ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ТЕХНИЧЕСКОМУ И ЭКСПОРТНОМУ КОНТРОЛЮ

Утвержден ФСТЭК России
12 мая 2026 г.

МЕТОДИЧЕСКИЙ ДОКУМЕНТ

МЕТОДИКА ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ И НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

(Выписка)

МОСКВА
2026

1. Общие положения

1.1. Настоящая Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении (далее – Методика) разработана в соответствии с подпунктом 4 пункта 8 Положения о Федеральной службе по техническому и экспортному контролю, утвержденного Указом Президента Российской Федерации от 16 августа 2004 г. № 1085.

1.2. Методика определяет порядок проведения исследований по выявлению уязвимостей и недекларированных возможностей (далее – НДВ) в соответствии с Требованиями по безопасности информации, устанавливающими уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий, утвержденными приказом ФСТЭК России от 2 июня 2020 г. № 76 (зарегистрирован Минюстом России 11 сентября 2020 г., регистрационный № 59772) (далее – Требования доверия) в программном обеспечении средств защиты информации (далее – объект оценки, ОО), за исключением исходного программного кода программируемых логических интегральных микросхем.

1.3. Методика ориентирована на проведение исследований, проводимых испытательными лабораториями и разработчиками в рамках сертификационных испытаний программных, программно-аппаратных средств защиты информации и защищенных программных, программно-технических средств, а также при внесении изменений в ранее сертифицированные средства. Разработчикам ОО рекомендуется использовать положения Методики для организации внутренних процессов жизненного цикла программного обеспечения в соответствии с ГОСТ Р 56939-2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования» (далее – ГОСТ Р 56939-2024).

1.4. В Методике применяются термины и определения, установленные национальными стандартами в области защиты информации и обеспечения информационной безопасности.

1.5. В связи с утверждением настоящего методического документа не применяются положения методического документа «Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении», утвержденного ФСТЭК России 25 декабря 2020 г.

2. Общие требования к проведению исследований по выявлению уязвимостей и недекларированных возможностей

2.1. Исследования ОО проводятся в целях выявления недостатков безопасности ОО: уязвимостей архитектуры, кода и конфигурации ОО, недекларированных возможностей, а также потенциально опасных декларированных возможностей ОО (избыточных внешних интерфейсов или функций).

2.2. Исследования ОО проводятся в соответствии с уровнями контроля, которые определяют состав и содержание проводимых исследований, методы исследований и требования к применяемым при проведении исследований инструментальным средствам анализа и контроля.

Устанавливается 6 уровней контроля. Самый низкий уровень – шестой, самый высокий – первый.

Требования к уровню контроля ОО предъявляются в соответствии с Требованиями доверия.

2.3. При проведении исследований ОО используются:

а) документация на ОО, предусмотренная Требованиями доверия;

б) исходный код ОО;

в) сборочная среда и система сборки ОО, включая их документацию;

г) дистрибутив ОО;

д) результаты выполнения процессов разработки ОО в соответствии с ГОСТ Р 56939-2024, включающие результаты анализа безопасности архитектуры и определения поверхности атаки, композиционного анализа, анализа безопасности конфигураций, статического анализа исходного кода, фаззинг-тестирования, тестирования на проникновение, результаты реализации программ по поиску уязвимостей ОО (Bug Bounty), если таковые проводились, результаты интеграционного, функционального, модульного тестирования, в том числе демонстрирующего выполнение заявленных функции безопасности (далее – ФБ);

е) перечень программных компонентов и образов контейнеров ОО (объем и форма представления перечней приведены в Приложении 1 к настоящей Методике);

ж) системные (интеграционные), функциональные, регрессионные, модульные тесты, фаззинг-тесты, включая специально сформированные синтетические, тесты, демонстрирующие выполнение заявленных ФБ ОО, а также описание конфигураций инструментальных средств анализа и контроля и иную тестовую документацию (предоставленная тестовая документация должна демонстрировать прослеживаемость заявленных ФБ и функциональных тестов и содержать описание целей тестирования, тестовых

процедур и ожидаемых результатов, а также фактические результаты тестирования (например, журналы регистрации событий или скриншоты);

з) план поддержки безопасности заимствованных компонентов сертифицированного ОО (в случае внесения изменения в сертифицированный ОО);

и) заданные и описанные в эксплуатационной документации разработчиком ОО в контейнерном исполнении списки действий (правила), разрешенные при взаимодействии компонентов ОО (контейнеров, микросервисов, иных ресурсов, характерных для выбранного типа оркестратора) между собой, с компонентами среды функционирования с внешними по отношению к ОО компонентами;

к) методики анализа поверхности атаки заимствованных компонентов с открытым исходным кодом, опубликованные на сайте Центра исследований безопасности системного программного обеспечения (далее – Центр исследований) в разделе «Методики анализа поверхности атаки» (portal.linuxtesting.ru);

л) методики статического анализа заимствованных компонентов с открытым исходным кодом, опубликованные на сайте Центра исследований в разделе «Методики проведения статического анализа» (portal.linuxtesting.ru);

м) методики тестирования заимствованных компонентов с открытым исходным кодом, опубликованные на сайте Центра исследований в разделе «Методики проведения тестирования» (portal.linuxtesting.ru);

н) методики фаззинг-тестирования заимствованных компонентов с открытым исходным кодом, опубликованные на сайте Центра исследований в разделе «Методики проведения фаззинг-тестирования» (portal.linuxtesting.ru).

2.4. Исследования ОО представлены на рисунке 1 и включают:

а) подготовку к проведению исследований ОО;

б) проведение исследований ОО;

в) оформление результатов исследований ОО.

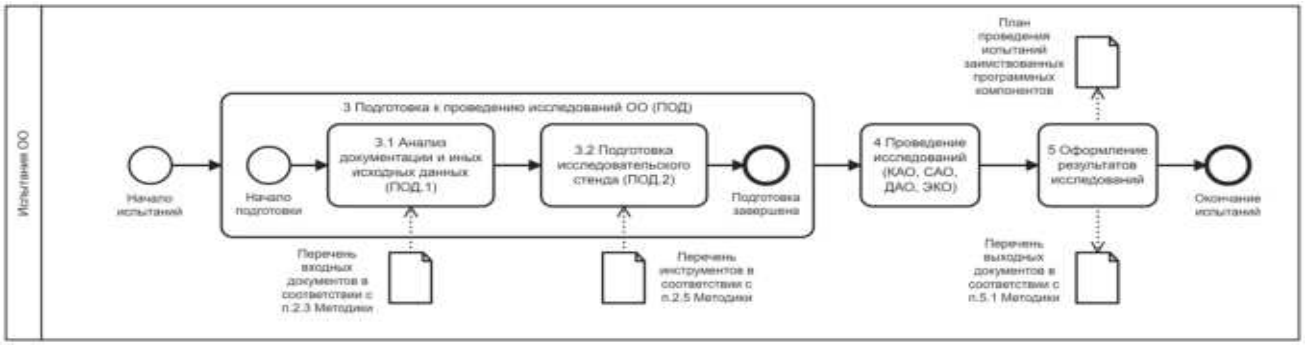

Рисунок 1 – Порядок проведения исследований

2.5. При проведении исследований применяются инструментальные

средства анализа и выявления ошибок, уязвимостей и НДВ, функциональные возможности которых обеспечивают реализацию положений настоящей Методики, не имеющие каких-либо ограничений по применению и адаптации (доработке) на территории Российской Федерации.

2.6. Исследования ОО по выявлению уязвимостей и НДВ в соответствии с положениями настоящей Методики проводятся специалистами испытательной лаборатории (далее – испытательная лаборатория)[1], а также специалистами организации-разработчика, ответственными за обеспечение разработки безопасного программного обеспечения, с привлечением специалистов, непосредственно участвующих в разработке ОО (далее – разработчик ОО).

2.7. Выявленные уязвимости архитектуры, кода и конфигурации ОО, НДВ, а также потенциально опасные декларированные возможности допускается устранять в ходе проведения исследования ОО.

По результатам устранения уязвимостей и НДВ повторное исследование ОО требуется выполнять в отношении интерфейсов, компонентов, модулей и исходного кода, прямо или косвенно затронутых внесенными изменениями, в объеме требований настоящей Методики.

[1] В случае если организация-разработчик имеет сертификат соответствия процессов безопасной разработки программного обеспечения средств защиты информации требованиям национального стандарта ГОСТ Р 56939-2024, в качестве специалистов испытательной лаборатории рассматриваются специалисты указанной организации-разработчика, ответственные за обеспечение разработки безопасного программного обеспечения.

3. Подготовка к проведению исследований по выявлению уязвимостей и недекларированных возможностей (ПОД)

В ходе подготовки к проведению исследований по выявлению уязвимостей и НДВ объекта оценки должны выполняться:

а) анализ документации и иных исходных данных (ПОД.1);

б) подготовка исследовательского стенда (ПОД.2).

Перечень исследований при подготовке к проведению исследований по выявлению уязвимостей и НДВ объекта оценки представлен в таблице 1.

Таблица 1

Шифр	Наименование исследований	Уровень контроля		
		6	5	4
ПОД.1	Анализ документации и иных исходных данных	+	1	1, 2, 3, 4, 5
ПОД.2	Подготовка исследовательского стенда	+	1	1, 2
Примечание: «+» – исследования подлежат выполнению для соответствующего уровня контроля; «цифра» – дополнительные требования к исследованиям для соответствующего уровня контроля				

3.1. Анализ документации и иных исходных данных (ПОД.1)

Задачами исследования являются:

а) формирование представления о:

функциональных возможностях и ФБ ОО;

условиях безопасной эксплуатации, режимах и параметрах функционирования ОО;

структуре ОО на уровне интерфейсов, компонентов, модулей, файлов, структурных элементов кода;

сторонних программных компонентах ОО;

параметрах и особенностях функционирования сборочной среды и системы сборки ОО;

б) оценка состава и полноты полученных для испытуемого ОО результатов выполнения процессов разработки в соответствии с ГОСТ Р 56939-2024, иными национальными стандартами в области информационной безопасности;

в) разработка методики проведения исследований.

Исходные данные для проведения исследования предоставляются в соответствии с требованиями пункта 2.3 настоящей Методики.

Требования к проведению исследования:

В ходе анализа документации и иных исходных данных проверяются исходные данные на соответствие требованиям настоящей Методики в части достоверности, полноты, актуальности, согласованности (отсутствие противоречий).

При проведении исследований оцениваются представленные разработчиком ОО результаты анализа ОО, проводимого в соответствии с ГОСТ Р 56939-2024, иными национальными стандартами в области защиты информации, с целью использования в качестве материалов исследований на предмет:

а) правильности выбора и настройки разработчиком средств выявления уязвимостей и НДВ;

б) соответствия полноты проводимых исследований требованиям уровня контроля, включая полноту набора анализируемых функциональных возможностей, интерфейсов, модулей и подпрограмм (функций);

в) правильности интерпретации результатов, полученных от средств выявления уязвимостей и НДВ, полноты и однозначности разметки предупреждений.

Испытательной лабораторией оценивается полнота поверхности атаки[2] ОО, представленной разработчиком в исходных данных. Для этого выполняется анализ поверхности атаки ОО с учетом методик анализа поверхности атаки заимствованных компонентов с открытым исходным кодом, указанных в пункте 2.3 настоящей Методики, и проводится сопоставление полученных результатов.

Входными данными для анализа поверхности атаки являются декларируемые перечни аппаратных, программных и пользовательских интерфейсов, доступных для атаки потенциальным нарушителям.

Анализ поверхности атаки представляет собой следующие действия:

для пользовательских и аппаратных интерфейсов в ОО определяются соответствующие программные интерфейсы ОО, которые объединяются с входным перечнем программных интерфейсов;

для каждого программного интерфейса из объединенного перечня определяются модули ОО, в которых реализуется обработка контролируемых потенциальным нарушителем входных данных.

В качестве минимальных требований к полноте поверхности атаки ОО

---

[2] Поверхность атаки (программного обеспечения) – множество подпрограмм (функций) программного обеспечения, обрабатывающих данные из интерфейсов, непосредственно или косвенно подверженных потенциальному риску атаки.

выступают включенные в ее состав подпрограммы (функции):

а) непосредственно доступные для вызова потенциальным нарушителям[3];

б) входные данные, которые потенциальный нарушитель способен гарантированно сформировать определенным образом через доступные ему интерфейсы.

К интерфейсам ОО, непосредственно доступным для атаки потенциальным нарушителям, рекомендуется относить служебные интерфейсы ОО, в том числе функционирующие ограниченно по времени, доступные периодически, отладочные интерфейсы, интерфейсы удаленного управления, интерфейсы обновления ОО или используемых баз данных[4].

К числу интерфейсов, доступных для атаки потенциальным нарушителям в ходе штатного функционирования, не относят интерфейсы, доступные только пользователям, успешно прошедшие идентификацию и аутентификацию и обладающим максимальными привилегиями[5] из административных ролей ОО. Данное правило не применяется, если набор привилегий административной роли (в том числе наиболее привилегированной) является настраиваемым.

Интерфейсы, доступные пользователям, обладающим административной ролью, допускающей настройку набора привилегий, относятся к интерфейсам ОО, доступным для атаки потенциальным нарушителем из числа наименее привилегированных администраторов[6].

Интерфейсы, реализующие шифрование в соответствии с требованиями ФСБ России, не относятся к интерфейсам ОО, доступным для атаки потенциальным нарушителям. При этом интерфейсы, которые становятся доступными посредством использования интерфейсов, реализующих шифрование, относятся к интерфейсам ОО, доступным для атаки потенциальным нарушителям[7].

---

[3] В качестве потенциальных нарушителей рассматриваются в том числе все штатные пользователи ОО, обладающие произвольными ролями, за исключением оговоренных в пункте 3.1 настоящей Методики частных случаев.

[4] Например, интерфейс ОО в виде TCP-сокета, однократно создаваемый ОО раз в сутки с целью сетевого доступа к централизованному репозиторию решающих правил и их загрузки, интерфейс периодического доступа к службе точного времени, интерфейс доступа к DHCP-серверу, интерфейс доступа к LDAP-каталогу и другие интерфейсы.

[5] Например, «главный администратор», «суперпользователь», «root» и аналогичные роли, в том числе – отвечающие за развертывание и приведение ОО в сертифицированное состояние в соответствии с эксплуатационной документацией ОО.

[6] Например, к сетевому веб-интерфейсу администрирования ОО имеют доступ все пользователи, обладающие ролью «Администратор». При этом имеется возможность доступа пользователя к конкретным разделам административного портала: анализ записей в хранилище инцидентов, прямое взаимодействие с базой решающих правил, возможность создания новых ролей/пользователей и другие возможности, определяемые набором привилегий, доступных конкретному пользователю.

[7] Например, сетевой веб-интерфейс администрирования ОО, доступный пользователям сети, доступ к которому защищается с использованием СКЗИ, тем не менее, является поверхностью атаки

По результатам анализа документации и иных исходных данных оценивается состав и полнота полученных для испытуемого ОО результатов выполнения процессов разработки в соответствии с ГОСТ Р 56939-2024 в целях учета этих результатов при составлении методики проведения исследований.

При проведении дальнейших исследований ОО испытательная лаборатория контролирует перечень анализируемых интерфейсов и модулей ОО с целью выявления недекларированных интерфейсов, модулей и реализуемого ими функционала, в том числе неиспользуемого кода и неиспользуемых функциональных возможностей ОО.

Дополнительные требования к исследованиям (усиления)

1. Испытательной лабораторией выполняется анализ сборочной среды и системы сборки ОО на предмет наличия инструментов и действий по модификации исходного кода, в том числе интерпретируемого кода, байт-кода или машинного кода ОО, помимо стандартных средств применяемых систем программирования, таких как компилятор и компоновщик.

Разработчик ОО предоставляет описание вносимых модификаций и обосновывает, что они не приводят к возникновению недостатков безопасности ОО. Испытательная лаборатория оценивает представленное описание, в том числе его достоверность, полноту, актуальность, согласованность, и по результатам принимает решение о возможности проведения дальнейших исследований ОО.

Если результатом модификации исходного кода является код на языке программирования высокого уровня, то в отношении модулей, затронутых данной модификацией, необходимо провести статический анализ как исходного кода, так и модифицированного кода в соответствии с требованиями пункта 4.2 настоящей Методики.

Если результатом модификации кода модуля ОО является интерпретируемый код (байт-код) или машинный код, то все затронутые модули ОО должны быть подвергнуты экспертизе в соответствии с требованиями пункта 4.4 настоящей Методики. Если суммарный объем байт-кода или машинного кода, полученного в результате модификации, превосходит 10 000 инструкций (команд), испытательная лаборатория принимает решение о невозможности проведения дальнейших исследований ОО.

2. В ходе исследований проверяется, что при наличии в составе поверхности атаки ОО функций, написанных на интерпретируемых языках

для потенциального внутреннего нарушителя из числа пользователей и наименее привилегированных администраторов ОО за исключением модулей, непосредственно входящих в состав средства криптографической защиты информации.

программирования или на языках программирования, компилируемых в промежуточное представление, разработчик ОО включил в перечень анализируемых модулей соответствующие модули среды исполнения применяемых языков программирования[8]. При отсутствии в поверхности атаки требуемых модулей среды исполнения испытательная лаборатория самостоятельно добавляет их в перечень анализируемых модулей.

3. При проведении исследований проверяется, что разработчик ОО включил в перечень анализируемых модулей модули, реализующие интерфейсы ОО, доступные периодически, в том числе служебные интерфейсы ОО, отвечающие за периодические обновления ОО с удаленных источников. При отсутствии этих модулей испытательная лаборатория самостоятельно добавляет их в перечень анализируемых модулей.

4. Испытательной лабораторией проводится проверка того, что разработчик ОО выделил в подлежащих анализу модулях перечень критичных для безопасности участков кода, в который включил участки кода, взаимодействующие с данными, поступающими от потенциального нарушителя (парсеры, анализаторы, в том числе в составе кода веб-приложений), и имеющие высокую сложность[9]. Разработчик ОО предоставляет обоснование выбора выделенных участков кода с описанием методики оценки сложности кода.

При отсутствии или недостаточной обоснованности выбора испытательная лаборатория самостоятельно корректирует перечень критичных участков кода.

5. В ходе исследований проверяется полнота перечня анализируемых модулей с использованием инструментальных средств определения поверхности атаки. При выявлении неполноты испытательная лаборатория самостоятельно расширяет состав анализируемых модулей.

Требования к результатам исследования

На основе собранных и проанализированных сведений испытательной лабораторией разрабатывается методика проведения исследований в соответствии с Положением о системе сертификации средств защиты информации, утвержденным приказом ФСТЭК России от 3 апреля 2018 г. № 55. В методике по каждому виду исследований определяются состав и содержание исследований, средства анализа и их настройки, указывается конкретный порядок действий, выполняемых в ходе реализации

[8] Модули среды исполнения интерпретируемых языков или языков, компилируемых в промежуточное представление, включают в себя интерпретатор или виртуальную машину выполнения промежуточного языка, библиотеки поддержки времени выполнения и стандартные библиотеки в составе виртуальной машины.

[9] Для оценки сложности кода должна использоваться одна из известных метрик, например, цикломатическая сложность программы (цикломатическое число Мак-Кейба), мера Холстеда, мера Овиедо и другие метрики.

соответствующих исследований, учитывающий особенности ОО, в том числе особенности, связанные с языками программирования, на которых написан ОО, составом реализуемых ОО ФБ, составом механизмов защиты от исследований, реализованных в ОО, типами внешних интерфейсов ОО, иные архитектурные особенности ОО, влияющие на порядок проведения исследований.

В случае, если исследования проводятся в связи с внесением изменений в сертифицированный ранее ОО, методика не разрабатывается, а результаты ПОД.1 определяют порядок проведения исследований и включаются в протокол исследований.

В методике проведения исследований (или протоколе исследований) приводится описание поверхности атаки в графической нотации в объеме сущностей не меньшем, чем:

интерфейсы непосредственной поверхности атаки;

модули, реализующие интерфейсы поверхности атаки;

подсистемы, включающие в себя модули, реализующие поверхность атаки (например, микросервисы, логически объединенные группы функций или компонентов, привлекаемые инфраструктурные компоненты).

Также указываются как минимум следующие характеристики модулей:

используемые языки программирования;

выполнение модулем функции веб-сервиса;

выполнение модулем функции среды выполнения для интерпретируемых языков программирования или языков программирования, компилируемых в промежуточное представление.

Пример графической нотации представлен в Приложении 3 к настоящей Методике.

Содержание ПОД.1:

Наименование исследования	Уровень контроля		
	6	5	4
ПОД.1	+	+	+
Дополнительные требования к ПОД.1		1	1, 2, 3, 4, 5

3.2. Подготовка исследовательского стенда (ПОД.2)

Задачами исследования являются:

а) подготовка исследовательского стенда, обеспечивающего проведение исследований в соответствии с представлением об ОО и разработанной методикой проведения исследований (ПОД.1);

б) подготовка специальных отладочных сборок ОО.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики;

сведения, полученные по результатам анализа документации и иных исходных данных (ПОД.1).

Требования к проведению исследования

Испытательной лабораторией формируется перечень сред выполнения исследований ОО, определенных в рамках КАО.2, ДАО.1 настоящей Методики и методикой функционального тестирования ОО, включающий не менее одной среды для каждого семейства сред функционирования.

Каждая операционная система, составляющая среду функционирования, формирует собственное семейство сред функционирования. Отнесение к семейству среды функционирования для сред, реализованных в формате средства контейнеризации, определяется парой «базовая технология средства контейнеризации — операционная система», в которой функционирует средство контейнеризации. Отнесение к семейству среды функционирования для сред, реализованных в формате средства виртуализации, определяется базовой технологией средства виртуализации.

При выборе среды проведения исследований из семейства следует экспертным методом определить среду, обладающую наименьшим защитным потенциалом.

Испытательной лабораторией выполняется контролируемая сборка дистрибутивов ОО с целью выявления:

взаимодействий сборочных сред и систем сборки с артефактами сборки, размещенными за пределами контролируемого разработчиком репозитория;

использования в сборочном процессе программного кода, в отношении которого анализ не был выполнен в требуемом виде и объеме;

включения в состав дистрибутива исполняемых файлов, не собираемых разработчиком из исходных кодов.

В случае выявления указанных фактов испытательная лаборатория расценивает их в качестве недостатков безопасности ОО и принимает решение о невозможности проведения дальнейших исследований ОО.

При подготовке исследовательского стенда проводятся проверки в отношении того, что:

в исследовательском стенде реализованы меры защиты, направленные на обеспечение целостности ОО и среды функционирования, а также на исключение возможности несанкционированного доступа к результатам проведения исследований;

все выбранные среды функционирования, а также инструментальные средства анализа и контроля, обеспечивающие проведение исследований ОО, развернуты и настроены в соответствии со своей эксплуатационной документацией. Испытательная лаборатория должна убедиться, что в ходе исследований не нарушаются условия эксплуатации сертифицированных сред функционирования ОО, в том числе операционных систем;

в подготовленных средах функционирования установлен и настроен дистрибутив ОО в соответствии с руководством по безопасной установке и настройке средства, приведенном в эксплуатационной документации.

В ходе установки ОО в среду функционирования и его настройки проводится контроль отсутствия недекларированных взаимодействий ОО с сетевыми ресурсами и ресурсами среды функционирования.

По результатам установки ОО проверяется, что был выполнен антивирусный контроль ОО. Антивирусный контроль ОО предусматривает анализ дистрибутива ОО, всех файлов установленного ОО, а также среды функционирования с использованием не менее двух сертифицированных средств антивирусной защиты от различных разработчиков с актуальными на момент проведения контроля базами данных признаков вредоносных компьютерных программ (вирусов). В случае отсутствия для среды функционирования ОО сертифицированных средств антивирусной защиты, могут применяться иные средства антивирусной защиты.

Затем проводится сравнение контрольных сумм дистрибутива ОО и исполняемых файлов установленного ОО с контрольными суммами, зафиксированными в документации ОО. Контрольные суммы должны быть рассчитаны в соответствии с национальным стандартом Российской Федерации ГОСТ 34.11-2012 «Информационная технология. Криптографическая защита информации. Функции хэширования» (алгоритм «Стрибог»). Контрольное суммирование выполняется средствами сертифицированной операционной системы либо сертифицированным средством контрольного суммирования.

В случае реализации ОО с использованием технологий контейнеризации и (или) виртуализации, контрольные суммы исполняемых файлов ОО рассчитываются относительно снимков состояния (снапшотов) файловых систем выполняющихся контейнеров и (или) виртуальных машин.

Дополнительные требования к исследованиям (усиления)

1. В ходе исследований проверяется, что разработчиком ОО подготовлены специальные сборки ОО со встраиванием инструментальных датчиков срабатывания ошибок при компиляции модулей, подвергаемых динамическому анализу, для каждой среды функционирования, включенной в исследования в соответствии с пунктом 3.2 настоящей Методики.

Испытательной лабораторией совместно с разработчиком ОО определяется перечень требуемых датчиков ошибок (ошибки работы с памятью, неопределенное поведение, нарушения потока управления и т.п.), исходя из технических возможностей, обусловленных используемыми в ОО языками программирования и средой функционирования.

Если система (системы) сборки для выбранной среды (сред) функционирования не позволяет штатным образом встраивать такие датчики, допускается не подготавливать специальную отладочную сборку ОО, а использовать в динамическом анализе любые другие средства встраивания датчиков срабатывания ошибок, такие как динамическое двоичное инструментирование и динамическая компоновка либо запуск интерпретатора с дополнительными опциями.

Если для языка исходных кодов ОО и среды его функционирования ряд типов датчиков срабатывания ошибок не применимы или отсутствуют, их использование не требуется.

2. Исследовательский стенд должен обеспечивать проведение динамического анализа ОО для каждой процессорной архитектуры среды функционирования ОО.

Требования к результатам исследования

По результатам подготовки исследовательского стенда в документации и акте отбора образцов ОО после завершения исследований ОО фиксируются значения контрольных сумм дистрибутива, исполняемых файлов и файлов исходных текстов (в соответствии с уровнем контроля) ОО.

Содержание ПОД.2:

Наименование исследования	Уровень контроля		
	6	5	4
ПОД.2	+	+	+
Дополнительные требования к ПОД.2		1	1, 2

4. Проведение исследований по выявлению уязвимостей и недекларированных возможностей

В ходе проведения исследований по выявлению уязвимостей и недекларированных возможностей объекта оценки должны выполняться:

а) анализ архитектуры ОО;

б) статический анализ ОО;

в) динамический анализ ОО;

г) экспертиза кода ОО.

4.1. Анализ архитектуры объекта оценки (КАО)

Анализ архитектуры ОО включает:

а) анализ архитектуры ОО статическими методами (КАО.1);

б) анализ архитектуры ОО динамическими методами (КАО.2).

4.1.1. Общие требования к проведению всех видов анализа архитектуры ОО

При выполнении всех видов анализа архитектуры ОО анализируются полномочия (права, привилегии, разрешения, политики и другие)[10] ОО и его составных частей (файлов, модулей, процессов и других, в том числе в отношении интерпретаторов, веб-серверов и серверов приложений), назначенные в соответствии с руководством по безопасной установке и настройке ОО и его составных частей, а также руководствами администратора и пользователя из состава эксплуатационной документации ОО**.**

В ходе анализа архитектуры контролируется назначение ОО и его составным частям полномочий, минимально необходимых для функционирования ОО. Выявленные избыточные полномочия ОО и его составных частей рассматриваются в качестве недостатков безопасности ОО.

Перечень исследований при анализе архитектуры ОО представлен в таблице 2.

[10] Примерами повышенных полномочий являются: использование флагов доступа SUID/SGID; функционирование от имени суперпользователя или администратора, имеющего наивысший уровень полномочий в операционной системе, функционирование в пространстве ядра операционной системы, функционирование на повышенном уровне мандатной конфиденциальности или целостности, или с нетривиальным набором неиерархических мандатных категорий, иные виды полномочий, определяемые средой функционирования или непосредственно объектом оценки.

Таблица 2

Шифр	Наименование исследования	Уровень контроля		
		6	5	4
КАО.1	Анализ архитектуры ОО статическими методами	+	1, 2, 3	1, 2, 3
КАО.2	Анализ архитектуры ОО динамическими методами	+	+	1
Примечание: «+» – исследования подлежат выполнению для соответствующего уровня контроля; «цифра» – дополнительные требования к исследованиям для соответствующего уровня контроля				

4.1.2. Анализ архитектуры ОО статическими методами (КАО.1)

Задачей исследования является выявление недостатков безопасности и известных уязвимостей ОО.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики;

сведения, полученные по результатам выполнения ПОД.1;

испытательный стенд и тестовые сборки ОО, полученные по результатам выполнения ПОД.2.

Требования к проведению исследования

Испытательной лабораторией самостоятельно собираются и анализируются материалы из общедоступных источников (публикации разработчика, пользователей и исследователей об ОО и его заимствованных компонентах, базы данных уязвимостей, сведения о типовых уязвимостях и ошибках, тематические информационные ресурсы) о недостатках (слабостях) и уязвимостях как самого ОО, так и аналогичного по архитектуре программного обеспечения в дополнение к исходным данным, предоставленным разработчиком ОО. В качестве источников информации о типовых недостатках (слабостях) и уязвимостях следует использовать в том числе:

а) банк данных угроз безопасности информации ФСТЭК России, а также иные отечественные базы уязвимостей;

б) зарубежные базы данных уязвимостей и недостатков (слабостей)

(Common Vulnerabilities and Exposures, Common Weakness Enumeration и иные), а также общий перечень шаблонов атак (Common Attack Pattern Enumeration and Classification);

в) публикации и форумы, содержащие отзывы пользователей;

г) результаты исследований, полученные сторонними исследователями, как в отношении ОО, так и в отношении программных продуктов, аналогичных по архитектуре исследуемому ОО.

В ходе исследований оцениваются исходные данные согласно пункту 3.1 и пункту 2.3 настоящей Методики на соответствие требованиям проводимого исследования в целях использования в исследованиях ОО результатов, полученных разработчиком:

а) перечни программных компонентов и иные результаты композиционного анализа в части поиска известных уязвимостей заимствованных компонентов в составе ОО (в том числе в составе образов контейнеров) в публично доступных базах уязвимостей;

б) результаты автоматизированного анализа конфигурации ОО и составляющих ОО отдельных модулей, направленного на выявление уязвимостей конфигурации ОО, применение для контроля настроек ОО инструментальных средства анализа конфигураций при наличии таковых средств;

в) учет в архитектуре ОО требований безопасности, результатов композиционного анализа и анализа конфигураций.

Если предоставленные разработчиком ОО исходные данные отвечают требованиям, то испытательная лаборатория должна:

провести выборочную проверку результатов применяемых разработчиком ОО анализаторов, размер контрольной выборки должен быть не менее чем 10 предупреждений для каждого используемого анализатора;

для сторонних компонентов ОО, попавших в поверхность атаки, подтвердить отсутствие актуальных уязвимостей либо реализацию в их отношении компенсирующих мер.

Если предоставленные разработчиком результаты анализа архитектуры ОО статическими методами не отвечают требованиям, испытательная лаборатория проводит анализ архитектуры статическими методами самостоятельно, в соответствии с требованиями настоящей Методики либо принимает решение о невозможности проведения дальнейших исследований ОО.

В целях выявления НДВ, компрометирующих конструкций, недостатков архитектуры и кода должен проводится анализ комментариев в исходном коде ОО.

Испытательной лабораторией самостоятельно анализируются:

исходный код ОО в целях выявления открыто присутствующей чувствительной информации и «секретов» (пароли, приватные ключи и другие

данные);

статическими методами выявленные веб-интерфейсы, составляющие поверхность атаки, на предмет наличия уязвимостей, перечисленных в разделе «Типовые уязвимости веб-приложений» банка данных угроз безопасности информации ФСТЭК России.

Проводится оценка соответствия конфигураций запуска отдельных контейнеров и групп (сетей) контейнеров, составляющих ОО в контейнерном исполнении, публично доступным рекомендациям по безопасности и минимизации полномочий (например, рекомендации ФСТЭК России, CIS, Kubernetes Security Best Practices и другие рекомендации), в том числе:

а) для ОО в контейнерном исполнении обеспечивается запуск контейнеров с полномочиями, минимально необходимыми для функционирования ОО, на уровне процессов хостовой операционной системы;

б) для ОО в контейнерном исполнении, находящегося под управлением оркестратора контейнеров, обеспечивается в соответствии с заданными разработчиком средства правилами управление доступом между компонентами ОО (контейнерами, микросервисами, иными ресурсами, характерными для выбранного типа оркестратора), компонентами среды функционирования, внешними по отношению к ОО компонентами в соответствии с пунктом 2.3 настоящей Методики.

Избыточные полномочия компонентов ОО и избыточные разрешающие правила доступа к компонентам ОО рассматриваются в качестве недостатков безопасности ОО.

Дополнительные требования к исследованиям (усиления)

1. Испытательной лабораторией анализируется перечень программных компонентов и иные результаты композиционного анализа в части поиска известных уязвимостей привлекаемых компонентов среды функционирования ОО за исключением компонентов системы сборки и сборочной среды ОО.

2. В ходе исследований контролируются наличие в составе ОО кода (либо генераторов такового кода), выполняющегося в контексте браузера (например, JavaScript, ActiveX, WebAssembly и других), а также учет кода в перечне программных компонентов и выполнение разработчиком в отношении него композиционного анализа.

Выявление динамической загрузки кода в контекст браузера не из состава (либо результатов генерации) ОО требуется расценивать в качестве основания для прекращения исследований ОО.

Выявление возможностей взаимодействия кода со стандартными интерфейсами доступа к камере, микрофону, экрану, уведомлениям, геопозиции, буферу обмена, методам оплаты и прочим требуется расценивать

в качестве недостатка безопасности ОО.

3. Испытательной лабораторией анализируются конфигурации систем сборки и сборочных сред ОО и его компонентов на предмет использования применимых опций конфигурирования, компиляции и компоновки ОО, повышающих безопасность исполняемого кода ОО[11] в том числе за счет минимизации объема исполняемого кода ОО.

Испытательная лаборатория учитывает отсутствие выбора и использования указанных опций в качестве недостатка безопасности ОО.

Требования к результатам исследования

Если предоставленные разработчиком результаты анализа архитектуры ОО статическими методами, проводимого при разработке безопасного программного обеспечения, соответствуют требованиям пункта 3.1 и пункта 2.3 настоящей Методики, испытательной лаборатории допускается повторно использовать результаты анализа при оформлении материалов исследований.

Результаты исследований фиксируются в материалах исследований в объеме, соответствующем требованиям пункта 5.2 настоящей Методики.

Содержание КАО.1:

Наименование исследования	Уровень контроля		
	6	5	4
КАО.1	+	+	+
Дополнительные требования к КАО.1		1, 2, 3	1, 2, 3

4.1.3. Анализ архитектуры ОО динамическими методами (КАО.2)

Задачей исследования является выявление недостатков безопасности и известных уязвимостей ОО, самостоятельно и в результате анализа результатов выполнения процессов разработки ОО в соответствии с ГОСТ Р 56939-2024, методами и инструментами, требующими выполнения ОО.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики;

сведения, полученные по результатам выполнения ПОД.1;

[11] При оценке использования применимых опций испытательной лаборатории следует учитывать положения ГОСТ Р 71206-2024 «Разработка безопасного программного обеспечения. Безопасный компилятор языков С/С++», а также методические рекомендации ФСТЭК России по повышению безопасности программных компонентов.

испытательный стенд и тестовые сборки ОО, полученные по результатам выполнения ПОД.2;

сведения об архитектуре ОО, полученные по результатам выполнения КАО.1.

Требования к проведению исследования

Испытательной лабораторией выполняется не менее трех различных сценариев функционирования ОО с контролем сетевой активности ОО с целью выявления утечек чувствительных данных (например, учетные записи и пароли, приватные ключи и другие данные) и НДВ. Примерами сценариев являются: создание учетной записи пользователя и ее редактирование, изменение набора правил фильтрации сетевого трафика, выполнение антивирусного анализа файла и другие сценарии. При выборе сценариев следует отдавать предпочтение сценариям, затрагивающим порождение (модификацию) чувствительных данных, выполнение административных функций, а также предполагающих взаимодействие с подсистемами, расположенными на различных узлах сети.

При проведении исследований выполняется анализ сетевых интерфейсов ОО, составляющих поверхность атаки, с использованием анализаторов безопасности, предполагающих запуск сценариев, реализующих логику автоматизированного определения недостатков безопасности ОО.

Испытательной лабораторией выполняется анализ веб-интерфейсов из состава поверхности атаки динамическими методами на предмет наличия уязвимостей, перечисленных в разделе «Типовые уязвимости веб-приложений» банка данных угроз безопасности информации ФСТЭК России.

Дополнительные требования к исследованиям (усиления)

1. В ходе исследований анализируются рекомендованные ФСТЭК России либо применяемые разработчиком ОО методы автоматизированного динамического анализа, направленные на подтверждение устойчивости ОО к специфическим для данного типа ОО угрозам[12].

Испытательной лабораторией выполняется автоматизированный динамический анализ ОО, направленный на подтверждение устойчивости СЗИ к специфическим угрозам, выбранным подмножеством методов. Испытательная лаборатория предоставляет инструменты анализа и может привлекать разработчика ОО для организации и проведения анализа.

Требования к результатам исследования

Результаты исследований фиксируются в материалах исследований

[12] Например, устойчивость ОО к компрометации приватных параметров, составляющих ОО и различных сервисов, доступных учетной записи, от имени которой нарушитель проник в систему, устойчивость систем массового обслуживания к DDoS-атакам, устойчивость базовой системы ввода-вывода к взаимодействию через SMI-прерывания и другие угрозы.

в объеме, соответствующем требованиям пункта 5.2 настоящей Методики.

Содержание КАО.2:

Наименование исследования	Уровень контроля		
	6	5	4
КАО.2	+	+	+
Дополнительные требования к КАО.2			1

4.2. Статический анализ объекта оценки (САО)

Перечень исследований при проведении статического анализа ОО представлен в таблице 3.

Таблица 3

Шифр	Наименование исследования	Уровень контроля		
		6	5	4
САО	Статический анализ исходных кодов ОО		+	1, 2, 3
Примечание: «+» – исследования подлежат выполнению для соответствующего уровня контроля; «цифра» – дополнительные требования к исследованиям для соответствующего уровня контроля				

Задачей исследования является выявление недостатков безопасности ОО самостоятельно и в результате анализа результатов выполнения процессов разработки ОО в соответствии с ГОСТ Р 56939-2024 методами и инструментами статического анализа исходного кода.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики, в первую очередь подпункта «б» пункта 2.3 настоящей Методики;

сведения, полученные по результатам выполнения ПОД.1;

испытательный стенд, полученный по результатам выполнения ПОД.2;

план по разметке предупреждений статического анализатора по результатам предыдущих исследований ОО (если применимо).

Требования к проведению исследования

При проведении статического анализа оцениваются исходные данные согласно пункту 3.1 настоящей Методики, а также проводится оценка реализации следующих требований:

а) статический анализ выполняется разработчиком ОО в отношении

исходного кода всех модулей, составляющих поверхность атаки ОО;

б) статический анализ выполняется разработчиком для всех высокоуровневых языков программирования, которые встречаются в исходном коде модулей[13];

в) используется статический анализатор, реализующий метод автоматизированного анализа исходного кода на уровне синтаксического дерева;

г) используются конфигурации статического анализатора, учитывающие специфику используемых в ОО языков программирования и заимствуемых модулей ОО;

д) выполняется статический анализ исходного кода, который штатным образом строится (генерируется) непосредственно в процессе функционирования ОО, транслируется (компилируется) и выполняется как часть ОО. Выбранные для проведения статического анализа конфигурации генератора исходного кода и средств трансляции (компиляции) обосновываются разработчиком ОО и должны соответствовать типовому сценарию функционирования ОО[14];

е) разработчиком ОО выполнена ручная разметка всех выданных анализатором предупреждений о критических ошибках в соответствии с классификацией национального стандарта Российской Федерации ГОСТ Р 71207-2024 «Защита информации. Разработка безопасного программного обеспечения. Статический анализ программного обеспечения. Общие требования»;

ж) разработчиком ОО выполнена ручная разметка всех выданных анализатором предупреждений, предусмотренных планом поддержки безопасности заимствованных компонентов (в случае внесения изменения в сертифицированный ОО);

з) разработчиком ОО выполнена ручная разметка всех выданных анализатором предупреждений, предусмотренных программой исследований безопасности модулей с открытым исходным кодом Центра исследований, если разработчик ОО выполняет исследование модулей ОО в рамках деятельности Центра исследований.

Испытательной лабораторией оценивается объем кода из состава подлежащих статическому анализу модулей ОО, в отношении которого не выполнены требования пункта 4.2 настоящей Методики.

---

[13] Не требуется проведение статического анализа исходного кода, написанного на языках разметки (например, html, css3, xaml и т.п.), не требуется проведение статического анализа исходного кода, написанного на языке ассемблера.

[14] Для трансляции (компиляции) используются средства как среды функционирования ОО так и средства, включенные в состав ОО.

Если разработчик ОО выполняет исследование модулей ОО в рамках деятельности Центра исследований и в ходе исследования ОО выполнил статический анализ подлежащих статическому анализу модулей ОО в объеме программы исследований безопасности, то требования пункта 4.2 настоящей Методики в отношении данных модулей считаются выполненными.

В противном случае испытательная лаборатория может выполнить статический анализ кода самостоятельно в соответствии с требованиями настоящей Методики либо принять решение о невозможности проведения дальнейших исследований ОО. Если суммарный объем такого кода не превосходит 10 000 строк, допускается вместо статического анализа выполнить экспертизу участков исходного кода ОО (ЭКО).

Если предоставленные разработчиком ОО исходные данные отвечают требованиям настоящего пункта Методики, испытательная лаборатория проводит выборочную проверку разметки результатов статического анализа из состава исходных данных. Выборочная проверка выполняется в отношении контрольной выборки предупреждений, составленной по следующему принципу:

в состав выборки включается не менее чем 20 предупреждений о критических ошибках для каждого из проанализированных языков программирования высокого уровня;

в состав выборки включаются предупреждения, относящиеся не менее чем к 5 подлежащим анализу модулям;

в состав выборки включается не менее 10 предупреждений от каждого из использованных статических анализаторов;

для каждого из использованных статических анализаторов в состав выборки включаются предупреждения различных типов.

При составлении контрольной выборки предупреждений следует руководствоваться следующими приоритетами:

«истинные, не требующие исправления», в первую очередь;

«ложноположительные», во вторую очередь;

в порядке убывания определенного анализатором уровня критичности предупреждений.

Если доля неверно размеченных разработчиком предупреждений в составе контрольной выборки не превосходит 10 %, исследования ОО продолжаются после устранения разработчиком выявленных недостатков безопасности ОО.

Если доля неверно размеченных разработчиком ОО предупреждений превосходит 10 %, испытательной лабораторией выполняется статический анализ кода самостоятельно в соответствии с требованиями настоящей Методики либо принимается решение о невозможности проведения дальнейших

исследований ОО.

Испытательной лабораторией самостоятельно выполняется статический анализ исходного кода не менее чем 5 различных модулей, составляющих поверхность атаки ОО, в объеме не менее чем по 10 предупреждений для каждого высокоуровневого языка программирования в составе исходных кодов модулей. Испытательная лаборатория может использовать любой инструмент статического анализа, соответствующий требованиям Методики[15].

Дополнительные требования к исследованиям (усиления)

1. В ходе исследований испытательной лабораторией проверяется, что состав анализируемых модулей дополнен модулями, реализующими ФБ ОО.

2. Испытательной лабораторией проверяется, что используемый статический анализатор отвечает требованиям национального стандарта Российской Федерации ГОСТ Р 71207-2024 «Защита информации. Разработка безопасного программного обеспечения. Статический анализ программного обеспечения. Общие требования».

3. В ходе исследований испытательной лабораторией проверяется, что для подлежащих статическому анализу модулей с открытым исходным кодом, для которых определены методики статического анализа, указанные в подпункте «л» пункта 2.3 настоящей Методики, статический анализ выполнен в соответствии с требованиями этих методик.

Требования к результатам исследования

Если предоставленные разработчиком ОО результаты статического анализа исходного кода ОО, проводимого при разработке безопасного ПО, соответствуют требованиям пунктов 3.1 и 2.3 настоящей Методики, испытательной лаборатории допускается повторно использовать результаты анализа при оформлении материалов исследований.

Результаты исследований фиксируются в материалах исследований в объеме, соответствующем требованиям пункта 5.2 настоящей Методики. Дополнительно требуется зафиксировать в материалах исследований в формате электронных приложений:

конфигурации инструментов статического анализа, в том числе состав выборки проверяемых предупреждений;

машиночитаемые результаты статического анализа (база данных и иные формы представления) и результаты разметки;

исправления (патчи) истинных предупреждений либо иные свидетельства применения исправлений (патчей);

---

[15] Инструмент статического анализа предоставляется испытательной лабораторией, встраивание инструмента в исследовательский стенд осуществляется на этапе подготовки исследовательского стенда (ПОД.2).

список модулей ОО, содержащих участки исходного кода, написанные с использованием подлежащих статическому анализу языков программирования, не поддерживаемых используемыми статическими анализаторами, либо содержащих ассемблерные вставки или целиком написанных на языке ассемблера, а также наименование неподдерживаемого языка программирования (языка ассемблера), и объем кода в строках на неподдерживаемом языке (языке ассемблера).

Содержание САО:

Наименование исследования	Уровень контроля		
	6	5	4
САО		+	+
Дополнительные требования к САО			1, 2, 3

4.3. Динамический анализ объекта оценки (ДАО)

Динамический анализ ОО включает:

а) тестирование модулей ОО (ДАО.1);

б) фаззинг-тестирование ОО (ДАО.2).

4.3.1 Общие требования к проведению всех видов динамического анализа ОО

Для 4-5 уровней контроля тестирование модулей ОО (ДАО.1) и фаззинг-тестирование ОО (ДАО.2) должны выполняться в отношении специальных сборок ОО со встроенными датчиками срабатывания ошибок. Специальные отладочные сборки ОО подготавливаются на этапе ПОД.2 в рамках выполнения требований усиления 1 (с учетом уровня контроля).

Перечень используемых датчиков фиксируется в протоколе исследований.

Испытательной лабораторией фиксируются факты выявления механизмов, встроенных в исходный или исполняемый код ОО и препятствующих проведению динамического анализа. В случае выявления в ходе исследований таковых механизмов, эти механизмы и защищенные ими программные модули (участки кода) должны быть проанализированы при проведении экспертизы кода (ЭКО). Если суммарный объем исходных кодов данных механизмов и защищенных модулей (участков кода) превосходит 10 000 строк, делается вывод о невозможности проведения дальнейших исследований.

Перечень исследований при динамическом анализе ОО представлен в таблице 4.

Таблица 4

Шифр	Наименование исследований	Уровень контроля		
		6	5	4
ДАО.1	Тестирование модулей ОО	+	1	1, 2, 3, 4, 5
ДАО.2	Фаззинг-тестирование ОО	+	1	1, 2, 3, 4, 5
Примечание: «+» – исследования подлежат выполнению для соответствующего уровня контроля; «цифра» – дополнительные требования к исследованиям для соответствующего уровня контроля				

4.3.2. Тестирование модулей ОО (ДАО.1)

Задачей исследования является выявление недостатков безопасности ОО, в том числе ошибок реализации функций безопасности ОО, самостоятельно и в результате анализа результатов выполнения процессов разработки ОО в соответствии с ГОСТ Р 56939-2024, методом и инструментами автоматического динамического тестирования модулей ОО со сбором покрытия.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики, в первую очередь подпункта «ж» пункта 2.3 настоящей Методики;

сведения, полученные по результатам выполнения ПОД.1;

испытательный стенд, полученный по результатам выполнения ПОД.2.

Требования к проведению исследования

При тестировании модулей оцениваются исходные данные согласно пункту 3.1 настоящей Методики, а также проводится оценка реализации разработчиком ОО следующих требований:

а) выполняется автоматическое тестирование[16] всех модулей, реализующих ФБ ОО;

б) выполняется тестирование модулей для каждой процессорной архитектуры, поддержка которой обеспечивается исследовательским стендом;

в) выполняется сбор тестового покрытия по функциям исходного или исполняемого кода для каждого тестируемого модуля ОО;

г) выполняется анализ журналов тестирования для каждого тестируемого модуля на предмет наличия сбоев и нарушений требований, проверяемых тестами, в том числе порождаемых датчиками срабатывания ошибок.

[16] Тесты могут быть реализованы на различных уровнях – системном, интеграционном, модульном.

При проведении исследований оценивается число модулей из состава подлежащих тестированию модулей ОО, в отношении которого не выполнены требования пункта 4.3.2 настоящей Методики. В исследовании ДАО.1 не учитываются тесты, в ходе выполнения которых не вызываются функции подлежащих тестированию модулей ОО.

Если разработчиком ОО выполняется исследование модулей ОО в рамках деятельности Центра исследований безопасности системного ПО и в ходе исследования ОО выполнено тестирование подлежащих тестированию модулей ОО в объеме программы исследований безопасности, то требования пункта 4.3.2 настоящей Методики в отношении данных модулей считаются выполненными.

В противном случае:

если число таких модулей не превосходит 5, испытательной лабораторией проводится тестирование модулей самостоятельно в соответствии с требованиями настоящей Методики либо принимается решение о невозможности проведения дальнейших исследований ОО;

если число таких модулей превосходит 5, испытательной лабораторией принимается решение о невозможности проведения дальнейших исследований ОО.

Если предоставленные разработчиком ОО исходные данные отвечают требованиям пункта 4.3.2 настоящей Методики, испытательной лабораторией проводится выборочная проверка результатов анализа тестов, выявивших сбои и нарушение требований. Выборочная проверка выполняется в отношении контрольной выборки тестов, составленной по следующему принципу:

в состав выборки включается не менее 50 тестов, выявивших сбои и нарушение требований;

в состав выборки включаются тесты, относящиеся не менее чем к 5 различным модулям;

в состав выборки включаются все тесты, выявившие сбои и нарушение требований, явно связанные с безопасностью регрессий[17], не исправленных в текущей версии модуля ОО.

Если доля неверно размеченных разработчиком результатов тестов в составе контрольной выборки не превосходит 10 %, исследования ОО продолжаются после устранения разработчиком выявленных недостатков безопасности ОО.

Если доля неверно размеченных разработчиком ОО результатов тестов

[17] Под регрессиями понимаются тесты, подтверждающих наличие известных недостатков безопасности, выявленных в предыдущих версиях модуля.

в составе контрольной выборки превосходит 10 %, испытательная лаборатория принимает решение о невозможности проведения дальнейших исследований ОО.

Дополнительные требования к исследованиям (усиления)

1. В ходе исследований испытательной лабораторией проверяется, что состав тестируемых модулей дополнен модулями, составляющими непосредственную поверхность атаки ОО.

2. Испытательной лабораторией проверяется, что для тестируемых модулей с открытым исходным кодом, для которых определены методики тестирования, указанные в подпункте «м» пункта 2.3 настоящей Методики, тестирование выполнено в соответствии с требованиями этих методик.

3. В ходе исследований испытательной лабораторией проверяется, что сбор тестового покрытия осуществляется по строкам исходного кода или по базовым блокам исполняемого кода для каждого тестируемого модуля ОО.

4. Испытательной лабораторией проверяется, что совокупный достигнутый процент тестового покрытия для каждого тестируемого модуля составляет не менее 25 %.

5. В ходе исследований испытательной лабораторией проверяется, что состав тестируемых модулей дополнен модулями, обеспечивающими реализацию ФБ ОО.

Требования к результатам исследования

Если предоставленные разработчиком результаты тестирования модулей ОО, проводимого при разработке безопасного ПО, соответствуют требованиям пунктов 3.1 и 2.3 настоящей Методики, испытательной лаборатории разрешается повторно использовать результаты анализа при оформлении материалов исследований.

Результаты исследований фиксируются в материалах исследований в объеме, соответствующем требованиям пункта 5.2 настоящей Методики. Дополнительно требуется зафиксировать в материалах исследований в формате электронных приложений структурное покрытие для каждого тестируемого модуля.

Содержание ДАО.1:

Наименование исследования	Уровень контроля		
	6	5	4
ДАО.1	+	+	+
Дополнительные требования к ДАО.1		1	1, 2, 3, 4, 5

4.3.3. Фаззинг-тестирование ОО (ДАО.2)

Задачей исследования является выявление недостатков безопасности ОО самостоятельно и в результате анализа результатов выполнения процессов разработки ОО в соответствии с ГОСТ Р 56939-2024 методом и инструментами фаззинг-тестирования и сбора покрытия ОО.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики;

сведения, полученные по результатам выполнения ПОД.1;

испытательный стенд, полученный по результатам выполнения ПОД.2;

сведения о покрытии модулей, достигаемом по результатам выполнения ДАО.1.

Требования к проведению исследования

При проведении фаззинг-тестирования оцениваются исходные данные согласно пункту 3.1 настоящей Методики, а также проводится оценка реализации разработчиком ОО следующих требований:

а) выполняется фаззинг-тестирование интерфейсов, относящихся к поверхности атаки ОО. Не требуется выполнять фаззинг-тестирование в отношении кода, выполняющегося в контексте браузера. Допускается проводить фаззинг-тестирование только в одной в среде функционирования, обеспечивающей наиболее эффективное выполнение тестирования;

б) выполняется фаззинг-тестирование интерфейсов, для которых доступны специализированные фаззеры, адаптированные для тестирования конкретных типов интерфейсов (например, SMI, USB, SQL и другие интерфейсы), в том числе с использованием данных фаззеров. Фаззинг-тестирование интерфейсов веб-приложений выполняется специализированными фаззерами, адаптированными для решения такого класса задач.

Если применяются фаззеры, поддерживающие работу с коллекциями входных данных, для каждого из подлежащих фаззинг-тестированию интерфейсов должны быть сформированы и применены коллекции примеров входных данных. Коллекции должны вызывать использование различных функциональных возможностей тестируемого интерфейса. Формирование коллекций следует выполнять на основе публично доступных коллекций, использования генераторов коллекций, использования инструментов автоматического определения и формирования коллекций данных или экспертным способом.

Если применяются фаззеры, не поддерживающие работу с коллекциями входных данных, то для каждого из подлежащих фаззинг-тестированию интерфейсов должны быть сформированы и применены правила

порождения входных данных. Правила должны вызывать использование различных функциональных возможностей тестируемого интерфейса.

Если применяются фаззеры, поддерживающие работу со словарями, то для интерфейсов, где применим такой подход, должны быть сформированы и применены словари лексем;

в) выполняется для каждой тестируемой цели (интерфейс, участок кода) анализ журналов фаззера на предмет фактов зависаний и сбоев работы ОО, в том числе порождаемых датчиками срабатывания ошибок, и на предмет подтверждения воспроизведения данных фактов в штатном режиме функционирования ОО.

Испытательной лабораторией оценивается число целей из состава подлежащих фаззинг-тестированию, в отношении которых не выполнены требования пункта 4.3.3 настоящей Методики.

Если разработчик ОО выполняет исследование модулей ОО в рамках деятельности Центра исследований безопасности системного ПО и в ходе исследования ОО выполнил фаззинг-тестирование подлежащих фаззинг-тестированию модулей ОО в объеме программы исследований безопасности, требования пункта 4.3.3 настоящей Методики в отношении данных модулей считаются выполненными.

Если число таких целей не превосходит 3, испытательной лабораторией выполняется фаззинг-тестирование самостоятельно в соответствии с требованиями настоящей Методики либо принимается решение о невозможности проведения дальнейших исследований ОО.

Если число таких целей превосходит 3, испытательной лабораторией принимается решение о невозможности проведения дальнейших исследований ОО.

Если предоставленные разработчиком ОО исходные данные отвечают требованиям текущего пункта настоящей Методики, испытательной лабораторией проводится проверка результатов фаззинг-тестирования, выявивших воспроизводимые сбои и зависания целей.

Если доля неверно размеченных разработчиком результатов фаззинг-тестирования в составе контрольной выборки не превосходит 10 %, исследования ОО продолжаются после устранения разработчиком выявленных недостатков безопасности ОО.

Если доля неверно размеченных разработчиком ОО результатов фаззинг-тестирования в составе контрольной выборки превосходит 10 %, испытательной лабораторией принимается решение о невозможности проведения дальнейших исследований ОО.

Испытательной лабораторией самостоятельно разрабатывается не менее

2 новых фаззинг-цели и выполняется их фаззинг-тестирование в составе ОО либо значительно дорабатывается не менее 3 существующих комплектов фаззинг-тестов для целей в составе ОО (например, улучшение коллекции, расширение словаря, добавление нового датчика срабатывания ошибок, учет сложных зависимостей мутируемых параметров). При выборе подлежащего тестированию кода следует в приоритетном порядке выполнять фаззинг-тестирование целей, наиболее значимых для безопасности ОО, с приведением обоснования данного выбора.

Дополнительные требования к исследованиям (усиления)

1. В ходе проведения исследований испытательной лабораторией проверяется, что используемый разработчиком ОО фаззер обеспечивает применение генетических алгоритмов фаззинг-тестирования за счет инструментирования кода в случае, если доступны инструменты, позволяющие осуществлять такое фаззинг-тестирование с учетом языков программирования, компиляторов и среды функционирования ОО. При этом допускается выполнять инструментирование:

как по исходным текстам, так и по бинарному (в том числе промежуточному) представлению;

по бинарному представлению, если модуль, написанный на интерпретируемом языке программирования, компилировался в промежуточное представление или подвергался принудительной компиляции в исполняемый код («Ahead of Time» компиляция).

2. Испытательной лабораторией проверяется выполнение в дополнение к условиям, указанным в других дополнительных требованиях, при фаззинг-тестировании ОО как минимум одного из следующих условий:

а) для фаззеров, подсчитывающих число найденных уникальных путей, число уникальных путей, найденных фаззером, должно не менее чем в два раза превышать число образцов входных данных, поданных на вход фаззеру;

б) для фаззеров, подсчитывающих достигнутое покрытие, процент покрытия, достигнутого фаззером, должен не менее чем в два раза превышать процент покрытия, достигаемый в результате подачи на вход фаззеру стартовых образцов входных данных.

При достижении покрытия по строкам исходного кода не менее 90 % (по базовым блокам не менее 80 %) испытательная лаборатория может не учитывать дальнейший прирост покрытия для принятия решения о завершении фаззинг-тестирования.

3. В ходе проведения исследований испытательной лабораторией проверяется, что для тестируемых модулей с открытым исходным кодом, для которых определены методики фаззинг-тестирования, указанные

в подпункте «н» пункта 2.3 настоящей Методики, фаззинг-тестирование выполнено в соответствии с требованиями этих методик.

4. Испытательной лабораторией проверяется, что фаззинг-тестирование выполнялось в отношении специально подготовленных синтетических целей, сформированных с целью эффективного тестирования участков кода, выделенных в ходе выполнения требований усиления 4 ПОД.1.

При фаззинг-тестировании интерфейсов ОО допускается ограничиваться фаззинг-тестированием в отношении специально подготовленных синтетических целей. Достаточность состава выделенных целей определяется испытательной лабораторией непосредственно в ходе проведения исследований ДАО.2 и фиксируется с обоснованием в протоколах исследований.

Функции-обертки обеспечивают вызов тестируемых участков кода целей, конвертируя мутируемые фаззером данные во входные параметры тестируемых функций в соответствии со спецификациями интерфейсов (протоколов), типов данных и контекста выполнения тестируемых функций в составе ОО.

5. Испытательной лабораторией проверяется, что тестирование каждой фаззинг-цели продолжается до состояния, когда в течение не менее двух часов фаззер не находит нового пути или отсутствует прирост покрытия при условии, что на вход подаются уникальные мутированные данные.

Требования к результатам исследования

Если предоставленные разработчиком результаты тестирования модулей ОО, проводимого при разработке безопасного ПО, соответствуют требованиям пунктов 3.1 и 2.3 настоящей Методики, испытательной лаборатории разрешается повторно использовать результаты анализа при оформлении материалов исследований.

Результаты исследований фиксируются в материалах исследований в объеме, соответствующем требованиям пункта 5.2 настоящей Методики. Дополнительно требуется зафиксировать в материалах исследований в формате электронных приложений структурное покрытие для каждого тестируемого модуля.

Дополнительно требуется зафиксировать в материалах исследований:

параметры сборки каждой фаззинг-цели;

принципы формирования коллекции, правил, словарей;

достигнутое структурное покрытие для каждой фаззинг-цели.

Испытательной лабораторией в протоколах исследований фиксируются результаты, полученные по пункту 4.3.3 настоящей Методики, которые в дальнейшем передаются разработчику ОО с целью развития применяемого разработчиком комплекта тестов.

Содержание ДАО.2:

Наименование исследования	Уровень контроля		
	6	5	4
ДАО.2	+	+	+
Дополнительные требования к ДАО.2		1, 2	1, 2, 3, 4, 5

4.4. Экспертиза кода объекта оценки (ЭКО)

Экспертиза кода ОО представляет собой ручной анализ участков исходного и восстановленного кода ОО с целью выявления уязвимостей и НДВ (ЭКО).

Перечень исследований при экспертизе кода ОО представлен в таблице 5.

Таблица 5

Шифр	Название исследований	Уровень контроля		
		6	5	4
ЭКО	Ручной анализ участков исходного и восстановленного кода ОО		+	+
Примечание: «+» – исследования подлежат выполнению для соответствующего уровня контроля; «цифра» – дополнительные требования к исследованиям для соответствующего уровня контроля				

Задачей исследования является самостоятельное выявление методами ручного направленного анализа в модулях ОО:

недостатков архитектуры;

уязвимостей конфигурации;

уязвимостей кода;

недекларированных возможностей.

Исходными данными при проведении исследования являются:

исходные данные в соответствии с требованиями пункта 2.3 настоящей Методики;

сведения, полученные по результатам анализа документации и иных исходных данных, в том числе сведения об инструментах и шагах инструментирования или иных модификациях бинарного, интерпретируемого или байт-кода, способных исказить результаты применения средств статического анализа исходных кодов (ПОД.1);

исходные коды ОО;

исполняемый код ОО;

список модулей ОО, содержащих участки исходного кода, написанные с использованием подлежащих статическому анализу языков программирования, не поддерживаемых используемым статическим анализатором, либо содержащих ассемблерные вставки или целиком написанных на языке ассемблера (САО);

сведения о механизмах, препятствующих проведению динамического анализа (ДАО.1, ДАО.2).

Требования к проведению исследования

Испытательной лаборатории требуется выполнить экспертизу участков исходного и восстановленного кода, предусматривающую ручной направленный анализ кода в отношении, как минимум:

программных модулей, исполняемый код которых был подвергнут инструментированию или иным модификациям бинарного, интерпретируемого или байт-кода, способных исказить результаты применения средств статического анализа исходных кодов в процессе сборки;

незадействованного в ходе динамического анализа из-за встроенных в код механизмов, препятствующих динамическому анализу;

кода, выполняющегося в контексте браузера (JavaScript, ActiveX, WebAssembly и другого кода);

участков исходного кода, написанного с использованием языков программирования, не поддерживаемых используемым статическим анализатором;

участков исходного кода, написанных на языке ассемблера.

Не требуется выполнять ручной направленный анализ кода в отношении включенных в состав поверхности атаки ОО компонентов среды функционирования (а именно: виртуальная машина, библиотеки поддержки времени выполнения и стандартные библиотеки в составе виртуальной машины) интерпретируемых языков или языков, компилируемых в промежуточное представление.

Если при проведении динамического анализа были выявлены механизмы, препятствующие проведению исследований, то программы (программные модули), участки кода, содержащие такие механизмы защиты, должны быть проанализированы совместно с разработчиком ОО. В случае отказа разработчика ОО от анализа испытательной лабораторией могут быть самостоятельно разработаны решения по снятию механизмов защиты от исследований. Если разработчик ОО отказался снимать механизмы защиты, препятствующие проведению исследований, или предлагаемые разработчиком решения по снятию механизма защиты не позволяют провести исследования в полном объеме, делается вывод о невозможности дальнейших исследований.

Рекомендуется выполнять ручной анализ участков исходного кода ОО в отношении как можно большего числа участков исходного кода, составляющего поверхность атаки, либо реализующего функции безопасности ОО.

Испытательной лаборатории требуется выполнить ручной анализ участков исходного и восстановленного кода ОО на предмет наличия программных и архитектурных ошибок, а также заведомо неудачных программных реализаций, способствующих возникновению ошибок и уязвимостей следующих типов:

а) использование потенциально опасных программных интерфейсов;

б) наличие в ОО заведомо некачественного программного кода (заведомо некорректное оформление исходного кода, невыполнимые или тождественные условия, не удаленные отладочные сообщения и другой код);

в) небезопасная загрузки библиотек;

г) некорректная обработка ошибок;

д) недостаточное ограничение прав доступа посторонних субъектов к временным файлам;

е) недостаточно полная проверка недопустимости присутствия во входных данных фрагментов программ на интерпретируемых языках (SQL-инъекции, XSS-атаки и другие типы атак);

ж) некорректная обработка длин буферов с данными;

з) некорректная обработка ситуаций, когда вместо обычного файла ОО предоставляется именованный канал, символическая ссылка или иной файловый объект специального вида;

и) ненадлежащее обеспечение корректности совместного доступа к глобальным данным в параллельных вычислениях;

к) ненадлежащее хранение аутентификационных данных в оперативной или внешней памяти;

л) переполнение буферов;

м) предоставление пользователям избыточной функциональности до аутентификации;

н) предоставление пользователям избыточных полномочий;

о) утечки чувствительных данных в журнальные файлы;

п) утечки оперативной памяти, файловых дескрипторов, графических ресурсов и других ошибок;

р) целочисленные переполнения;

с) наличие неиспользуемых программных компонентов: функций, процедур, переменных, классов и других компонентов;

т) наличие бинарных вставок и кодировок, в которых данные вставки представлены.

Определение программных ошибок должно осуществляться на основании сведений, содержащихся в соответствующих базах данных (например, Common Weakness Enumeration или Fortify Taxonomy: Software Security Errors), а также руководствах и стандартах, отражающих лучшие практики разработки безопасных программ (например, SEICERTC++ Coding Standard или The CERT Oracle Secure Coding Standard for Java).

В ходе экспертизы кода помимо наличия программных ошибок должен проводиться контроль отсутствия фактов использования сторонних программ (программных модулей), не входящих в состав ОО при реализации ОО функций безопасности.

В ходе экспертизы участков исходного текста и восстановленного кода должен быть зафиксирован перечень проверяемых программных ошибок и критерии их выявления.

Испытательной лаборатории требуется выполнять ручной анализ исходного кода с использованием инструментов, обеспечивающих навигацию на уровнях модулей (файлов) и структурных элементов языка программирования анализируемой программы. Если код написан на языке программирования, для которого отсутствуют инструменты с такими возможностями, допускается проводить исследование с помощью штатных утилит работы с текстами и файлами, такими как ls, find, grep, cat, sort и других утилит.

Инструментальные средства для исследования физической структуры программы должны обеспечивать определение количества файлов исходного кода программы, их расположения по каталогам, типов файлов по языкам программирования, размера и количества строк в файлах, имен файлов, поиск файлов в дереве исходных кодов.

Инструментальные средства для исследования логической структуры программы должны обеспечивать:

а) определение сущностей программы: процедур, методов, классов, полей, глобальных переменных программы;

б) определение для каждой сущности: места определения в исходном файле, списка вложенных сущностей (методы и поля в классах, вложенные классы в процедурах и методах), объемлющей сущности-родителя;

в) определение мест использования сущностей в исходном коде программы с указанием файла, строки: вызов процедур, чтение и запись полей и глобальных переменных;

г) определение связей между сущностями в иерархии классов программы: список родителей и наследников заданного класса, список методов, переопределяющих данный метод, список методов, реализующих данный абстрактный метод;

д) определение внешних процедур и переменных (не определенных в исходном коде), интерфейсных процедур и переменных (могут быть вызваны или изменены извне программы).

При проведении экспертизы восстановленного исполняемого кода применяемый дизассемблер должен выполнять итеративное дизассемблирование, обеспечивающее лучшее покрытие дизассемблируемого кода в сравнении с линейным дизассемблированием. Рекомендуется применять программные инструменты, обеспечивающие восстановление и визуализацию потоков управления и данных на уровне исполняемого кода, либо декомпиляторы в язык высокого уровня.

Требуется выполнить действия, предписанные пунктами 2.7 настоящей Методики в отношении выявленных недостатков безопасности ОО.

Требования к результатам исследования

Результаты исследований фиксируются в материалах исследований в объеме, соответствующем требованиям пункта 5.2 настоящей Методики.

Содержание ЭКО:

Наименование исследования	Уровень контроля		
	6	5	4
ЭКО		+	+
Дополнительные требования к ЭКО			

5. Оформление результатов исследований по выявлению уязвимостей и недекларированных возможностей

5.1. Результаты проведения исследований по выявлению уязвимостей и НДВ ОО оформляются отдельным протоколом исследований, разрабатываемым в соответствии с Положением о системе сертификации средств защиты информации, утвержденным приказом ФСТЭК России от 3 апреля 2018 г. № 55. В протоколе должны быть отражены условия и результаты подготовительных действий и исследований, проведенных в соответствии с требованиями разделов 3 и 4 настоящей Методики.

Результаты, представленные в цифровой форме, должны прилагаться к протоколам на одном или нескольких электронных носителях.

5.2. Протокол проведения исследований должен включать:

а) общие положения, объект и цели исследований;

б) средства и условия проведения исследований;

в) технические результаты выполненных исследований;

г) результаты исследований;

д) перечень выявленных недостатков безопасности ОО;

е) выводы.

5.3. Для подтверждения результатов всех видов инструментальных исследований ОО в протоколы исследований должны быть включены технические результаты, необходимые для подтверждения выполненных действий и пояснения достигнутых результатов, а именно: файлы конфигураций и журналы (логи) запуска инструментов анализа, журналы (логи) работы с результатами анализа, разметка результатов анализа, скриншоты, а также результаты в соответствии с требованиями соответствующих пунктов разделов 3 и 4 настоящей Методики.

Каждое размеченное предупреждение должно быть снабжено комментарием, поясняющим принятое решение. Комментарий должен быть достаточен, чтобы другой участник процессов сертификационных испытаний мог понять основания для принятого решения без проведения тщательного анализа исходного кода или конфигурации ОО. Формулировка пояснений общего вида, таких как «Не эксплуатируемо», «Не несет угроз безопасности информации», «Не применимо» и иные формулировки, а также применение единообразной групповой разметки общего вида в отношении не полностью идентичных причин срабатывания анализатора не допускается и расценивается как некачественно выполненная разметка.

5.4. Требуется включить в материалы исследований подготовленные разработчиком ОО результаты исследований, выполненных в соответствии с планом поддержки безопасности заимствованных компонентов сертифицированного средства защиты.

5.5. Выявленные в ходе исследований недостатки безопасности, принятые разработчиком ОО меры по устранению недостатков, вердикты и рекомендации испытательной лаборатории по устранению комплексных недостатков подлежат фиксации в протоколах исследований в соответствии с формой, приведенной в таблице 6.

Сведения об уязвимостях ОО и его компонентов направляются разработчиком ОО в банк данных угроз безопасности информации ФСТЭК России.

Таблица 6

№ п/п	Наименование недостатка	Наименование компонента ОО	Принятые разработчиком меры по устранению недостатка	Вердикт испытательной лаборатории о корректности устранения

				недостатка

5.6. Протокол подписывается исследователем или группой исследователей, проводивших исследования по выявлению уязвимостей и НДВ. Исследователи несут ответственность за выводы, сделанные ими по результатам проведенных исследований.

Приложение 1 к Методике выявления
уязвимостей и недекларированных
возможностей в программном обеспечении

Формат представления перечня программных компонентов

Перечень программных компонентов в машиночитаемом формате представляется в соответствии со спецификацией CycloneDX[18] в нотации JSON[19] в соответствии с RFC 8259 в виде объекта, содержащего поля, описанные в таблице П.1. Допускается наличие дополнительных полей, соответствующих спецификации CycloneDX версии 1.6, версия которой указана в поле specVersion корневого объекта перечня программных компонентов.

Таблица П.1 – Требования к полям корневого объекта перечня программных компонентов

Имя поля	Тип	Описание	Требования к наличию
bomFormat	Строка	Указание спецификации документа. Должно содержать значение «CycloneDX»	Обязательно
specVersion	Строка	Версия спецификации документа. Должно содержать значение «1.6» или «1.7».	Обязательно
serialNumber	Строка	Уникальный серийный номер документа, сгенерированный в соответствии с RFC-4122. Например, «urn:uuid:3e671687-395b-41f5-a30f-a58921a69b79»	Опционально
version	Целое число	Версия документа. Первоначальное значение – 1. При модификациях должна увеличиваться на 1	Обязательно
metadata	Объект	Дополнительная информация о перечне и программном обеспечении. Требования к заполнению полей объекта описаны в таблице П.2	Обязательно
components	Массив объектов	Список компонентов. Если в программном обеспечении	Обязательно при наличии

[18] Спецификация CycloneDX https://ecma-international.org/publications-and-standards/standards/ecma-424.

[19] ISO/IEC 21778:2017 Информационная технология. Синтаксис обмена данными JSON.

Имя поля	Тип	Описание	Требования к наличию
		используется несколько версий одного и того же компонента, то для каждой версии компонента должна быть отдельная запись (в том числе для компонентов, заимствованных из других сертифицированных средств защиты информации). Не допускается объединять в одну запись компоненты, исходный код которых хранится в нескольких различных репозиториях систем управления версиями. Требования к заполнению полей объектов описаны в таблице П.5.	в составе программного обеспечения компонентов
annotations	Массив объектов	Произвольные комментарии, касающиеся компонентов, сервисов, уязвимостей или самого документа. Требования к заполнению полей объектов описаны в таблице П.4.	Опционально

Таблица П.2 – Требования к полям объекта, описывающего дополнительную информацию о перечне и программном обеспечении

Имя поля	Тип	Описание	Требования к наличию
timestamp	Строка	Дата формирования перечня в формате по ГОСТ Р 7.0.64-2018 (ИСО 8601:2004). Например, «2022-04-25T09:30:00Z»	Обязательно
component	Объект	Информация о ПО. Требования к заполнению полей объекта описаны в таблице П.3	Обязательно

Таблица П.3 – Требования к полям объекта, описывающего информацию о программном обеспечении

Имя поля	Тип	Описание	Требования к наличию
type	Строка	Тип ПО. Должно содержать одно из значений, определенных спецификацией CycloneDX, например следующие значения: application, framework, library, operating-system, device-driver, firmware.	Обязательно

name	Строка	Название программного обеспечения, совпадающее с названием программного обеспечения, указанным в Формуляре.	Обязательно
version	Строка	Версия программного обеспечения. Длина строки не должна превышать 1024 символа	Обязательно
manufacturer	Объект	Информация о производителе программного обеспечения. Требования к заполнению полей объекта описаны в таблице П.4	Обязательно

Таблица П.4 – Требования к полям объекта, описывающего информацию о разработчике программного обеспечения

Имя поля	Тип	Описание	Требования к наличию
name	Строка	Название разработчика программного обеспечения, совпадающее с названием организации, указанным в Формуляре.	Обязательно

Таблица П.5 – Требования к полям объекта, описывающего программный компонент

Имя поля	Тип	Описание	Требования к наличию
type	Строка	Описывает тип компонента. Допускается использовать следующие значения: application, framework, library, container, platform, operating-system, device, device-driver, firmware, file, machine-learning-model, data, cryptographic-asset	Обязательно
name	Строка	Наименование компонента	Обязательно
version	Строка	Версия компонента. Длина строки не должна превышать 1024 символа	Обязательно
purl	Строка	Идентификатор компонента в формате Package URL[20]	Обязательно
cpe	Строка	Идентификатор компонента в формате CPE	Опционально

[20] Спецификация формата https://github.com/package-url/purl-spec

Имя поля	Тип	Описание	Требования к наличию
externalReferences	Массив объектов	Массив объектов, описывающих источник получения компонента, его исходных кодов, документации и других данных. Требования к заполнению полей объектов описаны в таблице П.6. Среди элементов массива обязательно должен присутствовать хотя бы один объект с типом vcs (ссылка на репозиторий в системе контроля версий) или source-distribution (ссылка на архив) за исключение следующих случаев: - исходный код программного компонента полностью определяется исходным кодом его подкомпонентов, которые описаны в поле components данного объекта; - программный компонент заимствован из сертифицированного средства защиты информации, входящего в состав среды функционирования ОО, и в поле properties указано свойство GOST:provided_by, содержащее название данного сертифицированного СЗИ; - программный компонент не является компонентом с открытым исходным кодом и в поле licenses содержится запись с полем name, равным «Proprietary». В последнем случае, заимствованные подкомпоненты такого компонента должны быть описаны в поле components данного объекта.	Обязательно

Имя поля	Тип	Описание	Требования к наличию
properties	Массив объектов	Массив объектов, описывающих дополнительные свойства компонента. Требования к заполнению полей объектов описаны в таблице П.8. Среди элементов массива обязательно должны присутствовать объекты со свойствами GOST:attack_surface и GOST:security_function.	Обязательно
components	Массив объектов	Список подкомпонентов данного компонента. Требования к заполнению полей объектов описаны в таблице П.5.	Опционально

Таблица П.6 – Требования к полям объекта, описывающего источники получения и внешние ссылки компонента

Имя поля	Тип	Описание	Требования к наличию
type	Строка	Описывает тип внешнего ресурса. Должно содержать одно из значений, определенных спецификацией CycloneDX1.6, версия которой указана в поле specVersion корневого объекта перечня программных компонентов.	Обязательно
url	Строка	Если программный компонент является компонентом с открытым исходным кодом и поле type имеет значение vcs, то в поле url помещается унифицированный указатель ресурса URL для репозитория с исходными кодами компонента, из которого осуществлялось копирование исходных кодов (например, выполнением команды «git clone») в собственный репозиторий разработчика. Если программный компонент является компонентом с открытым исходным кодом и поле type имеет значение source-distribution, то в поле url помещается унифицированный указатель ресурса URL	Обязательно

		для архива с исходным кодом компонента или файла с исходным кодом компонента, который был использован для помещения исходных кодов компонента в собственный репозиторий разработчика.	
hashes	Массив объектов	Содержит список хэш-кодов содержимого внешнего ресурса, полученных при помощи различных алгоритмов. Требования к заполнению полей объекта описаны в таблице П.7. Если поле type имеет значение source-distribution, то данное поле является обязательным и среди его элементов обязательно должен присутствовать объект с идентификатором алгоритма Streebog-256, Streebog-512, STREEBOG-256 или STREEBOG-512.	Опционально

Таблица П.7 – Требования к полям объекта, описывающего хэш-коды содержимого внешнего ресурса

Имя поля	Тип	Описание	Требования к наличию
alg	Строка	Содержит идентификатор алгоритма, которым был вычислен хэш-код. Должно содержать одно из значений, определенных спецификацией CycloneDX или одно из следующих значений: STREEBOG-256 или STREEBOG-512.	Обязательно
content	Строка	Содержит значение хэш-кода. Если поле alg имеет значение Streebog-256, Streebog-512, STREEBOG-256 или STREEBOG 512, то поле content должно содержать хэш-код архива с исходными текстами компонента, посчитанный по алгоритму ГОСТ 34.11 – 2012 с длиной хэш-кода 256 и 512 бит соответственно.	Обязательно

Таблица П.8 – Требования к полям объекта, описывающего дополнительные свойства компонента

Имя поля	Тип	Описание	Требования к наличию
name	Строка	Название свойства компонента. Значение GOST:attack_surface описывает свойство принадлежности компонента к поверхности атаки. Значение GOST:security_function описывает свойство принадлежности компонента к реализующим функции безопасности. Значение GOST:source_langs описывает язык исходного кода компонента. Если компонент реализован на нескольких языках программирования, то каждый язык программирования указывается в виде отдельного свойства. Значение GOST:provided_by описывает название сертифицированного СЗИ.	Обязательно
value	Строка	Значение свойства компонента. Если поле name имеет значение GOST:attack_surface, то поле value должно содержать одно из трёх значений: 1. yes – компонент входит в непосредственную поверхность атаки; 2. indirect – компонент входит в косвенную поверхность атаки; 3. no – в иных случаях. Если поле name имеет значение GOST:security_function, то поле value должно содержать одно из трёх значений: yes – если функции компонента непосредственно реализуют функции безопасности средства защиты информации (например: принимают решение по возможности доступа субъекта к объекту; принимают решение о принадлежности анализируемого объекта к классу (например,	Обязательно

Имя поля	Тип	Описание	Требования к наличию
		вредоносного ПО, вредоносного трафика); формируют и осуществляют запись сведений о событиях в журнал; управляют функционалом безопасности ядра ОС и процессора, предоставляющим возможности виртуализации и т. п.); indirect – если функции компонента участвуют в реализации функций безопасности средства защиты информации, взаимодействуя с компонентами, реализующими функции безопасности средства защиты информации (например, осуществляя различные виды обработки/подготовки данных, выделение структур данных из массива байт, выполнение вычислений над значениями аргументов функции и т. п.), но при этом не принимают непосредственных решений о запуске, остановке, изменение режимов и параметров работы компонентов, реализующих функции безопасности; no – если функции компонента не участвуют в реализации функций безопасности средства защиты информации. Если поле name имеет значение GOST:source_langs, то поле value должно содержать одно из значений: 1C, Assembly, C, C#, C++, Clojure, Dart, Elixir, Erlang, F#, Fortran, Go, Groovy, Haskell, Java, JavaScript, Julia, Kotlin, Lisp, Lua, Nix, Objective-C, OCaml, Perl, PHP, PowerShell, Python, R, Ruby, Rust, Scala, Swift, TypeScript, WebAssembly. В случае отсутствия в глоссарии используемого языка программирования указывается его каноническое название.	

Имя поля	Тип	Описание	Требования к наличию
		Если поле name имеет значение GOST:provided_by, то поле value должно содержать название сертифицированного средства защиты информации, из состава которого заимствован данный компонент, при выполнении следующих условий: - указанное средство защиты информации обладает действующим сертификатом соответствия требованиям безопасности ФСТЭК России; - указанное средство защиты информации является составляющей среды функционирования ОО; - компонент в средстве, из которого происходит заимствование, отнесен к непосредственной поверхности атаки средства, если данный компонент входит в непосредственную поверхность атаки ОО; - компонент в средстве, из которого происходит заимствование, непосредственно реализует функции безопасности средства, если данный компонент непосредственно реализует функции безопасности ОО.	

Приложение 2 к Методике выявления
уязвимостей и недекларированных
возможностей в программном обеспечении

Формат представления перечня образов контейнеров в машиночитаемом формате

Перечень образов контейнеров в машиночитаемом формате представляется в соответствии со спецификацией CycloneDX в нотации JSON в соответствии с RFC 8259 в виде объекта, содержащего поля описанные в таблице П.9. Допускается наличие дополнительных полей, соответствующих спецификации CycloneDX, версия которой указана в поле specVersion корневого объекта перечня программных компонентов.

Таблица П.9 – Требования к полям корневого объекта перечня образов контейнеров в машиночитаемом формате

Имя поля	Тип	Описание	Требования к наличию
bomFormat	Строка	Спецификация формата документа. Должно содержать значение «CycloneDX».	Обязательно
specVersion	Строка	Версия спецификации документа. Должно содержать значение «1.6» или «1.7».	Обязательно
serialNumber	Строка	Уникальный серийный номер документа, сгенерированный Разработчиком в соответствии с RFC-4122. Например, «urn:uuid:3e671687-395b-41f5-a30f-a58921a69b79»	Опционально
version	Целое число	Версия документа. Первоначальное значение – 1. При модификациях должна увеличиваться на 1.	Обязательно
metadata	Объект	Дополнительная информация о перечне и продукте. Требования к заполнению полей объекта описаны в таблице П.10.	Обязательно
components	Массив объектов	Список образов контейнеров, входящих в состав средства.	Обязательно

		Требования к заполнению полей объектов описаны в таблице П.13.	

Таблица П.10 – Требования к полям объекта, описывающего дополнительную информацию о перечне и продукте

Имя поля	Тип	Описание	Требования к наличию
timestamp	Строка	Дата и время формирования перечня в формате ГОСТ Р 7.0.64-2018 (ИСО 8601:2004). Например, «2022-04-25T09:30:00Z»	Обязательно
component	Объект	Информация о продукте. Требования к заполнению полей объекта описаны в таблице П.11.	Обязательно

Таблица П.11 – Требования к полям объекта, описывающего информацию о продукте

Имя поля	Тип	Описание	Требования к наличию
type	Строка	Описывает тип продукта. Допускается использовать следующие значения: application, framework, library, operating-system, device-driver, firmware. Данное поле является обязательным для совместимости со спецификацией CycloneDX, выбор типа компонента осуществляется по усмотрению разработчика.	Обязательно
name	Строка	Название программного обеспечения, совпадающее с названием программного обеспечения, указанным в Формуляре.	Обязательно
version	Строка	Версия продукта. Длина строки не должна превышать 1024 символа.	Обязательно
manufacturer	Объект	Информация об изготовителе продукта. Требования к заполнению полей объекта описаны в таблице П.12.	Обязательно

Таблица П.12 – Требования к полям объекта, описывающего информацию об изготовителе продукта

Имя поля	Тип	Описание	Требования к наличию
name	Строка	Название разработчика программного обеспечения, совпадающее с названием организации, указанным в Формуляре.	Обязательно

Таблица П.13 – Требования к полям объекта, описывающего образ контейнера.

Имя поля	Тип	Описание	Требования к наличию
type	Строка	Описывает тип компонента. Должно содержать значение «container».	Обязательно
name	Строка	Наименование образа контейнера.	Обязательно
version	Строка	Версия образа контейнера.	Обязательно
description	Строка	Назначение образа контейнера.	Обязательно
purl	Строка	Опциональный идентификатор компонента в формате Package URL.	Опционально
properties	Массив объектов	Массив объектов, описывающих свойства компонента. Требования к заполнению полей объектов описаны в таблице П.14. Среди элементов массива обязательно должны присутствовать объекты со свойствами GOST:attack_surface и GOST:security_function.	Обязательно
components	Массив объектов	Список программного обеспечения, входящего в состав образа контейнера. Требования к заполнению полей объектов описаны в таблице П.15.	Обязательно

Таблица П.14 – Требования к полям объекта, описывающего свойства компонентов.

Имя поля	Тип	Описание	Требования к наличию
name	Строка	Название свойства компонента.	Обязательно

Имя поля	Тип	Описание	Требования к наличию
value	Строка	Значение свойства компонента. Если поле name имеет значение GOST:attack_surface, то поле value должно содержать одно из трёх значений: yes – в случае, если подпрограммы (функции) компонента реализуют программный интерфейс, который непосредственно доступен потенциальному нарушителю или входные данные которого потенциальный нарушитель способен гарантированно сформировать определенным образом; indirect – в случае, если подпрограммы (функции) компонента реализуют программный интерфейс, на входные данные которого потенциальный нарушитель способен целенаправленно повлиять, но это влияние существенно ограничено теми или иными факторами; no – в противном случае. Если поле name имеет значение GOST:security_function, то поле value должно содержать одно из трёх значений: yes – если функции компонента непосредственно реализуют функции безопасности средства защиты информации (например, принимают решение по возможности доступа субъекта к объекту; принимают решение о принадлежности анализируемого объекта к классу (например, вредоносного ПО, вредоносного трафика); формируют и осуществляют запись сведений о событиях в журнал; управляют функционалом безопасности ядра ОС и процессора, предоставляющим возможности виртуализации и т. п.);	Обязательно

Имя поля	Тип	Описание	Требования к наличию
		indirect – если функции компонента участвуют в реализации функций безопасности средства защиты информации, взаимодействуя с компонентами, реализующими функции безопасности средства защиты информации (например, осуществляя различные виды обработки/подготовки данных, выделение структур данных из массива байт, выполнение вычислений над значениями аргументов функции и т. п.), но при этом не принимают непосредственных решений о запуске, остановке, изменении режимов и параметров работы компонентов, реализующих функции безопасности; no – если функции компонента не участвуют в реализации функций безопасности средства защиты информации. Если поле name имеет значение GOST:provided_by, то поле value должно содержать название сертифицированного средства защиты информации, из состава которого заимствован данный компонент, при выполнении следующих условий: - указанное средство защиты информации обладает действующим сертификатом соответствия требованиям безопасности ФСТЭК России; - указанное средство защиты информации является составляющей среды функционирования ОО; - компонент в средстве, из которого происходит заимствование, отнесен к непосредственной поверхности атаки средства, если данный компонент входит в непосредственную поверхность атаки ОО;	

Имя поля	Тип	Описание	Требования к наличию
		- компонент в средстве, из которого происходит заимствование, непосредственно реализует функции безопасности средства, если данный компонент непосредственно реализует функции безопасности ОО.	

Таблица П.15 – Требования к полям объекта, описывающего программное обеспечение, входящее в состав образа контейнера.

Имя поля	Тип	Описание	Требования к наличию
type	Строка	Описывает тип компонента. Допускается использовать следующие значения: application, framework, library, container, platform, operating-system, device, device-driver, firmware, file, machine-learning-model, data, cryptographic-asset. Данное поле является обязательным для совместимости со спецификацией CycloneDX, выбор типа компонента осуществляется по усмотрению разработчика.	Обязательно
name	Строка	Наименование компонента.	Обязательно
version	Строка	Версия компонента.	Обязательно
purl	Строка	Идентификатор компонента в формате Package URL[21] Поле является обязательным для компонентов, поле components которых не содержит описания вложенных подкомпонентов.	Опционально
properties	Массив объектов	Массив объектов, описывающих свойства компонента. Требования к заполнению полей объектов описаны в таблице П.14. Среди элементов массива обязательно должны присутствовать объекты со свойствами GOST:attack_surface и GOST:security_function.	Обязательно

[21] Спецификация формата https://github.com/package-url/purl-spec

components	Массив объектов	Список подкомпонентов данного компонента. Допускается перечислять подкомпоненты как в массиве components родительского компонента, так и в массиве components корневого объекта перечня. Требования к заполнению полей объектов описаны в таблице П.15.	Опционально

Приложение 3 к Методике выявления
уязвимостей и недекларированных возможностей
в программном обеспечении

Модельный пример анализа поверхности атаки

1. Краткое описание модельного ОО

ОО является средством защиты информации «Ромашка» представляет собой программный комплекс, предназначенный для автоматизированного выявления уязвимостей в сторонних приложениях и системах. ОО осуществляет комплексное сканирование свободного ПО, используя современные методы анализа, включая сигнатурный поиск по известным шаблонам уязвимостей (рисунок П.1).

2. Внешние интерфейсы (ИФБО)

EXT.1 – Пользовательский веб-интерфейс реализован модулями Nginx, vue.js, Node.js TCP-сокет, внешний контур, основной UI для пользователей;

EXT.2 – Прикладной программный интерфейс реализован модулями Nginx, FastAPI, Backend, RabbitMQ, TCP-сокет, внешний контур REST API для интеграций;

EXT.3 – Интерфейс управления Главного Администратора, реализован модулями Nginx, Backend, RabbitMQ, TCP-сокет внешний контур, защищен СКЗИ «точка-точка»;

EXT.4 – Служебный интерфейс обновления БДУ Сканер уязвимостей, реализован модулем Romashka-Vuln-Scanner, TCP-сокет ограниченный по времени, внешний контур;

EXT.5 – Исходящий сетевой интерфейс SNMP-протокола Мониторинг состояния, реализован модулем Romashka-Observer, TCP-сокет, Внешний контур, однонаправленная отправка данных;

EXT.6 – Сетевой интерфейс обновления реализован модулем Romashka-Vuln-DB-Updater, TCP-сокет, Внешний контур;

EXT.7 – Клиентский интерфейс, реализован модулем Romashka-Agent, TCP-сокет, Внешний контур.

3. Подробное описание ОО

1. Подсистема проксирования запросов

Назначение: Обработка HTTP-запросов пользователей системы.

Модули:

Nginx (язык: C) – балансировщик нагрузки и обратный прокси;

FastAPI (язык: Python) – высокопроизводительный API-шлюз;

Использует фреймворк Starlette.

Требования безопасности:

для УД.4 и выше: испытания в объеме САО.1, ДАО.1, ДАО.2;

находится на непосредственной поверхности атаки.

2. Подсистема веб-интерфейса

Назначение: Отображение пользовательского интерфейса и обеспечение функциональности СЗИ.

Модули:

vue.js (язык: JavaScript) – фронтенд-фреймворк с Server-Side Rendering;

Node.js (язык: C) – среда исполнения JavaScript.

Требования безопасности:

Модуль Nginx: САО.1, ДАО.1, ДАО.2 (непосредственная поверхность атаки);

Веб-интерфейс на АРМ: КАО.1 (непосредственная поверхность атаки);

Node.js: САО.1, ДАО.1, ДАО.2 (среда исполнения JavaScript);

3. Подсистема управления

Назначение: Настройка СЗИ, управление правами пользователей.

Модули:

Nginx (язык: C) – веб-сервер для административного интерфейса;

RabbitMQ (языки: JavaScript, Erlang) – брокер сообщений;

Romashka-Control (язык: Rust) – основной backend-модуль управления;

СКЗИ – средства криптографической защиты информации.

Требования безопасности:

Интерфейс EXT.3 защищен шифрованием «точка-точка».

4. Подсистема управления очередями

Назначение: Управление задачами на проведение сканирования.

Модули:

Romashka-Scheduler (язык: Go) – планировщик задач сканирования;

Romashka-Worker (язык: Go) – исполнитель задач сканирования.

5. Подсистема сканирования

Назначение: Реализация функции по сканированию сторонних приложений и систем.

Модули:

Romashka-Vuln-Scanner (язык: Go) – основной сканер уязвимостей;

Romashka-Vuln-DB-Updater (язык: Go) – модуль обновления БД уязвимостей.

Требования безопасности:

Для УД.4 и выше: Romashka-Vuln-Scanner – САО.1, ДАО.1, ДАО.2;

Реализует ограниченный по времени служебный интерфейс.

6. Подсистема авторизации

Назначение: Управление доступом и аутентификация.

Модули:

Keycloak (язык: Java) – система управления доступом и идентификации;

OpenJDK (язык: C++) – среда исполнения Java-приложений.

7. Клиентская подсистема

Назначение: Агентские компоненты на защищаемых системах.

Модули:

Romashka-Agent (язык: Go) – клиентский агент для сканирования.

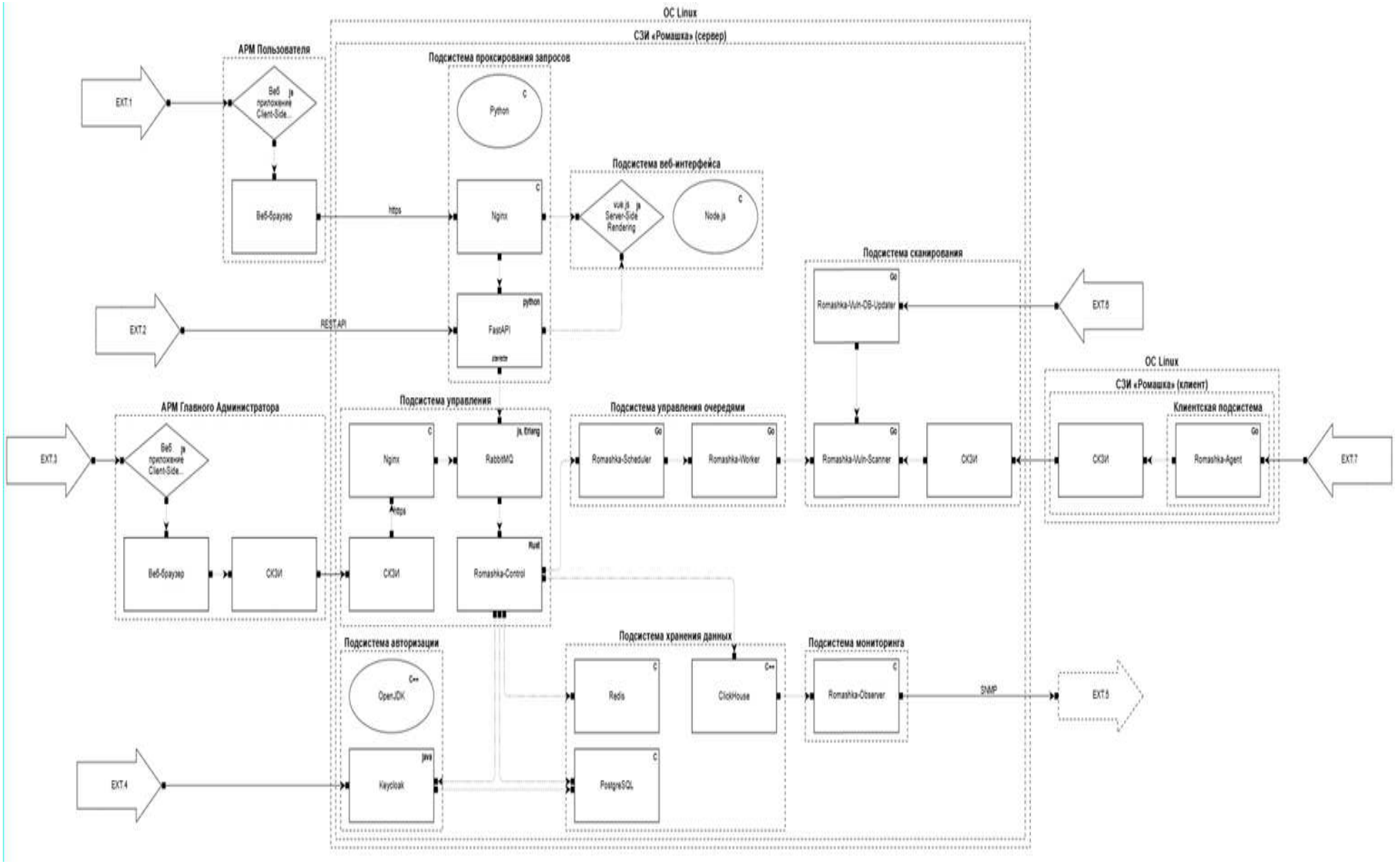

Рисунок П.1 – Схема архитектуры модельного примера

4 Анализ поверхностей атаки и требования к испытаниям

Непосредственная поверхность атаки (УД.4 и выше) представлена на рисунке П.2:

модуль Nginx «Подсистемы проксирования запросов»;

модуль FastAPI «Подсистемы проксирования запросов»;

модуль «Веб-интерфейс» на стороне АРМ Пользователя;

модуль «Node.js» «Подсистемы веб-интерфейса»;

модуль «Сканер уязвимостей» «Подсистемы сканирования»;

модуль «Romashka-Vuln-DB-Updater» «Подсистемы сканирования».

Объем испытаний: САО.1, ДАО.1, ДАО.2

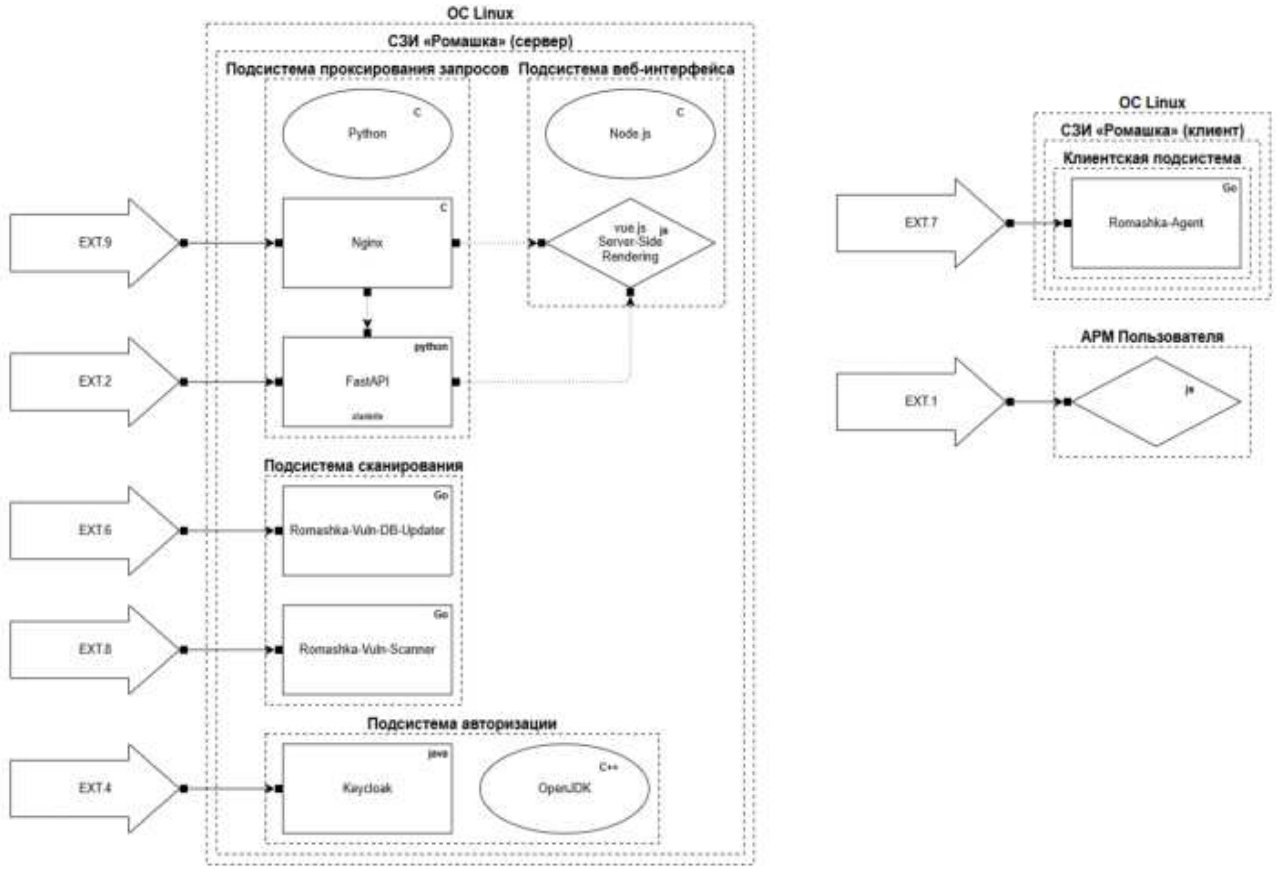

Рисунок П.2 – Схема поверхности атаки модельного примера для УД.4