
**ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ**

**НАЦИОНАЛЬНЫЙ
СТАНДАРТ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

ГОСТ Р
*(проект,
первая редакция)*

Защита информации

РАЗРАБОТКА БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Композиционный анализ программного обеспечения

Общие требования

Настоящий проект стандарта не подлежит применению до его утверждения

Москва

202X

Предисловие

1 РАЗРАБОТАН Федеральной службой по техническому и экспортному контролю (ФСТЭК России), Федеральным государственным бюджетным учреждением науки Институт системного программирования имени В. П. Иванникова Российской академии наук (ФГБУН «ИСП РАН»), Обществом с ограниченной ответственностью «Профископ» (ООО «Профископ») и Обществом с ограниченной ответственностью Научно-технический центр «Фобос-НТ» (ООО НТЦ «Фобос-НТ»), Федеральным автономным учреждением «Государственный научно-исследовательский испытательный институт проблем технической защиты информации Федеральной службы по техническому и экспортному контролю» (ФАУ «ГНИИИ ПТЗИ ФСТЭК России»)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 362 «Защита информации»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ приказом Федерального агентства по техническому регулированию и метрологии от « » _____ 202X г. №

4 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.rst.gov.ru).

© Оформление. ФГБУ «Институт стандартизации», 202_

Настоящий стандарт не может быть воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии.

Содержание

1 Область применения.....	
2 Нормативные ссылки.....	
3 Термины и определения, обозначения и сокращения.....	
4 Общие положения.....	
5 Требования к порядку внедрения и применения композиционного анализа программного обеспечения.....	
5.1 Порядок выполнения композиционного анализа.....	
5.2 Требования к этапу подготовки внедрения композиционного анализа программного обеспечения.....	
5.3 Требования к этапу внедрения композиционного анализа программного обеспечения.....	
5.4 Требования к этапу регулярного применения композиционного анализа программного обеспечения.....	
6 Требования к технологиям композиционного анализа программного обеспечения.....	
7 Требования к инструменту композиционного анализа программного обеспечения.....	
8 Требования к перечню программных компонентов.....	
Приложение А (обязательное) Формат представления перечня программных компонентов.....	
Приложение Б (обязательное) Состав данных описания модели машинного обучения.....	

Введение

Безопасность программного обеспечения (ПО) зависит от стороннего ПО, заимствованного для включения в состав разрабатываемого ПО или привлекаемого для обеспечения процессов разработки. Наличие в стороннем ПО уязвимостей и недекларированных возможностей, а также определенных особенностей способно негативно влиять на развитие и поддержку разрабатываемого ПО, создавать угрозы безопасности информации. Противодействуют этому процессы разработки безопасного ПО и программные инструменты композиционного анализа, которые способствуют повышению безопасности заимствованного и привлекаемого ПО.

Процесс композиционного анализа позволяет установить состав программных компонентов, соотнести его с известными уязвимостями и нежелательными особенностями использования этих компонентов, своевременно отреагировать на выявленные уязвимости и особенности использования.

Настоящий стандарт вводит понятие контролируемого репозитория, хранилища стороннего ПО, проходящего проверки, и задает требования безопасности в отношении стороннего ПО, применяемого в разработке. Стандарт вводит понятие перечня программных компонентов и устанавливает требования, предъявляемые к процессам композиционного анализа и проверки кода на предмет внедрения вредоносного программного обеспечения через цепочки поставок, а также – к применяемым инструментам композиционного анализа. Стандарт задает требования к формату представления перечня программных компонентов.

Настоящий стандарт применяется совместно с ГОСТ Р 56939-2024 и входит в комплекс стандартов, направленных на достижение целей, связанных с повышением безопасности разрабатываемого ПО.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Защита информации

РАЗРАБОТКА БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Композиционный анализ программного обеспечения

Общие требования

Information protection. Secure software development.

Software composition analysis. General requirements

Дата введения - _____

1 Область применения

Настоящий стандарт устанавливает общие требования к содержанию и порядку выполнения работ, направленных на предотвращение появления либо нейтрализацию уязвимостей и недекларированных возможностей сторонних компонентов и выявление особенностей их использования в программном обеспечении (ПО).

Примечание – К особенностям использования сторонних компонентов в ПО относятся аспекты, связанные с наличием компрометирующих конструкций в исходных кодах, соблюдением требований лицензионных соглашений, наличием стабильного коллектива разработчиков, а также иные, определяющие степень доверия к сторонним компонентам.

Применение настоящего стандарта позволит улучшить оперативность выявления известных уязвимостей и упростить внедрение специализированных процессов разработки безопасного ПО, позволит повысить эффективность мер по противодействию угрозам безопасности ПО, стандартизировать информационный обмен между участниками цепочки поставки ПО.

Настоящий стандарт предназначен:

- для организаций-разработчиков программного обеспечения, в том числе – разработчиков средств защиты информации, средств обеспечения безопасности информационных технологий;
- для организаций-разработчиков инструментальных средств, применяемых в процессах композиционного анализа ПО и обеспечения безопасности цепочки поставок;
- для организаций, выполняющих оценку соответствия процессов разработки безопасного ПО в части выполнения требований настоящего стандарта.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ 34.003-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.

ГОСТ Р 7.0.64-2018 (ИСО 8601:2004) Система стандартов по информации, библиотечному и издательскому делу. Представление дат и времени. Общие требования

ГОСТ Р 34.11-2012 Информационная технология. Криптографическая защита информации. Функция хэширования

ГОСТ Р 56939-2024 Защита информации. Разработка безопасного программного обеспечения. Общие требования.

Примечание – При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный стандарт, на

который дана недатированная ссылка, то рекомендуется использовать действующую версию этого документа с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого документа с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения, обозначения и сокращения

3.1 В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1.1 база данных уязвимостей: Структурированное хранилище информации об уязвимостях программного обеспечения.

3.1.2 заимствованный компонент (программного обеспечения): Полученный (приобретенный) на определённых условиях (в том числе – безвозмездно) компонент разрабатываемого ПО.

Примечания

1 Заимствованный компонент ПО может быть в виде:

- включаемых в процесс компиляции исходных кодов;
- объектного кода статических библиотек и динамически загружаемых модулей;
- кода интерпретируемых программных модулей.

2 Совместно с кодом в составе заимствованного компонента могут поставляться также сопутствующие данные: текст, изображение, аудио-, видеоинформация или иное представление данных.

3.1.3

<p>известная уязвимость: Уязвимость, опубликованная в общедоступных источниках с описанием соответствующих мер защиты информации, исправлений недостатков или соответствующих обновлений.</p>
--

<p>[[1], пункт 3.7]</p>

3.1.4 инструмент композиционного анализа программного обеспечения: Инструментальное средство программной инженерии, которое выполняет композиционный анализ ПО в автоматизированном или автоматическом режиме.

3.1.5

исходный код программы: Программа в текстовом виде на каком-либо языке программирования.

[[2], пункт 3.2]

3.1.6 композиционный анализ программного обеспечения: Анализ, основанный на инвентаризации сторонних компонентов ПО, определении особенностей их использования, составлении перечня известных уязвимостей и/или иных недостатков компонентов.

3.1.7 компрометирующая конструкция: Недостаток ПО, реализованный в виде содержащего противоправную информацию комментария в исходном коде или сопутствующих данных.

3.1.8

компонент (программного обеспечения): Программа, рассматриваемая как единое целое, выполняющая законченную функцию.

[Адаптировано из [3], пункт 1.2]

Примечание – Программный пакет является основной, но не единственной формой компонента ПО. Заимствование и включение в состав разрабатываемого ПО компонента может происходить на уровне отдельных файлов или отдельных фрагментов кода.

3.1.9 контролируемый репозиторий: репозиторий, в отношении содержимого которого выполняются проверки, направленные на повышение безопасности разрабатываемого ПО и предусмотренные регламентами и иными организационно-распорядительными документами организации.

3.1.10 манифест системы управления компонентами:

Машиночитаемый конфигурационный документ, определяющий применяемые в процессе сборки сторонние компоненты, этапы и способы сборки.

Примечание – Формируется автоматизированным способом с применением системы управления компонентами, средствами разработки или ручным способом (редактируется разработчиком).

3.1.11 обогащение (метаданных): Дополнение описания объекта (события) тематическим блоком стандартизованных атрибутов, как правило, получаемым от информационной системы, анализирующей соответствующую тематику, в целях улучшения полноты и точности результатов.

3.1.12 пакетная экосистема: Децентрализованное хранилище компонентов, обеспечивающее их доступность конечным пользователям и разработчикам программного обеспечения, в том числе посредством системы управления компонентами.

Примечание – Пакетные экосистемы поддерживаются организациями, частными компаниями или некоммерческими сообществами разработчиков программного обеспечения и, как правило, формируются для конкретного языка программирования.

3.1.13 перечень программных компонентов; ППК:

Машиночитаемый документ, содержащий в себе структурированную информацию о компонентах программного обеспечения и отношениях между ними.

Примечание – Перечень программных компонентов является отечественным аналогом англоязычного термина SBoM (Software Bill of Materials (набор структурированных машиночитаемых данных о составе программного обеспечения)).

3.1.14

поверхность атаки (программного обеспечения): Множество подпрограмм (функций) программного обеспечения, обрабатывающих

данные из интерфейсов, непосредственно или косвенно подверженных потенциальному риску атаки.

Примечания

1 В состав поверхности атаки входят подпрограммы (функции) API (Application programming interface, прикладной программный интерфейс) ПО, которые непосредственно доступны потенциальному нарушителю. Указанные подпрограммы (функции) относят к непосредственной поверхности атаки.

2 В состав поверхности атаки входят подпрограммы (функции) ПО, которые не могут быть непосредственно вызваны потенциальным нарушителем через API, но их входные данные потенциальный нарушитель способен гарантированно сформировать определенным образом. Указанные подпрограммы (функции) также относят к непосредственной поверхности атаки.

3 В состав поверхности атаки входят подпрограммы (функции) ПО, которые не могут быть непосредственно вызваны потенциальным нарушителем через API, но на их входные данные потенциальный нарушитель способен целенаправленно повлиять. Указанные подпрограммы (функции) относят к косвенной поверхности атаки.

[Адаптировано из ГОСТ 56939-2024, пункт 3.9]

3.1.15

политика безопасности (в отношении сторонних компонентов):

Совокупность правил, процедур или руководящих принципов в области безопасности использования сторонних компонентов, действующих в организации.

[Адаптировано из [4], подпункт 3.1.49]

3.1.16 привлекаемое (программное обеспечение): Полученное (приобретенное) на определённых условиях, в том числе – безвозмездно, ПО, не являющееся заимствованным компонентом, но используемое в процессе разработки ПО как средство разработки или являющееся частью среды функционирования разрабатываемого ПО.

3.1.17 разработчик программного обеспечения: Лицо, осуществляющее деятельность по разработке безопасного ПО.

3.1.18

регламент композиционного анализа (в организации):

Регламент осуществления процесса разработки безопасного ПО с использованием стороннего ПО, который содержит информацию об обязанностях сотрудников и их ролях при реализации процесса композиционного анализа в соответствии с принятыми политиками безопасности в отношении сторонних компонентов. Может быть оформлен как самостоятельный документ в электронном или физическом виде, или в рамках общего документа, например, руководства по разработке безопасного ПО.

[Адаптировано из ГОСТ Р 56939-2024, пункт 4.12]

3.1.19 репозиторий: Автоматизированная система, которая обеспечивает хранение, доступ и манипулирование данными (файлами) и метаданными.

Примечание – Для обеспечения процессов разработки ПО может применяться несколько репозиториев, в которых размещаются определенные виды данных: исходные коды, компоненты в бинарном или исходном коде, и иные сопутствующие файлы разрабатываемого ПО. Технические возможности репозитория могут позволить совместно разместить в нем разнородные данные.

3.1.20 репозиторий компонентов (артефактов): Репозиторий, обеспечивающий цепочку поставки компонентов (артефактов), в котором размещены компоненты (артефакты) в бинарном виде или исходных кодах, инструкции, выполняемые для регистрации/настройки программ при установке, обновлении и удалении компонентов ПО, информация, описывающая взаимосвязь между компонентами ПО, а также метаданные в форматах, поддерживаемых пакетными менеджерами.

3.1.21 репозиторий разработки: Репозиторий, в котором размещены исходные коды и сопутствующие данные ПО, инструкции системы сборки и сборочной среды ПО, обеспечивающий управление их изменениями.

3.1.22 система контроля версий: Программное обеспечение, предназначенное для управления изменениями ПО, включая уникальную идентификацию изменений, вносимых в исходные коды и иные сопутствующие файлы разрабатываемого ПО участниками разработки при их совместной работе.

3.1.23 система управления компонентами: Набор программного обеспечения, позволяющий управлять процессом установки, удаления, сборки, тестирования, конфигурирования и обновления компонентов программного обеспечения.

Примечание – Системы управления компонентами также известны под нестандартизованным названием «пакетные менеджеры».

3.1.24 стороннее программное обеспечение, стороннее ПО: Совокупность заимствованных компонентов ПО и привлекаемого ПО.

3.1.25

<p>хэш-код (hash-code): Строка бит, являющаяся выходным результатом хэш-функции.</p>

<p>[ГОСТ Р 34.11-2012, статья 3.1.5]</p>
--

3.1.26 цепочка поставки программного обеспечения: Совокупность организационно-технических мер и действий по обеспечению необходимыми компонентами процессов сборки и доставки поставляемого ПО.

3.2 В настоящем стандарте применены следующие обозначения и сокращения:

БДУ – банк данных угроз безопасности информации ФСТЭК России;

Инструмент – инструмент или набор инструментов анализа и (или) контроля сторонних компонентов;

Лицензия – лицензионное соглашение компонента;

Организация – лицо, являющееся разработчиком программного обеспечения;

ПО – программное обеспечение;

ППК – перечень программных компонентов;

API – Application Programming Interface (прикладной программный интерфейс, описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другими программами);

CycloneDX – формат передачи данных и метаданных о компонентах, который поддерживается некоммерческой организацией OWASP (Open Web Application Security Project, открытый проект по безопасности веб-приложений) и определен спецификацией ECMA-424 (European Computer Manufacturers Association, европейская ассоциация производителей компьютеров);

CVSS – Common Vulnerability Scoring System (общая система оценки уязвимости);

JSON – JavaScript Object Notation (текстовый формат обмена данными, основанный на языке программирования JavaScript);

GHSA – GitHub Security Advisory (база данных уязвимостей, поддерживаемая компанией GitHub);

NVD – National Vulnerability Database (база данных уязвимостей, поддерживаемая Национальным институтом стандартов и технологий США);

OCI – Open Container Initiative (инициатива открытого контейнера);

OSSIndex – Open Source Software Index (база данных уязвимостей, поддерживаемая компанией Sonatype);

OSV – Open Source Vulnerability (база данных уязвимостей, поддерживаемая корпорацией Google);

PURL – Package Uniform Resource Locator (формат¹ универсального идентификатора компонента);

SCA – Software Composition Analysis (анализ состава программного обеспечения, композиционный анализ программного обеспечения);

SPDX – Software Package Data Exchange (обмен данными программных пакетов).

¹ Спецификация формата <https://github.com/package-url/purl-spec>

4 Общие положения

4.1 Предотвращение появления и устранение уязвимостей программ достигается путем реализации их разработчиком комплекса мер по разработке безопасного ПО, представленных в ГОСТ Р 56939-2024. Настоящий стандарт фокусируется на предотвращении появления либо нейтрализации уязвимостей и недекларированных возможностей в стороннем ПО.

4.2 При разработке безопасного ПО, разработчик выполняет композиционный анализ в соответствии с требованиями разделов 5-8 в дополнение к положениям ГОСТ Р 56939-2024.

4.3 Условия реализации композиционного анализа выражены в форме следующих сущностей: требование, рекомендация или допустимое действие. С целью подчеркнуть различие между разными формами условий для реализации процессов разработки безопасного ПО в настоящем стандарте используются вспомогательные глаголы «должен», «рекомендуется» и «может». Глагол «должен» – для выражения условия, требуемого для соответствия требованию; «рекомендуется» — для выражения рекомендации по реализации, «может» — для того чтобы отразить возможные направления допустимых действий.

4.4 Для соответствия требованиям настоящего стандарта разработчик должен реализовывать процессы композиционного анализа и проверки кода на предмет внедрения вредоносного программного обеспечения через цепочки поставок при использовании сторонних компонентов ПО в ходе разработки и последующего сопровождения. Реализуемые процессы должны обеспечивать ведение перечня программных компонентов ПО и выявление известных уязвимостей, а также особенностей использования программных компонентов. Своевременное выявление в ПО известных уязвимостей или нарушений

порядка использования компонентов способствует оперативному устранению недостатков в ПО или разработке компенсирующих мер и, соответственно, снижению угроз безопасности информации, обрабатываемой с использованием ПО.

4.5 Разрабатываемое ПО, ПО среды функционирования² и программные средства разработки размещаются в контролируемом репозитории разработчика ПО, цепочка поставок которых представлена на рисунке 1.

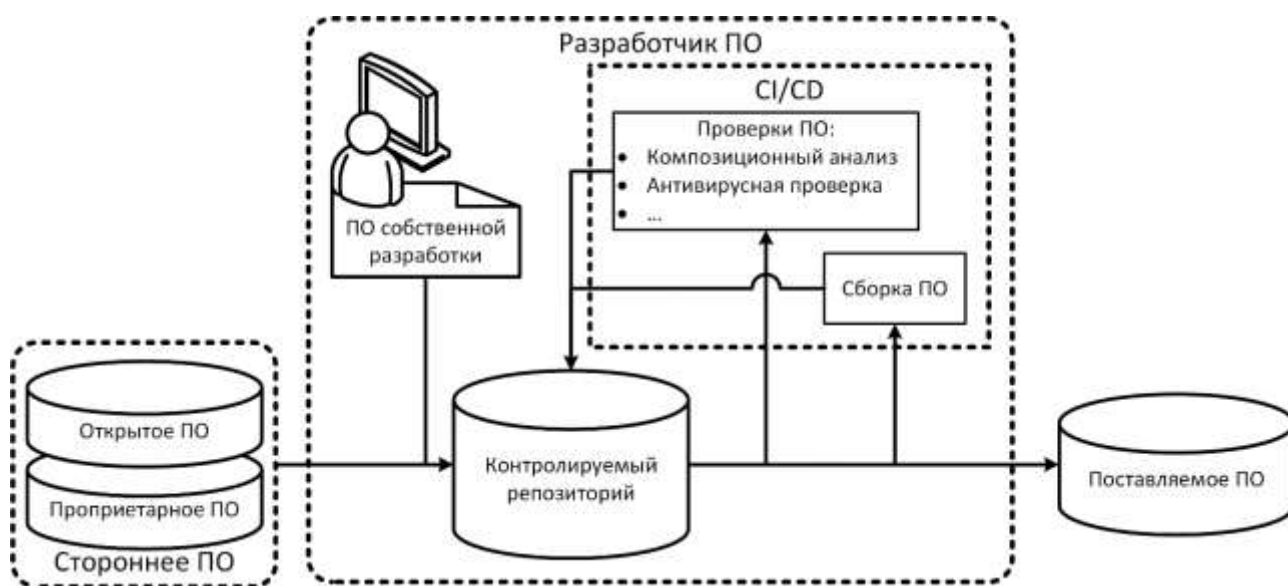


Рисунок 1 – Цепочка поставки программного обеспечения

4.6 В контролируемом репозитории разработчика ПО хранятся как компоненты ПО собственной разработки, так и стороннее ПО. Для хранения ПО может использоваться как один, так и несколько репозиториях.

² Если разработчик ПО ведет разработку нескольких программных продуктов, в отношении одного из них другой программный продукт может являться компонентой среды функционирования или средством разработки. Пример первой ситуации – когда разработчик наряду с приложениями ведет разработку операционной системы, второй ситуации – когда разрабатывает компилятор.

4.7 Использование в разработке любой разновидности стороннего ПО, требует его размещения в контролируемом репозитории и проведения проверок безопасности ПО.

4.8 Требования к порядку внедрения и применения разработчиком процессов композиционного анализа ПО и проверки кода на предмет внедрения вредоносного ПО через цепочки поставок приведены в разделе 5.

4.9 Описание композиционного анализа и требования, предъявляемые к технологиям композиционного анализа, приведены в разделе 6.

4.10 Требования к инструменту композиционного анализа приведены в разделе 7.

4.11 Требования к перечню программных компонентов приведены в разделе 8.

4.12 При отсутствии одного инструмента композиционного анализа ПО, отвечающего всем требованиям разделов 6 и 7, должен использоваться набор инструментов, в том числе – входящие в состав системы сборки ПО, наиболее полно удовлетворяющий данным требованиям.

5 Требования к порядку внедрения и применения композиционного анализа программного обеспечения

5.1 Порядок выполнения композиционного анализа

5.1.1 Внедрение и применение композиционного анализа ПО состоит из трех этапов: этап подготовки внедрения, этап внедрения, этап регулярного применения.

5.1.2 Рекомендуемый порядок выполнения композиционного анализа при разработке ПО приведен на рисунке 2.

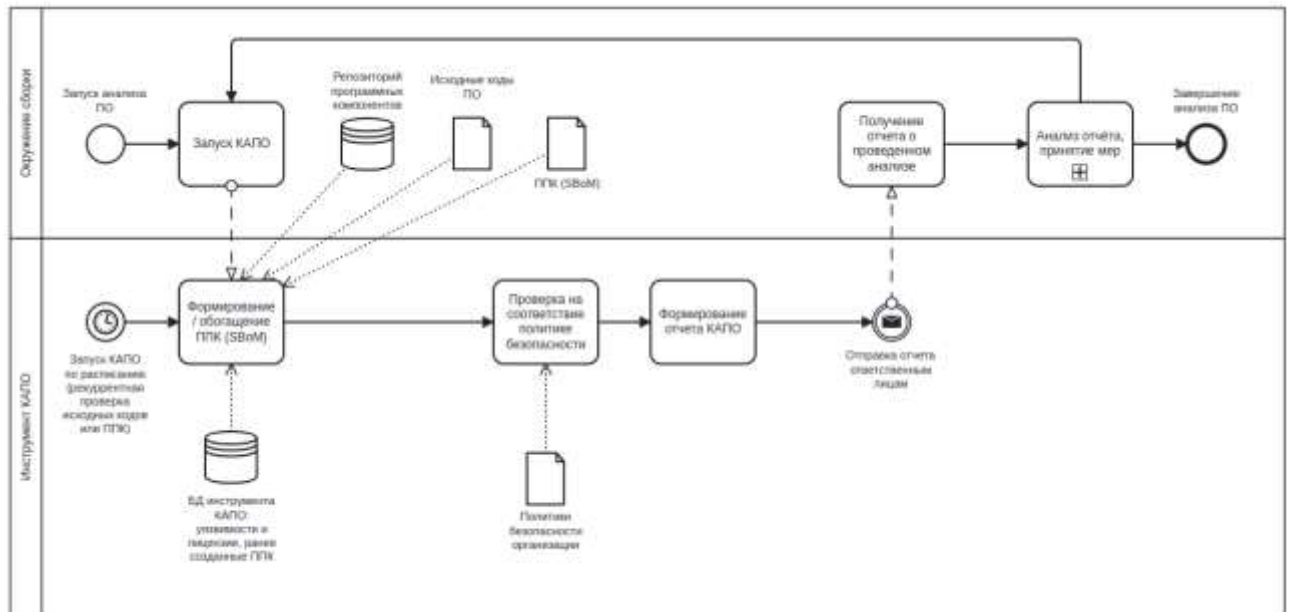


Рисунок 2 – Рекомендуемый порядок выполнения композиционного анализа разработчиком программного обеспечения

5.1.3 Выполнение композиционного анализа обеспечивают инженер автоматизации разработки ПО, инженер безопасной разработки ПО и инженер-разработчик ПО.

5.1.4 Запуск композиционного анализа осуществляется как вручную, так и автоматически, например, средствами системы непрерывной интеграции.

5.1.5 Композиционный анализ производит сканирование сборок ПО, исходных кодов ПО и связанных с ними компонентов, получаемых из контролируемых репозиториях для установления состава компонентов и их версий, используемых в ПО.

5.1.6 Результат анализа выражается в формировании нового и/или обогащении уже существующего перечня программных компонентов.

5.1.7 Разработчик должен сформировать и поддерживать в актуальном состоянии перечень программных компонентов, отражающий компонентный состав ПО.

5.1.8 Разработчик должен обеспечивать проверку результатов композиционного анализа на соответствие политикам безопасности в

отношении стороннего ПО в автоматизированном или ручном режиме и формировать отчет о результатах проведенного анализа.

5.1.9 Разработчик должен производить устранение выявленных проблем и принятие компенсирующих мер на основании отчета о результатах композиционного анализа.

5.1.10 Разработчик должен обеспечивать непрерывность выполнения процесса композиционного анализа на протяжении всего жизненного цикла разработки программного обеспечения.

5.1.11 В целях защиты цепочек поставки разработчик ПО создает и использует в разработке ПО контролируемый репозиторий артефактов с проверкой выполнения политик безопасности организации в отношении разрабатываемого ПО.

Порядок организации контролируемого репозитория артефактов приведен на рисунке 3.

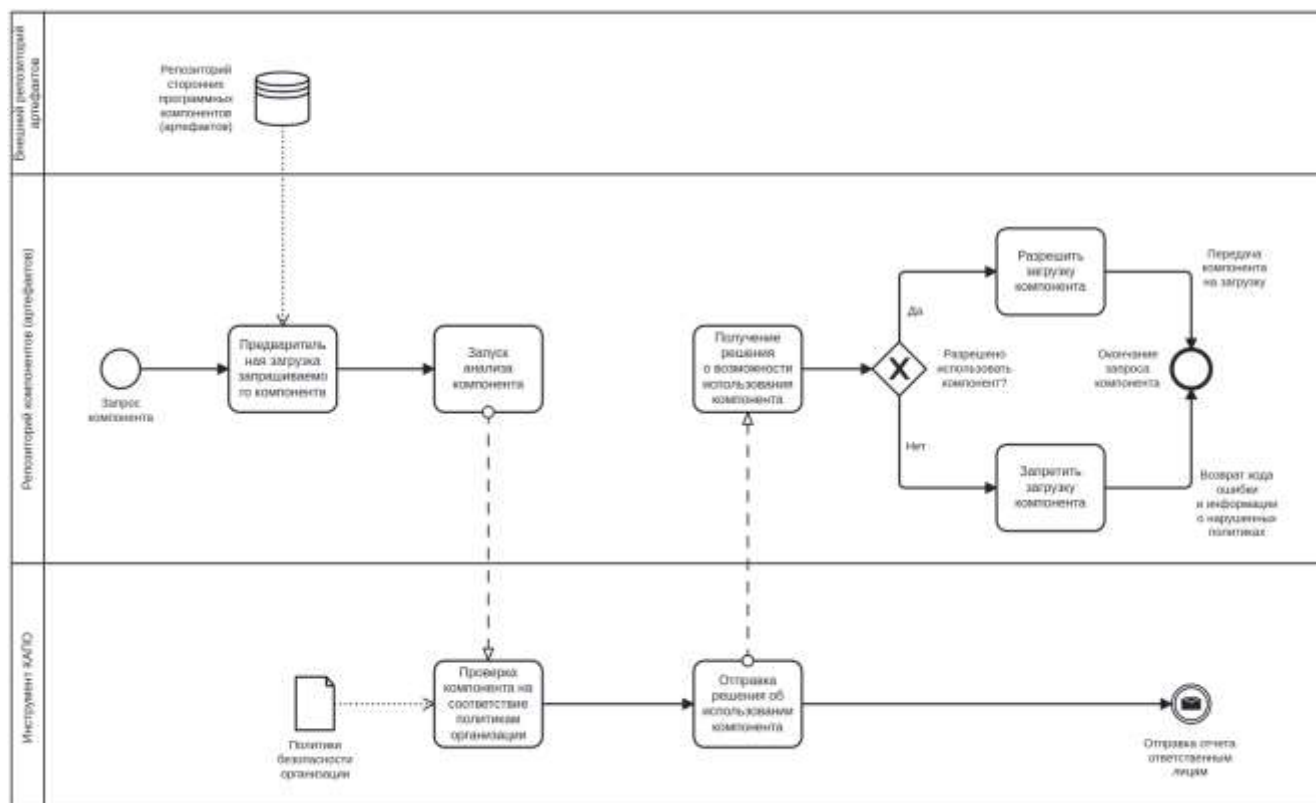


Рисунок 3 – Рекомендуемый порядок выполнения композиционного анализа в отношении репозитория компонентов (артефактов)

5.1.12 Сопровождение контролируемого репозитория артефактов должны выполнять инженеры автоматизации разработки ПО. Определение политик безопасности проверки артефактов находится в зоне ответственности инженеров безопасной разработки ПО.

5.1.13 Компоненты, размещенные в репозитории артефактов, должны проходить проверку на соответствие политикам безопасности организации перед использованием в процессах разработки или сборки ПО.

5.1.14 Выбор безопасных к применению версий артефактов ПО находится в зоне ответственности инженера-разработчика ПО.

5.2 Требования к этапу подготовки внедрения композиционного анализа программного обеспечения

5.2.1 Должно быть назначено лицо (одно или более), ответственное за организацию внедрения процессов композиционного анализа и проверки кода на предмет внедрения вредоносного программного обеспечения через цепочки поставок.

5.2.2 Разработчиком ПО должна быть произведена инвентаризация технологий (языков программирования, систем управления компонентами, пакетных экосистем, репозиториях программных компонентов, репозиториях разработки), применяемых в организации. Для определения языков программирования, на которых написано ПО, а также параметров и инструментов сборки ПО, должен быть выполнен анализ документации ПО, а также анализ исходного кода ПО.

5.2.3 Разработчик ПО должен сформировать ППК для всех разрабатываемых и поддерживаемых версий программного обеспечения.

5.2.4 В соответствии с результатами 5.2.2 и/или 5.2.3 должны быть приняты решения об организации контролируемого репозитория, включая решения об обеспечении его функционирования техническими и

программными средствами, штатом эксплуатационного персонала, решения о разработке регламента композиционного анализа ПО, включающего проверку кода на предмет внедрения вредоносного программного обеспечения через цепочки поставок.

5.2.5 Рекомендуется провести предварительную апробацию инструментов композиционного анализа на репрезентативном ПО, отражающем особенности разрабатываемого и/или сопровождаемого ПО. При оценке апробируемых инструментов композиционного анализа рекомендуется использовать требования, приведенные в разделе 6. При выборе инструмента композиционного анализа рекомендуется выбирать для внедрения тот, который отвечает не только обязательным, но и рекомендательным требованиям.

5.3 Требования к этапу внедрения композиционного анализа программного обеспечения

5.3.1 В организации должно быть назначено лицо (одно или более) ответственное за выполнение композиционного анализа. Для разного ПО могут быть назначены разные лица, отвечающие за:

- эксплуатацию контролируемого репозитория, содержащего стороннее ПО;
- формирование и регулярную актуализацию регламента композиционного анализа;
- анализ отчетов о наличии известных уязвимостей в используемых компонентах;
- отнесение используемых компонентов к компонентам, составляющим поверхность атаки;
- анализ отчетов инструмента композиционного анализа о выполненных операциях;

– определение и принятие компенсирующих и защитных мер по противодействию выявленным угрозам безопасности в цепочке поставки программного обеспечения.

5.3.2 Ответственным лицом должны быть выполнены: подготовка среды функционирования, установка инструмента композиционного анализа в среду функционирования и его интеграция со смежными системами (сборочные среды, репозитории программных компонентов и репозитории разработки).

5.3.3 Ответственным лицом должны быть выполнены: формирование перечней программных компонентов (ППК) для разрабатываемого ПО, для сформированных ППК собрана информация из баз уязвимостей и иных источников, проведена оценка (анализ) собранной информации для последующего уточнения политики безопасности.

5.3.4 В организации должна быть сформирована политика безопасности в отношении стороннего ПО. Политику рекомендуется включать в регламент композиционного анализа. Политика может быть сформулирована и включена в иные организационно-распорядительные документы организации. В политике должны быть определены:

– порядок выбора нового стороннего ПО (порядок может быть описан в рамках принципов проектирования архитектуры ПО и включен не в регламент композиционного анализа, а иные организационно-распорядительные документы организации);

– общий порядок эксплуатации контролируемого репозитория (порядок может быть описан в отдельных организационно-распорядительных документах организации);

– критерии отслеживания уязвимостей и лицензионной чистоты компонентов, участвующих в сборке ПО;

– иные критерии, учитывающие особенности использования стороннего ПО;

- критерии реагирования на результаты композиционного анализа, определяющие время реакции ответственного лица и время выполнения действий по устранению выявленных нарушений политики безопасности или разработке компенсирующих мер;

- критерии регулярной перепроверки находящегося в эксплуатации ПО на предмет наличия новых угроз безопасности в используемых компонентах, а также регулярность и периодичность такой перепроверки.

5.3.5 Разработчику ПО рекомендуется определить дополнительные правила и критерии политики безопасности:

- критерии регулярной перепроверки наличия компонентов, указанных в ранее сформированных ППК, в репозитории программных компонентов;

- правила отслеживания включений сторонних компонентов в собственный код и их перемещение в репозиторий программных компонентов или репозитории разработки, в отношении которых применяется композиционный анализ;

- правила использования инструмента, минимизирующие права ответственных лиц организации при взаимодействии с инструментом.

5.3.6 Порядок эксплуатации контролируемого репозитория должен обеспечивать невозможность использования в разработке ПО стороннего ПО, которое не прошло успешно весь комплекс проверок ПО, предусмотренных регламентами и иными распорядительными документами разработчика.

5.3.7 В отношении контролируемых репозиториях (разработки и/или артефактов) должно применяться резервное копирование всей находящейся в них информации.

5.3.8 Контролируемый репозиторий может быть создан как в виде отдельной системы, так и в составе уже существующих репозиториях, функционирующих у разработчика.

5.3.9 Для заимствованных компонентов ПО, помещенных в контролируемый репозиторий, должны выполняться композиционный и статический анализ исходного кода, должны выполняться функциональные тесты. Для привлекаемого ПО должен выполняться композиционный анализ. При размещении в контролируемом репозитории исполняемого или интерпретируемого кода, полученного из внешних источников, он должен проходить антивирусную проверку. Для стороннего ПО, размещаемого в контролируемом репозитории, могут выполняться и иные виды проверок.

5.4 Требования к этапу регулярного применения композиционного анализа программного обеспечения

5.4.1 В целях своевременного выявления уязвимостей и особенностей стороннего ПО компонентов, разработчик ПО должен регулярно выполнять композиционный анализ на протяжении всего жизненного цикла ПО.

5.4.2 Должно быть обеспечено выявление нарушений политик безопасности:

- при проверке сборки разрабатываемого ПО и формировании ППК;
- при проверке сформированного ППК;
- при анализе отдельного компонента.

5.4.3 Ответственное лицо организации должно ознакомляться с результатами композиционного анализа в течение времени, определенного регламентом композиционного анализа.

5.4.4 Ответственное лицо организации должно определять и принимать компенсирующие и/или защитные меры по противодействию выявленным угрозам безопасности в цепочке поставки сторонних компонентов при подготовке очередной версии разрабатываемого ПО.

5.4.5 Ответственное лицо организации должно регулярно проверять наличие известных уязвимостей в используемых компонентах и

формировать отчет при нарушении политики безопасности в процессе разработки или сопровождения ПО.

5.4.6 Разработчик ПО должен автоматизировать процедуру построения ППК и проведение анализа. Поверку требований политики безопасности рекомендуется автоматизировать. Этот процесс может быть настроен как средствами инструмента композиционного анализа, так и, например, с использованием системы непрерывной интеграции и тестирования ПО.

5.4.7 Разработчик ПО должен включать в процесс композиционного анализа компоненты, составляющие поверхность атаки, до сборки или при осуществлении сборки, а также на протяжении жизненного цикла ПО.

5.4.8 Разработчик ПО должен проводить композиционный анализ компонентов ПО, получаемых из репозитория программных компонентов.

5.4.9 Разработчик ПО должен сопровождать каждую выпускаемую версию ПО своим ППК.

5.4.10 Разработчику ПО рекомендуется выполнять перепроверку ранее выпущенных версий ПО на наличие новых угроз безопасности в сторонних компонентах.

5.4.11 Разработчику ПО рекомендуется отслеживать компоненты, используемые в сборочных средах, репозиториях программных компонентов и репозиториях разработки.

5.4.12 Конфигурация и настройки параметров инструмента композиционного анализа должны регулярно пересматриваться. Пересмотр рекомендуется проводить при изменении состава анализируемых языков программирования, нотаций (версий) манифестов систем управления компонентами, выходе новых версий инструмента композиционного анализа.

5.4.13 Должен быть обеспечен регулярный пересмотр политики безопасности, срок такого пересмотра должен определяться регламентом композиционного анализа, принятого в организации.

5.4.14 Разработчик ПО должен обеспечивать требования информационной безопасности в отношении результатов композиционного анализа.

6 Требования к технологиям композиционного анализа программного обеспечения

6.1 Композиционный анализ ПО как технология включает в себя:

- выявление заимствований и идентификацию компонентов в составе ПО;
- построение перечня программных компонентов (ППК);
- обогащение ППК и его актуализацию;
- оповещение о выявленных в ПО уязвимостях и нежелательных особенностях использования компонентов.

6.2 Схематично композиционный анализ ПО показан на рисунке 4.

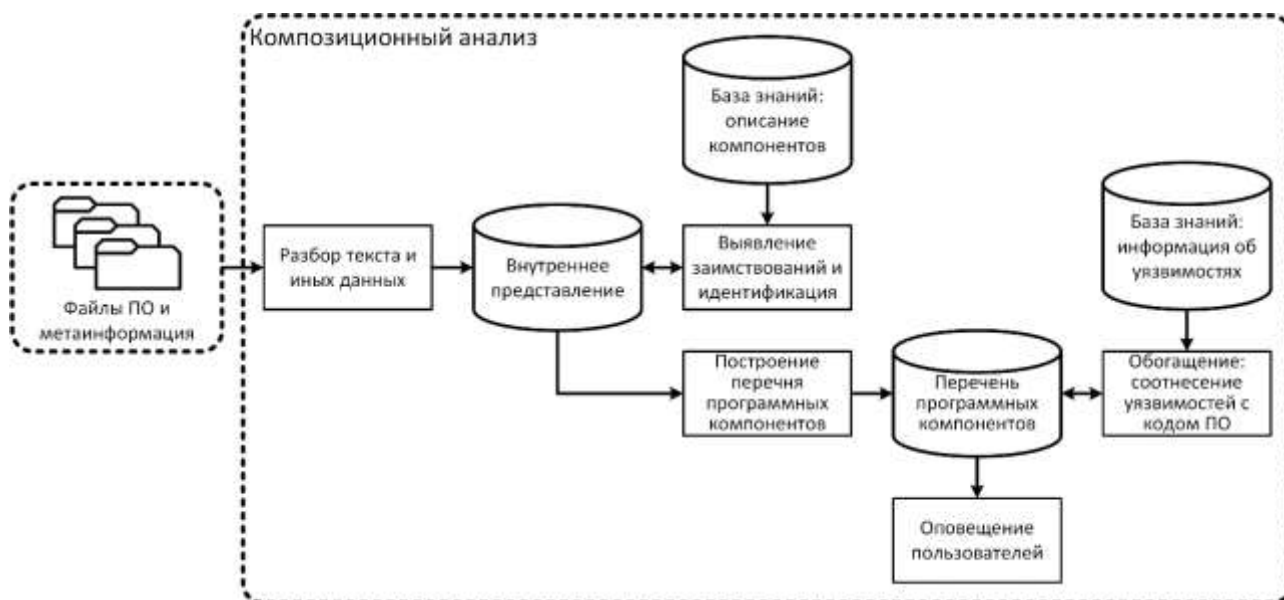


Рисунок 4 – Технология композиционного анализа программного обеспечения

6.3 Входными данными для композиционного анализа являются файлы ПО и метаданные этих файлов.

6.4 Исходный код анализируемого ПО переводится во внутреннее представление. Композиционный анализ выявляет включение в состав ПО (заимствование) на уровне:

- компонента (программного пакета) в полном составе;
- отдельных файлов заимствованного компонента;
- отдельных фрагментов кода заимствованного компонента.

6.5 Уровни перечислены в порядке возрастания технологической сложности композиционного анализа.

6.6 Композиционный анализ должен позволять обнаруживать включение в состав ПО (заимствование) на уровне компонента (программного пакета) в полном составе.

6.7 Для выявления и идентификации компонентов композиционный анализ использует базу знаний, в которой содержится описание компонентов. Описание позволяет идентифицировать компонент и его версию.

6.8 Аналогично статическому анализу, качество композиционного анализа при выявлении заимствований и идентификации состава ПО определяется тремя характеристиками: долей ложноотрицательных срабатываний, долей ложноположительных срабатываний, скоростью анализа.

6.9 Ошибочный пропуск стороннего ПО, на самом деле включенного в состав разрабатываемого ПО является ложноотрицательным срабатыванием.

6.10 Ошибочное включение в состав компонентов, неправильная идентификация версии компонента является ложноположительным срабатыванием.

6.11 По результатам выявления заимствований и идентификации состава ПО композиционный анализ строит первичный ППК.

6.12 Исходя из информации об известных уязвимостях и нежелательных особенностях использования компонентов,

композиционный анализ обогащает ППК. Информация берется из базы, агрегирующей знания о ранее выявленных уязвимостях. Содержимое базы знаний накапливается как за счет внешних источников, баз данных уязвимостей, так и путем самостоятельного выявления и анализа уязвимостей: посредством инструментального контроля кода и конфигураций ПО, экспертизы кода, аналитической работы с тематическими источниками.

6.13 Базы знаний с описанием компонентов и информацией об уязвимостях могут быть как совмещены, так и разделены. Первичный ППК может не выдаваться пользователю, допускается технологическое устройство композиционного анализа, когда первое обогащение ППК проводится сразу после его построения.

6.14 Обогащение ППК должно включать соотнесение информации об уязвимостях и нежелательных особенностях с составом ПО и его внутренней структурой. Соотнесение информации показывает возможность реализации уязвимости компонента в конкретном контексте заимствования или привлечения компонента.

6.15 Качество композиционного анализа при соотнесении информации определяется тремя характеристиками: долей ложноотрицательных срабатываний, долей ложноположительных срабатываний, скоростью анализа.

6.16 Ошибочный вывод о невозможности реализации уязвимости стороннего компонента в рамках разрабатываемого ПО, пропуск нежелательной особенности использования являются ложноотрицательным срабатыванием.

6.17 Ошибочное предположение о возможности реализации уязвимости стороннего компонента в рамках разрабатываемого ПО, ошибочное предположение о наличии нежелательной особенности использования являются ложноположительным срабатыванием.

6.18 Обогащенные ППК актуализируются – композиционный анализ повторно обращается в базу знаний, содержимое которой может обновиться.

6.19 По результатам обогащения (актуализации обогащения) ППК композиционный анализ рассылает оповещения о выявленных в ПО уязвимостях и нежелательных особенностях использования компонентов.

6.20 В процессе композиционного анализа должна выполняться идентификация стороннего ПО и его версий при сканировании сборок ПО, образов контейнеров, исходных кодов и связанных с ними артефактов, требуемых для сборки. В состав реализуемых методов идентификации должен быть включен анализ манифестов систем управления компонентами, также рекомендуется включать следующие методы анализа:

- анализ компонентов в окружении сборочной среды ПО;
- анализ компонентов в окружении среды функционирования;
- анализ с идентификацией компонентов по хэш-кодам;
- анализ частичных включений кода заимствованных компонентов в исходный код ПО;
- рекуррентный (транзитивный) анализ выявленных компонентов в целях идентификации подкомпонентов;

6.21 Композиционный анализ должен агрегировать в базе знаний информацию из публично-доступных баз данных известных уязвимостей. В качестве таковых рассматривается одна (или несколько) из следующих баз: БДУ, NVD, OSV, OSSIndex, GHSA и иные. Рекомендуется поддерживать получение сведений из БДУ.

6.22 В процессе композиционного анализа должны идентифицироваться особенности использования сторонних компонентов.

6.23 При соотнесении с составом ПО и его внутренней структурой информации об уязвимостях рекомендуется уточнять достижимость уязвимых методов в сторонних компонентах посредством анализа графа вызовов функций.

6.24 При обогащении ППК рекомендуется автоматически проверять ПО на соответствие принятым политикам безопасности.

7 Требования к инструменту композиционного анализа программного обеспечения

7.1 Инструмент должен обеспечивать формирование, экспортирование и импортирование ППК в открытом машиночитаемом формате, для возможности использования Инструмента в составе комплексов средств разработки и в системах непрерывной интеграции. В качестве таковых может рассматриваться один из следующих форматов: формат Приложения А настоящего стандарта, CycloneDX и иные.

7.2 Инструмент должен иметь механизм автоматического обогащения ППК сведениями об известных уязвимостях и лицензиях.

7.3 Инструмент должен поддерживать анализ манифестов систем управления компонентами, применяемых при разработке ПО, в целях выявления состава сторонних компонентов и определения отношений между компонентами.

Примечание – Примерами систем управления компонентами являются: Maven, Gradle, NPM, NuGet, pip, Poetry, RubyGems, Cocoapods, Packagist, Go, CONAN и иные.

7.4 Инструмент должен поддерживать анализ образов контейнеров для ПО, поставляющегося в контейнерном исполнении. Инструменту рекомендуется производить анализ всех доступных слоёв образа контейнера.

Примечание – Примерами форматов образов контейнеров являются OCI и Docker.

7.5 Инструмент должен обеспечивать формирование отчетов о наличии известных уязвимостей в анализируемых компонентах. Инструмент должен обеспечивать формирование настраиваемых отчетов о выполненных операциях: дата и время начала анализа, субъект доступа (идентификатор учетной записи пользователя или процесс Инструмента), объект доступа, содержание регистрируемой информации, длительность операции (если применимо).

7.6 Инструмент должен иметь функциональность оповещения о завершении и результатах анализа. Инструменту рекомендуется иметь функциональность оповещения по протоколу электронной почты. Инструмент может иметь функциональность оповещения по иным протоколам и иными способами.

7.7 Инструмент должен предоставлять информацию об уязвимостях, содержащую как минимум следующие сведения: публичный идентификатор уязвимости, описание, оценка критичности, при наличии таковой.

7.8 Инструмент должен поддерживать возможность анализа ППК. Инструменту рекомендуется поддерживать возможность загрузки ППК формата Приложения А настоящего стандарта, запуска и получения результатов анализа ППК, получения информации о регламентированных политиками безопасности событиях.

7.9 Инструменту рекомендуется иметь API, повторяющий функциональные возможности, доступные в интерфейсе пользователя или в режиме командной строки, для возможности встраивания в среду разработки ПО.

7.10 Инструменту рекомендуется поддерживать возможность автоматического построения ППК по результатам инвентаризации доступных репозиториях разработки.

Примечание – Примерами систем хранения репозиториях являются: GitFlic, GitVerse, Gitlab, Github, Bitbucket и иные.

7.11 Инструменту рекомендуется поддерживать анализ компонентов, хранящихся в репозиториях программных компонентов.

Примечание – В качестве репозитория программных компонентов могут рассматриваться: Sonatype Nexus Repository Manager, Harbor, Gitlab, JFrog Artifactory.

7.12 Инструменту рекомендуется обеспечивать проверку сторонних компонентов на наличие в них компрометирующих конструкций согласно используемым базам данных угроз безопасности информации.

7.13 Инструменту рекомендуется выдавать расширенную диагностику, указывая версию заимствованного компонента, свободную от выявленной уязвимости, при наличии таковой.

7.14 Инструменту рекомендуется обеспечивать формирование графа зависимостей сторонних компонентов. Инструменту рекомендуется вносить информацию сформированного графа в формируемый ППК.

7.15 Инструменту рекомендуется идентифицировать включение сторонних компонентов в исходном коде разрабатываемого ПО и учитывать выявленную информацию в формируемом ППК.

7.16 Инструменту рекомендуется поддерживать настройку и отслеживание критериев политик безопасности на уровне организации, её структурном подразделении или отдельно выбранном разрабатываемом ПО или их совокупности, а также для отдельных стадий использования сторонних компонентов.

7.17 Инструменту рекомендуется поддерживать настройку и отслеживание критериев политик безопасности по отношению к информации о выявленных уязвимостях и других атрибутах, связанных со сторонним компонентом: название, версия, автор, дата выпуска (возраст) и лицензионное соглашение, в том числе совместимость лицензии разрабатываемого ПО с лицензиями всех его компонентов.

7.18 Инструменту рекомендуется обеспечивать многопользовательский доступ пользователей инструмента к объектам

(перечень ППК, перечень выявленных компонентов, уязвимостей и лицензий) и настройкам инструмента с разграничением прав доступа.

7.19 Инструмент должен обеспечивать регистрацию выполняемых операций. Регистрация должна осуществляться для всех выполняемых операций, необходимых для диагностики работы Инструмента, включая: проведение анализа сторонних компонентов с отметкой о совершении операции, её длительности и результате выполнения, а также идентификатора пользователя или системы, запустившей анализ. События получения и изменения доступа к Инструменту, настройки политик безопасности, применяемых баз данных уязвимостей рекомендуется хранить в журнале регистрации событий.

7.20 Инструмент может обеспечивать выявление и исключение дублирующей информации об уязвимости при работе с несколькими базами данных уязвимостей.

7.21 Инструмент может обеспечивать возможность оставлять комментарии к выявленным уязвимостям с последующим занесением этой информации в ППК. Инструмент может обеспечивать возможность переноса комментариев от предыдущего ППК к следующему.

7.22 Инструмент может обеспечивать возможность ведения в организации собственных баз данных уязвимостей.

7.23 Инструмент может формировать отчеты в формате динамических веб-страниц, допускающих фильтрацию, сортировку и экспорт в формат текстового документа.

7.24 Инструмент может предоставлять в отношении сторонних компонентов дополнительные сведения о популярности, актуальности и соответствии принципам открытого ПО, а также обеспечивать дополнительную информацию об особенностях применения компонентов.

7.25 Инструмент может обеспечивать автоматическую проверку достижимости уязвимых методов в сторонних компонентах.

8 Требования к перечню программных компонентов

8.1 ППК может быть сформирован в соответствии со следующими форматами: формат Приложения А, CycloneDX и иными, отвечающими требованиям настоящего стандарта.

8.2 ППК должен содержать как минимум информацию о программных компонентах, а также информацию о параметрах его формирования, приведенную в таблице 1.

Т а б л и ц а 1 – Минимальное содержимое перечня программных компонентов

Обозначение поля	Описание поля
Автор компонента	Наименование организации, группы лиц или отдельных лиц, позволяющее идентифицировать автора компонента
Наименование компонента	Название компонента
Версия компонента	Версия компонента
Идентификатор компонента	Идентификатор компонента, позволяющий точно идентифицировать компонент в базах уязвимостей. В качестве идентификатора может применяться PURL или иной
Автор ППК	Идентификатор, позволяющий определить организацию, роль или конкретного исполнителя в соответствии с регламентом организации отвечающего за формирование ППК
Дата и время формирования ППК	Дата и время формирования перечня программных компонентов

8.3 Для каждой версии анализ манифестов систем управления компонентами программного обеспечения должен быть сформирован отдельный ППК.

8.4 Операции экспорта и импорта ППК рекомендуется выполнять в соответствии с форматом Приложения А.

8.5 В ППК рекомендуется включать информацию о принадлежности компонентов к поверхности атаки.

8.6 В ППК рекомендуется включать информацию о взаимозависимости заимствованных компонентов.

8.7 В ППК рекомендуется включать информацию об известных уязвимостях для перечисляемых компонентов. Информация об уязвимостях должна включать в себя следующие поля данных: идентификатор в первичной базе данных уязвимостей, идентификатор уязвимости при наличии в иных базах уязвимостей, название, описание, тип ошибки, идентификатор ошибки, дата выявления, базовый вектор уязвимости (возможные стандарты описаний CVSS 2.0, CVSS 3.0, CVSS 3.1, CVSS 4.0), уровень опасности уязвимости. Рекомендуется, чтобы информация об уязвимостях содержала отметку о наличии опубликованного способа эксплуатации уязвимости, стратегии устранения или компенсации риска.

Примечания

1 В случае, если уровень опасности уязвимости не определен, он должен быть установлен ответственным лицом (или Инструментом автоматически) в соответствии с регламентом, принятым в организации.

2 Первичной базой данных уязвимостей считается такая база, которая определена основным источником знаний об уязвимостях в организации.

3 Базовый вектор уязвимости описывает критерии опасности уязвимости в соответствии с Common Vulnerability Scoring System³. На основании базового вектора уязвимости рассчитывается уровень опасности уязвимости.

8.8 В ППК рекомендуется включать информацию о лицензионных соглашениях для перечисленных в нём компонентов. Информация о лицензиях должна включать идентификатор лицензии (согласно

³ <https://bdu.fstec.ru/documents/26>

кодификатору SPDX⁴ или иному публично доступному индексу лицензионных соглашений).

8.9 ППК рекомендуется подписывать цифровой подписью организации, выпускающей программное обеспечение.

8.10 В ППК должны быть включены хэш-коды перечисляемых компонентов.

8.11 В ППК рекомендуется включать отметку о выявлении компрометирующих конструкций в компонентах.

8.12 ППК может содержать метаданные перечисляемых компонентов: тип компонента, краткое описание компонента, идентификатор компонента в формате PURL, текст лицензионного соглашения, перечень языков программирования.

8.13 Если в состав ПО включены модели машинного обучения (одна или несколько), в ППК должна быть включена следующая информация: название и версия модели, тип модели (вид обрабатываемых данных), лицензионные соглашения, ассоциированные с компонентами или данными, лежащими в основе модели, привлекаемые компоненты. Разработчик должен формировать описание модели машинного обучения согласно составу сведений, приведенных в Приложении Б.

⁴ <https://spdx.org/licenses/>

Приложение А

(обязательное)

Формат представления перечня программных компонентов

Перечень программных компонентов в машиночитаемом формате представляется в соответствии со спецификацией CycloneDX⁵ в нотации JSON по [6] в виде объекта, содержащего поля описанные в Таблице А.1. Допускается наличие дополнительных полей, соответствующих спецификации CycloneDX.

Т а б л и ц а А.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 версию спецификации CycloneDX, в соответствии с которой сформирован документ	Обязательно
serialNumber	Строка	Уникальный серийный номер документа, сгенерированный в соответствии с RFC-4122 [7]	Опционально
version	Целое число	Версия документа. Первоначальное значение — 1. При модификациях должна увеличиваться на 1	Обязательно
metadata	Объект	Дополнительная информация о перечне и ПО. Требования к заполнению полей объекта описаны в таблице А.2	Обязательно
components	Массив объектов	Список компонентов. Если в ПО используется несколько версий одного и того же компонента, то для каждой версии компонента должна быть отдельная запись. 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 А.5	Обязательно при наличии в составе ПО компонентов

⁵ Спецификация CycloneDX <https://ecma-international.org/publications-and-standards/standards/ecma-424>

Окончание таблицы А.1

Имя поля	Тип	Описание	Требования к наличию
annotations	Массив объектов	Произвольные комментарии, касающиеся компонентов, сервисов, уязвимостей или самого документа. Требования к заполнению полей объектов описаны в таблице А.4	Опционально

Таблица А.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)	Обязательно
component	Объект	Информация о ПО. Требования к заполнению полей объекта описаны в таблице А.3	Обязательно

Таблица А.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	Обязательно
name	Строка	Название ПО	Обязательно
version	Строка	Версия ПО. Длина строки не должна превышать 1024 символа	Обязательно
manufacturer	Объект	Информация о производителе ПО. Требования к заполнению полей объекта описаны в таблице А.4	Обязательно

Т а б л и ц а А.4 – Требования к полям объекта, описывающего информацию о производителе программного обеспечения

Имя поля	Тип	Описание	Требования к наличию
name	Строка	Название организации — производителя ПО	Обязательно

Т а б л и ц а А.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 ⁶	Опционально
externalReferences	Массив объектов	Массив объектов, описывающих источник получения компонента, его исходных кодов, документации и т. д. Требования к заполнению полей объектов описаны в таблице А.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)	Обязательно

⁶ Спецификация формата <https://github.com/package-url/purl-spec>

Окончание таблицы А.5

Имя поля	Тип	Описание	Требования к наличию
properties	Массив объектов	Массив объектов, описывающих дополнительные свойства компонента. Требования к заполнению полей объектов описаны в таблице А.8. Среди элементов массива обязательно должен присутствовать объект GOST:attack_surface	Обязательно
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 А.5.	Опционально

Т а б л и ц а А.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 CycloneDX	Обязательно
url	Строка	Если поле type имеет значение vcs, то в поле url помещается унифицированный указатель ресурса URL для репозитория с исходными кодами компонента. Если поле type имеет значение source-distribution, то в поле url помещается унифицированный указатель ресурса URL для архива с исходным кодом компонента	Обязательно
hashes	Массив объектов	Содержит список хэш-кодов содержимого внешнего ресурса, полученных при помощи различных алгоритмов. Требования к заполнению полей объекта описаны в таблице А.7. Обязательно наличие хэш-кодов, вычисленных с применением алгоритма STREEBOG-256 или STREEBOG-512, по ГОСТ Р 34.11-2012	Опционально

Т а б л и ц а А.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 компонента. Должно содержать одно из значений, определенных спецификацией CycloneDX	Обязательно
content	Строка	Содержит значение хэш-кода компонента	Обязательно

Т а б л и ц а А.8 – Требования к полям объекта, описывающего дополнительные свойства компонента

Имя поля	Тип	Описание	Требования к наличию
name	Строка	<p>Название свойства компонента.</p> <p>Значение GOST:attack_surface описывает свойство принадлежности компонента к поверхности атаки.</p> <p>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.</p>	Обязательно
value	Строка	<p>Значение свойства компонента.</p> <p>Если поле name имеет значение GOST:attack_surface, то поле value должно содержать одно из трёх значений:</p> <ol style="list-style-type: none"> 1. yes — компонент входит в непосредственную поверхность атаки; 2. indirect — компонент входит в косвенную поверхность атаки; 3. no — в иных случаях. <p>Если поле name имеет значение GOST:source_langs, то поле value должно содержать перечисление из возможных 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 или иных.</p> <p>В случае отсутствия в глоссарии используемого языка программирования указывается его каноническое название</p>	Обязательно

Приложение Б
(обязательное)

Состав данных описания модели машинного обучения

Т а б л и ц а Б.1 – Общее описание модели

Название	Название модели
Версия	Версия модели
Тип	Тип модели, например «генерация текста» или «обработка изображений»
Разработчик	Наименование организации, группы лиц или отдельных лиц, позволяющее идентифицировать автора модели
Лицензия(ии)	Лицензионные соглашения, ассоциированные с использованием модели, например, Apache 2.0, MIT или иные
Привлекаемые компоненты	Фреймворки машинного обучения, использованные для создания модели, например PyTorch, Tensorflow или иные
Источник	Адрес в сети, где можно получить модель
Ссылки	Дополнительная информация о модели, ссылки на документацию, статьи, сайты, листы рассылки или иные
Подпись(и)	Цифровая подпись описания модели, обеспечивающая подлинность и целостность информации

Т а б л и ц а Б.2 – Архитектура модели

Наборы данных	Названия, версии и ссылки на наборы данных, использованные для тренировки модели
Базовая модель	Название, версия и источник получения базовой модели, на которой основана описываемая модель. При наличии
Архитектура модели	Название публичной архитектуры модели, например, «BLOOM». Если применимо
Семейство архитектур	Название семейства архитектур, например, «GPT». Если применимо
Аппаратное обеспечение	Информация об аппаратном обеспечении, которое использовалось для тренировки модели и/или может использоваться для её исполнения
Программное обеспечение	Информация о программном обеспечении, которое использовалось для тренировки модели или может использоваться для её исполнения

Окончание таблицы Б.2

Наборы данных	Названия, версии и ссылки на наборы данных, использованные для тренировки модели
Тип данных входа	Тип данных, которые получает модель на вход, например изображения или текст
Тип данных выхода	Тип данных, которые формируют модель на выход, например изображения или текст
Требования к дополнительному ПО	Перечень привлекаемых компонентов, необходимых для запуска модели

Т а б л и ц а Б.3 – Использование модели

Назначение модели	Описание того, как модель должна использоваться
Сценарии вне основной сферы применения	Сценарии использования модели вне определенной её разработчиками сферы применения
Неправильное или вредоносное использование	Сценарии ненадлежащего или вредоносного использования модели

Библиография

- | | | |
|-----|-----------------------------|---|
| [1] | ГОСТ Р 56545-2015 | Защита информации. Уязвимости информационных систем. Правила описания уязвимостей |
| [2] | ГОСТ Р 58412-2019 | Защита информации. Разработка безопасного программного обеспечения. Угрозы безопасности информации при разработке программного обеспечения |
| [3] | ГОСТ 19.101-77 | Единая система программной документации. Виды программ и программных документов |
| [4] | ГОСТ Р ИСО/МЭК 15408-1-2012 | Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель |
| [5] | ГОСТ Р ИСО/МЭК 12207-2010 | Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств |
| [6] | ISO/IEC 21778:2017 | Information technology – The JSON data interchange syntax (Информационная технология. Синтаксис обмена данными в формате JSON) |
| [7] | RFC 4122 | A Universally Unique IDentifier (UUID) URN Namespace (Пространство имен URN с Всемирным универсальным идентификатором (UUID)) |

УДК 004:006.86

ОКС 35.030 : 35.080

Ключевые слова: защита информации, программное обеспечение, заимствованные и привлекаемые компоненты программного обеспечения, композиционный анализ программного обеспечения
