СРАВНИТЕЛЬНЫЙ АНАЛИЗ И ВЫБОР СТАТИЧЕСКИХ АНАЛИЗАТОРОВ БЕЗОПАСНОСТИ КОДА

Марков А. С.[1], Антипов И. С.[2], Арустамян С. С.[3], Магакелова Н. А.[4]

DOI: 10.21681/2311-3456-2024-5-79-88

Цель работы: *разработка методического подхода к проведению сравнительного анализа статических анализаторов безопасности исходного кода, применимых при сертификации средств защиты информации, по критериям результативности, применимости, функциональности и удобства, а также его демонстрация на примерах.*

Метод исследования: *анализ нормативных и методических документов по проведению статического анализа и по оценке статических анализаторов исходного кода программ с целью формирования методики их сравнения и выбора.*

Полученный результат: *приведены результаты анализа и синтеза системы показателей качества проприетарных анализаторов безопасности кода и анализаторов кода с открытым кодом, а также результаты их сравнения на реальных продуктах, что позволяет сформировать необходимую инструментальную базу сертификационных испытаний программных средств защиты информации по требованиям безопасности информации и сертификации процессов разработки безопасного программного обеспечения.*

Научная новизна: *проанализированы нормативные документы в области статического анализа кода применительно к решению задачи анализа и подбора нескольких статических анализаторов безопасности, приведены критерии выбора, выбраны тестовые продукты и проведен эксперимент, который продемонстрировал различную результативность анализаторов безопасности кода при сертификации.*

Вклад авторов: *Марков А. С. – разработка методического подхода, редактирование, Антипов И. С., Арустамян С. С., Магакелова Н. С. – проведение эксперимента.*

Ключевые слова: *программная безопасность, безопасные программные ресурсы, анализ безопасности программ, уязвимости, недекларированные возможности, программные закладки, инструментарий сертификационных испытаний.*

Введение

Статические анализаторы безопасности кода являются обязательным инструментарием выявления дефектов безопасности и неизвестных уязвимостей как в процессе испытаний программных средств защиты информации [1], так и при оценке соответствия процессов разработки безопасного программного обеспечения [2]. Применение статического анализа при испытаниях, как известно, обусловлено возможностью управления полнотой покрытия кода, принципиальным выявлением закладок (связанных с редко используемыми входными данными), независимостью от среды выполнения, прозрачностью расследований в случае преднамеренных инцидентов, контролем качества кодирования кода, возможностью выявления заимствованных компонент и подобных фрагментов (с целью оценки безопасности и выполнения распараллеливания и прочей оптимизации) и пр. В рамках разработки безопасного программного обеспечения внедрение статических анализаторов возможно на самых ранних стадиях разработки, что радикально снижает затраты на исправление ошибок и уязвимостей на последующих этапах жизненного цикла изделий.

Особенностью отечественного законодательства является замечание по использованию нескольких статических анализаторов кода, что обусловлено не просто борьбой с ошибками 1 и 2 рода, а повышением уровней достоверности и полноты испытаний. В то же время понятно, что различные статические анализаторы отличаются друг от друга множеством параметров, как-то: полнота поддержки базы дефектов (в частности, CWE), интеграция с динамическим анализом и всем жизненным циклом сертификации, удобством, поддержкой языков программирования и сред, возможностью абстрагирования от синтаксиса языка, стоимостью продукта и стоимостью поддержки, наконец, наличием сертификата и т.д. Вопросом комплексного сравнения статических анализаторов, применимых при сертификации средств защиты информации, посвящено данное исследование. Указанная статья родилась не на пустом месте, так как авторы участвовали в обсуждении

1 Марков Алексей Сергеевич, доктор технических наук, заведующий кафедрой 43 НИЯУ МИФИ, Москва, Россия. E-mail:asmarkov@mephi.ru
2 Антипов Илья Сергеевич, эксперт испытательной лаборатории АО «НПО «Эшелон», Москва, Россия. E-mail: mail@cnpo.ru
3 Арустамян Сас Сергеевич, заместитель руководителя испытательной лаборатории АО «НПО «Эшелон», Москва, Россия E-mail:s.arustamyan@npo-echelon.ru
4 Магакелова Нелли Александровна, эксперт испытательной лаборатории АО «НПО «Эшелон», Москва, Россия E-mail:mail@cnpo.ru

международного стандарта по статическим анализаторам – SATEC (Static Analysis Technologies Evaluation Criteria) [3] и проводили ранее подобные изыскания [4, 5], а также участвовали в обсуждении требований к статическим анализаторам по линии технического комитета ТК-362 («Защита информации»). В литературе можно встретить ряд весьма интересных тематических исследований для отдельных языков и приложений [6–12]. Авторы постарались учесть накопленный опыт и представить материал с точки зрения испытательной лаборатории.

Постановка задачи

В настоящее время известно множество способов и техник статического анализа безопасности программ, наиболее популярными из которых можно назвать следующие [1, 13]:

- тестирование по формальным моделям представления программного кода;
- сигнатурный метод выявления и анализа потенциально опасных фрагментов кода (по заранее выявленным шаблонам);
- эвристический метод выявления и анализа потенциально опасных фрагментов кода;
- межпроцедурный контекстно-чувствительный анализ;
- чувствительный к путям выполнения анализ;
- межмодульный анализ;
- анализа потока управления;
- анализ потока данных и др.[5]

Данная работа не имеет целью исследования глубины реализации указанных способов и техник, а ориентирована на повышение эффективности работы испытательной лаборатории, которая руководствуется необходимыми отечественными нормативно-правовыми актами. В этом плане были определены четыре задачи исследования:

1. Оценить результативность применения статических анализаторов;
2. Оценить применимость их в испытательных лабораториях, аккредитованных в различных системах сертификации;
3. Оценить функциональность анализаторов в плане необходимости и достаточности при выполнении сертификационных испытаний;
4. Оценить дополнительные показатели качества, которые бывают полезными и удобными.

В качестве стенда исследования использовался реальный полигон испытательной лаборатории НПО «Эшелон». В качестве объектов испытаний были выбраны пять доступных статических анализаторов безопасности кода, а именно:

- AK-BC 3.0 – коммерческий анализатор, поддерживающий C/C++, C#, Java, Python, PHP;
- Cppcheck – open-source анализатор, поддерживающий C/C++;
- Clang SA – open-source анализатор, поддерживающий C/C++, Objective-C;
- Horusec – open-source анализатор, поддерживающий C#, Java, Kotlin, Python, Ruby, Golang, Terraform, Javascript, Typescript, Kubernetes, PHP, C, HTML, JSON, Dart, Elixir, Shell, Nginx;
- Svace – коммерческий анализатор, поддерживающий C/C++, C#, Java, Kotlin и Go.

Оценка результативности статических анализаторов

С точки зрения программной безопасности базовым ее фактором представляются дефекты безопасности (weaknesses). Поэтому, казалось бы, логичным для оценки безопасности использовать программные тестовые средства с *известными* уязвимостями и сопоставить дефекты. Таким качеством обладают синтетические тесты [4, 5]. С другой стороны, испытательная лаборатория проводит, по сути, исследование кода с целью досконального изучения его уровня безопасности, а не тривиально сканирует программный продукт по базе известных уязвимостей. В данном исследовании из любопытства для тестирования был выбран ряд популярных продуктов с открытым кодом, а именно:

- FFmpeg – open-source коллекция библиотек и инструментов для обработки мультимедийного контента, такого как аудио, видео, субтитры и связанные с ними метаданные;
- Firebird – open-source реляционная база данных, предлагающая множество стандартных функций ANSI SQL, которая работает на Windows, MacOS, Linux и других различных платформах Unix;
- Httpd – open-source веб-сервер, совместимый с HTTP/1.1;
- Librdkafka – open-source реализация библиотеки C протокола Apache Kafka;
- Ruby – это интерпретируемый объектно-ориентированный язык программирования, часто используемый для веб-разработки;
- Unit – легкий и универсальный сервер с открытым исходным кодом;
- Vim – open-source редактор.

Исследование указанных продуктов показало наличие достаточно большого количества некорректностей кода и дефектов безопасности, которые, как известно, классифицируют по каталогу MITRE CWE (Common Weakness Enumeration). Как известно, дефекты потенциально способны привести к уязвимостям, наличие которых может составлять угрозу, что требует дополнительного тестирования продукта

5 ГОСТ Р 71207-2024 «Защита информации. Разработка безопасного программного обеспечения. Статический анализ программного обеспечения. Общие требования». М.: Российский институт стандартизации, 2024. – 20 с.

Таблица 1

Дефекты кода по классификации CWE, обнаруженные статическими анализаторами

Продукт	Статический анализатор кода				
	АК-ВС 3	**Cppcheck**	**Clang SA**	**Horusec**	**Svace**
FFmpeg	14, 114, 134, 259, 394, 398, 480, 511, 563, 569, 570, 571, 628, 665, 667, 672, 690, 758, 798	190, 398, 457, 476, 562, 682, 758, 775, 786, 788	-	2, 20, 12, 36, 37, 78, 119, 120, 126, 134, 190, 327, 352, 362, 367, 676, 798, 807, 82	-
firebird	378, 401, 465, 476, 563, 672	401, 415, 457, 562	-	312, 798	-
httpd	121, 188, 243, 259, 369, 416, 465, 467, 476, 561, 563, 587, 672, 676, 690, 704, 775, 788, 824, 1041	398, 457, 476, 758	-	489	-
librdkafka	14, 369, 398, 401, 416, 465, 476, 561, 563, 569, 570, 571, 670, 676, 761, 763, 770, 824	398, 401, 476, 628, 682, 685, 768	-	2, 12, 20, 36, 37, 78, 119, 120, 134, 190, 312, 327, 676, 704, 798, 807	-
ruby	14, 134, 401, 404, 480, 563, 570, 672, 676, 770	398, 401, 415, 457, 476, 562, 628, 682, 758	-	2, 12, 20, 22, 36, 37, 73, 78, 119, 120, 126, 134, 190, 250, 312, 352, 362, 327, 676, 785, 798, 807, 829	-
unit	401, 416, 465, 476, 561, 563, 570, 763, 824	398, 401, 457, 476, 562	-	2, 12, 20, 22, 36, 37, 73, 89, 119, 120, 134, 190, 209, 250, 327, 330, 362, 367, 539, 611, 676, 704, 785, 798, 807	-
vim	121, 134, 188, 369, 401, 416, 465, 466, 476, 502, 561, 563, 569, 587, 672, 676, 681, 704, 763, 775, 788, 824, 1041	398, 401, 457, 476, 672, 775, 788	-	352, 798	-
«-» – не поддерживает идентификаторы CWE					

в реальных условиях на реальном объекте информатизации.

Дефекты безопасности, которые экспертами испытательной лаборатории признаны достоверными, приведены в табл. 1, которая содержит идентификаторы по MITRE CWE.

Представленная таблица демонстрирует, что тезис экспертов ТК-362 по использованию пары анализаторов при проведении статического анализа, является допустимым подходом, поскольку каждый статический анализатор обладает своими особенностями и алгоритмами.

Легко показать, что продемонстрированный в таблице результат позволяет перейти к количественным (объективным) показателям оценки продукта и планирования испытаний, воспользовавшись *математическими моделями полноты испытаний* [1, 14]. Например, для простоты предположим, что в испытании

Таблица 2.

Соответствие методике ФСТЭК России

Критерий	Статический анализатор кода				
	АК-ВС 3.0	Cppcheck	Clang SA	Horusec	Svace
Основные требования по статическому анализу кода	+	+	+	+	+
Дополнительные требования по статическому анализу*	+	-	+	-	+
* – отдельные требования выполняется экспертным путем					

Таблица 3.

Соответствие руководящему документу Гостехкомиссии России

Критерий		Статический анализатор кода				
		АК-ВС 3.0	Cppcheck	Clang SA	Horusec	Svace
Статический анализ исходных текстов программ	Контроль полноты и отсутствия избыточности исходных текстов ПО (на уровне файлов)	+	-	-	-	-
	Контроль соответствия исходных текстов ПО его объектному (загрузочному) коду	-*	-*	-*	-*	-*
	Контроль связей функциональных объектов по управлению	+	-	-	-	-
	Контроль связей функциональных объектов по информации	+	-	-	-	-
	Контроль информационных объектов	+	-	-	-	-
	Контроль наличия заданных конструкций в исходных текстах	+	-	-	-	-
	Формирование перечня маршрутов выполнения функциональных объектов	+	-	-	-	-
	Анализ критических маршрутов выполнения функциональных объектов	+	-	-	-	-
	Анализ алгоритма работы функциональных объектов на основе блок-схем, диаграмм и т. п., построенных по исходным текстам контролируемого ПО	+	-	-	-	-
	Контроль связей функциональных объектов по управлению	+	-	-	-	-
«-*» – выполняется экспертным путем						

приложения vim участвует две группы экспертов: одна группа использует коммерческий сканер, а вторая – сканеры с открытым кодом. Тогда, опираясь на модель парной оценки, можно предположить, что в продукте еще присутствует 24 дефекта CWE:

$$\bar{N} = N - (N_1 + N_2 - N_{12}) = 23.4, \quad (1)$$

где: N_1 – число дефектов, обнаруженных АК-ВС, N_{12} – число совпавших дефектов, $N = N_1N_2/N_{12}$.

Оценка применимости статических анализаторов

Степень применимости статических анализаторов оценивалась, исходя из реализации ими требований регуляторов (Минобороны России, ФСБ России и ФСТЭК России). Как известно, системы сертификации средств защиты информации опираются на два документа, а именно:

- Методический документ. Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении (утв. ФСТЭК России 2021 г.);
- Руководящий документ. Защита от НСД к информации. Часть 1. ПО СЗИ. Классификация по уровню контроля отсутствия недекларированных возможностей (утв. Гостехкомиссией России 1999 г.).

Методический документ обязателен в сфере компетенции ФСТЭК России, а руководящий документ используется в остальных системах обязательной сертификации средств защиты информации, а также рядом отраслевых регуляторов.

Поэтому в следующих табл. 3 и 4 приведен сравнительный анализ статических анализаторов на предмет выполнения требований указанных документов.

Как видно из табл. 2, ряд статических анализаторов технически применим при испытаниях по требованиям методического документа ФСТЭК России (в части межмодульного анализа, символьного выполнения и анализа, чувствительного к путям выполнения и др.). Заметим, что методический документ ФСТЭК России перекликается с ГОСТ Р 71207-2024 «Защита информации. Разработка безопасного программного обеспечения. Статический анализ программного обеспечения. Общие требования».

Заметим, что в данном исследовании мы ограничились требованиями именно по статическому анализу, в то время как комплексный продукт (в данном случае АК-ВС) может поддерживать весь цикл сертификационных испытаний.

Оценка функциональности и удобства статических анализаторов

Хорошие практики информационной безопасности утверждают, что вопросы безопасности должны быть в гармонии еще с двумя свойствами, как-то: функциональностью и удобством [15]. Исходя из этого, авторы воспользовались популярными зарубежными стандартами, касающимися качества статических анализаторов, а именно:

1. NIST 500-268. Source Code Security Analysis Tool Function Specification [16];
2. Static Analysis Technologies Evaluation Criteria [17].

Стандарт NIST SP 500-268 определяет требования к работе анализаторов исходных текстов, что позволяет выявить минимальный набор возможностей анализатора.

SATEC является метастандартом и содержит подробный список «общих» критериев оценки, которые могут быть основой для формирования гибких частных требований!

Сравнительный анализ по критериям функциональности и удобства включал следующие пункты оценки:

- наличие минимально необходимых возможностей (согласно NIST);
- наличие дополнительных возможностей (согласно SATEC);

Результаты указанных оценок представлены соответственно в табл. 4 и 5.

Следует отметить, что существует три способа управления результатами (отчетами) статического анализа: группировка и сортировка, удаление ненужных результатов, комментарии к результатам работ статического анализатора. Группировка и сортировка позволяет, среди прочего, пометить группу триггерных событий как «ложноположительные» (false positive), таким образом, нет необходимости отмечать каждое триггерное событие отдельно.

Сравнив данные табл. 4 и 5, можно сделать вывод, что коммерческие анализаторы справляются с задачей предоставления результатов анализа кода в удобной для экспертов форме. Большинство анализаторов кода позволяют группировать и сортировать результаты, переопределять ложноположительные триггерные события, в частности, АК-ВС 3, по мнению авторов, имеет возможность получать удобные отчеты с разным уровнем детализации [17, п. 6.1].

Надо понимать, что доверие к сертификации основано на объяснении результатов. В этом плане АК-ВС 3.0, Cppcheck и Horusec соответствуют ожиданиям, так как поддерживают международный каталог дефектов CWE. Указанное позволяет подстроиться под систематику ФСТЭК России (угрозы по БДУ – уязвимости по БДУ – дефекты CWE).

Следует отметить важность возможности настройки набора правил проверки статических анализаторов, что позволяет рационально использовать временные и материальные ресурсы испытательной лаборатории. В данном случае мы можем говорить не только о результативности испытаний, но и об эффективности.

Таблица 4.

Соответствие руководящему документу Гостехкомиссии России

Критерий	Статический анализатор кода				
	АК-ВС 3.0	**Cppcheck**	**Clang SA**	**Horusec**	**Svace**
Формирование высокоуровневого отчета [16, п. 2.1]					
Определение выбранного класса дефектов	+	+	+ (CWE не пишется в явном виде)	+	+ (CWE не пишется в явном виде)
Сообщение о типе и месте дефекта	+	+	+	+	+
Создание отчета, совместимого с другими инструментами	+ (csv)	+ (.txt, xml)	-	+ (Sonarqube)	+ (Sarif, csv)
Возможность отключения сообщений о выбранных дефектах	+ (через комментарии в исходном коде)	+ (поштучно или по классам)	+ (по файлам)	+ (сложная настройка)	-
Использование общепринятых имен для классов дефектов	+	+	-	+	-
Обязательные требования [16, п. 2.2]					
SCSA-RM-1: Определение классов дефектов по NIST	+	+	-	+	-
SCSA-RM-2: Текстовый отчет о любых дефектах	+	+	+	+	+
SCSA-RM-4: Возможность указания каталога, файла и номера строки для дефекта	+	+	+	+	+
Дополнительные требования [16, п. 2.3]					
SCSA-RO-1: Возможность создания отчета в формате XML	-	+	-	-	-
SCSA-RO-3: Указание номера и класса CWE	+	+	-	+	-

Таблица 5.

Расширенные возможности статических анализаторов

Критерий	Статический анализатор кода				
	АК-ВС 3.0	**Cppcheck**	**Clang SA**	**Horusec**	**Svace**
Поддержка командной строки [17, п. 3.1]					
Поддержка командной строки	+	+	+	+	+
Поддержка интеграции с IDE [17, п. 3.2]					
Указание, какие IDE (и версии) поддерживаются, и что включает в себя проверка кода с помощью IDE	+	+	+	+	+
Системная поддержка [17, п. 3.3]					
Перечисление поддерживаемых систем сборки и их версии	+	+	-	+	+

Индивидуальная настройка [17, п. 3.4]					
Возможность добавить, удалить, изменить основные сигнатуры	-	+	-	+	-
Возможность создания авторских сигнатур	-	+	-	+	-
Возможности конфигурирования анализом [17, п. 3.5]					
Возможность планирования проверки	-	-	-	-	-
Возможность просмотра статуса выполняемых проверок в режиме РВ	+	+	+	+	+
Возможность сохранять конфигурации и повторно использовать их в качестве шаблонов конфигурации	-	+	+	-	-
Возможность одновременного выполнения нескольких проверок	+	-	+	+	+
Возможность поддержки нескольких пользователей	+	-	+	+	+
Возможность выполнения инкрементальных проверок	-	+	+	-	+
Возможности тестирования по безопасности [17, п. 3.6]					
Возможность точно идентифицировать и сообщать о возможных атаках и уязвимостях безопасности	+	+	+	+	+
Поддержка отчетов по ролям [17, п. 6.1]					
Представление краткого изложения результатов проверки на высоком уровне (для руководства)	+	+	+	+	+
Предоставление технического детального отчета: ■ с кратким описанием проблемы, включая категорию дефектов	+	-	-	+	-
■ с указанием местонахождения проблемы, включая имя файла и номер строки кода	+	+	+	+	+
■ с рекомендациями по устранению, которые должны быть настроены для каждой проблемы и включают примеры кода на выбранном языке	-	+	-	-	-
■ с подробной информацией о потоке, которая показывает поток помеченных (tainted) данных от источника к получателю	+	+	+	-	+
Персонализация отчетов [17, п. 6.2]					
Возможность включить в отчет комментарии экспертов	+	-	-	+	-
Возможность отмечать результаты как ложные срабатывания и удалять их из отчета	-	-	-	+	-
Форматы отчетов [17, п. 6.3]					
Поддерживаемые форматы отчетов (PDF, XML, HTML и т. д.)	+ (html, csv)	+ (xml, html)	+ (html)	+ (html, json)	+ (pdf, csv, json)
Поддержка корпоративного уровня [17, п. 7]					
Интеграция в системы отслеживания ошибок	+	+	+	+	+
Наличие платной лицензии (поддержка)	+	+	-	-	+

Выводы по эксперименту

Для визуализации результатов сравнений статических анализаторов использована лепестковая диаграмма (рис. 1), где ее параметрами являются экспертная оценка показателей *применимости* (полнота соответствия документам регуляторов), *функциональности* (выполнимости необходимых и достаточных требований) и *удобства* (реализации дополнительных удобных функций). Материалы первой таблицы (*результативность*) решили не выносить на рисунок, так как два статических анализатора в принципе не поддерживают CWE (оценку об обязательности этого критерия оставим на решение экспертов органов по сертификации), ну и выбор тестовых продуктов выполнен экспертным путем, то есть обладает известной степенью субъективизма.

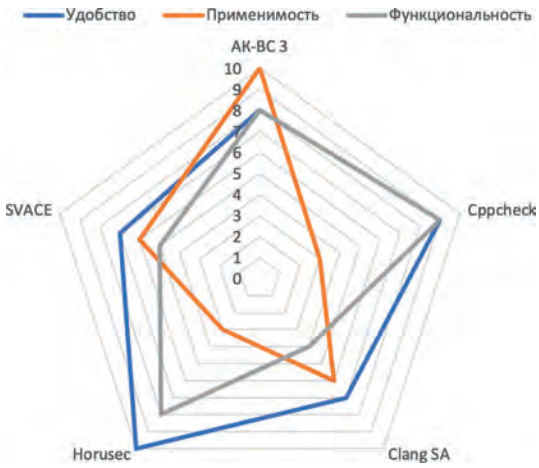

Рис. 1. Диаграмма выбора статического анализатора

Заключение

Использование статических анализаторов безопасности программного кода является необходимой процедурой оценки соответствия программного обеспечения по требованиям безопасности информации. Востребованность в таком инструментарии растет в связи с внедрением процедур разработки безопасного программного обеспечения и сертификации соответствующих процессов [2].

Приведенное исследование показало, что на сегодня оправдано указание по использованию нескольких статических анализаторов в процессе сертификации. Такой подход может быть легко обоснован математически [14].

Авторы полагают, что, с точки зрения испытательной лаборатории, предпочтителен унифицированный подход, как в плане формирования единой интегрированной среды всех сертификационных проверок и отчетов, так и поддержки концепции БДУ ФСТЭК России (BDU-CWE). В статье отмечен большой пул параметров, которые можно использовать при детальном сравнении (например, поддержка языков, наличие сертификата и поддержки). Интересен и подход SATEC к созданию метастандарта.

Проведенный эксперимент не претендует на абсолютную полноту, однако демонстрирует допустимость предложенного методического подхода. Исследование было сфокусировано именно на сертификационных испытаниях, то есть обсуждение ряда анализаторов кода, используемых при аудите отдельных программных приложений, вышло за рамки исследования.

В заключение следует сказать, что нормативно-методическое направление анализа безопасности программ находится в активном развитии и, благодаря деятельности ТК-362, обсуждаются российские требования к оценке анализаторов кода. Авторы надеются, что данный методический подход и эксперимент будут любопытны всем лицам, увлекающимся разработкой и оценкой безопасных продуктов.

Литература

1. *Марков А. С., Цирлов В. Л., Барабанов А. В. Методы оценки несоответствия средств защиты информации. М.: Радио и связь, 2012. 192 с.*
2. *Арустамян С. С., Варенница В. В., Марков А. С. Методические и реализационные аспекты внедрения процессов разработки безопасного программного обеспечения // Безопасность информационных технологий. 2023. Т. 30. № 2. С. 23–37.*
3. *Static Analysis Technologies Evaluation Criteria v1.0./Ed. by Sherif Koussa; Russian translation by Alec Shcherbakov and Alexey Markov, Web Application Security Consortium, 2013. Режим доступа: http://projects.webappsec.org/w/page/71979863/Static%20Analysis%20Technologies%20Evaluation %20Criteria %20-%20Russian/.*
4. *Марков А. С., Фадин А. А., Швец В. В. Сравнение статических анализаторов безопасности программного кода // Защита информации. Инсайд. 2015. № 6 (66). С. 38–47.*
5. *Markov A., Fadin A., Shvets V., Tsirlov V. The Experience of Comparison of Static Security Code Analyzers // International Journal of Advanced Studies. 2015. V. 5. N 3. С. 55–63.*
6. *Галатенко В. А., Костюхин К. А., Шмырев Н. В., Аристов М. С. Использование свободно распространяемых средств статического анализа исходных текстов программ в процессе разработки приложений для операционных систем реального времени // Программная инженерия. 2012. № 5. С. 2–5.*
7. *Пономарев Н. С., Таланов К. Е. Исследование особенностей анализаторов кода на выявление уязвимостей с использованием метода анализа иерархий Т. Л. Саати. В сборнике: XXXVI Международные Плехановские чтения. Сборник статей участников конференции. В 4-х томах. Москва, 2023. С. 232–238.*

8. Федоров А. Ю., Портнов Е. М., Кокин В. В. Исследование возможностей статических анализаторов кода по поиску ошибок памяти в языках C/C++ // Информатизация и связь. 2017. № 4. С. 45–49.
9. Fatima A. and etc. Comparative study on static code analysis tools for C/C++, In: 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2018, pp. 465–469, DOI: 10.1109/IBCAST.2018.8312265.
10. Kuszczyński K., Walkowski M. Comparative Analysis of Open-Source Tools for Conducting Static Code Analysis // Sensors. 2023. V. 23. № 18. P. 79–78.
11. Shaukat R. and etc. Probing into code analysis tools: A comparison of C# supporting static code analyzers. In: 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2018, pp. 455–464, DOI: 10.1109/IBCAST.2018.8312264.
12. Stefanović D., Nikolić D., Dakić D., Spasojević I., Ristić S. Static Code Analysis Tools: A Systematic Literature Review // In: Annals of DAAAM and Proceedings of the International DAAAM Symposium. 31. 2020. P. 565–573.
13. Бударный Г. С., Пестов И. Е., Штеренберг И. Г. Сравнение методов статического анализа исходного кода программы // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки. 2024. № 1. С. 5–12.
14. Марков А. С. Модели оценки и планирования испытаний программных средств по требованиям безопасности информации. Вестник МГТУ им. Н. Э. Баумана. Сер. «Приборостроение». 2011. Спецвыпуск «Технические средства и системы защиты информации». С. 90–103.
15. Барабанов А. В., Дорофеев А. В., Марков А. С., Цирлов В. Л. Семь безопасных информационных технологий / Под. ред. А. С .Маркова. М.: ДМК Пресс, 2017. 224 с.
16. NIST 500-268. Source Code Security Analysis Tool Function Specification / Black P. E, Kass M., Koo M. Fong E. SSD ITL NIST, 2011. v.1.1. 14 p.
17. Static Analysis Technologies Evaluation Criteria / Ed. by Sherif Koussa – Web Application Security Consortium, 2013. v.1.0. 19 p.

COMPARATIVE ANALYSIS OF STATIC CODE SAFETY ANALYSERS

Markov A. S.[6], Antipov I. S.[7], Arustamyan S. S.[8], Magakelova N. A.[9]

Purpose of work: *development of a methodical approach to comparative analysis of static source code security analysers applicable to the certification of information protection tools by the criteria of performance, applicability, functionality and convenience, as well as its demonstration on examples.*

Research method: *analysis of normative and methodical documents on conducting static analysis and on evaluating static analysers of software source code in order to form a method of their comparison and selection.*

Obtained result: *the results of analysis and synthesis of the system of quality indicators of proprietary code safety analyzers and opensource code analyzers are given, as well as the results of their comparison on real products, which allows to form the necessary tool base for certification tests of software information protection means on information safety requirements and certification of safe software development processes.*

Scientific novelty: *normative documents in the field of static code analysis are analysed in relation to the solution of the task of analysis and selection of several static security analysers, selection criteria are given, test products are chosen and an experiment is carried out which demonstrated different efficiency of code security analysers during certification.*

Authors' contribution: *Markov A. S. – development of methodical approach, editing, Varenitsa V. V. – development of test bench architecture, Antipov I. S., Arustamyan S. S., Magakelova N. S. – conducting the experiment.*

Keywords: *software security, secure software resources, software security analysis, vulnerabilities, undeclared capabilities, backdoors, certification testing toolkit.*

References

1. Markov A. S., Cirlov V. L., Barabanov A. V. Metody ocenki nesootvetstviya sredstv zashchity informacii. M.: Radio i svyaz', 2012. 192 s.
2. Arustamjan S. S., Varenica V. V., Markov A. S. Metodicheskie i realizacionnye aspekty vnedrenija processov razrabotki bezopasnogo programmnogo obespechenija // Bezopasnost' informacionnyh tehnologij. 2023. T. 30. № 2. S. 23–37.
3. Static Analysis Technologies Evaluation Criteria v1.0./Ed. by Sherif Koussa; Russian translation by Alec Shcherbakov and Alexey Markov, Web Application Security Consortium, 2013. – Rezhim dostupa: http://projects.webappsec.org/w/page/71979863/Static%20Analysis%20Technologies%20Evaluation%20Criteria%20-%20Russian/.
4. Markov A. S., Fadin A. A., Shvec V. V. Sravnenie staticheskih analizatorov bezopasnosti programmnogo koda // Zashhita informacii. Insajd. 2015. № 6 (66). S. 38–47.

6 Aleksey S. Markov, Dr.Sc., Head of Department 43, Moscow, Russsia. E-mail: asmarkov@mephi.ru
7 Ilya S. Antipov, Expert of testing laboratory of Echelon, Moscow, Russia. E-mail: mail@cnpo.ru
8 Sas S. Arustamyan, Deputy head of testing laboratory of Echelon, Moscow, Russia. E-mail: s.arustamyan@npo-echelon.ru
9 Nelly A Magakelova, Expert of testing laboratory of Echelon, Moscow, Russia. E-mail: mail@cnpo.ru

5. *Markov A., Fadin A., Shvets V., Tsirlov V. The Experience of Comparison of Static Security Code Analyzers // International Journal of Advanced Studies. 2015. V. 5. N 3. S. 55–63.*
6. *Galatenko V. A., Kostjuhin K. A., Shmyrev N. V., Aristov M. S. Ispol'zovanie svobodno rasprostranjaemyh sredstv staticheskogo analiza ishodnyh tekstov programm v processe razrabotki prilozhenij dlja operacionnyh sistem real'nogo vremeni // Programmnaja inzhenerija. 2012. № 5. S. 2–5.*
7. *Ponomarev N. S., Talanov K. E. Issledovanie osobennostej analizatorov koda na vyjavlenie ujazvimostej s ispol'zovaniem metoda analiza ierarhij T. L. Saati. V sbornike: XXXVI Mezhdunarodnye Plehanovskie chtenija. Sbornik statej uchastnikov konferencii. V 4-h tomah. Moskva, 2023. S. 232–238.*
8. *Fedorov A. Ju., Portnov E. M., Kokin V. V. Issledovanie vozmozhnostej staticheskih analizatorov koda po poisku oshibok pamjati v jazykah S/S++ // Informatizacija i svjaz'. 2017. № 4. S. 45–49.*
9. *Fatima A. and etc. Comparative study on static code analysis tools for C/C++, In: 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2018, pp. 465–469, DOI: 10.1109/IBCAST.2018.8312265.*
10. *Kuszczyński K., Walkowski M. Comparative Analysis of Open-Source Tools for Conducting Static Code Analysis // Sensors. 2023. V. 23. № 18. P. 7978.*
11. *Shaukat R. and etc. Probing into code analysis tools: A comparison of C# supporting static code analyzers. In: 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2018, pp. 455–464, DOI: 10.1109/IBCAST. 2018.8312264.*
12. *Stefanović D., Nikolić D., Dakić D., Spasojević I., Ristić S. Static Code Analysis Tools: A Systematic Literature Review // In: Annals of DAAAM and Proceedings of the International DAAAM Symposium. 31. 2020. P. 565–573.*
13. *Budarnyj G. S., Pestov I. E., Shterenberg I. G. Sravnenie metodov staticheskogo analiza ishodnogo koda programmy // Vestnik Sankt-Peterburgskogo gosudarstvennogo universiteta tehnologii i dizajna. Serija 1: Estestvennye i tehnicheskie nauki. 2024. № 1. S. 5–12.*
14. *Markov A. S. Modeli ocenki i planirovanija ispytanij programmnyh sredstv po trebovanijam bezopasnosti informacii. Vestnik MGTU im.N. Je. Baumana. Ser. «Priborostroenie». 2011. Specvypusk «Tehnicheskie sredstva i sistemy zashhity informacii». S.90–103.*
15. *Barabanov A. V., Dorofeev A. V., Markov A. S., Cirlov V. L. Sem' bezopasnyh informacionnyh tehnologij/Pod. red. A.S .Markova. M.: DMK Press, 2017. 224 s.*
16. *NIST 500-268. Source Code Security Analysis Tool Function Specification / Black P. E, Kass M., Koo M. Fong E. – SSD ITL NIST, 2011. – v.1.1. – 14 p.*
17. *Static Analysis Technologies Evaluation Criteria / Ed. by Sherif Koussa - Web Application Security Consortium, 2013. – v.1.0. – 19 p.*