**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**УНИВЕРСИТЕТ ИТМО**

**А.Н. Бегаев, С.Н. Бегаев, С.В. Кашин**

# АНАЛИЗ ПРОГРАММНОГО КОДА ПРИ ПРОВЕДЕНИИ СЕРТИФИКАЦИОННЫХ ИСПЫТАНИЙ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки (специальности) 10.03.01 «Информационная безопасность»
в качестве учебного пособия для реализации основных профессиональных
образовательных программ высшего образования бакалавриата

УНИВЕРСИТЕТ ИТМО

**Санкт-Петербург**

**2018**

Бегаев А.Н., С.Н. Бегаев, Кашин С.В. Анализ программного кода при проведении сертификационных испытаний. – СПб: Университет ИТМО, 2018. – 41 с.

**Рецензент:** Кременчуцкий Александр Лазаревич, профессор, к.т.н.

Учебно-методическое пособие содержит теоретический материал, посвященный основным вопросам и порядку проведения сертификационных испытаний программного обеспечения, законодательной базе сертификации программного обеспечения. Пособие содержит общие знания о существующих анализаторах исходных текстов программ, подробные сведения о составе функций «Анализатора исходных текстов «АК-ВС 2» и стадии их выполнения.

Учебное пособие предназначено для студентов, обучающихся по направлению подготовки 10.03.01 «Информационная безопасность» по дисциплине «Технология сертификации средств защиты информации».

Рекомендовано к печати Советом мегафакультета КТиУ (протокол №2 от 14 февраля 2018 года).

УНИВЕРСИТЕТ ИТМО

**Университет ИТМО** – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО является участником программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО заключается в становлении исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

# ОГЛАВЛЕНИЕ

## ВВЕДЕНИЕ

Данное учебно-методическое пособие содержит материалы, которые позволят студентам закрепить теоретические знания о порядке проведения процедуры сертификации программного обеспечения в ходе изучения дисциплины «Технология сертификации средств защиты информации» и получить практические навыки работы со средством автоматизации проведения анализа исполняемого кода и исходных текстов программ, проводимого в рамках сертификационных испытаний, «АК-ВС 2».

Структурно пособие состоит из четырех разделов, три из которых теоретические - предназначены для подготовки к выполнению лабораторной работы и закреплению материала, один раздел – практический, содержит инструкцию по выполнению лабораторной работы и задание на лабораторную работу. Пособие содержит контрольные вопросы, призванные помочь студентам в освоении результатов обучения по дисциплине «Технология сертификации средств защиты информации».

Дополнительно приведен библиографический список (список рекомендуемой литературы), который включает руководящие документы и литературу, рекомендуемую авторами для более глубокого освоения содержания дисциплины.

# 1 ОСНОВЫ СЕРТИФИКАЦИИ

Без программного обеспечения, которое значительно расширяет функциональные возможности и повышает оперативность обработки и использования информации, невозможно представить ни одно современное средство. Однако в ряде случаев применение программного обеспечения – это риски, которые вызваны внутренними свойствами программного обеспечения и возможностью внешнего влияния на них. Вследствие этого возникает необходимость проведения объективной независимой (в т.ч. метрологической) проверки программного обеспечения. Такая проверка может проходить, к примеру, в виде *сертификации программного обеспечения.* Сертификация программного обеспечения – это процедура установления соответствия программного обеспечения требованиям нормативных документов.

Защита пользовательских интересов, интересов государства и ведомств, заключающаяся в контроле качества продукции и гарантирования высоких потребительских свойств – это основная цель сертификации программных продуктов.

## 1.1 Цели и виды сертификации

Процесс подготовки и принятия решения о целесообразности выдачи сертификата, а также само получение сертификата соответствия является формальной целью сертификации. Рациональность или целесообразность определяется с учетом следующих факторов:

1) достоверность, полнота, точность исходного технического задания (ТЗ) и спецификации требований, предоставленных на программное средство (ПС) и на технологию поддержки его жизненного цикла (ЖЦ);
2) точность измерения, обобщение результатов и достоверность сертификационных испытаний;
3) качество и методология интерпретации данных о программном средстве, предоставленном на испытания с учетом достоверных оценок, квалификации испытателей и их объективности, а также объективности заказчиков и пользователей.

Общие цели сертификации:

1) определение соответствия или несоответствия установленным требованиям действующих стандартов;
2) определение эффективности программной продукции с точки зрения соответствия поставленным в организации целям для обеспечения качества;

3) обнаружение слабых (узких) мест в сертифицируемых продуктах, которые отрицательно влияют на их показатели;
4) предотвращение финансовых убытков в виде штрафов за дефектное качество продукции;
5) возможность получения организацией дополнительных маркетинговых преимуществ.

Сертификация бывает обязательной и добровольной. Такая классификация обусловлена областью применения информационной системы, в которой используется ПС, а также назначением и классом самого программного средства.

Для программных средств и информационных систем, которые осуществляют особо критические функции, и в которых дефектное качество и ошибки могут нанести большой ущерб для жизни и здоровья людей, необходима о*бязательная сертификация*.

К системам обязательной сертификации относятся: системы Федеральной службы безопасности РФ, Федеральной службы по техническому и экспортному контролю РФ, Министерства обороны РФ.

В том случае, когда процедура установления соответствия качества программного средства проводится с целью повышения конкурентоспособности или расширения сферы применения или же с целью получения дополнительных экономических преимуществ, применяется *добровольная сертификация*.

Экономические цели сертификации – увеличение прибыли разработчиков и поставщиков программных средств, уменьшение количества претензий от пользователей, увеличенная тиражность продукции при ее производстве, большая длительность жизненного цикла.

## 1.2 Исходные документы и участники сертификации

Исходные документы для сертификации:

1) спецификация требований и/или технические условия, а также комплект эксплуатационной документации на программное средство, его компоненты и на систему обеспечения их качества;
2) действующие ведомственные, государственные и международные стандарты на разработку и испытания программных комплексов и на техническую документацию;
3) программа испытаний по всем требованиям ТЗ и положениям эксплуатационной документации;
4) методики испытаний по каждому разделу требований ТЗ и документации.

*Сертификат соответствия* является результатом положительных сертификационных испытаний. Сертификат соответствия – это документ,

который удостоверяет соответствие предъявленных заявителем продуктов установленным требованиям и который издан в соответствии с правилами Системы сертификации. Срок действия сертификата ограничен и зависит от системы сертификации. Действие сертификата может прекратиться досрочно по мотивированному решению органа, выдавшего сертификат, или может быть продлено по результатам инспекционного контроля в случае, если программный продукт не был подвергнут значительным изменениям.

Сертификационные испытания программных средств, в том числе программных средств, предназначенных для защиты информации, проводятся в государственных и отраслевых сертификационных центрах (испытательных лабораториях).

Участники сертификации:

1) Федеральный орган по сертификации;
2) орган сертификации – орган, ответственный за контроль процесса сертификации продукции (в некоторых системах сертификации отсутствует, его обязанности выполняет федеральный орган по сертификации);
3) испытательные лаборатории – ответственные за проведение сертификационных испытаний определенного вида продукции.
4) заявители – это те участники процесса сертификации, которые хотят получить сертификаты соответствия на продукцию. В роли заявителей могут выступать ретейлеры продукции и ее исполнители.

В России действуют 4 Федеральных органа по сертификации - ФСБ России, ФСТЭК России, Министерство обороны РФ, Росстандарт.

Федеральный орган по сертификации:

1) создает систему сертификации;
2) определяет метод подтверждения соответствия СЗИ;
3) формирует перечень СЗИ, для которых необходима сертификация;
4) устанавливает правила аккредитации испытательных лабораторий, органов по сертификации СЗИ, а также центральных органов систем сертификации и проводит соответствующие аккредитации;
5) выдает сертификаты и лицензии на применение знака соответствия;
6) ведет реестр участников сертификации и сертифицированных средств;
7) осуществляет надзор и контроль за сертифицированными СЗИ, а также за соблюдением правил сертификации участниками сертификации;
8) рассматривает апелляции по вопросам сертификации;
9) публикует информацию о сертификации;
10) организовывает аттестацию и подготовку аудиторов-экспертов;

11) занимается отменой, продлением и приостановлением действия ранее выданных сертификатов.

Функции органа сертификации:

1) представление заявителю необходимой информации по сертификации;
2) актуализация и формирование фонда методических и нормативных документов, которые являются необходимыми для сертификации, участие в разработке этих документов;
3) ходатайство об отмене или приостановлении действия ранее выданных сертификатов перед федеральным органом по сертификации;
4) хранение оригиналов подтверждающей сертификацию средств защиты документации;
5) участие в инспекционном контроле за деятельностью испытательных центров (лабораторий) и организация инспекционного контроля за стабильностью характеристик сертифицированных средств защиты информации;
6) участие в аккредитации органов по аттестации объектов информатизации и испытательных лабораторий;
7) организация предварительных проверок и аттестации производства сертифицируемых средств защиты информации;
8) оформление экспертного заключения по сертификации средств защиты информации и предоставление его в федеральный орган по сертификации;
9) проведение экспертиз материалов сертификационных испытаний, а также экспертиза эксплуатационной и технической документации на средства защиты информации;
10) утверждение методики и программы проведения сертификационных испытаний;
11) рекомендация испытательных лабораторий заявителю;
12) участие в установлении схемы проведения сертификации средств защиты информации с учетом предложений заявителя;
13) уточнение требований, на соответствие которым проводятся сертификационные испытания.

Функции испытательных центров (лабораторий):

1) участие в аттестации производства сертифицируемых средств защиты информации;
2) маркировка сертифицированных СЗИ знаком соответствия;
3) разработка программы и методики сертификационных испытаний, осуществление сертификационных испытаний СЗИ, оформление протоколов сертификационных испытаний и технических заключений;

4) для проведения сертификационных испытаний производят отбор образцов СЗИ.

Испытательные центры (лаборатории) ответственны за точность измерений, их достоверность, объективность, полноту испытаний средств защиты информации, и своевременную, требуемую проверку средств измерений и аттестацию испытательного оборудования.

Для определения способности проведения работ по сертификации органами сертификации средств защиты информации и испытательными лабораториями, федеральный орган по сертификации проверяет их и выдает разрешение на право проведения таких работ, эта процедура называется аккредитацией.

На базе испытательных лабораторий, прошедших процедуру аккредитации, проводится сертификация. При должном контроле со стороны органа сертификации в ряде случаев возможно проведение испытаний на базе заявителя.

Сертификация программного средства включает в себя следующие этапы:

1) Заказчик или разработчик анализирует рынок и выбирает орган и лабораторию, компетентные для выполнения сертификационных испытаний.
2) Заявитель подает заявку на испытания в орган сертификации.
3) Федеральный орган по сертификации принимает решение по данной заявке, выбор лаборатории и схемы сертификации.
4) Заявителем определяется и фиксируется версии программного обеспечения, подлежащего испытаниям.
5) Испытательная лаборатория проводит процесс сертификационных испытаний программного обеспечения.
6) Орган сертификации анализирует полученные результаты и составляет заключение о прохождении сертификационных испытаний.
7) Федеральный орган по сертификации выдает сертификат и лицензию на применение знака соответствия и выпуск сертифицированной продукции заявителю.
8) Испытательная лаборатория осуществляет инспекционный контроль за сертифицированной продукцией.
9) Заявитель проводит регистрацию и публикацию информации о результатах сертификации продукции.

Заявитель предоставляет на сертификацию следующие документы:

- комплект технической документации, которая включает техническое задание или спецификацию требований и эксплуатационную документацию на компоненты и само программное средство;
- программное средство;

- заявка на проведение сертификации;
- проект договора на процедуру сертификационных испытаний.

Документы органа сертификации:

- регистрационная карта на сертифицируемый объект;
- заключение, полученное по результатам рассмотрения органом заявки;
- задание на проведение сертификации и предъявляемые к ней требования;
- план проведения сертификационных испытаний;
- заключение, полученное по результатам проведения сертификационных испытаний;
- сертификат соответствия.

Документы испытательной лаборатории:

- характеристики программного средства;
- комплект технической документации на ПС;
- действующие ведомственные, государственные и международные стандарты на разработку и испытания ПС и на техническую документацию;
- программа проведения сертификационных испытаний, соответствующая требованиям технического задания и положениям документации;
- методика проведения сертификационных испытаний для каждого раздела требований технического задания и документации;
- методы испытаний и инструментальные средства;
- регистрационная карта сертификационных испытаний;
- протоколы сертификационных испытаний;
- предложение о выдаче сертификата и отчет о проведенных испытаниях.

# 2 СЕРТИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПО ТРЕБОВАНИЯМ БЕЗОПАСНОСТИ

В настоящее время сертификация по требованиям безопасности проводится для программных продуктов, имеющих встроенные СЗИ.

Под сертификацией программного обеспечения средств защиты информации (СЗИ) по требованиям безопасности информации (далее сертификация) понимается проверка (подтверждение) характеристик СЗИ на соответствие требованиям нормативных документов и государственных стандартов по защите информации, выпущенные уполномоченными федеральными органами: ФСТЭК России (Гостехкомиссией России), ФСБ России и Министерство обороны.

Сертификация ПО по требованиям безопасности проводится:

1. На соответствие реальных и декларируемых в документации функциональных возможностей (РДВ).
2. На отсутствие недекларированных возможностей (далее по тексту – НДВ).

## 2.1 Сертификация ПО на соответствие реальных и декларируемых в документации функциональных возможностей

Сертификационные испытания на соответствие реальных и декларируемых в документации функциональных возможностей проводится для установления факта соответствия или несоответствия реальных и декларируемых функциональных возможностей программного средства или иного изделия функциональным возможностям, которые указаны в документации на него. Испытания на соответствие РДВ не регулируются руководящими или нормативными документами.

Целью испытаний на соответствие РДВ является – доказательство пригодности и достаточности документации для эксплуатации программного средства или иного изделия.

Качественная характеристика, проверяемая в ходе испытаний на соответствие реальных и декларируемых в документации функциональных возможностей – соответствие предоставленного на сертификационные испытания программного средства или другого изделия его документации. Основные документы, которые рассматриваются в ходе проведения таких испытаний: эксплуатационные документы, указанные в ведомости эксплуатационной документации изделия и технические условия. Технические условия – это основной документ, который определяет функциональные возможности ПС или изделия. На основании технических условий эксперт испытательной лаборатории формирует перечень

декларируемых возможностей, на соответствие которым и будут проводиться проверочные работы.

Заключение о достаточности и корректности документации на изделие эксперт формирует на основании актов предварительных и государственных испытаний.

Для каждой функциональной возможности изделия, определенной в технических условиях, производится определение документов и их разделов, которые содержат описания, относящиеся к конкретной указанной возможности.

Если эксперту удалось сопоставить документы для всех возможностей, которые указаны в технических условиях, он формирует заключение о достаточности описания функциональных возможностей изделия.

Второй шаг проверки – формирование перечня функциональных возможностей изделия на основании сведений, содержащихся в эксплуатационной документации, и сопоставление полученного перечня функциональных возможностей с перечнем, сформированным на основании технических условий на изделие.

Если перечни, полученные в ходе проверки, являются идентичными, эксперт делает вывод о полноте документирования функциональных возможностей изделия.

Третий шаг проверки – эксперт проверяет на соответствие описаниям, которые приведены в эксплуатационной документации. В том случае, если описания являются достаточными и полными для их выполнения, реакция системы корректно описана, а сообщения, реально выдаваемые эксперту в процессе проверки, соответствуют их описанию в документации и т.п., эксперт формирует заключение о корректности документации на изделие.

Четвертый шаг проверки – оценка пользовательского интерфейса программного обеспечения изделия. Эксперт внимательно изучает интерфейс ПС и каждый его отдельный элемент сравнивает с набором эвристик. Вместе с этим на первом этапе эксперт формирует общее представление о программном средстве, выявляет базовые принципы взаимодействия с пользователем. На втором этапе эксперт концентрируется на отдельных элементах интерфейса. В процессе изучения интерфейса эксперт фиксирует свои комментарии и подтверждает их снимками экрана. Эксперт на основании этих комментариев формирует заключение о соответствии интерфейса ПС каждой заданной эвристики. Эксперт по результатам заключений формирует заключительную оценку пользовательского интерфейса. Формирование положительной оценки пользовательского интерфейса происходит в случае, если для всех эвристик были получены положительные заключения.

## 2.2 Сертификация ПО на отсутствие недекларированных возможностей

Целью контроля на отсутствие НДВ является формирование на основании анализа результатов проверочных действий заключения об отсутствии (наличии) недекларированных возможностей программного обеспечения, в том числе отсутствия (наличия) программных закладок. Список действий для проверки, которые необходимо выполнить в рамках соответствующего уровня контроля, определяется положениями руководящего документа Гостехкомиссии России «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей» (установлен Приказом Председателя Гостехкомиссии России № 114 от 04.06.1999 г., далее по тексту – РД НДВ).

РД НДВ устанавливает четырёхуровневый контроль программного обеспечения, который отличается условиями, объёмом и глубиной проведения испытаний. Многоуровневый контроль ПО регламентирует степень конфиденциальности информации. Разные уровни контроля позволяют классифицировать степень вероятности отсутствия в исследуемом ПО недекларированных возможностей.

Четвёртый уровень – самый низкий уровень контроля. Этот уровень предназначен для проверки программного обеспечения, которое используется при защите (обработке, хранении, передаче и т.п.) конфиденциальной информации. Для программного обеспечения, которое используется для защиты информации, отнесенной к государственной тайне, во время проведения испытаний необходим третий уровень контроля и выше. Причем, проведение испытаний по соответствующим уровням контроля отсутствия НДВ обязательно и соответствует требованиям отечественного законодательства для программных средств, которые предназначены для обработки информации, составляющей государственную тайну.

В соответствии с пунктом 2.1 РД НДВ под недекларированными возможностями понимаются функциональные возможности программного обеспечения (далее по тексту – ПО), не указанные или не соответствующие указанным в документации, использование которых может повлечь нарушение целостности, конфиденциальности и доступности обрабатываемой информации. Программные закладки (в частности) – реализация недекларированных возможностей.

В соответствии с пунктом 2.2 РД НДВ под программными закладками понимаются намеренно интегрированные в ПО функциональные объекты, инициирующие (при определённых условиях) выполнение не указанных в документации функций ПО, которые приводят к нарушению целостности, конфиденциальности или доступности обрабатываемой информации.

Контроль отсутствия недекларированных возможностей программного обеспечения предполагает глубокое исследование программного обеспечения и связан с анализом как исполняемого кода, так и исходных текстов программ. Общие принципы анализа программ с учетом аспектов, которые связаны с информационной безопасностью, являются основой таких исследований.

Качественные характеристики, оцениваемые при выполнении испытаний, в соответствии с РД НДВ:

1. Контроль состава и содержания документации – экспертиза документации на соответствие требованиям, предъявляемым к содержанию и составу документации.

Контроль содержания и состава документации подразумевает, что на ПО сертифицируемого изделия выпущена следующая документация: спецификация в соответствии с ГОСТ19.202-78, описание программы в соответствии с ГОСТ 19.402-78, описание применения в соответствии с ГОСТ 19.502-78, исходные тексты программ в соответствии с ГОСТ 19.401-78, пояснительная записка в соответствии с ГОСТ 19.404-79.

2. Контроль исходного состояния ПО - расчет контрольных сумм и фиксация размеров файлов исходных текстов и дистрибутивов компонентов ПО, и сравнение полученных результатов с приведенными в документации.

Контроль исходного состояния ПО подразумевает, что при идентификации объекта сертификации, заключающейся в расчете указанным разработчиком способом (технологическим средством) контрольных сумм модулей исполняемого кода и исходных текстов программного средства, полученные данные расчетов при последующем сравнении совпадут с данными, приведенными в документации на сертифицируемое программное средство.

3. Статический анализ исходных текстов программ - это анализ программного обеспечения, выполняемый без фактического выполнения программ. В большинстве случаев анализ выполняется над какой-либо версией исходного кода, в других случаях анализу подвергается какой-нибудь вид объектного кода.

Статистический анализ исходных текстов программ включает следующие технологические операции:

- контроль полноты и отсутствия избыточности исходных текстов ПО на уровне файлов – подразумевает, что подконтрольное программное средство не содержит в своем составе программных модулей (файлов), явно неиспользуемых другими программными модулями;
- контроль соответствия исходных текстов ПО его объектному (загрузочному) коду – подразумевает мониторинг сборки исполняемого кода из исходного с последующим побайтным

сравнением результата сборки с исполняемым кодом, представленным заявителем для осуществления испытаний;

- контроль отсутствия избыточности и полноты исходных текстов и ПО на уровне функциональных объектов (процедур) – подразумевает отсутствие в составе исходных текстов подконтрольного программного обеспечения функциональных объектов (процедур, функций, задач и защищенных объектов), явно неиспользуемых множеством исполнений подконтрольного программного обеспечения;
- контроль связей функциональных объектов (модулей, процедур, функций) по управлению – подразумевает построение взаимосвязей указанных объектов контролируемого программного обеспечения по управлению и сравнение построенных взаимосвязей с взаимосвязями, описанными в программной документации;
- контроль связей функциональных объектов (модулей, процедур, функций) по информации – подразумевает построение взаимосвязей указанных объектов контролируемого программного обеспечения по информации и сравнение построенных взаимосвязей с взаимосвязями, описанными в программной документации;
- контроль информационных объектов различных типов (например, локальных переменных, глобальных переменных, внешних переменных и т.п.) – подразумевает выявление избыточности программного обеспечения на уровне переменных (локальных, глобальных), а также контроль использования объявленных переменных;
- формирование списка маршрутов осуществления функциональных объектов (процедур, функций) – подразумевает формирование перечня всех возможных маршрутов выполнения контролируемого программного обеспечения для последующего сравнения с наблюдаемым при динамическом анализе маршрутом выполнения контролируемого программного обеспечения;
- контроль отсутствия избыточности и полноты исходных текстов подконтрольного ПО на уровне функциональных объектов (функций);
- синтаксический контроль присутствия заданных конструкций из списка (базы) потенциально опасных программных конструкций в исходных текстах ПО;
- составление списка маршрутов выполнения функциональных объектов (ветвей);

- анализ критических маршрутов выполнения функциональных объектов (процедур, функций) для заданных перечней информационных объектов экспертом – подразумевает определение маршрутов выполнения функциональных объектов подконтрольного программного обеспечения, зависящих от состояния некоторых информационных объектов, а также маршрутов выполнения, производящих изменения указанных (определенных экспертом) информационных объектов. При этом, непосредственно под критическим маршрутом выполнения функциональных объектов понимается такой маршрут, который влечёт за собой возможность неконтролируемого нарушения установленных правил обработки информационных объектов при его выполнении;
- построение для подконтрольного ПО блок-схем, диаграмм и т.п. по исходным текстам, сравнительный анализ алгоритма работы, приведенного в “Пояснительной записке” и алгоритма работы функциональных объектов (процедур, функций);
- семантический контроль присутствия заданных конструкций из списка (базы) потенциально опасных конструкций в исходных текстах ПО.

4. Динамический анализ исходных текстов программ – это анализ программного обеспечения, выполняемый путем исполнения программ на реальном или виртуальном процессоре.

Динамический анализ исходных текстов программ должен включать следующие технологические операции:

- Подконтрольное выполнение функциональных объектов (ветвей, функций, процедур);
- Сравнение маршрутов, построенных в процессе проведения статического анализа и фактических маршрутов выполнения функциональных объектов (ветвей, функций, процедур).

Статический анализ производится с помощью специального программного обеспечения – статических анализаторов кода. Существует большое количество коммерческих и бесплатных статических анализаторов кода, написанных на различных языках программирования (C/C++, Java, .NET, Python и т.д).

# 3 СРЕДСТВО ДЛЯ ПРОВЕДЕНИЯ ИСПЫТАНИЙ «АК-ВС 2»

## 3.1 Общая информация и основные функции «АК-ВС 2»

В данном учебно-методическом пособии рассматривается функционал программного обеспечения «Анализатор исходных текстов программ «АК-ВС 2». «АК-ВС 2» предназначен для анализа исходных текстов программ на языках C/C++, C#, Java и PHP и генерации отчетов тестовых испытаний.

«АК-ВС 2» должен реализовывать следующие основные функции:

а) анализ исходных текстов программ и программных комплексов;

б) формирование следующих отчетов анализа исходных текстов:

Отчеты по статическому анализу:

1. Отчёт по метрикам;
2. Список файлов проекта;
3. Список информационных объектов (ИО);
4. Перечень функциональных объектов (ФО) (функций и процедур);
5. Перечень функциональных объектов (ветвей);
6. Перечень невызываемых ФО;
7. Перечень неопределенных ФО;
8. Таблица связей ФО по управлению;
9. Маршруты выполнения ФО;
10. Таблица связей ФО по информации;
11. Критические маршруты для выбранного ИО;
12. Таблица связей функций и ветвей;
13. Маршруты выполнения ФО с ветвями;
14. Блок-схемы ФО;
15. Отчет о поиске потенциально опасных конструкций.

Отчеты по динамическому анализу:

1. Отчёт по метрикам;
2. Отработавшие ФО (процедуры и функции);
3. Отработавшие связи между ФО (процедурами и функциями);
4. Отработавшие ФО (ветви);
5. Отработавшие связи между ФО (процедурами, функциями и
6. ветвями).

## 3.2 Работа с «АК-ВС 2»

*1. Добавление нового проекта*

Вкладка «Проекты» показывает все текущие проекты. Вкладка «Датчики» показывает имеющиеся датчики для динамического анализа. Вкладка «О программе» содержит адрес электронной почты технической поддержки продукта и информацию о лицензии.

1) во вкладке «Проекты» нажмите кнопку «Добавить» (Рисунок 2 показывает, как настроить новый проект);

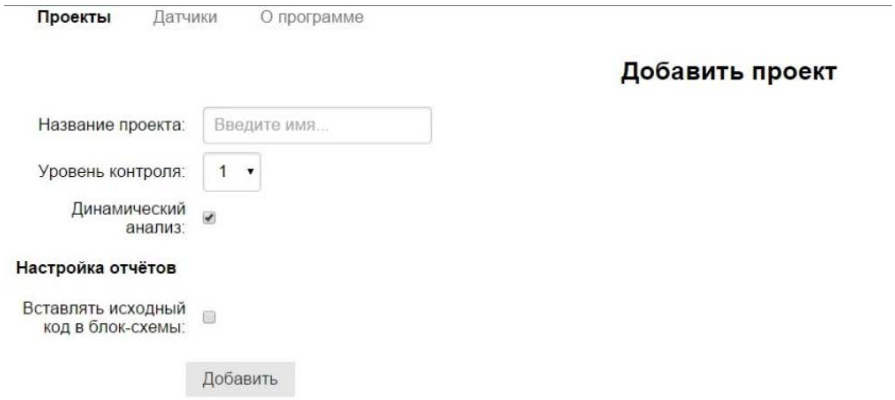



Рисунок 2 – Добавление нового проекта

2) в поле «Имя» введите имя проекта;
3) в поле «Уровень контроля» выберите уровень контроля в соответствии с РД «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей»;
4) включите опцию «Динамический анализ», если требуется выполнить его после проведения статического анализа;
5) включите опцию «Вставлять исходный код в бок-схемы», если требуется, чтобы исходный код был вставлен в блок-схемы при составлении отчетов;
6) нажмите «Добавить».

*2. Статический анализ*

1) для проведения статического анализа с помощью «АК-ВС 2» необходимо загрузить в проект исходные тексты, которые будут анализироваться (исходные тексты должны находиться в архиве с расширением .zip);
2) выберите конфигурацию парсера, то есть программной части, выполняющей синтаксический анализ. Конфигурация парсера должна находиться в архиве с расширением .zip. Также можно пропустить этап загрузки конфигурации парсера. Для этого нажмите на кнопку «Пропустить этап конфигурации».
3) Дождитесь окончания статического анализа. После этого на странице проекта появятся разделы «Отчеты» и «Журналы». В разделе «Отчеты» нажмите кнопку «Отчеты по статическому анализу». После этого откроется страница с отчетами по статическому анализу как на Рисунке 3.

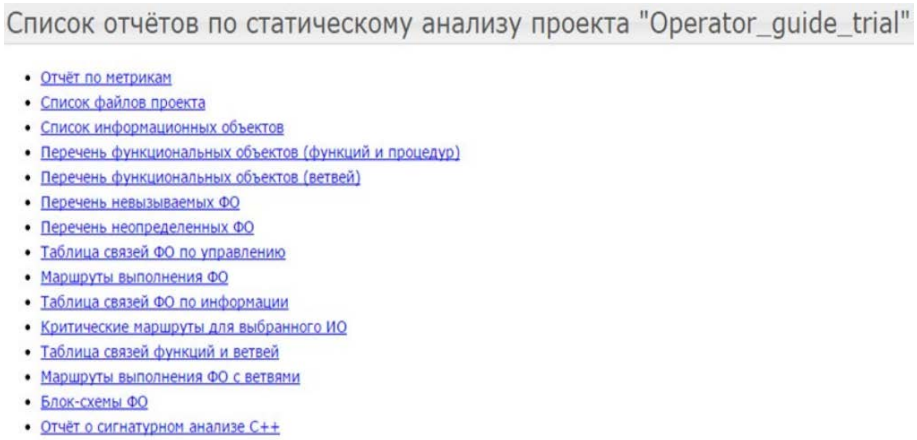



Рисунок 3 - Страница с отчетами по статическому анализу

На рисунке 4 представлен сформированный анализатором список информационных объектов, что соответствует одному из пунктов статического анализа согласно РД НДВ.

**Operator_guide_trial: список информационных объектов**

| № | QID | Название | Расположение |
|---|---|---|---|
| 1 | 0:0 | caught | home/denis/memcached/timedrun.c:11 |
| 2 | 0:2 | caught_signal::which | home/denis/memcached/timedrun.c:13 |
| 3 | 0:3 | caught_signal::_akvs_probe_1_207 | home/denis/memcached/timedrun.c:14 |
| 4 | 0:5 | wait_for_process::pid | home/denis/memcached/timedrun.c:20 |
| 5 | 0:6 | wait_for_process::_akvs_probe_3_271 | home/denis/memcached/timedrun.c:21 |
| 6 | 0:7 | wait_for_process::rv | home/denis/memcached/timedrun.c:23 |
| 7 | 0:8 | wait_for_process::stats | home/denis/memcached/timedrun.c:24 |
| 8 | 0:9 | wait_for_process::i | home/denis/memcached/timedrun.c:25 |
| 9 | 0:10 | wait_for_process::sig_handler | home/denis/memcached/timedrun.c:26 |
| 10 | 0:12 | wait_for_process::_akvs_probe_9_739 | home/denis/memcached/timedrun.c:38 |
| 11 | 0:13 | wait_for_process::p | home/denis/memcached/timedrun.c:40 |
| 12 | 0:15 | wait_for_process::_akvs_probe_11_806 | home/denis/memcached/timedrun.c:41 |
| 13 | 0:16 | wait_for_process::FI::__in | home/denis/memcached/timedrun.c:44 |
| 14 | 0:17 | wait_for_process::FI::__i | home/denis/memcached/timedrun.c:44 |
| 15 | 0:18 | wait_for_process::FI::__in | home/denis/memcached/timedrun.c:45 |
| 16 | 0:19 | wait_for_process::FI::__i | home/denis/memcached/timedrun.c:45 |
| 17 | 0:20 | wait_for_process::FI::__in | home/denis/memcached/timedrun.c:46 |
| 18 | 0:21 | wait_for_process::FI::__i | home/denis/memcached/timedrun.c:46 |
| 19 | 0:23 | wait_for_process::_akvs_probe_18_1012 | home/denis/memcached/timedrun.c:50 |
| 20 | 0:24 | wait_for_process::sig | home/denis/memcached/timedrun.c:52 |
| 21 | 0:26 | wait_for_process::_akvs_probe_20_1169 | home/denis/memcached/timedrun.c:56 |
| 22 | 0:28 | wait_for_process::_akvs_probe_21_1246 | home/denis/memcached/timedrun.c:58 |
| 23 | 0:30 | wait_for_process::_akvs_probe_22_1395 | home/denis/memcached/timedrun.c:67 |
| 24 | 0:33 | wait_for_process::_akvs_probe_24_1559 | home/denis/memcached/timedrun.c:77 |

Рисунок 4 - Список информационных объектов

Также в соответствии с требованиями РД НДВ «АК-ВС 2» формирует перечни функциональных объектов (функций, процедур, ветвей) (Рисунок 5, Рисунок 6).

**Operator_guide_trial: перечень функциональных объектов (функций и процедур)**

| № | QID | Название | Расположение |
|---|---|---|---|
| 1 | 0:1 | caught_signal | home/denis/memcached/timedrun.c:13 |
| 2 | 0:4 | wait_for_process | home/denis/memcached/timedrun.c:20 |
| 3 | 0:34 | spawn_and_wait | home/denis/memcached/timedrun.c:92 |
| 4 | 0:44 | main | home/denis/memcached/timedrun.c:120 |
| 5 | 1:4 | cache_create_test | home/denis/memcached/testapp.c:36 |
| 6 | 1:7 | cache_constructor | home/denis/memcached/testapp.c:48 |
| 7 | 1:13 | cache_constructor_test | home/denis/memcached/testapp.c:55 |
| 8 | 1:17 | cache_fail_constructor | home/denis/memcached/testapp.c:68 |
| 9 | 1:22 | cache_fail_constructor_test | home/denis/memcached/testapp.c:73 |
| 10 | 1:29 | cache_destructor | home/denis/memcached/testapp.c:93 |
| 11 | 1:33 | cache_destructor_test | home/denis/memcached/testapp.c:99 |
| 12 | 1:36 | cache_reuse_test | home/denis/memcached/testapp.c:112 |
| 13 | 1:43 | cache_bulkalloc | home/denis/memcached/testapp.c:132 |
| 14 | 1:53 | test_issue_161 | home/denis/memcached/testapp.c:159 |
| 15 | 1:58 | cache_redzone_test | home/denis/memcached/testapp.c:172 |
| 16 | 1:64 | test_safe_strtoul | home/denis/memcached/testapp.c:208 |
| 17 | 1:67 | test_safe_strtoull | home/denis/memcached/testapp.c:233 |
| 18 | 1:70 | test_safe_strtoll | home/denis/memcached/testapp.c:253 |
| 19 | 1:73 | test_safe_strtol | home/denis/memcached/testapp.c:283 |
| 20 | 1:76 | start_server | home/denis/memcached/testapp.c:322 |
| 21 | 1:103 | test_issue_44 | home/denis/memcached/testapp.c:465 |
| 22 | 1:107 | lookuphost | home/denis/memcached/testapp.c:476 |
| 23 | 1:121 | connect_server | home/denis/memcached/testapp.c:504 |
| 24 | 1:138 | test_vperror | home/denis/memcached/testapp.c:549 |
| 25 | 1:149 | send_ascii_command | home/denis/memcached/testapp.c:594 |

Рисунок 5 - Перечень функциональных объектов (функций и процедур)

**Operator_guide_trial: перечень функциональных объектов (ветвей)**

| № | QID | Название | Расположение |
|---|---|---|---|
| 1 | 0:11 | for | home/denis/memcached/timedrun.c:38 |
| 2 | 0:14 | if | home/denis/memcached/timedrun.c:41 |
| 3 | 0:22 | if | home/denis/memcached/timedrun.c:50 |
| 4 | 0:25 | switch | home/denis/memcached/timedrun.c:56 |
| 5 | 0:27 | if | home/denis/memcached/timedrun.c:58 |
| 6 | 0:29 | switch | home/denis/memcached/timedrun.c:67 |
| 7 | 0:31 | switch | home/denis/memcached/timedrun.c:73 |
| 8 | 0:32 | if | home/denis/memcached/timedrun.c:77 |
| 9 | 0:39 | switch | home/denis/memcached/timedrun.c:100 |
| 10 | 0:41 | switch | home/denis/memcached/timedrun.c:107 |
| 11 | 0:43 | switch | home/denis/memcached/timedrun.c:115 |
| 12 | 1:26 | if | home/denis/memcached/testapp.c:82 |
| 13 | 1:40 | for | home/denis/memcached/testapp.c:120 |
| 14 | 1:48 | for | home/denis/memcached/testapp.c:140 |
| 15 | 1:51 | for | home/denis/memcached/testapp.c:148 |
| 16 | 1:56 | if | home/denis/memcached/testapp.c:163 |
| 17 | 1:85 | if | home/denis/memcached/testapp.c:347 |
| 18 | 1:90 | if | home/denis/memcached/testapp.c:374 |
| 19 | 1:92 | while | home/denis/memcached/testapp.c:396 |
| 20 | 1:95 | if | home/denis/memcached/testapp.c:403 |
| 21 | 1:98 | while | home/denis/memcached/testapp.c:413 |
| 22 | 1:100 | if | home/denis/memcached/testapp.c:415 |
| 23 | 1:115 | if | home/denis/memcached/testapp.c:487 |
| 24 | 1:117 | if | home/denis/memcached/testapp.c:489 |
| 25 | 1:119 | if | home/denis/memcached/testapp.c:493 |
| 26 | 1:128 | if | home/denis/memcached/testapp.c:509 |

Рисунок 6 - Перечень функциональных объектов (ветвей)

На рисунке 7 представлен перечень невызываемых ФО (функциональных объектов).

**Operator_guide_trial: перечень невызываемых ФО**

| № | QID | Название | Расположение |
|---|---|---|---|
| 1 | 0:1 | caught_signal | home/denis/memcached/timedrun.c:13 |
| 2 | 0:44 | main | home/denis/memcached/timedrun.c:120 |
| 3 | 1:4 | cache_create_test | home/denis/memcached/testapp.c:36 |
| 4 | 1:7 | cache_constructor | home/denis/memcached/testapp.c:48 |
| 5 | 1:13 | cache_constructor_test | home/denis/memcached/testapp.c:55 |
| 6 | 1:17 | cache_fail_constructor | home/denis/memcached/testapp.c:68 |
| 7 | 1:22 | cache_fail_constructor_test | home/denis/memcached/testapp.c:73 |
| 8 | 1:29 | cache_destructor | home/denis/memcached/testapp.c:93 |
| 9 | 1:33 | cache_destructor_test | home/denis/memcached/testapp.c:99 |
| 10 | 1:36 | cache_reuse_test | home/denis/memcached/testapp.c:112 |
| 11 | 1:43 | cache_bulkalloc | home/denis/memcached/testapp.c:132 |
| 12 | 1:53 | test_issue_161 | home/denis/memcached/testapp.c:159 |
| 13 | 1:58 | cache_redzone_test | home/denis/memcached/testapp.c:172 |
| 14 | 1:64 | test_safe_strtoul | home/denis/memcached/testapp.c:208 |
| 15 | 1:67 | test_safe_strtoull | home/denis/memcached/testapp.c:233 |
| 16 | 1:70 | test_safe_strtoll | home/denis/memcached/testapp.c:253 |
| 17 | 1:73 | test_safe_strtol | home/denis/memcached/testapp.c:283 |
| 18 | 1:76 | start_server | home/denis/memcached/testapp.c:322 |
| 19 | 1:103 | test_issue_44 | home/denis/memcached/testapp.c:465 |

Рисунок 7 - Перечень невызываемых ФО (Функциональных объектов)

На рисунке 8 представлен перечень неопределенных ФО.

**Operator_guide_trial: перечень неопределенных ФО**

| | № | Название | Количество вызовов |
|---|---|---|---|
| + | 1 | APPEND_NUM_STAT() | 8 |
| − | 2 | APPEND_STAT() | 7 |

| № | QID вызывающего ФО | Название вызывающего ФО | Файл и строка с вызовом |
|---|---|---|---|
| 1 | 0:101 | do_slabs_stats | home/denis/memcached/slabs.c:461 |
| 2 | 0:101 | do_slabs_stats | home/denis/memcached/slabs.c:462 |
| 3 | 6:89 | do_item_stats_totals | home/denis/memcached/items.c:539 |
| 4 | 6:89 | do_item_stats_totals | home/denis/memcached/items.c:541 |
| 5 | 6:89 | do_item_stats_totals | home/denis/memcached/items.c:543 |
| 6 | 6:89 | do_item_stats_totals | home/denis/memcached/items.c:545 |
| 7 | 6:104 | do_item_stats_sizes | home/denis/memcached/items.c:636 |

Стр. 1 из 1

| | | | |
|---|---|---|---|
| + | 3 | MEMCACHED_ASSOC_FIND() | 1 |
| + | 4 | MEMCACHED_SLABS_ALLOCATE() | 1 |
| + | 5 | MEMCACHED_SLABS_ALLOCATE_FAILED() | 1 |
| + | 6 | MEMCACHED_SLABS_FREE() | 1 |
| + | 7 | MEMCACHED_SLABS_SLABCLASS_ALLOCATE() | 1 |
| + | 8 | MEMCACHED_SLABS_SLABCLASS_ALLOCATE_FAILED() | 1 |
| + | 9 | STATS_LOCK() | 12 |
| + | 10 | STATS_UNLOCK() | 13 |
| + | 11 | __assert_fail(const char *, const char *, unsigned int, const char *) | 96 |
| + | 12 | __builtin_va_end(__va_list_tag *) | 2 |
| + | 13 | __builtin_va_start(__va_list_tag *, ...) | 2 |
| + | 14 | __ctype_b_loc() | 5 |
| + | 15 | __errno_location() | 22 |
| + | 16 | _akvs_probe() | 1463 |

Рисунок 8 - Перечень неопределенных ФО

На рисунке 9 представлена таблица связей ФО по управлению.

**Operator_guide_trial: таблица связей ФО по управлению**

| № | QID вызывающего ФО | Название вызывающего ФО | QID вызываемого ФО | Название вызываемого ФО |
|---|---|---|---|---|
| 1 | 0:34 | spawn_and_wait | 0:4 | wait_for_process |
| 2 | 1:159 | test_issue_92 | 1:149 | send_ascii_command |
| 3 | 1:162 | test_issue_102 | 1:149 | send_ascii_command |
| 4 | 1:173 | shutdown_memcached_server | 1:149 | send_ascii_command |
| 5 | 1:507 | test_issue_101 | 1:252 | test_binary_noop |
| 6 | 5:104 | worker_libevent | 5:46 | register_thread_initialized |
| 7 | 5:107 | thread_libevent_process | 5:46 | register_thread_initialized |
| 8 | 5:107 | thread_libevent_process | 5:86 | cqi_free |
| 9 | 5:225 | thread_init | 5:41 | wait_for_thread_registration |
| 10 | 7:28 | slabs_preallocate | 7:56 | do_slabs_newslab |
| 11 | 7:56 | do_slabs_newslab | 7:48 | split_slab_page_into_freelist |
| 12 | 7:62 | do_slabs_alloc | 7:56 | do_slabs_newslab |
| 13 | 7:213 | do_slabs_reassign | 7:202 | slabs_reassign_pick_any |
| 14 | 11:478 | server_socket_unix | 11:470 | new_socket_unix |
| 15 | 11:556 | main | 11:26 | stats_init |
| 16 | 11:556 | main | 11:30 | settings_init |
| 17 | 11:556 | main | 11:32 | conn_init |
| 18 | 11:556 | main | 11:498 | clock_handler |
| 19 | 11:556 | main | 11:506 | usage |
| 20 | 11:556 | main | 11:508 | usage_license |
| 21 | 11:556 | main | 11:510 | save_pid |

Рисунок 9 - Таблица связей ФО по управлению

На рисунке 10 представлены маршруты выполнения ФО.

**Operator_guide_trial: выберите исходный ФО для построения маршрута**

| № | QID ФО | Название ФО | Маршрут |
|---|---|---|---|
| 1 | 0:33 | slabs_preallocate | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 2 | 0:54 | split_slab_page_into_freelist | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 3 | 0:62 | do_slabs_newslab | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 4 | 0:70 | do_slabs_alloc | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 5 | 0:133 | slabs_alloc | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 6 | 0:138 | slabs_free | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 7 | 0:241 | do_slabs_reassign | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 8 | 4:53 | test_issue_161 | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 9 | 4:169 | test_issue_92 | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 10 | 4:172 | test_issue_102 | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 11 | 4:186 | shutdown_memcached_server | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 12 | 4:352 | test_binary_replace_impl | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 13 | 4:372 | test_binary_delete_impl | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 14 | 4:413 | test_binary_getq_impl | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 15 | 4:498 | test_binary_concat_impl | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 16 | 4:549 | test_binary_pipeline_hickup | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 17 | 4:561 | test_issue_101 | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 18 | 7:34 | spawn_and_wait | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 19 | 10:183 | process_bin_stat | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 20 | 10:368 | process_stat | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 21 | 10:408 | process_get_command | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 22 | 10:578 | server_socket_unix | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 23 | 10:656 | main | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 24 | 11:37 | stats_prefix_record_get | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |

Рисунок 10 - Маршруты выполнения ФО

Нажмите на ссылку в поле «Маршрут», чтобы посмотреть маршрут в интерактивном режиме. После этого откроется страница, изображенная на рисунке 11. Нажмите соответствующую кнопку, чтобы продолжить маршрут вперед или назад. Полученные маршруты можно сохранить в формате SVG. На рисунке 11 представлен пример построения маршрутов от точек выхода до выбранного ФО. Нажмите кнопки “продолжить маршрут вперед” или “продолжить маршрут назад”, чтобы попасть в

соответствующие меню. Полученные маршруты можно сохранить в формате SVG.

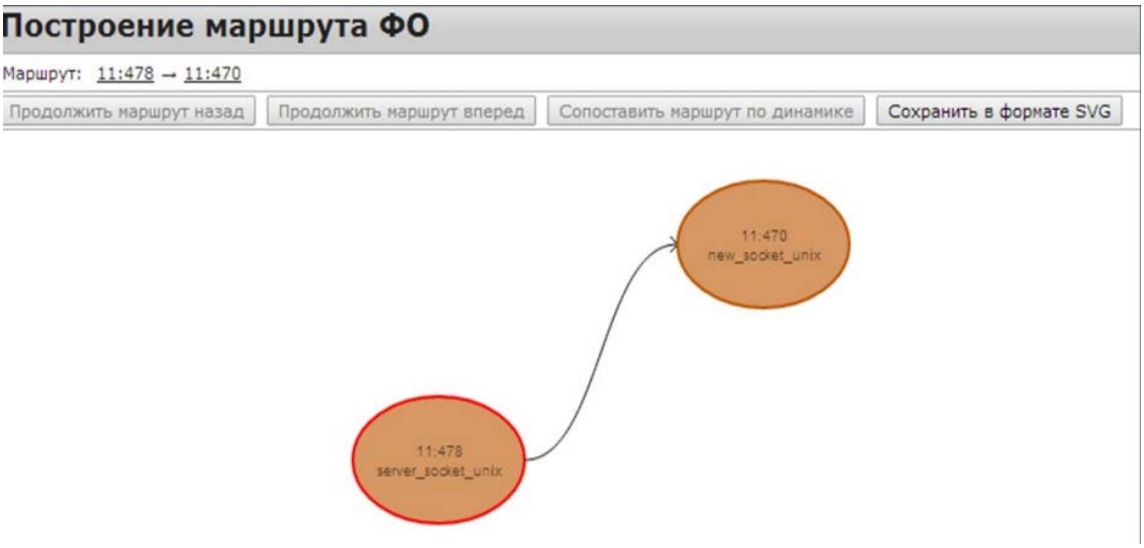



Рисунок 11 - Пример построения маршрутов от точек входа до выбранного ФО

На рисунке 12 показаны все маршруты от точки входа до выбранного ФО. Нажмите на соответствующую ссылку, чтобы просмотреть интересующий маршрут. Маршруты можно сохранить в формате SVG. Также имеется возможность сопоставить маршруты по динамике, если был проведен динамический анализ.

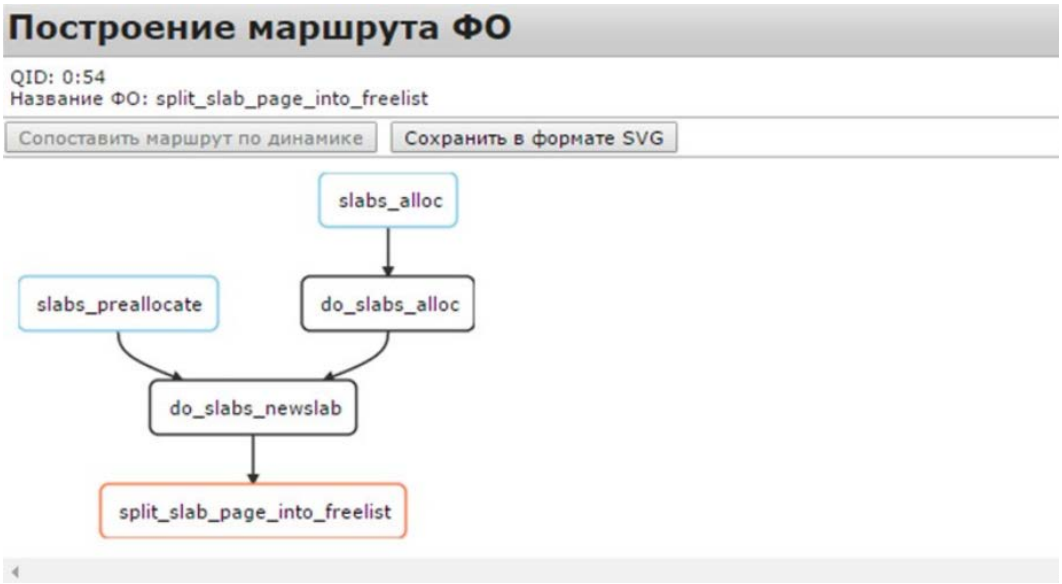



Рисунок 12 – Пример построения маршрутов от точек входа до выбранного ФО

На рисунке 13 представлена таблица связей ФО по информации.

**Operator_guide_trial: таблица связей ФО по информации**

| № | QID ИО | Название ИО | QID ФО | Название ФО |
|---|---|---|---|---|
| 1 | 0:0 | caught | 0:1 | caught_signal |
| 2 | | | 0:4 | wait_for_process |
| 4 | 1:1 | TEST_SKIP | 1:507 | test_issue_101 |
| 5 | | | 1:540 | main |
| 7 | | | 1:4 | cache_create_test |
| 8 | | | 1:13 | cache_constructor_test |
| 9 | | | 1:33 | cache_destructor_test |
| 10 | | | 1:36 | cache_reuse_test |
| 11 | | | 1:43 | cache_bulkalloc |
| 12 | | | 1:53 | test_issue_161 |
| 13 | | | 1:58 | cache_redzone_test |
| 14 | | | 1:64 | test_safe_strtoul |
| 15 | | | 1:67 | test_safe_strtoull |
| 16 | | | 1:70 | test_safe_strtoll |
| 17 | | | 1:73 | test_safe_strtol |
| 18 | | | 1:103 | test_issue_44 |
| 19 | | | 1:159 | test_issue_92 |
| 20 | | | 1:162 | test_issue_102 |
| 21 | | | 1:167 | start_memcached_server |
| 22 | | | 1:169 | stop_memcached_server |
| 23 | | | 1:173 | shutdown_memcached_server |
| 24 | 1:2 | TEST_PASS | 1:252 | test_binary_noop |
| 25 | | | 1:258 | test_binary_quit_impl |
| 26 | | | 1:271 | test_binary_set_impl |
| 27 | | | 1:290 | test_binary_add_impl |
| 28 | | | 1:313 | test_binary_replace_impl |

Рисунок 13 - Таблица связей по ФО по информации

На рисунке 14 представлен отчет по критическим маршрутам для выбранного ИО.

**Operator_guide_trial: выберите ИО**

| № | QID ИО | Название ИО |
|---|---|---|
| 1 | 0:0 | caught |
| 2 | 0:2 | caught_signal::which |
| 3 | 0:5 | wait_for_process::pid |
| 4 | 0:7 | wait_for_process::rv |
| 5 | 0:8 | wait_for_process::stats |
| 6 | 0:9 | wait_for_process::i |
| 7 | 0:10 | wait_for_process::sig_handler |
| 8 | 0:13 | wait_for_process::p |
| 9 | 0:24 | wait_for_process::sig |
| 10 | 0:35 | spawn_and_wait::argv |
| 11 | 0:37 | spawn_and_wait::rv |
| 12 | 0:38 | spawn_and_wait::pid |
| 13 | 0:45 | main::argc |
| 14 | 0:46 | main::argv |

Рисунок 14 - Отчет по критическим маршрутам

На рисунке 15 представлены критические маршруты для выбранного ИО. Нажмите на надпись «Проложить маршрут» для просмотра маршрута.

**Operator_guide_trial: критические маршруты для выбранного ИО**

| № | QID | Название | Маршрут |
|---|---|---|---|
| 1 | 1:13 | cache_constructor_test | проложить маршрут |
| 2 | 1:22 | cache_fail_constructor_test | проложить маршрут |
| 3 | 1:33 | cache_destructor_test | проложить маршрут |
| 4 | 1:497 | test_binary_pipeline_hickup | проложить маршрут |

Стр. 1 из 1 1000 Просмотр 1 - 4 из 4

Рисунок 15 - Пример отчета по критическим маршрутам для выбранного ИО

На рисунке 16 представлена таблица связей функций и ветвей.

**Operator_guide_trial: таблица связей функций и ветвей**

| № | QID вызывающего объекта | Название вызывающего объекта | QID вызываемого объекта | Название вызываемого объекта |
|---|---|---|---|---|
| 1 | 0:4 | wait_for_process | 0:11 | for |
| 2 | 0:11 | for | 0:14 | if |
| 3 | 0:11 | for | 0:22 | if |
| 4 | 0:22 | if | 0:25 | switch |
| 5 | 0:22 | if | 0:29 | switch |
| 6 | 0:22 | if | 0:31 | switch |
| 7 | 0:22 | if | 0:32 | if |
| 8 | 0:25 | switch | 0:27 | if |
| 9 | 0:34 | spawn_and_wait | 0:39 | switch |
| 10 | 0:34 | spawn_and_wait | 0:41 | switch |

Рисунок 16 - Пример таблицы связей функций и ветвей

На рисунке 17 представлены маршруты выполнения ФО с ветвями. Чтобы посмотреть интересующий маршрут, нажмите на соответствующую ссылку. Страница для построения интерактивных маршрутов и страница, содержащая все маршруты от точек входа до выбранного ФО, аналогичны рисунку 11 и 12.

**Operator_guide_trial: выберите исходный ФО для построения маршрута**

| № | QID ФО | Название ФО | Маршрут |
|---|---|---|---|
| 1 | 0:17 | slabs_clsid | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 2 | 0:23 | slabs_init | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 3 | 0:33 | slabs_preallocate | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 4 | 0:44 | grow_slab_list | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 5 | 0:54 | split_slab_page_into_freelist | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 6 | 0:62 | do_slabs_newslab | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 7 | 0:70 | do_slabs_alloc | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 8 | 0:101 | do_slabs_stats | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |
| 9 | 0:119 | memory_allocate | Построить маршрут в интерактивном режиме<br>Построить все маршруты от точек входа до выбранного ФО |

Рисунок 17 - Маршруты выполнения ФО с ветвями

На рисунке 18 представлена таблица для построения блок-схемы ФО.

**Operator_guide_trial: выберите ФО для построения блок-схемы**

| № | QID ФО | Название ФО |
|---|---|---|
| 1 | 0:1 | caught_signal |
| 2 | 0:4 | wait_for_process |
| 3 | 0:34 | spawn_and_wait |
| 4 | 0:44 | main |
| 5 | 1:4 | cache_create_test |
| 6 | 1:7 | cache_constructor |
| 7 | 1:13 | cache_constructor_test |
| 8 | 1:17 | cache_fail_constructor |
| 9 | 1:22 | cache_fail_constructor_test |
| 10 | 1:29 | cache_destructor |
| 11 | 1:33 | cache_destructor_test |
| 12 | 1:36 | cache_reuse_test |

Рисунок 18 - Таблица выбора ФО для построения блок-схемы

Нажмите на один из представленных ФО, для просмотра блок-схемы. После этого откроется страница, показанная на рисунке 19. Представленную блок-схему можно сохранить в формате SVG.

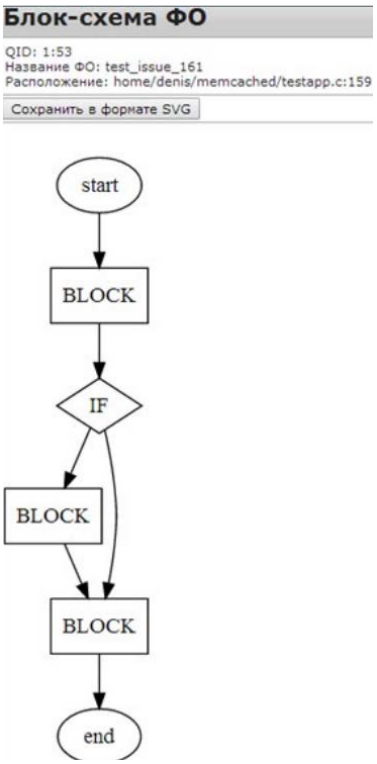



Рисунок 19 - Блок-схема для выбранного ФО

На рисунке 20 представлен отчет о сигнатурном анализе, отображающем дефекты программного кода с сопоставлением им соответствующих CWE-идентификаторов.

**Отчёт о сигнатурном анализе**

| № | Описание по CWE | Файл | № Строки | Строка |
|---|---|---|---|---|
| 1 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/sc | 1413 | if ( GetAPOPTimeStamp ( m_serverReply _ timeStamp |
| 2 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/sc | 1415 | sprintf ( authString _ %s %s _ GETP_String ( PROFILE_ |
| 3 | Переполнение буфера на стеке (Срыв стека) (CWE: | ../../data/projects/2/src/popcorn_195_source/source/sc | 2329 | strcpy ( s _ username ) ; |
| 4 | Шпионское ПО (CWE:512) | ../../data/projects/2/src/popcorn_195_source/source/m | 68 | if ( 0 != GetTempPath ( MAX_PATH _ m_appTempDir ) ) |
| 5 | Вызов сторонних модулей. (CWE: 510) | ../../data/projects/2/src/popcorn_195_source/source/m | 152 | } { m_richEdDLL = LoadLibrary ( riched20.dll ) ; |
| 6 | Переполнение буфера на стеке (Срыв стека) (CWE: | ../../data/projects/2/src/popcorn_195_source/source/m | 172 | strcpy ( m_appCmdLine _ commandLine ) ; |
| 7 | Предопределенный IP-адрес. (CWE: 489) | ../../data/projects/2/src/popcorn_195_source/source/se | 55 | static char * DefaultSpellCheckURL = http://dictionary.. |
| 8 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/se | 1067 | sprintf ( buff _ \%s_%d_%08x_%08x _ font-&rt;name |
| 9 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/se | 1187 | sprintf ( section _ %s%d _ m_settingsecs [ 2 ] . name |
| 10 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/se | 1325 | sprintf ( section _ %s%d _ m_settingsecs [ 2 ] . name |
| 11 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/m | 832 | if ( strlen ( m_to ) ) { if ( to ) sprintf ( m_header + strle |
| 12 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/m | 834 | else sprintf ( m_header + strlen ( m_header ) _ HF_TC |
| 13 | Переполнение буфера (CWE:120) | ../../data/projects/2/src/popcorn_195_source/source/m | 844 | } if ( strlen ( m_cc ) ) { if ( cc ) sprintf ( m_header + str |

Рисунок 20 - Отчет о сигнатурном анализе

CWE (Common Weakness Enumeration) представляет собой список дефектов безопасности программного обеспечения. CWE служит в качестве базового стандарта для идентификации дефектов, смягчения их последствий и профилактических мероприятий. CWE охватывает более 15000 дефектов безопасности и включает в себя детальную структуру классификации из различных научных источников и примеров.

CWE позволяет компаниям ускорить проверку разрабатываемого программного обеспечения, а также помогает пользователям в выборе продуктов и услуг, соответствующих их требованиям.

CWE-совместимость позволяет продукту быть зарегистрированным в качестве официально “CWE-совместимого”. CWE-совместимый продукт должен предоставлять пользователям возможность поиска дефектов безопасности по CWE-идентификаторам и получения связанных с ними CWE-идентификаторов.

В сводном отчете:

- Чтобы получить более подробную информацию о дефекте, нажмите на CWE-идентификатор;
- Чтобы воспользоваться поиском дефектов по CWE-идентификаторам, кликните на значок лупы в левом нижнем углу отчета, выберите параметр поиска «Описание по CWE», укажите условие «содержит» и введите CWE-идентификатор.

Чтобы скачать все отчеты по статическому анализу:

В разделе «Отчеты» нажмите кнопку «Скачать все отчеты». Отчеты скачиваются в архиве с расширением .zip.

Чтобы посмотреть отчеты статического анализа: Откройте в браузере файл index.html.

Чтобы посмотреть журнал статического анализа:

В разделе «Журналы» нажмите кнопку «Журнал статического анализа».

*3. Динамический анализ*

Внимание! Динамический анализ проводится после статического.

1) В ходе статического анализа в исходный текст внедряются датчики. Скачайте исходные тексты с датчиками. В разделе «Дальнейшие действия» нажмите кнопку «Скачать исходные тексты с датчиками»;
2) Соберите приложение с внедренными датчиками, используя включенные в поставку «АК-ВС 2» исходные файлы датчиков для соответствующего языка. Необходимые датчики можно скачать во вкладке «Датчики»;
3) Внедрение датчиков

*3.1 Сборка проекта C/C++*

Расположение датчика: файл _akvs_probe.h

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 2 вставляет в начало каждого файла конструкцию:

#include “_akvs_probe.h”

а в те файлы, где есть функция main/WinMain, конструкцию:

#define _AKVS_PROBE_IMPLEMENTATION_

#include “_akvs_probe.h”

Внимание! Датчик поддерживает многопоточные приложения. Отключите режим многопоточности, если он не используется в программе, удалив из файла _akvs_probe.h строчку #define _AKVS_PROBE_MULTITHREAD

Чтобы собрать проект C/C++:

1) скачайте датчик _akvs_probe.h;
2) скопируйте его в include path проекта.

(Для unix-систем можно разместить его в /usr/include, чтобы не копировать датчик в каждый подпроект.)

*3.2 Сборка проекта C#*

Расположение датчика: файл _akvs_probe.h

Чтобы собрать проект C#

1) скачайте датчик _akvs_probe.cs;
2) добавьте его в проект.

*3.3 Сборка проекта Java*

Расположение датчика: пакет ru.cnpo.akvs2.probejava;

Чтобы собрать проект Java:

1) создайте путь ru/cnpo/akvs2/probejava в директории с исходными файлами;
2) скопируйте туда датчик JavaProbe.java.

*4 Обработка результатов*

Запустите собранное приложение и проведите максимально глубокое функциональное тестирование. Можно проводить тестирование в несколько этапов, сохраняя получившиеся файлы лога.

Ответственность за полноту тестирования несет человек, проводящий тестирование. По завершении тестирования в рабочей директории приложения будет находиться файл лога отработки датчиков (с расширением .log).

В разделе «Дальнейшие действия» нажмите кнопку «Обзор…» и выберите логи отработки динамического анализа, чтобы загрузить логи отработки датчиков в проект.

Логи должны находиться в архиве с расширением .zip.

При необходимости имеется возможность задать кодировку zip-архива. После этого нажмите кнопку «Начать загрузку».

При этом статус проекта должен измениться на «Обработка результатов динамического анализа».

Необходимо дождаться окончания динамического анализа. При необходимости следует нажмите кнопку «Обновить».

При окончании анализа статус проекта должен измениться на «Динамический анализ завершен». После этого на странице проекта в разделах «Отчеты» и «Журналы» появятся дополнительные кнопки для просмотра отчетов и журналов по динамическому анализу.

Чтобы посмотреть отчет по динамическому анализу в разделе «Отчеты» нажмите кнопку «Отчеты по динамическому анализу».

*5 Примеры отчетов*

На рисунке 21 показана страница с отчетами по динамическому анализу.

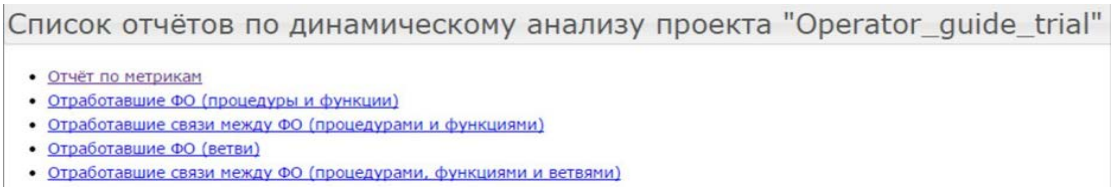



Рисунок 21 - Отчеты по динамическому анализу

Чтобы посмотреть интересующий отчет, необходимо навести на него курсор и нажать левую кнопку мыши. Примеры отчетов представлены на рисунках 22 – 24. На рисунке 22 представлен отчет по метрикам динамического анализа.

**Operator_guide_trial: отчёт по метрикам динамического анализа**

| | |
|---|---|
| Количество подтвердившихся вызовов ФО/общее количество ФО | 0/279 |
| Процент покрытия по ФО | 0% |
| Количество подтвердившихся связей ФО/общее количество связей ФО | 0/42 |
| Процент покрытия по связям ФО | 0% |
| Количество подтвердившихся вызовов ветвей/общее количество ветвей | 0/399 |
| Процент покрытия по ветвям | 0% |
| Количество подтвердившихся связей ФО с ветвями/общее количество связей ФО с вет | 0/416 |
| Процент покрытия по связям ФО с ветвями | 0% |

Рисунок 22 - Отчет по метрикам динамического анализа

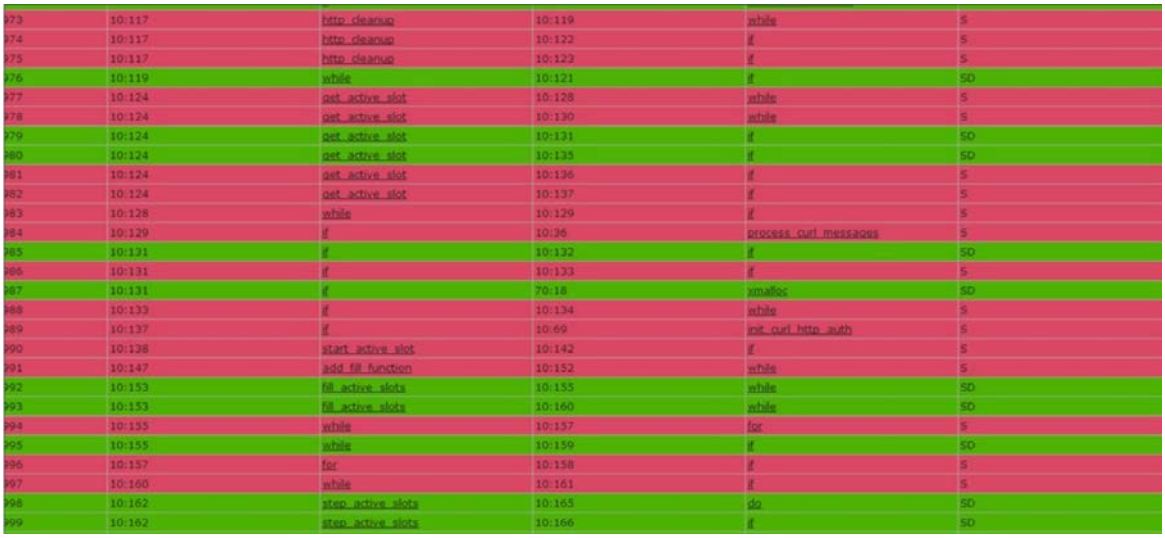
На рисунке 23 представлены отработавшие ФО (процедуры и функции). Красным цветом выделены ФО, не отработавшие в ходе динамического анализа, зеленым — отработавшие.

**Git: отработавшие ФО (процедуры и функции)**

| № | QID | Название | Отработал |
|---|---|---|---|
| 1 | 0:8 | cmd_capabilities | - |
| 2 | 0:10 | terminate_batch | - |
| 3 | 0:11 | read_ref_note | - |
| 4 | 0:21 | parse_rev_note | - |
| 5 | 0:33 | note2mark_cb | - |
| 6 | 0:46 | regenerate_marks | - |
| 7 | 0:51 | check_or_regenerate_marks | - |
| 8 | 0:64 | cmd_import | - |
| 9 | 0:85 | cmd_list | - |
| 10 | 0:87 | do_command | - |
| 11 | 0:101 | main | - |
| 12 | 1:1 | main | - |
| 13 | 2:3 | get_next | - |
| 14 | 2:5 | set_next | - |
| 15 | 2:8 | compare_strings | - |
| 16 | 2:13 | main | - |
| 17 | 3:1 | git_user_agent | + |
| 18 | 3:5 | git_user_agent_sanitized | + |
| 19 | 4:0 | check_removed | - |

Рисунок 23 - Отработавшие ФО (процедуры и функции)

На рисунке 24 представлены отработавшие связи между ФО (процедурами, функциями и ветвями). В отчете имеются поля, аналогичные представленным в отчете по отработавшим связям между ФО (процедурами и функциями).



| | | | | | |
|---|---|---|---|---|---|
| 973 | 10:117 | http_cleanup | 10:119 | while | S |
| 974 | 10:117 | http_cleanup | 10:122 | if | S |
| 975 | 10:117 | http_cleanup | 10:123 | if | S |
| 976 | 10:119 | while | 10:121 | if | SD |
| 977 | 10:124 | get_active_slot | 10:128 | while | S |
| 978 | 10:124 | get_active_slot | 10:130 | while | S |
| 979 | 10:124 | get_active_slot | 10:131 | if | SD |
| 980 | 10:124 | get_active_slot | 10:135 | if | SD |
| 981 | 10:124 | get_active_slot | 10:136 | if | S |
| 982 | 10:124 | get_active_slot | 10:137 | if | S |
| 983 | 10:128 | while | 10:129 | if | S |
| 984 | 10:129 | if | 10:36 | process_curl_messages | S |
| 985 | 10:131 | if | 10:132 | if | SD |
| 986 | 10:131 | if | 10:133 | if | S |
| 987 | 10:131 | if | 70:18 | xmalloc | SD |
| 988 | 10:133 | if | 10:134 | while | S |
| 989 | 10:137 | if | 10:69 | init_curl_http_auth | S |
| 990 | 10:138 | start_active_slot | 10:142 | if | S |
| 991 | 10:147 | add_fill_function | 10:152 | while | S |
| 992 | 10:153 | fill_active_slots | 10:155 | while | SD |
| 993 | 10:153 | fill_active_slots | 10:160 | while | SD |
| 994 | 10:155 | while | 10:157 | for | S |
| 995 | 10:155 | while | 10:159 | if | SD |
| 996 | 10:157 | for | 10:158 | if | S |
| 997 | 10:160 | while | 10:161 | if | S |
| 998 | 10:162 | step_active_slots | 10:165 | do | SD |
| 999 | 10:162 | step_active_slots | 10:166 | if | SD |

Рисунок 24 - Отработавшие связи между ФО (процедурами, функциями и ветвями)

# 4 ЛАБОРАТОРНАЯ РАБОТА

Цели лабораторной работы:

1. закрепление студентами теоретических знаний, полученных в ходе курса лекций по дисциплине «Технология сертификации средств защиты информации»;

2. получение практических навыков проведения сертификационных испытаний, а именно:

2.1. анализа исходных текстов программ при помощи анализатора исходных текстов «АК-ВС 2»;

2.2. ознакомления с отчетами и анализа выявленных проблем;

2.3. написания протокола-отчета о проведении сертификационных испытаний.

Перед началом работы необходимо подготовить 2 архива:

1. src.zip - архив с исходным текстом проекта;

2. prj.zip - архив с конфигурационным файлом и используемыми сторонними заголовочными файлами.

Запуск программы:

Для анализа исходного кода необходимо подключиться в веб-интерфейсу «АК-ВС 2» и авторизоваться, используя логин «admin» и пароль «admin» по адресу http://itmo.nwechelon.ru:11000.

При успешной авторизации открывается вкладка «Проекты», где представлены все текущие проекты (Рисунок 25). На вкладке «№» — идентификаторы проектов, в поле «Имя» — имя проекта, в поле «Статус» — стадия, на которой находится проект.

**Список проектов**

| № | Имя | Статус |
|---|---|---|
| 1 | OpenSSL | Статический анализ завершен |
| 2 | I2P | Статический анализ завершен |
| 3 | sasa | Статический анализ завершен |
| 4 | OpenRA | Статический анализ завершен |
| 5 | EMpty | Статический анализ завершен |
| 6 | sc_test | Статический анализ завершен |
| 7 | cs_test | Ошибка статического анализа |

Рисунок 25 - Вкладка «Проекты» веб-интерфейса

Во вкладке «Проекты» нажмите кнопку «Добавить», заполните необходимые поля и приступите к выполнению задания на лабораторную работу.

Для проведения статического анализа необходимо ввести название проекта и уровень контроля, после чего нажать кнопку «submit». В появившейся вкладке можно увидеть информацию, заполненную ранее, которую можно сбросить или удалить. Сброс настроек позволяет обнулить загруженные ранее файлы. «Удаление» полностью стирает проект. Необходимо выбрать архив с исходным кодом и загрузить его.

После загрузки исходных кодов вам будет предложено загрузить архив с конфигурационным файлом и используемыми сторонними заголовочными файлами. После загрузки поле «статус» изменится и начнется подготовка отчетов программы «АК-ВС 2».

После окончания процесса генерации отчетов появляются дополнительные пункты, позволяющие просмотреть полученные отчеты или загрузить их на локальный компьютер. При необходимости можно посмотреть логи работы “АК-ВС 2”. При открытии вкладки с отчетами отображается список сгенерированных отчетов по статическому анализу проекта доступных для просмотра.

Для открытия отчета необходимо выбрать его из предложенного списка и кликнуть по нему. Приведем пример одного из таких отчетов «отчет о сигнатурном анализе». В списке (рисунок 25) можно увидеть обнаруженные уязвимости с указанием номера в реестре уязвимостей, выявленных в исследуемом проекте.

Задание на лабораторную работу

1. Необходимо провести анализ выданных ранее исходных данных при помощи “АК-ВС 2” на соответствие 4 класса НДВ.
2. Ознакомиться с полученными отчетами “АК-ВС 2” и проанализировать их.
3. Выполнить анализ всех найденных CWE с описанием каждой уязвимости.
4. Описать содержание отчетов, генерируемых “АК-ВС 2”.
5. Оформить отчет по лабораторной работе в виде протокола-отчета о проведении сертификационных испытаний.

## 5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое сертификация?
2. Назовите цели сертификации.
2. Опишите процесс сертификации программных средств.
3. Назовите законы в области сертификации средств защиты информации, действующие на территории РФ.
4. Перечислите исходные документы для сертификации.
5. Перечислите документы, на степень соответствия которым может оцениваться программное средство при его сертификации.
6. Перечислите документы заявителя, необходимые для организации сертификации программных средств.
7. Перечислите документы органа сертификации, поддерживающие организацию сертификации программных средств.
8. Перечислите документы испытательной лаборатории, поддерживающие организацию сертификации программных средств.
9. Сколько уровней контроля на отсутствие недекларируемых возможностей устанавливает РД НДВ? Охарактеризуйте их.

## ЗАКЛЮЧЕНИЕ

Данное учебно-методическое пособие призвано помочь студентам, изучающим в рамках своей образовательной программы дисциплину «Технология сертификации средств защиты информации», эффективнее освоить изучаемый материал – основные принципы и цели сертификации, законодательство в области сертификации средств защиты информации, особенности сертификации программного обеспечения, порядок проведения сертификации, участников процесса сертификации, ключевые моменты взаимодействия участников, состав сертификационных испытаний, анализ исходных текстов программ, отличия и порядок проведения динамического и статистического анализа исходных кодов, выполнение статистического и динамического анализа с помощью программных средств.

Лабораторная работа, включенная в состав данного учебно-методического пособия, призвана помочь понять, как на практике выполняются проверочные действия (испытания) на отсутствие (наличие) недекларированных возможностей программного обеспечения.

## СПИСОК ЛИТЕРАТУРЫ


1. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей [Текст] : руководящий документ от 4 июня 1999 г. N 114.
2. Скабцов Н.В. Аудит безопасности информационных систем [Текст]. – М.: Питер, 2018. – 272 с.
3. Стародубцев Ю.И. Управление качеством информационных услуг [Текст] / Ю.И. Стародубцев, А.Н. Бегаев, М.А. Дятлова; под общ. ред. Ю.И. Стародубцева. – СПб: Изд-во Политехн. Ун-та, 2017. – 454 с.
4. Бегаев А.Н., Тарасюк М.В. Контроль безопасности программного кода в составе объекта информатизации [Текст] / Защита информации. Инсайд. 2013. № 5 (53). С. 63-67.
5. Костогрызов А.И., Липаев В.В. Сертификация функционирования автоматизированных информационных систем [Текст]. М.: Изд. «Вооружение. Политика. Конверсия», 1996. 280 с.
6. Марков А., Никулин М., Цирлов В. Сертификация средств защиты персональных данных: революция или эволюция [Текст] / Защита информации. Инсайд. 2008. № 5 (23). С. 20-25.
7. Марков А., Цирлов В. Сертификация программ: мифы и реальность [Текст] / Открытые системы. СУБД. 2011. № 6. С. 26.
8. Марков А.С., Цирлов В.Л., Барабанов А.В. Методы оценки несоответствия средств защиты информации [Текст]. Под. ред. А.С.Маркова. -М.: Радио и связь, 2012. 192 с.
9. Стародубцев Ю.И., Бегаев А.Н., Давлятова М.А. Управление качеством информационных услуг . СПб.: Изд-во Политехн. ун-та, 2017. 454 с.
10. Барабанов А., Марков А., Цирлов В. Сертификация систем обнаружения вторжений [Текст] / Открытые системы. СУБД. 2012. № 3. С. 31-33.
11. Барабанов А.В., Марков А.С., Цирлов В.Л. Испытания межсетевых экранов по требованиям безопасности информации [Текст]: Учебное издание. М.: НЦПИ при Минюсте России, 2017. 44 с.
12. Барабанов А.В., Марков А.С., Цирлов В.Л. Международная сертификация в области информационной безопасности [Текст] / Стандарты и качество. 2016. № 7. С. 30-33.
13. Барабанов А.В., Федичев А.В. Разработка типовой методики анализа уязвимостей в веб-приложениях при проведении сертификационных испытаний по требованиям безопасности информации [Текст] / Вопросы кибербезопасности. 2016. № 2 (15). С. 2-8.
14. Горюнов М.Н., Юдичев Р.М., Фадин А.А. Внедрение сертификации в жизненный цикл программного обеспечения [Текст] / Защита информации. Инсайд. 2016. № 3 (69). С. 28-35.

15. Марков А.С., Шеремет И.А. Теоретические аспекты сертификации средств защиты информации [Текст] / Оборонный комплекс - научно-техническому прогрессу России. 2015. № 4 (128). С. 7-15.
16. Александров Я.А., Сафин Л.К., Чернов А.В., Трошина К.Н. Определение границ подпрограмм при статическом анализе бинарных образов [Текст] / Вопросы кибербезопасности. 2016. № 1 (14). С. 53-60.
17. Барабанов А. Инструментальные средства проведения испытаний систем по требованиям безопасности информации [Текст] / Защита информации. Инсайд. 2011. № 1 (37). С. 49-51.
18. Барабанов А.В., Марков А.С., Фадин А.А., Цирлов В.Л. Статистика выявления уязвимостей программного обеспечения при проведении сертификационных испытаний [Текст] / Вопросы кибербезопасности. 2017. № 2 (20). С. 2-8.
19. Жидков И.В., Кадушкин И.В. О признаках потенциально опасных событий в информационных системах [Текст] / Вопросы кибербезопасности. 2014. № 1 (2). С. 40-48.
20. Марков А.С., Барабанов А.В., Фадин А.А. Выявление недекларированных возможностей в декомпилированных текстах программного обеспечения [Текст] / Известия Института инженерной физики. 2010. Т. 4. № 18. С. 24-26.
21. Марков А.С., Матвеев В.А., Фадин А.А., Цирлов В.Л. Эвристический анализ безопасности программного кода [Текст] / Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия: Приборостроение. 2016. № 1 (106). С. 98-111.
22. Марков А.С., Фадин А.А. Статический сигнатурный анализ безопасности программ [Текст] / Программная инженерия и информационная безопасность. 2013. № 1. С. 50-56.
23. Марков А.С., Фадин А.А., Цирлов В.Л. Концептуальные основы построения анализатора безопасности программного кода [Текст] / Программные продукты и системы. 2013. № 1. С. 10.
24. Мельников П.В., Горюнов М.Н., Анисимов Д.В. Подход к проведению динамического анализа исходных текстов программ [Текст] / Вопросы кибербезопасности. 2016. № 3 (16). С. 33-39.
25. Менщиков А.А., Комарова А.В., Гатчин Ю.А. Изучение поведения средств автоматизированного сбора информации с веб-ресурсов [Текст] / Вопросы кибербезопасности. 2017. № 3 (21). С. 49-54.
26. Осовецкий Л.Г. Технология выявления недекларированных возможностей при сертификации промышленного программного обеспечения по требованиям безопасности информации [Текст] / Вопросы кибербезопасности. 2015. № 1 (9). С. 60-64.
27. Осовецкий Л.Г., Ефимова А.В. Гарантии выявления недекларированных возможностей в промышленном программном обеспечении [Текст] / Известия Института инженерной физики. 2016. Т. 2. № 40. С. 59-63.

28. Рибер Г., Малмквист К., Щербаков А. Многоуровневый подход к оценке безопасности программных средств [Текст] / Вопросы кибербезопасности. 2014. № 1 (2). С. 36-39.
29. Самарин Н.Н. Виды потенциально-опасных возможностей, реализуемых вредоносным кодом [Текст] / Успехи современной науки и образования. 2016. Т. 4. № 9. С. 199-202.
30. Markov A., Fadin A., Shvets V., Tsirlov V. The experience of comparison of static security code analyzers [Текст] / International Journal of Advanced Studies. 2015. V. 5. N 3. P. 55-63.
31. Markov A.S., Fadin A.A., Tsirlov V.L. Multilevel metamodel for heuristic search of vulnerabilities in the software source code [Текст] / International Journal of Control Theory and Applications. 2016. V. 9. No 30. P. 313-320.

**Миссия университета** – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

---

## НАПРАВЛЕНИЕ ПОДГОТОВКИ (СПЕЦИАЛЬНОСТИ) 10.03.01 «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Направление подготовки (специальности) 10.03.01 «Информационная безопасность» реализуется как профессиональная образовательная программа высшего образования бакалавриата в Университете ИТМО. Кафедра проектирования и безопасности компьютерных систем осуществляла подготовку бакалавров в области информационной безопасности компьютерных систем по данному направлению подготовки.

## ИСТОРИЯ РЕАЛИЗАЦИИ НАПРАВЛЕНИЯ

**1945-1966 РЛПУ** (кафедра радиолокационных приборов и устройств). Решением Советского правительства в августе 1945 г. в ЛИТМО был открыт факультет электроприборостроения. Приказом по институту от 17 сентября 1945 г. на этом факультете была организована кафедра радиолокационных приборов и устройств, которая стала готовить инженеров, специализирующихся в новых направлениях радиоэлектронной техники, таких как радиолокация, радиоуправление, теленаведение и др. Организатором и первым заведующим кафедрой был д.т.н., профессор С.И. Зилитинкевич (до 1951 г.). Выпускникам кафедры присваивалась квалификация инженер-радиомеханик, а с 1956 г. – радиоинженер (специальность 0705).

В разные годы кафедрой заведовали доцент Б.С. Мишин, доцент И.П. Захаров, доцент А.Н. Иванов.

**1966–1970 КиПРЭА** (кафедра конструирования и производства радиоэлектронной аппаратуры). Каждый учебный план специальности 0705 коренным образом отличался от предыдущих планов радиотехнической

специальности своей четко выраженной конструкторско-технологической направленностью. Оканчивающим институт по этой специальности присваивалась квалификация инженер-конструктор-технолог РЭА.

Заведовал кафедрой доцент А.Н. Иванов.

**1970–1988 КиПЭВА** (кафедра конструирования и производства электронной вычислительной аппаратуры). Бурное развитие электронной вычислительной техники и внедрение ее во все отрасли народного хозяйства потребовали от отечественной радиоэлектронной промышленности решения новых ответственных задач. Кафедра стала готовить инженеров по специальности 0648. Подготовка проводилась по двум направлениям – автоматизация конструирования ЭВА и технология микроэлектронных устройств ЭВА.

Заведовали кафедрой: д.т.н., проф. В.В. Новиков (до 1976 г.), затем проф. Г.А. Петухов.

**1988–1997 МАП** (кафедра микроэлектроники и автоматизации проектирования). Кафедра выпускала инженеров-конструкторов-технологов по микроэлектронике и автоматизации проектирования вычислительных средств (специальность 2205). Выпускники этой кафедры имеют хорошую технологическую подготовку и успешно работают как в производстве полупроводниковых интегральных микросхем, так и при их проектировании, используя современные методы автоматизации проектирования. Инженеры специальности 2205 требуются микроэлектронной промышленности и предприятиям-разработчикам вычислительных систем.

Кафедрой с 1988 г. по 1992 г. руководил проф. С.А. Арустамов, затем снова проф. Г.А. Петухов.

С 1996 г. кафедрой заведует д.т.н., профессор Ю.А. Гатчин.

**1997–2011 ПКС** (кафедра проектирования компьютерных систем). Кафедра выпускала инженеров по специальности 210202 «Проектирование и технология электронно-вычислительных средств». Область профессиональной деятельности выпускников включала в себя проектирование, конструирование и технологию электронных средств, отвечающих целям их функционирования, требованиям надежности, дизайна и условиям эксплуатации. Кроме того, кафедра готовила специалистов по защите информации, специальность 090104 «Комплексная защита объектов информатизации». Объектами профессиональной деятельности специалиста по защите информации являются методы, средства и системы обеспечения защиты информации на объектах

информатизации.

В 2009 и 2010 годах кафедра заняла второе, а в 2011 году – почетное первое место в конкурсе среди кафедр университета.

С **2011 года ПБКС** (кафедра проектирования и безопасности компьютерных систем). Кафедра осуществляет подготовку бакалавров и магистров по направлениям 090900 «Информационная безопасность» (с 2013 г. коды направления: для бакалавров 10.03.01, для магистров 10.04.01) и 211000 «Конструирование и технология электронных средств» (с 2013 г. коды направления: для бакалавров 11.03.03, для магистров 11.04.03), а также продолжает подготовку инженеров по специальностям 090104 и 210202.

С 2017 года кафедрой заведовал к.т.н., доцент Д.А. Заколдаев.

За время своего существования кафедра выпустила более 4750 инженеров, специалистов, бакалавров и магистров. На кафедре защищено 100 кандидатских и 16 докторских диссертаций.

В связи с реорганизацией структуры мегафакультета компьютерных технологий и управления, факультета безопасности информационных технологий, одним из подразделений которых являлась кафедра ПБКС, осуществление руководства направлением подготовки (специальности) 10.03.01 «Информационная безопасность» возложено на отдел «дирекция образовательных программ факультета безопасности информационных технологий».