МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЕДИНАЯ ЦИФРОВАЯ ПЛАТФОРМА РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ» ДЛЯ СОЗДАНИЯ, РАЗВИТИЯ И ЭКСПЛУАТАЦИИ ГОСУДАРСТВЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТОВ ЕДИНОЙ ЦИФРОВОЙ ПЛАТФОРМЫ РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ»

версия 1.0

2022 г.

Содержание

1. Ссылки, термины и определения

В настоящих методических рекомендациях использованы ссылки на следующие нормативно-правовые акты и стандарты:

Таблица 1. Правовые акты и стандарты

Обозначение документа	Наименование документа
ГОСТ Р 51275-2006	Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения;
ГОСТ Р 56939-2016	Защита информации. Разработка безопасного ПО. Общие требования;
ГОСТ Р 58412 -2019	Защита информации. Разработка безопасного ПО. Угрозы безопасности информации при разработке ПО;
ГОСТ 19.201-78	Единая система программной документации. Техническое задание. Требования к содержанию и оформлению;
ГОСТ 19.101-77	Единая система программной документации (ЕСПД). Виды программ и программных документов;
ГОСТ Р ИСО/МЭК 27001-2006	Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента ИБ. Требования;
ГОСТ Р ИСО/МЭК 12207-2010	Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств;
ГОСТ Р ИСО/МЭК 15408-3-2013	Информационная технология (ИТ). Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности;
ГОСТ Р ИСО/МЭК 27002-2021	Информационные технологии (ИТ). Методы и средства обеспечения безопасности. Свод норм и правил применения мер обеспечения информационной безопасности;
Приказ ФСТЭК России №17	Приказ ФСТЭК России от 11 февраля 2013 г. № 17 «Об утверждении

Обозначение документа	Наименование документа
	Требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах»;
Приказ ФСТЭК России №21	Приказ ФСТЭК России от 18 февраля 2013 г. № 21 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных»;
Приказ ФСТЭК России №239	Приказ ФСТЭК России от 25 декабря 2017 г. № 239 «Об утверждении Требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры Российской Федерации»;
Приказ ФСТЭК России №76	Приказ ФСТЭК России от 2 июня 2020 г. № 76 «Об утверждении Требований по безопасности информации, устанавливающих уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий»;
Методический документ ФСТЭК России от 11 февраля 2014 г.	Методический документ «Меры защиты информации в государственных информационных системах», утвержденный ФСТЭК России 11 февраля 2014 г.

В настоящих методических рекомендациях применены следующие термины и сокращения с соответствующими определениями:

Таблица 2. Термины, сокращения и определения

Термин, сокращение	Определение
Безопасное ПО (БПО)	ПО, разработанное с использованием совокупности мер, направленных на предотвращение появления и устранение уязвимостей программы;
ГИС на Платформе «ГосТех»	Государственные информационные системы, создаваемые, развиваемые,

Термин, сокращение	Определение
	эксплуатируемые с использованием программно-аппаратной среды, цифровых продуктов, включенных в каталог цифровых продуктов Платформы "ГосТех", а также инструментов, информационных технологий Платформы «ГосТех»;
Динамический анализ кода программы	Вид работ по инструментальному исследованию программы, основанный на анализе кода программы в режиме непосредственного исполнения (функционирования) кода;
Жизненный цикл	Развитие системы, продукта, услуги, проекта или других изготовленных человеком объектов, начиная со стадии разработки концепции и заканчивая прекращением применения;
Заказчик	Государственные органы, иные организации, уполномоченные на осуществление мероприятий по созданию, развитию, эксплуатации государственных информационных систем в соответствии с нормативными правовыми актами Российской Федерации, обеспечивающие создание, развитие, эксплуатацию государственных информационных систем на платформе «ГосТех» и (или) использование цифровых продуктов платформы «ГосТех», органы местного самоуправления, государственные (муниципальные) предприятия и учреждения, государственные корпорации, государственные компании, публично-правовые компании, а также хозяйственные общества, более пятидесяти процентов акций (долей) в уставном капитале которых находится в государственной (муниципальной) собственности;

Термин, сокращение	Определение
Информационная система	Совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств;
Квалификационное тестирование	Тестирование, проводимое разработчиком и санкционированное приобретающей стороной (при необходимости) с целью демонстрации того, что программный продукт удовлетворяет спецификациям и готов для применения в заданном окружении или интеграции с системой, для которой он предназначен;
Комплексирование ПО	Создание ПО и программ из множества программных элементов, соответствующих определенным программным требованиям, архитектуре и проекту;
Критическая ошибка в программе	Ошибка, срабатывание которой может привести к нарушению безопасности обрабатываемой информации: конфиденциальности, целостности, доступности;
Недостаток программы	Любая ошибка, допущенная в ходе проектирования или реализации программы, которая в случае ее не исправления может являться причиной уязвимости программы;
Ошибка в программе	Конструкция или набор конструкций в исходном коде программы, наличие которой указывает на возможность нарушения безопасности обрабатываемой информации и/или на ошибку реализованного в программе алгоритма;
Платформа «ГосТех»	Цифровая экосистема создания, развития и эксплуатации государственных информационных систем, включающая в себя единую программно-аппаратную среду, цифровые продукты, информацию, информационные технологии,

Термин, сокращение	Определение
	государственные информационные системы, необходимые для реализации функций платформы «ГосТех», а также совокупность нормативных правовых, организационных, методологических правил и процедур, обеспечивающих деятельность участников отношений, возникающих в связи с созданием и функционированием платформы «ГосТех»;
Продуктивная среда функционирования ПО	Интегрированная система, включающая в себя аппаратные средства, ПО, программно-аппаратные средства, процедуры и документы, необходимые для корректного функционирования ПО в режиме промышленной эксплуатации;
Проект архитектуры программы	Логическая структура программы, в которой идентифицированы компоненты, их интерфейсы и концепция взаимодействия между ними;
Разработчик ПО	Лицо, осуществляющее выполнение работ, оказание услуг по разработке ПО в составе ГИС на Платформе «ГосТех», в рамках заключенного государственного контракта (договора, соглашения) на создание, развитие, эксплуатацию ГИС на Платформе «ГосТех»;
Санитайзер	Средство анализа и контроля входных данных позволяющее предотвратить внедрение вредоносного кода в ПО;
Сборка ПО	Процесс построения средствами систем программирования исполнимых программных модулей ПО (для компилируемых языков – с объектным кодом, для интерпретируемых языков – с машинно-независимым кодом низкого уровня, например, байт-кодом) из исходных модулей;

Термин, сокращение	Определение
Система управления конфигурацией ПО	Интегрированная система, включающая в себя аппаратные средства, ПО, программно-аппаратные средства, процедуры и документы, необходимые для разработки ПО;
Статический анализ исходного кода программы	Вид работ по инструментальному исследованию программы, основанный на анализе исходных текстов программы с использованием специализированных инструментальных средств (статических анализаторов) в режиме, не предусматривающем реального выполнения кода, и выполняемый для определения свойств программы. В частности, статический анализ может определять потенциальные места ошибок в программе;
Тестирование на проникновение	Вид работ по выявлению (подтверждению) уязвимостей программы, основанный на моделировании (имитации) действий потенциального нарушителя;
Фаззинг-тестирование программы	Вид работ по исследованию программы, направленный на оценку ее свойств и основанный на передаче программе случайных или специально сформированных входных данных, отличных от данных, предусмотренных алгоритмом работы программы.

2. Общие положения

Методические рекомендации по обеспечению безопасности при разработке ПО с использованием компонентов единой цифровой платформы Российской Федерации «ГосТех» (далее – Методические рекомендации) разработаны в целях реализации положений «Концепции создания и функционирования единой цифровой платформы Российской Федерации «ГосТех».

Методические рекомендации направлены на обеспечение необходимого уровня защиты информации, обрабатываемой ГИС на Платформе «ГосТех», посредством реализации мер, минимизирующих количество уязвимостей и недостатков программ на всех этапах жизненного цикла ПО.

Настоящий документ содержит рекомендации по реализации требований по разработке БПО на Платформе «ГосТех», установленных ГОСТ Р 56939.

При выявлении подтвержденных уязвимостей и недостатков программ критического и высокого уровня влияния во вновь создаваемом ПО при проведении его тестирования инструментальными средствами запрещается перевод такого ПО в продуктивную среду функционирования ГИС на Платформе «ГосТех».

На всех этапах жизненного цикла ПО должна быть обеспечена конфиденциальность информации о проекте архитектуры программы, исходном коде, средствах и результатах проводимых инструментальных тестирований ПО и его компонентов. В случаях привлечения к проведению инструментальных тестирований ПО экспертных организаций, с такими организациями должны быть заключены соответствующие соглашения о неразглашении определяющие требования по обеспечению конфиденциальности, контроль и ответственность за их невыполнение, либо обязательства о неразглашении должны быть включены в соответствующий государственный контракт (договор, соглашение), заключаемый с экспертной организацией.

Использование заимствованных компонентов (библиотек, фреймворков, интерпретируемых скриптов и др.) сторонних разработчиков допускается только при наличии текстов исходного кода данных компонентов, кроме компонентов, включённых в состав Платформы «ГосТех» в соответствии с настоящим документом и «Методическими рекомендации по включению сервисов в единую цифровую платформу Российской федерации «ГосТех».

Графическое описание организации рекомендуемого процесса разработки БПО на Платформе «ГосТех» приведено в приложении № 1.

3. Область применения

Методические рекомендации предназначены для использования Заказчиками при разработке технических заданий, на создание, развитие, эксплуатацию ГИС на Платформе «ГосТех» и Разработчиками ПО, выполняющими работы по созданию, развитию и эксплуатации ПО ГИС на Платформе «ГосТех» в рамках заключенных государственных контрактов (договоров, соглашений).

Данные методические рекомендации следует применять совместно с ГОСТ Р 56939 и ГОСТ Р 58412 для распределение ролей и обязанностей, связанных с реализацией мер по разработке безопасного ПО, между работниками Разработчика ПО, а также для идентификации и оценки угроз безопасности информации, которые могут возникать при разработке ПО.

Методические рекомендации не применяются при разработке ПО средств защиты информации и средств криптографической защиты информации. Такого типа ПО должно разрабатываться в строгом соответствии с требованиями законодательства Российской Федерации.

4. Анализ и формирование требований к разрабатываемому ПО

4.1. Разработка требований по безопасности

В целях предотвращения появления уязвимостей программы Заказчиком должны быть сформулированы требования по безопасности, предъявляемые к разрабатываемому ПО.

В качестве источников для формирования требований Заказчик может использовать требования законодательства Российской Федерации, правовых актов федеральных органов исполнительной власти, национальных и отраслевых стандартов, перечень требований Заказчика, предполагаемые сценарии применения ПО.

Для формирования требований Заказчику необходимо собрать информацию об области применения разрабатываемого ПО, составе планируемой к обработке информации, факте отнесения обрабатываемой информации в ГИС к категории информации ограниченного доступа и классе защищенности ГИС.

На основании полученных данных Заказчику необходимо выполнить:

- предварительный анализ потенциальных угроз безопасности информации ПО в соответствии с рекомендациями пункта 4.2 данного документа;
- идентифицировать и сформулировать цели защиты информации, достижение которых должно обеспечиваться разрабатываемым ПО;
- сформулировать и задокументировать требования по безопасности в форме функциональных и нефункциональных требований[1].

Требования по безопасности в зависимости от их назначения рекомендуется описывать в подразделах «Требования к надежности» и/или «Требования к функциональным характеристикам» раздела «Требования к программе или программному изделию» технического задания, разрабатываемого в соответствии с ГОСТ 19.201 или в отдельном разделе технического задания на создание или развитие ГИС на Платформе «ГосТех».

Техническое задание на создание, развитие ГИС на ЕЦП «ГосТех» должно быть согласовано Заказчиком с федеральным органом исполнительной власти в области обеспечения безопасности и федеральным органом исполнительной власти, уполномоченным в области противодействия техническим разведкам и технической защиты информации, в пределах их полномочий в части требований о защите информации, в соответствии с Постановление Правительства РФ от 6 июля 2015 г. № 676 «О требованиях к порядку создания, развития, ввода в эксплуатацию, эксплуатации и вывода из эксплуатации государственных информационных систем и дальнейшего хранения содержащейся в их базах данных информации».

1 Функциональные требования по безопасности: описывают действия, которые должно выполнять ПО в целях нейтрализации угроз безопасности информации, а нефункциональные требования описывают свойства и параметры ПО, имеющие отношение к нейтрализации угроз безопасности информации.

Пример требований по безопасности к разрабатываемому ПО:

- все поступающие в ПО входные данные должны подлежать контролю целостности, полноты и аутентичности;

- в ПО должна быть реализована обработка исключительных ситуаций, возникающих в процессе функционирования программы;

- информационные сообщения пользователю ГИС на Платформе «ГосТех» не должны содержать информацию о наименовании программных технических средств, использующихся в среде функционирования ПО и их версиях.

Рекомендованные требования по безопасности к разрабатываемому ПО в составе ГИС на Платформе «ГосТех» для различных классов защищенности, в соответствии с приказом ФСТЭК России от 11 февраля 2013 г. № 17 «Об утверждении Требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах», приведены в Приложении № 2.

4.2. Моделирование угроз безопасности информации

В целях определения актуальных угроз безопасности информации, а также в целях формирования исходных данных Разработчику ПО необходимо разработать и задокументировать проект архитектуры программы и провести моделирование угроз безопасности информации.

Проектирование архитектуры должно удовлетворять следующим принципам:

1) разумной простоты: ПО и его элементы, должны быть возможно более простыми (иметь минимально возможный размер исходного кода), чтобы уменьшить вероятность возникновения уязвимостей и упростить проверки безопасности;

2) безопасных умолчаний: доступ к информации, информационное и (или) управляющее взаимодействие в отсутствие разрешения на эту операцию должны быть запрещены;

3) полноты перекрытия: реализация механизма обеспечения безопасности должна иметь сквозной характер (охватывать все структурные элементы ПО);

4) разделения привилегий: механизм обеспечения безопасности должен позволять выполнять доступ к информации или выполнение иной операции на основе двух и более независимых факторов, где это возможно;

5) наименьших привилегий: каждый процесс должен обладать минимально необходимыми привилегиями для выполнения установленных для него задач;

6) минимизации зависимостей: количество элементов ПО, которые используются несколькими пользователями или субъектами с различными привилегиями, должно быть минимально;

7) эшелонированной защиты: наличие множественной защиты, в частности — в виде уровней, с целью предотвращения или сдерживания угроз безопасности информации;

8) трудозатрат: стоимость защиты не должна превосходить стоимость потенциальных потерь в случае успешного внешнего воздействия на ПО;

9) компромиссного механизма обнаружения: использование (где это позволяют требования безопасности информации) недорогого механизма надежного своевременного обнаружения факта внешнего воздействия на ПО взамен дорогостоящего механизма обеспечения безопасности, который способен предотвратить это воздействие.

Моделирование угроз безопасности информации должно быть проведено Разработчиком ПО на этапе выполнения проектирования архитектуры программы жизненного цикла ПО.

По результатам моделирования угроз Разработчику ПО необходимо скорректировать (уточнить) требования по безопасности, предъявляемые к разрабатываемому ПО и проект архитектуры программы.

Процедура моделирования угроз должна быть проведена для выявления потенциальных угроз безопасности информации, обоснования требований по безопасности, предъявляемых к разрабатываемому ПО в составе ГИС на Платформе «ГосТех», и выработки возможных мер по нейтрализации актуальных угроз безопасности информации.

В процессе моделирования должны быть рассмотрены угрозы безопасности информации, которые могут возникать в процессах разработки и использования ПО в составе ГИС на Платформе «ГосТех».

Для формирования перечня и описания угроз безопасности информации, которые могут возникать в процессах разработки ПО, следует использовать ГОСТ Р 58412 «Защита информации. Разработка безопасного ПО. Угрозы безопасности информации при разработке ПО».

Для формирования перечня и описания угроз безопасности информации, которые могут возникать при использовании ПО в составе ГИС на Платформе «ГосТех», Разработчику ПО необходимо использовать:

- сведения о предполагаемых компонентах программы;
- сведения об интерфейсах компонентов программ;
- сведения о предполагаемых алгоритмах, методах и протоколах взаимодействия компонентов программы;
- сведения о вариантах и методах конфигурирования программы, её компонентов, а также планируемой среды функционирования ПО;
-информацию о заимствованных компонентах сторонних разработчиков;
- информацию из банка данных угроз безопасности информации ФСТЭК России (https://bdu.fstec.ru).

Также допускается использовать любую иную информацию, в том числе из открытых источников, связанную с типовыми сценариями компьютерных атак и угрозами безопасности информации.

Моделирование угроз рекомендуется проводить в рамках следующих этапов:

- структурный анализ потоков данных;
- перечисление угроз безопасности информации и оценка их актуальности;
- разработка мер для снижения рисков реализации актуальных угроз.

Реализация разработанных мер и проверка их эффективности должна быть проверена на этапах конструирования и комплексирования ПО и выполнения квалификационного тестирования ПО соответственно.

На этапе структурного анализа данных Разработчику ПО необходимо построить диаграмму потоков данных и проанализировать ее на наличие актуальных угроз, связанных с нарушением конфиденциальности, целостности и доступности обрабатываемой информации, например угроз, связанных с подменой данных, несанкционированным доступом, раскрытием информации, нарушением функционирования программы, исполнением произвольного кода и др.

Диаграмма потоков данных должна содержать и описывать следующие компоненты программы:

- внешние элементы;
- внутренние элементы;
- процессы;
- хранилище данных;
- потоки данных.

Внешние элементы – объекты (источники и потребители информации), являющиеся внешними по отношению к разрабатываемому ПО. К внешним источникам, например, можно отнести пользователей программы, программы и информационные системы, с которыми осуществляется обмен информацией.

Внутренние элементы – объекты (источники и потребители информации), входящие в состав разрабатываемого ПО как отдельные объекты.

Процессы – любые процессы, реализуемые в программе и осуществляющие обработку информации (преобразование входных данных в выходные).

Примером могут служить процедуры выполнения каких-либо расчетов и сортировки данных.

На диаграмме потоков данных каждый процесс должен быть пронумерован в формате a.b, где, a – номер группы процессов, реализующих одну бизнес-функцию в программе, b – номер процесса, идентифицирующий его в группе процессов. В случае если процесс самостоятельно реализует бизнес-функцию, то для такого процесса допускается нумерация вида a.0. Нумерация процессов осуществляется Разработчиком ПО для последующей реализации возможности прослеживания исходного кода программы к проекту архитектуры программы на этапе конструирования и комплексирования ПО.

Хранилище данных – файлы, репозитории, базы данных, таблицы в базах данных и иные сущности, осуществляющие хранение/временное хранение информации для ее последующего использования внешними элементами или процессами разрабатываемой программы.

Потоки данных – направления получения и передачи информации между внешними элементами, процессами и хранилищами данных. При описании потоков данных Разработчику ПО необходимо также указывать краткое описание входных и выходных данных, например «идентификатор пользователя», «результаты расчета», «бизнес-данные». Для повышения

качества моделирования угроз, Разработчику ПО рекомендуется выделять на диаграмме потоки передачи информации ограниченного доступа и критических бизнес-данных, влияющих на процесс функционирования программ и ГИС на Платформе «ГосТех».

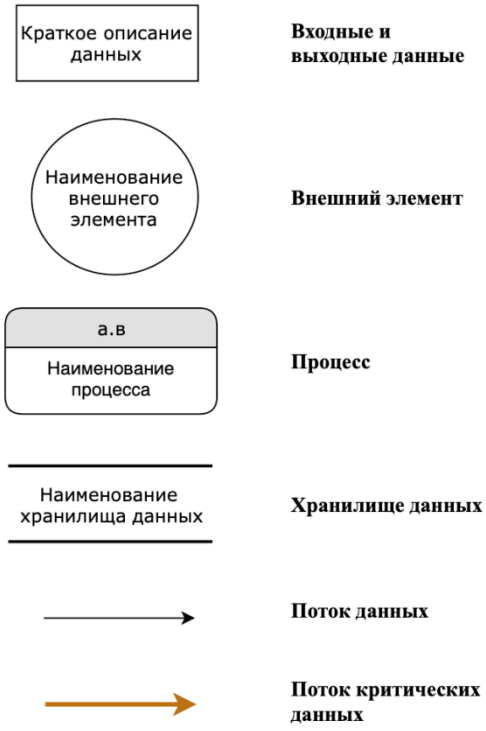


Рисунок 1. Графическое изображение элементов программы для диаграммы потоков данных

Рекомендуемое графическое изображение на диаграмме потоков данных компонентов программы представлено на рисунке 1. Пример диаграммы потоков данных программы, составленный с учетом рекомендации настоящего документа представлен на рисунке 2.

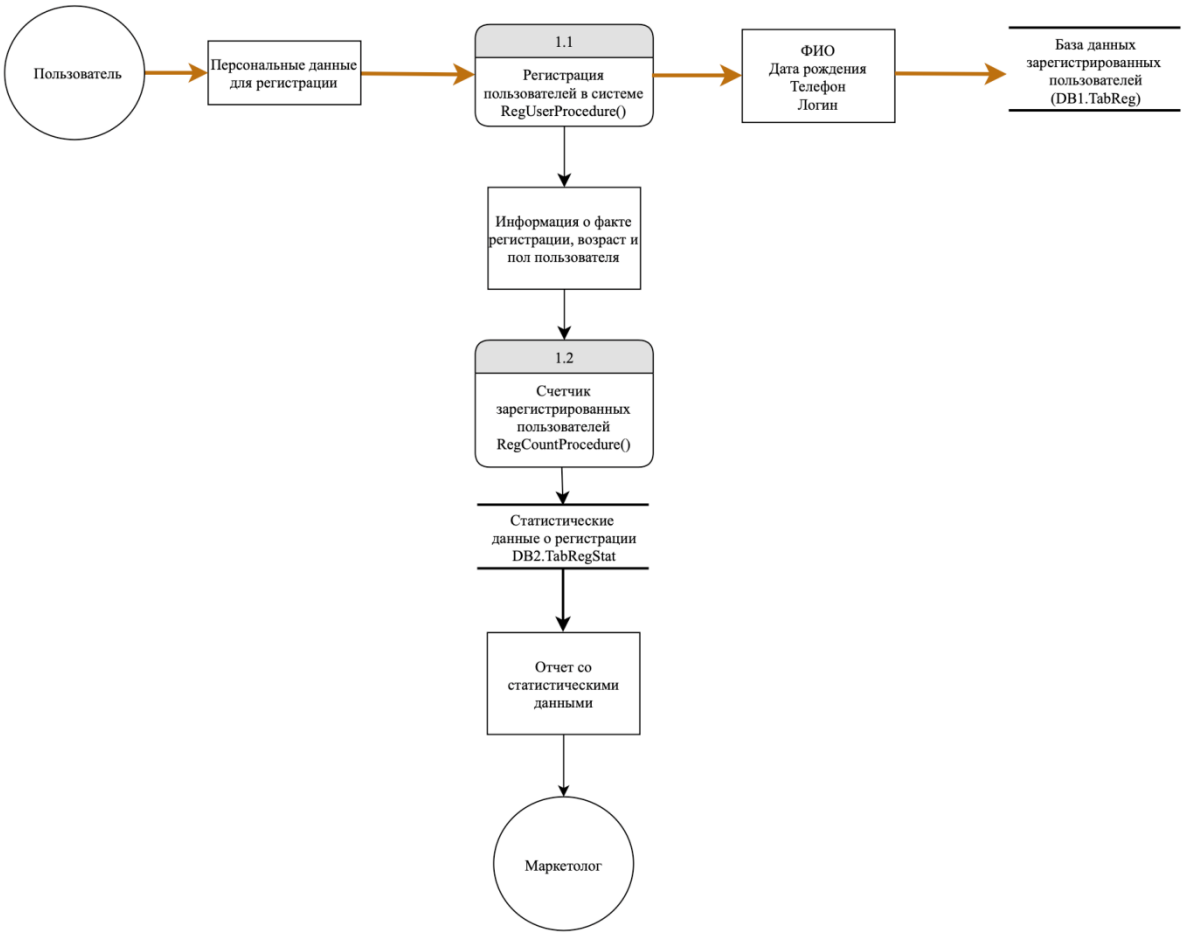


Рисунок 2. Пример диаграммы потоков данных программы

По результатам структурного анализа все выявленные угрозы безопасности информации, их описание и возможные меры по из нейтрализации включаются Разработчиком ПО в итоговый перечень угроз безопасности информации, которые могут возникать в процессах разработки ПО и при использовании ПО в составе ГИС на Платформе «ГосТех», для их последующей оценки.

Оценка выявленных угроз безопасности информации проводится Разработчиком ПО и согласовывается с Заказчиком.

По результатам оценки должен быть определен уровень риска реализации для каждой выявленной актуальной угрозы (критический, высокий, средний, низкий).

Итоговый перечень актуальных угроз безопасности информации должен быть составлен в формате таблицы № 3.

Таблица 3. Перечень актуальных угроз безопасности информации

№ п/п	Наименование/ Идентификатор УБИ (при наличии)	Описание угрозы	Уровень риска реализации УБИ	Возможные меры нейтрализации
1	УБИ. 007	Угроза заключается в возможности повышения нарушителем своих привилегий в дискредитированной системе (получения привилегии дискредитированных программ) путём использования ошибок в программах и выполнения произвольного кода с их привилегиями. Данная угроза обусловлена слабостями механизма проверки входных данных и команд, а также мер по разграничению доступа.	средний	Установка обновленной версии ПО, где такая уязвимость устранена

Результаты моделирования угроз должны быть отражены в разделе «Требования к программе или программному изделию» технического задания на разрабатываемое ПО и в моделях угроз безопасности информации ГИС на Платформе «ГосТех».

5. Конструирование и комплексирование ПО

5.1. Идентификация инструментальных средств разработки ПО

В целях получения непротиворечивых и предсказуемых результатов Разработчик ПО должен использовать идентифицированные инструментальные средства разработки (трансляторы, компиляторы, прикладные программы, используемые для проектирования и документирования, редакторы исходного кода программ, отладчики, интегрированные среды разработки и др.).

Для каждого используемого при разработке в составе ГИС на Платформе «ГосТех» инструментального средства Разработчику ПО необходимо определить и зафиксировать следующую информацию:

- наименование и идентификационные признаки инструментального средства;
- наименование разработчика инструментального средства;
- ссылку на эксплуатационные документы инструментального средства;

- значения применяемых при создании программы конфигураций инструментального средства.

Результаты проведенной процедуры идентификации должны быть отражены в проектной документации на разрабатываемое ПО или на создание/развитие ГИС на Платформе «ГосТех».

Рекомендуемая форма представления результатов в проектной документации и пример заполнения представлены в таблице № 4.

Таблица 4. Перечень инструментальных средств, используемых в процессе разработки

№ п/п	Наименование	Версия	Разработчик	Ссылка на документацию	Параметры настройки
1	Eclipse IDE	2021-03 R (4.22)	The Eclipse Foundation	https://www.eclipse.org/documentation/	Описаны в разделе 6.1 настоящего документа

Разработчик ПО должен актуализировать информацию об используемых инструментальных средствах в случаях добавления новых средств или исключения ранее использованных из процессов разработки ПО.

5.2. Создание программы на основе уточненного проекта архитектуры программы

В целях повышения уровня контроля и предотвращения появления недокументированных функций в разрабатываем ПО Разработчик ПО должен создавать программу на основе проекта архитектуры программы, определенного в ходе выполнения процессов проектирования архитектуры программы.

Для достижения заявленной цели и создания БПО Разработчик ПО должен:

- декомпозировать и уточнить проект архитектуры программы для постановки конкретных технических задач работникам Разработчика ПО, ответственным за создание программы;
- обеспечить создание исходного кода программы на основе актуального проекта архитектуры программы;
- обеспечить прослеживаемость исходного кода программы к проекту архитектуры программы (описанию проектных решений, обеспечивающих выполнение идентифицированных требований к реализуемым функциям ПО и требований по безопасности);
- проверить корректность выполнения технических задач.

Проект архитектуры ПО должен описывать структуру программы, включающую в себя подсистемы и программные модули, их свойства и отношения между ними.

Архитектуру программы рекомендуется описывать в терминах подсистем и программных модулей.

Допускается декомпозиция программных модулей с выделением функций, классов, методов и внешних интерфейсов.

Документирование архитектуры программы рекомендуется выполнять с использованием следующих нотаций и в соответствии с положениями документа «Методические рекомендации по проектированию и утверждению целевой архитектуры домена с использованием Платформы «ГосТех»:

- DFD (*Data Flow Diagram*) – диаграмма потоков данных.
- Диаграммы, используемые в методологиях группы IDEF, ArchiMate и C4;
- UML-диаграммы.

При проектировании программы Разработчику ПО следует руководствоваться принципами проектирования БПО, например: принцип минимальных привилегий, принцип эшелонированной защиты, принцип разделения обязанностей, принцип модульного проектирования, применение принятых стандартов, протоколов, форматов обмена данными.

Прослеживаемость исходного кода программы к проекту архитектуры должна быть обеспечена с помощью уникального ссылочного номера, с использованием которого возможно определить отношение между текстом исходного кода и описанием функциональных компонентов программы (функций, классов, методов, внешних интерфейсов др.) в проекте архитектуры.

Разработчик ПО для обеспечения прослеживаемости исходного кода к проекту архитектуры программы может использовать инструментальные средства.

Пример разметки исходного кода программы на языке программирования C++ для проекта архитектуры программы, представленной в формате диаграммы потоков данных (рисунок № 2), для обеспечения прослеживаемости:

```
/// Функция 1.1 RegUserProcedure (Краткое описание)
/*!
   Регистрация пользователей в системе (Подробное описание)
*/

function RegUserProcedure($vars) {}
```

Для различных языков программирования разметка исходного кода может различаться в зависимости от их особенностей.

Разработчику ПО следует обеспечить конфиденциальность исходных кодов ПО и его компонентов, в том числе при применении в его отношении средств инструментального тестирования.

Прослеживаемость к архитектуре программы должна быть продемонстрирована Разработчиком ПО в проектной документации в виде диаграмм или таблиц, также рекомендуется продемонстрировать прослеживание для каждого файла или группе файлов с исходным кодом программы с указанием в названии идентификатора соответствия компоненту из проекта архитектуры программы (например, модуль или подсистема).

5.3. Порядок оформления исходного кода программы

В целях снижения вероятности появления недостатков при создании программы, увеличения вероятности обнаружения недостатков программы и уменьшения количества ресурсов, на устранение недостатков Разработчик ПО при создании программы должен использовать рекомендованный данным документом перечень правил и рекомендаций оформления исходного кода, направленных на устранение недостатков программ в составе ГИС на Платформе «ГосТех» (потенциально уязвимых конструкций) в исходном коде.

В случае невозможности выполнения рекомендаций по оформлению исходного кода Разработчик ПО должен указывать обоснование факта отказа от их использования в форме комментариев в исходном коде программы.

Основные правила и рекомендации по оформлению исходного кода программ определяются для каждого из языков программирования, разрешенных к использованию для разработки ПО в составе ГИС на Платформе «ГосТех».

Примеры правил и рекомендаций оформления исходного кода для языков C#, JavaScript, Java представлены в Приложении № 3 настоящего документа и являются рекомендованными к применению при разработке ПО на Платформе «ГосТех».

Контроль соблюдения правил и рекомендаций осуществляется на этапах проведения статического анализа кода и экспертизы исходного кода как в автоматизированном режиме с использованием компонентов Платформы «ГосТех», так и ручном режиме.

Разработчик ПО должен проинформировать команды разработки о действующих правилах и рекомендациях по оформлению исходного кода на Платформе «ГосТех» и осуществлять самостоятельный контроль их соблюдения, так как выявление нарушений правил оформления исходного кода на этапе передачи ПО Заказчику может увеличить общее время создания ГИС на Платформе «ГосТех» и, как следствие, привести к нарушению сроков государственных контрактов (договоров, соглашений) и выставлению штрафных санкций Заказчиком в адрес Разработчика ПО.

Устранение нарушений правил и рекомендаций по оформлению исходного кода Разработчику ПО рекомендуется проводить сразу после получения отчетов с результатами проведения контрольных процедур на Платформе «ГосТех».

5.4. Статический анализ исходного кода программы

Статический анализ исходного кода программы должен проводится в целях выявления недостатков программы (потенциально уязвимых конструкций) в исходном коде программы и проверки соответствия исходного кода порядку его оформления.

Статический анализ всех компонентов, входящих в состав разрабатываемого ПО должен быть проведен до перевода ПО в продуктивную среду функционирования, с привлечением сторонней организации, обладающей

компетенцией в области выявления уязвимостей программ, при этом в целях сокращения временного периода на создание(развитие) ГИС на Платформе «ГосТех» Разработчик ПО должен обеспечить проведение статического анализа добавленных или измененных частей ПО после каждого внесенного изменения.

Рекомендуется выбирать средства статического анализа, предоставляющие возможность проведения анализа различными методами статического анализа: сигнатурного, метода анализа потоков данных, межпроцедурного контекстно-чувствительного анализа, синтаксического анализа, выявления пути распространения ошибки в программе, формальной верификации.

Для повышения уровня покрытия используемых при разработке ПО языков программирования должны использоваться средства статического анализа, позволяющие осуществить статический анализ в отношении всего исходного кода ПО.

Дополнительно рекомендуется при сравнении и выборе средств статического анализа руководствоваться следующими параметрами и характеристиками:

- поддерживаемые языки программирования;
- типы обнаруживаемых недостатков программы;
- методы и правила категоризации выявленных недостатков по уровням критичности.

В целях сокращения времени на создание(развитие) ГИС на Платформе «ГосТех» рекомендуется осуществлять программную интеграцию средств статического анализа со средствами автоматизации процесса разработки ПО, например такими, как: интегрированная среда разработки, инструментальные средства для управления версиями ПО, которые используются в системе управления конфигурацией ПО, инструментальные средства, используемые для автоматизации процесса преобразования исходного кода программы в исполняемые файлы.

Статический анализ исходного кода программы необходимо проводить как в отношении исходного кода разрабатываемой программы, так и в отношении исходного кода компонентов, заимствованных у сторонних разработчиков.

Статический анализ исходного кода заимствованных компонентов возможно не проводить в случае использования доверенных заимствованных компонентов, в отношении которых проведен статический анализ исходного кода разработчиком таких компонентов.

При проведении статического анализа должны выявляться и устраняться следующие недостатки программы:

- несоответствие правилам и рекомендациям по оформлению исходного кода (приложение № 3);
- ошибки, связанные с безопасностью программы;
- остаточный отладочный код.

Результаты каждого проведенного статического анализа должны сохраняться в систему управления конфигурацией ПО.

Потенциальные ошибки в программе, выявленные в ходе статического анализа ПО, должны быть обработаны вручную для экспертного определения их истинности или ложности. Истинные ошибки в программе должны быть помечены как подлежащие исправлению. Просмотр выданных анализатором предупреждений необходимо выполнять в случае анализа измененных частей ПО – не позже, чем через 3 рабочих дня после выполнения анализа, в случае анализа ПО целиком – не позже, чем через 10 рабочих дней после выполнения анализа.

Устранение выявленных потенциальных ошибок в программе, которые в ходе просмотра были отнесены к истинным с уровнем влияния критический и высокий, должно быть выполнено при анализе измененных частей ПО – не позже, чем через 10 рабочих дней после выполнения анализа, при анализе ПО целиком – не позднее начала квалификационного тестирования ПО. Выполненные изменения исходного кода ПО должны быть отмечены в системе контроля версий исходного кода как исправляющие предупреждения статического анализа с указанием идентификаторов выданных предупреждений в хранилище результатов анализатора.

Выявленные в ходе статического анализа потенциальные ошибки в программе, которые не были признаны истинными в ходе разметки предупреждений, следует использовать в ходе динамического анализа на этапе квалификационного тестирования ПО в качестве дополнительной информации о возможно уязвимых местах программы, если используемые инструменты динамического анализа поддерживают такую возможность.

Контроль над ходом устранения выявленных потенциальных ошибок в программе следует выполнять не реже одного раза в месяц. Рекомендуется использовать соответствующие средства разработки (система контроля версий, отслеживания ошибок в программе) для формирования отчета о количестве выявленных и исправленных ошибок в программе.

В ходе проведения регулярного статического анализа ПО конфигурация и настройка статического анализатора должны пересматриваться по мере развития ПО, добавления сторонних компонентов, выхода новых версий статического анализатора и других факторов влияния. При этом должен учитываться заданный уровень безопасности, предъявляемый к конфигурации статического анализатора.

Обязательно статический анализ должен быть проведен для каждой версии ПО, поставляемой Заказчику.

В случае использования языка программирования, не поддерживаемого статическим анализатором, должна быть проведена экспертиза исходного кода исследуемого ПО.

По результатам проведения статического анализа исходного кода программы Разработчиком ПО должны быть устранены все дефекты критического и высокого уровня влияния.

Результаты проведенного статического анализа и факт устранения выявленных недостатков программы в отношении финальной версии ПО должны быть отражены в проектной документации на создание (развитие) ГИС

на Платформе «ГосТех». Форма представления результатов приведена в таблице № 5. Допускается в процессе разработки ПО использовать электронные отчеты в формате, формируемом средствами статического анализа.

Следует обеспечить конфиденциальность информации, связанной с выявленными в ходе статического анализа исходного кода программы уязвимостями программы.

Таблица 5. Сведения о результатах проведения статического анализа исходного кода

Период проведения анализа: с ДД.ММ.ГГГГ по ДД.ММ.ГГГГ					
Наименование средства статического анализа, версия, сборка (при наличии)					
Наименование файла и номер строки в исходном коде	**Потенциально уязвимые конструкции**	**Уровень критичности**	**Описание уязвимости**	**Описание действий по устранению**	**Обоснование невозможности/отсутствия необходимости устранения (при наличии)**
sysctl_net_ipv4.c:325	user_key[i * 4],	высокий	Переполнение буфера (Чтение за границей массива)	Необходимо правильно определить индекс массива	-
bpf_struct_ops.c:505	WARN_ON_ONCE (1);	средний	Недостижимый код	-	Реализация защитного программирования

Общие рекомендации по организации процесса статического анализа исходного кода программы представлены на рисунке № 3.

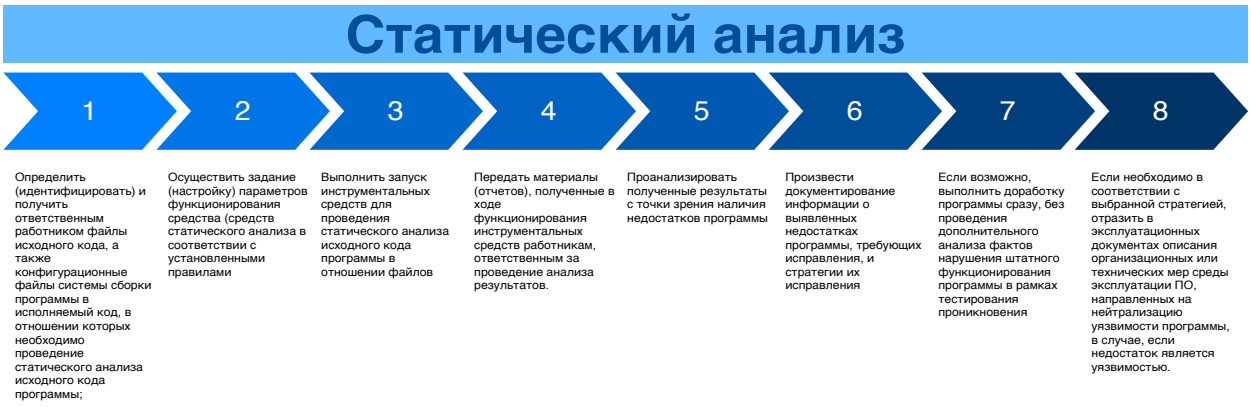


Рисунок 3. Организация процесса проведения статического анализа

5.5. Экспертиза исходного кода программы

В целях выявления потенциально уязвимых конструкций в исходном коде программы после проведения статического анализа исходного кода должна быть проведена экспертиза исходного кода, разрабатываемой программы и компонентов, заимствованных у сторонних разработчиков, если для этих компонентов доступен исходный код программы.

Экспертиза исходного кода всех компонентов, входящих в состав разрабатываемого ПО должна быть проведена до перевода ПО в продуктивную

среду функционирования, с привлечением сторонней организации, обладающей компетенцией в области выявления уязвимостей программ, при этом в целях сокращения временного периода на создание(развитие) ГИС на Платформе «ГосТех» Разработчик ПО должен обеспечить проведение экспертизы исходного кода добавленных или измененных частей ПО после каждого внесенного изменения.

В связи с трудоемкостью процедуры, экспертизу исходного кода рекомендуется проводить в отношении подмножеств исходного кода программы, реализующих наиболее критичные функции с точки зрения бизнес-процессов Заказчика и создания БПО.

Для проведения экспертизы исходного кода рекомендуется использовать инструментальные средства.

Инструментальные средства для проведения экспертизы исходного кода рекомендуется выбирать исходя из следующих параметров и характеристик:

- качество проводимого анализа;
- поддерживаемые форматы входных и выходных данных;
- возможность интеграции с другими инструментальными средствами, используемыми в процессе разработки ПО;
- поддержка многопользовательского режима работы;
- простота освоения, наличие эксплуатационных документов;
- производительность;
- требуемые ресурсы;
- стоимость.

Проведение экспертизы исходного кода рекомендуется проводить после устранения недостатков программы, выявленных по результатам статического анализа.

Экспертизу исходного кода заимствованных компонентов возможно не проводить в случае использования доверенных заимствованных компонентов, в отношении которых проведена экспертиза исходного кода разработчиком таких компонентов.

Результаты экспертизы и факт устранения выявленных недостатков программы должны быть представлены в виде протокола.

Общие рекомендации по организации процесса проведения экспертизы исходного кода программы представлены на рисунке № 4.

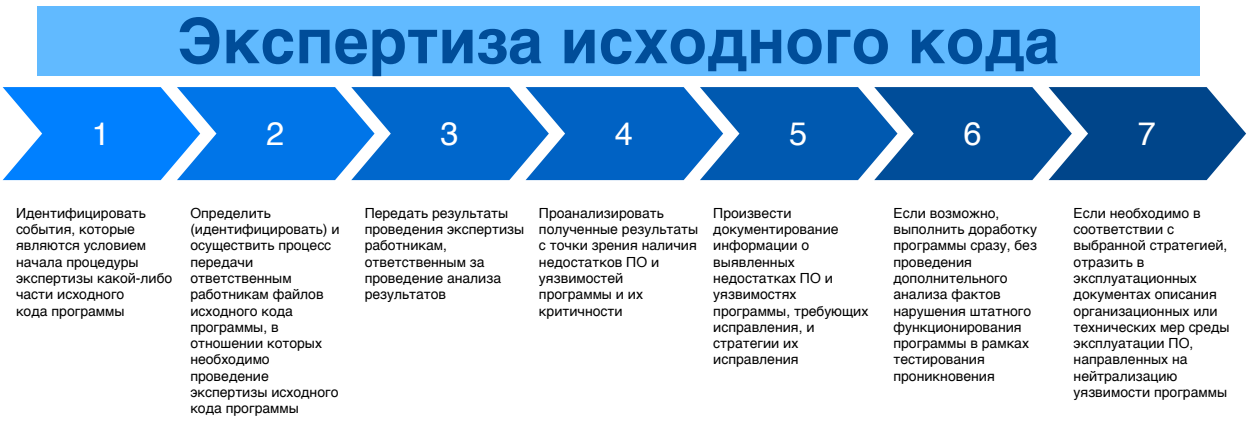


Рисунок 4. Организация процесса проведения экспертизы исходного кода программы

6. Квалификационное тестирование ПО

6.1 Функциональное тестирование программы

Для подтверждения выполнения требований по безопасности, предъявляемых к разрабатываемому ПО в составе ГИС на Платформе «ГосТех», должно проводиться функциональное тестирование программы.

Функциональное тестирование всех компонентов, входящих в состав разрабатываемого ПО должно быть проведено до перевода ПО в продуктивную среду функционирования, с привлечением сторонней организации, обладающей компетенцией в области выявления уязвимостей программ, при этом в целях сокращения временного периода на создание(развитие) ГИС на Платформе «ГосТех» Разработчик ПО должен обеспечить проведение функционального тестирования добавленных или измененных частей ПО после каждого внесенного изменения.

Методика функционального тестирования по проверке требований по безопасности к ПО должна быть включена в программу и методику испытаний на создание(развитие) ГИС на Платформе «ГосТех».

Оформление программы и методики испытаний должно осуществляться в соответствии с ГОСТ 19.301.

Программа и методика испытаний должна содержать план проведения функционального тестирований ПО, описание выполняемых тестов и перечень инструментальных средств, используемых для проверки, а также проверки обхода функциональных возможностей штатными средствами.

Рекомендованный формат описания процедуры тестирования представлен в таблице № 6.

Таблица 6. Описание процедуры функционального тестирования ПО

№ п/п	Наименование тестовой процедуры	Функциональное требование	Наименование пункта технического задания	Условия для выполнения процедуры	Шаги тестирования	Ожидаемый результат
1	**FAU_GEN.1**	**Генерация данных аудита**	**3.1.2**	**Определить события, регистрация которых производится по умолчанию в ОС. Определить журналы, в которые происходит запись событий. Просмотреть**	**Запуск и завершение выполнения функций аудита Аудит событий в ОС ведется рядом сервисов: auditd, rsyslogd, journald. Логирование происходит в файлы /var/log/audit/audit.log, /var/log/messages. Запись аудита для событий, связанных с истечением установленного администратором**	**в ОС можно настроить систему аудита для указанных в требовании событий, в каждой записи содержится указанная в требовании информация**

				содержание конфигурационных файлов /etc/audit/audit.rules, /etc/audit/rules.d/, /etc/systemd/journal.conf, /etc/rsyslog.conf, /etc/rsyslog.d/. Определить события, регистрация которых не производится по умолчанию в ОС. Сформировать правила для таких событий. Продемонстрировать, что сообщения о событиях заносятся в журнал. Убедиться, что для каждого события указаны дата, время, тип, идентификатор субъекта и результат события.	**срока действия пароля События аудита записываются в файле /var/log/secure. Запись аудита для событий, связанных с истечением установленного администратором срока действия идентификатора пользователя ОС (учетной записи) События аудита записываются в файле /var/log/secure. Блокирование учетной записи в результате превышения максимального числа неуспешных попыток входа в систему Для настройки ограничения по неуспешным попыткам доступа необходимо в файл /etc/pam.d/system-auth и /etc/pam.d/password-auth добавить 2 строки: - в секцию auth (установить самой верхней строкой): auth required pam_tally2.so file=/var/log/tallylog deny=3 even_deny_root unlock_time=1200 где deny=3 задает количество неуспешных попыток входа пользователя до его блокировки unlock_time — время, на которое блокируется пользователь (если не указывать данный параметр, то разблокировать пользователя можно будет только вручную с помощью суперпользователя root), - в секцию account (после строки “account required pam_unix.so”) account required pam_tally2.so События будут записываться в файл**	

Программа и методика испытаний должна предусматривать проверки для всех требований, предъявляемых к разрабатываемому ПО и обеспечивать наиболее полную проверку этих требований.

Для автоматизации выполнения тестовых процедур, а также для автоматизации управления процессом тестирования и хранения информации о результатах проведения функционального тестирования могут использоваться инструментальные средства.

Все используемые инструментальные средства должны быть идентифицированными в соответствии с рекомендациями п. 5.1 данного документа.

Результаты проведения функционального тестирования ПО должны быть включены в протокол предварительных испытаний ГИС на Платформе «ГосТех».

Рекомендуемый формат оформления результатов функционального тестирования ПО представлен в таблице № 7.

Таблица 7. Результаты функционального тестирования ПО

Функциональное требование	Функция	ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ (РЕКВИЗИТЫ ДОКУМЕНТА)		Результат тестирования
		№	Наименование тестовой процедуры	
FAU_GEN.1	Регистрация событий и аудит	3.1.2	FAU_GEN.1 Генерация данных аудита	В ОС можно настроить систему аудита для указанных в требовании событий, в каждой записи содержится указанная в требовании информация

Все выявленные недостатки программы критического и высокого уровня влияния должны быть устранены Разработчиком ПО до момента ввода ГИС на Платформе «ГосТех» в эксплуатацию.

Общие рекомендации по организации процесса проведения функционального тестирования приведены на Рисунке № 5.

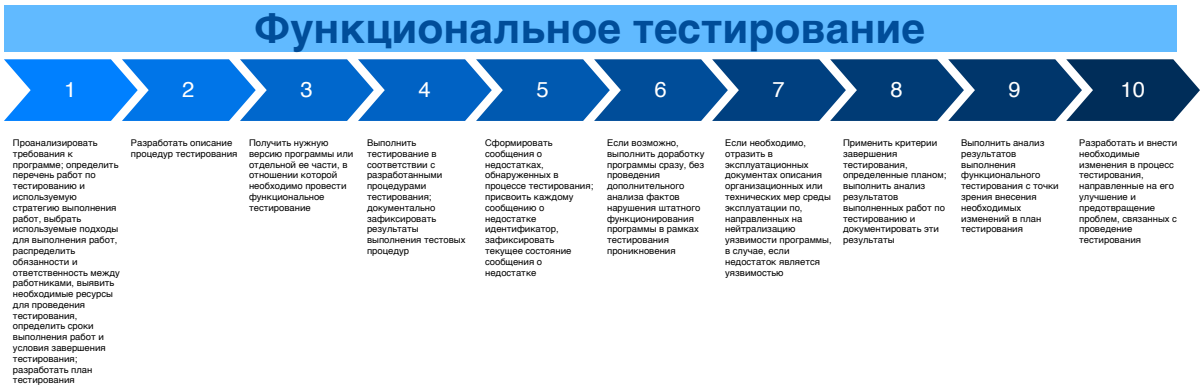


Рисунок 5. Организация процесса проведения функционального тестирования

6.2 Фаззинг-тестирование программы

В рамках проведения квалификационного тестирования разрабатываемого ПО должно быть проведено фаззинг-тестирование программы для выявления уязвимостей программы.

Фаззинг-тестирование должно быть проведено до перевода ПО в продуктивную среду функционирования, с привлечением сторонней организации, обладающей компетенцией в области выявления уязвимостей программ, при этом в целях сокращения временного периода на создание(развитие) ГИС на Платформе «ГосТех» Разработчик ПО должен обеспечить проведение фаззинг-тестирования добавленных или измененных частей ПО после каждого внесенного изменения.

В процессе фаззинг-тестирования осуществляется манипулирование входными данными программы или ее компонентов и фиксирование нарушений штатного поведения в результате обработки таких данных.

Выявленные нарушения анализируются с целью формирования перечня потенциальных уязвимостей программы, который в дальнейшем используется при проведении тестирования проникновения.

Требуется выполнять фаззинг-тестирование модулей ПО с инструментированием кода модулей, обеспечивающим реализацию генетических алгоритмов фаззинг-тестирования, в случае если доступны инструменты, позволяющие осуществлять такое фаззинг-тестирование в отношении ПО и его модулей с учетом языков программирования, компиляторов и среды функционирования ПО.

При выборе инструментальных средств рекомендуется руководствоваться следующими характеристиками:

- поддерживаемые типы фаззинг-тестирования;
- поддерживаемые протоколы и форматы входных данных;
- наличие функции автоматического формирования отчетов о результатах тестирования (функция обратной связи).

Рекомендуется использовании в фаззинг-тестировании сборки ПО с санитайзерми, с целью выявления критических ошибок в программе, не приводящих к аварийному завершению.

Рекомендуется проводить фаззинг-тестирование в отношении всей программы или ее отдельных компонентов, например библиотек, исполняемых компонентов, обеспечивающих обработку входящего сетевого трафика, команд или файлов. Следует сосредоточить усилия в первую очередь на компонентах программы, требующих для функционирования повышенных привилегий или реализующий функции безопасности. Рекомендуется проводить фаззинг-тестирование для экспортируемых интерфейсов библиотек и для всех интерфейсов, через которые программа получает внешние данные. Следует проводить фаззинг-тестирование в отношении заимствованных у сторонних разработчиков компонентов.

Формирование тестовых наборов входных данных осуществляется вручную или с использованием инструментальных средств (фаззер).

Для генерации тестовых наборов входных данных фаззер может использовать: эталонные образцы входных данных (файлы), исходный код программы или компонента программы, сведения о форматах входных данных программы, используемых протоколах взаимодействия.

Результаты фаззинг-тестирования должны быть представлены в виде протокола.

Следует обеспечить конфиденциальность информации, связанной с выявленными в ходе фаззинг-тестирования программы уязвимостями программы.

Общие рекомендации по организации процесса проведения фаззинг-тестирования приведены на Рисунке № 6.

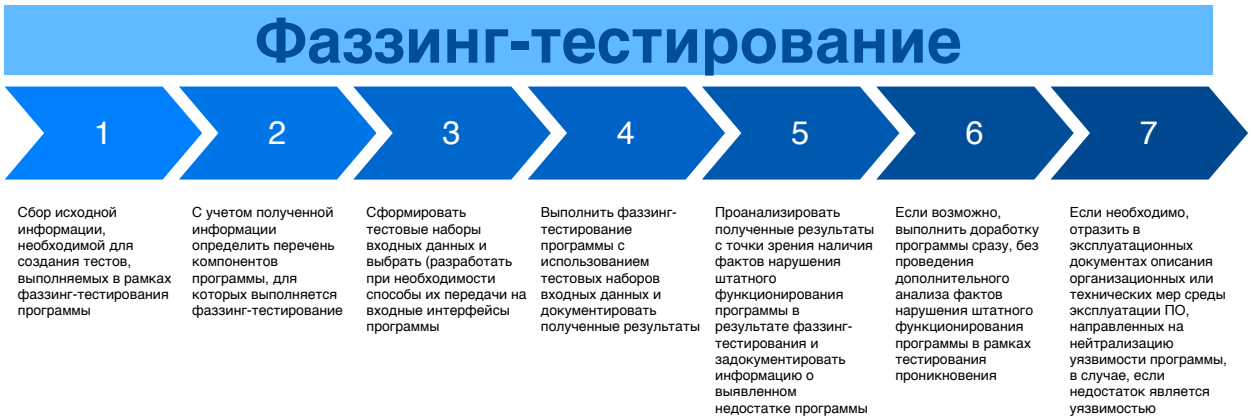


Рисунок 6. Организация процесса проведения фаззинг-тестирования

6.3 Динамический анализ кода программы

В рамках проведения квалификационного тестирования разрабатываемого ПО для ГИС на Платформе «ГосТех» необходимо проводить динамический анализ кода программы для выявления недостатков программы.

Динамический анализ кода программы должен быть проведен до перевода ПО в продуктивную среду функционирования, с привлечением сторонней организации, обладающей компетенцией в области выявления уязвимостей программ, при этом в целях сокращения временного периода на создание(развитие) ГИС на Платформе «ГосТех» Разработчик ПО должен обеспечить проведение динамического анализа кода программы добавленных или измененных частей ПО после каждого внесенного изменения.

Динамический анализ кода программы должен проводится в режиме непосредственного исполнения (функционирования) кода с применением инструментальных средств.

Результаты тестирования должны быть представлены в виде протокола.

Следует обеспечить конфиденциальность информации, связанной с выявленными в ходе динамического анализа кода программы уязвимостями программы.

Для выполнения динамического анализа рекомендуется применять следующие методы:

- динамическая бинарная трансляция;
- анализ помеченных данных;
- снятие трассы, выявление пути распространения ошибки.

При выборе инструментальных средств рекомендуется руководствоваться следующими параметрами:

- поддерживаемые среды функционирования программ, для которых возможно проведение динамического анализа;
- типы обнаруживаемых недостатков и реализованные методы динамического анализа (анализ активности и потоков взаимодействия программы, динамическая бинарная трансляция, отслеживание помеченных данных, сканирование интерфейсов получения входных данных, трассировка).

Динамический анализ рекомендуется проводить поэтапно для различных частей исходного кода, разделенных по степени его критичности с точки зрения обеспечения информационной безопасности.

В зависимости от выбранных инструментальных средств и соответствующих методов проведения динамического анализа кода программы, в некоторых случаях, рекомендуется выполнение инструментации исходного кода, которую необходимо выполнить до осуществления преобразования исходного кода программы в исполняемый код. Этот факт следует учесть при внедрении процедур динамического анализа кода программы в процесс разработки ПО, а именно в описании процедуры преобразования исходного кода в исполняемый необходимо предусмотреть этап получения инструментированного экземпляра исходного кода программы или отдельных ее модулей.

Запуск инструментальных средств динамического анализа может выполняться непосредственно работниками вручную или осуществляться автоматически инструментальными средствами, которые используются в процессе конструирования и комплексирования ПО, в зависимости от наличия технческой возможности автоматизации в конкретной реализации информационной технологии.

Динамический анализ рекомендуется применять на регулярной основе при выполнении одного или нескольких из следующих условий:

- любое преобразование исходного кода в исполняемый и выпуск любых промежуточных версий ПО;
- выпуск версий ПО, предназначенных для передачи Заказчику;
- наступление определенных временных условий, определяемых организационно-распорядительными документами Разработчика ПО (рекомендуется проводить динамический анализ не реже 1 раза в 180 дней);
- переход процесса разработки на новую стадию.

Общие рекомендации по организации процесса проведения динамического анализа приведены на Рисунке № 7.

Рисунок 7. Организация процесса проведения динамического анализа

6.4 Тестирование на проникновение

В целях выявления уязвимостей Заказчиком должно быть организовано проведение тестирования на проникновение в отношении ГИС на Платформе «ГосТех», с привлечением сторонних организаций, обладающих компетенцией в области проведения такого рода испытаний.

В рамках организованного тестирования на проникновение также должны быть проведены тесты, направленные на выявление уязвимостей в ПО в составе ГИС на Платформе «ГосТех».

Тестирование на проникновение предполагает выявление уязвимостей программы путем моделирования (имитации) действий потенциального нарушителя.

Выполняемые тесты должны разрабатываться с учетом:

- проекта архитектуры программы;
- результатов моделирования угроз безопасности информации;
- результатов статического анализа;
- результатов экспертизы исходного кода;
- результатов динамического анализа кода программы;
- фаззинг-тестирования.

При определении проверяемых в ходе тестирования на проникновения тестовых конфигураций программы рекомендуется зафиксировать характеристики элементов среды функционирования ПО, например версии операционных систем или систем управления базами данных.

Для разработки тестов рекомендуется использовать общепризнанные стандарты и руководства по анализу защищенности, включая:

- Penetration Testing Execution Standard (PTES);
- Open Source Security Testing Methodology Manual (OSSTMM);
- Open Web Application Security Project (OWASP) Testing Guide;
- Common Vulnerability Scoring System (CVSS).

Тестирование на проникновение должно включать в себя следующие этапы:

- внешнее тестирование на проникновение, состоящее из анализа защищенности сетевого периметра среды функционирования ПО, направленное

на получение доступа к ПО через элементы среды функционирования и дальнейшее развитие атаки через эти элементы;

- внутреннее тестирование на проникновение.

Внешнее тестирование на проникновение должно быть направлено на демонстрацию возможных сценариев атаки, позволяющие нарушителю обойти механизмы защиты среды функционирования ПО и функций безопасности ПО в целях получения максимальных привилегии в критически важных компонентах. Работы рекомендуется проводить методом «черного ящика» (без предварительных знаний о ПО и среде функционирования) от лица высококвалифицированного внешнего нарушителя со стороны сети Интернет.

В рамках внешнего тестирования на проникновение рекомендуется провести анализ защищенности сетевого периметра среды функционирования, направленный на получение доступа к ее элементам с наивысшими привилегиями.

Для этого рекомендуется провести следующие мероприятия:

- пассивный сбор информации об элементах среды функционирования ПО;
- активный сбор информации о среде функционирования ПО и самой программы, в том числе сканирование портов, идентификация доступных сетевых служб, ручной анализ отдельных служб;
- внешнее инструментальное сканирование, направленное на выявление уязвимостей в доступных сетевых службах среды функционирования;
- анализ безопасности программы с использованием инструментальных средств и ручного анализа;
- ручной анализ уязвимостей, включающий выявление недостатков аутентификации, важной информации в открытом доступе, недостатков контроля доступа;
- подбор учетных данных;
- эксплуатацию одной или нескольких выявленных уязвимостей.

Развитие атаки с использованием полученных привилегий и перечисленных выше методов.

Внутреннее тестирование должно продемонстрировать получение максимально возможного уровня доступа в критически важные элементы среды функционирования ПО с использованием модели внутреннего нарушителя.

Внутреннее тестирование на проникновение рекомендуется проводить без использования наложенных средств защиты информации среды функционирования и отключенными функциями защиты информации среды функционирования (при их наличии и технической возможности их отключения).

При внутреннем тестировании рекомендуется проводить следующие виды проверок (которые проводятся в зависимости от текущего уровня привилегий и приемлемости рисков, связанных с данными проверками):

- подключение к среде функционирования ПО, в том числе, при необходимости, обход механизмов защиты;

- активный сбор информации о среде функционирования, включая инструментальное сканирование открытых портов и идентификацию доступных сетевых служб;

- инструментальное сканирование, направленное на поиск известных уязвимостей на исследуемых компонентах ПО и ресурсах среды функционирования;

- анализ защищенности ПО как инструментальными, так и ручными методами;

- подбор учетных данных (идентификаторов, паролей, ключей) как напрямую путем прямых попыток доступа к ресурсам, так и офлайн-подбор на основе хеш-значений;

-эксплуатация одной или нескольких выявленных уязвимостей и повышение привилегий (при наличии возможности);

- развитие атаки с использованием перечисленных выше методов вплоть до получения доступа к одному или нескольким критически важным компонентам среды функционирования либо до исчерпания всех доступных на момент тестирования методов атак.

Результаты тестирования на проникновение должены быть отражены в тестовой документации на ГИС на Платформе «ГосТех». Описание организационных и (или) технических мер защиты, направленных на нейтрализацию недостатков программы и уязвимостей, выявленных по результатам тестирования, должны быть отражены в эксплуатационной документации на ГИС на Платформе «ГосТех».

Следует обеспечить конфиденциальность информации, связанной с выявленными в ходе тестирования на проникновение уязвимостями программы.

Общие рекомендации по организации процесса проведения тестирования на проникновение приведены на Рисунке № 8.

Рисунок 8. Организация процесса проведения тестирования на проникновение

7. Инсталляция программы и поддержка приемки ПО

7.1. Обеспечение защиты ПО в процессе его передачи Заказчику

В целях обнаружения модификации ПО или любого расхождения между оригиналом и версией, полученной Заказчиком, рекомендуется использовать механизмы электронной подписи Разработчика ПО (технология code signing) и(или) средства контрольного суммирования поставляемого дистрибутива программы, пломбирование упаковки с поставляемым дистрибутивом программы и документацией, наклейкой, разрываемой при первом вскрытии упаковки.

Средство контрольного суммирования должно соответствовать требованиям документов ФСТЭК России и ФСБ России и обеспечивать:

- вычисление контрольных сумм, заданных файлов/директорий по алгоритмам ГОСТ Р 34.11-2012;
- совместимость с операционными системами семейства Linux;
- возможность выгрузки результатов в структурированном формате, например CSV;
- возможность сравнения контрольных сумм заданных файлов/директорий в соответствии со списком.

При использовании средств автоматизации процесса доставки электронных версий документации и дистрибутивов ПО все версии электронных документов и дистрибутивов должны быть подписаны квалифицированной усиленной электронной подписью ответственного (уполномоченного) сотрудника Разработчика ПО.

Также в целях обнаружения модификации файлов программы (или отдельных ее частей, например, обновлений), распространяемых на физическом носителе или по каналам связи, рекомендуется обеспечивать возможность аутентификации источника, который служит хранилищем для распространяемых файлов.

В качестве мер для программ (и их частей), которые распространяются на физических носителях, применяется упаковка с применением средств, позволяющих обнаружить нарушение ее физической целостности (защитные ленты, наклейки, печати, которые разрушаются при нарушении целостности упаковки).

В комплекте эксплуатационных документов следует включить описание действий Заказчика, позволяющие выполнить проверку целостности упаковки, проверке электронных подписей и контрольных сумм экземпляров ПО.

Данные рекомендации должны применяться в совокупности с рекомендациями, представленными в документе «Методические рекомендации по включению сервисов в единую цифровую платформу Российской Федерации «ГосТех».

7.2. Поставка Заказчику эксплуатационных документов

В состав поставляемого ПО должны быть включены эксплуатационные документы (Руководство пользователя и Руководство администратора) в объеме, достаточном для безопасной установки, инициализации и безопасного применения программы.

Руководство пользователя должно содержать описание:

- режимов работы средства;
- принципов безопасной работы средства;
- функций и интерфейсов функций средства, доступных каждой роли пользователей;
- параметров (настроек) безопасности средства, доступных каждой роли пользователей, и их безопасных значений;
- типов событий безопасности, связанных с доступными функциями средства;
- действий после сбоев и ошибок эксплуатации средства.

Руководство администратора должно содержать описание:

- действий по приемке поставленного средства;
- действий по безопасной установке и настройке средства;
- действий по реализации функций безопасности среды функционирования средства.

При развертывании ПО и его компонентов должна использоваться эталонная безопасная конфигурация.

В эксплуатационных документах следует определить перечень и эталонные значения конфигурационных параметров программы, которые должны быть разработаны с учетом требования безопасности. Данную информацию можно использовать для выявления уязвимостей программы, появившихся в результате определения конфигурации (параметров настройки) программы. Виды разрабатываемых эксплуатационных документов могут соответствовать ГОСТ 19.101.

Данные рекомендации должны применяться в совокупности с «Методическими рекомендациями по включению сервисов в единую цифровую платформу Российской Федерации «ГосТех».

8. Решение проблем в процессе эксплуатации

8.1. Отслеживание и исправление обнаруженных недостатков программы и уязвимостей

Целью отслеживания и исправления обнаруженных недостатков и уязвимостей является сокращение негативного влияния недостатков программы на процессы Заказчика, которые автоматизируются ГИС на Платформе «ГосТех».

Отслеживание и исправление обнаруженных недостатков и уязвимостей должно осуществляться Разработчиком ПО в период действия гарантийных обязательств, указанных в государственном контракте (договоре, соглашении) на создание, развитие, эксплуатацию ГИС на Платформе «ГосТех».

Разработчик ПО должен определить и реализовать процедуры, позволяющие выполнять отслеживание и исправление обнаруженных ошибок в программе и уязвимостей, в том числе процедуры обеспечивающие прием и обработку сообщений от Заказчика об ошибках в программе и уязвимостях, а также запросов на их устранение путем обновления ПО и информировании Заказчика о выявленных уязвимостях и рекомендациях по их устранению.

Документация Разработчика ПО должна содержать:

- описание процедуры отслеживания и исправления обнаруженных ошибок в программе и уязвимостей;
- описание методов приема и обработки сообщений от Заказчика об ошибках в программе и уязвимостях;
- описание методов доведения до Заказчика информации о выявленных ошибка в программе и уязвимостях, а также рекомендаций по их устранению, в том числе путем обновления ПО.

Отслеживание и исправление обнаруженных ошибок в программе и уязвимостей обычно включают в себя следующие этапы:

- сбор информации об ошибках в программ и уязвимостях по результатам тестирований, проведенных на этапах «Конструирования и комплексирования ПО» и «Квалификационного тестирования»;
- классификация и категоризация ошибок в программе и уязвимостей по уровням критичности;
- формирование задач на устранение ошибок в программе и уязвимостей «критического» и «высокого» уровней влияния;
- устранение ошибок в программке и уязвимостей;
- проверка и контроль устранения;
- выпуск экземпляра программы с устраненными ошибками в программе и уязвимостями;
- информирование Заказчика о выявленных ошибках в программе и уязвимостях , а также о рекомендациях по их устранению.

Разработчик ПО может осуществлять выпуск обновлений ПО в обход стандартной процедуры выпуска новых версий ПО в экстренных ситуациях[2] или Разработчик ПО должен предложить альтернативные меры нейтрализации угроз безопасности информации или угроз нарушения штатного режима функционирования ПО, включая использование дополнительных средств защиты информации.

Для выполнения рекомендаций Разработчик ПО может использовать идентифицированные инструментальные средства, обеспечивающие хранение

2 К экстренным ситуациям относятся: выявление недостатков программы или уязвимостей или заимствованных компонентов в составе ПО, которые могут привести к нарушению свойств защиты информации, обрабатываемой ПО в составе ГИС на Платформе «ГосТех».

информации о недостатках программы, отслеживание и управление недостатками программы.

Ответственным за отслеживание и исправление обнаруженных недостатков и уязвимостей программы является Разработчик ПО.

Информацию о недостатках и уязвимостях рекомендуется фиксировать в виде структурированного отчета, содержащего следующие основные данные: идентификатор, краткое описание, подробное описание, шаги по воспроизведению, важность, срочность, категорию (уровень влияния/критичности), возможность применения обходного/компенсирующего решения, версия программы, в которой обнаружен недостаток. Дополнительно в отчет рекомендуется включать ссылки на различные файлы и другие материалы, необходимые для описания недостатка.

Идентификатор – это уникальное значение, которое позволяет отличить один отчет от другого.

Краткое описание отражает основную суть недостатка.

Подробное описание предоставляет детальные сведения о недостатке.

Шаги по воспроизведению описывают порядок действий, которые необходимо выполнить, чтобы недостаток проявился.

Возможность обхода предоставляет информацию, необходимую для описания условий, выполнение которых необходимо для исключения проявления недостатка без необходимости изменения исполняемого кода ПО.

Отчет о недостатках рекомендуется дополнять по мере изменения текущего состояния недостатка.

Целесообразно предусмотреть несколько значений, позволяющих определить текущее состояние недостатка в любой момент времени. Указанные значения могут отражать следующие состояния:

- начальное состояние, после обнаружения;
- состояние, когда недостаток подтвержден и начаты работы по его устранению;
- состояние после устранения недостатка;
- состояние после подтверждения устранения недостатка;
- завершение процесса устранения недостатка;
- устранение недостатка отложено;
- устранение недостатка не планируется.

Информация о недостатках и уязвимостях может быть получена от пользователей ГИС на платформе «ГосТех», работников Заказчика, работников Разработчика ПО, экспертных организаций и иных лиц, обеспечивающих поиск недостатков и уязвимостей программы с использованием функционального тестирования, статического анализа, динамического анализа, экспертизы исходного кода, фаззинг-тестирования, тестирования на проникновение, систематического поиска уязвимостей программы.

Информацию о недостатках, полученную из различных источников, целесообразно фиксировать и обрабатывать в едином хранилище. Для управления таким хранилищем и взаимодействия с ним целесообразно использовать специализированное ПО.

При устранении недостатков рекомендуется выполнять анализ причин возникновения недостатков, чтобы предотвратить появление новых недостатков по похожим причинам.

Для устранения недостатка в исходный код ПО и другие части разрабатываемого ПО вносятся необходимые изменения. После устранения недостатка следует выполнить проверку устранения и оповещение ответственных работников о данном факте, после чего внести изменение в отчет о недостатке, предоставить заказчику экземпляр ПО с устраненным недостатком.

8.2. Систематический поиск уязвимостей программы

В целях контроля и анализа защищенности ПО в составе ГИС на Платформе «ГосТех» должен проводиться систематический поиск уязвимостей.

Систематический поиск уязвимостей должен проводится Разработчиком ПО как в отношении частей программы, которые создаются непосредственно Разработчиком ПО, так и в отношении заимствованных компонентов сторонних разработчиков на регулярной основе (не реже 1 раза в месяц) в период действия гарантийных обязательств, указанных в государственном контракте (договоре, соглашении) на создание, развитие, эксплуатацию ГИС на Платформе «ГосТех».

Выявление уязвимостей рекомендуется осуществлять как с использованием инструментальных средств, так и в ручном режиме.

Для выявления уязвимостей ПО рекомендуется использовать данные об уязвимостях из открытых источников, документацию на программу, а также данные, являющиеся результатом выполнения различных процедур безопасной разработки ПО (результаты статического анализа исходного кода, результаты экспертизы исходного кода, результаты функционального тестирования, результаты динамического анализа кода, результаты фаззинг-тестирования).

В случае выявления уязвимости в тестируемой версии программы следует идентифицировать все остальные версии программы, в которых есть обнаруженная уязвимость.

Разработчик ПО в отношении уязвимостей критического уровня влияния обязан принять меры по их устранению и в срок до 24 часов, с момента обнаружения уязвимости, проинформировать оператора ГИС на Платформе «ГосТех» и оператора Платформы «ГосТех» об о их наличии и способах устранения.

Разработчик ПО в отношении уязвимостей выского уровня влияния обязан принять меры по их устранению и в срок до 7 дней, с момента обнаружения уязвимости, проинформировать оператора ГИС на Платформе «ГосТех» и оператора Платформы «ГосТех» об о их наличии и способах устранения.

В общем случае устранение выявленных уязвимостей должно осуществляться в соответствии с рекомендациями п. 8.1 данного документа.

9. Управление документацией и конфигурацией программы

9.1. Уникальная маркировка каждой версии ПО

Для идентификации различных версий программы Разработчику ПО следует проводить уникальную маркировку каждой версии программы и ее частей.

Для обозначения версий программ, исполняемых файлов, файлов установочного комплекта рекомендуется использовать группу цифр (номеров), разделенных точками.

Новые версии должны маркироваться посредством изменения вышеуказанных групп цифр в сторону увеличения, в зависимости от типа изменений, внесенных в программу. Группы цифр в некоторых случаях возможно дополнять буквенными значениями, также обладающими определенным смыслом.

Описание конкретного способа маркировки Разработчик ПО должен отразить в эксплуатационной документации, поставляемой Заказчику.

9.2. Использование системы управления конфигурацией

В целях поддержания соответствия разрабатываемого ПО заданным требованиям в течение всего жизненного цикла Разработчик ПО должен создать и использовать систему управления конфигурацией ПО, которая должна предоставлять возможность и средства для определения всех элементов конфигурации, на которые воздействует модификация тот или иной элемент конфигурации.

Рекомендуемый список элементов конфигурации разрабатываемого ПО определен в Приложении 4.

Для идентификации принадлежности и контроля целостности всех элементов конфигурации рекомендуется использовать технологии электронной подписи Разработчика ПО.

Подобная идентификация выполняется с использованием различных меток или путем создания отдельного перечня идентификаторов элементов конфигурации, которые связаны с реализацией функций безопасности.

10. Управление инфраструктурой среды разработки ПО

10.1. Защита от несанкционированного доступа к элементам конфигурации

Разработчик ПО должен обеспечить защиту элементов конфигурации ПО от угроз безопасности информации, связанных с нарушением их конфиденциальности, целостности, доступности и аутентичности.

Для защиты элементов конфигурации Разработчик ПО должен применять технические и организационные меры.

Разработчику ПО рекомендуется включать в состав документации на ПО, связанной с реализацией меры по разработке безопасного ПО, документы Разработчика ПО, соответствующие требованиям п. 5.8.3.1 ГОСТ Р 56939.

10.2. Резервное копирование элементов конфигурации

Разработчик ПО должен осуществлять резервное копирование элементов конфигураций.

Резервное копирование рекомендуется осуществлять в отношении всех элементов конфигурации ПО.

Периодичность проведения резервного копирования рекомендуется определять с учетом активности выпуска новых версий ПО и изменения его конфигураций.

При резервном копировании элементов конфигураций Разработчиком ПО должна обеспечиваться их конфиденциальность, целостность, доступность и аутентичность.

При восстановлении конфигурации из резервной копии Разработчиком ПО должна осуществляться проверка ее целостности и аутентичности.

Реализацию мер по резервному копированию и восстановлению элементов конфигурации следует осуществлять в соответствии с ГОСТ Р ИСО/МЭК 27002.

10.3 Регистрация событий, связанных с фактами изменения элементов конфигурации

Разработчик ПО должен применять технические и организационные меры, обеспечивающие регистрацию всех событий, связанных с фактами изменения элементов конфигурации, в журналах регистрации событий.

Рекомендуется осуществлять регистрацию следующей информации:

- инициатор изменения;
- идентификатор элемента конфигурации;
- дата и время изменения элемента конфигурации;
- информация о субъекте, внесшим изменения (IP-адрес и/или иная информация, позволяющая идентифицировать субъект);
- информация о внесенном изменении.

Реализацию мер, обеспечивающих регистрацию всех событий, связанных с фактами изменения элементов конфигурации, в журналах регистрации событий следует осуществлять в соответствии с ГОСТ Р ИСО/МЭК 27002.

11. Обучение работников и периодический анализ программы обучения

Разработчик ПО должен проводить периодическое обучение сотрудников с целью повышения их осведомленности в области разработки безопасного ПО.

В программу обучения рекомендуется включать курсы, посвященные моделированию угроз безопасности информации, проведению экспертизы исходного кода программы, тестированию на проникновение, статическому анализу исходного кода программы, сертификации средств защиты информации и иным направлениям обучения, связанным с разработкой БПО.

Программа обучения сотрудников должна ежегодно пересматриваться Разработчиком ПО для установления ее пригодности, адекватности и результативности для достижения установленных целей в области разработки безопасного ПО.

Обучение работников в области разработки безопасного ПО рекомендуется проводить с привлечением сторонних организаций, обладающих компетенцией в этой области и имеющих лицензию на осуществление образовательной деятельности

Программы обучения необходимо организовывать с учетом должностных и функциональных обязанностей соответствующих работников.

Для отдельных ролей может быть предусмотрено более детальное изучение аспектов обеспечения безопасности информации в зависимости от возложенных обязанностей: создание безопасного исходного кода, базовые методы преодоления механизмов защиты информации, типовые компьютерные атаки. Определенные работники могут проходить углубленное изучение проблем и методик в области обеспечения безопасности информации для последующего выполнения специфических задач в области безопасной разработки (статический анализ, тестирование на проникновение, динамический анализ, фаззинг-тестирование) и передачи знаний остальным работникам.

Программу обучения работников следует составлять с учетом технологий, языков программирования и инструментальных средств, которые используются в процессе разработки безопасного ПО на Платформе «ГосТех».

Разработчику ПО рекомендуется проводить регулярную оценку знаний работников в области разработки безопасного ПО. Оценка может быть проведена в виде тестирования. Периодическое тестирование позволяет получить сведения о наличии у работников необходимых знаний в области безопасной разработки и получить необходимые данные об эффективности принятой программы обучения. Эффективным средством для оценки реализованной программы обучения работников являются метрики, характеризующие процесс разработки ПО. Набор рассчитываемых метрик зависит от особенностей процесса разработки. Примеры метрик, которые могут быть использованы при оценке программы обучения, следующие: общее количество обнаруженных уязвимостей и недостатков программы за период времени, общее количество уязвимостей и недостатков программы, обнаруженных после передачи ПО Заказчику, средний уровень важности

обнаруженных уязвимостей и недостатков программы, успешное прохождение тестовых процедур. Динамика изменения метрик в наблюдаемых периодах времени позволяет определить необходимость обновления программы обучения или отсутствие такой необходимости.

Дополнительные сведения об эффективности и пригодности используемой программы обучения могут быть также получены путем опроса работников, которые прошли обучение.

Приложение № 1

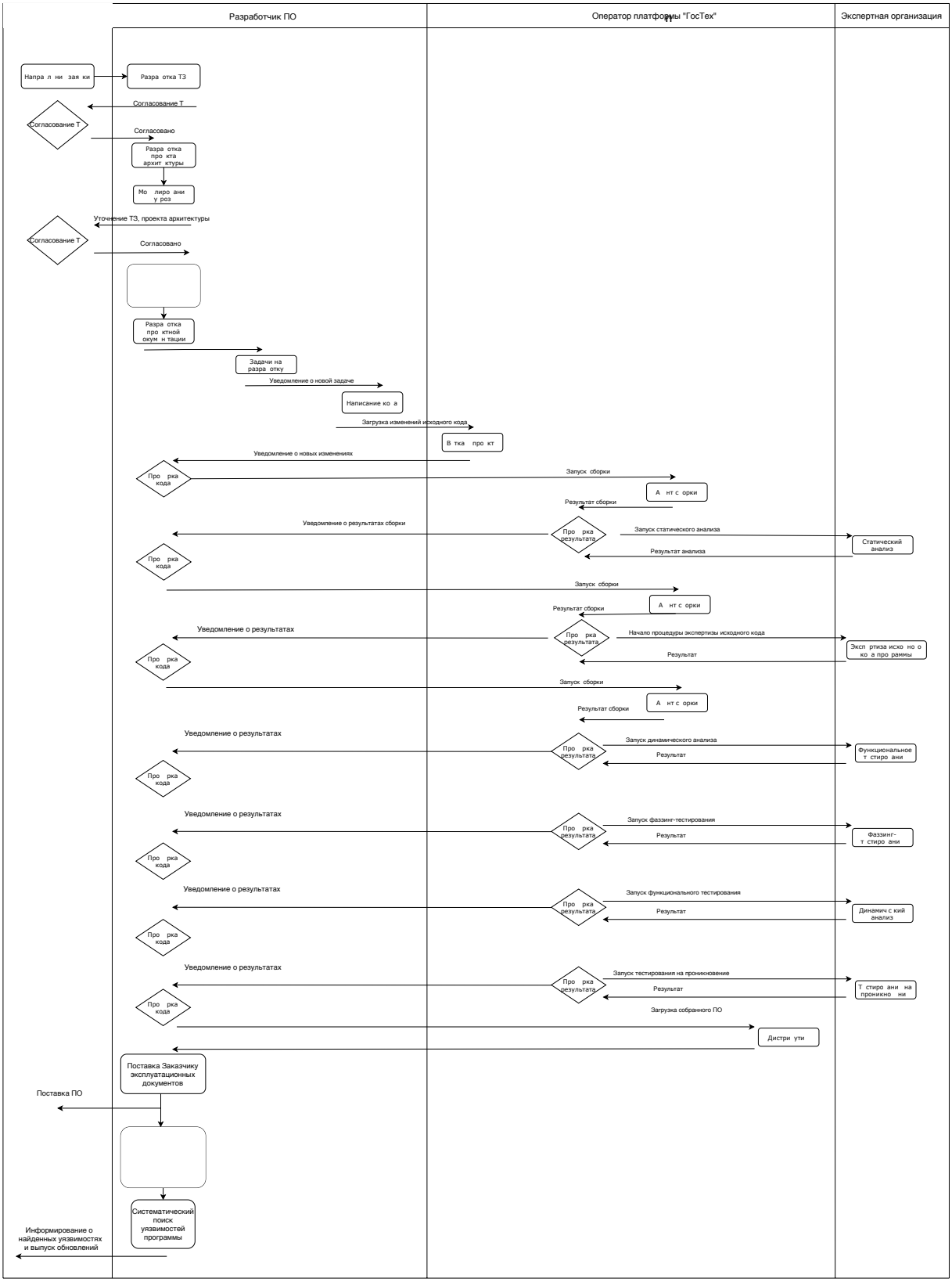
МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЕДИНАЯ ЦИФРОВАЯ ПЛАТФОРМА РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ» ДЛЯ СОЗДАНИЯ, РАЗВИТИЯ И ЭКСПЛУАТАЦИИ ГОСУДАРСТВЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТОВ ЕДИНОЙ ЦИФРОВОЙ ПЛАТФОРМЫ РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ»

Схема организации процесса разработки безопасного ПО на Платформе «ГосТех»

2022 г.


Разработчик ПО
Оператор платформы "ГосТех"
Экспертная организация
Напра л ни зая ки
Разра отка ТЗ
Согласование Т
Согласование Т
Согласовано
Разра отка про кта архит ктуры
Мо лиро ани у роз
Уточнение ТЗ, проекта архитектуры
Согласование Т
Согласовано
Разра отка про ктной окум нтации
Задачи на разра отку
Уведомление о новой задаче
Написание ко а
Загрузка изменений исходного кода
В тка про кт
Уведомление о новых изменениях
Про рка кода
Запуск сборки
А нт с орки
Результат сборки
Уведомление о результатах сборки
Про рка результата
Запуск статического анализа
Статический анализ
Результат анализа
Про рка кода
Запуск сборки
А нт с орки
Результат сборки
Уведомление о результатах
Про рка результата
Начало процедуры экспертизы исходного кода
Эксп ртиза исхо но о ко а про раммы
Результат
Про рка кода
Запуск сборки
А нт с орки
Результат сборки
Уведомление о результатах
Про рка результата
Запуск динамического анализа
Функциональное т стиро ани
Результат
Про рка кода
Уведомление о результатах
Про рка результата
Запуск фаззинг-тестирования
Фаззинг-т стиро ани
Результат
Про рка кода
Уведомление о результатах
Про рка результата
Запуск функционального тестирования
Динамич с кий анализ
Результат
Про рка кода
Уведомление о результатах
Про рка результата
Запуск тестирования на проникновение
Т стиро ани на проникно ни
Результат
Про рка кода
Загрузка собранного ПО
Дистри ути
Поставка Заказчику эксплуатационных документов
Поставка ПО
Систематический поиск уязвимостей программы
Информирование о найденных уязвимостях и выпуск обновлений

Приложение № 2

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЕДИНАЯ ЦИФРОВАЯ ПЛАТФОРМА РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ» ДЛЯ СОЗДАНИЯ, РАЗВИТИЯ И ЭКСПЛУАТАЦИИ ГОСУДАРСТВЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПО С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТОВ ЕДИНОЙ ЦИФРОВОЙ ПЛАТФОРМЫ РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ»

Рекомендованные требования по безопасности к разрабатываемому программному обеспечению в составе ГИС на Платформе «ГосТех»

2022 г.

Содержание

1. Требования к архитектуре и моделированию угроз

1.1. Требования к жизненному циклу безопасной разработки ПО

№	Описание	К3	К2	К1
1.1.1	Безопасность должна обеспечиваться на всех этапах жизненного цикла разработки ПО		x	x
1.1.2	Для каждого изменения архитектуры программы должно осуществляться моделирование угроз, планирование компенсирующих мер, вариантов реагирования на угрозы и разработка алгоритмов тестирования запланированных мер	x	x	x
1.1.3	Все пользовательские функции выполняют необходимые требования безопасности (разграничение действий по чтению, записи и исполнению)		x	x
1.1.4	Разработка документации с обоснованием всех границ доверия, ролевой модели доступа и фиксации событий аудита программы, ее компонентов и всех значимых потоков данных		x	x
1.1.5	Анализ безопасности высокоуровневой архитектуры программы и всех подключаемых удалённых служб		x	x
1.1.6	Реализация централизованных, надежных и переиспользуемых функций безопасности, разработанных с применением мер по разработке ПО		x	x
1.1.7	Наличие перечня обязательных требований по безопасному программированию, требований безопасности к разрабатываемому ПО, рекомендаций или политик для всех разработчиков и тестировщиков		x	x

1.2. Требования к архитектуре аутентификации

№	Описание	К3	К2	К1
1.2.1	Для всех компонентов приложений, служб и серверов должны использоваться уникальные или специальные учётные записи операционной системы с минимально необходимым уровнем привилегий		x	x
1.2.2	Взаимодействие между компонентами приложения, включая API-интерфейсы, промежуточное ПО и уровни данных, должно быть аутентифицировано. Компоненты		x	x

№	Описание	К3	К2	К1
	приложения должны иметь минимально необходимые привилегии			
1.2.3	Для аутентификации и авторизации приложений должен использоваться централизованный механизм аутентификации, обеспечивающий наличие всех необходимых функций безопасности		x	x
1.2.4	Должна быть исключена возможность понижения уровня аутентификации и авторизации всех взаимодействий приложения, включая API–интерфейсы, определенного в конфигурации ПО		x	x

1.3 Требования к архитектуре контроля доступа

№	Описание	К3	К2	К1
1.3.1	Должен обеспечиваться дополнительный контроль выполнения требований безопасности на ключевых интеграционных точках и точках управления (например, шлюзы управления доступом, серверы и бессерверные функции обеспечивают контроль доступа)		x	x
1.3.2	Для разграничения доступа к защищаемой информации и другим ресурсам должен использоваться централизованный механизм, обеспечивающий наличие всех необходимых функций безопасности. Доступ к данным должен предоставляться только по результатам применения такого механизма		x	x
1.3.3	Управление доступом должно осуществляться на основе ролевой модели доступа. На всех этапах жизненного цикла ролевой модели должны соблюдаться принципы: минимизации полномочий, разделения полномочий и взаимоисключения полномочий. Должен быть разработан набор типовых ролей доступа. При регистрации клиента или пользователя в системе ему необходимо предоставлять роль из перечня типовых ролей	x	x	x

1.4. Требования к архитектуре ввода-вывода

№	Описание	К3	К2	К1
1.4.1	Требования по вводу, выводу и обработке данных определяются, в зависимости от типа данных, степени их секретности и класса защищенности ГИС в соответствии с требованиями законодательства, правовых актов и иных документов		x	x
1.4.2	Сериализация не должна использоваться при взаимодействии с ненадежными клиентами. Если это невозможно, должны применяться дополнительные меры контроля целостности и подлинности (например, шифрование при отправке данных на недоверенном интервале взаимодействия), чтобы предотвратить атаки десериализации, включая внедрение объекта		x	x
1.4.3	Проверка ввода должна выполняться механизмами централизованной службы		x	x
1.4.4	Кодировка выходных данных происходит рядом с интерпретатором, для которого оно предназначено или им самим		x	x

1.5. Регистрация ошибок и ведение журнала

№	Описание	К3	К2	К1
1.5.1	В системе должен использоваться общий формат и подход к ведению журнала		x	x
1.5.2	Журналы безопасно передаются и хранятся в централизованной по отношению к приложению системе для анализа, обнаружения аномалий, оповещения и эскалации		x	x

1.6. Требования к архитектуре защиты данных

№	Описание	К3	К2	К1
1.6.1	Вся информация ограниченного доступа[3] должна быть идентифицирована и классифицирована по классам защищенности. Сформирован и поддерживается в актуальном состоянии реестр такой информации.		x	x
1.6.2	Для каждого класса защищенности должен быть сформирован связанный набор требований защиты, таких как требования к шифрованию, требования целостности и подлинности, хранения, конфиденциальности и другие. Перечень таких требований должен учитываться при разработке архитектуры		x	x

1.7. Требования к архитектуре связи

№	Описание	К3	К2	К1
1.7.1	Приложение должно шифровать взаимодействие на уровне сессии: если в нем передаются секреты в открытом виде; между компонентами, если эти компоненты находятся в разных контейнерах, системах, сайтах или облачных провайдерах		x	x
1.7.2	Компоненты приложения проверяют подлинность каждой стороны канала связи, чтобы предотвратить атаки типа «человек в середине». Например, компоненты приложения должны проверять сертификаты и цепочки TLS		x	x

1.8. Требования к реализации мониторинга событий разработки ПО

№	Описание	К3	К2	К1
1.8.1	Система управления исходным кодом должна иметь контроль доступа, обеспечивать однозначную идентификацию пользователей и фиксировать производимые ими действия, чтобы можно было отслеживать каждое изменение		x	x

[3] П. 2, ст. 5 Федерального закона от 27.06.2006 №149-ФЗ.

1.9. Требования к архитектуре бизнес-логики

№	Описание	К3	К2	К1
1.9.1	Все компоненты приложения определены и задокументированы с точки зрения бизнес-функций или функций безопасности, которые они предоставляют		x	x
1.9.2	Все важные потоки бизнес-логики, включая аутентификацию, управление сеансами и контроль доступа, не разделяют несинхронизированное состояние		x	x
1.9.3	Все важные потоки бизнес-логики, включая аутентификацию, управление сеансами и контроль доступа, являются потокобезопасными и устойчивы к Time-of-check to time-of-use ошибкам			x

1.10. Требования к архитектуре безопасной загрузки файлов

№	Описание	К3	К2	К1
1.10.1	Должна быть реализована политика безопасности загружаемого контента, для снижения риска XSS - векторов или других атак из загружаемых файлов. Загрузка файлов пользователем должна осуществляться с помощью octet - stream загрузки, либо из несвязанного домена, например, из облачного хранилища		x	x

1.11. Требования к архитектуре конфигурации

№	Описание	К3	К2	К1
1.11.1	Компоненты с разными уровнями доверия разделены с помощью чётко определенных средств управления безопасностью, правил брандмауэра, шлюзов API, обратных прокси-серверов, облачных групп безопасности (cloud-based security groups) или аналогичных механизмов		x	x
1.11.2	При запуске двоичных файлов на ненадёжных устройствах используются двоичные подписи, доверенные соединения и проверенные конечные точки		x	x
1.11.3	При сборке пайплайн (build pipeline) предупреждает об устаревших или небезопасных компонентах и принимает соответствующие меры		x	x

№	Описание	К3	К2	К1
1.11.4	Пайплайн содержит этап для автоматической сборки и проверки безопасного развёртывания приложения, особенно если инфраструктура приложения определяется программным обеспечением, например, скрипты запуска в облачной среде		x	x
1.11.5	Развертывание приложения осуществляется в песочнице, приложение контейнеризировано и/или изолированно на сетевом уровне, чтобы не допустить атаки на другие приложения, особенно когда они выполняют опасные действия, такие как десериализация		x	x
1.11.6	Приложение не использует неподдерживаемые, небезопасные или устаревшие клиентские технологии, такие как плагины NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL или клиентские Java-апплеты		x	x

2. Требования к аутентификации

2.1. Требования к проведению процедуры восстановления учетных данных

№	Описание	К3	К2	К1
2.1.1	Сгенерированный системой секрет начальной активации или восстановления не отправляется пользователю открытым текстом	x	x	x
2.1.2	Подсказки для проверки пароля или проверки подлинности на основе знаний (так называемые «секретные вопросы») должны быть исключены	x	x	x
2.1.3	Проверка пароля при восстановлении учетных данных не должна допускать возможность раскрытия пароля	x	x	x
2.1.4	Должно обеспечиваться удаление или отключение учетных записей по умолчанию (например, root, admin, sa и т.д.)	x	x	x
2.1.5	Должен обеспечиваться механизм усиления аутентификации путем использования криптографических аутентификаторов или одноразовых паролей (OTP) для каждого сеанса или выполнения одной транзакции			x
2.1.6	При изменении метода или уровня аутентификации должно обеспечиваться уведомление об этом пользователя	x	x	x
2.1.7	Забытый пароль и другие пути восстановления используют безопасный механизм восстановления, такой как OTP на	x	x	x

№	Описание	К3	К2	К1
	основе времени (TOTP) или другой программный токен, мобильное push-уведомление или другой механизм автономного восстановления			
2.1.8	При утрате или компрометации секрета однофакторной аутентификации, либо 1 из факторов многофакторной аутентификации, необходимо обеспечить проверку подлинности не ниже, чем при первоначальной регистрации		x	x

2.2. Требования к проведению процедуры служебной аутентификации

№	Описание	К3	К2	К1
2.2.1	Внутрисервисные секреты не зависят от неизменяемых учетных данных, таких как пароли, ключи API или общие учетные записи с привилегированным доступом		Средства ОС	HSM
2.2.2	Если для аутентификации службы необходимо использовать данные учетной записи (УЗ), не должна использоваться УЗ по умолчанию (например, в некоторых службах во время установки по умолчанию используются root или admin)		Средства ОС	HSM
2.2.3	Должна обеспечиваться невозможность компрометации пароля. Хранение пароля в конфигурационном файле в открытом виде должно быть запрещено		Средства ОС	HSM
2.2.4	Пароли, интеграция с базами данных и сторонними системами, исходные данные и внутренние секреты, а также ключи API управляются надежно и не включаются в исходный код или не хранятся в репозиториях исходного кода. Такое хранилище ДОЛЖНО противостоять атакам в автономном режиме. Для хранения паролей рекомендуется использовать защищенное хранилище программных ключей (К3), аппаратный TPM или HSM (К1)		Средства ОС	HSM

3. Требования к управлению сессиями (Session Management)

3.1. Основы безопасности управления сессии

№	Описание	К3	К2	К1
3.1.1	Приложение никогда не показывает токены сессии в параметрах URL	x	x	x

3.2. Требования к привязке к сессии

№	Описание	К3	К2	К1
3.2.1	Приложение генерирует новый токен сессии при аутентификации пользователя	x	x	x
3.2.3	Приложение должно хранить токены сессии только в браузере с использованием безопасных методов, таких как надлежащим образом защищенные файлы cookie или хранилище сеансов HTML 5	x	x	x

3.3. Требования к проведению завершения сессии

№	Описание	К3	К2	К1
3.3.1	Выход из системы и истечение срока действия делают токен сессии недействительным, так что кнопка возврата или нижестоящая проверяющая сторона не возобновляют аутентифицированный сеанс, в том числе между проверяющими сторонами	x	x	x
3.3.2	Если средства аутентификации позволяют пользователям оставаться в системе, убедитесь, что повторная аутентификация периодически выполняется как при активном использовании, так и после периода простоя	30 дней	12 часов или 30 минут бездействия, 2FA, опционально	12 часов или 15 минут бездействия с 2FA
3.3.3	Приложение предоставляет возможность завершить все другие активные сеансы после успешной смены пароля (включая изменение с		x	x

№	Описание	К3	К2	К1
	помощью сброса / восстановления пароля) во всем приложении при объединенном входе в систему (если присутствует) и любых проверяющих сторонах			
3.2.4	Пользователи могут просматривать и (повторно введя учетные данные для входа) выходить из любых или всех активных в данный момент сессий и устройств		x	x

3.4. Требования к управлению сессиями на основе Cookie-файлов

№	Описание	К3	К2	К1
3.4.1	Токены сессии на основе файлов cookie имеют установленный атрибут "Secure"	x	x	x
3.4.2	Токены сессии на основе файлов cookie имеют установленный атрибут "HttpOnly"	x	x	x
3.4.3	Токены на основе файлов cookie используют атрибут "Same Site" для снижения уязвимости атакам на подделку межсайтовых запросов	x	x	x
3.4.4	Токены сессии на основе файлов cookie используют префикс "Host-", поэтому файлы cookie отправляются только на хост, который изначально установил файл cookie	x	x	x
3.4.5	Если приложение размещено под доменным именем с другими приложениями, которые устанавливают или используют сеансовые файлы cookie, которые могут раскрывать сеансовые файлы cookie, задайте атрибут path в токенах сессии на основе файлов cookie, используя максимально точный возможный путь	x	x	x

3.5. Требования к процедуре управления сессиями на основе токенов

№	Описание	К3	К2	К1
3.5.1	Приложение позволяет пользователям отзывать токены OAuth, которые формируют доверительные отношения со связанными приложениями		x	x

№	Описание	К3	К2	К1
3.5.2	Приложение использует токены сессии, а не статические секреты и ключи API, за исключением устаревших реализаций		x	x
3.5.3	Токены сеанса без состояния используют цифровые подписи, шифрование и другие контрмеры для защиты от взлома, обхода, воспроизведения, нулевого шифрования и атак подмены ключей		x	x

3.6. Требования к интегрированному управлению аутентификацией

№	Описание	К3	К2	К1
3.6.1	Проверяющие стороны (RP) указывают максимальное время проверки подлинности Поставщикам услуг учётных данных (CSP). Поставщик учётных данных должен провести повторную аутентификацию пользователя при истечении времени проверки подлинности			x
3.6.2	Поставщики учетных данных (CSP) информируют Проверяющие стороны (RP) о времени последней аутентификации, чтобы позволить Проверяющей стороне определить, нужно ли повторно аутентифицировать пользователя			x

3.7. Требования к процессу защиты от эксплойтов в процессе управления сессиями

№	Описание	К3	К2	К1
3.7.1	Приложение обеспечивает полную, действительную сессию входа в систему или требует повторной аутентификации или вторичной проверки, прежде чем разрешить любые транзакции или изменения учетной записи	x	x	x

4. Требования к контролю доступа

4.1. Требования к общей архитектуре контроля доступа

№	Описание	К3	К2	К1
4.1.1	Приложение применяет правила контроля доступа на уровне доверенной службы, особенно если присутствует контроль доступа на стороне клиента, который можно обойти	x	x	x
4.1.2	Просмотр каталогов недоступен, если нет специального разрешения. Кроме того, приложения не должны разрешать обнаружение или раскрытие метаданных файлов или каталогов, таких как Thumbs.db, .DS_Store, .git или .svn	x	x	x
4.1.3	Конечные пользователи не могут изменять все атрибуты пользователя и данные, а также информацию о политике, используемой средствами контроля доступа, без специального разрешения	x	x	x
4.1.4	Существует принцип наименьших привилегий - пользователи должны иметь доступ только к функциям, файлам данных, URL-адресам, контроллерам, службам и другим ресурсам, для которых у них определены полномочия. Это подразумевает защиту от подделки и превышения привилегий	x	x	x
4.1.5	Средства управления доступом надежно работают при сбое, в том числе при возникновении исключения	x	x	x

4.2. Требования к контролю доступа на эксплуатационном уровне

№	Описание	К3	К2	К1
4.2.1	Конфиденциальные данные и API защищены от IDOR-атак, направленных на создание, чтение, обновление и удаление записей, таких как создание или обновление чужой записи, просмотр записей каждого пользователя или удаление всех записей	x	x	x
4.2.2	Приложение или платформа применяет надежный механизм защиты от межсайтовой подделки запроса (CSRF) для защиты аутентифицированных функций, а эффективная защита от автоматизации или защита от межсайтовой подделки запроса защищает неаутентифицированные функции	x	x	x

5. Требования к валидации, очистке и кодированию

5.1. Требования к проверке входных данных

№	Описание	К3	К2	К1
5.1.1	Приложение защищено от атак, связанных с изменением параметров HTTP, особенно если платформа приложения не делает различий в источнике параметров запроса (GET, POST, cookies, заголовки или переменные среды)	x	x	x
5.1.2	Фреймворки защищают от атак массового присвоения параметров или что приложение имеет контрмеры для защиты от небезопасного присвоения параметров, такие как пометка полей частными или аналогичными	x	x	x
5.1.3	Все входные данные (поля HTML-формы, запросы REST, параметры URL, заголовки HTTP, файлы cookie, пакетные файлы, RSS-каналы и т.д.) проверены с помощью положительной валидации (списка разрешений)	x	x	x
5.1.4	Структурированные данные строго типизированы и проверены на соответствие определенной схеме, включая разрешенные символы, длину и шаблон или проверено, что 2 связанных поля совместимы	x	x	x
5.1.5	Перенаправление и пересылка URL-адресов разрешается только на доверенные адреса, которые отображаются в списке разрешенных, или показывается предупреждение при перенаправлении на потенциально ненадёжный ресурс	x	x	x

5.2. Требования к очистке и изоляции вводимых данных

№	Описание	К3	К2	К1
5.2.1	Весь ненадежный ввод HTML из редакторов WYSIWYG или аналогичных должен быть очищен с помощью библиотеки или фреймворка HTML sanitizer	x	x	x
5.2.2	Неструктурированные данные очищены для соблюдения мер безопасности, таких как допустимые символы и длина	x	x	x
5.2.3	Приложение очищает вводимые пользователем данные перед передачей в почтовые системы для защиты от внедрения SMTP или IMAP	x	x	x

№	Описание	К3	К2	К1
5.2.4	Приложение избегает использования eval() или других функций динамического выполнения кода. Там, где альтернативы нет, любой пользовательский ввод должен быть очищен или изолирован перед выполнением	x	x	x
5.2.5	Приложение защищает от атак с внедрением шаблонов, гарантируя, что любой пользовательский ввод очищен или изолирован	x	x	x
5.2.6	Приложение защищает от атак SSRF, проверяя или очищая ненадежные данные или метаданные HTTP-файлов, такие как имена файлов и поля ввода URL-адресов, и использует разрешенные списки протоколов, доменов, путей и портов	x	x	x
5.2.7	Приложение очищает, отключает или изолирует предоставляемый пользователем контент для сценариев масштабируемой векторной графики (SVG), особенно если они относятся к XSS, полученным в результате встроенных сценариев, и foreignObject	x	x	x
5.2.8	Приложение очищает, отключает или изолирует содержимое, предоставляемое пользователем на языке скриптов или шаблонов выражений, такое как Markdown, таблицы стилей CSS или XSL, BBCode или аналогичное	x	x	x

5.3. Требования к кодированию выходных данных и предотвращению заражения

№	Описание	К3	К2	К1
5.3.1	Кодировка вывода соответствует требуемому интерпретатору и контексту. Например, используйте кодировщики специально для значений HTML, атрибутов HTML, JavaScript, параметров URL, заголовков HTTP, SMTP и других, как того требует контекст, особенно для ненадежных входных данных (например, имен с Юникод или апострофом, таких как ♘♘ или O'Хара).	x	x	x
5.3.2	Кодировка вывода сохраняет выбранный пользователем набор символов и языковой стандарт, так что любая точка символа Юникод важна и безопасно обрабатывается	x	x	x
5.3.3	Для защиты от отраженных, сохраненных и основанных на DOM XSS должно быть реализовано автоматизированное контекстно зависимое экранирование вывода	x	x	x

№	Описание	К3	К2	К1
5.3.4	Запросы к базе данных (например, SQL, HQL, ORM, NoSQL) должны быть защищены от атак типа «Внедрение SQL-кода» при помощи параметризованных запросов, ORM, entity framework и т.д.	x	x	x
5.3.5	При отсутствии параметризованных запросов или более безопасных механизмов запросов к базе данных для защиты от атак типа «Внедрение SQL-кода» используется контекстно зависимая кодировка вывода, например экранирование SQL	x	x	x
5.3.6	Приложение защищает от атак с использованием JSON, атак с использованием JSON eval и вычисления выражений JavaScript	x	x	x
5.3.7	Приложение защищает от уязвимостей, связанных с внедрением LDAP, или реализованы специальные средства безопасности для предотвращения внедрения LDAP	x	x	x
5.3.8	Приложение защищает от внедрения команд операционной системы, вызовы операционной системы используют параметризованные запросы операционной системы или используют контекстную кодировку вывода командной строки	x	x	x
5.3.9	Приложение защищает от атак локального включения файлов (LFI) или удаленного включения файлов (RFI)	x	x	x
5.3.10	Приложение защищает от атак с использованием XPath или XML-инъекций	x	x	x

5.4. Требование к памяти, срокам и неуправляемому коду

№	Описание	К3	К2	К1
5.4.1	Приложение использует безопасную для памяти строку, безопасное копирование в память и арифметику указателей для обнаружения или предотвращения переполнения стека, буфера или кучи		x	x
5.4.2	Форматирующие строки не принимают потенциально опасные входные данные и являются постоянными		x	x
5.4.3	Для предотвращения переполнения целочисленной переменной чисел должны использоваться методы проверки входных данных (знака и диапазона)		x	x

5.5. Требования к процессу предотвращения десериализации

№	Описание	К3	К2	К1
5.5.1	Сериализованные объекты используют проверки целостности или зашифрованы для предотвращения создания враждебных объектов или подделки данных	x	x	x
5.5.2	Приложение ограничивает синтаксические анализаторы XML использованием только максимально возможной конфигурации и гарантирует, что небезопасные функции, такие как разрешение внешних объектов, отключены для предотвращения атак XML eXternal Entity (XXE)	x	x	x
5.5.3	Десериализация ненадежных данных предотвращена или защищена как в пользовательском коде, так и в сторонних библиотеках (таких как анализаторы JSON, XML и YAML)	x	x	x
5.5.4	При синтаксическом анализе JSON в браузерах или серверных частях, работающих на JavaScript, для синтаксического анализа документа JSON используется JSON.parse. Функция eval() не применяется для синтаксического анализа JSON	x	x	x

6. Требования к обработке ошибок и ведению журнала

6.1. Требования к содержимому журнала

№	Описание	К3	К2	К1
6.1.1	Приложение не регистрирует учётные данные или платежные реквизиты. Токены сессии должны храниться в журналах только в необратимой хэшированной форме	x	x	x
6.1.2	Приложение не регистрирует иную информацию ограниченного доступа кроме, определенной в соответствии с законодательством Российской Федерации или политикой безопасности	x	x	x
6.1.3	Приложение регистрирует события, относящиеся к безопасности, включая события успешной и неудачной аутентификации, сбои управления доступом, сбои десериализации и сбои проверки ввода		x	x
6.1.4	Каждое событие журнала содержит необходимую информацию, которая позволила бы детально изучить, когда происходит событие		x	x

6.2. Требования к обработке журнала

№	Описание	К3	К2	К1
6.2.1	Все инциденты аутентификации регистрируются без сохранения токенов сессии или паролей. Инцидент должен быть зарегистрирован с соответствующими метаданными, необходимыми для расследований в сфере безопасности		x	x
6.2.2	Все действия по управлению доступом и все неудачные попытки должны быть записаны в журнал		x	x

6.3. Требования к защите журнала

№	Описание	К3	К2	К1
6.3.1	Все компоненты ведения журнала надлежащим образом кодируют данные, чтобы предотвратить проникновение в журнал		x	x
6.3.2	Журналы безопасности защищены от несанкционированного доступа и изменения		x	x
6.3.3	Все источники времени должны быть синхронизированы с правильным временем и часовым поясом. Рекомендуется регистрироваться только в UTC, если системы являются глобальными, чтобы помочь в проведении криминалистического анализа после инцидента		x	x

6.4. Требования к обработке ошибок

№	Описание	К3	К2	К1
6.4.1	При возникновении непредвиденной ошибки или ошибки, связанной с безопасностью, отображается общее сообщение, возможно, с уникальным идентификатором, который сотрудники службы поддержки могут использовать для расследования	x	x	x
6.4.2	Обработка исключений (или функциональный эквивалент) используется во всей кодовой базе для учета ожидаемых и непредвиденных условий ошибки		x	x
6.4.3	Определён целевой обработчик ошибок последнего средства, который будет перехватывать все необработанные исключения		x	x

7. Требования к защите данных

7.1. Общие требования к защите данных

№	Описание	К3	К2	К1
7.1.1	Приложение защищает данные от кэширования в серверных компонентах, таких как балансировщики нагрузки и кэши приложений		x	x
7.1.2	Все кэшированные или временные копии данных, хранящиеся на сервере, защищены от несанкционированного доступа или удалены / аннулированы после того, как авторизованный пользователь получит доступ к данным		x	x
7.1.3	Приложение сводит к минимуму количество параметров в запросе, таких как скрытые поля, переменные Ajax, файлы cookie и значения заголовка		x	x
7.1.4	Приложение может обнаруживать и предупреждать пользователя о ненормальном количестве запросов, например, по IP, общему количеству за час или день, или о том, что имеет смысл для приложения		x	x
7.1.5	Выполняется регулярное создание резервных копий данных и тестовое восстановление данных			x
7.1.6	Хранение резервных копий должно осуществляться достаточно надёжно, чтобы предотвратить кражу или повреждение данных			x

7.2. Требования к передаче и обработке информации ограниченного доступа

№	Описание	К3	К2	К1
7.2.1	Приложение устанавливает заголовки для защиты от кэширования, чтобы информация ограниченного доступа не кэшировалась в современных браузерах	x	x	x
7.2.2	Данные, хранящиеся в хранилище браузера (например, localStorage, sessionStorage, IndexedDB или cookies), не содержат информации ограниченного доступа	x	x	x
7.2.3	Аутентифицированные данные удаляются из клиентского хранилища, такого как DOM браузера, после завершения работы клиента или окончания сеанса	x	x	x

7.3. Требования к информации ограниченного доступа (персональные данные)

№	Описание	К3	К2	К1
7.3.1	Информация ограниченного доступа отправляется на сервер в теле или заголовках HTTP – сообщения. Параметры строки запроса любого HTTP - глагола не содержат информации ограниченного доступа	x	x	x
7.3.2	У пользователей есть способ удалять или экспортировать свои данные по требованию	x	x	x
7.3.3	Пользователям предоставлена четкая формулировка относительно сбора и использования предоставленных персональных данных, пользователи дали согласие на использование данных перед любой обработкой	x	x	x
7.3.4	Вся информация ограниченного доступа, созданная и обрабатываемая приложением, быть идентифицирована и определена как защищаемая, а правила работы с такой информацией документированы	x	x	x
7.3.5	Доступ к информации ограниченного доступа проверяется (без регистрации информации), если информация собираются в соответствии с соответствующими указаниями по защите данных, или когда требуется регистрация доступа		x	x
7.3.6	Информация ограниченного доступа, содержащаяся в памяти, перезаписывается, как только она больше не требуется для предотвращения атак сброса памяти с использованием нулей или случайных данных		x	x

8. Требования к каналам связи

8.1. Общие требования к коммуникации

№	Описание	К3	К2	К1
8.1.1	Протокол TLS используется для всех подключений клиентов и не возвращается к незащищенным или незашифрованным сообщениям	x	x	x
8.1.2	Убедиться, используя современные средства тестирования TLS, что используются только надежные наборы шифров, причем предпочтительными являются самые надежные наборы шифров	x	x	x

№	Описание	К3	К2	К1
8.1.3	Используются только последние рекомендуемые версии протокола TLS, такие как TLS 1.2 и TLS 1.3. Предпочтительным вариантом должна быть последняя версия протокола TLS	x	x	x

9. Требования к защите от вредоносного воздействия

9.1. Требования к целостности кода

№	Описание	К3	К2	К1
9.1.1	Используется инструмент анализа кода, который может обнаруживать потенциально вредоносный код, такой как временные функции, небезопасные файловые операции и сетевые подключения			x

9.2. Требования к поиску вредоносного кода

№	Описание	К3	К2	К1
9.2.1	Исходный код приложения и библиотеки сторонних производителей не содержат недекларированных возможностей для сбора данных. Там, где такая функциональность существует, получить разрешение пользователя на ее работу, прежде чем собирать какие-либо данные	x	x	x
9.2.2	Приложение не запрашивает ненужных или чрезмерных разрешений для функций или датчиков, связанных с информацией ограниченного доступа, таких как контакты, камеры, микрофоны или местоположение		x	x
9.2.3	Исходный код приложения и библиотеки сторонних производителей не содержат скрытых возможностей, таких как жестко запрограммированные или дополнительные недокументированные учетные записи или ключи, запутывание кода, недокументированные двоичные объекты, руткиты или средства защиты от отладки, небезопасные функции отладки или другие устаревшие, небезопасные или скрытые функции, которые могут быть использованы злонамеренно, если будут обнаружены			x

№	Описание	К3	К2	К1
9.2.4	Исходный код приложения и сторонние библиотеки не содержат каких-либо недокументированных функционалов.			x

9.3. Требования к целостности приложения

№	Описание	К3	К2	К1
9.3.1	Если в приложении есть функция автоматического обновления клиента или сервера, обновления должны быть получены по защищенным каналам и подписаны электронной подписью. Код обновления должен проверять цифровую подпись обновления перед установкой или выполнением обновления	x	x	x
9.3.2	В приложении используются средства защиты целостности, такие как подпись кода или функции контроля целостности вложенных ресурсов. Приложение не должно загружать или выполнять код из ненадежных источников, таких как загрузка компонентов, модулей, плагинов, кода или библиотек из ненадежных источников или сети Интернет	x	x	x
9.3.3	Приложение защищено от захвата поддоменов, если приложение использует записи DNS или поддомены DNS, такие как доменные имена с истекшим сроком действия, устаревшие указатели DNS или CNAMEs, истекшие проекты в репозиториях с открытым исходным кодом или временные облачные API, бессерверные функции или хранилища и т.д.	x	x	x

10. Требования к бизнес-логике

10.1. Требования к процедуре обеспечения безопасности бизнес-логики

№	Описание	К3	К2	К1
10.1.1	Приложение будет обрабатывать потоки бизнес-логики только для 1 и того же пользователя в последовательном порядке шагов и без их пропуска	x	x	x
10.1.2	Приложение должно обрабатывать только потоки бизнес - логики, при этом все этапы обрабатываются в реальном времени и последовательно, т.е. порядок выполнения транзакций не должен нарушаться	x	x	x

10.1.3	Приложение имеет ограничения для конкретных бизнес-действий или транзакций, которые правильно применяются для каждого пользователя	x	x	x
10.1.4	Приложение имеет средства защиты от автоматизации для защиты от чрезмерных вызовов, таких как массовая фильтрация данных, запросы бизнес-логики, загрузка файлов или атаки типа «отказ в обслуживании»	x	x	x
10.1.5	Приложение имеет ограничения бизнес-логики или проверки для защиты от вероятных бизнес-рисков или угроз, выявленных с помощью моделирования угроз или аналогичных методологий	x	x	x
10.1.6	Приложение не имеет проблем типа «времени проверки до времени использования» (TOCTOU) или других из-за ускорения операций		x	x
10.1.7	Приложение отслеживает необычные события или действия с точки зрения бизнес-логики. Например, попытки выполнить действия не по порядку или действия, которые обычный пользователь никогда бы не предпринял		x	x
10.1.8	Приложение имеет настраиваемое оповещение при обнаружении автоматических атак или необычной активности		x	x

11. Требования к файлам и ресурсам

11.1. Требования к передаче файлов

№	Описание	К3	К2	К1
11.1.1	Приложение не должно принимать большие файлы, которые могут заполнить хранилище или вызвать отказ в обслуживании	x	x	x
11.1.2	Приложение проверяет сжатые файлы (например, zip, gz, docx, odt) на соответствие максимально допустимому несжатому размеру и максимальному количеству файлов, прежде чем распаковывать файл		x	x
11.1.3	Установлена квота на размер файла и максимальное количество файлов на пользователя, чтобы гарантировать, что один пользователь не сможет заполнить хранилище		x	x

11.2. Требования к целостности файла

№	Описание	К3	К2	К1
11.2.1	Файлы, полученные из ненадежных источников, проверены на соответствие ожидаемому типу на основе содержимого файла		x	x

11.3. Требования к выполнению файла

№	Описание	К3	К2	К1
11.3.1	Метаданные пользователя не используются в имени файла на файловой системе, и для URL API реализована защита от атаки обхода пути	x	x	x
11.3.2	Именование файлов на файловой системе, содержащее в своей составе элементы метаданных пользователя, провалидировано и построено так, чтобы предотвратить раскрытие, создание, обновление или удаление локальных файлов (LFI)	x	x	x
11.3.3	Именование файлов на файловой системе, содержащее в своей составе элементы метаданных пользователя, провалидировано и построено так, чтобы предотвратить раскрытие или выполнение удаленных файлов с помощью атак удаленного включения файлов (RFI) или подделки запросов на стороне сервера (SSRF)	x	x	x
11.3.4	Приложение защищает от отражающей загрузки файлов (RFD), проверяя или игнорируя отправленные пользователем имена файлов в параметре JSON, JSONP или URL, заголовок типа содержимого ответа должен быть установлен в text/plain, а заголовок Content-Disposition должен иметь фиксированное имя файла	x	x	x
11.3.5	Реализован механизм защиты от атаки внедрения команд ОС	x	x	x
11.3.6	Приложение не включает и не выполняет функции из ненадежных источников, таких как непроверенные сети распространения контента, библиотеки JavaScript, библиотеки node npm или библиотеки DLL на стороне сервера		x	x

11.4. Требования к хранилищу файлов

№	Описание	К3	К2	К1
11.4.1	Файлы, полученные из ненадежных источников, хранятся вне корневого каталога веб-сайта с ограниченными разрешениями	x	x	x
11.4.2	Файлы, полученные из ненадежных источников, сканируются антивирусными сканерами, чтобы предотвратить загрузку и распространение вредоносного контента	x	x	x

11.5. Требования к скачиванию файлов

№	Описание	К3	К2	К1
11.5.1	Веб-уровень настроен на обслуживание только файлов с определенными расширениями, чтобы предотвратить непреднамеренную утечку информации и исходного кода. Например, файлы резервных копий (например, .bak), временные рабочие файлы (например, .swp), сжатые файлы (.zip, .tar.gz и т.д.) и другие расширения, обычно используемые редакторами, должны быть заблокированы, если они не требуются	x	x	x
11.5.2	Прямые запросы к загруженным файлам не должны выполняться как содержимое HTML/JavaScript	x	x	x

11.6. Требования к защите от SSRF

№	Описание	К3	К2	К1
11.6.1	Веб-сервер или сервер приложения настроен с разрешенным списком ресурсов или систем, на которые сервер может отправлять запросы или загружать данные /файлы	x	x	x

12. Требования к API и веб-сервисам

12.1. Общие требования к безопасности веб-сервисов

№	Описание	К3	К2	К1
12.1.1	Все компоненты приложения используют одни и те же кодировки и синтаксические анализаторы, чтобы	x	x	x

№	Описание	К3	К2	К1
	избежать атак синтаксического анализа, которые используют различные URI или поведение синтаксического анализа файлов, которые могут быть использованы в SSRF и RFI атаках			
12.1.2	URL-адреса API не раскрывают информацию, такую как ключи API, токены сеанса и т.д.		x	x
12.1.3	Решения об авторизации принимаются как в URI, принудительно с помощью программной или декларативной безопасности на контроллере или маршрутизаторе, так и на уровне ресурсов, принудительно с помощью разрешений на основе модели	x	x	x
12.1.4	Запросы, содержащие неожиданные или отсутствующие типы содержимого, отклоняются с соответствующими заголовками (статус ответа HTTP 406 Неприемлемый или 415 Неподдерживаемый тип носителя)		x	x

12.2 Требования к REST-реализации

№	Описание	К3	К2	К1
12.2.1	Включенные методы RESTful HTTP являются допустимым выбором для пользователя или действия, например, запрещают обычным пользователям использовать DELETE или PUT on protected API или ресурсы	x	x	x
12.2.2	Проверка схемы JSON выполнена перед принятием входных данных	x	x	x
12.2.3	Веб-службы REST, использующие файлы cookie, защищены от подделки межсайтовых запросов с помощью, по крайней мере, 1 или нескольких из следующих способов: двойной шаблон отправки файлов cookie, одноразовые номера CSRF или проверка заголовка исходного запроса	x	x	x
12.2.4	Службы REST явно проверяют ожидаемый тип входящего содержимого, например application/xml или application/json		x	x
12.2.5	Заголовки сообщений и полезная нагрузка являются надежными и не изменяются при передаче. Требование надежного шифрования для транспорта (только TLS) может быть достаточным во многих случаях, поскольку		x	x

№	Описание	К3	К2	К1
	оно обеспечивает как конфиденциальность, так и целостность. Цифровые подписи для каждого сообщения могут обеспечить дополнительную гарантию в дополнение к защите при транспортировке для приложений с высоким уровнем безопасности, но влекут за собой дополнительные сложности и риски, которые необходимо сопоставить с преимуществами			

12.3. Требования к SOAP-реализации

№	Описание	К3	К2	К1
12.3.1	Проверка схемы XSD выполняется для обеспечения правильного формирования XML-документа, за которым следует проверка каждого поля ввода перед любой обработкой этих данных	x	x	x
12.3.2	Полезная нагрузка сообщения подписана с использованием WS-Security, чтобы обеспечить надежную передачу между клиентом и службой		x	x

12.4. Требования к реализации GraphQL

№	Описание	К3	К2	К1
12.4.1	Список разрешенных запросов или комбинация ограничения глубины и ограничения объема используются для предотвращения отказа в обслуживании GraphQL или DoS в результате потока запросов. Для более сложных сценариев следует использовать анализ затрат на запрос		x	x
12.4.2	Логика авторизации GraphQL или другого уровня данных должна быть реализована на уровне бизнес-логики вместо уровня GraphQL		x	x

13. Требования к конфигурации

13.1. Требования к сборке и развертыванию

№	Описание	К3	К2	К1
13.1.1	Настроен автоматизированный конвейер сборки и развертывания CI/CD. Осуществляется автоматическое		x	x

№	Описание	К3	К2	К1
	управление конфигурациями и сценариями автоматического развертывания			
13.1.2	Флаги компилятора настроены для включения всех доступных средств защиты и предупреждений о переполнении буфера, включая рандомизацию стека, предотвращение выполнения данных и прерывание сборки при обнаружении небезопасных операций с указателем, памятью, строкой формата, целым числом или строкой		x	x
13.1.3	Конфигурация сервера защищена в соответствии с рекомендациями используемого сервера приложений и фреймворков		x	x
13.1.4	Приложение, конфигурация и все зависимости могут быть повторно развернуты с помощью сценариев автоматического развертывания, созданы из документированного и протестированного журнала выполнения в разумные сроки или своевременно восстановлены из резервных копий		x	x
13.1.5	Авторизованные администраторы могут проверять целостность всех конфигураций, связанных с безопасностью, для обнаружения несанкционированного доступа			x

13.2. Требования к зависимостям

№	Описание	К3	К2	К1
13.2.1	Все компоненты обновлены, предпочтительно с помощью средства проверки зависимостей во время сборки или компиляции	x	x	x
13.2.2	Все ненужные функции, документация, примеры приложений и конфигураций удалены	x	x	x
13.2.3	Если ресурсы приложения, такие как библиотеки JavaScript, CSS или веб-шрифты, размещены внутри сети распространения контента (CDN) или у внешнего поставщика, для проверки целостности ресурса используется Subresource Integrity (SRI)	x	x	x
13.2.4	Сторонние компоненты поступают из заранее определенных, надежных и постоянно поддерживаемых репозиториев		x	x

13.2.5	Поддерживается спецификация ПО для всех используемых сторонних библиотек		x	x
13.2.6	Поверхность атаки уменьшена путем изолирования или инкапсуляции сторонних библиотек, чтобы предоставить приложению только требуемое поведение		x	x

13.3. Действия по предотвращению непреднамеренного раскрытия информации о безопасности

№	Описание	К3	К2	К1
13.3.1	Режимы отладки веб-сервера или сервера приложений и платформы приложений отключены в рабочей среде, чтобы исключить функции отладки, консоли разработчика и непреднамеренное раскрытие информации о безопасности	x	x	x
13.3.2	Заголовки HTTP или любая часть HTTP-ответа не содержат подробной информации о версии системных компонентов	x	x	x

13.4. Требования к безопасности заголовков HTTP

№	Описание	К3	К2	К1
13.4.1	Каждый HTTP-ответ содержит заголовок типа содержимого. Также необходимо указать безопасный набор символов (например, UTF-8, ISO-8859-1), если типами содержимого являются text/*, /+xml и application/xml. Содержимое должно совпадать с предоставленным заголовком типа содержимого	x	x	x
13.4.2	Все ответы API содержат заголовок, соответствующий требованиям политики безопасности Content-Disposition: attachment; filename="api.json" (или другое подходящее имя файла для типа содержимого)	x	x	x
13.4.3	Установлен заголовок ответа CSP, который помогает снизить воздействие XSS-атак, таких как уязвимости HTML, DOM, JSON и JavaScript	x	x	x
13.4.4	Все ответы содержат заголовок X-Content-Type-Options: nosniff	x	x	x
13.4.5	Заголовок, соответствующий требованиям политики безопасности, включен во все ответы и для всех	x	x	X

	поддоменов, таких как Strict-Transport-Security: max- age=15724800; includeSubDomains			
13.4.6	Включен подходящий заголовок, соответствующий требованиям политики безопасности, чтобы избежать раскрытия информации ограниченного доступа в URL-адресе через передачу ссылки ненадежным сторонам	x	x	x
13.4.7	Содержимое веб-приложения по умолчанию не может быть встроено на сторонний сайт и что встраивание точных ресурсов разрешено только там, где это необходимо, с помощью подходящих заголовков, отвечающих требованиям политики безопасности: frame-ancestors и X-Frame-Options	x	x	x

13.5. Требования к проверке заголовка HTTP-запроса

№	Описание	К3	К2	К1
13.5.1	Сервер приложений принимает только методы HTTP, используемые приложением /API, включая параметры подготовки и регистрирует / предупреждает о любых запросах, которые недопустимы для контекста приложения	x	x	x
13.5.2	Предоставленный исходный заголовок не используется для принятия решений о проверке подлинности или контроле доступа, поскольку исходный заголовок может быть легко изменен нарушителем	x	x	x
13.5.3	Заголовок Access-Control-Allow- Origin для совместного использования ресурсов (CORS) использует строгий разрешенный список доверенных доменов и поддоменов для сопоставления и не поддерживает "нулевое" происхождение	x	x	x
13.5.4	HTTP-заголовки, добавленные доверенным прокси-сервером или устройствами единого входа, такими как bearer-токен, аутентифицируются приложением		x	x

Приложение № 3

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЕДИНАЯ ЦИФРОВАЯ ПЛАТФОРМА РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ» ДЛЯ СОЗДАНИЯ, РАЗВИТИЯ И ЭКСПЛУАТАЦИИ ГОСУДАРСТВЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПО С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТОВ ЕДИНОЙ ЦИФРОВОЙ ПЛАТФОРМЫ РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ»

Пример правил и рекомендаций оформления исходного кода

2022 г.

Язык программирования C#

1. Именование переменных

1.1. Pascal casing

Описываются имена:

• всех определений типов, в том числе пользовательских классов, перечислений, событий, делегатов и структур;
• значения перечислений;
• readonly полей и констант;
• интерфейсов;
• методов;
• пространств имен (namespace);
• свойств;
• публичных полей;

1.2. Camel casing

Описываются имена:

• локальных переменных;
• аргументов методов;
• защищенных (protected) полей.

1.3. Суффиксы и префиксы

Применяются следующие суффиксы и префиксы:

• имена пользовательских классов исключений всегда заканчиваются суффиксом Exception;
• имена интерфейсов всегда начинаются с префикса «I»;
• имена пользовательских атрибутов всегда заканчиваются суффиксом «Attribute»;
• имена делегатов обработчиков событий всегда оканчиваются суффиксом EventHandler, имена классов-наследников от EventArgs всегда заканчиваются суффиксом EventArgs.

1.4. Аббревиатуры

При использовании аббревиатур в именах, капитализации подлежат аббревиатуры с двумя символами, в остальных аббревиатурах необходимо приводить к верхнему регистру только первый символ.

Например: namespace Sample.IO, class HttpUtil

2. Именование методов

Используйте конструкцию «глагол-объект» для именования методов. В частном случае, для методов, которые возвращают значение, используйте в паре «глагол-объект» для глагола Get, а для объекта – описание возвращаемого значения.

Например: ShowUserID(), GetUserId()

3. Именование переменных, полей и свойств.

При именовании переменных избегайте использования сокращенных вариантов, не отражающих содержание переменной (например, n вместо number). Не используйте венгерскую нотацию или используйте ее только для закрытых членов. Для имен элементов управления и локальных переменных

сложных типов указывайте префиксы, описывающие тип элемента. Например: txtSample.
Имена закрытых полей всегда начинаются с префикса «_» остальная часть имени описывается с помощью Camel Casing.
Например: private int _samplePrivateField;
Не используйте публичных или защищенных полей, вместо этого используйте свойства. Используйте автоматические свойства. Всегда указывайте модификатор доступа private, даже если разрешено его опускать. Всегда инициализируйте переменные, даже когда существует автоматическая инициализация.
4. Оформление кода
Используйте пустую строку между логическими секциями в исходном файле, классе, методе.
Располагайте открывающие и закрывающие фигурные скобки на новой строке.
Используйте фигурные скобки для выражения if, даже когда в выражение входит только одна строка кода.
Используйте промежуточную переменную для передачи bool-значения результата функции в условное выражение if;
Например:

```
bool boolVariable = GetBoolValue();
if (boolVariable)
{
}
```

Ставьте пробелы вокруг операторов (= +-*/) и после запятой.
5. Объем кода
Избегайте файлов с более чем 500 строками кода.
Избегайте методов с более чем 200 строками кода.
Избегайте методов с более чем 5 аргументами, используйте структуры для передачи большого числа параметров.
Одна строка кода не должна иметь длину более 120 символов.

Язык программирования JavaScript

1. Именование переменных
Для имен идентификаторов (переменных и функций) используйте camelCase. Для глобальных переменных используйте верхний регистр. Для констант (например, PI) используйте верхний регистр. При именовании переменных избегайте использования сокращенных вариантов, не отражающих содержание переменной.
2. Оформление логических операций
Используйте пустую строку между логическими секциями в исходном файле, классе, методе. Используйте промежуточную переменную для передачи bool-значения результата функции в условное выражение if. Например:

```
bool boolVariable = GetBoolValue();
```

```
if (boolVariable)
{
}
```

3. Оформление операторов

Всегда заканчивайте простой оператор точкой с запятой.

Правила для сложных операторов:

• поместите открывающую скобку в конец первой линии;
• используйте один пробел перед открывающей скобкой;
• поместите закрывающую скобку на новую линию без пробелов;
• не заканчивайте сложный оператор точкой с запятой.

Например:

Функция:

```
function divideByTwo(number) {
   return (number / 2);
}
```

Петля:

```
for (i = 0; i <2; i++) {
   x += i;
}
```

Условие:

```
if (time <5) {
   mark = "5";
} else {
   mark = "4";
}
```

4. Общие правила для определений объектов:

• поместите открывающую скобку на ту же линию, что и имя объекта;
• используйте двоеточие плюс один пробел между каждым свойством и его значением;
• используйте кавычки вокруг строковых значений, а не вокруг числовых значений;
• не добавляйте запятую после последней пары значение свойства;
• поместите закрывающую скобку на новую линию без пробелов;
• всегда заканчивайте определение объекта точкой с запятой.

Пример:

```
var person = {
   firstName: "John",
   lastName: "Doe",
   age: 50,
   eyeColor: "blue"
};
```

Короткие объекты могут быть записаны сжатыми, на одной линии, используя пробелы только между свойствами, как это:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

5. Отступы

Ставьте пробелы вокруг операторов (= +-*/) и после запятой. Используйте 4 пробела для отступа блоков кода. Не используйте вкладки (таблицы) для отступа.

6. Объем кода

Избегайте строк длиной более 80 символов.

Если инструкция JavaScript не умещается на одной строке, нужно разбить ее после оператора или запятой.

Например:

```
document.getElementById("demo").innerHTML =
    "Hello Dolly.";
```

7. Загрузка JavaScript в HTML

Используйте простой синтаксис для загрузки внешних скриптов (атрибут Type не нужен):

```
<script src="myscript.js"></script>
```

Язык Java

1. Именование переменных, классов и интерфейсов

Тип идентификатора	Правила для именования	Примеры
Классы	Имена классов должны быть существительными, набранным в смешанном регистре (каждое слово – с большой буквы). Постарайтесь, чтобы ваши имена классов были простыми и наглядными. Используйте целые слова – избегайте сокращений и аббревиатур (если только аббревиатура делает имя класса более наглядным и понятным, чем длинная форма, например такие как URL или HTML)	class Raster; class ImageSprite;
Интерфейсы	Имена интерфейсов должны начинаться с заглавных букв и именоваться согласно тем же правилам	interface RasterDelegate; interface Storing;

Методы	Методы должны быть глаголами, набранными в смешанном регистре, первая буква первого слова в нижнем регистре, у всех последующих слов первая буква в верхнем регистре	run();
		runFast();
		getBackground();

Переменные	За исключением переменных, все экземпляры классов и констант классов	int i;

	набираются в смешанном регистре с первым символом в нижнем регистре. Последующие слова набираются с большой буквы. Имена переменных должны быть короткими, но имеющими смысл. Выбранное имя переменной должно быть запоминающееся – то есть кратко описывающим то, что в ней содержится. Односимвольных имен переменных следует избегать, за исключением временных «одноразовых» переменных. Общепринятыми именами для временных переменных являются i, j, k, m и n для целых чисел; c, d и e для символов	char *cp; float myWidth;

Константы	Имена переменных, объявленные константами класса, и ANSI-константы должны быть набраны в верхнем регистре (заглавными буквами) с разделяемыми знаком подчеркивания "_" между словами. (ANSI-констант следует избегать для более легкой отладки)	int MIN_WIDTH = 4;
		int MAX_WIDTH = 999;
		int GET_THE_CPU = 1;

Рекомендуется использовать одно объявление на строку, так как это облегчает комментирование.

```
int level;
int size;
```

Примечание: в приведенных выше примерах используется один пробел между типом и идентификатором. Другой приемлемой альтернативой является использование табуляции для выравнивания идентификаторов в один линию (использование клавиши Tab которая равна обычно 4-м пробелам), например:

```
int     level;          // уровень отступа
int     size;           // размер таблицы
Object          currentEntry;   // текущий выделенный экземпляр таблицы
```

2. Объявление классов и интерфейсов

Часть объявления класса/интерфейса	Заметки
Комментарий для документации по классам и интерфейсам (/**...*/)	
Оператор class или interface	

Если необходимо, то указать комментарий реализации класса/интерфейса (/*...*/)	Здесь содержится любая дополнительная информация по классу или интерфейсу, которая не подходит для документирующего комментария.
(static) Переменные класса	Сначала переменные класса public, затем protected, и только потом private
Переменные экземпляра	Сперва public, затем protected – и только потом private
Конструкторы	
Методы	Методы должны быть сгрупированным по функциональности, а не по области или доступности. Для примера метод класса private может находится между двумя public методами экземпляра. Цель – в облегчении чтения и понимания кода

Методы должны быть сгруппированы по функциональности, а не по области или доступности. Для примера метод класса private может находиться между двумя public методами экземпляра.

3. Оформление кода

Размещайте объявления только в начале блоков (блоком является любой код, заключенный в фигурные скобки "{"" и "}"). Не ждите объявления переменных до их первого использования;

```
void MyMethod() {
   int int1;
   if (условие) {
      int int2;
      ...
   }
}
```

Единственным исключением являются индексы циклов for, которые в Java могут быть объявлены в операторе for:

```
for (int i = 0; i < maxLoops; i++) { ...
```

Избегайте локальных объявлений, перекрывающих объявления более высокого уровня. Например, не объявляйте одну и ту же переменную перед блоком кода и во внутреннем блоке.

Старайтесь инициализировать локальные переменные там, где вы их объявляете. Единственная причина не инициализировать переменную в месте её объявления — если её начальное значение зависит от некоторых предварительных вычислений.

При написании Java классов и интерфейсов необходимо соблюдать следующие правила форматирования:

• не ставьте пробел между именем метода и скобкой "(" внутри которой указывается список его параметров;
• открывающая скобка "{" ставится в конце той же строки, где указан оператор объявления класса или интерфейса;
• закрывающая скобка "}" ставится на отдельной строке с тем же отступом, что и у соответствующего ему открывающего оператора, кроме случаев, когда имеем пустой оператор, то "}" должна появиться сразу после "{"

```
class Sample extends Object {
   int ivar1;
   int ivar2;

   Sample(int i, int j) {
      ivar1 = i;
      ivar2 = j;
   }

   int emptyMethod() {}

   ...
}
```

• между методами должна быть одна пустая строка.

В качестве единицы отступов следует использовать 4 пробела. Табуляция должна быть установлена ровно каждые 8 пробелов (а не 4). Избегайте строк длиннее 80 символов.

Если выражение не помещается на одной строке, разбейте его в соответствии со следующими принципами:
• перенос осуществляется после запятой;
• перенос осуществляется перед оператором;
• предпочитайте переносы более высокого уровня для переноса нижнего уровня (вложенном уровне);
• выравнивайте новую строку выражения так, чтобы его начало было на том же уровне, как и в предыдущей строке;
• если приведенные выше правила приводят к запутанному (плохо читающемуся) коду или коду, который сжимается с правым краем, просто введите 8 пробелов вместо этого.

Пример:

```
function(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));

longName1 = longName2 * (longName3 + longName4 - longName5)
           + 4 * longname6;
```

Отступы в строках с оператором if следует применить по правилу 8 пробелов, так как если использовать стандартные 4 пробела, то поиск тела оператора будет затруднителен.

Пример:

```
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Способы форматирования тернарных выражений:

```
alpha = (aLongBooleanExpression) ? beta : gamma;
alpha = (aLongBooleanExpression) ? beta
                                 : gamma;
alpha = (aLongBooleanExpression)
        ? beta
        : gamma;
```

4. Организация файлов

Файл состоит из разделов, которые должны быть разделены пустыми строками и необязательным комментарием, идентифицирующим каждую секцию. Файлы длиной более 2000 строк являются громоздкими и их следует избегать.

5. Файлы с исходным кодом Java

Каждый файл с исходным кодом Java содержит один class с ключевым словом public или интерфейс. Когда классы с ключевым словом private и интерфейсы связанны с public-классом, то их можно поместить в тот же файл с иходным кодом, что и public-класс. Класс с ключевым словом public должен быть размещен как первый класс или интерфейс в файле.

Файлы с исходным кодом Java имеют следующий порядок:

Начальный комментарий

Операторы package и import

Объявления классов и интерфейсов

6. Начальный комментарий

Все файлы с исходным кодом должны начинаться с комментария в стиле языка C, в котором перечислены авторы программы, дата, сведения об авторских правах, а также краткое описание того, что делает программа. Например:

```
/*
 * Имя класса
 *
 * Информация о версии
 *
 * Информация об авторском праве
 */
```

7. Операторы

7.1. Простые операторы

Каждая строка должна содержать не более 1 оператора. Не используйте запятую для группировки нескольких операторов, даже если это видно невооруженным глазом.

7.2. Составные операторы

Составные операторы - это операторы, содержащие списки операторов, заключенные в фигурные скобки "{ операторы }". Вложенные операторы должны иметь отступ на один уровень больше, чем составной оператор. Открывающая скобка должна быть в конце той строки, с которой начинается составной оператор; закрывающая скобка должна начинаться с новой строки и с отступом, соответствующим началу составного оператора.

Скобки используются во всех операторах, даже в одиночных, когда они входят в состав управляющей структуры, таких, как оператор if-else или for. Это необходимо, чтобы избежать ошибок в случае добавления новых операторов, когда забыли указать фигурные скобки (если фигурных скобок нет, то управляющая конструкция типа if будет выполнять только одну строку после нее до знака ";").

7.3. Оператор return

Оператор return, возвращающий значение, не должен использовать скобки, если только их использование не сделает возвращаемое значение более понятным. Например:

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

7.4. Операторы if, if-else, if-else-if-else

Операторы if всегда должны использоваться с фигурными скобками "{ }".

Группа операторов (как и единичное использование) if-else должна иметь следующий вид:

```
if (условие) {
   операторы;
}

if (условие) {
   операторы;
} else {
   операторы;
}

if (условие) {
   операторы;
} else if (условие) {
   операторы;
```

```
} else if (условие) {
   операторы;
}
```

7.5. Оператор цикла for

Оператор цикла for должен иметь следующий вид:

```
for (инициализация; условие; итерация) {
   операторы;
}
```

Пустой оператор цикла for (тот, в котором вся работа выполняется в инициализации, условии и итерации) должен иметь следующий вид:

```
for (инициализация; условие; итерация);
```

При использовании оператора запятой в блоке инициализации или итерации оператора цикла for избегайте использования более чем трех переменных. Если необходимо, используйте отдельные операторы перед циклом for (для случая блока инициализации) или в конце цикла (для случая блока итерации).

7.6. Оператор цикла while

Оператор цикла while должен иметь следующий вид:

```
while (условие) {
   операторы;
}
```

Пустой оператор цикла while должен иметь следующий вид:

```
while (условие)
```

7.7. Оператор цикла do-while

Оператор цикла do-while должен иметь следующий вид:

```
do {
   операторы;
} while (условие)
```

7.8. Оператор switch

Оператор switch должен иметь следующую форму:

```
switch (условие) {
case ABC:
   операторы;
   /* провал */
case DEF:
   операторы;
   break;

case XYZ:
   операторы;
   break;
```

```
default:
   операторы;
   break;
}
```

Каждый раз, когда выбор проваливается (не включая оператор break), добавьте комментарий, где обычно находится оператор break. Это показано в предыдущем примере кода с комментарием /* провал */.

Каждый оператор switch должен включать выбор по умолчанию (default). Оператор break в выборе по умолчанию лишний, но он предотвращает возникновение ошибки, если позже еще один выбор добавится.

7.9. Оператор try-catch

Оператор try-catch должен иметь следующий формат:

```
try {
   операторы;
} catch (ExceptionClass e) {
   операторы;
}
```

8. Отступы

8.1. Пустые строки

Пустые строки улучшают читабельность, выделяя разделы кода, которые логически связаны между собой.

Две пустые строки всегда должны использоваться в следующих случаях:

• между секциями в файле исходного кода;

• между определениями классов и интерфейсов.

Одна пустая строка всегда должна использоваться в следующих случаях:

• между методами;

• между локальными переменными метода и его первым оператором;

• перед блочным или однострочным комментарием;

• между логическими участками кода внутри метода для улучшения читабельности.

8.2. Расстановка пробелов

Разделяющие пробелы должны ставиться при следующих обстоятельствах.

• Ключевое слово, за которым следует скобка, должны быть разделены пробелом. Например:

```
while (true) {
   ...
}
```

Обратите внимание, что пробел не должен использоваться между именем метода и его открывающей скобкой. Это помогает отличать ключевые слова от вызовов метода.

• Разделяющий пробел должен появляться после запятой в списке аргументов.

• Все бинарные операторы кроме "." должны быть отделены от своих операндов пробелами. Пробелом никогда не должны разделятся операнды и их унарные операторы, такие как унарный минус, инкремент («++») и декремент ("--") от их операндов. Например:

```
a += c + d;
a = (a + b) / (c * d);

while (d++ = s++) {
   n++;
}
prints("size is " + foo + "\n").
```

• Выражения в операторе цикла for должны быть разделены пробелами. Например:

```
for (expr1; expr2; expr3)
```

• За приведением типа должен следовать пробел. Например:

```
myMethod((byte) aNum, (Object) x);
myFunc((int) (cp + 5), ((int) (i + 3))
                       + 1);
```

Язык разметки документов HTML

1.1. Тип документа

Всегда объявляйте тип документа в качестве первой строки в документе:

```
<!DOCTYPE html>
```

Если требуется согласованность с тегами нижнего регистра, можно использовать:

```
<!doctype html>
```

Рекомендуется использовать имена элементов нижнего регистра.

```
<section>
  <p>This is a paragraph.</p>
</section>
```

2.1 Закрытие всех элементов HTML

Рекомендуется закрыть все элементы HTML.

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```

3.1. Закрытие пустых элементов HTML

В HTML5 необязательно закрывать пустые элементы.

Разрешены:

```
<meta charset="utf-8">
```

Также разрешено:

```
<meta charset="utf-8" />
```

Однако в XHTML и XML требуется закрывающая косая черта (/).

4.1. Использовать имена атрибутов нижнего регистра

Рекомендуется использовать имена атрибутов нижнего регистра, а также заключать значения атрибутов в кавычки.

<table class="striped">

5.1. Атрибуты изображения

Всегда добавляйте атрибут alt к изображениям. Этот атрибут важен, когда изображение по какой-либо причине не может быть отображено. Кроме того, всегда определяйте ширину и высоту изображения. Это уменьшает мерцание, потому что браузер может зарезервировать место для изображения перед загрузкой.

<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">

6.1. Пробелы и знаки равенства

Не ставьте пробелы вокруг знака равенства. <link rel="stylesheet" href="styles.css">

Старайтесь избегать строк кода длиной более 80 символов.

7.1. Пустые строки и отступы

Не добавляйте пустые строки без причины.

Для удобочитаемости добавьте пустые строки для разделения больших или логических блоков кода.

Для удобочитаемости добавьте 2 пробела отступа. Не используйте клавишу TAB.

Не используйте ненужные пустые строки и отступы. Нет необходимости отступить каждый элемент:

```
<body>

<h1>Famous Cities</h1>

<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.
It is the seat of the Japanese government and the Imperial Palace,
and the home of the Japanese Imperial Family.</p>

</body>
```

Пример таблицы:

```
<table>
  <tr>
    <th>Name</th>
```

```
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```

Пример списка:

```
<ol>
  <li>London</li>
  <li>Paris</li>
  <li>Tokyo</li>
</ol>
```

8.1. Пропуск <html>, <body>, <head>

Не рекомендуется опускать тег <head>.

Не рекомендуется опускать <html>, <body>, <head>.

Элемент <html> является корнем документа. Это рекомендуемое место для указания языка страницы:

```
<!DOCTYPE html>
<html lang="en-US">
```

Объявление языка важно для приложений специальных возможностей (программы чтения с экрана) и поисковых систем.

Пропуск <html> или <body> может аварийно завершить работу DOM и XML ПО.

Пропуск <body> может привести к ошибкам в старых браузерах (IE9).

9.1. Метаданные

Элемент <title> необходим в HTML5. Сделайте название как можно более осмысленным:

```
<title>HTML5 Syntax and Coding Style</title>
```

Чтобы обеспечить правильную интерпретацию и правильную индексацию поисковых систем, как язык, так и кодировка символов должны быть определены как можно раньше в документе:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Syntax and Coding Style</title>
</head>
```

10.1. Комментарии в формате HTML

Краткие комментарии должны быть написаны на одной строке, как это:

```
<!-- This is a comment -->
```

Комментарии, охватывающие более одной строки, должны быть написаны так:

```
<!--
  This is a long comment example. This is a long comment example.
  This is a long comment example. This is a long comment example.
-->
```

Длинные комментарии легче заметить, если они имеют отступ двух пробелов.

11.1. Таблицы стилей

Используйте простой синтаксис для связывания с таблицами стилей (атрибут Type не требуется):

```
<link rel="stylesheet" href="styles.css">
```

Короткие правила могут быть записаны сжатыми на одной строке:

```
p.intro {font-family: Verdana; font-size: 16em;}
```

Длинные правила должны быть написаны на несколько строк:

```
body {
  background-color: lightgrey;
  font-family: "Arial Black", Helvetica, sans-serif;
  font-size: 16em;
  color: black;
}
```

• поместите открывающую скобку на ту же линию, что и селектор;
• используйте один пробел перед открывающей скобкой;
• использовать 2 пробела отступа;
• используйте точку с запятой после каждой пары значение свойства, включая последнюю;
• используйте кавычки только вокруг значений, если значение содержит пробелы;
• поместите закрывающую скобку на новую линию без пробелов;
• избегайте линий более 80 символов.

12.1. Загрузка JavaScript в HTML

Используйте простой синтаксис для загрузки внешних скриптов (атрибут Type не нужен):

```
<script src="myscript.js">
```

Приложение № 4

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЕДИНАЯ ЦИФРОВАЯ ПЛАТФОРМА РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ» ДЛЯ СОЗДАНИЯ, РАЗВИТИЯ И ЭКСПЛУАТАЦИИ ГОСУДАРСТВЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПО С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТОВ ЕДИНОЙ ЦИФРОВОЙ ПЛАТФОРМЫ РОССИЙСКОЙ ФЕДЕРАЦИИ «ГОСТЕХ»

Рекомендуемый список элементов конфигурации разрабатываемого ПО

2022 г.

1. Программа (дистрибутив программы) и эталонные конфигурации.
2. Рекомендуемые схемы размещения элементов ПО для сред его функционирования (диаграммы развёртывания).
3. Инструкция по установке ПО и его компонентов.
4. Руководство по обеспечению безопасности.
5. Программные и эксплуатационные документы.
6. Исходный код программы.
7. Среда функционирования и развертывания для разрабатываемого ПО.
8. Любая информация об изменениях элементов конфигурации, включая обновления ПО и устранения уязвимостей.
9. Информация, связанная с обновлениями ПО и устранениями уязвимостей программы.
10. Перечень выявленных недостатков программы и информация об их состоянии в базе данных уязвимостей Разработчика ПО.
11. Безопасные, тестовые, технологические конфигурации ПО (например, файлы cfg, ini или описание этих настроек для каждого сценария).
12. Документация на систему управления конфигурацией и ее элементы.
13. Документацию на систему управления изменениями; конфигурации контуров Разработчика ПО, тестового и промышленной эксплуатации (оборудование, ОС и др.).
14. Средства разработки и их опции, применяемых при разработке конкретного экземпляра ПО.

Дополнительно в список элементов конфигурации рекомендуется включать следующее:

документ, содержащий требования по безопасности, предъявляемые к разрабатываемому ПО;

документ, содержащий сведения о результатах моделирования угроз безопасности информации;

документ, содержащий сведения о проекте архитектуры программы;

документ, описывающий используемые инструментальные средства;

документ, содержащий информацию о прослеживаемости исходного кода программы к проекту архитектуры программы;

документ, содержащий порядок оформления исходного кода программы;

документ, содержащий сведения о результатах проведения статического анализа исходного кода программы;

документ, содержащий сведения о результатах проведения экспертизы исходного кода программы;

документ, содержащий сведения о результатах проведения функционального тестирования программы;

документ, содержащий сведения о результатах проведения тестирования на проникновение;

документ, содержащий сведения о результатах проведения динамического анализа кода программы;

документ, содержащий сведения о результатах проведения фаззинг-тестирования программы;

документ, содержащий описание процедуры передачи ПО Заказчику;

документ, содержащий описание процедур отслеживания и исправления обнаруженных ошибок в программе и уязвимостей;

документ, содержащий описание процедуры поиска Разработчиком ПО уязвимостей программы;

документ, описывающий реализацию и использование процедуры уникальной маркировки каждой версии ПО;

документ, описывающий использование системы управления конфигурацией ПО;

документ, описывающий меры, используемые для защиты инфраструктуры среды разработки ПО;

документ, содержащий сведения об обучении работников.