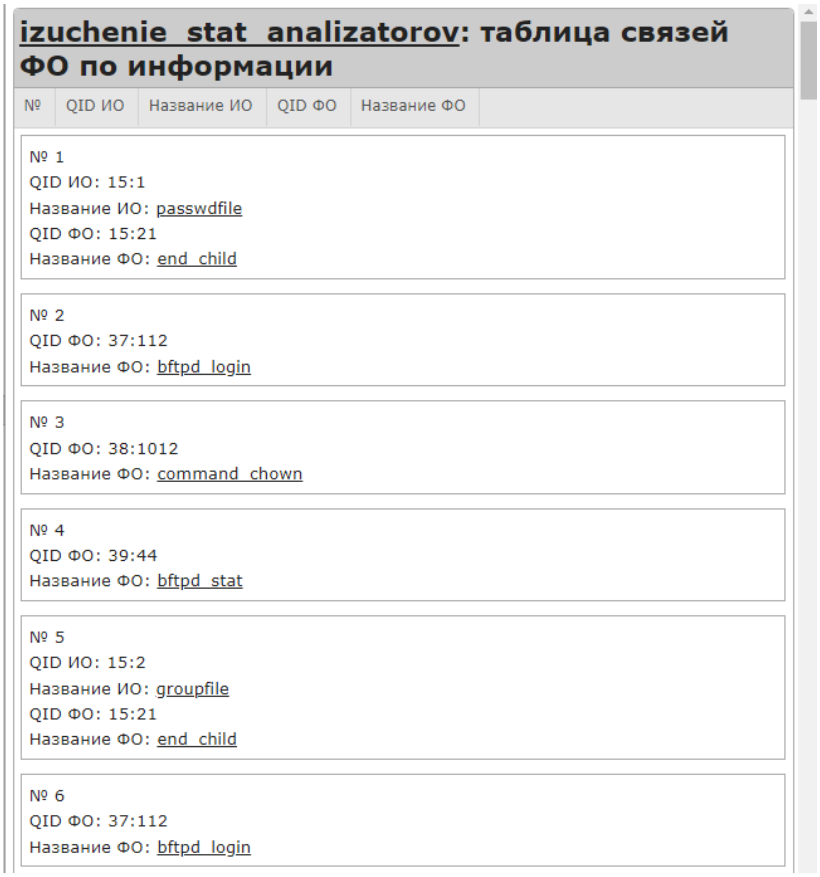

Рисунок 37 — Таблица выбора ФО для построения блок-схемы

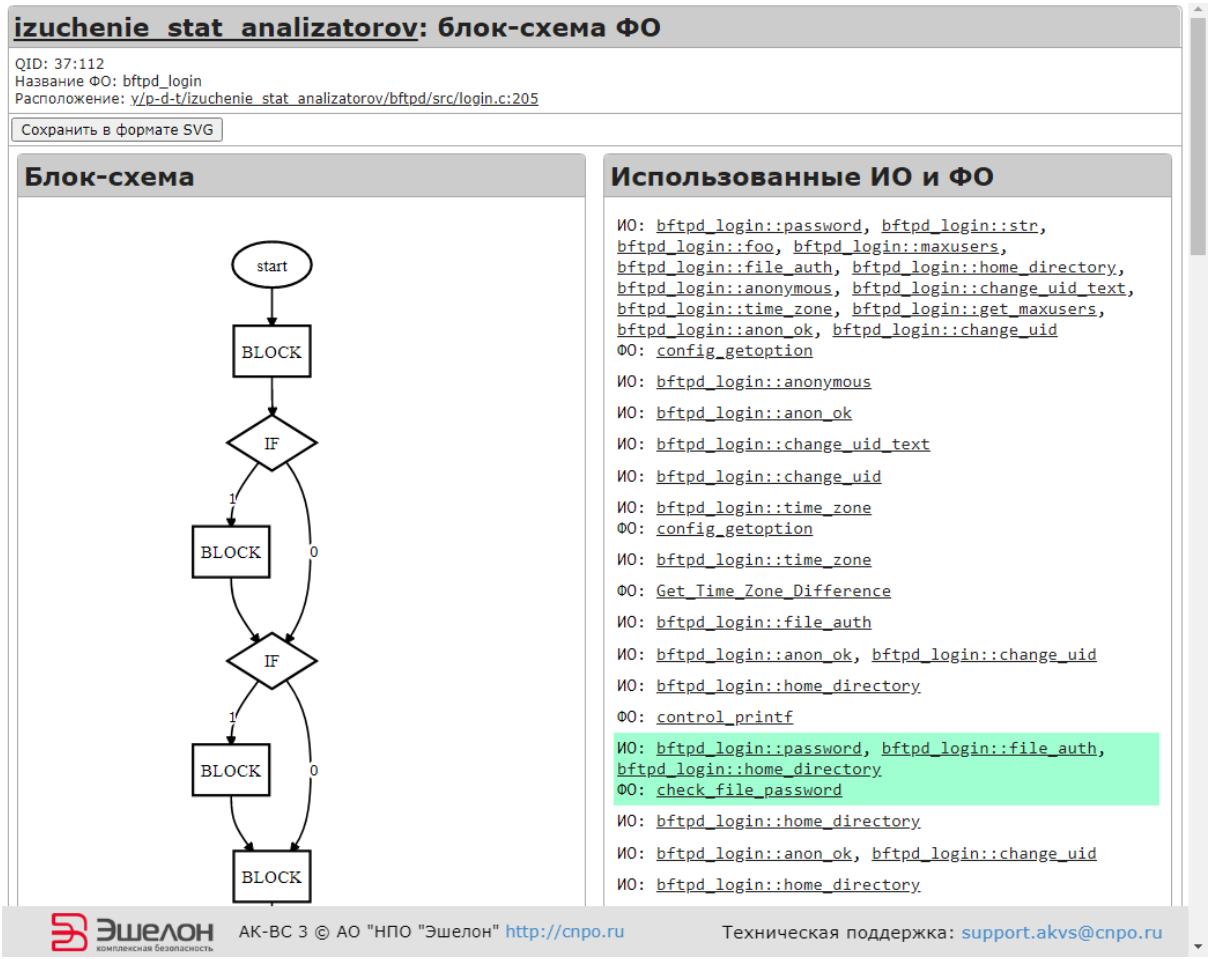

Рисунок 38 — Пример блок-схемы для выбранного ФО

На рис. 39 представлен отчёт о выявленных вставках кода показывает вставки кода на языках, не поддерживаемых статическим анализатором. В случае C/C++ это – язык ассемблера.

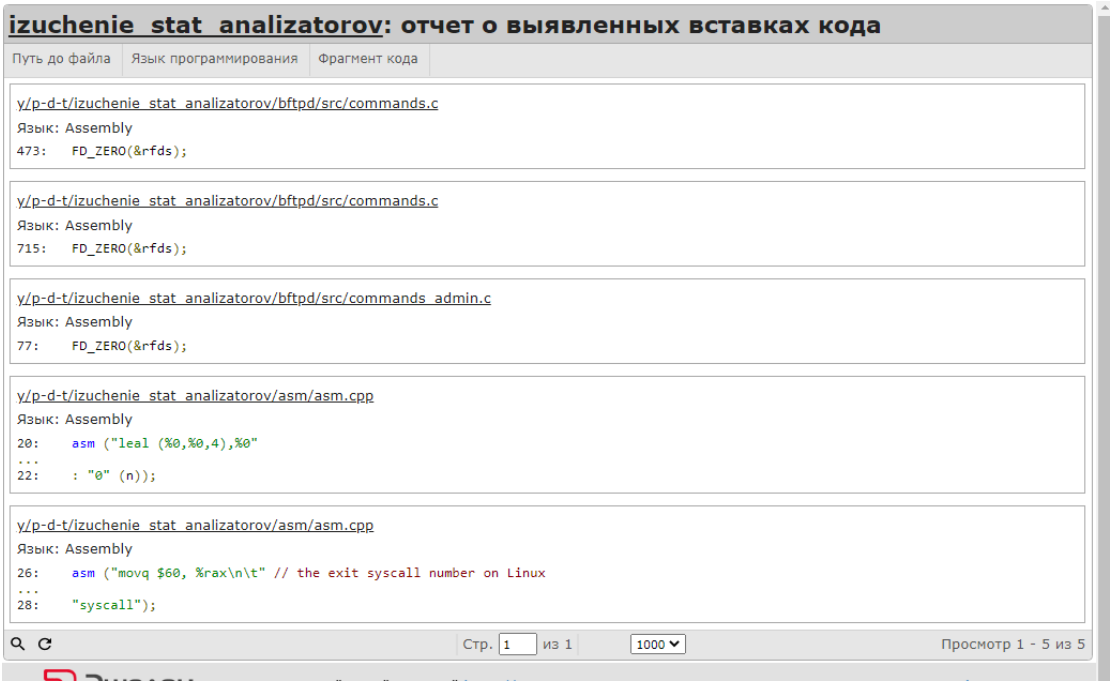

Рисунок 39 — Отчёт о выявленных вставках кода

На рис. 40 представлен отчет о выявленных программных ошибках.

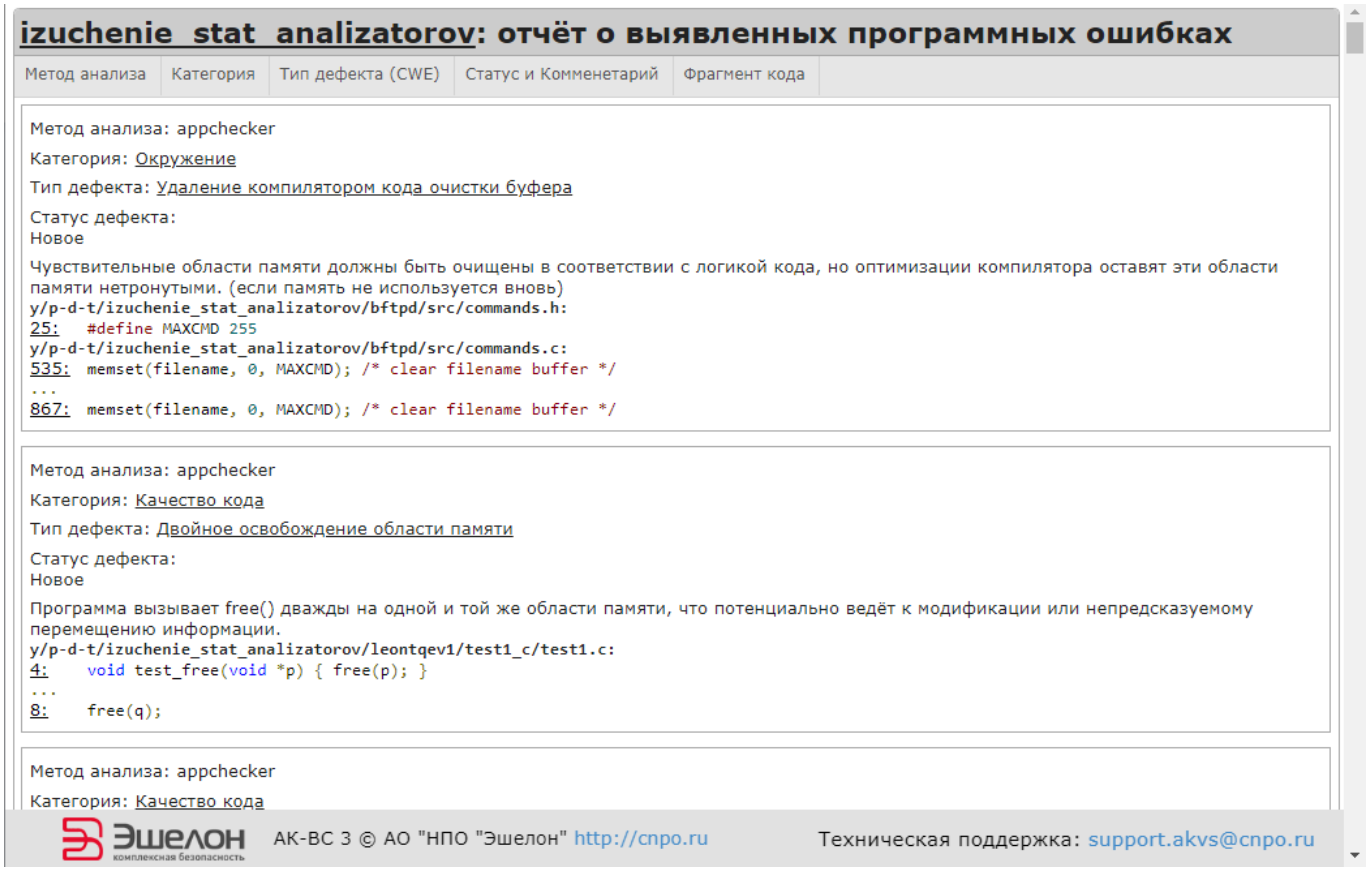

Рисунок 40 — Отчет о выявленных программных ошибках

3.4.2.3. Журнал работы статического анализа

Для просмотра журнала статического анализа (рис. 41) необходимо на странице проекта (рис. Рисунок 22) в разделе «Журнал работы» нажать кнопку «Открыть журнал работы статического анализа».

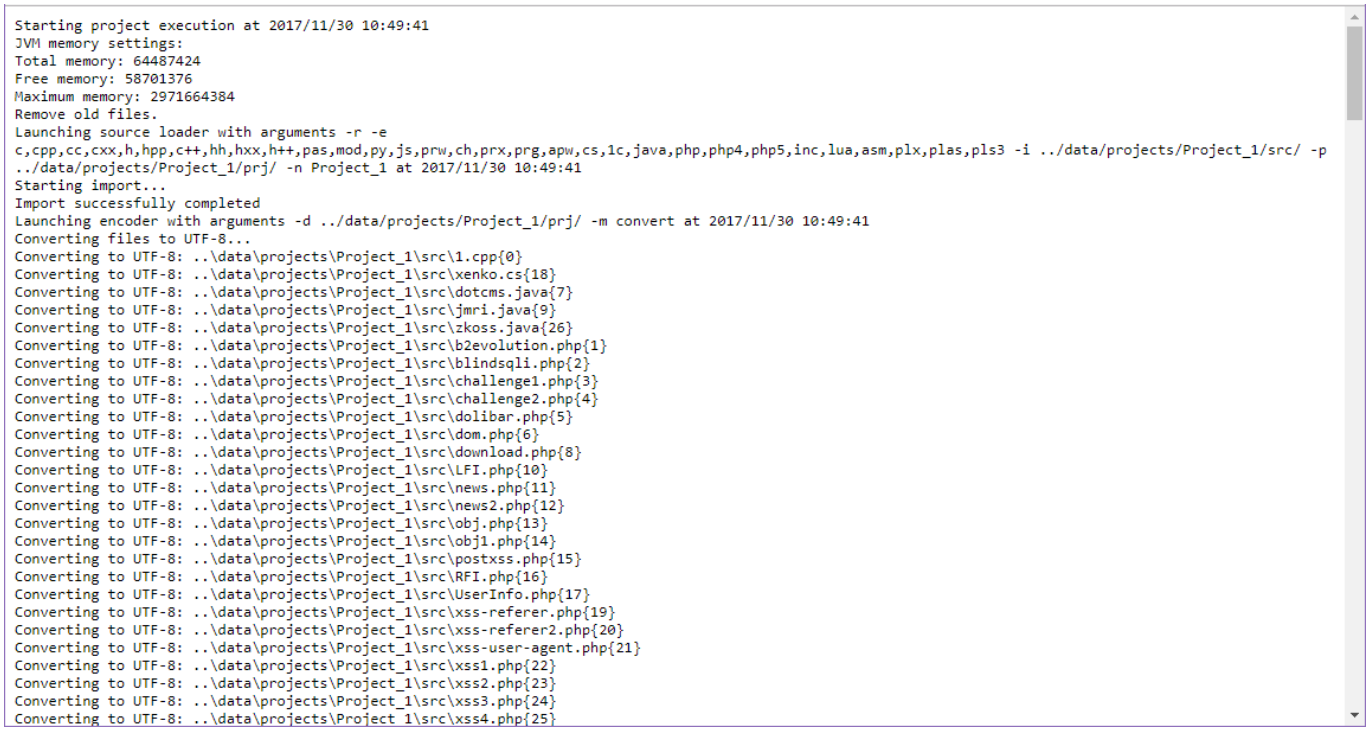

Рисунок 41 — Журнал статического анализа

3.4.3. Динамический анализ по РД НДВ

«АК-ВС 3» может инструментировать исходные тексты программ, подвергшихся статическому анализу, вставляя в них датчики. Далее можно собрать модифицированные исходные тексты и выполнить программу, собирая сведения о том, какие из датчиков отработали. Для этого необходимо на странице проекта в разделе «Динамический анализ» нажать кнопку «Скачать исходные тексты с датчиками». После этого необходимо собрать приложение с внедренными датчиками, используя включенные в поставку «АК-ВС 3» исходные файлы датчиков для соответствующего языка. Необходимые датчики можно скачать во вкладке «Инструменты» (рис. 17).

Ответственность за полноту тестирования несет эксперт, проводящий тестирование. По завершении тестирования в директории «АК-ВС 3» будет находиться один или несколько файлов лога отработки датчиков (с расширением «.log»). Допустимо проводить тестирование в несколько этапов, сохраняя получившиеся файлы лога.

3.4.3.1. Внедрение датчиков в проект на C/C++

Чтобы собрать проект C/C++:

- скачайте датчик «_akvs_probe.h»;
- скопируйте его в «includepath» проекта.

Примечание. Для unix-систем можно разместить его в «/usr/include», чтобы не копировать датчик в каждый подпроект.

Датчик для C/C++ состоит из двух частей:

- декларация датчика;
- функция датчика (определение функции).

Декларация датчика должна быть в каждом исходном файле, а определение - только в одном из набора файлов, объектные коды которых передаются линковщику.

«АК-ВС 3» вставляет в начало каждого файла конструкцию:

```
#include "_akvs_probe.h"
```

В те файлы, где есть функция «main/WinMain», конструкцию:

```
#define _AKVS_PROBE_IMPLEMENTATION_
#include "_akvs_probe.h"
```

Датчик поддерживает многопоточные приложения. Отключите режим многопоточности, если он не используется в программе, удалив (или закомментировав) из файла «_akvs_probe.h» строчку:

```
#define _AKVS_PROBE_MULTITHREAD
```

3.4.3.2. Внедрение датчиков в проект на C#

Расположение датчика – файл «_akvs_probe.cs».

Чтобы собрать проект C#:

- скачайте датчик «_akvs_probe.cs»
- добавьте его в проект.

3.4.3.3. Внедрение датчиков в проект на Java

Датчик для Java предоставляется в двух видах:

- в исходном «JavaProbe.java»;
- в скомпилированном «probejava-1.0.jar».

Расположение датчика –пакет «ru.cnpo.akvs2.probejava».

Чтобы собрать проект Java с помощью «JavaProbe.java»:

- создайте путь «ru/cnpo/akvs2/probejava» в директории с исходными файлами;
- скачайте датчик «JavaProbe.java»;
- скопируйте датчик «JavaProbe.java» в директорию.

Чтобы собрать проект Java, использующий систему сборки maven:

- скачайте датчик «probejava-1.0.jar»;
- выполните команду:

```
mvn install:install-file -Dfile=probejava-1.0.jar -DgroupId=ru.cnpo.ak
vs2 -DartifactId=probejava -Dversion=1.0 -Dpackaging=jar
```

- добавьте зависимость (dependency) в файл «pom.xml»:

```
groupId — ru.cnpo.akvs2
artifactId — probejava
version — 1.0
```

Пример представлен ниже.

```
<project>
  ...
  <dependencies>
    ...
    <dependency>
      <groupId>ru.cnpo.akvs2</groupId>
      <artifactId>probejava</artifactId>
      <version>1.0</version>
    </dependency>
    ...
  </dependencies>
  ...
</project>
```

3.4.3.4. Загрузка файлов

Далее необходимо запаковать файл(ы) лога в архив в формате ZIP (файлы лога должны находиться в корневой директории архива) и загрузить этот архив для обработки проекта, нажав кнопку «Выберите файл» на главной странице проекта (рис. 42).

Примечание. Все логи в ОС семейства Linux можно архивировать с помощью команды: `find . -name "akvs*.log" -print | zipdyn_log.zip -@.`

При необходимости поставьте галочку напротив «Догрузить к существующим файлам», чтобы не были удалены логи предыдущих запусков.

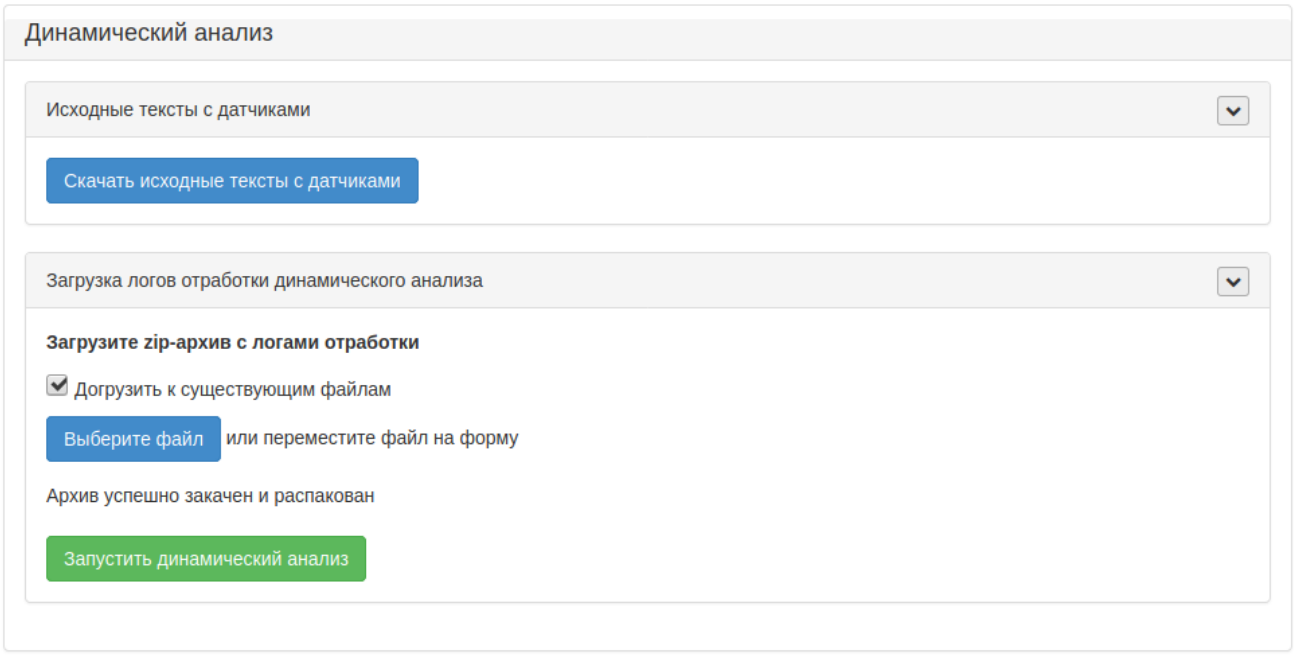

Рисунок 42 — Динамический анализ

Далее необходимо нажать кнопку «Запустить динамический анализ» (рис. 42). Динамический анализ будет запущен. После этого статус проекта изменится на «Обработка результатов динамического анализа». По завершению динамического анализа статус проекта изменится на «Динамический анализ завершен» (рис. 43).

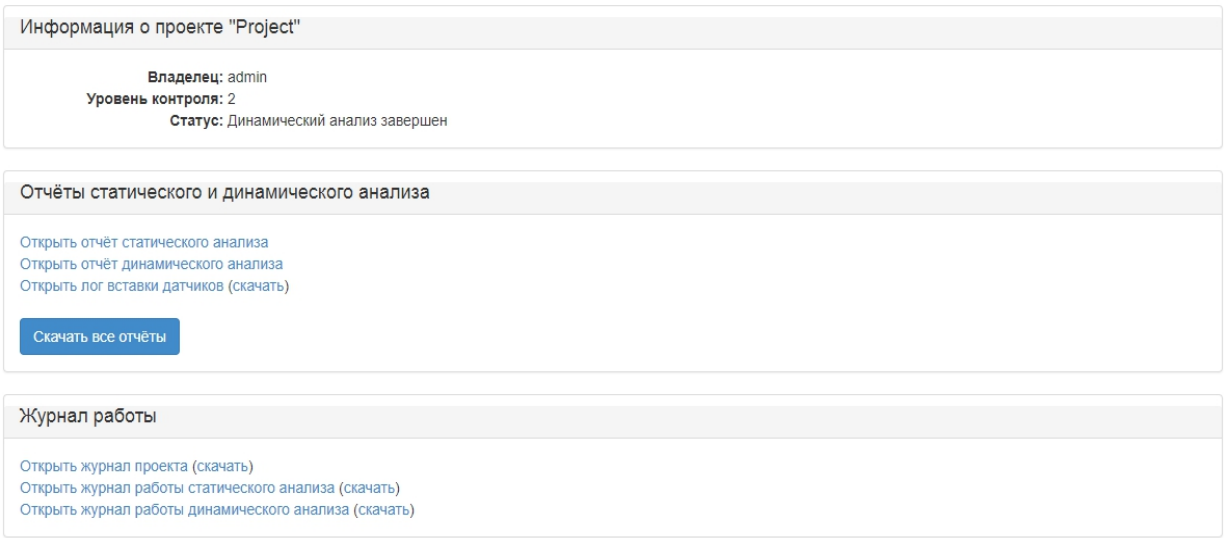

Рисунок 43 — Динамический анализ завершен

3.4.3.5. Отчеты динамического анализа

Для просмотра отчетов по динамическому анализу необходимо на вкладке «Проекты» навести курсор на интересующий проект и нажать левую кнопку мыши. Далее в разделе «Отчеты статического и динамического анализа» (рис. 43) необходимо выбрать «Открыть отчет динамического анализа». После этого откроется страница с отчетами по динамическому анализу (рис. 44).

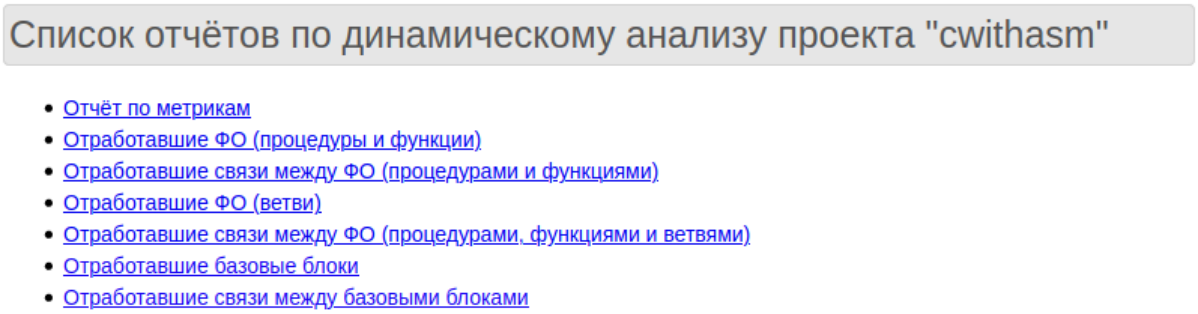

Рисунок 44 — Список отчетов по динамическому анализу

Для просмотра интересующего отчета необходимо навести на него курсор и нажать левую кнопку мыши.

На рис. 45 представлен отчет по метрикам динамического анализа.

Project: отчёт по метрикам динамического анализа	
Количество подтвердившихся вызовов ФО/общее количество ФО	257/427
Процент покрытия по ФО	61%
Количество подтвердившихся связей ФО/общее количество связей ФО	958/1415
Процент покрытия по связям ФО	68%
Количество подтвердившихся вызовов ветвей/общее количество ветвей	645/2172
Процент покрытия по ветвям	30%
Количество подтвердившихся связей ФО с ветвями/общее количество связей ФО с ветвями	3051/2995
Процент покрытия по связям ФО с ветвями	100%

Рисунок 45 — Отчет по метрикам динамического анализа

На рис. 46 представлены отработавшие ФО (процедуры и функции). Красным цветом выделены ФО, не отработавшие в ходе динамического анализа, зеленым — отработавшие.

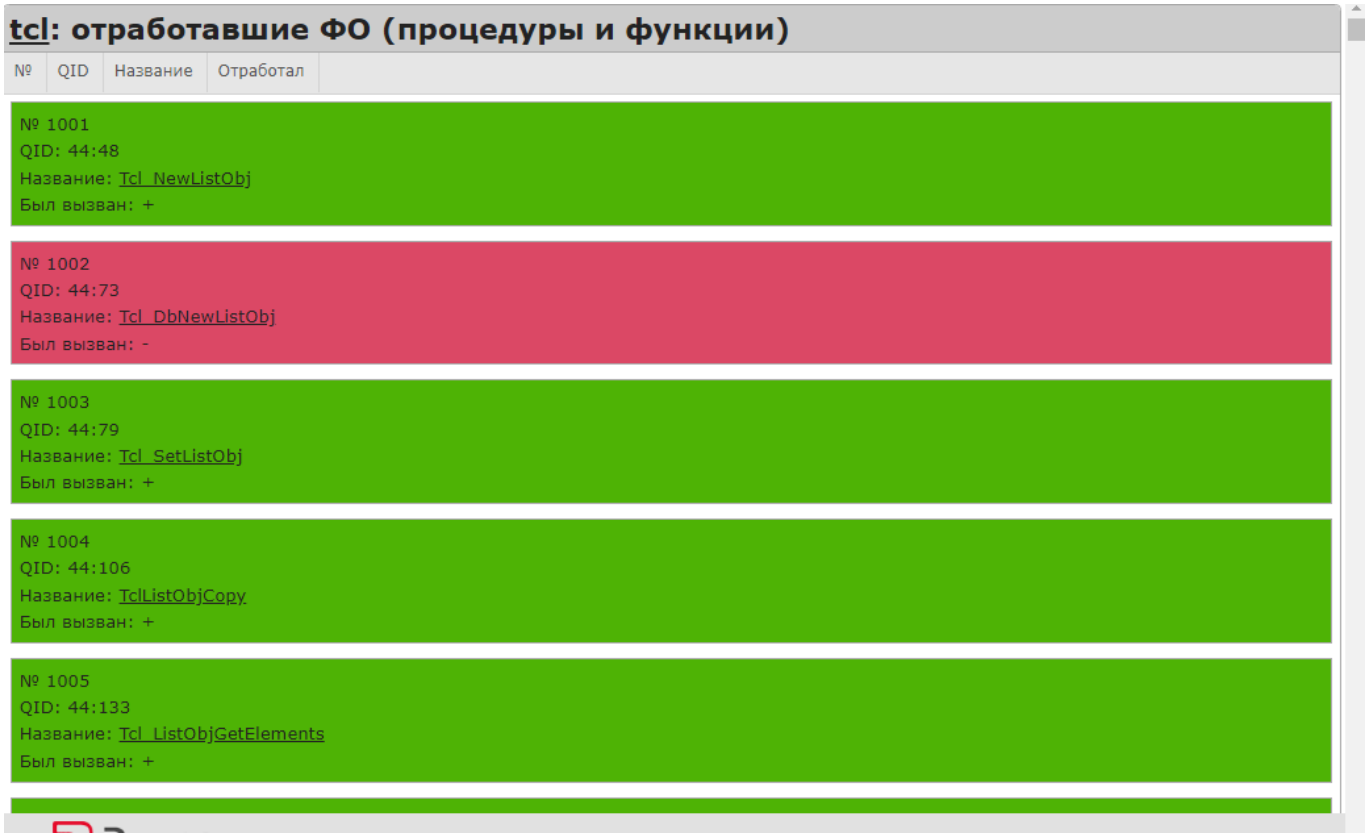

Рисунок 46 — Список отработавших ФО (процедуры и функции)

Остальные отчёты выглядят аналогично. При необходимости можно скачать файлы отчетов. Для этого нажмите кнопку «Скачать все отчеты» (рис. 43). Отчеты будут скачаны в архиве формата ZIP.

3.4.3.6. Журнал работы динамического анализа

Для просмотра журнала динамического анализа (рис. 47) необходимо на странице проекта в разделе «Журнал работы» нажать кнопку «Открыть журнал динамического анализа» (рис. 43).

```
Starting project execution at 2017/12/05 10:33:22
JVM memory settings:
Total memory: 64487424
Free memory: 58740248
Maximum memory: 2971664384
Launching dynamic report with arguments ../data/projects/Project_1/prj/ -l 2 at 2017/12/05 10:33:22
Error while pasring log file C:\AKVS2\backend\..\data\projects\Project_1\prj\dlogs\zkoss.java
Error while pasring log file C:\AKVS2\backend\..\data\projects\Project_1\prj\dlogs\zkoss.java
Error while pasring log file C:\AKVS2\backend\..\data\projects\Project_1\prj\dlogs\zkoss.java
Error while pasring log file C:\AKVS2\backend\..\data\projects\Project_1\prj\dlogs\zkoss.java
Project completed successfully at 2017/12/05 10:33:23
```

Рисунок 47 — Журнал динамического анализа

3.4.4. Перезапуск статического и динамического анализов

Чтобы перезапустить статический или динамический анализы нажмите кнопку «Перезапустить анализ» на главной странице проекта.

3.4.5. Администрирование

На вкладке «Администрирование» содержатся следующие разделы: «Управление сервером», «Пользователи», «Группы пользователей», «Управление ролями», «Интеграция с LDAP/AD», «Шаблоны команд СУВ», «Журналы работы» (рис. 48).

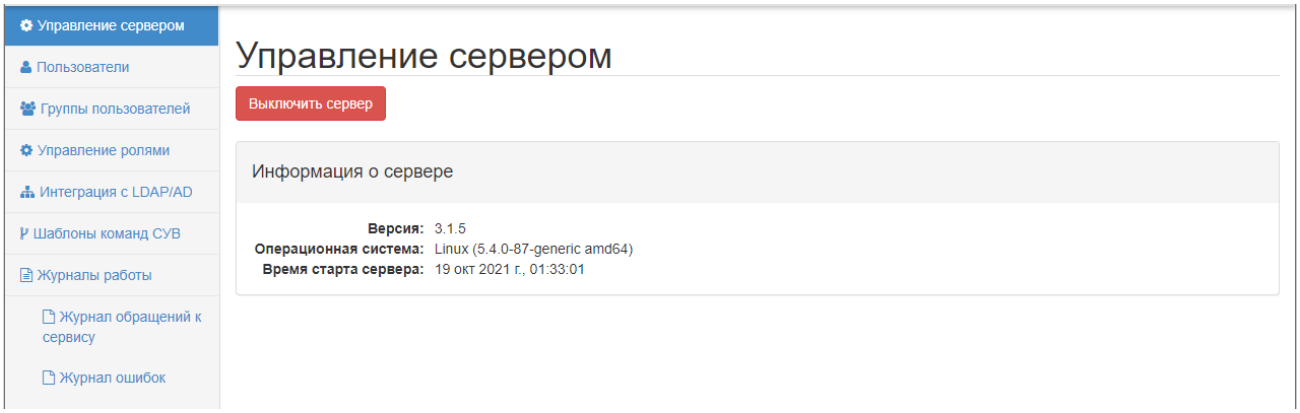

Рисунок 48 — Вкладка «Администрирование»

3.4.5.1. Управление сервером

Раздел «Управление сервером» содержит общую информацию о сервере. Для выключения сервера необходимо нажать кнопку «Выключить сервер» (рис. 48).

3.4.5.2. Пользователи, группы и роли

«АК-ВС 3» поддерживает создание пользователей и объединение пользователей в группы. Для добавления нового пользователя в разделе «Пользователи» необходимо нажать кнопку «Добавить пользователя» (рис. 49). В открывшемся окне необходимо ввести имя пользователя и пароль (рис. 50). Для удаления пользователя необходимо нажать кнопку «Удалить» напротив пользователя (рис. 49). При необходимо можно изменить пароли для каждого из пользователей. Для этого необходимо нажать на кнопку «Сменить пароль» напротив интересующего пользователя. Для управления группами и ролями для каждого из пользователей необходимо нажать на кнопки «Управление группами» и «Управление ролями» соответственно (рис. 49).

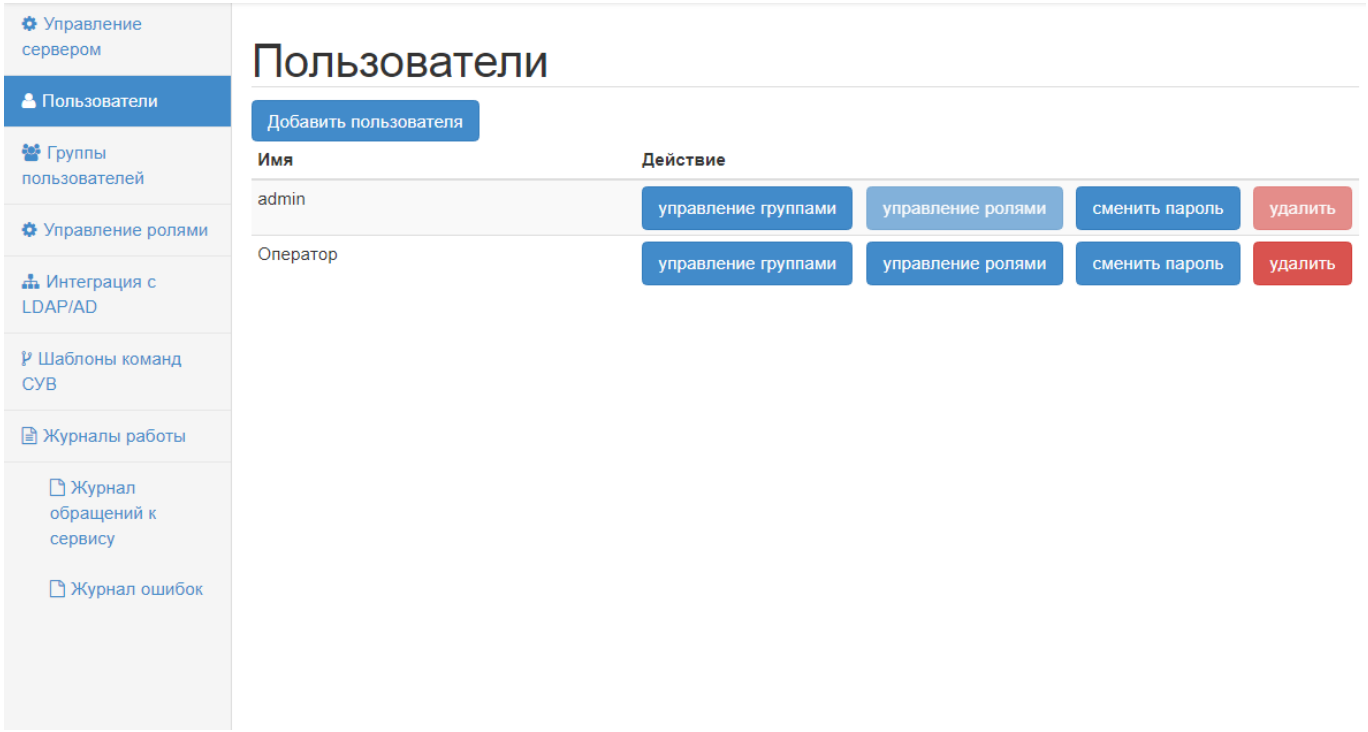

Рисунок 49 — Раздел Пользователи

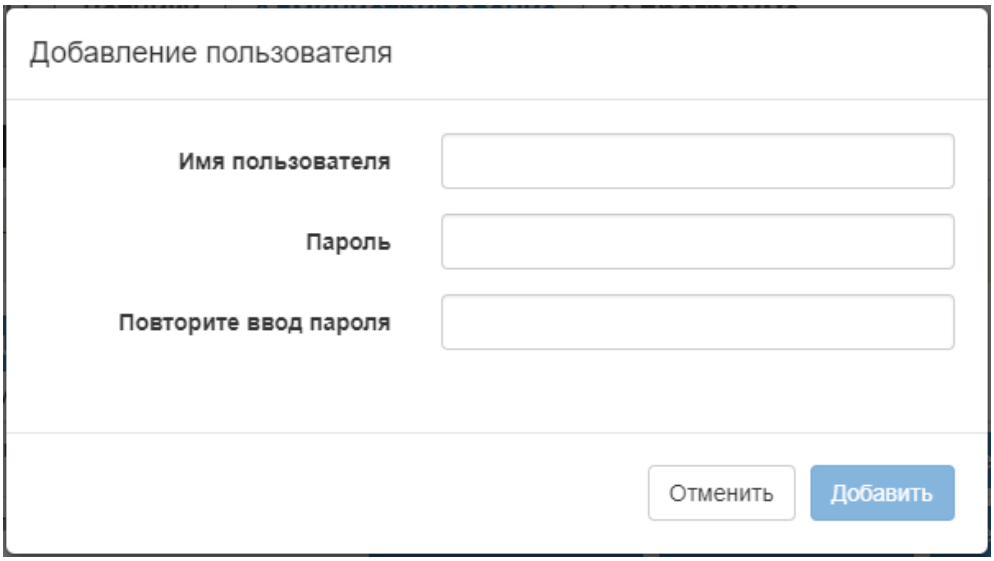

Рисунок 50 — Добавление пользователя

Для создания новой группы пользователей в разделе «Группы пользователей» необходимо нажать кнопку «Добавить группу» (рис. 51). В открывшемся окне введите имя группы и нажмите кнопку «Добавить» (рис. 52). Для удаления группы необходимо нажать кнопку «удалить» напротив группы (рис. 51). Для управления ролями пользователей в группе нажмите кнопку «управление ролями» (рис. 51).

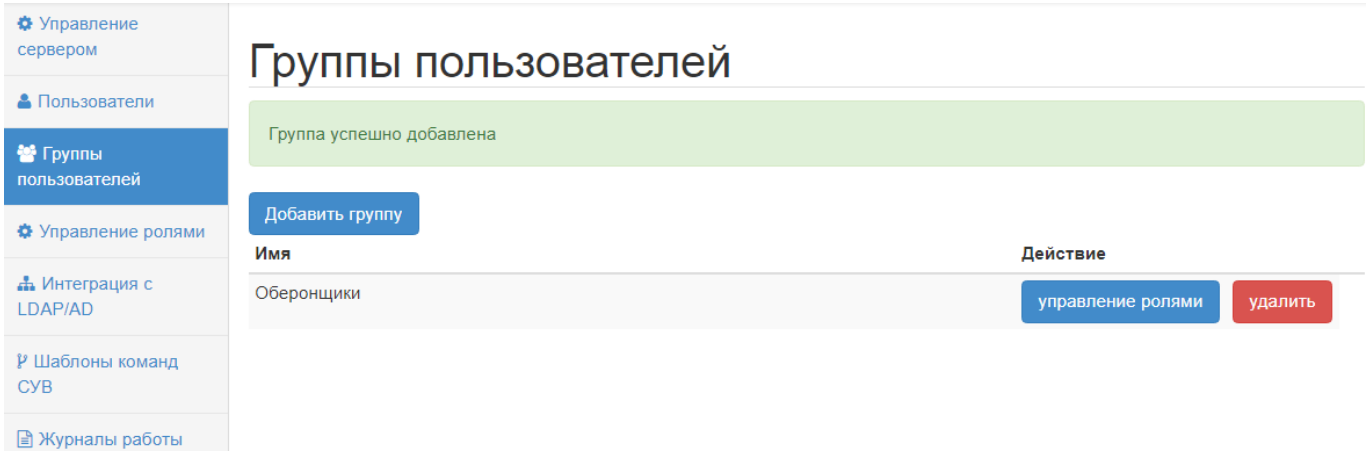

Рисунок 51 — Раздел группы пользователей

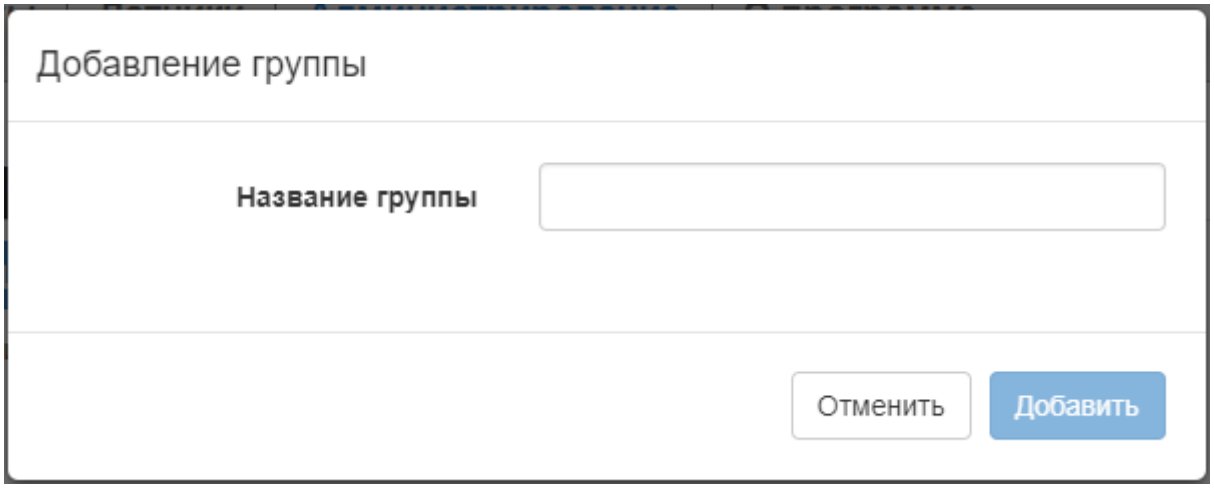

Рисунок 52 — Добавление группы

«АК-ВС 3» поддерживает создание ролей и назначения ролей пользователям и группам. Роль - это набор разрешенных действий. Роли делятся на системные и проектные. В разделе «Управление ролями» можно добавлять, редактировать и удалять роли для пользователей и групп (рис. 53).

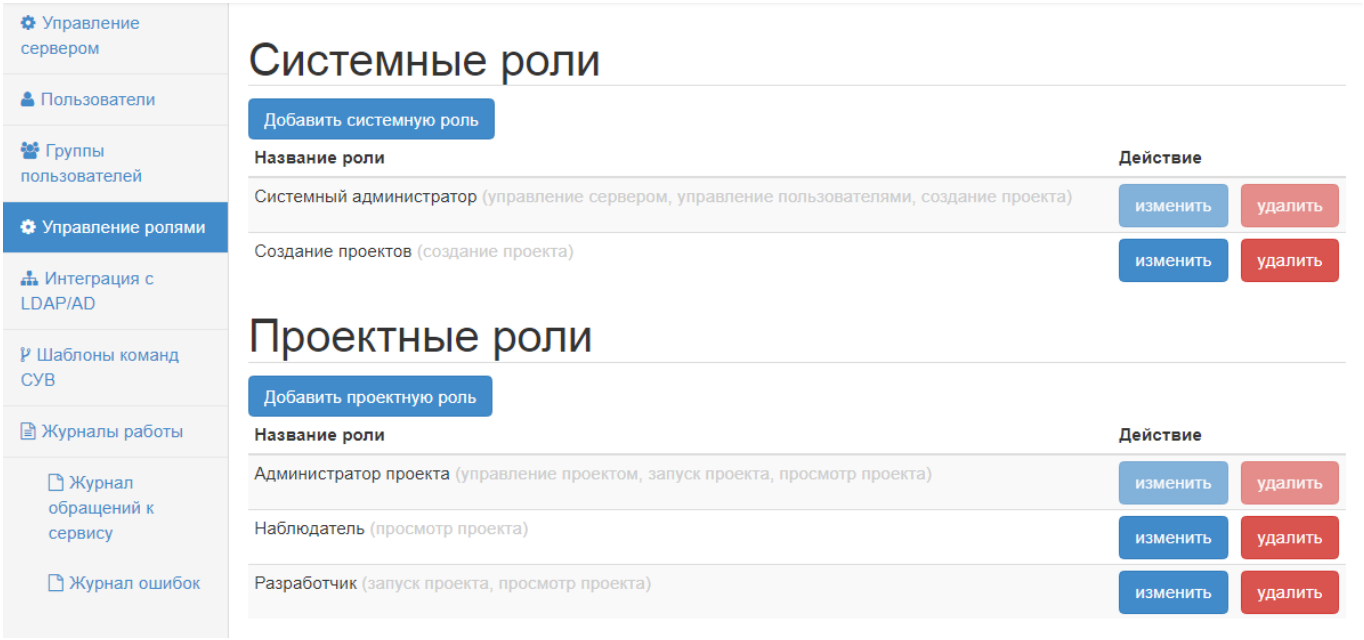

Рисунок 53 — Раздел Управление ролями

Системные роли содержат следующий набор разрешений: «управление сервером», «управление пользователями» и «создание проекта». Разрешение «управление сервером» позволяет выключать сервер, создавать и редактировать шаблоны СУВ, а также просматривать журналы работы. Разрешение «управление пользователями» позволяет добавлять, редактировать и удалять пользователей, их роли и группы. Разрешение «создание проекта» позволяет создавать новый проект в системе.

В «АК-ВС 3» предустановлены две системные роли: «Создание проектов» и «Системный администратор». Роль «Создание проектов» имеет разрешение «создание проекта». Роль «Системный администратор» имеет максимальный набор разрешений. Ее нельзя изменить или удалить.

Проектные роли содержат следующий набор разрешений: «управление проектом», «запуск проекта» и «просмотр проекта». Разрешение «управление проектом» позволяет удалять проекты. Разрешение «запуск проекта» позволяет запускать проект на новое сканирование и загружать новые версии исходного кода. Разрешение «просмотр проекта» позволяет просматривать отчеты проекта. Необходимо указать проект при назначении проектной роли. Так же можно создать проектную роль, которая будет распространяться на все проекты в системе.

В «АК-ВС 3» предустановлены три проектные роли: «Администратор проекта», «Разработчик», «Наблюдатель». Роль «Наблюдатель» имеет разрешение «просмотр проекта». Роль «Разработчик» имеет разрешения «просмотр проекта» и «запуск проекта». Роль «Администратор проекта» имеет максимальный набор разрешений и ее нельзя изменить или удалить.

Для назначения роли пользователю необходимо нажать кнопку «управление ролями» в разделе «Пользователи» напротив необходимого пользователя (рис. 49).

Для назначения роли группе необходимо нажать кнопку «управление ролями» в разделе «Группы пользователей» напротив нужной группы.

3.4.5.3. Интеграция с LDAP/AD

«АК-ВС 3» поддерживает интеграцию с LDAP и AD. Для настройки перейдите в раздел «Интеграция с LDAP/AD», введите необходимые данные и нажмите кнопку «Сохранить» (рис. 54).

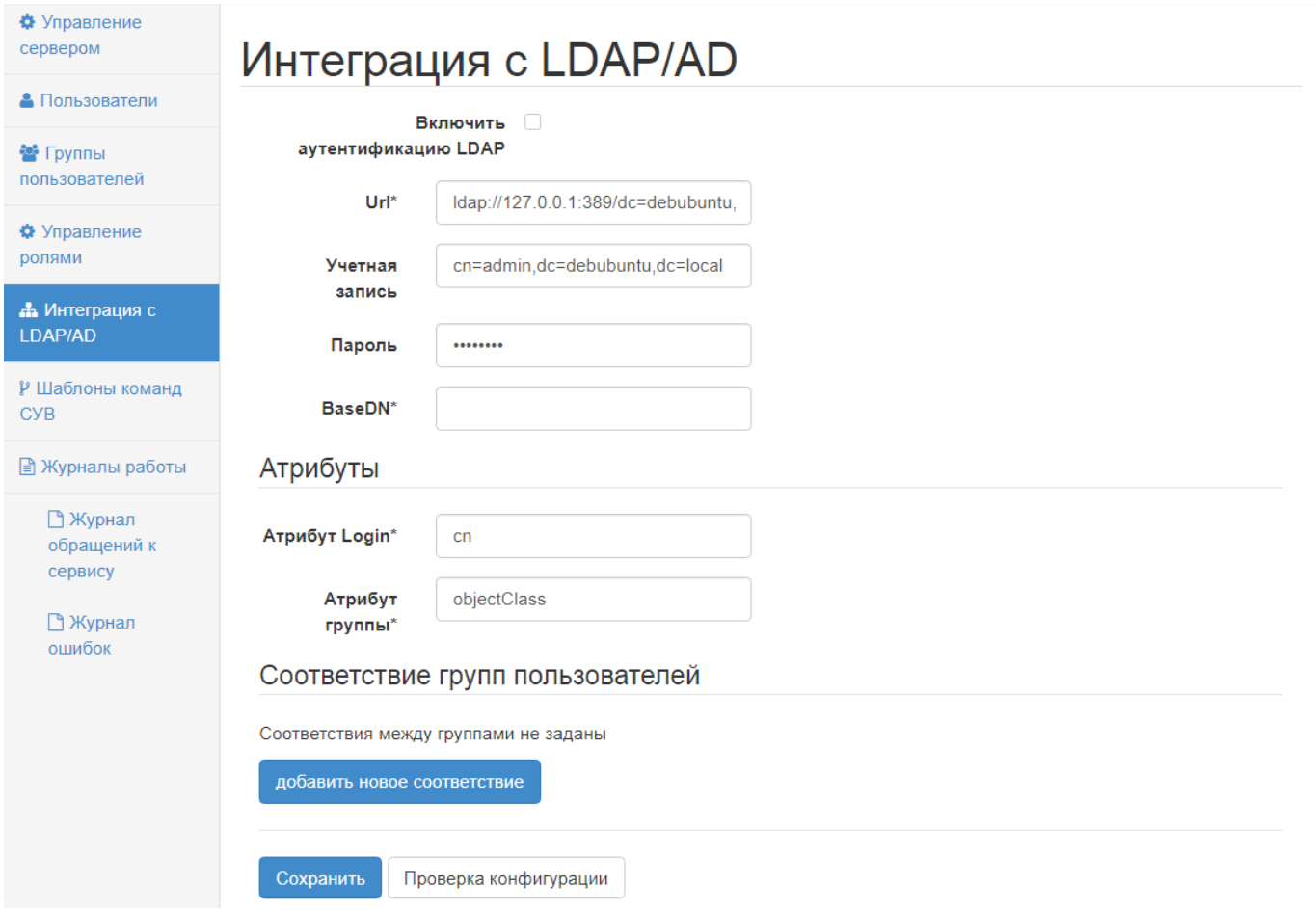

Рисунок 54 — Раздел «Интеграция с LDAP/AD»

3.4.5.4. Журналы работы

Раздел «Журналы работы» содержит «Журнал обращений к серверу» и «Журнал ошибок» (рис. 55).

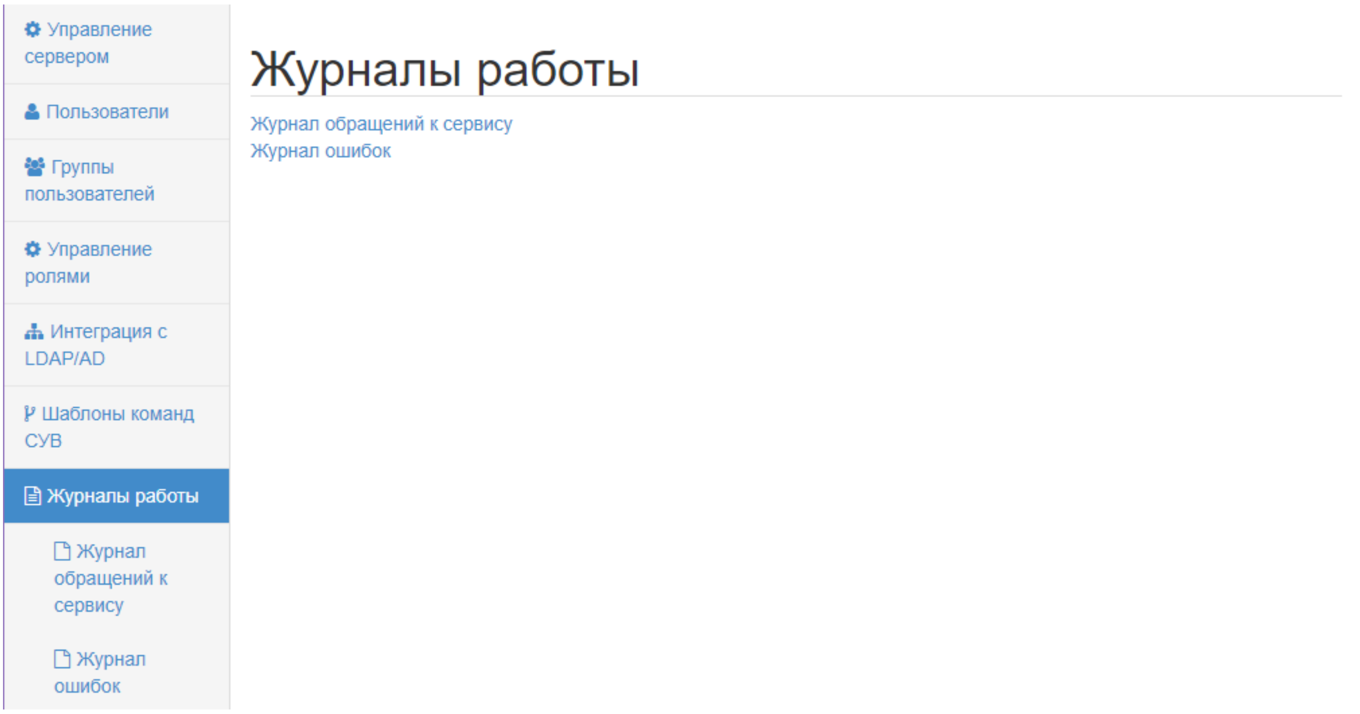

Рисунок 55 — Раздел Журналы работы

Журнал обращений к серверу регистрирует входящие запросы. Для просмотра журнала обращений к сервису необходимо нажать кнопку «Журнал обращений к сервису» (рис. 56).

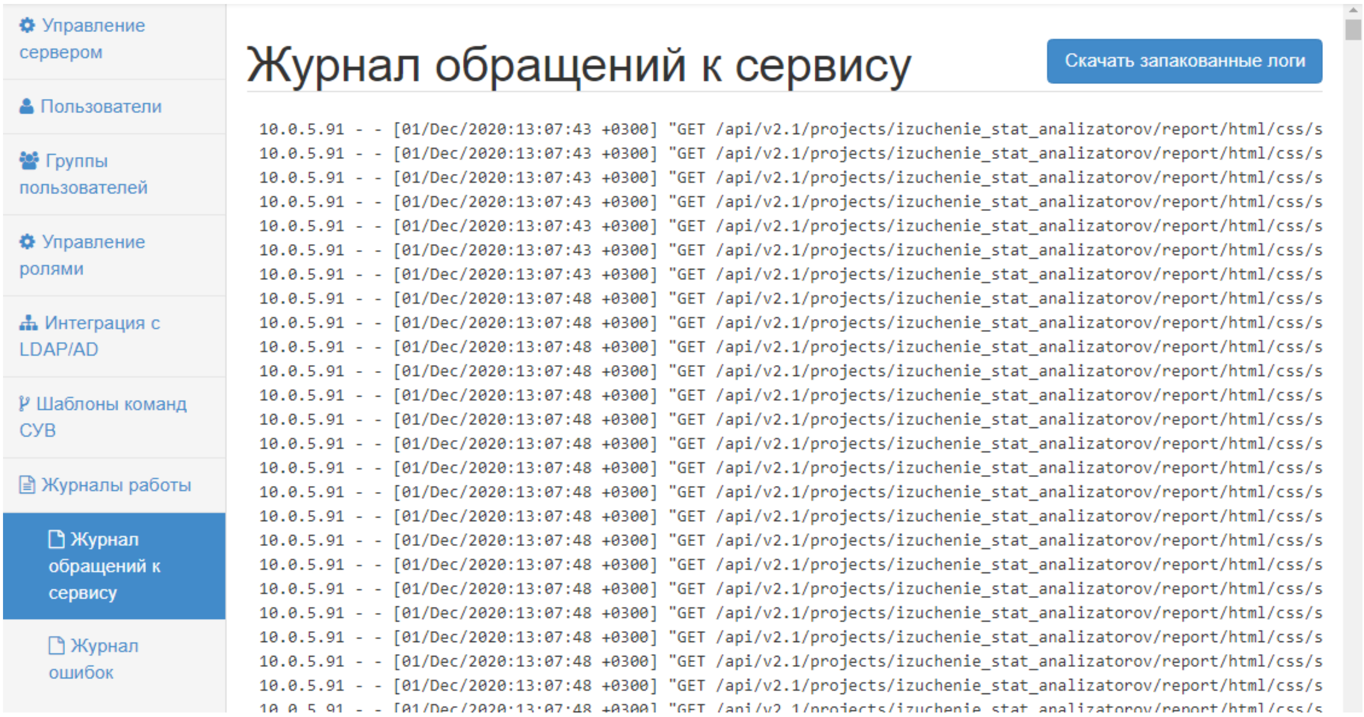

Рисунок 56 — Журнал обращений к сервису

Журнал ошибок регистрирует сбои в работе программы. Для просмотра журнала ошибок необходимо нажать кнопку «Журнал ошибок».

3.4.6. Завершение работы с программой

Для завершения работы программы необходимо нажать кнопку «Выключить сервер» на вкладке «Администрирование» в разделе «Управление сервером» (рис. 48).

4. УТИЛИТА ИССЛЕДОВАНИЯ СОСТАВА ИСПОЛНЯЕМЫХ ФАЙЛОВ

4.1. Описание

Binfo – утилита для извлечение различной информации из исполняемых и линкуемых файлов.

Поддерживаются следующие форматы:

- Portable Executable (acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp)
- Executable and Linkable Format (none, .axf, .bin, .elf, .o, .prx, .puff, .ko, .mod, .so)
- JavaArchive (.jar)

4.2. Загрузка утилиты

Утилита устанавливается в составе дистрибутива для сборочного стенда и содержится в архиве binfo.tar.gz.

4.3. Зависимости

Перед работой необходимо установить objdump (пакет binutils) и JDK 8, например, openjdk-8-jre.

4.4. Синтаксис

```
-d [путь к файлу|папке]
```

Файл/Папка для анализа. Для анализа jar необходимо указать директорию, в которой находится этот jar

```
-o [путь к папке]
```

Путь для вывода (создаётся, если не существует)

```
-d [путь к папке]
```

Путь до binfo-backend (по умолчанию – текущая директория). Binfo-backend – служебная директория, содержащая утилиты, выполняющие

4.5. Вывод

Результаты работы будут в пути, указанном в ключе ‘-o’. Если ключу ‘-d’ была дана директория, то вывод будет в xml-файлах с тем же относительным путем что и у анализируемых файлов.

4.6. Примеры

```
binfo -d /foo/bar/baz.so -o ./out
```

Анализ файла baz.so, вывод данных в out/

```
binfo -d /foo/bar/ -o ./out
```

Анализ всех возможных файлов в bar/, вывод данных в out/

4.7. Формат выходных данных

4.7.1. Общая часть:

- Filename – путь до файла;
- Magic – сигнатура;
- Type – тип;
- Architecture – архитектура;
- Compiler – компилятор;
- ProgrammingLanguage – язык программирования;
- Size – размер файла (в теле тега):
  - Unit – в чём измеряется.
- CompileTime – группа тегов – время компиляции:
  - Unix – время в формате unix;
  - DateTime – дата и время.

4.7.2. Для ELF:

- Format – формат;
- Endianess – размещение байт;
- ElfVersion – версия формата elf;
- OperatingSystem – операционная система;

- Libraries – подключаемые библиотеки:
  - Library – подключаемая библиотека, имя в теле тега.
- Imports – описания импортов:
  - Function – одна импортируемая функция:
    - Атрибут from – из какой библиотеки;
    - Имя функции – в теле тега.
- Exports – экспортируемые функции:
  - Function – одна функция:
    - Атрибут Address – адрес;
    - Атрибут Name – имя;
- Sections – секции:
  - Section – одна секция:
    - Тег Name – имя;
    - Тег Size – размер:
      - Атрибут unit – в чём измеряется размер.
    - Тег Flags – флаги.

4.7.3. Для JAR:

- Manifest – манифест:
  - ManifestVersion – версия манифеста;
  - ArchiverVersion – версия архиватора;
  - SpecificationVendor – имя поставщика спецификации;
  - CreatedBy – чем создан;
  - Name – имя;
  - BuiltBy – имя пользователя, который собрал;
  - SpecificationTitle – заголовок спецификации;
  - BuildJdk – версия jdk, которым собран;
  - SpecificationVersion – версия спецификации;
  - MainClass – имя главного класса.
- Imports – список импортов:
  - Тег Class – имя класса в теле тега.

- Exports – список экспортов:
  - Тег Class – имя класса в теле тега.

4.7.4. Для Portable Executable:

- OsVersion – версия операционной системы;
- LinkerVersion – версия компоновщика;
- Checksum – контрольная сумма;
- CodeRVA – RVA начала кода программы (секции кода);
- CodeSize – размер секции кода:
  - Атрибут unit – единица измерения.
- DataRVA – RVA начала кода программы (секции данных);
- DataSize – размер секции данных:
  - Атрибут unit – единица измерения.
- Libraries – библиотеки;
  - Library – имя одной библиотеки.
- Flags – флаги:
  - Flag – один флаг;
- Imports – импорты:
  - Function – одна импортируемая функция:
    - Атрибут from – из какой библиотеки;
    - Имя функции – в теле тега;
- Exports – экспортируемые функции:
  - Function – одна функция:
    - Атрибут Address – адрес;
    - Атрибут Name – имя.
- Sections - секции:
  - Section – одна секция:
    - Тег Name – имя;
    - Тег Size – размер:
      - Атрибут unit – в чём измеряется размер;
    - Тег Flags – флаги.

5. ПОЛНОСИСТЕМНЫЙ ДИНАМИЧЕСКИЙ АНАЛИЗ

5.1. Описание

Код2 предназначен для динамического полносистемного исследования программы. Код2 осуществляет отслеживание помеченных данных (ОПД).

ОПД – это метод динамического исследования программ, основанный на отслеживании данных из внешнего источника (сеть, жёсткий диск, стандартный поток ввода и т. д.). В данном случае речь идёт о полносистемном исследовании бинарного кода. Для трансляции используется эмулятор QEMU.

Анализ состоит из нескольких шагов:

1. Создание образа и установка на него гостевой системы (или использование образа с предустановленной OS).
2. Копирование исследуемой программы в гостевую систему.
3. Формирование *konf* файла для плагина `osi`
4. Запись сценария работы программы.
5. Воспроизведение записанного сценария с активированными плагинами исследования распространения помеченных данных и записи трассы.
6. Анализ результатов - лог файлы.
7. При необходимости отладка программы с помощью `gdb`

Важно: для некоторых плагинов нужен интерфейс `osi`, поэтому при запуске ОС в эмуляторе ядру необходимо передавать опцию `nokaslr` (подробнее в описании плагина osi_linux). Также рекомендуется отключать `ASLR` для анализа программ в usermode (пользовательском режиме) режиме:

```
$ echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
```

Далее следует описание каждого шага.

5.2. Создание образа гостевой системы

Создать образ файловой системы командой:

```
$ /opt/kod2/bin/qemu-img create -f qcow2 ubuntu_18_04.qcow2 30G
```

Установить ОС:

```
$ sudo /opt/kod2/bin/kod2-system-x86_64 \
    -cdrom ubuntu-18.04.4-desktop-amd64.iso -m 2.5G -monitor stdio \
```

```
    -net nic -net user -boot once=d -hda ubuntu_18_04.qcow2 -accel kvm
```

5.3. Копирование исследуемой программы в гостевую систему

Запустить эмуляцию гостевой системы:

```
$ /opt/kod2/bin/kod2-system-x86_64 -m 2.5G -monitor stdio \
    -hda ubuntu_18_04.qcow2 -display sdl
```

В гостевой системе с помощью `ftp` скопировать программу на хост-систему:

```
$ ftp open 192.168.122.1
> `user_login`
> pass
> GET example.bin
```

user_login - пользователь хост системы,

IP адрес 192.168.122.1 – адрес, создаваемый по умолчанию. (Внимание: команда ping в режиме сети `user` не поддерживается).

В случае если гостевая система не поддерживает сеть надо воспользоваться командами для монтирования образа *qcow2* при выключенной эмуляции:

```
$ modprobe nbd max_part=16
$ /opt/kod2/bin/qemu-nbd -c /dev/nbd0 image.qcow2
$ partprobe /dev/nbd0
$ mount /dev/nbd0p1 /mnt/image
```

5.4. Формирование .konf файла для плагина OSI

Для работы плагинов `tainted_hd` и `file_taint` необходимо активировать плагины группы `osi`. Для этого надо сформировать файл конфигурации ядра:

1. Скопировать файл *kernelinfo.zip* на *host* систему.
2. Установить пакеты заголовков ядра и сборки:

```
$ apt-get install build-essential linux-headers-`uname -r`
```

3. Разархивировать kernelinfo.zip:

```
$ unzip kernelinfo.zip
```

4. Собрать kernelinfo:

```
$ cd kernelinfo
$ make
```

5. Загрузить модуль ядра:

```
$ sudo insmod -f kernelinfo.ko
```

6. Скопировать файл kernelinfo.conf на host систему.
7. Отредактировать файл kernelinfo.conf приведя его к формату GLib key-value. Подробнее в описании плагина osi_linux.

5.5. Запись сценария работы программы

В мониторе *QEMU* ввести команду для начала записи сценария:

```
> begin_record scenario_01
```

Выполнить сценарий. Для завершения записи ввести:

```
> end_record
```

5.6. Воспроизведение записанного сценария с активированными плагинами исследования распространения помеченных данных и записи трассы

Доступны следующие плагины kod2:

- Анализ:
  - `stringsearch` - поиск данных
  - `tstringsearch` - отслеживание помеченных данных
  - `file_taint` - чтение помеченных данных из файла
  - `serial_taint` - отправка или получение по последовательному интерфейсу (RS-232)
  - `tainted_net` - отправка или получение из сети
  - `tainted_hd` - запись на диск
  - `tscan_memory` - сканирование памяти
  - `tstack_tracing` - переход на помеченные адреса
  - `tainted_branch` - связь по управлению
- Вспомогательные:
  - osi - интерфейс ОС
  - osi_linux- интерфейс Linux
- Запись трассы:
  - `insn_capstone` - запись ассемблерной трассы в виде opcodes

Для воспроизведения сценария используется опция -replay. Пример команды анализа сценария с активированными плагинами:

```
$ /opt/kod2/bin/kod2-system-i386 -m 2G -monitor stdio\
    -net nic -net user -machine pc-q35-2.9 \
-drive format=raw,file=u14x86.raw,format=raw,id=drive1,if=none \
    -device driver=ide-hd,drive=drive1,discard_granularity=512 \
    -kod2 osi:os_family=linux -kod2 syscalls2:load-info=true \
    -kod2
osi_linux:kconf_file=kernelinfo.conf,kconf_group=my_kernel_info,load_n
ow=true \
    -os linux-32-ubunt -kod2
tainted_net:query_outgoing_network=true,file=quick_tnss.csv \
    -kod2 stringsearch:str=alpha1 -kod2 tstringsearch \
    -kod2 tainted_hd -kod2 tscan_memory -replay scenario_01
```

5.7. Пример: исследование клиента Secure Socket Shell (ssh)

Исследуем openssh версии 7.6p1. Для этого:

1. Копируем исходные файлы в гостевую систему с помощью ftp.
2. Скомпилируем в гостевой системе программу статически используя опцию -static и отключив опцию PIE (position independent executable). Для этого отредактируем Makefile заменив в нём строку с переменными CFLAGS, LIBS, LDFLAGS:
   CFLAGS=-g -O0 -pipe -no-pie
   LIBS=-lresolv -lcrypto -ldl -lutil -lz -lcrypt -static
   LDFLAGS=-L. -Lopenbsd-compat/ -Wl,-z,relro -Wl,-z,now -Wl,-z,noexecstack -fstack-protector-all
3. Отключим технологии KASLR и ASLR (как это сделать описано в плагине osi_linux)
4. Запишем следующий сценарий:
   - Запуск приложения командой ./ssh echelon@192.168.122.1
   - Ввод пароля "alpha1" не нажимая Enter
   - Начало записи сценария
   - Ввод пароля - нажатие клавиши Enter
   - После успешного соединения с сервером работающим на хосте – выход из оболочки (команда exit)

- Завершение записи сценария

5. Запустим анализ программы с помощью команды

```
$ /opt/kod2/bin/kod2-system-i386-m2.5G \
    -net user -net nic -monitor stdio -kod2 stringsearch:str=alpha1 \
    -kod2 tstringsearch -replay ssh -os linux-64-ubuntu \
    -kod2 osi:os_family=linux -kod2 \
    osi_linux:kconf_file=/home/echelon/panda_run/kernelinfo.conf,kcon
    f_group=u14x86_raw,load_now=true -kod2 taint2:detaint_cb0=true \
    -kod2 tainted_net:query_outgoing_network=true,file=quick_tnss.csv
    -kod2 tainted_hd -kod2 tainted_branch -kod2log foo.plog \
    -kod2 tstack_tracing -kod2 tscan_memory
```

Активированы плагины tainted_net, tainted_hd, tainted_branch, tstack_tracing, tscan_memory.

6. В результате работы kod2 сформированы лог файлы

- foo.plog - плагин taint_branch
- quick_tnss.csv плагин tainted_net
- scan_ram.txt, scan_ram_plain.txt - плагин tscan_memory
- stack_tracing - плагин tstack_tracing
- tainted_hd - плагин tainted_hd

Рассмотрим каждый лог файл отдельно:

В первом файле foo.plog в двоичном формате представлены все случаи связи по управлению в зависимости от помеченных данных. Для перевода в текстовый файл надо воспользоваться командой

```
$ ../kod2/panda/scripts/plog_reader.py ./foo.plog  > foo.txt
```

В файле txt представлены в сгруппированном виде все места в программе, где обнаружена связь по управлению. Приведём фрагмент:

```
{
  "pc": "136024657",
  "taintedBranch": {
    "callStack": {
      "addr": [
        "134789916",
        "134783208"
      ]
```

```
    },
    "taintQuery": [
      {
        "tcn": 4,
        "uniqueLabelSet": {
          "ptr": "139853188809848",
          "label": [
            10
          ]
        },
        "ptr": "139853188809848",
        "offset": 0
      }
    ]
  },
  "instr": "414001"
},
```


- pc - значение счётчика программ в момент обнаружения связи по управлению с точностью до ББ в десятичном виде;
- tcn - счётчик меток (taint compute number);
- ptr - адрес набора меток;
- offset - смещение набора меток в рамках адреса ptr;
- instr - гостевой номер инструкции.

Для интерпретации данных этого файла необходимо сопоставить адреса "pc" с виртуальными адресами бинарного кода программы. Для этого дизассемблируем программу командой

```
$ objdump -D ssh > ssh.asm
```

Фрагмент файла ssh.asm:

```
81b920e:  66 90                   xchg   %ax,%ax

081b9210 <strlen>:
81b9210:  8b 44 24 04             mov    0x4(%esp),%eax
81b9214:  ba 03 00 00 00          mov    $0x3,%edx
…
```

```
81b924e:  4a                      dec    %edx
81b924f:  73 58                   jae    81b92a9 <strlen+0x99>
81b9251:  31 ca                   xor    %ecx,%edx
81b9253:  81 e2 00 01 01 01       and    $0x1010100,%edx
81b9259:  75 4e                   jne    81b92a9 <strlen+0x99>
81b925b:  8b 08                   mov    (%eax),%ecx
81b925d:  83 c0 04                add    $0x4,%eax
…
81b92cd:  66 90                   xchg   %ax,%ax
81b92cf:  90                      nop

081b92d0 <__strnlen>:
81b92d0:  53                      push   %ebx
81b92d1:  e8 ea fb e8 ff          call   8048ec0
<__x86.get_pc_thunk.bx>
```

Значение PC в лог файле соответствует ББ:

```
81b9251:  31 ca                   xor    %ecx,%edx
81b9253:  81 e2 00 01 01 01       and    $0x1010100,%edx
81b9259:  75 4e                   jne    81b92a9 <strlen+0x99>
```

Для нахождения всех мест срабатывания необходимо сопоставить все адреса лог файла с адресами в ssh.asm.

Второй файл quick_tnss.csv - содержит адрес назначения и листинг всех пакетов, отправленных в сетевое устройство. Помеченные данные имеют метку, отличную от NULL.

Фрагмент файла quck_tnss.csv:

```
"Address","Datum","Labels"
Destination IP: 192.168.122.1
0x55e7b2cff4d0,R, NULL
0x55e7b2cff4d1,U, NULL
0x55e7b2cff4d2,., NULL
0x55e7b2cff4d3,., NULL
0x55e7b2cff4d4,., NULL
0x55e7b2cff538,., NULL
0x55e7b2cff539,., NULL
...
```

```
0x55e7b2cff53a,., 10
0x55e7b2cff53b,., 10
0x55e7b2cff53c,-, 10
0x55e7b2cff53d,., 10
0x55e7b2cff53e,., 10
0x55e7b2cff53f,., 10
0x55e7b2cff540,., 10
0x55e7b2cff541,., 10
0x55e7b2cff542,b, NULL
0x55e7b2cff543,., NULL
```

Печатный символ записывается сразу после адреса сетевого драйвера, в случае если символ непечатный, то записывается точка. В нашем случае все данные перед отправкой в сеть были зашифрованы, поэтому почти все помеченные символы непечатные.

В файле только один адресат - 192.168.122.1, что соответствует host системе.

scan_ram.txt, scan_ram_plain.txt - файлы сканирования памяти. В этих файлах фиксируются все помеченные данные в конце записанного сценария в физической памяти гостевой системы. В нашем случае они пустые, то есть помеченных данных не обнаружено после закрытия программы.

stack_tracing - лог файл, в котором фиксируются все случаи перехода на помеченные адреса в результате выполнения ассемблерных инструкций RET CALL JMP. В нашем случае данный файл пуст. В случае если в файле фиксируются какие-либо случаи перехода на помеченные адреса - это могло бы говорить, например, о наличии уязвимости и выполнении так называемого shellcode.

tainted_hd - лог файл, в котором фиксируются все случаи записи помеченных данных в файловую систему. Фрагмент файла:

```
FILENAME : socket:TCP instr_count=6911362
Current process: ssh PID:1141 PPID:1023 Current asid: 0x36717600
0x99629214,., 10
0x99629215,., 10
0x99629216,-, 10
0x99629217,., 10
0x99629218,., 10
0x99629219,., 10
0x9962921a,., 10
```

Первая строчка содержит название файла (в нашем случае псевдо файла), socket:TCP и номер инструкции, когда произошёл системный вызов. На второй строчке - процесс ssh, его PID и PPID, а также номер asid (address space ideintifier) - уникальный номер процесса в адресном пространстве (для архитектуры x86 это регистр CR3). Следующие строки содержат физические адреса инструкций записи данных в файл, записываемые данные и значение метки.

6. СООБЩЕНИЯОПЕРАТОРУ

Предупреждение об удалении проекта (рис. 57) появляется при нажатии кнопки «Удалить проект» на странице проекта. Для удаления проекта необходимо нажать кнопку «Да». В противном случае следует нажать кнопку «Нет».

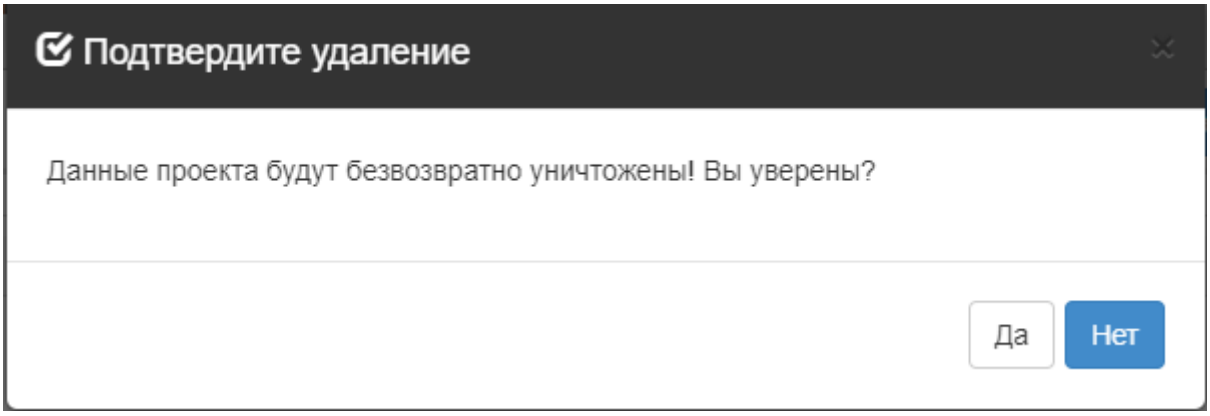

Рисунок 57 — Подтверждение удаления проекта

Предупреждение об удалении пользователя (рис. 58) появляется при нажатии кнопки «Удалить» напротив пользователя в разделе «Пользователи» на вкладке «Администрирование». Для удаления пользователя необходимо нажать кнопку «Да». В противном случае следует нажать кнопку «Нет».

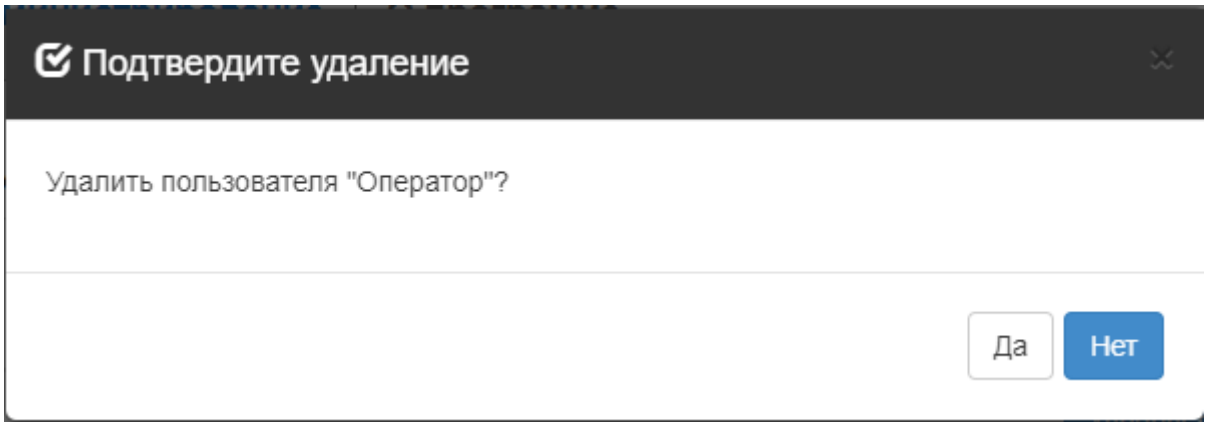

Рисунок 58 — Подтверждение удаления пользователя

Предупреждение об удалении группы (рис. 59), появляется при нажатии кнопки «Удалить» напротив группы в разделе «Группы пользователей» на вкладке «Администрирование». Для удаления группы пользователей необходимо нажать кнопку «Да». В противном случае следует нажать кнопку «Нет».

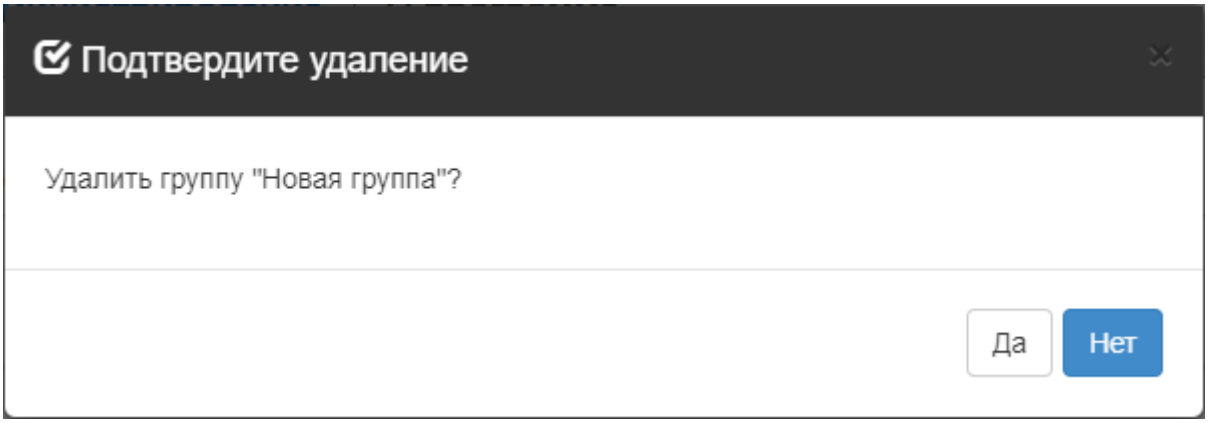

Рисунок 59 — Подтверждение удаления группы пользователей

Предупреждение об удалении роли (рис. 60) появляется при нажатии кнопки «Удалить» напротив роли в разделе «Управление ролями» на вкладке «Администрирование». Для удаления необходимо нажать кнопку «Да». В противном случае следует нажать кнопку «Нет».

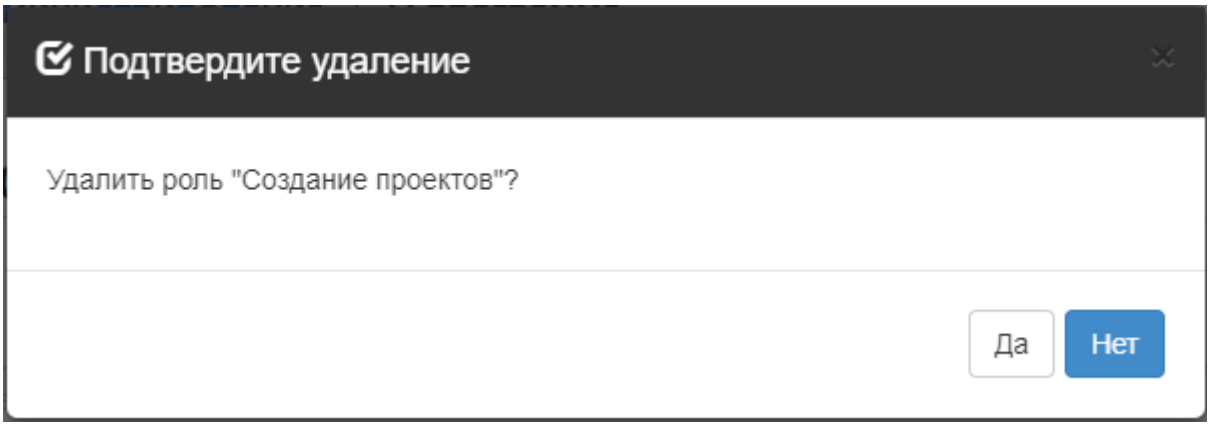

Рисунок 60 — Подтверждение удаления роли

Предупреждение об удалении шаблона СУВ (рис. 61) появляется при нажатии кнопки «Удалить» напротив шаблона в разделе «Шаблоны команд СУВ» на вкладке «Администрирование». Для удаления шаблона СУВ необходимо нажать кнопку «Да». В противном случае следует нажать кнопку «Нет».

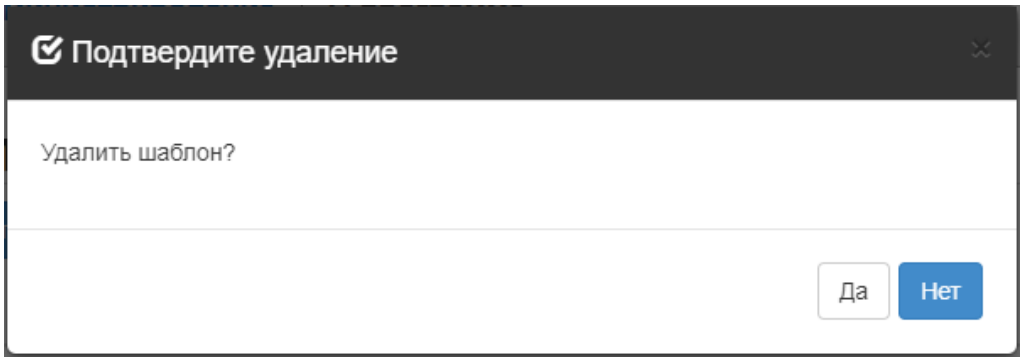

Рисунок 61 — Подтверждение удаления шаблона СУВ

Сообщение о выключении сервера (рис. 62) появляется при нажатии на кнопку «Выключить сервер» на вкладке «Администрирование» в разделе «Управление сервером». Окно с сообщением закроется автоматически.

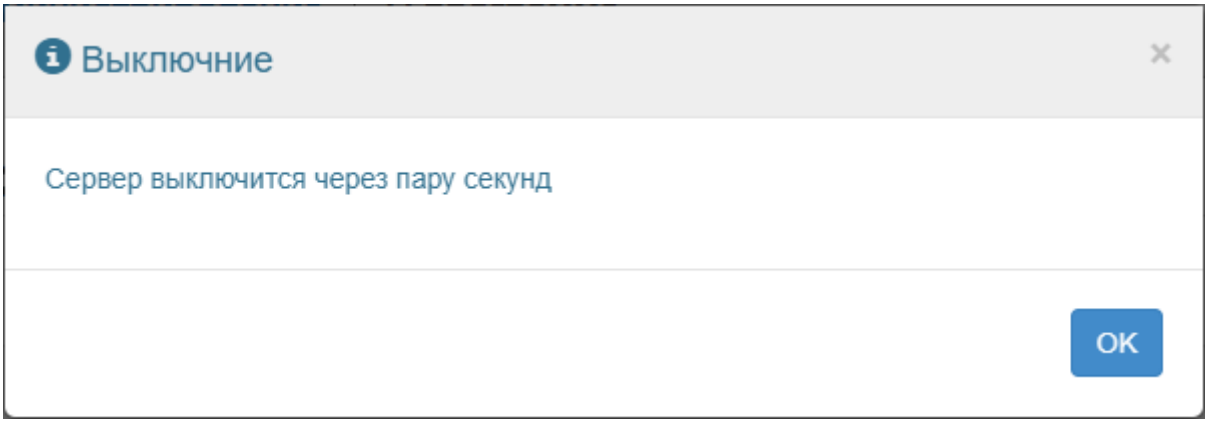

Рисунок 62 — Сообщение о выключении сервера

7. ИСР ЭШЕЛОНИУМ

7.1. Подготовка проекта

После анализа проекта пользователю будет доступна кнопка скачивания архива проекта для просмотра в «Эшелониум» (рис. Рисунок 63).

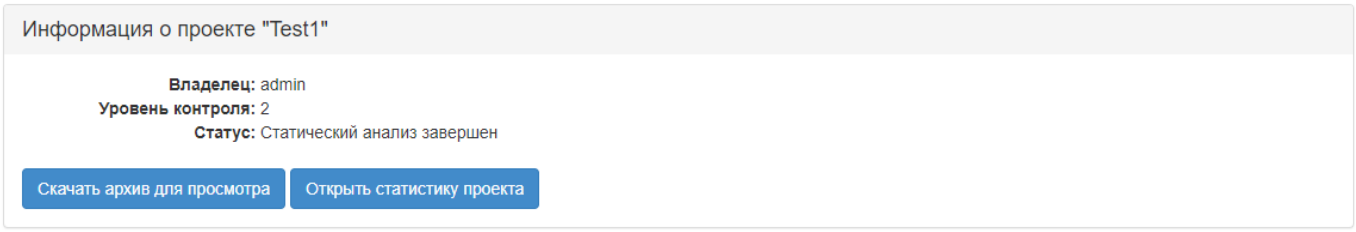

Рисунок 63 — Кнопка скачивания архива проекта

Распаковав архив, следует перейти в директорию «src» проекта и запустить скрипт run.sh (run.bat). Он запустит сервер для связи с отчетами и переместит compile_commands.json в корневую директорию проекта, если в этом есть необходимость (рис. Рисунок 64).

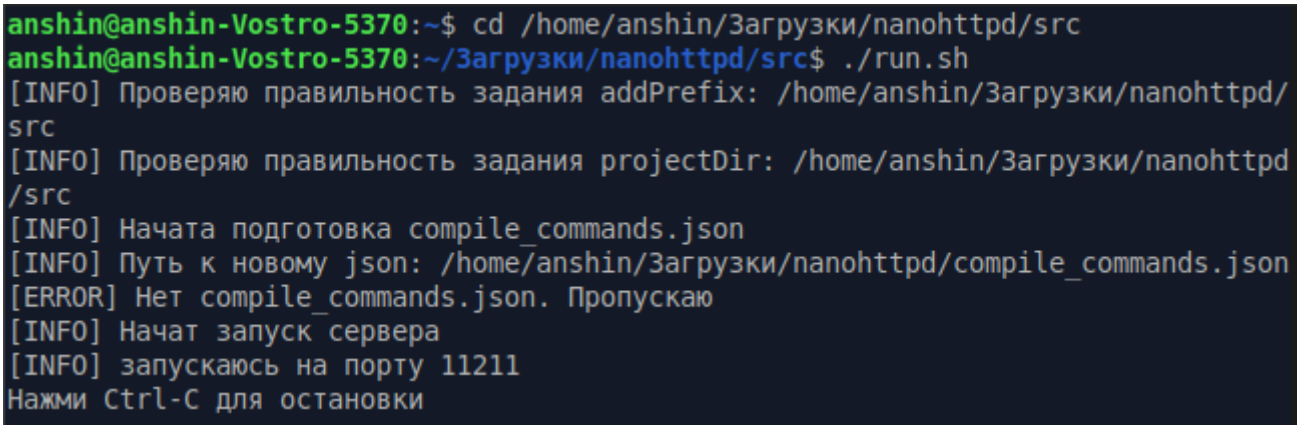

Рисунок 64 — Запуск run.sh

После этого при нажатии на ссылки в отчетах «Эшелониум» будет открывать соответствующий файл в проекте и ставить курсор на нужную строку (рис. Рисунок 65).

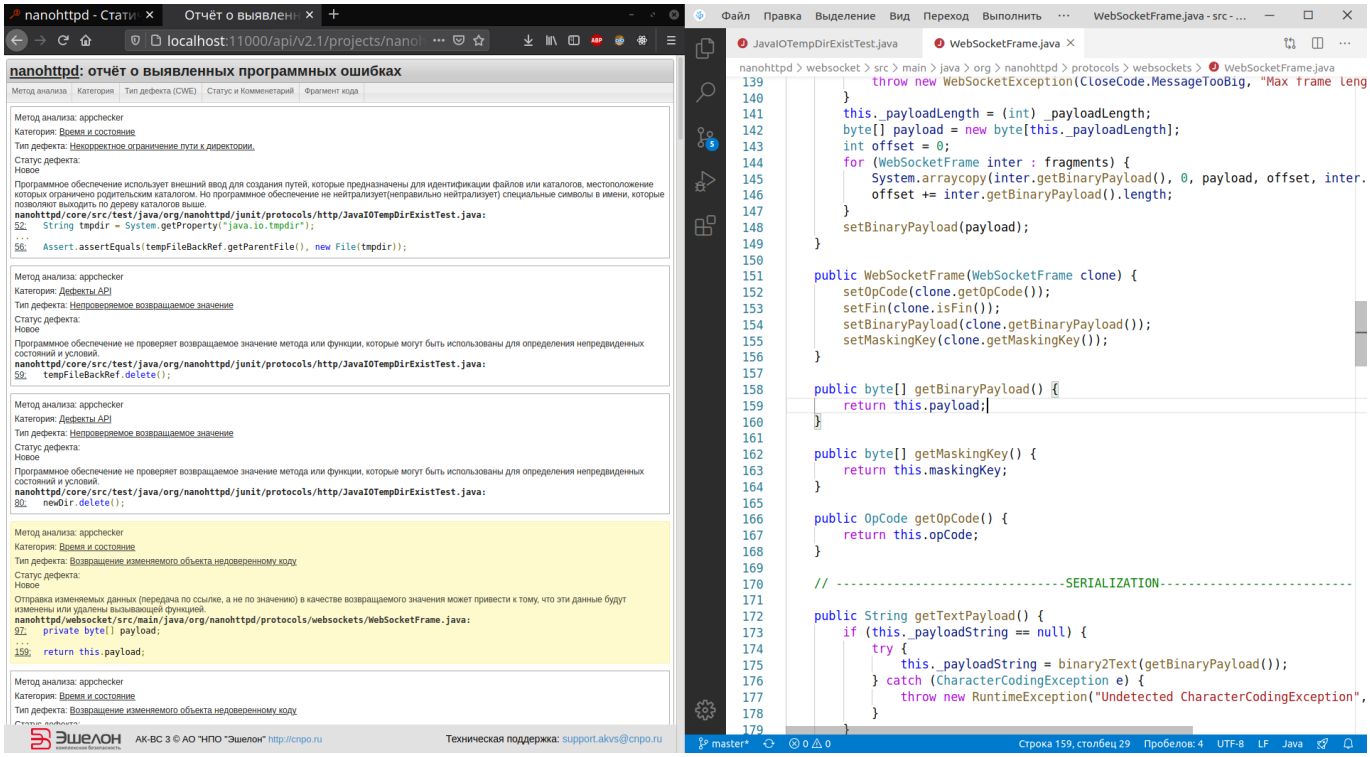

Рисунок 65 — Открытие при нажатии

Лишние файлы можно закрыть, нажав на крест на вкладке или использовав горячие клавиши (Ctrl + W). По желанию можно запустить run.sh сразу в «Эшелониум», вызвав терминал (Ctrl + Shift + ~ (Тильда)). При этом терминалов может быть несколько (рис. Рисунок 66). Добавляются они той же комбинацией или кнопкой «+».

Рисунок 66 — Выбор терминала

7.2. Работа с исходниками

Для проектов Java и C/C++ расширены возможности работы с исходным кодом. Дополнительно введены:

- больший охват переходов от использований функций и классов к их определениям;
- автодополнение;
- подсветка предупреждений и ошибок;
- автоимпорт;

- рефакторинг;
- форматирование.

Расширения придерживаются общих горячих клавиш с «Эшелониум», а именно F12 или Ctrl + клик мыши для перехода к определению, Enter для автодополнения, Ctrl+ . для выбора действия (импорт и рефакторинг), Ctrl + Shift + I для форматирования

7.2.1. Поддержка Java

Поддержка Java требует установки Java версии 8 и выше. Для активации поддержки Java нужно открыть .java файл и согласиться использовать расширение поддержки (рис. Рисунок 67). Признаком неустановленной или ненайденной Java будет сообщение: «The "path" argument must be of type string. Received type object.».

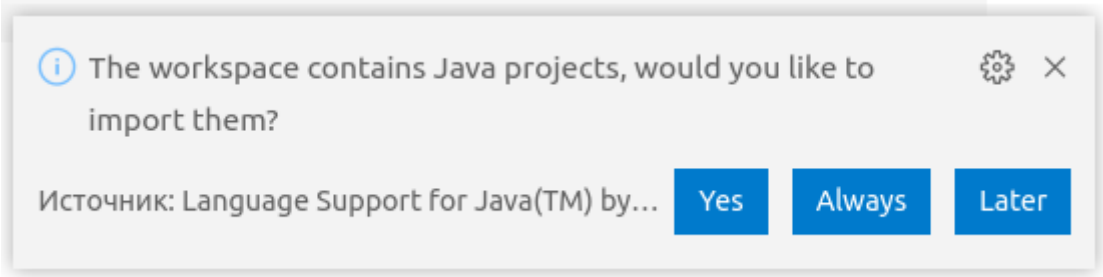

Рисунок 67 — Предложение включения поддержки

Прогресс запуска можно увидеть, если нажать на знак «Загрузка» или «Палец вверх» в правом нижнем углу. (рис. Рисунок 68).

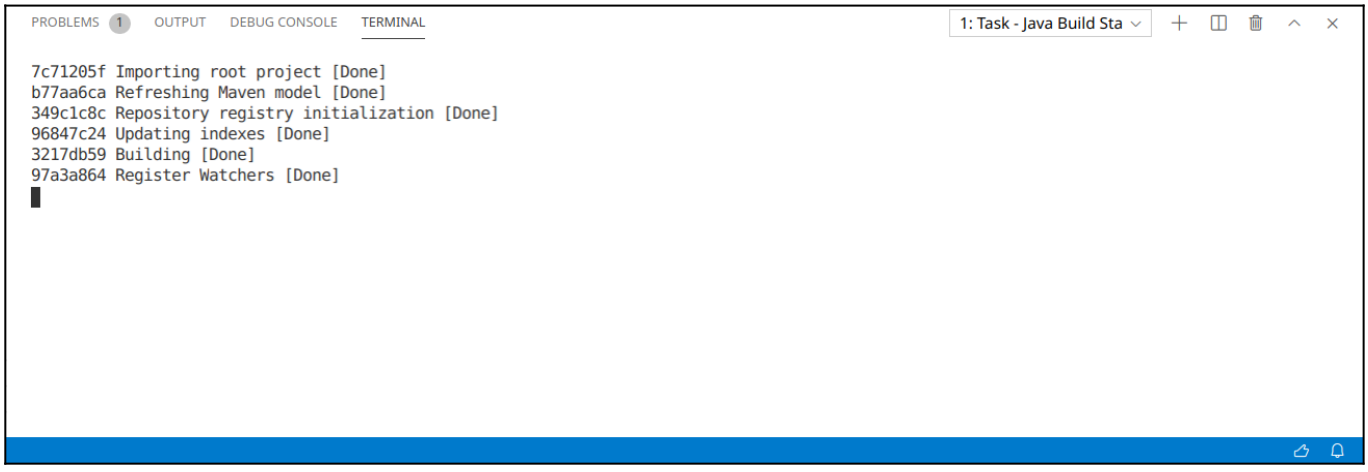

Рисунок 68—Запуск плагина для Java

После запуска в проекте заработает подсветка синтаксиса и ошибок, показ определений и переход к ним, автодополнение, анализ файлов сборщиков Maven(рис. Рисунок 69).

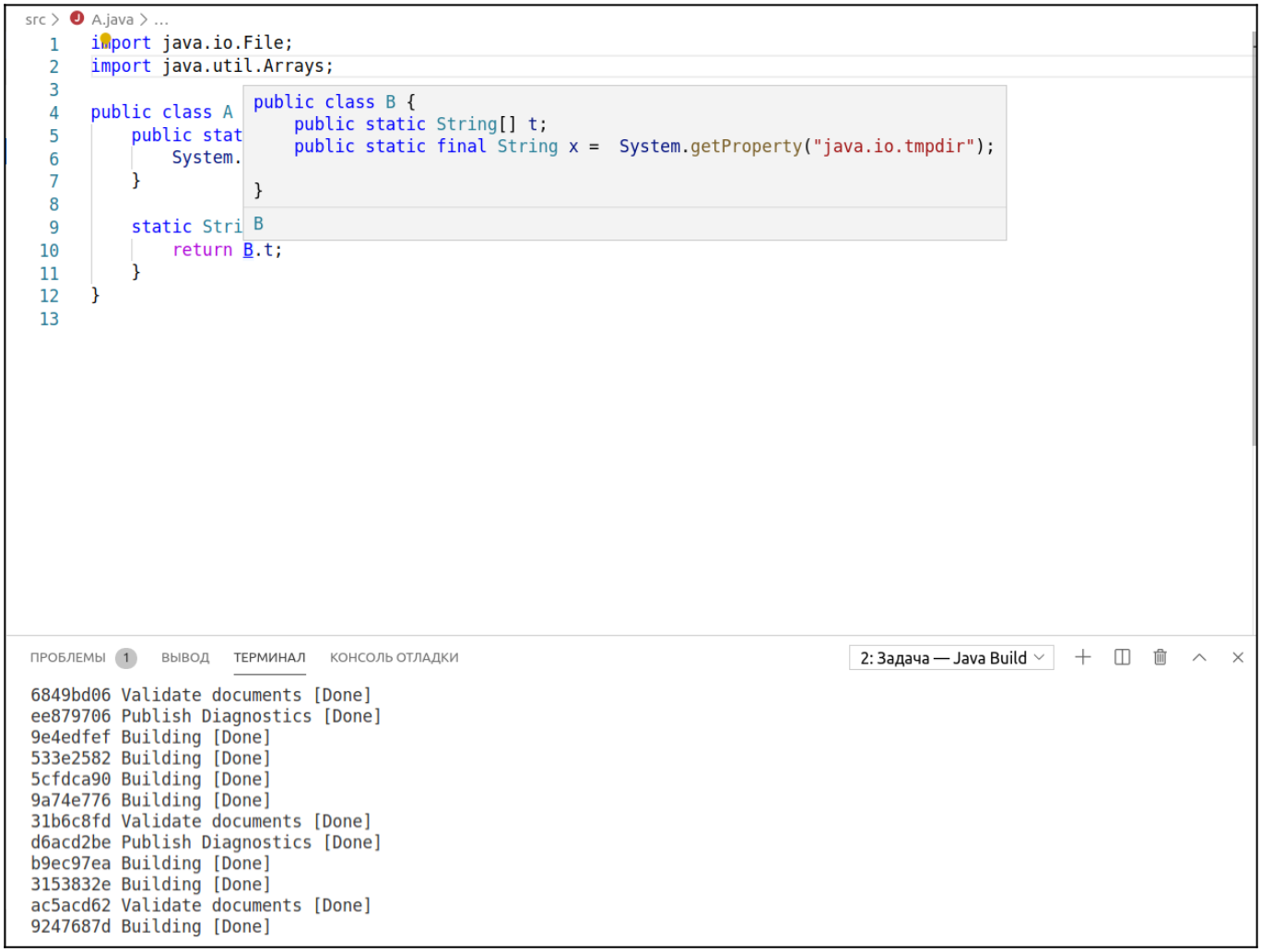

Рисунок 69 — Проект с активированным расширением

Перейти к определению можно, нажав F12 или Ctrl + клик мыши. Если определение не доступно в виде исходников, то будет использован дизассемблер (рис. Рисунок 70).

```
// Failed to get sources. Instead, stub sources have been generated by the disassembler.
 // Implementation of methods is unavailable.
package java.lang;
public final class String implements java.io.Serializable, java.lang.Comparable, java.lang.CharSequence {
```

Рисунок 70 — Результат работы дизассемблера

7.2.2. Поддержка C/C++

Дополнительный анализ кода проводится на основе clangd compiler. Для правильной работы расширенного функционала в корне проекта должен находиться compile_commands.json, который скачивается вместе с архивом проекта и перемещается в нужное место во время выполнения run.sh или run.bat (рис. Рисунок 71).

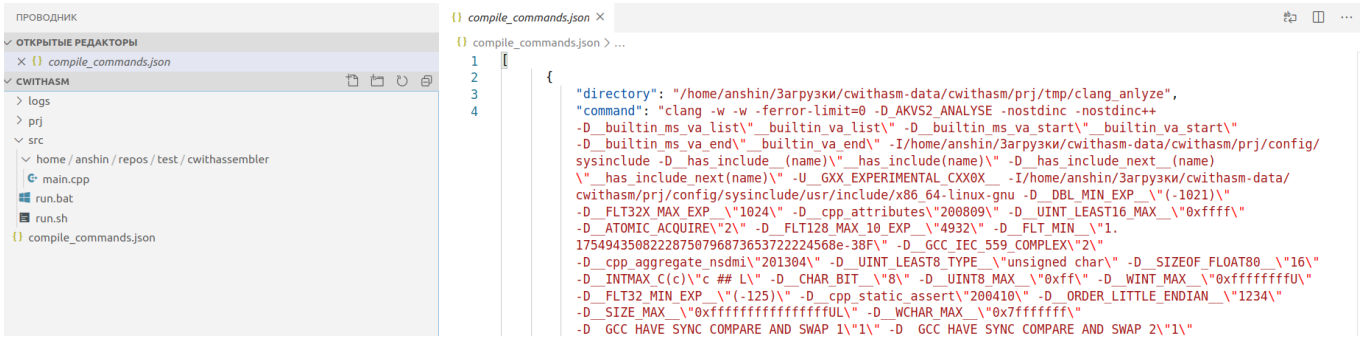

Рисунок 71 — Пример compile_commands.json в проекте

По аналогии с Java, расширение поддерживает переходы к определениям и автодополнения, анализ кода, но также добавляет поддержку include.

Функции рефакторинга включают в себя улучшенное переименование (F2), возможность автоопределения auto, извлечения сложных выражений в переменные или функции (рис. Рисунок 72).

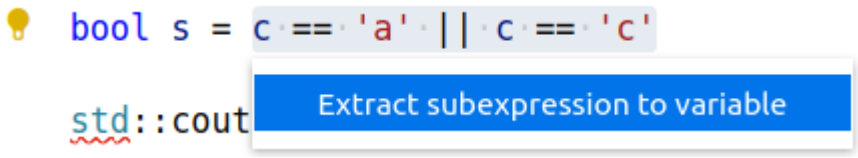

Рисунок 72 — Извлечение выражений

Для поиска дополнительно введена возможность открытия результатов в редакторе. Для этого сначала надо осуществить обычный поиск с помощью Ctrl + Shift + F, а затем нажать «Открыть в редакторе» (рис. Рисунок 73).

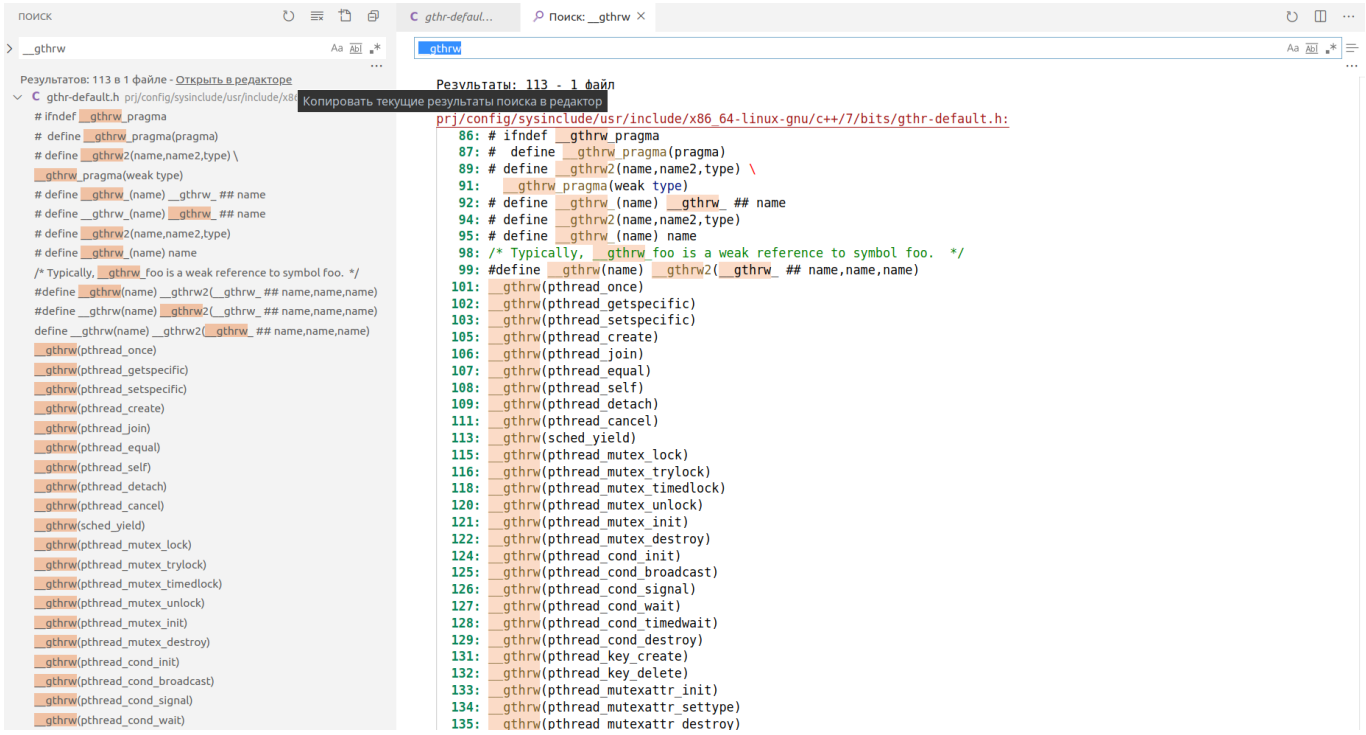

Рисунок 73 — Результаты поиска в редакторе

7.3. Интерфейс

По умолчанию поставлен языковой пакет для русского языка и светлая тема, но по желанию это можно изменить, вызвав исполнение команды (Ctrl + Shift + P) «Настройка языка интерфейса» и «Параметры: Цветовая тема».

7.4. Плагин «Echelonium-server»

Плагин является альтернативой веб-интерфейса с дополнительными возможностями.

7.4.1. Установка и первый запуск

При запуске инсталлятора клиента вместе с Эшелониумом устанавливается плагин Echelonium-server.

При успешной установке при запуске будет выведено сообщение о способе информирования пользователя (рис. Рисунок 74). Если нажать кнопку «Открыть», то откроется консоль плагина, где будет выводиться большинство информационных сообщений. Чтобы уведомление не открывалось при каждом запуске, можно нажать «Больше не показывать».

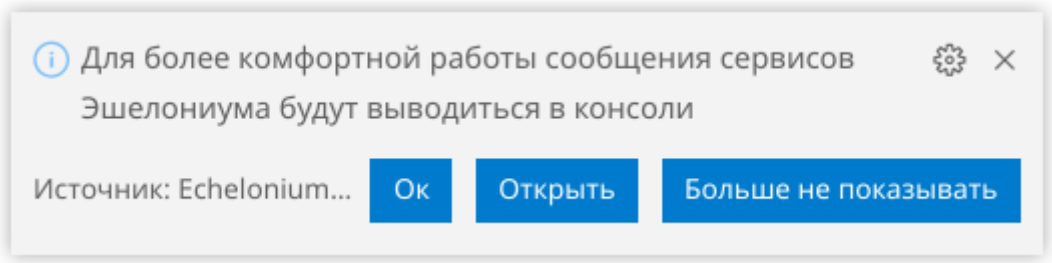

Рисунок 74 – Уведомление при запуске

Настроить папку загрузки и адрес сервера можно вызвав параметры (Ctrl + ,) и начав вводить в поиске «echelonium» или нажав «Конфигурация workspace» на главной странице после авторизации (рис. Рисунок 75). Также переход к вкладке настроек возможен нажатием на значок шестерёнки на форме авторизации.

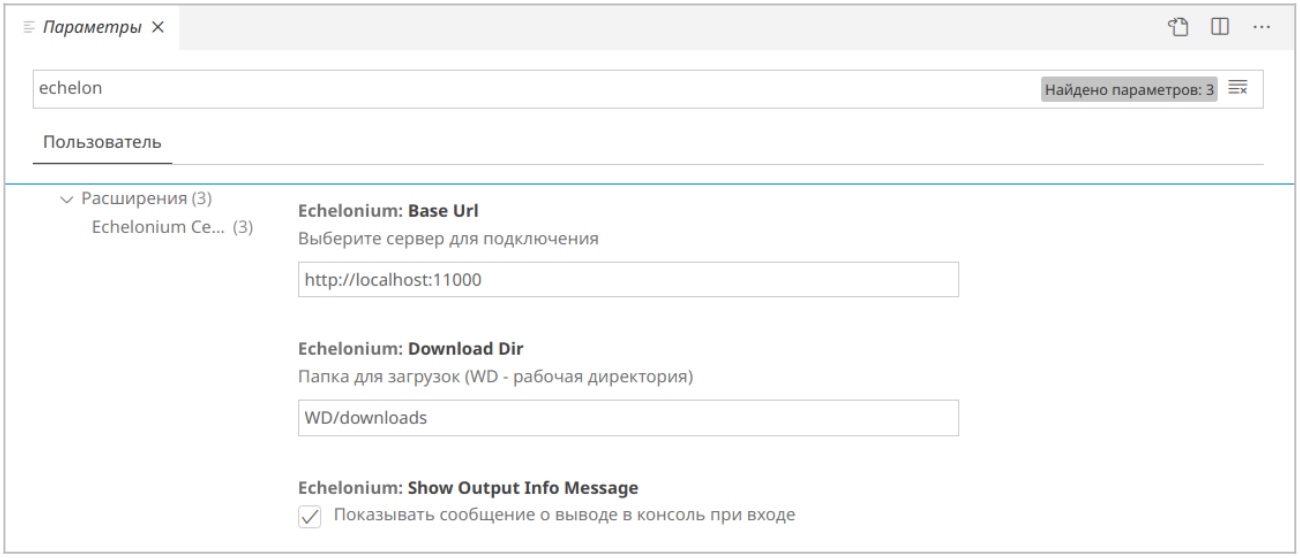

Рисунок 75 – Настройки плагина

После первоначальной настройки можно авторизоваться, вызвав набор команды (Ctrl + Shift + P) и введя команду «Панель сервера», после чего откроется форма для ввода логина и пароля. Эта же панель будет открываться при запуске, если не были открыты другие файлы. Авторизованный пользователь попадет на домашнюю страницу (рис. Рисунок 76).

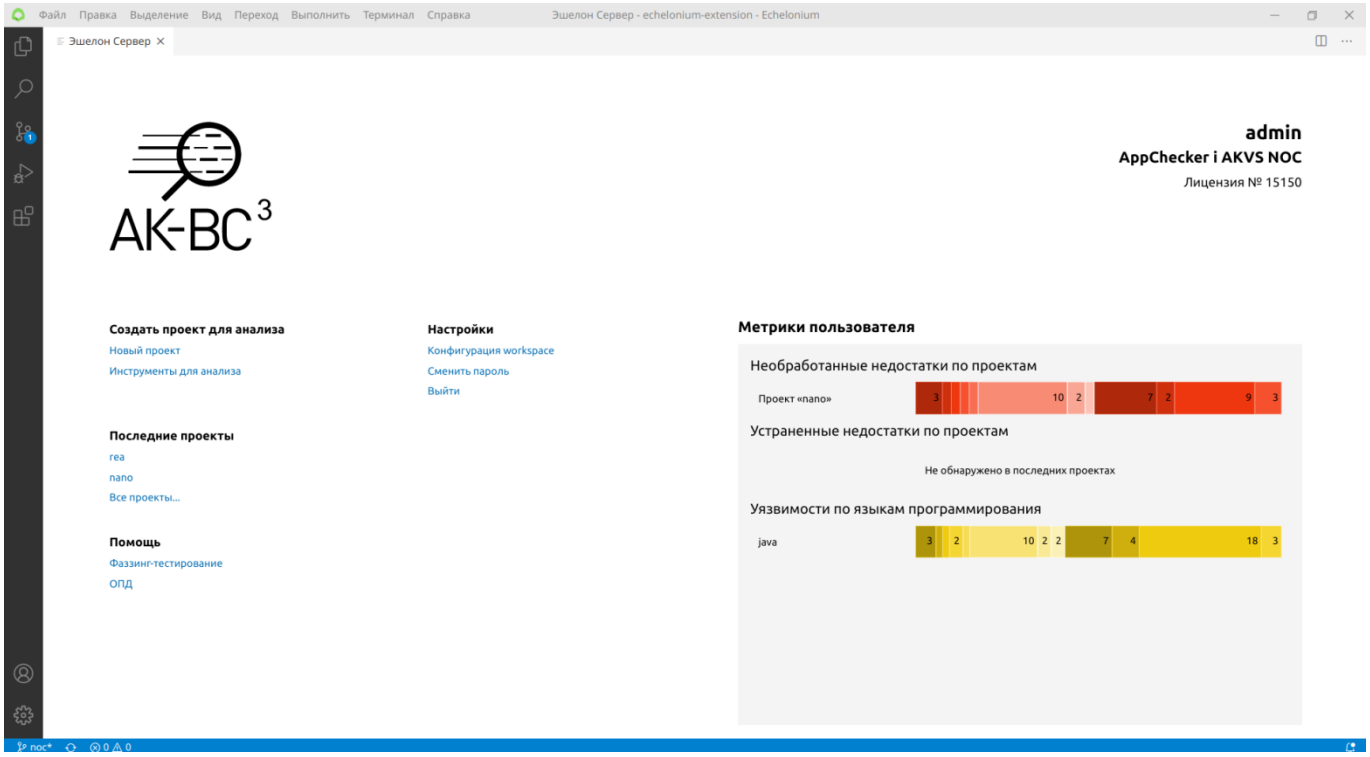

Рисунок 76 – Панель сервера

Для многих элементов интерфейса есть горячие клавиши, узнать которые можно, наведя мышь на элемент (рис. Рисунок 77).

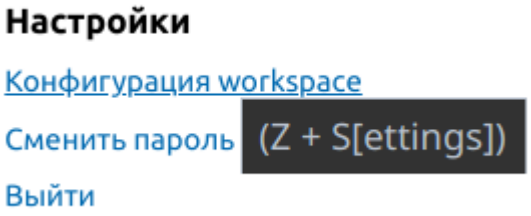

Рисунок 77 – Подсказка горячей клавиши

7.4.2. Анализ проекта

После создания проекта страница проекта. Для подготовки и запуска анализа необходимо нажать на кнопку «Статический» в разделе запуска анализа. После чего откроется форма загрузки исходников (рис. Рисунок 78). На первой странице нужно загрузить архив с исходным кодом, нажав и выбрав архив или перетащив его в область загрузки архива. Если в Эшелониуме открыта директория проекта, то при нажатии «Архивировать и загрузить Workspace» его загрузка произойдет автоматически.

Для подготовки архива с конфигурацией для проектов на C/C++ можно воспользоваться кнопкой «Запустить cppconf_linux.py» и далее следовать инструкции.

Для подготовки архива для анализа Javaможно использовать кнопку «Запустить akvs_jam».

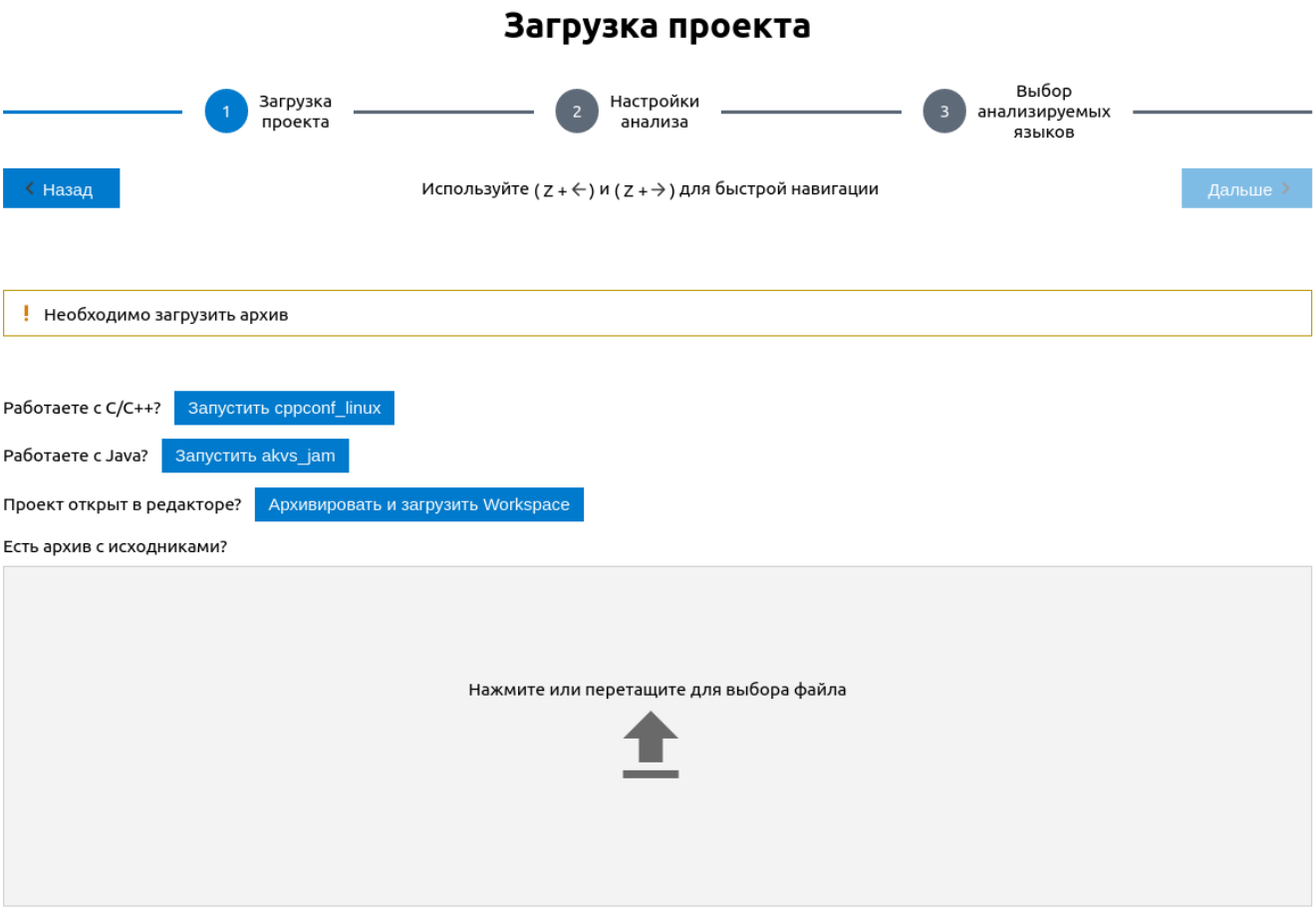

Рисунок 78 – Окно загрузки исходников в проект

После загрузки исходников и выставлении настроек анализа на последней форме будет кнопка «Начать анализ».

Логи работы находятся во вкладке «Журнал». При открытии журнала можно отключить перемотку и перенос строк кнопками в правом верхнем углу (рис. Рисунок 79).

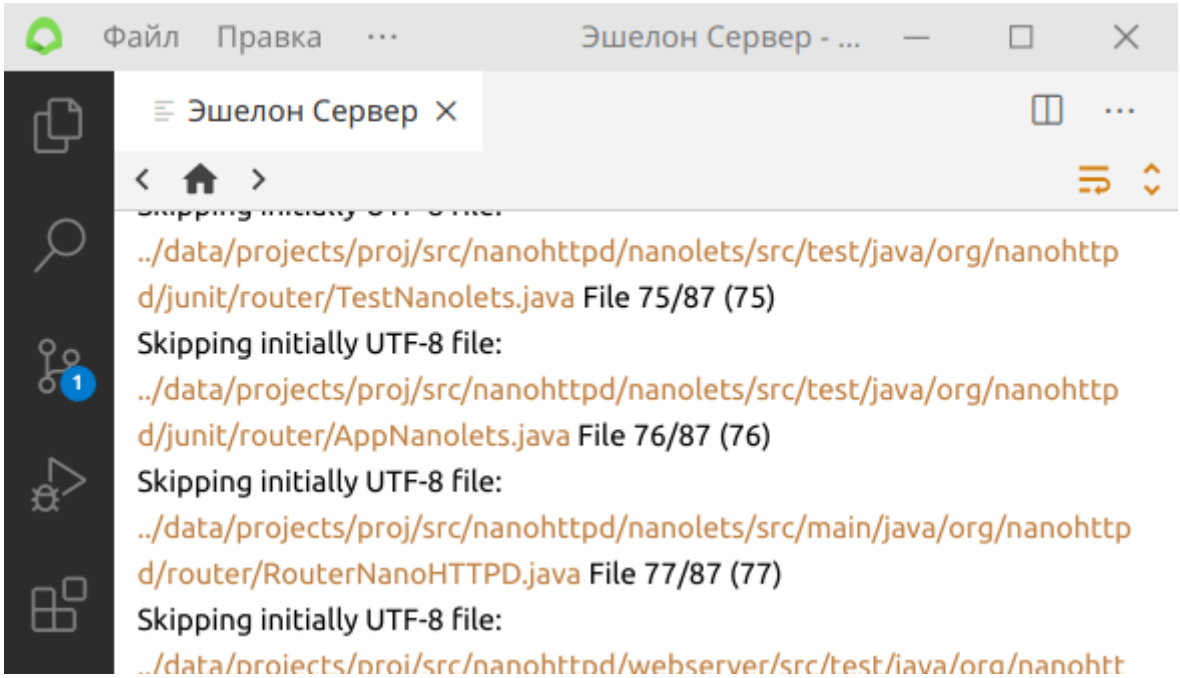

Рисунок 79 – Журнал анализа

После работы статического анализа появятся вкладки «Работа с дефектами», «Отчеты» и станет доступна кнопка «Динамический» (анализ), если был установлен флаг «Подготовить датчики». После завершения анализа к списку найденных дефектов можно перейти из таблички информации о проекте (рис. Рисунок 80) или из вкладки «Отчёты».

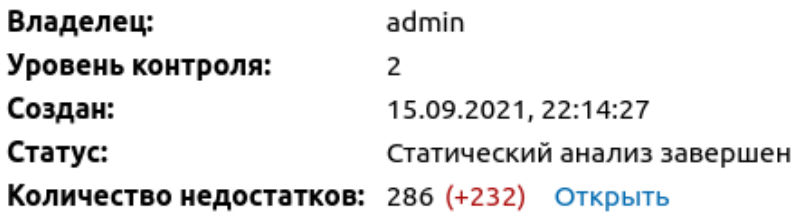

Рисунок 80 – Информация о проекте, кнопка открытия списка дефектов

При нажатии на лупу открывается меню поиска позволяющее группировать и фильтровать отображаемые дефекты. Если анализ проводился несколько раз, то в меню поиска также появляются пункты «Выбрать запуск» и «Сравнить с» для сравнения нескольких запусков между собой (рис. Рисунок 81).

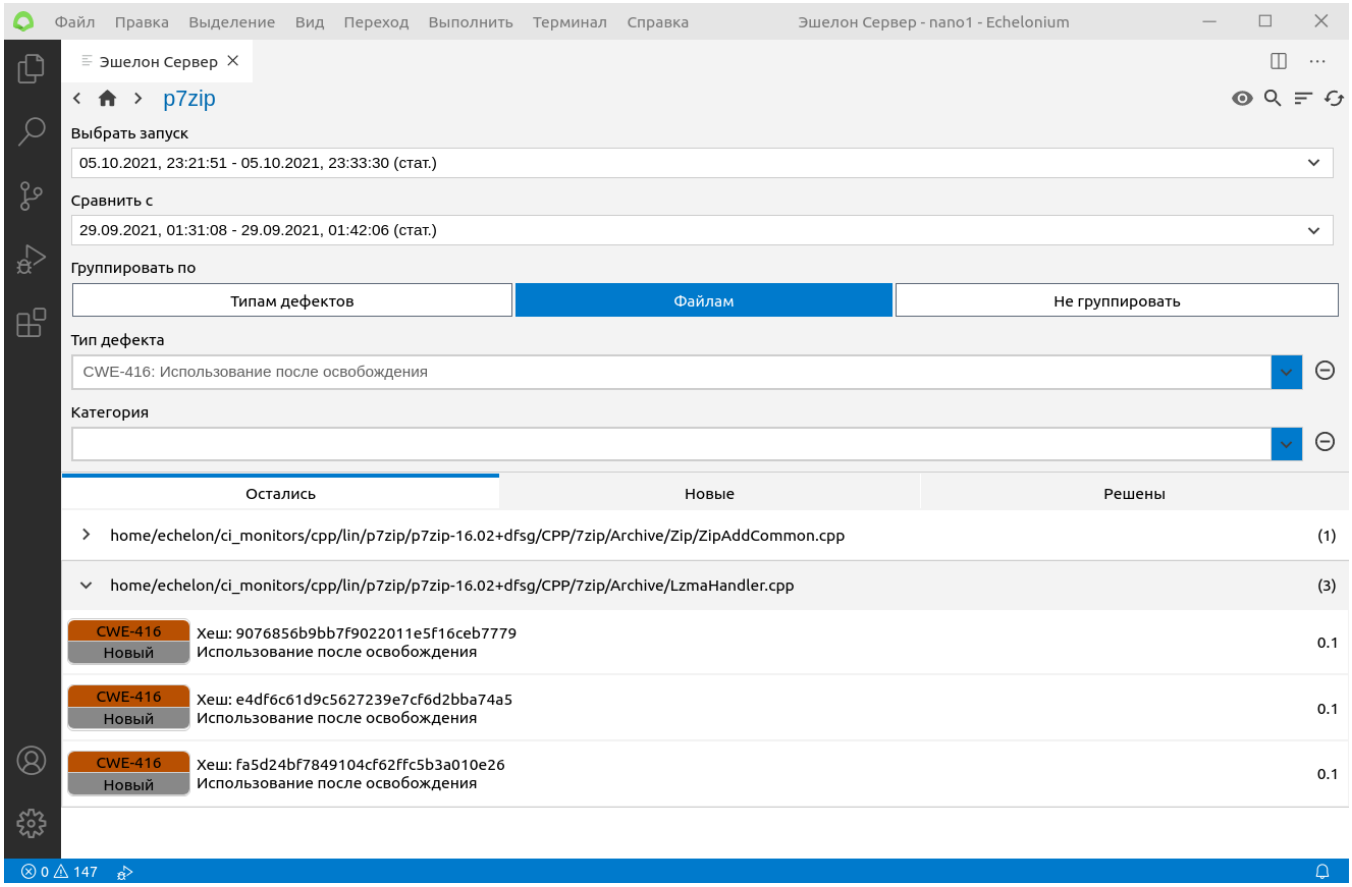

Рисунок 81 – Группировка, фильтрация и сравнение результатов

Окно динамического анализа похоже на окно загрузки исходников и настройки проекта (рис.Рисунок 82). На первой странице можно скачать исходники со вставленными датчиками, а также сами датчики. На второй странице предлагается загрузить получившиеся логи после сборки программы с датчиками и её выполнении.

Отчеты проекта разделены по категориям и могут быть скачаны с вставками частей кода из исходников или без при нажатии кнопок в правом верхнем углу вкладки «Отчеты» (рис. Рисунок 83).

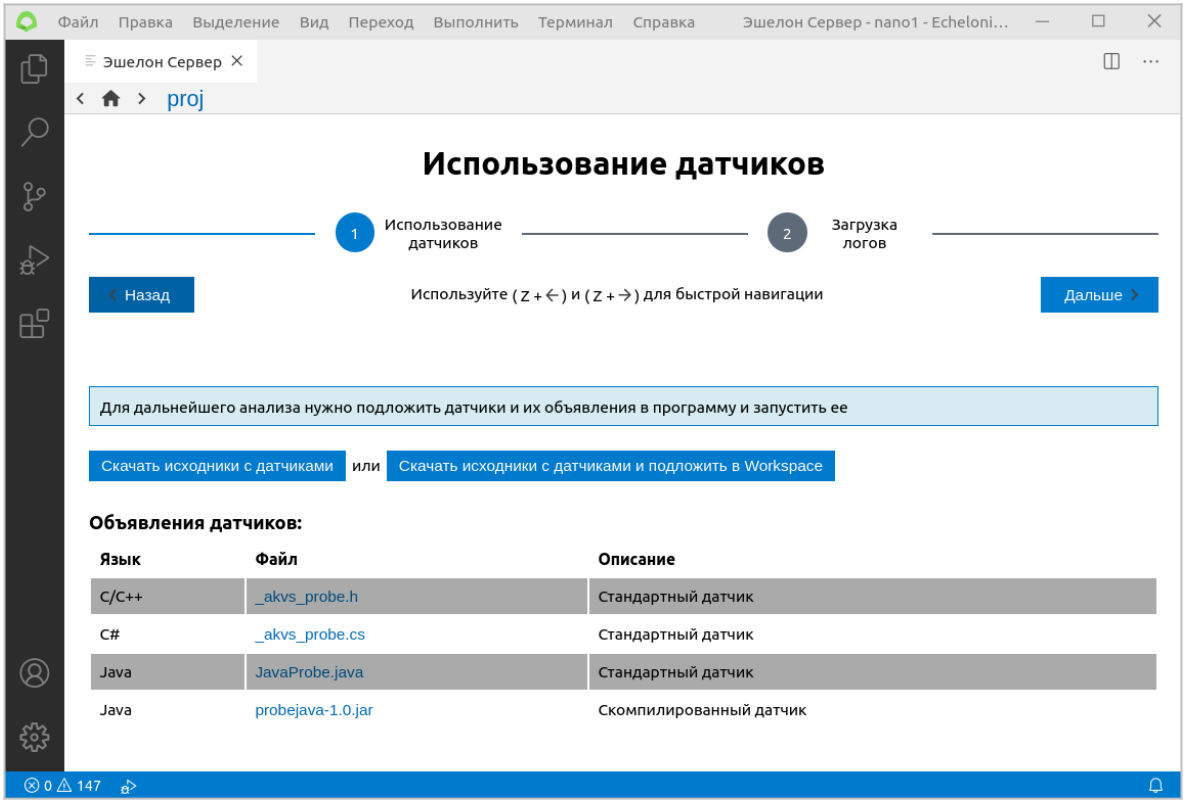

Рисунок 82 – Страница подготовки динамического анализа

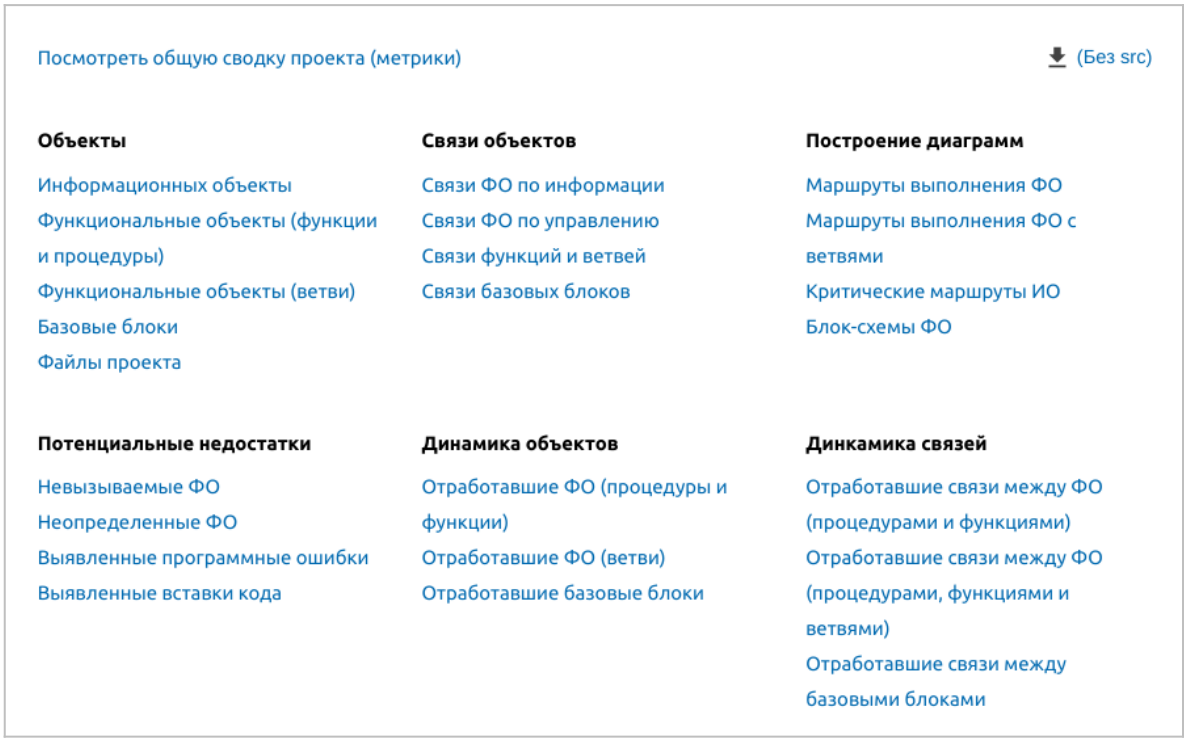

Рисунок 83 – Отчеты

Чтобы был доступен просмотр отчетов с помощью навигации отчетов, после статического анализа следует скачать архив с проектом, нажав кнопку скачивания в правом верхнем углу меню проекта. (рис. Рисунок 84). После скачивания сразу откроется директория проекта.

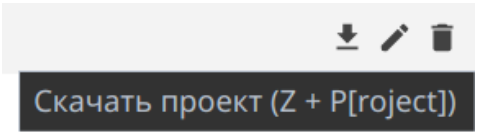

Рисунок 84 – Кнопка скачивания проекта

Если с момента последнего скачивания не был запущен анализ, появится уведомление с предупреждением об этом (рис. Рисунок 85).

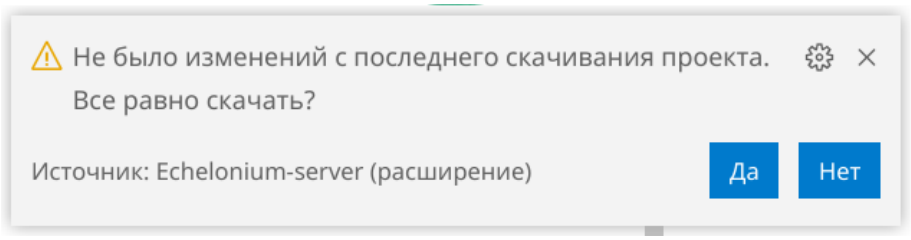

Рисунок 85 – Предупреждение

Если при открытом проекте нажать на ссылку в отчете, то соответствующее место откроется в Эшелониуме (рис. Рисунок 86).

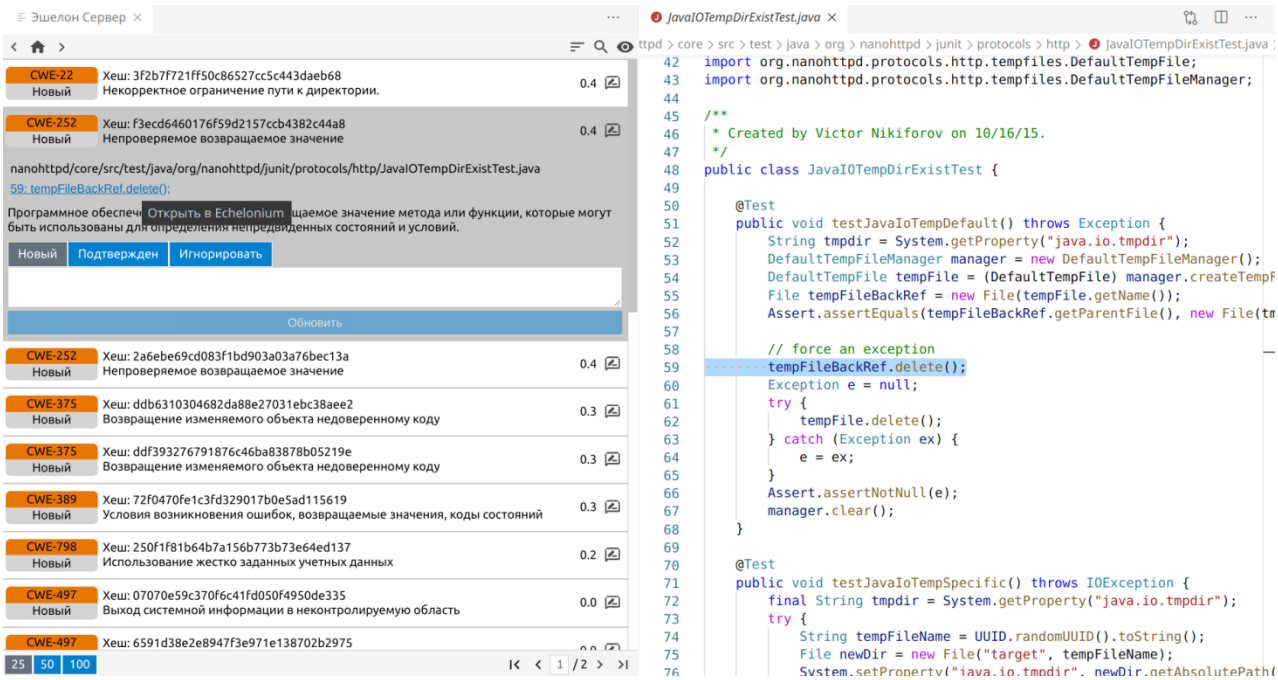

Рисунок 86 – Открытие недостатка

В правом левом углу доступны сортировка, фильтр по отчету и слежка за открываемыми файлами. При включенной слежки при открытии файла будет происходить фильтрация по пути файла. Данный функционал также доступен, если открыть «Ошибки проекта» в проводнике (рис.Рисунок 87).

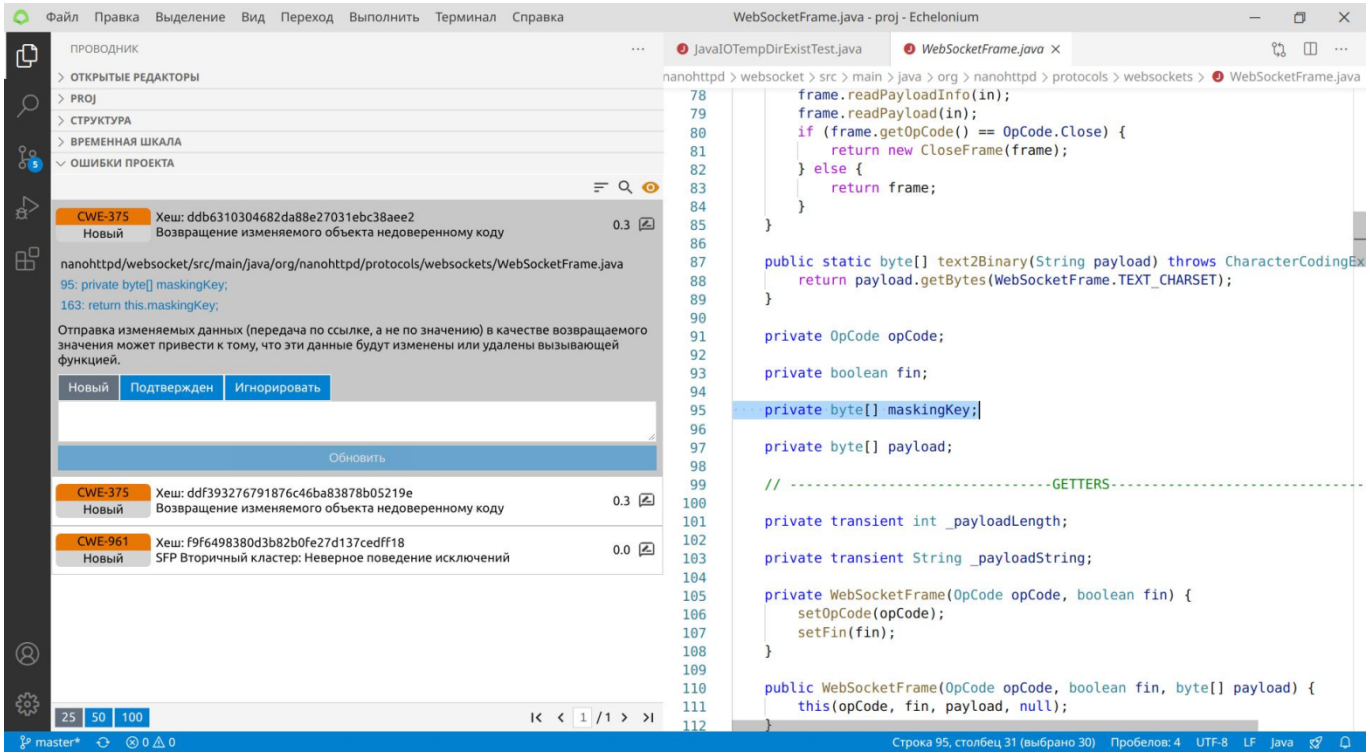

Рисунок 87 – Меню ошибок в проводнике

Приложение 1. Настройка сервера через конфигурационный файл DEFAULT.INI

При необходимости оператор может изменить настройки сервера «АК-ВС 3» через конфигурационный файл «default.ini», расположенный в директории «config». Для этого необходимо изменить настройки по умолчанию.

Пример файла конфигурации представлен ниже.

```
[rest]
#host=127.0.0.1
#port=11000
[analyzer]
#javaopts=-Xmx4g
[https]
#enabled=true
#keystoreFile=/home/user/path/to/keystore/keystore.p12
#keystorePass=password
#keystoreType=PKCS12
#keystoreProvider=SunJSSE
#keystoreAlias=tomcat
[mongo]
#login=dbuser
#password=password
#dbName=akvs3
#server=127.0.0.1
#port=27017
[options]
projectDirType=name
#maxAnalyzeTime=10000
#maxUploadChunkTime=60000
#hangUpCheckIntervalMinute=30
```

1.1. Описание параметров файла конфигурации сервера

Группа параметров [rest] описывает параметры работы веб-интерфейса (таблица 1.1).

Таблица 1.1 — Группа параметров [rest]

Параметр	Описание	Значение по умолчанию
host	Адрес сервиса	127.0.0.1
port	Порт сервиса	11000

Группа параметров [analyzer] описывает опции виртуальной машины Java при запуске анализатора (таблица 1.2).

Таблица 1.2 — Группа параметров [analyzer]

Параметр	Описание	Значение по умолчанию
javaopts	Дополнительные параметры	Отсутствует

Группа параметров [https] описывает настройки https-протокола (таблица 1.3).

Таблица 1.3 — Группа параметров [https]

Параметр	Описание	Значение по умолчанию
enabled	Дополнительные параметры	false
keystoreFile	Абсолютный путь к файлу-ключу	Отсутствует
keystorePass	Пароль	Отсутствует
keystoreType	Тип файла-ключа	Отсутствует
keystoreProvider	Наименование провайдера	Отсутствует
keystoreAlias	Алиас сервера	Отсутствует

Группа параметров [mongo] описывает настройки доступа к базе данных (таблица 1.4).

Таблица 1.4 — Описание параметров настройки доступа к базе данных

Параметр	Описание	Значение по умолчанию
login	Логин. Если данный параметр отсутствует, то авторизация не используется	Отсутствует
password	Пароль	Отсутствует
dbName	Имя базы данных	akvs3

Параметр	Описание	Значение по умолчанию
server	IP-адрес сервера	127.0.0.1
port	Порт сервера	27017

Группа прочих параметров [options] представлены в таблице 1.5.

Таблица 1.5 — Прочие параметры

Параметр	Описание	Значение по умолчанию
projectDirType	Формат названия директории с проектом. Возможные значения – name или id	name
maxAnalyzeTime	Максимальное время анализа проекта (в мс). 0 – без ограничений	0
maxUploadChunkTime	Максимальное время ожидания следующей части архива при его загрузке (в мс)	60000
hangUpCheckIntervalMinute	Интервал проверки логов анализа, влияет на чувствительность обнаружения предположительного зависания	30

Приложение 2. Инструкция по обновлению Лицензии

Обновление лицензии происходит в два этапа. На первом этапе формируется запрос файла лицензии. На втором этапе импортируется полученный файл.

2.1. Обновление лицензии в среде под управлением операционной системы семейства Linux

Для обновления лицензии «АК-ВС 3» в среде под управлением операционной системы семейства Linux необходимо сформировать файл запроса. Для этого выполните команду:

```
cmu --context 102455 --file1.WibuCmRaC
```

Отправьте сформированный файл по электронной почте в АО «Эшелон Технологии».

После получения файла лицензии загрузите полученный файл с помощью команды:

```
cmu --import –file имя_файла
```

После завершения всех действий лицензия будет обновлена.

Приложение 3. Модуль консольного клиента

Модуль консольного клиента REST-сервиса «АК-ВС 3» представляет собой jar-файл и позволяет в консольном режиме подключаться к веб-сервису «АК-ВС 3» и проводить статический и динамический анализы. Основные возможности модуля: проведение статического и динамического анализов, получение информации о проектах и лицензии.

3.1. Работа с модулем

Для запуска модуля введите в командной строке:

```
java -jar akvs3-rest-client.jar [static|dynamic|project] [OPTIONS]
```

Данный модуль поддерживает 3 режима работы: static, dynamic, project.

3.1.1. Режим static

В режиме static производится статический анализ. В директории вывода будут помещены отчеты статического анализа, исходные коды с датчиками, сами датчики. По желанию могут быть помещены данные проекта.

Для запуска режима необходимо использовать команду:

```
java -jar akvs3-rest-client.jar static ОПЦИИ
```

3.1.1.1. Опции модуля, имеющиеся в режиме static:

- -i path | --input path — путь до исходных текстов (директория или tre-файл, или архив в формате ZIP);
- -o directory | --output directory — путь до директории, куда будут помещены отчеты;
- -n project | --name project — необязательный параметр, название проекта (по умолчанию «Project»);
- -l level | --level level — необязательный параметр, уровень контроля НДВ (по умолчанию «1»);
- -s server | --server server — необязательный параметр, адрес сервера (по умолчанию «localhost»);
- -p port | --port port — необязательный параметр, порт (по умолчанию «11000»);
- -u user | --user user — необязательный параметр, имя пользователя «АК-ВС 3» (по умолчанию «admin»);

- -p password | --password password — необязательный параметр, пароль пользователя «АК-ВС 3» (по умолчанию «admin»);
- -pr | --insert-probes — необязательный параметр, вставлять датчики для динамического анализа;
- -to timeout | --timeout timeout — необязательный параметр, тайм-аут в секундах (по умолчанию «30»);
- -fma | -- force-mkcha-achkpv — необязательный параметр, включить МКЧА и АЧКПВ (по умолчанию включен и неизменяем для уровней 1 и 2);
- -fc | --insert-code — необязательный параметр, вставлять код в блок-схемы вместо названий объектов;
- -e | --add-to-existing — необязательный параметр, объединить старый и загружаемый код;
- -dd | --download-data — необязательный параметр, загрузить исходники, данные проекта (включая отчеты) и журналы;
- -rt type | --report_type type — необязательный параметр, выбрать тип скачиваемых отчетов (ws – с исходниками, ns – без исходников, all – все, по умолчанию – ws);
- -v | --verbose — необязательный параметр, подробный вывод программы.

Пример команды для запуска модуля в режиме static:

```
java -jar akvs3-rest-client.jar static -i d:/my_project/src.zip -o d:/my_project/report -l 2 --insert-probes
```

3.1.2. Режим dynamic

В режиме dynamic производится динамический анализ. В директорию вывода будет помещен отчет о проведении динамического анализа. Для запуска режима необходимо использовать команду:

```
java -jar akvs3-rest-client.jar dynamic ОПЦИИ
```

3.1.2.1. Опции модуля в режиме dynamic:

- -i path | --input path — путь до логов отработки датчиков (директория или файл-лог, или архив в формате ZIP);
- -o directory | --output directory — путь до директории с отчетами статического анализа;
- -s server | --server server — необязательный параметр, адрес сервера (по умолчанию «localhost»);
- -p port | --port port — необязательный параметр, порт (по умолчанию «11000»);

- -n project | --name project — необязательный параметр, название проекта (по умолчанию «Project»);
- -u user | --user user — необязательный параметр, имя пользователя «АК-ВС 3» (по умолчанию «admin»);
- -pw password | --password password — необязательный параметр, пароль пользователя «АК-ВС 3» (по умолчанию «admin»);
- -to timeout | --timeout timeout — необязательный параметр, тайм-аут в секундах (по умолчанию «30»).

Пример команды для запуска модуля в режиме dynamic:

```
java -jar akvs3-rest-client.jar dynamic -i /home/user/akvs/dyn_logs.zip -o /home/user/akvs/report/ --user test --password test -s http://192.168.0.1
```

3.1.3. Режим project

Режим project позволяет просматривать информацию о проектах в системе. Для запуска режима необходимо использовать команду:

```
java -jar akvs3-rest-client.jar project -a действие ОПЦИИ
```

Действие «list» или «l» служит для показа списка проектов, действие «delete» или «d» служит для удаления конкретного проекта.

3.1.3.1. Опции модуля в режиме project:

- -s server | --server server — необязательный параметр, адрес сервера (по умолчанию «localhost»);
- -p port | --port port — необязательный параметр, порт (по умолчанию «11000»);
- -u user | --user user — необязательный параметр, имя пользователя «АК-ВС 3» (по умолчанию «admin»);
- -pw password | --password password — необязательный, пароль пользователя «АК-ВС 3» (по умолчанию «admin»);
- -to timeout | --timeout timeout — необязательный, тайм-аут в секундах (по умолчанию «30»);
- -n project | --name project — необязательный параметр, название проекта (по умолчанию «Project»).

Примеры команд для запуска модуля в режиме project:

```
java -jar akvs3-rest-clien.jar project -a list
java -jar akvs3-rest-clien.jar project -a list --user test --password
test -s http://192.168.0.1
java -jar akvs3-rest-clien.jar project -a delete -n 42 -s
http://192.168.0.1
```

ПЕРЕЧЕНЬСОКРАЩЕНИЙ

Сокращение	Расшифровка
REST	(англ. *Representational State Transfer* – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределённого приложения в сети
XML	(англ. *eXtensible Markup Language*) - расширяемый язык разметки
АЧКПВ	Анализ, чувствительный к путям выполнения
МКЧА	Межпроцедурный контекстно-чувствительный анализ
ПО	Программное обеспечение
ФО	Функциональный объект
ЭВМ	Электронная вычислительная машина (компьютер)

Лист регистрации изменений										
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводит. документа и дата	Подпись	Дата